Semantic Multi-Criteria Decision Making in Autonomous Embedded Systems

Von der Fakultät für Elektrotechnik und Informatik

der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des Grades eines

DOKTORS DER INGENIEURWISSENSCHAFTEN

Dr.-Ing.

genehmigte Dissertation
von

M. Sc. Ghadi Mahmoudi

geboren am 10 Oktober 1975, in Latakia, Syrien

2009

Referent:	**Prof. Dr.-Ing. C. Müller-Schloer**
Koreferent:	**Prof. Dr. Nicola Henze**
Tag der Promotion:	**26. Juni 2009**

Summary

The increasing complexity of modern computer systems emerges as a real challenge for both the users and the designers. According to the innovative vision of *Organic Computing* systems, the system components will profit from their autonomy to perform a self-organization, and subsequently, to reach a high degree of adaptivity.

The goal of this thesis is to investigate the vision of Organic Computing and its realization in embedded computer systems. An interdisciplinary methodology called "*Semantic Multi-Criteria Decision Making*" (SeMCDM), has been suggested and evaluated regarding different aspects. The new methodology adopts a marketplace-oriented behavior pattern for the autonomous system components.

The shift of the design decisions from the human designer to the autonomous system components implies a balancing knowledge transfer in the same direction. From this central idea two issues emerge: The first issue addresses the form of the knowledge to be transferred, while the second issue deals with the appropriate mechanisms for autonomous decision making.

Similar to the Semantic Web, the necessity for a common, machine processable, understanding of the world has been recognized. An ontology-based description of the world offers an optimal solution to represent the knowledge of several human designers, to transfer the knowledge to the autonomous system components, and to build a communication language between the autonomous system components. While processing the ontological knowledge, the autonomous units share also the same semantic with the human designers and users.

This thesis defines the autonomous decision making as a selection problem, which is strongly affected by the existence of multiple, conflicting, criteria. Methods for making decision under multiple criteria, which are known from the research field of "Operations Research", have been categorized and analyzed to proof their adequacy for the purposes of autonomous system components. In the light of the adopted MCDM-Methods, development tools have been designed and actually embedded within an ontology development environment. A specially developed MCDM-ontology builds a bridge between two worlds: Ontologies and MCDM. A set of inference rules enhances a conventional inference engine to be a MCDM-capable inference engine.

Through the merge of both technologies, this thesis presents not only a new concept of Semantic Multi-Criteria Decision Making SeMCDM, but also a complete platform for the design and deployment of autonomous system components in self-organizing systems.

The investigation of current automotive communication systems, together with the creation of a catalog of requirements, targets at the identification of possible applications for SeMCDM-based self-organizing embedded systems. Applications of MOST (Media Oriented Systems Transport) bus system have been selected as example applications of SeMCDM.

The marketplace-oriented behavior pattern of the autonomous system components is also a theme of an intensive investigation in this thesis. The suggested market scenarios specify the details of the generally known marketplace-oriented behavior pattern, and reveal several possibilities of role assignments. Furthermore, the performance of the market scenarios has been investigated by means of simulation, taking the nature of the application environment into account. The results have been summarized as design recommendations.

Keywords: Organic Computing, Multi-Criteria Decision Making, Marketplace

Zusammenfassung

Die steigende Komplexität moderner Computersysteme entwickelt sich zu einer ernsthaften Herausforderung für die Benutzer und für die Entwickler. Nach der innovativen Vision der *Organischen Computer Systeme*, werden die Systemkomponenten von ihrer Autonomie profitieren, um eine selbst-organisation durchzuführen und damit eine hohe Adaptivitätsfähigkeit zu erreichen.

Das Ziel dieser Arbeit liegt in der Erforschung der Vision der organischen Computersysteme und ihrer Realisierung in eingebetteten Computerarchitekturen. Eine interdisziplinäre Methodologie, genannt „*Semantic Multi-Criteria Decision Making*“ (SeMCDM), wurde hier vorgeschlagen und unter mehreren Aspekten evaluiert. Die neue Methodologie übernimmt ein Marktplatz-orientiertes Verhaltensmuster für die autonomen Systemkomponenten.

Die Übertragung der Designentscheidung vom menschlichen Entwickler auf die autonomen Systemkomponenten setzt eine balancierende Wissensübertragung in der gleichen Richtung voraus. Aus diesem zentralen Gedanken sind zwei Zweigfragen entstanden: Die erste Frage betrifft die Form des zu übertragenden Wissens, während sich die zweite Frage mit den geeigneten Mechanismen zum autonomen Fällen von Designentscheidungen beschäftigt.

In einer Analogie zum Semantik Web, wurde für die autonomen Systemkomponenten der Bedarf nach einem einheitlichen, maschinell-verarbeitbaren, Verständnis der Welt erkannt. Eine Ontologie-basierte Weltbeschreibung bietet eine optimale Lösung zur Darstellung des Wissens mehrer menschlichen Entwicklern, zur Wissensübertragung in die autonomen Systemkomponenten und zur Kommunikation zwischen den autonomen Systemkomponenten. Auch bei der maschinellen Verarbeitung des ontologischen Wissens teilen sich alle autonomen Systemkomponenten die gleiche Semantik mit den menschlichen Entwicklern und Benutzern.

Diese Thesis definiert die autonome Designentscheidung als ein Selektionsproblem, das von der Existenz mehrerer, widersprüchlichen, Auswahlkriterien geprägt ist. Aus dem Forschungsgebiet „Operations Research“ wurden bekannte Methoden zur Entscheidung in Betrachtung mehrerer Kriterien (Multi-Criteria Decision Making, MCDM) kategorisiert und auf ihrer Eignung für Zwecke der autonomen Systemkomponenten geprüft. Zu den geeigneten MCDM-Methoden wurden Entwicklungswerkzeuge konzipiert und sogar in einer Ontologieentwicklungsumgebung eingebettet. Eine speziell entwickelte MCDM-Ontologie bildet eine Brücke zwischen den zwei Welten: Ontologien und MCDM. Ein Satz von Schlussfolgerungsregeln erweitert eine herkömmliche Inferenzmaschine zu einer MCDM-fähigen Inferenzmaschine.

Mit der Vereinigung dieser Technologien stellt diese Thesis nicht nur das neue Konzept zum Semantic Multi-Criteria Decision Making SeMCDM vor, sondern auch eine komplette Plattform zur Entwicklung und Verwendung autonomer Systemkomponenten in selbst-organisierenden Systemen.

Eine Untersuchung der heutigen Automobilkommunikationssysteme und die Erstellung eines Anforderungskatalogs zielen auf die Erkennung möglicher Applikationsgebieten für SeMCDM-basierte selbst-organisierende eingebettete Systeme. Anwendungen um das MOST (Media Oriented Systems Transport) Bussystem wurden als Beispielapplikationen of SeMCDM ausgewählt.

Das Marktplatz-orientiertes Verhaltensmuster der autonomen Systemkomponenten ist auch ein Thema einer intensiven Untersuchung in dieser Thesis. Die hier vorgeschlagenen Marktplatzszenarien konkretisieren das allgemein bekannte Verhaltenmuster und decken mehrere mögliche Rollenverteilungen auf. Ferner, die Performanz der Marktplatzszenarien wurde im Zusammenhang mit den Eigenschaften der Applikationsumgebung an Hand einer Simulation untersucht. Die Ergebnisse wurden zu Designempfehlungen zusammengefasst.

Schlagwörter: Organic Computing, Multi-Criteria Decision Making, Marktplatz

Contents

List of Figures

List of Tables

1. Introduction

The rapid development of electronics and software engineering opens the way for new applications and capabilities. Computers are not only our work platform. We meet them everywhere in our environment; more in their embedded form than in their traditional "personal" form.

Beside this positive trend, the complexity of the systems increases. Portable telephones are equipped now with technologies, which were only some years ago, a luxury on a personal computer. The interconnection of distributed systems, which have to work together to fulfil our expectations, represents a real challenge for engineers. Such difficulties are not any more scientific fiction. The automotive systems show the limits of the traditional way towards a "well functioning design". Main manufacturers and huge numbers of suppliers are no more able to cope with the integration problem of their "intelligent devices". Unexpected behavior and failure are the results of such situation. Additionally and surely more dangerous problems arise at runtime. Technicians in the repair and assembly shop have usually restricted possibilities to identify the source of failure. The possibilities to correct it are consequently not better. The integration of new devices or the updating of available software to new versions is not imaginable without side effects.

The manufacturers of automotive systems expressed in the last years their interest on new concepts and solutions. Especially the shift of the solution from the design time to the run-time has become a conceivable idea. The kernel of such approaches is the autonomy of system components. Through their cooperative behavior, the components are expected to find a solution in dangerous or sub-optimal situations. The resulting adaptive system is described as self-organizing.

Autonomy and the self-organization are basic concepts in the research field of "Organic Computing" (OC). The organic dimension is inherited from fascinating creatures, which enjoy high performance and find innovative solutions through the cooperation of a high number of, in reality weak, individuals. The ants' colonies, for example, have inspired scientists with new and practical ideas. Alone the idea that "the overall performance seems to be more than the sum of the individuals' efforts" stands behind many research projects of possibly high importance for future engineering solutions.

However, the autonomy of system components takes the design authority away from the hands of experienced engineers. Therefore, self-organization has been often met with sceptic questions about its feasibility and control.

1.1. Goals and criteria

This thesis investigates the possibilities of integrating self-organizing behavior in real applications. It shows that the autonomy of system components can play a positive role in building adaptive systems.

The main point here is an abstraction of the self-organization from a matter of control to a matter of knowledge. An autonomous system component should have a clear imagination about itself, about the other components, and about the expected functionality from the whole system. This knowledge has been always monopolized by the human designer. The time is coming to transfer the knowledge and to place it to the disposal of autonomous system components. Only the shift of knowledge will balance the shift of authority.

The first challenge towards the realization of this vision is to find an adequate language to express the engineering knowledge, which is a result of long experience, made by an

unknown number of people. This language should not be readable only by engineers, but also digitally processable by system components.

The second challenge is to support the system components with mechanisms to make use of the available knowledge. The system components should be capable of translating knowledge to decisions and actions.

The third challenge is to deliver design tools to support the knowledge transfer process. Such tools have to be able to ensure the consistency of the given knowledge. An especially important issue here is to find tools which support collaborative development of knowledge. Only the collaboration between engineers on the level of knowledge development will lead to collaborative self-organizing system components.

A suitable application field is vital for the deployment of the self-organizing concept in real systems. The automotive industry offers a wide range of complicated systems, with a real need for know-what and know-how knowledge. On the other hand, critical applications are not adequate as a test field of novel concepts. A reasonable balance should be always kept in mind.

The estimation of the success, the performance and resource demand would then give a clear image about the self-organizing concept.

1.2. Overview about the thesis

This thesis builds on interdisciplinary research. The roots of the main idea go back to Organic Computing, to approaches of organic adaptive systems in automotive industry, to the Semantic Web and to methods from operations research.

For the first time an approach has been established, which allows for a useful knowledge transfer from the human designer to self-organizing systems. The autonomous ontology-based decision making with the existence of multiple criteria is the main contribution of this thesis. This novel approach has the name of *Semantic Multi-Criteria Decision Making* (SeMCDM). The suggested concept is specified to a concrete architecture and implemented as a prototype to proof its functionality and to evaluate its performance in different technical aspects.

This thesis is organised as follows:

A state of the art survey is conducted in chapter 2. The organic computing is presented in section 2.1, while a special section 2.2 addresses the developments in the field of automotive systems with special concentration on approaches trying to involve principles of Organic Computing in the automotive industry.

Approaches around the Contract Net Protocol are summarized in section 2.3.

The Semantic Web is the theme of section 2.4.

Methods for Multi-Criteria Decision Making have been systematically categorized and explained in section 2.5.

First approaches proposing the integration of ontological knowledge and MCDM are presented in section 2.6.

Section 2.7 provides an overview about automotive communication platforms.

Chapter 3 discovers the shortcomings of existing approaches considering the adoption of Organic Computing principles in automotives and suggests the novel concept of Semantic Multi-Criteria Decision Making (SeMCDM). The expected advantages of SeMCDM have been then explained, before discovering a set of challenging design questions.

Chapter 4 considers these design questions and finds adequate solutions. In section 4.1, the suitable MCDM methods have been recognized by applying a requirement catalog on the methods surveyed in section 2.5.

Section 4.2 presents the ontologies of SeMCDM and illustrates their interrelation.

Section 4.3 deals with the effects of integrating the ontology-based semantic matching with MCDM methods and presents special matching rules to build the capability of "semantic matching for MCDM". The generalized matching process, presented in section 4.4, applies this capability to reach a complete specification of SeMCDM.

In section 4.5, the specified concept of SeMCDM finds its application on the base of a marketplace-oriented behavior of autonomous system components. The integration of both ideas shows different combinations and roles distribution possibilities on the marketplace. A set of market scenarios emerges in this section.

Having a clear specification of SeMCDM, suitable application areas in the automotive industry have been found in section 4.6.

Design tools of SeMCDM-capable autonomous system components have been specified and implemented, as shown in section 4.7.

The suggested concepts and architecture of SeMCDM have been evaluated in chapter 5. The implemented prototype validates the SeMCDM concept and its basic functionality, as shown in section 5.3. An estimation about the resource demands of SeMCDM is presented in section 5.4. The fitness of the market scenarios in different embedded application environments has been studied on the base of a special simulation model, presented in section 5.5.

The thesis concludes in chapter 6 with a summary and ideas for further research and future developments.

2. State of the art

Organic Computing represents the theoretical base of this thesis. Section 2.1 introduces the concepts of Organic Computing and its main notions like self-organization, autonomy and emergence.

Section 2.2 treats approaches from the world of automotive industry, where self-organization plays a central role. Additionally, approaches of marketplace-oriented behavior around the Contract Net Protocol are the theme of section 2.2. (On the base of information presented in sections 2.2 and 2.2, a comparison between these approaches and their shortcomings will be discussed in section 3.1).

This section gives also information about the enabling technologies and methodologies, which (may) play a role in the design of Semantic Multi-Criteria Decision Making SeMCDM. To this reason the Semantic Web is presented in section 2.4. The Semantic Web has been proven as a future technology, which faces similar questions to those known from approaches of self-organizing (automotive) systems.

Section 2.5 introduces problems of decision making under multiple criteria. It contains a survey of MCDM methods. The survey has the aim of categorization of these methods, in order to make a comparison between them and to select suitable candidates for usage in SeMCDM.

The integration of ontologies within a decision making process is the theme of section 2.6. Different integration attempts throw light on real need and benefits of such intention.

Section 2.7 deals with the communication platforms of automotive systems. On the base of the presented information important conclusions about adequate applications of SeMCDM in the automotive field will be drawn in section 4.6.

2.1. Organic Computing

The Organic Computing Initiative (OCI) promises to face the increasing complexity of engineering systems with the usage of principles observed in natural systems [MS04]. A leading example of such principles is the ability of ants to find the shortest path by indirect communication via pheromones. Research approaches about “swarm intelligence” [BW89] try to find, for example, the shortest path to information sources.

Application fields of Organic Computing are always supposed to be found where “the design complexity of the system is no more manageable, and the global behavior might be unexpected due to effects of interaction”. Such extreme situations can be more and more observed because of the “computerisation of our environment” [MS04].

From the point of view of the Organic Computing Initiative, more autonomy of our technical systems should optimize their orientation towards our needs and construct them as robust, safe, flexible and trustworthy as possible. The autonomy of technical systems leads to self-organized systems, which can adapt dynamically to the current conditions of the environment. Along with self-organization, a set of self-x attributes is expected to emboss the Organic Computing systems, like the self-configuration, self-healing, self-explanation, self-protection.

Similar attributes have been announced as the goal of the Autonomic Computing (AC) vision [KC03]. However, the Organic Computing initiative addresses more general (application independent) questions and focuses on large collections of intelligent devices providing services to humans [HS05], whereas the Autonomic Computing concentrates on a specific problem concerning the automated administration of networks and computer centres.

Self-organization is based on the idea that local behavior of components decides about the global behavior of the systems. One of the most important issues addressed by the Organic Computing initiative is related to potential differences between local and global behavior. Such differences arise especially if the system consists of large numbers of components. This phenomenon, known as "emergence", is defined as "an attribute of a total system which cannot be derived from the simple summation of properties of its constituent subsystems" [MS04]. Moreover, the definitions of emergence imply also a kind of surprising happenings, which are not pre-programmed explicitly into the subsystems. A resonant circuit may be the strongest example of the emergent behavior, where the resonance frequency constitutes a property of the total system which is not existent in the components.

The Organic Computing initiative shows - on the one hand - high confidence about the future of its vision, on the other hand it is conscious of the difficulties before the vision can find a real implementation: "It is not the question whether self-organized and adaptive systems will arise but how they will be designed" [HS05].

Especially the emergence phenomenon has been limited to its safe version called "controlled emergence", a mixture of hierarchical control (top-down design paradigm) self-organization (bottom-up design paradigm) [MS04]. A generic architectural concept for the design and analysis of Organic Computing systems has been developed: The Observer/Controller architecture [RMBMSS06]. The suggestion of such modelling architecture complies with the trend of combining concepts known from control theory in the design paradigms of computing systems [JHYPT05]. In a later work [SMS08], the degree of self-organization has been defined as a relation between the number of control mechanisms and the number of (autonomous) components. A system enjoys maximum "variability" if every component has a specific control mechanism.

The self-organization implies possibly unexpected behavior of the system. To generate more trust into the dependability of a system a new paradigm of Organic Computing has been suggested, where "the objective for system design should be to design controllable self-organizing systems, i.e. systems which allow for external control but have a high degree of autonomy as well" [SMS08]. The degree of autonomy of a system has been defined as the "complexity reduction", measured as the difference between the internal control actions (the overall number of system's attributes) and the external control actions (attributes to be set up by the human user). Strongly self-organized systems have high values of variability (highly distributed) and high degree of autonomy.

Emergence has been defined as "self-organized order" [MMS06]. Consequently, the entropy, known as measure of order, has been suggested as a way to quantify the emergence of organic systems.

A practical organic system can only be built on the base of distributed and reconfigurable systems. This means that a lot of small and intelligent components have to be available for the designer. Moreover, the designer would face a challenge when trying to determine local rules in the components in such a way, so that the expected global behavior of the system can be achieved.

2.2. Organic behavior in automotive systems

This section presents briefly the current situation and the trends in the field of automotive systems and their design methods. Ideas considering concepts of self-configuration and self-organization in automotive systems are depicted in the following subsections considering two major projects: DySCAS [ARCJBE07] and EvoArch [HL05].

The information provided in this section builds the base for a critique of current approaches in section 3.1, which proves the need for the novel concept of SeMCDM.

2.2.1 Automotive systems

The automotive systems underwent an enormous development in the last decades. Engine and transmission control, chassis control (brakes), body electronics (door control, climate control) and safety devices (air bags, pre-tensioning of seat-belts) are standard functionalities in modern vehicles. Infotainment and telematic devices, along with the personal portable devices of the driver extend the platform to new applications and possibilities.

Applications of the automotive systems took steps forward, while their design concepts could not make a similar development. Developing new functionalities in software has aggravated the situation more and more. Standardization efforts were always necessary on the network level. So CAN, LIN, FlexRay and MOST are common terms in the world of automotive industry (for more details see section 2.7). On the other hand, the design of hardware and software was an issue of individual OEMs (Original Equipment Manufacturer) and vendors. To develop portable and reusable application software, the OSEK consortium [OVP03] proposed more standardization on the level of operating systems and communication since 1993. OSEK stands for "Offene Systeme und deren Schnittstellen für die Elektronik in Kraftfahrzeugen" ("Open Systems and their Interfaces for the Electronics in Motor Vehicles"). The main contribution of the OSEK consortium is the specification of a real time operating system for automotive embedded systems. Additionally, the OSEK consortium defined the OSEK Implementation Language (known as OSEK-OIL), which enables a configuration of the operating system's resources in accordance with the application's needs. A "system generator" selects the required system resources (resource files), which can be then compiled together with the application source files to generate an executable [OV04a]. Data transfer between tasks and/or operating system services has been standardized under the OSEK-COM specification [OV04b]. The standard applies to internal communication within one electronic control unit (ECU) and for external communication (between different ECUs). To enhance the development process, OSEK run time interface (OSEK-RTI) enables communication between a debug tool and the embedded operation system. This way, valuable internal operating system data and information about task states can be made visible to the developer [OV05]. OSEK Network Management (OSEK-NM) specification provides standardized features, which ensure the functionality of inter-networking by standardized interfaces [OV04c]. The network management ensures the safety and the reliability of the communication network through network monitoring.

In the recent past, AUTOSAR (AUTomotive Open System ARchitecture) has been introduced as a further development of OSEK. AUTOSAR is "an open and standardized automotive software architecture, jointly developed by automobile manufacturers, suppliers and tool developers" [Aut03]. AUTOSAR defines a modular software architecture for automotive electronic control units (ECUs) (see Figure 2-1). AUTOSAR Runtime Environment (RTE) represents a special layer, separating the Basic SoftWare functionalities (BSW) from the AUTOSAR application software. AUTOSAR defines also the interfaces between the application software and the RTE, as well as between the RTE and the BSW. This way, the application software has been isolated from the hardware of the ECU and from the operating system (this architecture may remind software developers of the Java Virtual Machine and the Java technology [LY99]).

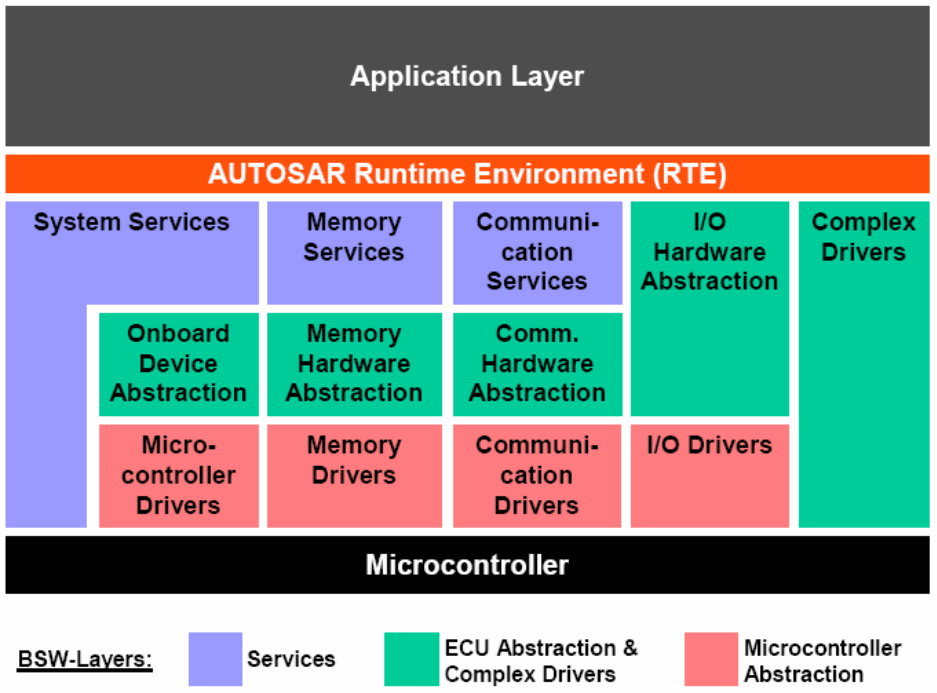


Figure 2-1 AUTOSAR ECU layered architecture (from [F+06])

In addition to this runtime environment, AUTOSAR presents a new development methodology [Aut08], see Figure 2-2. The AUTOSAR methodology covers steps of system development from the application source code to the executables on the ECUs.

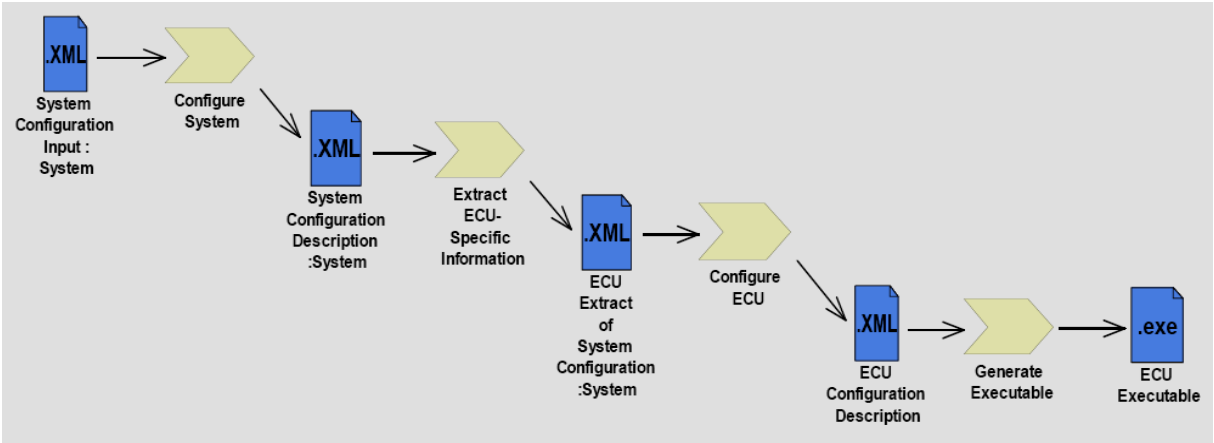


Figure 2-2 AUTOSAR development methodology (from [Aut08])

The start point of the system development is the "system configuration input", which follows in its structure predefined templates. The information included in these templates addresses three main packages: The software components (software API, data types, ports interfaces etc.), the ECU resources (processor unit, memory, peripherals, sensor actuators etc.) and system constraints (like bus topology, mapping of software components belonging together). The mapping of software components to the ECUs takes place with regard to resources and timing requirements. This process is called "System Configuration". From the resulting "System configuration Description", ECU-specific information is then extracted. The configuration process of each ECU considers more information, like task scheduling and necessary BSW modules (and their configuration). On the base of the resulting "ECU Configuration Description", executables will be then generated. This process involves code

generation (for the RTE and BSW), code compiling (of the generated code and of the software components available as source code) and finally linking everything together into an executable.

The system configuration in AUTOSAR is still a design issue, where software components are statically mapped to ECUs at design time.

2.2.2 DySCAS

DySCAS (Dynamically Self-Configuring Automotive Systems) is an European Commission funded project that started in June 2006 [ARCJBE07]. DySCAS declares the goal of building a vehicular control system architecture that supports self-configuration. Benefits of such an architecture meet needs of the designers and users of future automobiles. The dynamic upgrade of automotive software counts as an important advantage. If a serious fault occurs, dynamic upgrade in the field is the only way to avoid expensive recall. Moreover, owners have increasingly high expectations for infotainment services on the move and want to get benefit of their handheld devices, the embedded devices in the vehicle and the wireless connection to the external world. This software-centric and network supported environment recalls the computer networks and the idea of Grid Computing [GC]. So the authors of [ARCJBE07] speak of "automotive control grid".

DySCAS promises automatic discovery and incorporation of new devices, as well as self-optimisation to best use in terms of available resources (processing, storage, and communication resources), in addition to self-diagnostics.

According to [ARCJBE07], DySCAS will focus mainly on the reconfiguration or self-reconfiguration of software tasks in a standardized middleware. The underlying middleware is AUTOSAR.

The theoretical roots of DySCAS are derived from the vision of Autonomic Computing [KC03]. Autonomic Computing shares similar concepts with Organic Computing (presented in section 2.1), like the need for more autonomy and some of the expected self-x properties. The vision of Autonomic Computing has been developed originally by IBM, which dates its first appearance to March 2001, as IBM's senior vice president of research, Paul Horn, introduced this idea to the National Academy of Engineers at Harvard University.

A control theoretical approach serves as a think model for DySCAS. This model comprises four steps: Sensing, control, feedback and adaptive control (learning during design time and run-time).

The practical implementation of DySCAS was still an open question until 2007 where a study of existing technologies has been finished [Dys07] and a first suggestion of the DySCAS middleware has been proposed [AE07]. The middleware is supposed to combine two mechanisms: The policy-based computing and the utility functions.

The policy-based computing is "a technique where the business function (or intent) is expressed as a set of rules or configuration statements (a policy) and kept detached from the implementation mechanism" [Dys07]. The policy determines the behavior of the system using a high-level language (i.e. a standardized and platform-independent language) [AE07]. The policy itself is stored separately from the deploying mechanism and is loaded at initialisation or even during run-time.

Policies rely on a set of "context parameters" and their values. A context parameter can be, for example, the encoding type of a streaming device in the automobile. These context parameters are practically attributes of resources. Rules trigger actions on the base of values of such context parameters.

A study in [Dys07] distinguishes between static policies, open-loop policy adaptation (manual adaptation) and closed loop policy adaptation (automatic adaptation). The adaptation can be done within a policy by changing its rules; or using a meta-policy which selects and activates one of different available policies. This flexibility is one of the expected advantages of policy-based computing.

The utility functions (UF) "provide a means of choosing from several options, based on the instantaneous merit of each option" [Dys07]. The meant utility functions are exactly those presented in section 2.5.3.3 of this thesis, where multi-criteria decision making methods are surveyed. For purposes of DySCAS, the utility functions help to make decisions under the influence of several contextual factors. Utility functions require very low levels of processing and storage resources and are therefore suitable for use in embedded systems [Dys07]. They are preferred by the authors of [Dys07] because decisions can be made on the base of local knowledge, and so no more negotiations are needed. The disadvantage of being too static (the weights of utility functions are usually pre-defined at design time) is faced with ideas of dynamic weighting, which could be achieved by a combination of utility functions and policy-based computing. A policy may provide a higher-level configuration of a utility function by setting its weights dynamically. A policy can also dynamically select which of several differently configured utility functions should be used at any specific time.

The policy implementation is based upon the AGILE policy expression language and integration framework [ANT07]. The authors of [AE07] mention different advantages of AGILE like the support for hierarchical policies (enabling dynamic adaptation through policy selection), dynamic policy adaptation through rules reconfiguration, and the integration of utility functions into policy logic. AGILE has been developed in C++/.NET. The authors refer that the current version of AGILE is therefore not optimized for resource-constrained environments such as that represented by the DySCAS middleware. A lightweight version, AGILE-Lite is intended to be developed in C. However, no information is available now about the expected resource requirements or performance of AGILE-Lite. A prototype application of AGILE is available to be downloaded from [ANT07], in addition to some policies and examples.

No policy design tool is yet available (a tool with unspecified features is announced but it is yet "buggy" [ANT07]). A similar problem affects the usage of utility functions. DySCAS and AGILE don't try to suggest a way to set the weights of utility functions.

2.2.3 EvoArch

The efforts towards the Evolutionary Architecture (EvoArch) are motivated by the complexity of modern automobile systems. The issue is not only related to the number of embedded electronic systems in automobiles, and to their increasing complexity, but also to the integration process of these systems. The main manufacturer of automobiles is forced to gather experiences of other parties to implement his own design. As soon as the supporting firms deliver the specified sub-systems separately, problems of integration may arise and the functionality of the whole system (a "car") cannot be guaranteed.

In addition to problems of the design phase, the manufacturer of the automobile cannot react to malfunction of cars, or parts of, in run-time. This shows the need for a kind of self-healing in modern cars.

To face such situations EvoArch [HL05] aims at giving technical systems the capability of self-organization. For this purpose EvoArch proposes autonomous units gathered around an information space (called arena) to exchange offers and enquiries as in a marketplace (Figure 2-3).

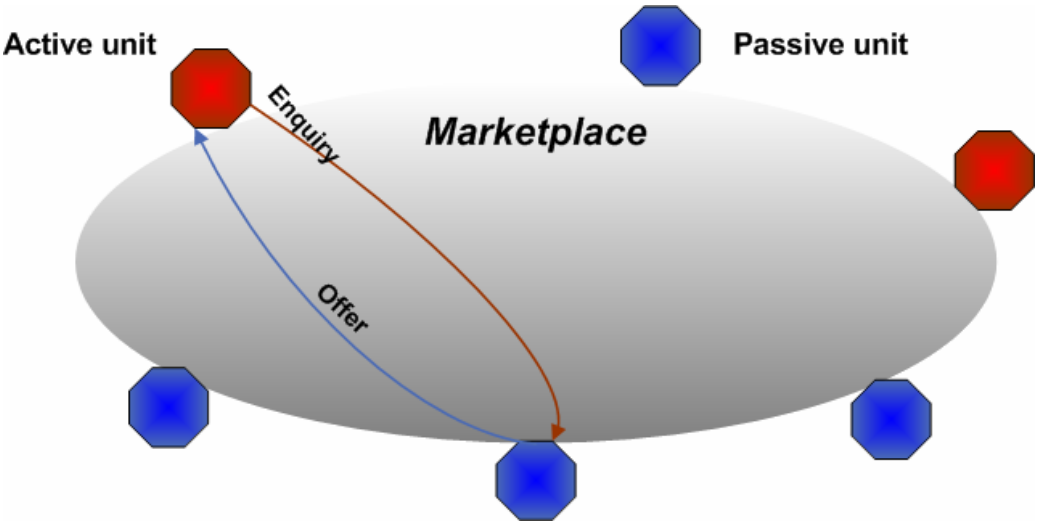


Figure 2-3 EvoArch suggests a marketplace-oriented behavior, where autonomic units exchange enquiries and offers.

The main ideas behind EvoArch can be described as follows [H+02]: An automobile consists of intelligent autonomous units. They have properties and aims. Autonomous units can be active or passive. Active units look for partners to enhance their own functionality, while passive units offer their own capability purposing to be accepted as a partner. If there are many available offers for one enquiry, one offer has to be selected. After the selection of a partner a contract has to be concluded.

The authors recognized challenges facing an autonomous unit on the arena: The search for a partner, the limitation of the candidate list, the selection of a suitable partner from a candidate list, the conclusion of a contract, and the legitimation of contracts.

The search for partners is supposed to be achieved through the exchange of offers and enquiries. The behavior of autonomous units on the arena is determined by the vision that active units look for passive units with special capabilities (by sending enquiries) and that passive units send a positive answer (offer) if they can meet the required capabilities.

The limitation of the candidate list arises as a requirement because of the possibly high number of autonomous units on the arena. The principle of Taxonomy selection (T-Selection) is presented as a possible solution. Accordingly, the autonomous units have to be ordered in a "taxonomy graph". To be close to the nature the authors refer to the classification of living organisms as suggested by Linne (see for example [LSL]). The T-Selection considers three kinds of restrictions: Inclusion of possible partners, favourite partners, and exclusion of not acceptable partners (see Figure 2-4).

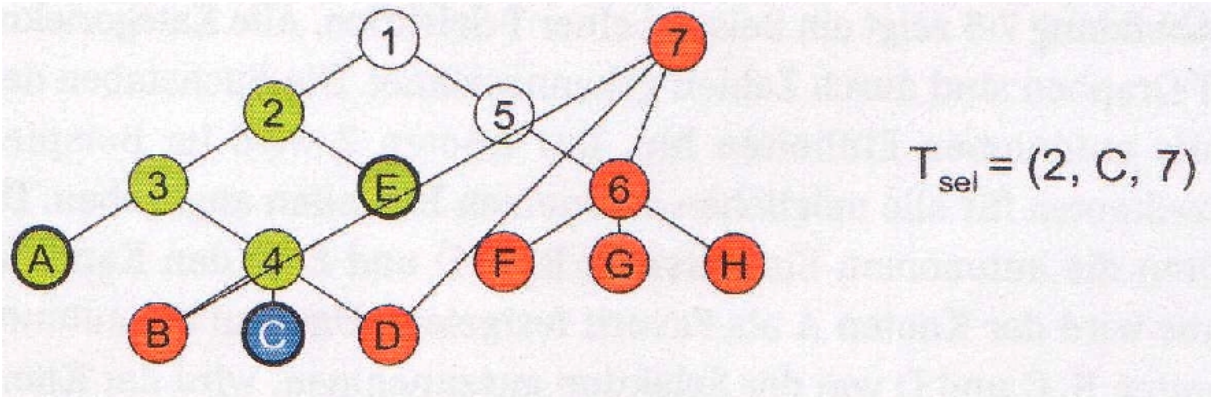


Figure 2-4 Taxonomy selection: Restrictions about possible partners (green), favorite partners (blue) and excluded partners (red) are made on taxonomically ordered autonomous units [H+02].

By giving T-Selection examples, the authors discovered that one “tree” is not sufficient to classify all autonomous units. This is due to the fact, that a taxonomy represents only one relation between its elements, normally the “subclass of” relation. The exclusion of autonomous units due to not acceptable values of some other property, emphasises the need for a multidimensional classification. This, however, is out of the scope of taxonomies.

The question of how to select of one specific offer from a list of offers is not addressed by EvoArch.

The conclusion of contracts in EvoArch is merely a technical step, but the legitimation of contracts is surely a complex theme. The authors speak about levels of contracts, where professional staff has full control on the contracts, while drivers get only limited control. Such vision indicates not only issues of practicability, but also the major issue of how to control self-organizing systems. The contradiction has been later addressed in [SMS08] where “controllable self-organizing systems” come into discussion (see section 2.1).

2.2.4 Conclusion

This section deals with the design methods of automotive systems. The survey begins with an illustration of standardization efforts like OSEK and AUTOSAR. The standardization (especially in the case of AUTOSAR) is then proved to be a necessary step to prepare the underground for following innovative approaches (DySCAS and EvoArch) which try to go a step further towards a self-organizing vehicle.

The autonomy and the self-organization win increasing recognition by automotive manufacturers and represent a paradigm change in the world of automotive systems, i.e. the change from the static human-controlled design to the dynamic and adaptive self-organization.

In section 3.1, DySCAS and EvoArch will be evaluated from the Organic Computing point of view.

2.3. The Contract Net Protocol

This section presents the Contract Net Protocol (CNET Protocol) as an established type of marketplace-oriented behavior. The information provided here will help to evaluate it from the Organic Computing point of view in section 3.1.3.

The Contract Net Protocol (CNET Protocol) has been suggested by Smith [Smi80] to specify problem-solving communication and control for nodes in a distributed problem solver. The network is assumed to consist of loosely coupled asynchronous agents, while each agent can communicate with every other agent by sending messages [XW01]. The agents play different roles in the network. A *manager agent* decomposes a task into subtasks and tries to find cooperating *contractor agents* to perform these subtasks.

The Contract Net Protocol addresses only the messages exchanged between the agents. The protocol defines a set of main messages: The task announcement (or call for proposals, cfp), the propose messages, accept/refuse proposal, and inform message (after performing the task by the participant). The concrete types of messages differ between versions of the Contract Net Protocol. Figure 2-5 shows the Contract Net Protocol as defined by the FIPA (Foundation for Intelligent Physical Agents) [FIPA02].

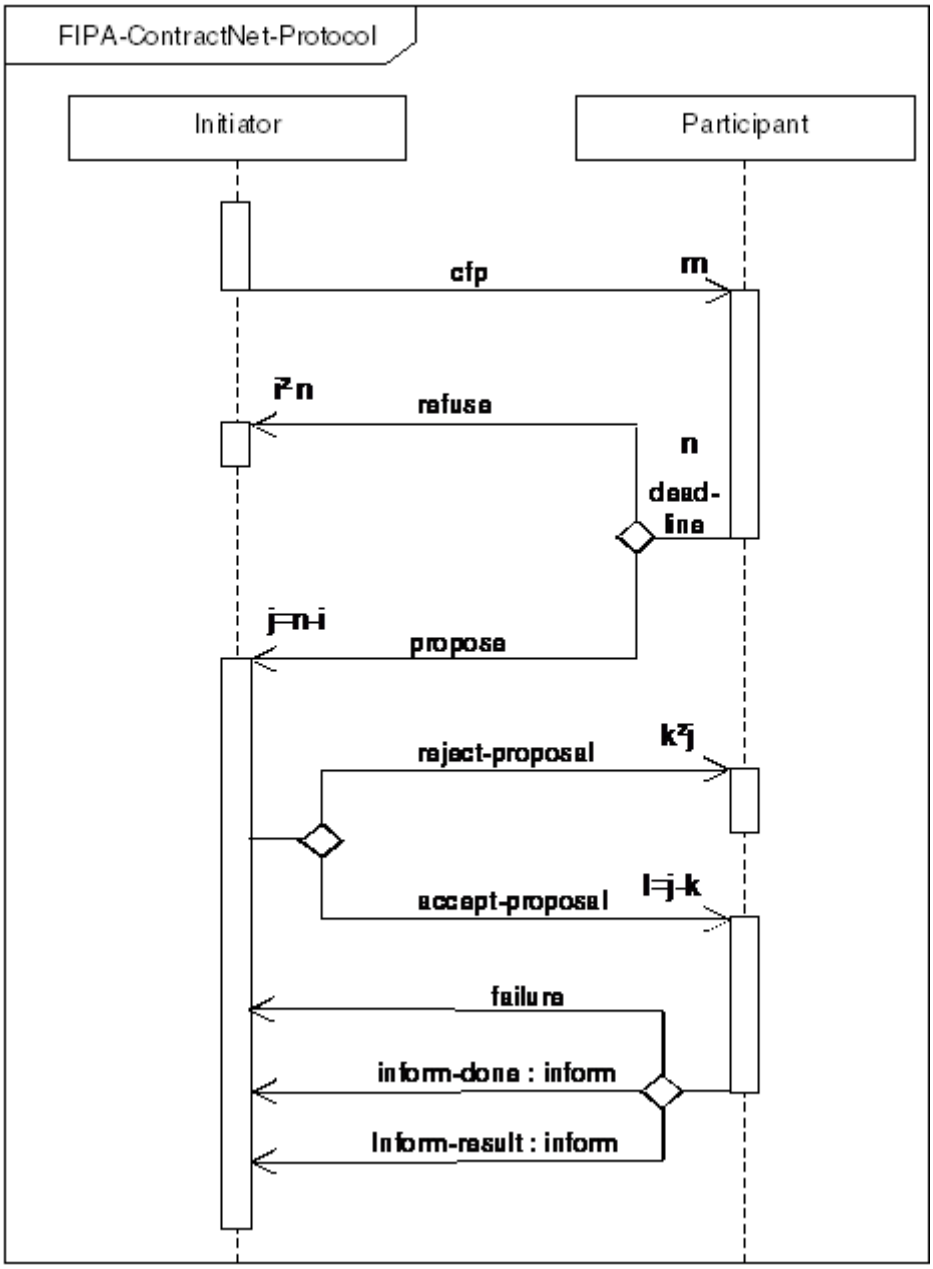


Figure 2-5 The Contract Net Protocol as defined by FIPA [FIPA02].

The Contract Net Protocol defines concrete forms of the messages. The task announcement message, for example, contains a task abstraction (with the type of the task) and eligibility specification (a kind of must-have properties), bidding specification (a scheme of description fields of expected bids) and the expiration time field of the announcement. On the base of the task type and the eligibility specification a contractor agent decides about sending a bid in response to a task announcement.

On the other side, the manager agent maintains a rank-ordered list of bids that have been received for the announced task. The manager awards the contract to the satisfactory bidder. The definition of satisfactory is task-specific [Smi80].

The Contract Net Protocol offers a framework that specifies the type of exchanged information, but it doesn't define the content of these messages. The matching between the announcements and bids must depend on a *common internode language*. However, the Contract Net Protocol suggests only that in contract nets all tasks are typed.

Only the approach called “Ontology-based Services for Agents Interoperability” [Mal06] discovers the need for an ontology-based common language to “facilitate the interoperability among heterogeneous agents that belong to a multi-agent system (MAS) dedicated to Business-to-Business (B2B) electronic commerce”. The ontology-based matching is implemented in a marketplace architecture, which adopts the Contract Net Protocol of FIPA.

The approach of Extended Contract Net Protocol (ECNET) in [PYM07] suggests the usage of TOPSIS (a method of Multi-Attribute Decision theory, see section 2.5.3) to add a standardized selection capabilities to the manager. To reduce the communication load it adopts a method called a Case-Based Reasoning (CBR). The CBR suggests to learn from the (bad/good) experience made with specific contractor (in terms of its success to perform previously awarded tasks, and of its cooperation by accepting awarded tasks) to develop a useful announcement strategy (to avoid the broadcasting of announcements as much as possible).

2.4. Semantic Web

The Semantic Web (SW) is of the most important inspiration sources of this thesis. This section presents the Semantic Web briefly.

The (normal) web offers the possibility to put huge numbers of pages and valuable information to the hand of the users. As the web grows through years of extensive usage, weaknesses of information management have been discovered. A basic service like the search for a specific word shows that web pages can only be searched for syntactically equal words. Unexpected search results appear when a word has different meanings in different contexts (polysemy problem). At the same time, the search is not always complete, because other people use other words to refer to the same meaning (synonymy problem).

Software tools can only process the contents of the normal web syntactically, because the meaning of the contents is not machine-accessible. This hinders the automatic processing of data, especially for extracting implicit knowledge from the explicitly available knowledge on the web pages. The interoperability and the data sharing between applications and enterprises suffer under the drawback of the normal web.

2.4.1 The Semantic Web

Most resources trace the birth of Semantic Web to Berners-Lee [BLHL01] and his famous words: “The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation”. In the Semantic Web, data are defined and linked in a way that enables its use for more effective discovery, automation, integration, and re-use across various applications.

2.4.2 The role of ontologies in the SW

To enable the vision, the Semantic Web is designed to rely on ontologies. The word ontology stems from philosophy, but it means from the engineering point of view “a formal and explicit specification of a conceptualization” [Gru93a, Gru93b]. Terms and relations between them build a conceptualization, i.e. a view of the world. Explicit specification means that the terms and the relations are explicitly given names and definitions. A formal specification has the advantage of being machine processable.

With the help of ontologies, knowledge is explicitly defined in a formal way (to make this knowledge available for “machines”) and ambiguity of concepts can be avoided by using a

common ontology or by mapping ontologies to each other. To enhance the search service, generalization or specialization of the searched term can be automatically suggested.

2.4.3 Ontologies versus taxonomies

Although typically linked to hierarchies of classes (concepts related to each other through sub-class relation; i.e. taxonomies), ontologies may include more information, like properties (any relation between -instances of- classes), value restrictions, disjointness statements, and cardinality of properties.

2.4.4 The SW in the practice

In the Semantic Web, web pages are annotated with meta-data, which use terms defined in ontologies. The meta-data give the meaning of the described web page (thus the term semantic). This way, "personal agents" can read the meta-description, extract information about resources (web pages or persons or things mentioned there), deploy ontologies to interpret retrieved information, and try to draw conclusions by applying logical reasoning [AH08].

2.4.5 Reasoning

Reasoning on ontologies helps to uncover knowledge, which is implicitly given. Reasoning (or inference) is known from the Artificial Intelligence (AI). From the AI it is known that sound (all derived statements follow semantically from the premises) and complete (all logical consequences of the premises can be derived) proof systems exist for predicate logic. Some ontology languages can be viewed as specializations of predicate logic. RDF, OWL-Lite and OWL-DL (see below) can be counted under these languages [AH08]. "DL" refers to the Description Logic, which is the theoretical background of ontology languages.

The Description Logics (DLs) denote "a family of knowledge representation formalisms that allow to represent the terminological knowledge of an application domain in a structured and well-defined way" [Kue01]. In the terms of description logics, the knowledge base (KB) consists of two parts: A TBox stores the conceptual knowledge (vocabulary) of an application domain, while an ABox introduces the assertional knowledge (instances with their properties). From a set of given concepts and properties, more complex concepts and properties are built using concept and properties constructors. DLs differ in the types of supported constructors. "Standard inferences" in the description logic cover three types of reasoning about the conceptual knowledge and two types of reasoning about the assertional knowledge. Types of reasoning about the conceptual knowledge are:

- Satisfiability check of a concept: Checks the concept about self-contradiction.
- Subsumption relation check between two concepts: Used to compute the subconcept/superconcept hierarchy of concepts.
- Equivalency check between concepts.

The reasoning about assertional knowledge answers one of two questions:

- The question about the consistency of the assertional knowledge in relation to the conceptual model (is there a model of the assertional knowledge and the conceptual knowledge).
- The question about relation of one instance to one concept: is it an instance of the concept.

There is a well studied [Kue01] trade-off between the expressive power of the DL language

(i.e. the set of supported constructors) and the complexity of the inference. High expressive constructs move the inference problem to higher complexity classes (in terms of the computational complexity theory) and can even make it undecidable.

Ontology editor tools support the standard inferences, known from the description logics. So a concept hierarchy (subsumption) can be automatically built, and the consistency check (in relation to concepts and instances) is always a part of ontology design. Building a hierarchy of concepts through the development process of an ontology is not only a way to present the ontology in a better way, enabling better browsing facilities and enhancing therefore the re-usability of the ontology. Just like the consistency check it is a way to evaluate the resulting ontology. If concepts occur at unexpected positions in the hierarchy then this indicates a mismatch between the intuition underlying a concept and its representation in the ontology [Kue01]. The check for equivalency helps to avoid the definition of redundant concepts in the ontology.

Adding user and application specific rules, the reasoning allows for new services in relation to domain knowledge discovery and/or action triggering (as a type of First Order Logic).

Retracing the inference steps provide an explanation for the conclusions. This great advantage is supposed to enhance the confidence in the Semantic Web. The retracing of proof can be also achieved by agents, in order to validate the proofs made by other agents [AH08].

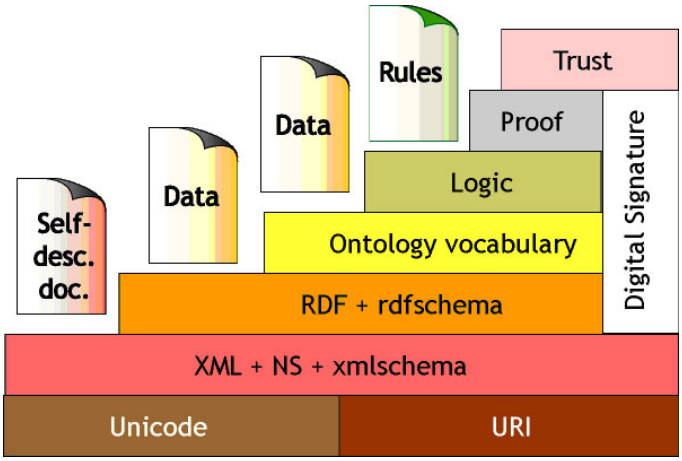


Figure 2-6 Stack of the Semantic Web (from www.Semantic-Conference.com)

2.4.6 The stack of the Semantic Web

Figure 2-6 shows the Semantic Web stack. The Semantic Web makes use of Uniform Resource Identifiers (URI), to identify resources, i.e. web pages and terms defined in the ontology. As a character set, the Semantic Web adopts the Unicode [Uni07]. The syntactic base of the documents is built on the eXtensible Markup Language XML [W3C06]. Using namespace, similar names can be used in different context, without causing conflicts. The data model of the Semantic Web is represented by the Resource Description Framework (RDF) [W3C04]. According to the RDF data model, each "statement" is a triple consisting of a subject, a predicate and an object. Subjects and objects, and even the predicates are "resources". RDF documents can be "serialized", i.e. transferred through the web, in their XML representation. RDF Schema can be viewed as a primitive language for writing ontologies. Key primitives are classes, properties, subclasses and the subproperty relationship, in addition to domain and range restrictions. The ontology languages add more complex relationships. For example OWL (Ontology Web Language [DS04]) adds details about properties (difference between object properties and datatype properties), restrictions about

the cardinality of properties and their ranges, beside boolean constructs and equivalence and disequivalence relations. All of these extensions are available in OWL-Full and partially in OWL Description Logic and to restricted extent in OWL-Lite.

The Logic layer enhances the whole concept by adding reasoning capabilities on the base of application specific knowledge/rules.

The proof layer achieves the proof validation, and uses special languages to present the proofs in suitable forms (retracing).

Additionally, the trust layer integrates digital signatures and encryption algorithms in the concept. This way, it is not only possible to know "how" a proof has been evolved, but also "who" is the responsible agent (or user).

2.5. Decision problems considering multiple criteria

This section provides a survey about methods of making decisions under the existence of multiple criteria. The importance of these methods for purposes of Organic Computing systems will be presented in section 3.2, while their suitability will be evaluated in section 4.1.

Multi-Criteria Decision Making refers in general to problems of decision making under multiple and mostly conflicting criteria. These problems are faced by the management and engineering team in different application areas: Business management [Ols96], energy planning, telecommunication network planning [FGE05], etc.

The decision making problems under multiple criteria have been given a kind of formal definition. It stems from the field of Operations Research (also known as Management Science, MS). Vansnick [Van95] suggested a model of decision problems under multiple criteria using three elements [Mar99]:

- A set of potential alternatives (or actions [Roy05])
- A set of criteria (or attributes)
- A set of performance evaluations of alternatives on each criterion.

A similar model, however a little more complicated, can be found in [Roy05]. The latter considers the type of the problem as a part of the model. At least from the "aiding tool" point of view there is a difference between "choice" problems (lying in the heart of the review in the following sections) and other types like: The problem of "sorting" alternatives into predefined categories, the problem of "classifying" alternatives (grouping similar alternatives into one class) or the problem of "ranking".

According to the nature of alternatives, the problems of decision making in presence of multiple criteria can be classified into two general groups [ES04]: Multi Objective Decision Making and Multi Attribute Decision Making.

2.5.1 Multi-Objective Decision Making MODM

MODM considers decision making problems with multiple objectives. An objective takes the form of maximization (or minimization) of a (linear) function gathering multiple decision variables in the continuous space (Although the term "criteria" is not usually used in the context of MODM, the decision variables can be understood as criteria). "Constraints" limit the acceptable values of the decision variables within a "feasible set". The objectives are usually interpreted as vectors in the continuous space, and the whole problem is therefore a vector optimization problem. In the general case, the objectives are conflicting, and there is accordingly no absolute "optimal solution", but only a set of optimal solutions, each with

respect to a certain objective function. A point in the space is called a "non-dominated" solution or "Pareto-optimal" solution (Pareto spoke in the 19th century about "maximum ophelimity[1]" [Ehr05]) if any displacement from it to achieve better performance of one objective means a worsening of (at least one) other objective. All non-dominated solutions are located on the boundary of the feasible set. The set of non-dominated solutions forms the "non-dominated frontier" or the "efficient frontier". All approaches trying to identify this frontier share the disadvantage of giving an infinite number of solutions, whereas the decision maker prefers to find only one - optimal - solution.

Different approaches address the problem of restricting the number of solutions to a finite set (of alternatives), which can be easily taken into consideration by the decision maker. Under these approaches, two effective and simple ones are known as the "weighting method" and the "constraint method". The former relies on giving weights to the - linear and convex - objectives in order to build one composite objective function. The latter method turns all but one of the objective functions into constraints with the help of –known or reasonably selected - target values. Both methods result in a linear programming problem, which can be solved by pertinent methods [ES04]. The process is usually carried out repeatedly, for different weights in the first method, and for different target values and objective functions in the second. Apart from the method used, the result is always a suggested set of "discrete solutions", building the typical kernel of Multi-Criteria Decision Making problems.

2.5.2 Multi-Criteria Decision Making MCDM

MCDM is a widely used term to refer to Multi Attribute Decision Making (MADM) or sometimes to Multi-Criteria Decision Aids MCDA [Ehr05]. However, all these names don't clearify the difference to the MODM counter problems. In fact, MCDM considers decision problems with a finite number of alternatives (discrete solutions), which are explicitly known. The outcome of alternatives in correspondence to the criteria is also supposed to be known and certain. Beside the classical MCDM methods, many approaches treat decision problems under special conditions. The "games against nature" treat the problem of the uncertainty about the outcome of the alternatives (therefore also called "decision making under uncertainty") [ES04][Ste05]. In this kind of games a rational player (the decision maker) faces a second player (the nature) which doesn't try to reach predefined goals and behaves therefore unexpectedly. Typical examples of players with unexpected behavior are the weather and the changing demand for specific goods on the market. The "game theory" supposes the existence of more players (2 at least), each with his own objectives and decision space [ES04]. Uncertainty about the outcome of the alternatives arises here because of the unknown reactions of other players.

The next section describes methods of MCDM with known alternatives, known criteria, and with certain outcome of alternatives. However, uncertainty about the decision maker's preferences is always present in MCDM problems. This type of uncertainty is taken into account by some of MCDM methods.

2.5.3 MCDM methods

Decision making under multiple criteria takes place generally according to a generic procedure, which can be broken down into four steps [Tri00]:

- Problem analysis
- Weighting of criteria

[1] The term "ophelimity" has the meaning of *i.* economic satisfaction or *ii.* the ability to please another [Wik08c]

- Utility assessment of alternatives' performances on the criteria
- Ranking and decision making.

2.5.3.1 Problem analysis

In this step the decision maker analyses the decision making problem to its basic component: A goal, a set of criteria, and to a set of alternatives. In the AHP (Analytical Hierarchy Process) [GWH89] method this step results in a tree (hierarchy) with the goal at the root, criteria as branches and the alternatives as leaves. The criteria build normally more than one hierarchical level. The problem analysis is important to clear the problem under consideration, especially to find the set of "all", "relevant" and "non complementary" criteria. In some MCDM methods like the value tree method [ES04], the tree of criteria helps also to determine the weights associated with the criteria. The importance of this step and its common points with ontology-based approaches (like SeMCDM) is discussed in section 2.6.

2.5.3.2 Weighting of criteria

Criteria have usually a compensatory nature, but they are of different importance for the decision maker. Each criterion has accordingly a "weight", which defines its importance in relation to other criteria. The sum of weights gives usually the value of 1. The given weights of criteria have to represent the preferences of the decision maker. Different methods have been developed to help the decision maker to transform his "nonnumeric" preferences into weights. A survey of these methods is presented in the following sections.

- ♦ Fixed point scoring

This method is the most direct way to obtain weighting information from the decision maker [HMS00]. The decision maker is required to give weights of criteria, while bearing in mind that the weights have always to sum up to 1.

- ♦ Fixed point scoring of hierarchically ordered criteria

This method is used in the so called "value tree" method, described in [ES04]. Building a tree of criteria and sub-criteria has the advantage that the decision maker is only asked to specify weights of criteria located on one level. The weighting will be then performed separately on each hierarchical level. While it appears to be a simple task compared to the situation of unordered criteria, this method suffers from the disadvantage of confronting the decision maker directly with the weighting problem.

- ♦ Rating method

The decision maker assigns an importance value to each criterion, usually within a predefined range: 1-5, 1-7 or 1-10 [HMS00]. The weights are then calculated through the normalization of these values. Still, the importance values have to be defined directly by the decision maker. However, compared with the fixed point scoring method, this method releases the decision maker from the obligation of keeping the sum of the given values at 1.

- ♦ Rating method of ranked criteria

This method is the weighting method of SMARTS (Simple Multi-Attribute Rating Technique with Swing weighting) described in [Ols96]. It is based on the "swing operation" to rank the criteria. The swing operation starts by giving a hypothetical worst case alternative (i.e. with worst performance on all criteria). The decision maker is then asked to select one criterion to improve its performance (to the best performance value) in order to achieve a maximum

improvement effect on the alternative evaluation. The same question will be repeated until all criteria are completely ranked. The most important criterion is the first one selected in the "swing operation", while the least important one is the last one selected. The most important criterion is then given automatically a score of 100. The decision maker gives the next criteria successively scores between 0 and 100. The weights can be then calculated, as in the rating method, by normalizing the scores.

- Ordinal ranking method

This method asks the decision maker to rank the criteria in an order of importance [HMS00]. Quantitative values of weights are then calculated by methods like the centroid method or the basic approach [HMS00]. SMARTER (Simple Multi-Attribute Rating Technique Exploiting Ranks [Ols96]), a following version of SMART and SMARTS, adopts the SMART step of criteria ranking beside the centroid method to calculate the weights automatically. The mathematical formula for weights calculations according to the centroid method can be found in [Ols96] ([HMS00] refers to the method as the "expected value method"). Such methods risk a kind of approximation errors, in order to offer more ease of use. The higher the number of criteria the smaller is the approximation error by SMARTER.

- Simple Multi-Attribute Rating Technique SMART

SMART, as described in [Ols96], suggests a complete MCDM procedure of 10 steps, whereas the steps 4 to 7 together deal with the weighting issue. After the step of criteria identification and criteria reduction to the important ones, the decision maker is asked to rank the remaining criteria. The weakest criteria (i.e. least important) gets the importance value of 10. In a following step the decision maker compares each criterion with all lower importance criteria, to give it relative importance ratios. This takes place in a successive way beginning with the second weakest criteria. The consistency of the relative importance ratios (to all criteria of lower importance) has to be proved for each criterion, pushing the decision maker to reconcile the ratings. In the last step the relative importance values are normalized into weights summing to one. The ascending number of needed comparisons and reconciliations with each additionally criteria justify the step of criteria reduction, where the decision maker prefers to turn a blind eye to some criteria. This method relies on paired comparisons (see methods allowing inconsistency) but it is characterized by enforcing iterative and manual consistency check.

- The multi-attribute value functions method

Although the method has been presented by Keeny and Raiffa (s. [ES04]) as an aspect of the multi-attribute value function method, it can be considered as a separate weighting method. The application of this method supposes that the minimum and maximum criterion values are known. To extract the decision maker preferences about two criteria the decision maker is asked to compare two hypothetical alternatives. These alternatives can be presented as points on a 2-dimensional space, built by two axes of criteria 1 and 2 under consideration. The first alternative has the coordinates (value of criterion 1 with maximum utility, value of criterion 2 with minimum utility) while the second alternative has the coordinates (value of criterion 1 with minimum utility, value of criterion 2 with maximum utility).

The decision maker has different impressions about the hypothetical alternatives. So, he will be asked to replace the weak alternative with a new one that makes a preference balance to the other alternative (the exact procedure is described in [ES04]). The preference balance is interpreted as a mathematical equation according to the weighted sum method (section 2.5.3.4). By repeating the same procedure on all criteria (in comparison to a selected "base

criterion") the preferences of the decision maker can be transformed into a set of equations. Recalling that the sum of weights gives always 1, the equation set can be then solved to calculate the weights of criteria.

- Preference cones

The preference cones method is described below under the ranking and decision making methods (section 2.5.3.4). As a side effect, this method produces constraints on the weights of criteria. The method doesn't promise a complete solution for the weighting problem (because, in some cases, a decision can be made without producing enough constraints on the weights). Additionally, the resulting weights are strongly related to the considered alternatives.

- Methods allowing inconsistency

These weighting methods are based on paired comparisons between the criteria. For n criteria, a perfectly accurate decision maker would need only to make n-1 comparisons to give a complete picture of his own preferences. Instead, the methods described here, ask the decision maker to make $n \cdot (n-1)/2$ comparisons (equals the number of all possible comparisons). The given estimations are usually ordered in a matrix of size $n \cdot n$. Consistent estimation matrices have the characteristics that the ratio $\text{Estimation}_{x,y}$ / $\text{Estimation}_{x,z}$ is the same for all values of x between 1 and n. The decision maker is usually not able to give consistent estimations, especially for higher numbers of required comparisons. However, incorporating all possible comparisons in the calculation process (of weights) the negative impact of inaccurate estimation is hopefully diminished [Tri00]. Therefore, these methods have the advantage of allowing a kind of inconsistent estimation from the decision maker. Some methods provide the decision maker also with feedback, on how inconsistent the estimations are (consistency check). The reliability of the obtained weights depends on the consistency of the estimations matrix. So, the decision maker is asked to try to enter better (i.e. relatively more consistent) estimations, when the consistency check shows useless estimations (i.e. it is too inconsistent).

There are two types of pairwise comparisons, ratio comparisons and difference comparisons [Tri00]. By the ratio comparisons the decision maker is asked to answer the question: "How many times is the criterion x more important than the criterion y?", while the question takes another form for difference comparisons: "How much is the criterion x more important than the criterion y". The methods described in this section are those used for MCDM, and they are all based on ratio comparisons.

Three methods are counted among the group of ratio comparison methods:

- The normalization method
- The Analytical Hierarchy Process (AHP)
- The geometric mean method.

The common start point of all of these methods is the estimation matrix which contains pair-wise ratio comparisons between the criteria. Elements on the main diagonal have always the value of 1 (comparison between a criterion and itself). Each element has always the reciprocal value of its diagonally symmetric element (i.e. the matrix is reciprocal).

The normalization technique (not to be confused with the normalization methods in section 2.5.3.2) is very simple: The weights are calculated as the average values of the rows, after the normalization of the columns' values.

The Analytical Hierarchy Process (AHP), as presented by Saaty [GWH89], is based on the so called "eigenvalue theory". Saaty proved that for a consistent estimation matrix, the number of criteria n is a right eigenvalue, and that the weights build a corresponding eigenvector.

According to Saaty, inconsistent estimation matrices are to be considered as a perturbation of the consistent case. Slight changes in the matrix elements induce similar changes in the eigenvalues. Moreover, the largest eigenvalue is a real number greater than *n* while the remaining eigenvalues are close to zero [Tri00]. According to Saaty, the eigenvector for the largest eigenvalue represents the weights of criteria, and this applies also for inconsistent estimation matrices (see the discussion below about the consistency ratio and its maximal acceptable value).

To calculate this eigenvector, an iterative method is used traditionally in combination with Saaty's approach. In each iteration step, the estimation matrix is squared and the rows are then normalized to calculate an approximation of the eigenvector for the largest eigenvalue. The iteration stops when the difference between two successive iteration steps is smaller than a predefined accuracy limit.

Following a general method to calculate the eigenvector for the largest eigenvalue, the calculation process of the weights of criteria from a matrix **A** containing inconsistent estimations of pair-comparisons can be solved in the next steps [ES04]:

Calculate $\lambda_{\max}$ the largest eigenvalue of the estimation matrix A by solving the equation:

$$\det\left|A - I\lambda_{\max}\right| = 0$$

Where: "I" denotes an identity matrix of appropriate size (the result is always a real number greater than n).

For the largest eigenvalue determine the corresponding normalized eigenvector, which represents the vector of weights W, by solving the n equations resulting from the eigenvector definition:

$$AW = \lambda_{\max} W$$

Augmented by the equation:

$$\sum_{i=1}^{n} w_i = 1$$

Saaty calculated also a Consistency Index CI using the formula:

$$CI = (\lambda_{\max} - n)/(n-1)$$

A Consistency Ratio CR is obtained by dividing the consistency index by the Random Consistency Index RCI:

$$CR = CI / RCI$$

where the RCI is an empirically calculated value that depends on the size of the matrix and can be taken from reference tables, like Table 2-1 from [ES04].

N	3	4	5	6	7	8	9	10
RCI	0.52	0.89	1.11	1.25	1.35	1.40	1.45	1.49

Table 2-1 Random Consistency Index (CRI) in relation to the matrix size n

Any consistency ratio in excess of 0.1 should be viewed with suspicion and should be double checked [ES04]. The value of 0.1 is not a trivially selected value, since the eigenvector method is proved to be applicable for inconsistent ratio estimations only if the consistency ratio is equal or less than 0.1. The mathematical origin is shown briefly in [Tri00], while a

completely different approach to measure consistency of the estimation matrix can be found in [ES04].

The RCI values in Table 2-1 are valid when using the scale of estimations proposed by Saaty, which contains the values set {9, 8, 7… 2, 1, 1/2 1/3... 1/7, 1/8, 1/9}. In its abstract definition, a scale is a one-to-one mapping between the set of linguistic expressions available for the decision maker and a discrete set of numbers which represent corresponding quantitative importance values of them. Saaty introduced a map between a set of qualitative (linguistic) importance expressions and the quantitative values in his scale as shown in Table 2-2.

Quantitative importance value	Linguistic importance expression
1	Equal importance
3	Weak importance of one over another
5	Essential or strong importance
7	Demonstrated importance
9	Absolute importance
2,4,6,8	Intermediate values between the two adjacent judgements

Table 2-2 Saaty's scale of relative importances

The adoption of this scale by Saaty is based on psychological experiments showing that most individuals cannot simultaneously compare more than seven objects, plus or minus two (Variations from Saaty's linear scale and other scales, for example the exponential scale, are also supported with some psychological arguments [Tri00]).

The geometric mean method, as presented in [ES04], suggests calculating the weights as the normalized geometric means of the rows of the estimation matrix.

[ES04] shows that weights obtained by the three methods (the normalization method, the AHP and the geometric mean method) have very similar values.

2.5.3.3 Utility assessment of alternatives' performances on the criteria

Every alternative has a specific performance value on each criterion. The values of alternatives' performances can be presented throw a matrix $m \times n$, where m is the number of alternatives (rows) and n is the number of criteria (the columns). This matrix is referred to as "decision matrix" or "achievements matrix". The elements of the matrix are only comparable in the same column (for one criterion), but not in the rows (i.e. for different criteria). Additionally, the performance values can be of any type: Numeric or qualitative. Therefore, multi-criteria decision making requires a transformation of these performance values into dimensionless comparable values called utility values or utilities. The utility values fall in the range [0:1]. This transformation process is called here as "utility assessment".

The next sections describe different methods of utility assessment.

- Normalization methods

The simplest way to transform the performance values into utilities is to normalize them on each criterion separately.

Accordingly, the performances of alternative i on criterion j are to be normalized using a mathematical relation like:

$$u(x_{ij}) = \frac{x_{ij}}{\sum_{i=1}^{n} x_{ij}}$$

Where: x_{ij} is the performance value of alternative *i* on criterion *j*.

To give every criteria a unit length of vector [HY81], some approaches like TOPSIS and ELECTRE [Tri00] normalize the performance values using the formula:

$$u(x_{ij}) = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{n} (x_{ij})^2}}$$

Some normalization methods assign the largest available performance value to the maximal utility value of 1, and the smallest performance value to the minimal utility value of 0. All other performance values get then a linearly distributed utility values according to their performance values.

In all these methods it is remarkable that the resulting utilities are related to the set of alternatives under consideration. This means that one alternative would be given different utility values in different consultations of available alternatives (this fact will play an important role in section 4.1 and especially in section 4.1.2.3).

- Predefined functions

Defining a utility function (called also value function) the performances of alternatives on a criterion can be converted to a utility value, between 0 and 1. How to choose the suitable shape of the utility function is an important question that faces the decision maker. A simple linear utility function can represent maximization criteria (higher performance values mean higher utility values) or minimization criteria (higher performance values mean lower utility values). However, a linear function cannot express the saturation effect, which is typically found by concave utility functions. The saturation effect can be explained by a typical example as follows: The first dollar in profits is much more important than an additional dollar that is gained when the profit has already reached a higher level [ES04]. To make a rough estimate about the shape of the utility function (linear or concave), the decision maker can try to assign a performance value to the utility value of 0.5. Putting this performance value in the middle point between the minimum and the maximum performance values indicates a linear utility function, other case; the utility function is merely concave shaped.

The last discussion guides to a next question about how to set the functions parameters. While the decision maker is usually not able to choose suitable values directly, the parameters should be calculated depending on "known points", given by the decision maker. The linear function, for example, has two parameters to be set. Consequently two points should be defined by the decision maker. A common approach used for linear utility function is to ask the decision maker to give the performance points corresponding to the maximum utility value (of 1) and to the minimum utility value (of 0). The utility function is then considered as a straight line between these points. This approach, adopted by SMART [Ols96], is known as linear interpolation or as Edwards procedure (it has been suggested by Edwards in the 1970th [ES04]). Edwards procedure is clearly limited to linear functions and to cases, where the decision maker is able to choose a maximum and a minimum performance values. So the resulting utility functions cannot be used generally for all possible alternatives, which may have performance values greater then the chosen maximum one, or smaller than the chosen minimum.

To specify utility functions with more complex shapes, the user has to give more information: Three points for quadratic functions or exponential functions of the form: $u(x) = \alpha + \beta x$ [ES04], and four points for cubic functions. In some commercially available tools, like Logical Decisions [LD], this step is accomplished graphically by specifying the performance

value which yields a utility value of 0.5, then a curve is automatically fit through three points of utility values 0, 0.5 and 1 [Ols96].

- Utility assessment of qualitative values using predefined scales

Numeric utility functions cannot be used to assess the utility of qualitative values. The predefined utility scales are qualitative counter parts of the numeric utility functions. A utility scale contains a finite number of utility values, covered with names like: “very good”, “good”, “medium”, “poor”, and “very poor”. While these names represent only 5-points scale, any practical number of points can be used. Beside the 5-points scales, 7- and 9-points scales are common ones. These scales are known as Likert scale [ES04]. By a 5-points scale, 5 utility values can be expressed, like: 1, 0.75, 0.5, 0.25 and 0.

The assignment of utility values to the qualitative values of the criteria should be done manually by the decision maker.

- Methods allowing inconsistency (paired comparisons methods)

The paired comparison methods (see section 2.5.3.2) can also be applied to assess the utilities of alternatives’ performance on the criteria. The comparison takes place this time between alternatives and for each criterion separately. The decision maker fills out a matrix ($m \times m$) for each criterion, where m is the number of available alternatives. Utility value of each alternative can be calculated using the normalization technique, the analytic hierarchy process, or the geometric mean method (see section 2.5.3.2).

2.5.3.4 Ranking and decision making

The ranking and decision making is the last step of the decision making procedure. In its simple and direct form, a mathematical formula gives each alternative a rank depending on the weights of criteria and on the utility of alternative on each criterion. Such mathematical formula is called the “aggregation function”. The most famous aggregation functions are the weighted sum and the weighted product functions. Normally, higher values of rank are related to better alternatives. However, the rank is not necessarily a result of an aggregation function. In the following the ranking and decision making methods are described and shortly discussed.

- Weighted sum (Additive weighting method)

The evaluation of each alternative is calculated as the weighted sum of its performance values on the criteria:

$$v(a) = \sum_{i=1}^{n} u_i(a) w_i$$

Where:

a : The considered alternative

$v(a)$: The evaluation of alternative a

$u_i(a)$: The utility value of the alternative a on criteria i

w_i : Weight of criteria i

The criteria must be numerical, comparable [CHH92] (the additive utility assumption [Tri00]) and independent of each other (performance of alternatives on one attribute is not related to their performance on other criteria). Non-numerical (qualitative) criteria have to be quantified

before applying this method. To obtain evaluations of alternatives with non-comparable criteria, its performance values on the criteria must be expressed as utilities (see section 2.5.3.3).

- Weighted product

Similarly to the weighted sum method, a weighted product can also be used to assess the alternatives:

$$v(a) = \prod_{i=1}^{n} u_i(a)^{w_i}$$

Where:

a : The considered alternative

$v(a)$: The evaluation of alternative a

$u_i(a)$: The utility value of the alternative a on criteria i

w_i : Weight of criteria i

This way, alternatives with poor performance values will be more heavily penalized [CHH92] (as shown in 2.5.3.3, the utility values fall in the range [0:1]). However, just like the weighted sum method, the weighted product method supports the compensation effect between the criteria. As done for the weighted sum. Non-numerical criteria have to be quantified and non-comparable criteria have to undergo the utility assessment process.

- Reference point methods (distance from target alternative)

These methods select the alternative which has the shortest distance to a "target alternative" [CHH92]. The criteria must be numerical (i.e. quantitative or quantified qualitative) and comparable (if the absolute performance values are used to calculate the distance) and independent of each other. In practical applications utilities, obtained by Edwards procedure, or normalized values are used for distance calculation. The weights of the criteria play a similar role as in the weighted sum method.

The famous TOPSIS method (Technique for Order Preference by Similarity to Ideal Solution) applies the reference point method using the normalization method (for utility assessment) and the Euclidean distance formula to two (hypothetical) alternatives: An ideal alternative and a negative-ideal alternative (the worst values on all criteria).

While these methods seem to be well-founded because of their direct and acceptable geometrical justification, the preference point methods suffer from the disadvantage of neglecting the need for predefined utility functions (the default utility function is a linear function). Like the normalization method used in AHP, a discrete example can demonstrate that the ranking results can be affected by the existence/absence of some alternatives. The same applies for the Edwards procedure (the resulting utilities depend on the existing alternatives). Furthermore, these methods cannot cope with situations where two or more preference points (ideal points) exist. TOPSIS assumes for this reason monotonically increasing or decreasing utility of criteria.

- Outranking methods

On the base on utilities and criteria weights, the outranking methods aim to derive dominance relations between the alternatives. The dominance relations are represented as a structure, usually a graph, which indicates which alternative is preferred to, or outranks, another. The

outranking may or may not be complete [ES04]. The famous ELECTRE method (Elimination et Choice Translating Algorithms) is not a complete method, while the PROMETHEE (Preference Ranking Organization Method for Enrichment Evaluations) method results in complete outranking (although only in its second version PROMETHEE II).

According to ELECTRE, two matrices have to be calculated: a concordance matrix and a discordance matrix. Both matrices represent comparisons between alternatives. In the concordance matrix, each element indicates the strength of the preference of alternatives to each other. A matrix element is calculated on the base of the proportion of those criteria weights, for which an alternative has a higher utility than another. The discordance matrix expresses how much worse alternatives are in comparison to each other. A matrix element here is calculated using the weights of criteria and the utility differences between the alternatives. Defining a minimal required concordance and a maximal allowable discordance (by the decision maker or as average values of the concordance and discordance accordingly) an outranking relation (x, y) between two alternatives (i.e. alternative x outranks alternative y) results if and only if: The concordance (x, y) >= the minimal required concordance AND the discordance(x, y) <= the maximal allowable discordance.

To recognize an outranking relation (x, y), ELECTRE II (a developed version of ELECTRE I) adds the condition that the concordance of the first alternative over the second (x, y) exceeds that of the second over the first (y, x). Still, ELECTRE doesn't result in a complete outranking.

The PROMETHEE method compares pairs of alternatives on each criterion by means of a preference matrix. To this reason, a preference function has to be first chosen for each criterion. Preference functions differ from the utility functions (described in 2.5.3.3), because they transform the performance difference (of two alternatives on one criterion) to utility difference. For example, for a criterion with a piece-wise linear preference function, the utility difference between two alternatives is set:

- To: 0: if the performance difference <= min. value
- To: (performance difference – min. value)/(max. value – min. value): if performance difference falls between min. value and max value
- To: 1 if the performance difference >= max value

Other forms of preference functions can also be used (like simple binary preference function, stepwise preference function, or exponential preference function). PROMETHEE don't define ways to choose and to characterize these preference functions.

However, the preference matrices will be then aggregated to one overall preference matrix using the weighted sum formula. From the preference matrix the row averages matrix ($\mathbf{P}^+$) and the column averages matrix ($\mathbf{P}^-$) are then calculated. PROMOETHEE I method gives out a graph using a similar outranking condition to that used by ELECTRE method. To get a complete ranking, PROMETHEE II ranks the alternatives according to the non-increasing values of the difference ($\mathbf{P}^+ - \mathbf{P}^-$).

♦ Preference cones

This method makes use of alternatives' utilities (utility matrix) and direct preference statements of the type: Alternative *x* is better than alternative *y*, i.e. alternative *x* dominates over alternative *y* or: $Dx > Dy$. The preference statements are then modified to unequations depending on the weighted sum of utilities. As the utilities of alternatives are known the unequations take the form of constraints on the weights. The question "is alternative *x* is also better than a third alternative *z* turns to a problem of finding a set of feasible weights that

ensures that alternative *z* can be better than alternative *x*. If such set of weights does not exist then the new preference statement *Dx* > *Dz* can be concluded. The procedure can be repeated with all existing alternatives. If the decision maker aims to select one alternative, preference statements have to be made until one alternative is proved to dominate all other alternatives. There is no way to know that additional alternatives could be proved as dominating or dominated alternatives. This limits the applicability of the preference cones method to the given set of alternatives.

- Special methods

Some MCDM methods do not follow the generic procedure described above. The Data Envelopment Analysis (DEA) method evaluates every alternative in comparison to a hypothetical alternative, which is generated as a linear convex combination of all other alternatives. The start point is the performance values of each alternative on the criteria. DEA method turns out to be a linear programming problem. The result of DEA is a kind of "efficiency" evaluation for each alternative. This value represents only a comparative value between the alternatives under consideration. [ES04] Shows that this method is not complete, i.e. some alternatives may become equal efficiency values. Furthermore, in some cases, only one criterion pushes the ranking towards a specific alternative, causing infeasible decision making.

Ariadne and Hipre3+ allow the decision maker to give less precise statements in form of range of weights and range of utilities. Such methods imply a kind of uncertainty about the made decision. In [Ols96] Ariadne is considered as a special case of SMART while Hipre3+ as a special case of AHP. Ariadne is not complete (i.e. it doesn't select a best alternative in all cases), in contrast to Hipre3+, which awaits more elicitation information from the decision maker until a best alternative is identified.

2.6. Ontologies and Multi-Criteria Decision Making problems

This section presents the approaches, which gather the ontological knowledge processing to methods of Multi-Criteria Decision Making. These approaches can be considered as basic forms of the methodology suggested in this thesis.

Although the decision making and ontologies have different origins and goals, a deeper investigation shows similarity and almost complementary aspects between them. From the theoretical point of view, the "explicit building of a criteria hierarchy", usually achieved with the help of multi-criteria decision making tools, is an important common point with the ontology based approaches. In [Mar99], Martal underlines the "positive effect" of the explicit building of criteria hierarchies on the modelling of decision making problems. This way, the human decision maker obtains a clear image of the decision problem under consideration.

The next subsections present practical approaches where ontologies and inference mechanisms meet the needs of a decision making process and enhance it.

2.6.1 Selection of optimal results of semantic queries with soft constraints

The approach around the Personal Preferences Search Service (PPSS) [A+07] addresses the search problem for learning resources. It provides enhanced search capabilities on the base of preference queries. In contrast to their traditional semantic counterparts, which allow only for hard constraints, the preference queries allow the usage of soft constraints. The soft constraints increase the expressivity of the queries by enabling a kind of a preference order, for example, an order for the preferred days of week for a special course (called scoring preferences).

To retrieve the optimal matches according to the user preferences, PPSS adopts the pareto composition between the criteria. This means that criteria are considered to have equal importance values. Such approach simplifies the usage of the PPSS, as no kind of criteria weighting is required.

The returned results of search have the form of a set of optimal items, found by applying the pareto domination principle.

PPSS has been implemented as a web service within the Personal Reader Framework [HK04]. Extensions of SPARQL (a query language) have been proposed to enable the enhanced search capability in a large data set (expressed in RDF) about available lectures.

PPSS offers high expressive, easy to use, enhanced semantic search capability. From the theoretical point of view, PPSS discovers an interesting application field of the pareto domination relation in semantic-based selection of a small set of optimal offers from a large set of available discrete offers.

However, PPSS doesn't propose to find the best available offer, but only to suggest a set of best available offers to a human decision maker.

2.6.2 Ontologies and decision making in loosely coupled management centres

The example application depicted by Lu *et al.*, in [LRLSM06], demonstrates the decision-making problem about design, operation and maintenance of the railway system in the UK. The privatization of the railway system has resulted in the separation of responsibility for various technical issues.

The privatization, in this context, is related to loosening the coupling between the management centres and consequently to more autonomy for distributed decision makers. This way, different separate parties have been involved in the decision making process through different stages of the system life cycle. A dangerous lack of knowledge evolves between these parties. While making a decision about a particular part of the railway system the dedicated decision maker is not aware of possible side effects on other parts.

The authors of [LRLSM06] suggest that the integration of the analysis tools has to take place through an internet-based decision support system DSS. [LRLSM06] adopts the definition of a decision support system from [Tur95] as "interactive computer-based system that aids the decision maker to gain a greater understanding of their business by taking advantages of both human intelligence and the ability of computers to access large quantities of data, develop models, interpret results and formulate and evaluate alternative solutions".

The solution suggested by [LRLSM06] does not rely only on a technical solution (internet) to integrate the tools of the different parties. It adopts also technologies of the Semantic Web to solve the problem of "expertise sharing". The standards related to the railway system and its components are gathered in a unique ontology. The implemented ontology has a structural hierarchy of three levels: Simple parts like wheels and rails, subsystems like wheel sets, bogies, tracks and vehicles, and the system level composed of vehicles and tracks together. There is no information about the size of the developed ontologies, but they are declared to represent 63 standards of railway systems in the UK.

With the help of a reasoner, the decision taken by one party can be proofed about its possible effects on other parts of the railway system, even throughout their whole life-cycle. In the kernel of the ontology is the property "affects".

The ontology development environment and the visualisation of the class hierarchy are based on Protégé [Pro08] and its plugins. The ontology can be accessed online by all parties.

2.6.3 Ontologies and basic forms of MCDM for competence management

A case study about the application of ontology-based competence management for healthcare training planning has been published in September 2006 [KS06]. The authors carried out a practical experiment in a German hospital to explore how skills- and competence management[2] concepts can be used in the healthcare domain, with special concentration on the training planning. The competence modelling shows the need for a common vocabulary between the separate wards of the hospital and led to "constructive discussions" between the involved employees. Moreover, the experiment emphasised the fact that competence modelling needs "multi hierarchies" as the primary modelling construct, and that "mono hierarchies" turn out to be not suitable because of many cross-references among disciplines.

[KS06] used the terms of "essential" and "desired" competencies. The difference is related to the suitable acquisition time of the competencies: Essential competencies must be acquired as soon as possible, while desired competencies are foreseen for future. In a following work [SK06], the same concepts have been implemented in an ontology for requirements' profiling. The authors declare "hard requirements" as "competencies that are absolutely needed" and "soft requirements" as "competencies that are a desired goal for short- to mid-term future".

A simple form of MCDM methods has been used, for example, to assess the utilities of the competences on the base of a five level scale (the authors refer to studies showing that the five level scale yields the "highest validity", without to define the meaning of the term). The so called "prioritization" (in the sense of weighting) of competencies has not been accomplished.

[KS06] states that the evaluation of the current competency modelling tools, in the context of human resources, has been "quite disappointing". According to the authors, available tools suffer from two main disadvantages: They seem to be designed for "small competency catalogues" (up to 100 competencies) and/or they support only mono-hierarchies in the form of trees. However, the authors do not refer to the evaluated tools. Moreover, to carry out the experiment, the case study adopted a mind mapping tool (MindManager X5, with a specific plug-in to allow "smooth ontology evolution") although they are aware of the lack of formality of mind maps, which give a strictly mono-hierarchical structure.

2.6.4 KOWIEN: Ontologies and goal programming for competence management systems

The project "KOWIEN" [SZ04] was funded by the Federal Ministry of Education and Research in Germany. The project consortium gathers the institute of Production and Industrial Information Management (PIM), University Duisburg-Essen, to five enterprises of different backgrounds: Mining services (Deutsche Montan Technologie GmbH), development of customized production machines (Karl Schumacher Maschinenbau GmbH), development of optical components and image processing units for production processes (TEMA GmbH), consulting (Roland Berger Strategy Consultants GmbH), and finally IT-Consulting and software development (Comma Soft AG).

The abbreviation "KOWIEN" is derived from the German title of the project: "Kooperatives Wissensmanagement in Engineering-Netzwerken", which means: "cooperative knowledge management in engineering networks".

The project suggests a new solution of the competence management problem in small and medium enterprises (SMEs). The competence management is one of the essential factors which affect the competitive ability of enterprises (costs of production, time-to-market, etc.).

[2] The term "skill" is used in relation to employees, while the term "competence" is used in relation to the enterprise, i.e. hospital.

The problem is especially noticeable in dynamic business environments embossed by personal turnover, virtual enterprises and job rotation. In KOWIEN, the "competence" is defined as "use-oriented know-what and know-how of actors". The natural actors in enterprises are the members of staff.

KOWIEN treats the competence management under major terms: The actors are supposed to work in cooperative way (excluding competitive coordination approaches), and the engineering tasks are knowledge intensive; i.e. they require human knowledge in form of know-what and/or know-how. Additionally, the engineering tasks shall be characterized by high "qualitative" diversity, i.e. the tasks differ in their nature and recommend different competences (excluding the management of "quantitative" tasks, where the number of available actors, but not their competences, is the key of solution).

KOWIEN defines two measures to give feeling (i.e. not in terms of mathematical formula) about the usability and quality of competency management: The effectiveness and the efficiency. The "effectiveness" of a competence management is determined by the degree of success reached while performing the identification and allocation steps. Both the effectiveness and the required efforts (made by the administration personal) decide about the "efficiency" of the competence management.

In the context of complex knowledge intensive engineering tasks, the consortium of KOWIEN argues that high effectiveness (and efficiency) is only to be achieved through the distribution of the engineering tasks on "partially autonomous networked actors". Hierarchical tasks allocation (by a central instance) would not take the specific competences and/or wishes of the employees into account. On the other side, competitive coordination between enterprises on the market would not lead to solutions for complex tasks, which cannot be split to smaller tasks. Therefore, the suggestion of KOWIEN is to adopt a hybrid coordination form "between the market and hierarchy". The autonomy of the actors reflects the fact that each actor is free while performing its task, without hierarchical orders. At the same time, the autonomy of actors is restricted so that their whole efforts lead to the fulfilment of a complex engineering task.

KOWIEN extends the interpretation of "competent actors" beyond the simple form of employees. In a multi-agent system the agents can be also "competent actors". A group of individual actors build also one competent actor, in the form of an enterprise for example.

KOWIEN authors know from experience that the application of competence management systems faces some difficulties, especially between separate enterprises where every enterprise has to put the competence profiles of its employees in the hands of other enterprises. The actualisation of the competency profiles may be also a real obstacle in the practice. But KOWIEN treats especially the main obstacle arising because of the usage of different languages and different languages terms between the enterprises. The problem takes place also between different departments of one enterprise. [ZAADW05] shows that the deployment of different software systems by the management personal has lead to the adoption of enterprise specific (or branch specific) terms. For these reasons, "synonym" and "homonym" terms hinder the application of competence management systems.

KOWIEN refers to the similarity of competence management problems to those found by the World Wide Web, and argues that the semantic dimension, as suggested by the Semantic Web (see section 2.4), would be a suitable solution. Therefore, ontologies build the kern of KOWIEN. The KOWIEN ontology consists of a top-level ontology and application specific ontology. The KOWIEN top-level ontology contains top-level concepts like entities, actor, personal competence, social competence, know-how competence, know-what competence, time point, time ranges, etc; in addition to relations like: works_for; has_address; has_competence_profile; contains_competence etc. The application specific ontology

contains for example a set of IT-competences, foreign languages' competences, planning methods competences, investment methods competences, analyse methods competences, presentation methods competences, etc.

On the base of the KOWIEN ontology, inference rules are suggested for two reasons: To discover the implicit knowledge included in the explicitly given ontology, and to check the consistency [YA03]. As an example of the first type, an inference rule $R_{Inference}$ says that a person *P* is competent in a specific theme *T* if she/he composed a document about a project, which is related to the theme *T*. An "integrity rule" $R_{Integrity}$ would then deactivate the inference rule $R_{Inference}$ if a fact exists already and it says that person *P* is not competent in the theme *T*.

KOWIEN deals with the question of how to develop the competence ontology and suggested a procedure consisting of multiple phases: Requirements specification, knowledge acquisition, conceptualization, implementation and evaluation of the ontology. Comma Soft AG, a member of the KOWIEN consortium, implemented a prototype of an ontology-based competence management system. The prototype has been deployed by other members.

To find the optimal allocation of engineering tasks to the actors (employees), KOWIEN adopts the "Goal Programming" solution. The Goal Programming is one of the Operations Research methods, designed to solve decision making problems under multiple criteria. In contrast to other methods, goal programming enables the decision maker to take multiple goals into account. KOWIEN makes use of this possibility and tries not only to "find the best employee to achieve a specific engineering task", but also to consider the preferences of the employees, i.e. "to find the best engineering task for each employee".

Each goal is a (mathematically expressed) composition of multiple (weighted) "deviations" on each criterion. The deviation on a criterion is the difference between a (predefined) target value on the criterion and the actual value (performance of alternative on the criterion). This reflects one of the important characteristics of the goal programming method.

The weighting of criteria (for each goal) is done according to the Analytic Hierarchy Process AHP (see section 2.5.3.2). In this context, KOWIEN recognizes that the weighting hierarchy of criteria (competences) can be found in the competence ontology. To this reason, KOWIEN stresses the importance of ontologies, not only as a base to incorporate intelligent solutions, but also as an interface to models of decision making problems.

2.6.5 Summary

This section deals with approaches trying to make use of both ontologies and methods of Multi-Criteria Decision Making.

Initial approaches (2.6.2) discovered the benefits of the explicit design of ontologies to clear the decision making problem (first step in the generic MCDM procedure. See section 2.5.3).

Other approaches (2.6.1) extend the semantic queries to consider soft constraints (user preferences), which have a comparable effect to the utility assessment (the third step in the generic MCDM procedure). The criteria are supposed to have equal importance values, i.e. the weighting of criteria (the second step of the generic MCDM procedure) is not considered in this context. The search using the preference queries gives out a set of "optimal candidates". Still, the selection of a single best candidate (the fourth step of the generic MCDM procedure) is taken out by the human user.

Approaches from the area of competence management (2.6.3 and specially 2.6.4) go a step further towards adopting both ontologies (and semantic matching) and MCDM principles. However, these approaches didn't try to integrate the semantic matching and the MCDM

methods and treated them as two separated processes. This applies for the design time, where the weighting of criteria and the utility assessment are supposed to be performed separately from the ontology development, and for the runtime, where the semantic matching and the decision making are designed to be performed in different and totally separated tools. Moreover, the semantic matching and the MCDM are performed in a central unit.

2.7. Automotive communication platforms

This section reviews the communication platforms in modern automotive systems. However, this section is not written to serve as a comprehensive review about the automotive communication platforms. But it delivers sufficient knowledge to evaluate the communication platforms from the Organic Computing point of view. Section 4.6.1 will define a set of requirements on the communication platforms, sections 4.6.2 to 4.6.5 will then evaluate each communication platform according to the requirements, and section 4.6.2.8 concludes with a comparison between the communication platforms.

The CAN Bus will be investigated in more details, because it is the most widespread serial bus system in the automotive industry (for the weaknesses of CAN bus, solutions will be suggested and evaluated in section 4.6.2).

2.7.1 CAN BUS

CAN (Controller Area Network) is characterized by the following features:

- It supports a multi-master hierarchy: To allow building of intelligent and redundant systems. The failure of one network node will not affect the operation of other nodes on the network [CAN08].
- Broadcast communication: A sender transmits information to all devices on the bus. The message is addressed to a node with identification code (ID), which matches the ID field in the message.
- Sophisticated error detecting mechanisms and re-transmission of faulty messages: The CAN bus error rate can be kept below 4.7x10-11 [CAN08]
- Transfer speed related to distance: For bit rates of 5KBit/sec CAN bus length can reach 10km, while the maximum length is restricted to 40m for the maximum bit rate of 1MBit/sec [ICP04].
- Arbitration: To lose conflicts arising when two or more nodes start transmitting messages at the same time, CAN bus adopts an arbitration mechanism on bits level. During the arbitration, every sending device compares the expected voltage level for its own transmitted bit with the actual voltage level monitored on the bus. If these levels are equal, the node will continue to send a following bit. When the expected level differs from the monitored level, the device has lost the arbitration and must stop its activity as a sender [ICP04].
- Real time transfer of messages: This feature is restricted to high priority messages with maximal length of 8 bytes.

CAN network can be configured to work with one of two different frame formats: The standard base frame format (CAN 2.0 A), or the extended frame format (CAN 2.0 B). These formats differ between the lengths of the identifiers: 11 bits in the base frame" and 29 bits in the extended format. These 29 bits are made up of the 11-bit identifier "base identifier" and a 18-bit identifier extension [Wik08b]. The standard format is widely deployed in the automotive systems. The extended format plays a role in conjunction with some higher level protocols such as SAE J1939 [May06].

On the base of the CAN Bus, many protocols have been developed to cover tasks such as flow control, device addressing, and transportation of long data blocks. Example protocols are: “DeviceNet”, “CANopen”, “SDS”, “CANaerospace”, “J1939”, “NMEA 2000”, “CAN Kingdom” and “SafetyBUS p” [Wik08b]. SAE J1939 is the vehicle bus standard used for communication and diagnostics, originally by the heavy duty truck industry in the United States. J1939 is used for communication in the engine compartment and between the tractor and trailer [Wik08b], while CANopen is preferred for body management, such as lights and locks [Mur03].

In order to send long messages (more than 8 bytes) CANopen offers a segmented transferring of the Service Data Objects (SDO). The model for the SDO communication is the Client/Server model [DCS99].

The segmented SDO transfer is combined with high overhead because of the confirmation of received segments. CANopen supports the mode of block transfer to avoid this overhead. A block consists of (a known number of) segments (maximally 127 segments). One confirmation message is sent for every block and better efficiency values can be accordingly achieved.

The next sections contain a study about the efficiency of data transfer on the plain CAN bus, the efficiency of CANopen segmented transfer and of block transfer.

2.7.1.1 The efficiency of data transfer on the plain CAN Bus

The efficiency of data transfer over the bus system is calculated as the ratio between *n*, the number of application’s data bytes, and the number of transferred bytes (*n* + overhead).

$$\text{CAN Bus Frame Efficiency} = \frac{n}{\text{Frame length}}$$

The base frame format is characterized by its 11 bit message identifier within 44 overhead bits [Wik08b]. Consequently, the efficiency of the base frame format:

$$\text{CAN Bus (Base) Frame Efficiency} = \frac{n}{44/8 + n}$$

In the extended frame format the message identifier occupies 29 bits within 64 overhead bits [Wik08b], and its efficiency can be calculated as:

$$\text{CAN Bus (Extended) Frame Efficiency} = \frac{n}{64/8 + n} = \frac{n}{8 + n}$$

The last relations are true only for a single frame, which can transfer a maximum number of data byte of 8 ($n <= 8$).

The efficiency of data transfer on the CAN Bus itself reaches its maximum value when sending 8 data bytes in one frame:

$$\text{Maximum CAN Bus Efficiency (Base Frame)} = \frac{n}{44/8 + n} = \frac{8}{44/8 + 8} = 0.59$$

$$\text{Maximum CAN Bus Efficiency (Extended Frame)} = \frac{n}{8 + n} = \frac{8}{16} = 0.50$$

This means that any higher level protocol will never reach an efficiency value better than 0,59 (for the base frame format) or 0,5 (for the extended frame format).

To get a feeling about the transfer speed, the transfer time of 8 data bytes will be calculated using the maximum baud rate of CAN Bus 1 Mbit/Sec. The base frame format would have the length of: 44/8 + 8 = 13,5 bytes. The transfer time of the frame can be calculated as:

(13,5* 8)/10 exp.+6 = 0,108 ms[3].

The transfer time of 8 data bytes in the extended frame frame format is then 0,128 ms.

For a message of 1KB (1024 ASCI characters), 128 CAN Bus frames have to be at least sent. The transfer time reaches 13,824 ms for base frame format and of 16,384 ms for the extended frame format.

However, the transfer of long messages is ruled by the segmentation protocol, which adds more overhead. To this reason the transfer time of a 1KB message is only calculated to be a theoretical reference value. The actual transfer time is to be taken from the next sections, which consider the segmentation overhead.

2.7.1.2 Efficiency of CANopen for segmented transfer

For a segmented SDO transfer of *n* data bytes, the total number of needed segments is calculated as:

$$\text{Number of segments} = n/7 + k$$

Where:

$k = 1$ when n mod 7 = 0 (1 segment is needed to initialize the data transfer) or

$k = 2$ when n mod 7 is not equal to 0

Consequently the total number of messages is:

$$\text{Number of messages} = (n/7 + k)*2$$

The total CANopen efficiency of segmented SDO transfer over CAN Bus can be calculated as [Zel01]:

$E = n / [(g+k)*16]$

Where:

n number of data bytes (length of data in bytes)

g = n/7; i.e. one segmented messages contains max. 7 data bytes

k =1 when n mod 7 = 0

k = 2 when n mod 7 ≠ 0

Table 2-3 gives the efficiency values of segmented SDO transfer for different numbers of data bytes.

Number of data bytes (n)	**CANopen efficiency with segmented SDO transfer**
7	0,219
8	0,159
64	0,356
256	0,415
1024	0,431

[3] Message prioritisation and transfer faults may cause unpredictable delays, which are not part of the transfer time of the message itself, and are not considered in the relation.

Table 2-3 Efficiency of CANopen with segmented SDO for different numbers of data bytes

Example:

For n = 1024, the number of CAN Bus messages is: (1024/7+2) * 2 = 296,571, with transfer time of 296,571 * 0,108 = 32,03 ms for the base frame format, and of 296,571 * 0,128 = 37,961 ms for the extended frame format[4].

2.7.1.3 Efficiency of CANopen for block transfer

According to [Zel01], the efficiency of transferring 127 segments in each block can be calculated as:

$E = n / [(g+h+i+5) * 8]$ when $n \bmod 7 \neq 0$

$E = n / [(g+h+i+4) * 8]$ when $n \bmod 7 = 0$

Where:

n: number of data bytes to be transferred

g = n/7 total number of data transfer segments

h = n/890 = n / (127 * 7) number of blocks

i = 0 for write access to the server

i = 1 for read access to the server

Table 2-4 shows the efficiency values of block transfer in CANopen for different lengths of data:

n: Number of data bytes	CANopen efficiency with block transfer
7	0,174
8	0,163
64	0,565
256	0,764
1024	0,840

Table 2-4 the efficiency values of block transfer in CANopen for different lengths of data

Example:

For n = 1024 data bytes the block transfer time is 13,824/0,840 = 16,457 ms for the base frame format and 16,384/0,840 = 19,505 ms for the extended frame format.

2.7.2 Local Interconnect Network LIN

The LIN bus is a slow network system (max. 20 kbaud), used to build sub-networks from the CAN bus network. Nodes in the LIN network are usually intelligent sensors and actuators [Wik08].

2.7.3 FlexRay

Along with the increase of the amount of electronics in the automobile and the introduction of several advances in safety, reliability and comfort, new and higher requirements on the communication platform appear continuously. These requirements include the combination of higher data rates to send the increasing number of control and status signals, deterministic

[4] The same values can be calculated as: transfer time of 1024 bytes on CAN Bus without segmentation overhead / efficiency of CANopen for 1024 data bytes.

behavior and support of failure tolerance. Efforts on developing a serial bus system according to these requirements evolved to the foundation of the FlexRay consortium, which combines mainly distinguished motor vehicle manufacturers and chip producers. The consortium [Fle06] introduced a new bus system as "a communication system that will support the needs of future in-car control applications. At the core of the FlexRay system there is the FlexRay communications protocol. The protocol provides flexibility and determinism by combining a scalable static and dynamic message transmission, incorporating the advantages of familiar synchronous and asynchronous protocols".

The determinism in FlexRay is a result of time-triggered communication (in contrast to the event-driven CAN bus). All nodes must conform to a precisely defined communication cycle that allocates a specific time slot to each FlexRay message (Time Division Multiple Access - TDMA) and therefore prescribes the send times of all FlexRay messages [May07a].

The flexibility of FlexRay results from its ability to send dynamic messages in the second part of each frame. While the static messages are transmitted in each message with guaranteed message latency, and therefore used typically for real-time communication, the dynamic messages are only transmitted when required and used to transmit diagnostic data, for example.

FlexRay protocol supports also features like:

- Fault-tolerant clock synchronization via a global time base: correction factors are calculated in each node after the reception of Network Idle Time NIT segment (NIT comes at the end of each cycle).
- Collision-free bus access: each Node/message pair has a pre-allocated slot.
- Single channel gross data rate of 10 Mbit/sec.
- Scalable system fault-tolerance via an additional (redundant) communication channel: to minimize the failure risk, FlexRay offers redundant layout of communication. Optionally the redundant channel can be used to increase (duplicate) the transfer data rate to 20 Mbit/sec.
- Support of different network topologies like:
    - Simple passive bus structure: where the nodes act solely as listeners (not as repeaters)
    - Star topology: offering the possibility of disconnecting faulty communication branches or FlexRay nodes. This topology suffers from the disadvantage of having a single point of failure (the central device)
    - A combination of both topologies.

2.7.4 Media Oriented Systems Transport MOST

MOST is a networking standard designed for connecting multimedia devices in automobiles. It is featured with higher bit rates than other bus systems. The total bit rate of MOST25 (the first step of the protocol specification) reaches about 22 Mbit/Sec, divided into 60 channels in each frame. MOST150 promises a bit rate of 150 Mbit/Sec [May07b].

According to the device profile, several channels can be reserved for one device. The reservation of channels takes place in the initialization phase of the connection (Plug-and-Play). A central channel master adopts the channel s' management and the synchronization of data transfer devices on the bus. The central synchronization enables the involvement of relatively simple devices to the bus.

As a synchronized bus with high bit rates, MOST offers an optimal solution for data streaming (multimedia) through its channels (guaranteed bandwidth with no need for buffering). But it supports also real-time transfer of any type of data, like control data [Mos08].

MOST covers all seven levels of the Open Systems Interconnect (OSI) reference model. The physical layer is realized using unshielded twisted pair (UTP) or optical fiber [Muy08]. On the application layer, predefined “functions” are gathered to build a “function block”. Each function has a set of operations (like set, get, etc.). This order of functions and function blocks is to be understood as a logical view on the system, because a function block can be located in reality on any device of the MOST bus. The management of this all is done by the central master, which has a registry of available functions and function blocks, along with their corresponding addresses.

3. Semantic Multi-Criteria Decision Making

3.1. Critique of current approaches

3.1.1 DySCAS

DySCAS addresses the reconfiguration (or self-configuration) of software tasks which share a standardized middleware like AUTOSAR. DySCAS finds its roots in the Grid Computing and considers the problem as an allocation problem of processing, storage and communication resources. The diversity of devices and features of automobile system makes such reduction impracticable.

The DySCAS solution is designed on the base of two mechanisms: Policies and utility functions.

The matching of the current context parameters, as well as their values, to those found in the policies (rules) is a syntactical matching. This means that tools are required to assure coherent naming of context parameters. Aside from the fact, that no policy design tools are yet available for AGILE, the policy-based computing cannot face the challenge of integrating resources (devices and/or software components) developed by different providers. This challenge triggered the engineering efforts like AUTOSAR (standardization on the implementation level), EvoArch (autonomy), as well as the architecture and design methodology presented in this thesis. Just because of this shortcoming, DySCAS is expected to be an isolated idea which will not find its applications in reality.

Design tools to set up the weights in the utility functions are also still missing. Before presenting a solution to define weights statically, dynamic weighting cannot be practically used (imaginable solutions on the base of “learning” would ruin the relative advantage of a lightweight policy-based computing).

Moreover, the relatively high degree of freedom available through the use of policy-based computing, should be balanced by a specific design schema; otherwise the rules could resolve to “no actions” or a “lot of actions”.

Generally, the suggestion of such special runtime architectures has to be followed by the introduction of suitable design tools.

In addition to the design difficulties, the utility functions in DySCAS are still thought of as mathematical formulas, where the values of context parameters have to be numeric. Utility functions in DySCAS cannot cope with a textual (qualitative) description of components (software components or other components). For example, there is no way to express user preferences like: “Video streaming is better than audio streaming”.

The policy expression language and integration platform AGILE is implemented as a .NET application, showing that DySCAS is not yet ready to take its place in embedded systems (due to limited resources).

3.1.2 EvoArch

The main contribution of EvoArch is the introduction of the market place arena, as well as the unified taxonomy as two corners of the evolutionary architecture. Each ECU represents an autonomous unit.

EvoArch adopts a marketplace-oriented behavior of the autonomous units, but the question about the most suitable order of actions on the marketplace is still open. Different forms of behavior, still on a marketplace, should by thought of and put up to discussion.

The T-Selection in its suggested form may be an adequate solution when long candidate lists have to be restricted to a shorter one. But an important problem will arise when the T-Selection cannot limit the candidate list to one specific candidate, i.e. when more than one candidate can be proofed as acceptable. Such situation is more than a theoretically thinkable one. The evolutionary architecture comes into consideration where engineering systems register a high level of complexity. In such environments, different acceptable partners can be surely available at the same time. The suggested concept overlooks this problem completely. In connection to this problem, some other questions deserve closer attention, like the question of description granularity: "How much information should be available about each autonomous unit?". In other words: Too precise descriptions lead to small success rates when searching for partners, while too coarse descriptions result in a large number of candidate partners.

The designers of EvoArch used to look at the problem from the point of view of a software developer. This explains the usage of principles and tools of the Unified Modelling Language UML [OMG08]. UML is originally designed as a supporting language for the software development. The similarity of the object-oriented modelling and the taxonomy inspires the authors to adopt the UML class diagrams to develop taxonomy graphs, to allocate autonomous units on it, and finally to develop a software to run on autonomous units (prototype in Java). This generalization of the UML usage is thought of as a simplification of the development process, but it hides a dangerous contradiction between the aims of software designers, developers and testers, and the expected benefits of taxonomy graphs. For example, it is not important for the software developer to know if a lamp controller will be set on the right side or on the left side of the car. At the same time, such "classification" includes vital information for the autonomous unit and its location on the taxonomy graph.

A main drawback of EvoArch can be registered because of the expectation that all designers and all manufacturers share a common taxonomy. Solely to move the integration problem from the designers' hands to the "autonomous units" does not seem to be a realistic solution. The autonomous units have to be supported with adequate knowledge about themselves and about other units. They need a common language, just like the human designers. It is not known how different manufacturers would be able to transfer a common language to their units, before all designers agree and formulate such language. It would be a great challenge to persuade the manufacturers to spend time developing a common taxonomy of all units constituting a modern car. The development of such a taxonomy internally in each manufacturer/vendor is only a first step. The next step has to consider the integration of taxonomies made by different manufacturers, coming from different countries with different natural languages, experiences and expectations. Such a process would be the most difficult problem, and maybe an unsolvable one.

A further problem would be the validation of the taxonomy. Validation tools are indispensable in such cases. Moreover, continuous advances in the technology mean that the taxonomy will be always out of date, and therefore, unusable by the autonomous units. The costly actualization of the taxonomy would be a permanent problem. From this point of view, the original integration problem gets new dimensions, related on the one hand to knowledge creation, integration and validation in the design time and, on the other hand, to knowledge utilization in the run time (by the autonomous units). In both phases, the authors do not suggest convincing solutions. So, EvoArch relies on dangerous and not sufficient syntactic matching between (names of) taxonomy nodes in the absence of adequate design, integration, validation and execution tools.

Table 3-1 summarizes the features and differences of DySCAS and EvoArch.

	DySCAS	EvoArch
Components	Software tasks	ECUs
Behavior	Application dependent	Marketplace oriented
Enabling technologies	- Policy-based computing - Selection through utility functions (only for quantitative values)	- Taxonomy selection (coarse selection) - No fine selection
Matching type	Syntactic matching between parameters (context parameters and rules parameters)	Syntactic matching between taxonomy nodes
Efforts towards a common description language	No	Yes
Development\Deployment support of a common language	No	No
Development tools	- Policy editor (under construction) - No tools to set up the utility functions	UML editors
Implementation	A prototype on .NET	A prototype in Java

Table 3-1 A comparison between DySCAS and EvoArch.

3.1.3 Approaches of Contract Net Protocol

The Contract Net Protocol (CNET Protocol), shown in section 2.3, represents a special approach for distributed problem-solving on the base of a marketplace-oriented behavior. However, the roles of the manager and of the contractors are statically defined: The manager sends a call for proposals (enquiry), the contractor answers it with a bid (offer) if it can fulfil the eligibility specifications, and the manager must select a bid from a list of available bids. Although it seems to be a "natural" protocol, other types of roles' assignment are principally possible. Such possibilities are not considered by any approach in the research field of Contract Net Protocol.

The Contract Net Protocol ignores also the question of how to construct a common language (between the managers and the contractors) and doesn't provide a solution for the selection problem of the satisfactory bidder. The nature of CNET applications explains this situation. Although the applications are found in distributed problem solving and resource allocation in relatively open environments, the distributed tasks, their restriction to specific known types, and the high similarities of the agents make the impression that a statically defined types would be a sufficient common language. Also following approaches about the CNET protocol ignore these questions and consider the Contract Net Protocol as communication protocol, deal with its modelling, or try to find new application areas (see [FIPA02], [JP02], [XW01] and [PCJ04]).

The approach of "Ontology-based Services for Agents Interoperability" [Mal06] gathers the marketplace-oriented behavior (Contract Net Protocol) to the ontology-based matching capability. However, the marketplace-oriented behavior is the default behavior in the application field of Business-to-Business (B2B). Therefore, the concentration of this approach goes to the ontology-mapping between different ontologies (the ontology mapping can be always required in distributed application environments like B2B or the Semantic Web), rather than the systematic integration of both technologies: Marketplace-oriented behavior and the ontology-based matching. In other words: In this approach the marketplace-oriented behavior is one of the realities found in the B2B application field and is not a suggested solution for any engineering problem (like the system complexity).

Moreover, the approach doesn't address the selection problem of the best available bid on the base of multiple attribute. It adopts also the traditional CNET Protocol without to study any other possibilities of roles distribution (the role of the manager and role of the contractor).

On the other side, the approach of "Extended Contract Net Protocol" (ECNET) oversees the interoperability problem (i.e. the need for a common language), but it adopts a method of MCDM (TOPSIS). ECNET doesn't give any justification for the usage of TOPSIS, while many other MCDM methods come into question for the same purpose. Moreover, ECNET doesn't address the question of providing suitable design tools of the decision making manager and adopts the traditional roles distribution.

3.2. The basic idea of Semantic Multi-Criteria Decision Making SeMCDM

Self-organization and autonomy have been already proposed to face the increasing complexity of technical systems. However, the shortcomings recognized in current approaches (EvoArch as well as DySCAS) push towards new ways of thinking and towards the search for new methods.

Similar to EvoArch, the architecture of Semantic Multi-Criteria Decision Making (SeMCDM) considers "parts" of the system as autonomous units, which try to (re)configure the whole system through marketplace-oriented behavior: The exchange of offers and enquiries. However, SeMCDM relies on a new vision and understanding of the self-organization in engineering systems.

SeMCDM suggests that autonomous individuals would be only able to build a "*suitable order*" if they are supplied with "*suitable knowledge*". A common view to the world between the designer and the autonomous units, as well as between the autonomous units themselves, is here the key of the solution. The problem is redefined as a problem of knowledge and its deployment to reach the expected behavior.

Through the integration of a "knowledge dimension", SeMCDM finds its counterpart in the "Semantic Web". This goes back to the similar nature of addressed questions: The Semantic Web aims to allow data to be shared and processed by automated tools as well as by people. As an extension of the current web, information of the Semantic Web is given a well-defined meaning, enabling computers and people to work in cooperation. Exactly this cooperation between machines and people as well as between the machines themselves builds the key solution for the self-organization problem in the design process and later at run-time.

The Semantic Web deploys the idea of knowledge formulation in "*ontologies*", as a common and machine readable view to the application domain. Effective methods of data discovery, integration and reuse have been declared as key technologies towards the Semantic Web. The incorporation of inference mechanisms enables the querying of the ontology, the explicitation of the implicit knowledge and the consistency check of ontologies. Moreover, inference rules can be deployed to get more (domain/application specific) knowledge.

SeMCDM benefits from available ideas, experiences and tools found in the world of Semantic Web to enable self-organization in engineering systems. On the base of this new understanding, SeMCDM suggests to build the knowledge of autonomous units on the base of an ontology, which represents an important bridge between the designer(s) and the autonomous unit(s). The autonomous units would be able to interact using semantically well-defined terms. Interaction models, like the market place arena would benefit from the "*semantic matching*" between the offers and the enquiries on the marketplace.

Moreover, SeMCDM addresses a further problem in relation to *multi-dimensional descriptions* of both offers and enquiries. In practical cases, it is highly expected that the autonomous units would be described with the help of *multiple features*. While the semantic

matching presents a solution for matching of features' pairs (one feature of the offer and one feature of the enquiry), the evaluation of the whole offer, with all of its features, moves the matching problem to a higher level of complexity. The analogy to problems of making decision under multiple criteria provides an incentive to a deeper study of mechanisms and methods known in the field of Operations Research. The adoption of "Multi-Criteria Decision Making" (MCDM) methods in SeMCDM is the proposed solution to support the autonomous units with *selection mechanisms*.

Figure 3-1 summarizes the methodologies integrated within SeMCDM.

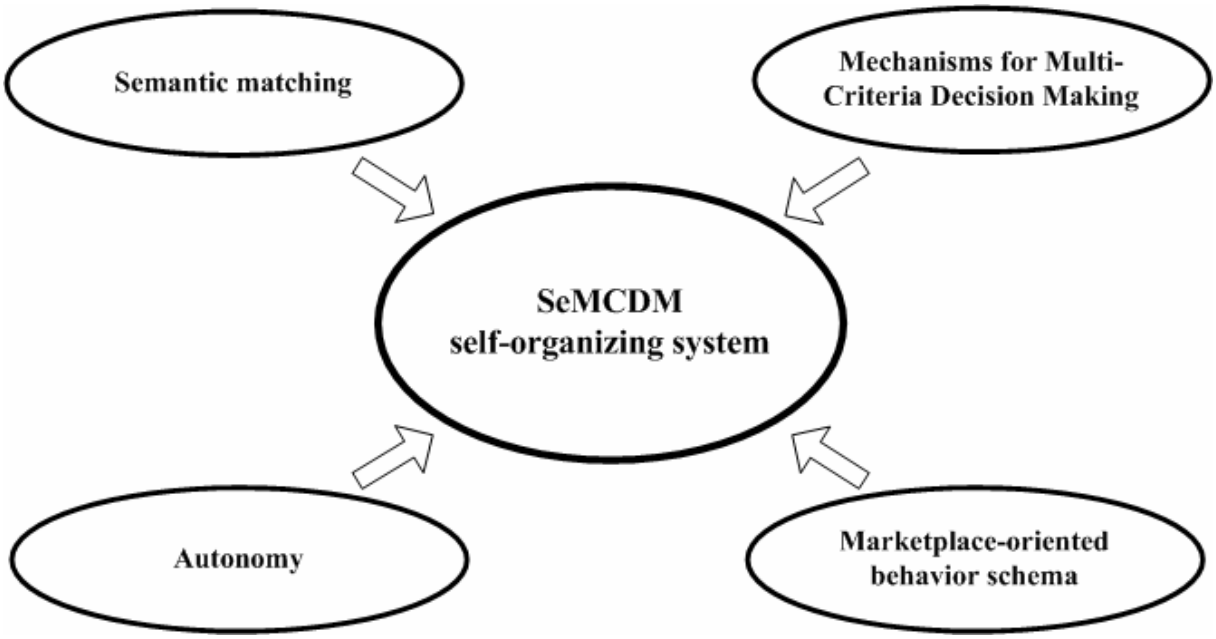


Figure 3-1 The main contribution of this thesis is the integration of concepts originating from different research areas into a practically usable methodology.

Figure 3-2 shows the basic idea of SeMCDM from the technical point of view. As in EvoArch (section 2.2.3), autonomous units are divided basically into active and passive units.

Active units are expected to be supplied with a wish list: A list of wished components with their specified features.

To give the description (and successively the autonomy) more flexibility, the features have been divided into *hard features* (exclusion features) and *soft features* (optimization features). For each component the active unit sends an enquiry to all passive units.

A passive unit tries to answer the question about its adequacy to meet the hard features in the enquiry. Features of the enquiry have to be semantically matched to self-description features of the passive unit. For this purpose, the passive unit relies on the common language (the ontology) and on a mechanism to answer enquiries (inference). The inference engine can be supported with additional rules to reach the real relation between features (implicit knowledge).

As soon as the hard features have been proven as available, the passive unit sends an offer to the active unit. The active unit receives eventually more than one offer for one specific enquiry. Therefore it carries out a kind of Multi-Criteria Decision Making. After making a decision to accept a specific offer, the corresponding passive unit is informed about the decision and a contract is made between both units.

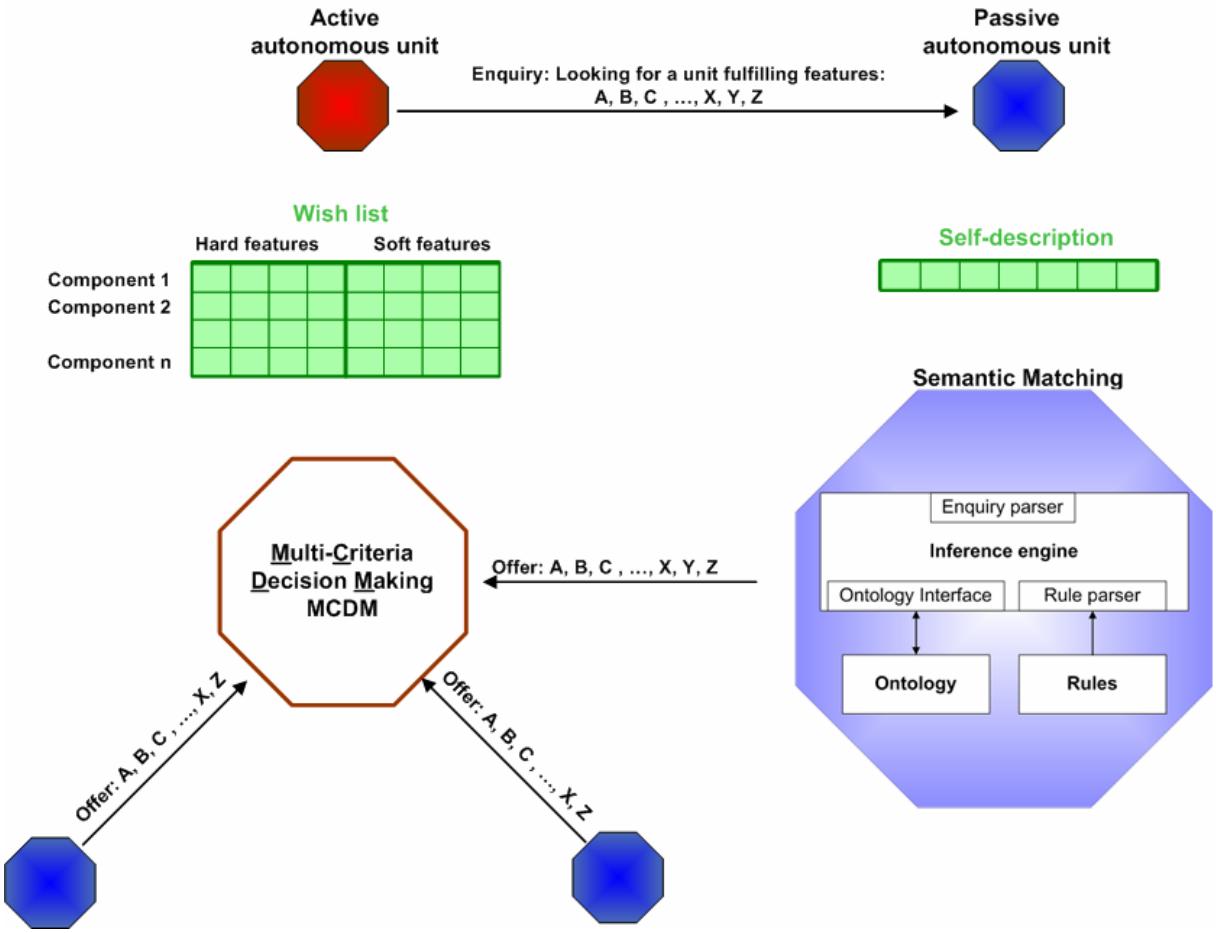


Figure 3-2 Semantic Multi-Criteria Decision Making: From the technical point of view.

A system configuration is built this way as the sum of all concluded contracts. The same process can be repeated on different levels of the system: The active unit can play the passive role as soon as it could find suitable offers for its components. Such units play a so called "passive-active role". A hierarchical configuration takes place successively.

3.3. Advantages of SeMCDM

SeMCDM discovers the knowledge-related dimension of self-organization. Knowledge representation in ontologies promises a formal conceptualization of the application domain. Research approaches from the field of Semantic Web provide supporting tools to develop the ontologies as a common language between the designers. Already available mechanisms for consistency check, extension and integration of ontologies developed by different parties are of vital importance in this case.

The same ontology will be processed by the autonomous units at run-time. The autonomous units share a common language and can "understand" the needs and competencies of each other. The semantic matching is one of the most important advantages of SeMCDM.

Application specific rules add more intelligence to the autonomous units and enable the designers to express their knowledge.

The designers' preferences are represented in a clear description schema (the wish list). Description of components through two categories of features (hard and soft features) promises more flexibility while expressing the designers' preferences. At run-time, the autonomous units make use of the flexible description, and discover sub-optimal solutions.

Methods of Multi-Criteria Decision Making extend the capabilities of the autonomous units to the selection of the best available offer and represent therefore an optimization of the self-organization. Moreover, with the adoption of multi-criteria decision making a new perspective has been added to the preference expression. Detailed preference acquisition enables better correlation between the view of the designer and the view of the autonomous units. The autonomous units will make their decisions in the sense of the designer needs. SeMCDM combines the autonomy of the units composing the engineering systems with designers' restrictions and opens the way for flexible and reasonable self-organizing systems.

3.4. Design issues and open questions

The field of MCDM is characterized by the existence of different methods to solve the selection problems (see section 2.5.2). The adoption of a specific method should consider its applicability for purposes of SeMCDM. This applies for all four steps in the decision making process. This issue is the theme of section 4.1.

The semantic matching between features and the Multi-Criteria Decision Making cannot be considered really as two separated steps. The semantic matching of two features decides basically about the comparability of these features from the semantic point of view, but it should tell more about the degree of fulfilment (utility). The combination of qualitative features and quantitative features presents a new challenge for the inference engine. SeMCDM shows the need for a MCDM capable inference. Three sections address this issue: Sections 4.2 present SeMCDM supporting ontologies, section 4.3 specifies an MCDM capable inference engine, and section 4.4 considers the flexibility of the wish list to define a generalized matching process.

The integration of MCDM and inference mechanisms at run-time also corresponds with a similar issue at design time. Two different worlds have to be integrated: Ontology design and the MCDM supporting design. In the light of the selected MCDM methods, a design concept of supporting tools should be developed to permit this integration. The design support in SeMCDM is the theme of section 4.7.

According to EvoArch and to the basic idea of SeMCDM, the actions carried out by the autonomous units build only a simplified analogy to marketplace-oriented behavior. The actions on the marketplace vary between sending an enquiry/offer, matching enquiries to offers, and making decisions. Before shifting such ideas to real engineering systems, concrete orders of actions have to be developed. A concrete sequence of action builds a "market scenario". A categorized set of possible market scenarios is defined in section 4.5. Different factors of the application environment may push towards specific scenarios, as the simulation results show in section 5.5.

Automotive applications differ in their complexity and requirements. Although characterized as embedded systems with real time support, some devices in modern automotive systems possess high computation power. The selection of suitable applications in automotive systems and more information about the requirements of SeMCDM may encourage designers to consider self-organization as a real alternative. Application fields of SeMCDM in automotive systems have been recognized in section 4.6.

The specified SeMCDM methodology has been evaluated in section 5.

4. Design of SeMCDM

The basic concepts of SeMCDM have been defined in section 3.2. This section answers the design questions found in section 3.4. A step by step design will take place until SeMCDM takes its well-specified form. The design of SeMCDM is made principally through a deep requirements analysis and investigation of available solutions.

4.1. Multi-Criteria Decision Making for autonomous systems

Section 3.2 presented the selection problem facing the autonomous units. It made also clear, that the selection problem is a type of decision making under multiple criteria.

In this section, terms from the MCDM research field are gathered and mapped to their corresponding terms from the suggested architecture of autonomous units.

Recalling the basic idea of the autonomous behavior, the features of the passive autonomous units conform to an "offer" and the "selection" has to take place between offers. Therefore, the offers are "alternatives", from the MCDM point of view.

The features of the enquiries (for each component) are to be understood as the "criteria" for the decision making problem.

Because the decision has to be made autonomously, the active autonomous units are the real "decision makers" on the market place (section 4.5 presents a detailed discussion about the possible roles of the units).

Still, a "human designer" carries out the first three steps of the decision making procedure (section 2.5.3) at design time: Features' analysis, utility assessment of features' values, and features' weighting. Additionally, the designer of passive units prepares his units, by setting up the features and their values.

For this reason, there is a difference between the "design phase" and the "run-time phase". Hence the following discussion distinguishes the "designer" from the "decision maker".

In the following section the requirements on the MCDM methods will be described. The following sections apply the requirements on the available methods and filter out the inadequate methods. This takes place separately for each step of the MCDM procedure (section 2.5.3).

4.1.1 Requirements

To select the adequate MCDM method for each MCDM step, the intended SeMCDM selection problem has been analysed to discover its requirements. The following requirements list has emerged as a result:

- **Finite number of discrete alternatives**: As the target application system consists of a "finite" number of autonomous units, the decision making problem considers only a finite number of discrete alternatives.
- **Stable and known performance values of alternatives on criteria**: The autonomous units are supposed to offer stable and known performance values on all criteria. It is also supposed that the designers know and declare the features of their autonomous units. (The topic of trustworthy designer knowledge is beyond the scope of this dissertation.) The integration of autonomous units with dynamic features, i.e. autonomous units with changing performance values on some criteria, is principally possible, if value changes trigger the reconfiguration process of the system. However,

the assumption of stable performance values of alternatives on all criteria is valid within each reconfiguration process.

- **Complete methods**: The aim of the decision making process is to select the best, and only one, available alternative. This means that ranking and decision making methods have to deliver a complete rank of alternatives. In the context of criteria weighting, the complete methods are those delivering weights for all criteria at hand.
- **General solution:** While weighting the criteria at design time, the designer is not able to predict the nature of alternatives offered later at runtime. Those weighting methods and utility assessment methods, which deliver alternatives' dependent results, are not applicable in this case.
- **Known criteria**: Other than the alternatives, the criteria (features) of the alternatives are supposed to be known at design time.
- **Uncertain designer preferences**: The fact that the designer has always uncertain preferences cannot be ignored by the weighting method.
- **Qualitative and quantitative criteria**: The criteria can be generally of any data type, i.e. qualitative or quantitative.
- **Adequacy to the components description schema**: The flexible schema in the wish list (see section 3.2) demands an adequate ranking function. The soft features build a typical case for the weighted sum method, whereas the hard features push towards a weighted product.
- **Help for the designer**: The decision making methods and tools are originally designed to help the human designer to clear the decision making problem at hand and to depict his preferences in a mathematical, or computerised, way. MCDM methods, which confront the designer with challenging questions, are not acceptable.
- **Ease of use**: The MCDM methods differ in their usability and user friendliness. This applies to the number of designer statements, and for the types of theses statements.
- **Integration within the ontology development process**: The integration of the ontology development process and the MCDM methods (features' analysis, utility assessment of features' values and features' weighting methods) is of high importance for the designer.

4.1.2 Selection of suitable decision making methods

In this section, the requirements from section 4.1.1 will be applied to MCDM methods (from section 2.5.3) in order to select the best ones for purposes of SeMCDM. Generally, unsuitable methods will be successively eliminated.

The first requirement makes it clear that the decision making problem considers only discrete alternatives. For this reason the decision making methods in continuous space can be excluded from this discussion, i.e. the Multi Objective Decision Making methods, and the solution falls under the branch called Multi-Criteria Decision Making.

Stable and known performance of offers on criteria (the second requirement in section 4.1.1) excludes the need for decision-making methods considering uncertainty (games against nature and games theory). Therefore, the following discussion will be limited to the methods shown in section 2.5.3. The following sections are organized to follow the 4 steps of the generic MCDM procedure (section 2.5.3).

4.1.2.1 Selecting suitable methods for the problem analysis step

Looking at the original decision making hierarchy, developed by Saaty for the Analytical Hierarchy Process AHP [GWH89], the similarity to the hierarchical nature of ontologies is very apparent. This similarity concerns especially the criteria hierarchy and the alternatives (instances) relations to the criteria. Therefore, the hierarchical analysis of the decision problem is by its nature the most adequate method to be integrated within the ontology development process.

However, the decision making hierarchy combines different criteria to clarify a specific decision making problem, while the ontology is a kind of general description of the world, or of a specific application domain.

This difference reflects rather different "ways of thinking" than real conflicts. Then "good domain ontologies" are originally designed to be highly reusable and are inherently a suitable base for a wide range of description problems, for example, as a source of criteria (features) and instances for a decision-making hierarchy.

On the other hand, designers of autonomous units can enrich the imported ontologies with new concepts, properties and instances and then share them with other designers. A common, formally well defined language, which enables consistency checking, turns the problem analysis in MCDM into a collaborative effort of different designers. These new possibilities are one of the major benefits of SeMCDM.

4.1.2.2 Selecting suitable methods for the step of criteria weighting

Table 4-1 shows the weighting methods (from section 2.5.3.2) in relation to the requirements described in section 4.1.1. Evaluated weighting methods are: The fixed point scoring methods; i.e. fixed point scoring (Hajkowicz et al. [12]) and fixed point scoring of hierarchically ordered criteria [10], the rating methods; i.e. rating method [12] and SMARTS rating of ranked criteria (Olson [13]), the ordinal ranking method [13], [12]), the preference cones method [10], SMART weighting method (steps 4-7 as in [13]), the multi-attribute value functions method [10] and the paired comparisons methods (AHP [9], normalization method and the geometric mean method [10]). These methods have been surveyed in section 2.5.3.2.

The criteria weighting is the most difficult step which faces the decision makers. Therefore, the amount of help delivered by the weighting method is the first requirement to apply when selecting a weighting method. Direct weighting methods like the fixed point scoring methods and the rating methods are, for this reason, unsuitable methods.

The SMART weighting method (steps 4-7 as in section 2.5.3.2) shows a major disadvantage concerning the ease of use, because its complexity increases dramatically with higher numbers of criteria. Therefore, SMART is a typical example for a generally good method, but with lack of user friendliness.

The preference cones method, the ordinal ranking method and the multi-attribute value functions method don't support uncertainty about the designer preferences. Additionally, the preference cones method suffers from being incomplete, while the multi-attribute value function method addresses only quantitative criteria. Weights resulting from the ordinal ranking method imply a kind of approximation errors, especially for small numbers of criteria.

Paired comparison methods can be applied if the criteria are known at design time, as expected in SeMCDM. The paired comparison methods meet all requirements in Table 4-1. Also they can especially deal with uncertainty about the designer preferences and support a kind of consistency check, like the Analytical Hierarchy Process AHP suggested by Saaty

[GWH89]. As all three paired comparison methods (shown in section 2.5.3) deliver similar results, the popular and theoretically founded AHP method is selected to weight the criteria in SeMCDM. For the practical implementation and integration of AHP in the ontology development process see section 4.7.

	Fixed point scoring methods	Rating methods	Ordinal ranking method	Preference cones	SMART	Multi-attribute value functions	Paired comparisons
Helpful methods	-	-	+	+	+	+	+
Uncertainty support	-	-	-	-	+	-	+
General solution	+	+	+	-	+	+	+
Qualitative & quantitative criteria	+	+	+	+	+	-	+
Completeness	+	+	+	-	+	+	+
Ease of use	+	+/~	+	-	-	~	+

Table 4-1 Comparison between weighting methods

4.1.2.3 Selecting suitable methods for the utility assessment of alternatives' performance on the features

As the performance values of alternatives on the criteria are stable and known, methods supporting uncertain information about the performance of alternatives (like Ariadne [13] and Hipre3+ [13]) can be excluded.

For the utility assessment of quantitative (numeric) features, three types of methods comes into question: The normalization methods, the paired comparison methods and the predefined utility functions. The results of the normalization methods (i.e. standard normalization, TOPSIS and ELECTRE methods) and the paired comparison methods (AHP, the geometric means method and the normalization technique) depend always on the considered alternatives. Only the predefined utility function methods (i.e. Edwards's linear interpolation adopted by SMART and the general case of utility functions) can meet the requirement of being a general solution and will be adopted for SeMCDM purposes. However, the designer has to take care that predefined utility functions deliver really a general solution (for example, by choosing the maximum and minimum values in the Edwards procedure).

The utility assessment of qualitative features can be achieved by paired comparison methods (alternatives dependent) or by the simple method of predefined scales, like the Likert scale. As possible qualitative values are predefined in the ontology, the predefined scales promise to be a general solution.

The utility assessment through predefined utility functions and predefined scales has been integrated in the ontology development process. The integration has been achieved by defining the SeMCDM ontology (see section 4.2) which supports these methods, and by developing a specific utility assessment tool within an ontology development environment (see its concept in details in section 4.7.2).

4.1.2.4 Selecting suitable methods for the ranking and decision making step

Table 4-2 shows the ranking and decision making methods in regard to relevant requirements (methods supporting uncertainty, like Ariadne and Hipre3+, have been already excluded).

Incomplete methods like the outranking methods and the preference cones are not adequate for purposes of autonomous decision making. PROMETHEE II is the sole exception of a complete outranking method, but it doesn't answer the question on how to specify preference functions in a reasonable and user friendly way. Moreover, PROMETHEE II implies the usage of the weighted sum formula, whereas no reason justifies the adoption of PROMETHEE II instead of the weighted sum method.

	Outranking methods	Preference cones	Data envelopment analysis	Weighted sum	Weighted product	Reference point methods
Completeness	-	-	+	+	+	+
General solution	-	-	-	+	+	-

Table 4-2 Ranking methods in terms of the concerning requirements

The weighted sum and the weighted product methods arise as suitable methods regarding the two requirements in Table 4-2. In light of the wish list demonstrated in section 3.2, a suitable aggregation function has to be defined. The form of the aggregation function decides about its adequacy to the components description schema. The next section discusses this issue.

4.1.2.5 Building a suitable aggregation function

The aggregation function has to reflect the description schema of looked-for component, as defined in section 3.2. So it must especially emphasize the difference between the hard features and the soft features.

The weighted production formula provides the possibility to refuse those alternatives, which don't offer even one of the hard features. On the other hand, the weighted sum formula enables a kind of soft compensation among features, and accordingly, it is suitable to aggregate values of soft features.

SeMCDM suggests a combination of both formulas to evaluate the offered alternatives:

$$v(a) = \prod_{j=1}^{k} u_j(a)^{w_j} [1 + \sum_{i=1}^{n} u_i(a) w_i]$$

Where:

a : The considered offer (or alternative)

$v(a)$: Evaluation of offer a

k: Number of the hard features

w_j : Weight of the hard feature j

$u_j(a)$: Utility value of offer a on the hard feature j

n: Number of soft features

$u_i(a)$: Utility value of offer a on the soft feature i

w_i : Weight of the soft feature i

The formula is specified in this way to combine the advantages of weighted sum and weighted product formulas: The weighted sum ensures a "soft" compensation effect between the soft features, the weighted product ensures a "hard" compensation effect between the hard features, and the combination (the multiplication) provides a total compensation effect between the hard and the soft features.

According to the defined aggregation function, the fitness values are always in the range:

$$0 \le v(a) \le 2$$

However, while the fitness values are used in a comparative way, the range of their absolute values doesn't affect the selection of the best available alternative.

If the user prefers to avoid the weighting of the hard features, a neutral weight of 1 can be assigned to all of these features. In this case the weighted product part would be practically simplified to a normal product of alternative's utilities on the hard features, and the formula takes the following form:

$$v(a) = \prod_{j=1}^{k} u_j(a) \ [1 + \sum_{i=1}^{n} u_i(a) w_i]$$

4.2. Ontologies of the SeMCDM architecture

The SeMCDM architecture is based on three types of ontologies: (i) the kernel ontology, (ii) the Multi-Criteria Decision Making ontology and (iii) the domain ontologies.

The kernel ontology reflects the description schema of autonomous units as presented in section 3.2. The MCDM ontology extends the kernel ontology to different types of utility functions enabling a semantic-aware selection mechanism under multiple criteria.

The next sections describe these ontologies and put emphasis on their semantics and interrelation.

4.2.1 The kernel ontology

The kernel ontology, shown in Figure 4-1, consists of the main elements required to describe the autonomous units according to the description schema presented in section 3.2.

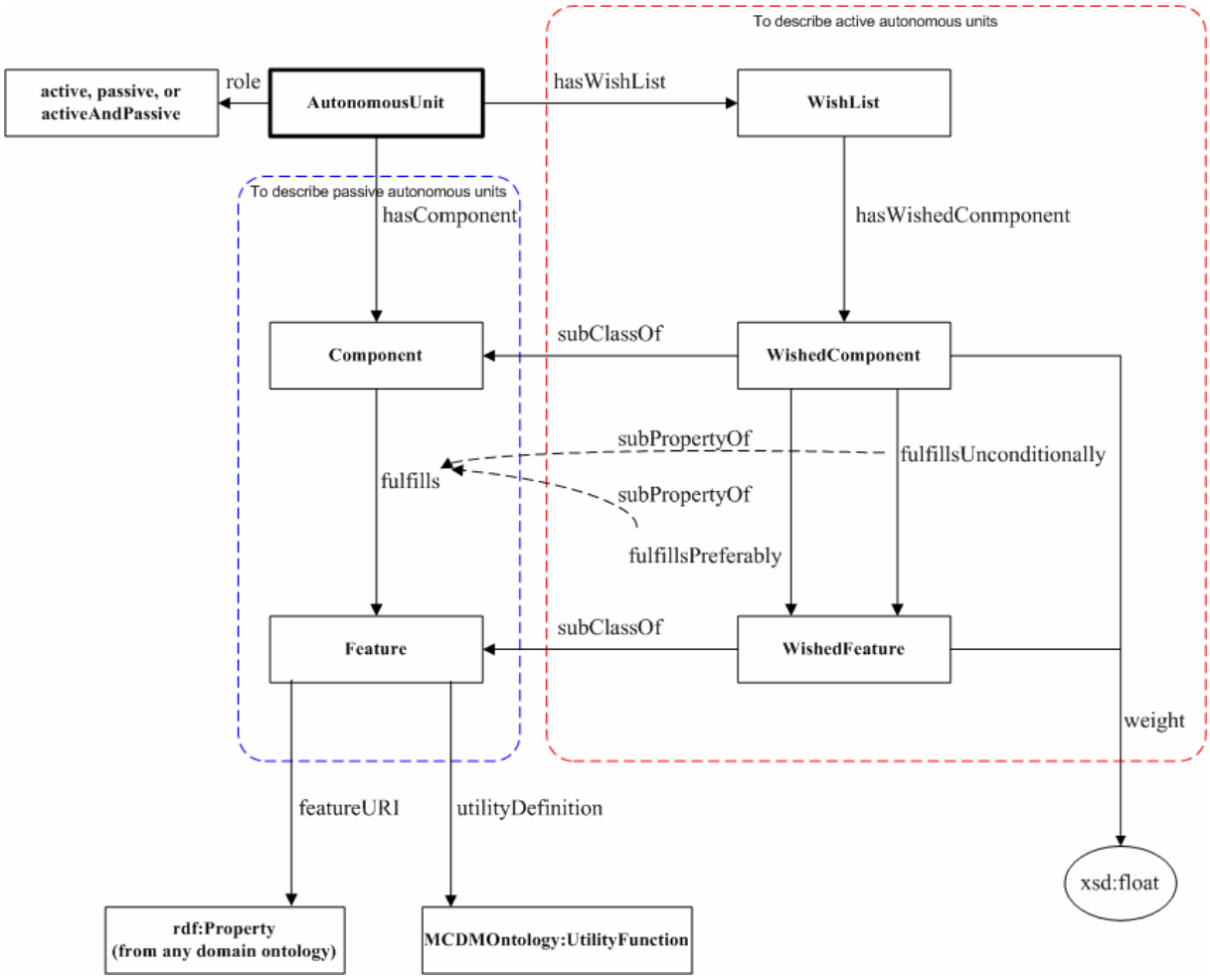


Figure 4-1 The kernel ontology of the SeMCDM architecture.

The kernel ontology includes the following concepts:

- *AutonomousUnit*: Represents an autonomous unit, with a specific role assigned through property *role*. Three roles are already predefined in the kernel ontology: *active*, *passive* and *activeAndPassive*. For more about the difference between these roles see section 3.2.
- *Component*: Functionalities of autonomous units are called Components. One autonomous unit can offer different kinds of functionalities, and therefore, it may have more than one component (property hasComponent). A component is related to one or more features by the property *fulfils*.
- *WishedComponent*: A sub-concept of *Component*, supplied with the *weight* property. A *WishedComponent* is related to instances of *WishedFeature* through sub-properties of the property *fulfils*: *fulfilsPreferably* and *fulfilsUnconditionally*.
- *WishList*: Represents a list of wished components, using the property *hasWishedComponent*.
- *Feature*: A feature of a Component defines which value(s) the component has in relation to one property. Properties are normally imported from the domain ontologies (like gain, noise, color, etc.). Values are assigned to the feature through the property *featureValue*, which refers to one instance of *UtilityFunction* (s. the MCDM ontology in section 4.2.2).

- *WishedFeature*: A sub-concept of *Feature*. It has an additional *weight* property: *Feature(s)* describe *Component*(s), while *WishedFeature(s)* describe *WishedComponent(s)*. For this reason, a feature is restricted to clear and specific values of the component, while a wished feature may enjoy a wider range of values, giving higher chance for finding an adequate offer (see section 4.3 for more about this point).

4.2.2 MCDM ontology

The MCDM ontology presents a set of concepts and instances originating from the world of Multi-Criteria Decision Making. It aims to build a bridge between the ontology-based description and the possibilities and techniques used for MCDM.

The concept "*MCDMFunctions*" is the father concept for all MCDM functions with their two types: *AggregationFunctions*, and *UtilityFunctions*. The emphasis is put on the utility functions, which assess the utility of the features' performance in the ontology-based description.

Utility functions are mathematical functions with special properties: They return values between 0 and 1. For this reason almost all sub-classes of *UtilityFunctions* are also sub-classes of the concept *MathematicalFunctions*, and they share the same properties like: Number of parameters, number of needed interpolation points to parameterize the function, and parameter values.

In section 4.1.2.3 a set of utility functions have been chosen and justified for the usage in connection with the proposed description schema of autonomous units. These functions are *i*. The predefined utility functions for the utility assessment of quantitative features and *ii.* Predefined scales for the utility assessment of qualitative features. The MCDM ontology supports these utility functions through a corresponding set of concepts, as described in the following sections.

4.2.2.1 *OnePointUtilityFunction*

With the help of a *OnePointUtilityFunction* a utility value (*point_y*) can be assigned to a numeric (float) value (*point_x*) of a feature (see Figure 4-2). It has two special cases: *OnePointMinUtility/MaxUtilityFunction*, where the utility of the point has special values (0 and 1 accordingly).

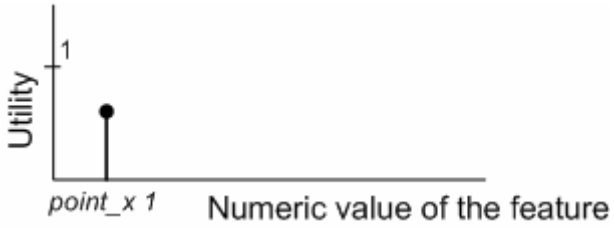


Figure 4-2 A *onePointUtilityFunction*

4.2.2.2 *MultiPointUtilityFunction*

The *MultiPointUtilityFunction* generalizes the *OnePointUtilityFunction* to support multiple points (see Figure 4-3). It has also a special case function *MultiplePointMaxUtilityFunction*.

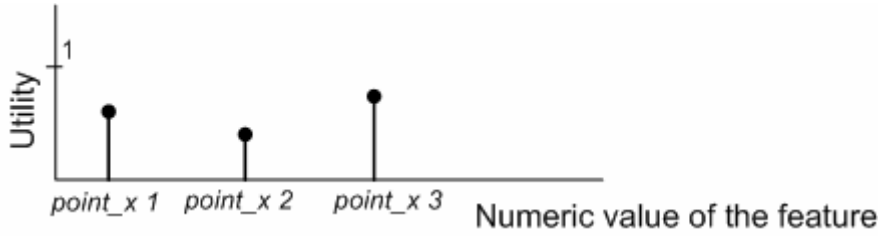


Figure 4-3 A *MultiPointUtilityFunction*

4.2.2.3 *Linear/Quadratic/Cubic/Exponentialutility functions*

This set of continuous functions gathers typical utility functions, which return the utility of numeric feature values. These utility functions are defined as following:

- LinearUtilityFunction (see Figure 4-4) of the form: $u(x) = \alpha x + \beta$
- QuadraticUtilityFunction of the form: $u(x) = \alpha x^2 + \beta x + \gamma$
- CubicUtilityFunction of the form: $u(x) = \alpha x^3 + \beta x^2 + \gamma x + \delta$
- ExponentialUtilityFunction of the form: $u(x) = \alpha + \beta e^{-\gamma \cdot x}$

To ensure a utility value between 0 and 1, the values of the input x have to be limited to a "rational" range. As this range is related to the functions parameters values (α, β, γ, δ), the designer has to consider this issue while setting up the parameters values. However, returned values above 1 can be "interpreted" in run time as a 1, whereas negative returned values can be interpreted as 0.

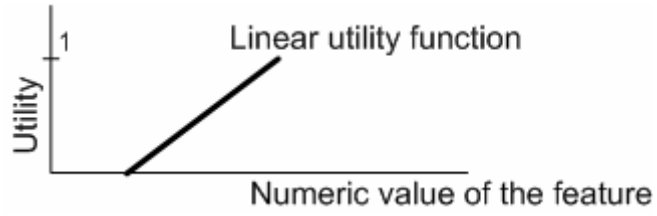


Figure 4-4 A *LinearUtilityFunction*

4.2.2.4 *LikertScale*

The concept *LikertScale* has only the property *quantitativeEvaluation*. Every instance of the Likert scale is related to one numeric value (its quantitative evaluation). The MCDM ontology defines 5-values (instances) of *LikertScale*: *veryGood* with the quantitative evaluation of 1, instance *good* of 0.75, instance medium of 0.5, instance *poor* of 0.25 and instance *veryPoor* with the quantitative evaluation of 0.

4.2.2.5 *LikertScaledPoint*

A *LikertScaledPoint* links a non-numeric value with an instance of *LikertScale*. The property *physicalValue* refers to a non-numeric instance of any concept (usually defined in a domain ontology). Property *qualitativeEvaluation* refers to an instance of *LikertScale*. The concept *LikertMaxScaledPoint* is a special case, which has the maximum *quantitativeEvaluation* (i.e. *veryGood*).

4.2.2.6 *LikertScaleUtilityFunction*

A *LikertScaleUtilityFunction* gathers one (or more) LikertScaledPoint(s) through the property hasLikertScaledPoint. LikertScaleMaxUtilityFunction is a special case with all points of type LikertMaxScaledPoint.

4.2.2.7 Interval utility functions

The interval utility functions are the solution for special cases, where a "range" of continuous values is to be selected, instead of a single value. Predefined instances of the concept *Ranges* are: *greaterThan*, *insideOf*, *lessThan* and *outsideOf* . Each interval is defined through its limits (through property *intervalLimits*). Interval limits are always instances of *OnePointUtilityFunction*.

Through the property *hasFunction*, any mathematical function can be chosen as a utility function within the selected range.

The concept *SimpleIntervalUtilityFunction* restricts the values of this property to the predefined instance *unityFunction* (it return always 1). The MCDM ontology defines four sub-concepts of *SimpleIntervalUtilityFunction*, each with one instance of *Ranges* (as above):

- *UnityFunctionLessThanLimit*: Defines acceptable values as those smaller than a specific value p
- *UnityFunctionGreaterThan*: Defines acceptable values as those greater than a specific value p
- *UnityFunctionInsideOfRange*: Defines acceptable values as those located between two specific points p1 and p2 (see Figure 4-5)
- *UnityFunctionOutsideOfRange*: Defines acceptable values as those located outside the range specified with two points p1 and p2 (see Figure 4-6)

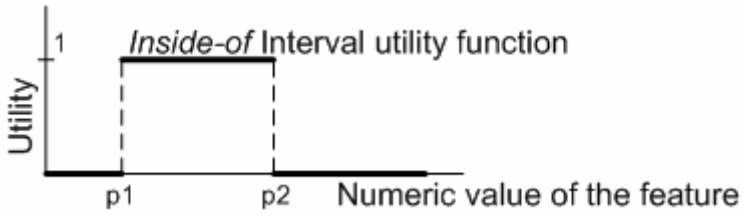


Figure 4-5 A *UnityFunctionInsideOfRange*

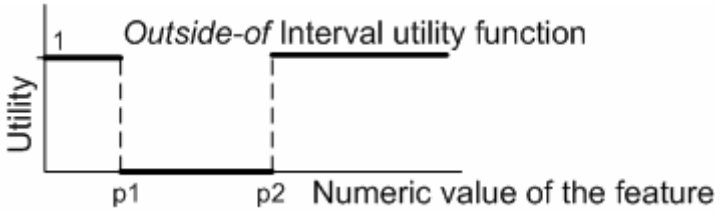


Figure 4-6 A unityFunctionOutsideOfRange

4.2.3 Domain ontologies

The domain ontologies are strongly related to the application domain. Concepts, instances and properties defined in the domain ontologies can be used to complement the *Features* with the desirable semantics. Both the kernel ontology and the MCDM ontology don't put any restriction on the usage of domain ontologies.

Figure 4-7 shows how SeMCDM combines all ontologies to describe the autonomous units.

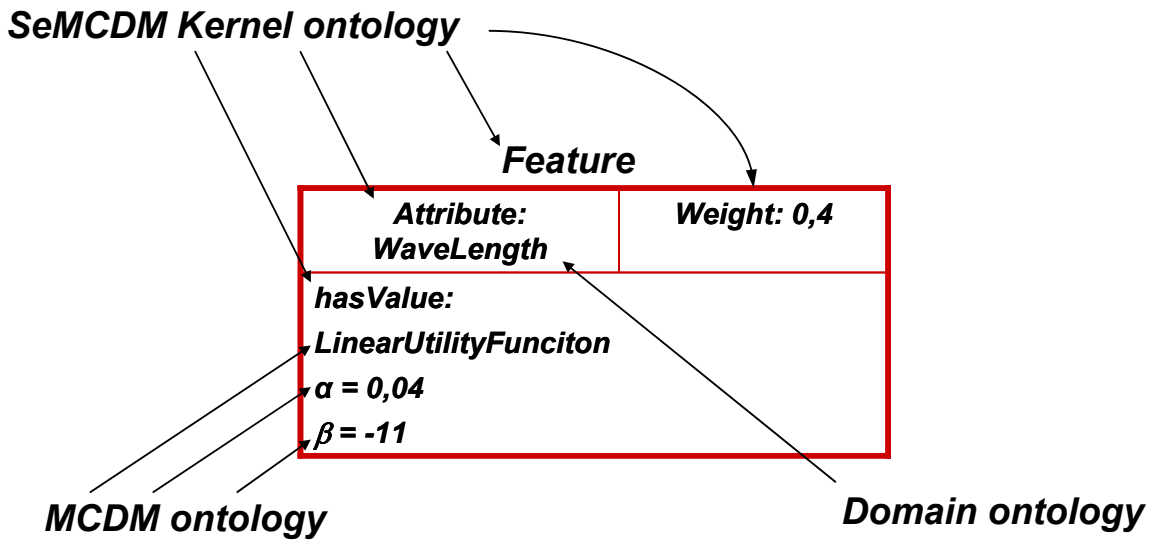


Figure 4-7 A feature of an (active) autonomous unit is described with the help of three types of ontologies.

4.3. Semantic matching for MCDM

This section considers the basic matching process, which takes place between pairs of features: A wished feature from the enquiry side, and one feature from the offering side. As a feature consists of a property (imported usually from a domain ontology) and its values (in the form of a utility function from the MCDM ontology), the matching process has to consider both parts. Therefore, the matching process suggested here is called "semantic matching for MCDM". Semantic matching for MCDM is the kernel of SeMCDM, and one of the main novel contributions of this thesis.

Section 4.3.3 addresses the semantic matching between the properties of two features and section 4.3.4 addresses the semantic matching between the utility functions of two features. Especially the semantic matching between utility functions must be based on a clear understanding of their semantics. Sections 4.3.1 and 4.3.2 address this issue. A clear difference is made between the semantics of offer features (instances of *Feature*) and enquiry features (instances of *WishedFeature*). The difference has its root in the degree of freedom given to each kind of features: Passive units are characterized by concrete values of their offer features, while looked-for components define flexible ranges of acceptable feature values. Consequently, the offer features (*Feature*) have limited degrees of freedom in comparison with enquiry features (*WishedFeature*). The next two sections show the effect of this fact on the interpretation of utility functions in relation to *Feaures* and *WishedFeatures*, providing a solid base for defining the process of semantic matching for MCDM in two following sections.

4.3.1 Semantic of the utility functions in relation to offers

One of the requirements in section 4.1.1 dictates that an alternative is expected to provide stable and known performance value on the criteria (features). In this section a slight extension (or a broad interpretation) of this requirement suggests that:

- The offer has only one - known and concrete - value for each feature (no fuzzy or multiple values)
- The performance of the offer on the feature is optimal, which means that the passive units completely fulfil the features with the given values in their offers. This restriction would simplify the aggregation function, to be just like usual aggregation functions used in MCDM problems. However, an extension of the aggregation

function to consider the offers performances is easily imaginable in future development, although it implicates more theoretical questions than practical implementation difficulties.

Consequently, the adequate utility function must be of type *OnePointMaxUtilityFunction* for numeric offer features, and of type *LikertScaleMaxUtilityFunction* for offer features with non-numeric value. *SimpleIntervalUtilityFunction(s)* can describe intervals of values with the interpretation of an interval as a spectrum of values (for example, an audio amplifier has a frequency range between 33 Hz and 30 kHz). This case is not to be confused with multiple values or with other possible interpretations like "the feature has an unknown value in the specified range".

4.3.2 Semantic of the utility functions in relation to enquiries

Features of enquiries have no restrictions about the usage of utility functions. Consequently, any utility function from the MCDM ontology can be used.

Continuous functions (section 4.2.2.3) are adequate utility functions of enquiry features. Using *MultiPointUtilityFunction* the enquiry can specify more than one acceptable numeric value of one feature, where every point has a particular utility value. Any numeric value outside the defined points has a default utility value of 0.

The same holds for non-numeric values using the *LikertScaleUtilityFunciton(s)*. *IntervalUtilityFunction(s)* in relation to enquiry features have to be interpreted as a spectrum of acceptable values.

4.3.3 Semantic matching between properties

It is the typical case of semantic matching, which makes use of the (domain) ontology and of inference rules to get a logical answer of yes or no. In order to get the expected matching results, a set of application-dependent matching rules can extend the matching capabilities of the inference engine.

4.3.4 Semantic matching between utility functions

The semantic matching between utility functions takes place when the semantic matching between properties delivers a positive answer. It deals with the question about the utility of the offered values in relation to the utility function of the enquiry feature. The question can be divided into 2 partial questions: "Is the enquiry feature fulfilled by the offered feature?" and "To which extent is the enquiry feature fulfilled by the offered feature?". The first question will be answered through a "utility check" (section 4.3.4.1), while the second is answered through a utility calculation in section 4.3.4.2[5]. Both sections adopt the semantics of the utility functions as defined in sections 4.3.1 and 4.3.2. The following figures in Table 4-3 show example combinations between pairs of utility functions.

[5] In some cases, there is no strict limit between the utility check and the utility calculation, because the utility calculation is sometimes a part of the utility check.

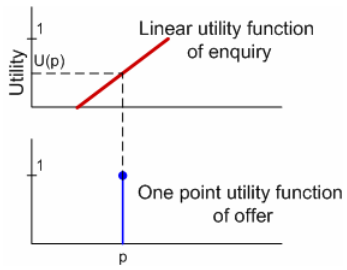
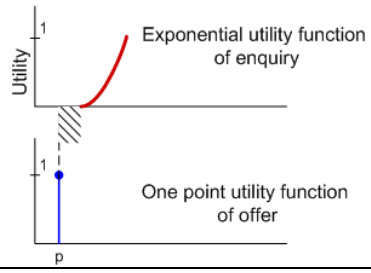
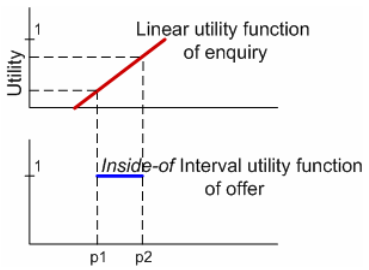
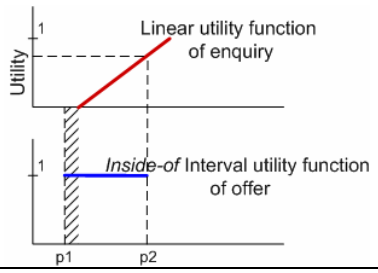
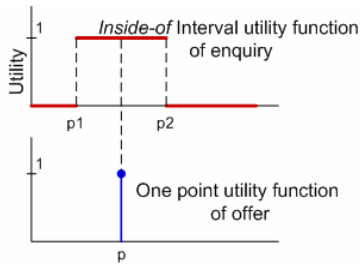
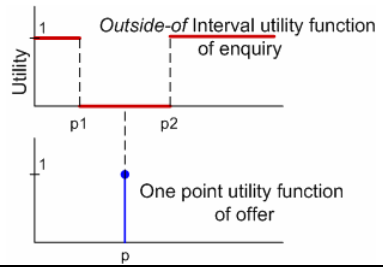
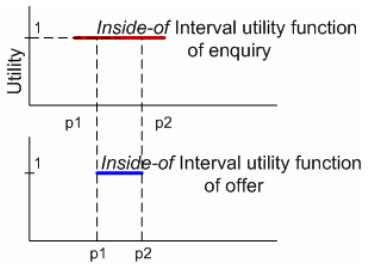
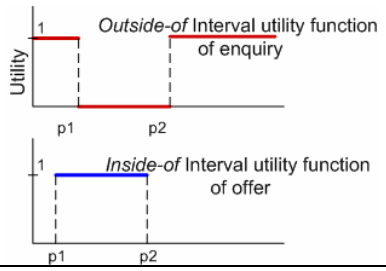
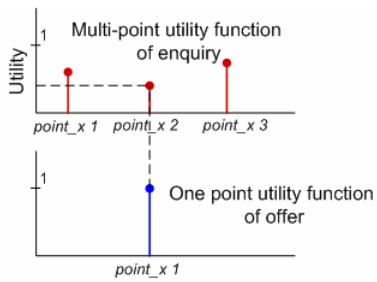
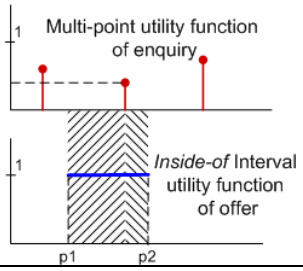
		Quantitative offer feature	
		One point (max)UF	(Simple) Interval UF
Quantitative enquiry feature	Continuous UF	*Example of a successful match*  *Example of a failed match* 	*Example of a successful match*  *Example of a failed match* 
	(Simple) Interval UF	*Example of a successful match*  *Example of a failed match* 	*Example of a successful match*  *Example of a failed match* 
	Multi-point UF	*Example of a successful match* 	*In this combination the match is always unsuccessful* 

Table 4-3 Example combinations between (quantitative) utility functions.

4.3.4.1 Utility check

The offered feature is considered to fulfil the enquiry feature if it has a utility value greater than 0. For this reason, the semantic matching between utility functions performs a kind of "utility check", which delivers a logical result (yes or no).

The utility check takes different forms according to the combinations of the specific utility functions on the offer side and on enquiry side.

Table 4-4 shows these combinations and the way of checking the utility of the feature's values. In the table, rows represent the utility functions of enquiry features, while the columns represent the utility functions of offer features.

			Offer feature		
			Quantitative		*Qualitative*
			One point (max)UF	(Simple) Interval UF	Likert-scale (max)UF (one point)
Enquiry feature	*Quantitative*	Continuous UF	If offer point has utility value > 0	If all interval points have utility values > 0	Application/property dependent
		(Simple) Interval UF	If the offer point falls within the enquiry Interval	If the offer interval falls totally within the enquiry interval	Application/property dependent
		Multi-point UF	If the offer point is one of the enquiry points	Doesn't match	Application/property dependent
	Qualitative	Likert-scale UF (Multi points)	Application/property dependent	Application/property dependent	If the offer point is one of the enquiry points.

Table 4-4 Utility check for all combinations of utility functions on the enquiry side (rows) and on the offer side (columns)

For pairs of quantitative and qualitative utility functions (LikertScaledUtiltiyFunction) there are no predefined rules for the utility check. This is due to the fact that the transformation from the world of qualitative values to the world of numeric (quantitative) ones is a very application dependent process. For example, the user can specify the interval (30Hz - 33KHz) as equal to the quantitative value "AudioRange". This transformation is only correct in the eyes of the beholder, and for a special property like "hasRangeOfFrequency" used for describing an amplifier: "amplifier X hasRangeOfFrequency between a and b".

4.3.4.2 Utility calculation

The second part of semantic matching between utility functions is the "utility calculation". It finds out the concrete utility value of the offered feature according to the utility function of the enquiry feature. While the utility check helps to select the acceptable offers (together with the semantic matching of properties), the utility calculation is the base of selecting the best available offer according to MCDM.

The utility calculation returns the utility value of the offered feature in relation to the enquiry feature. Methods of utility calculation differ for each combination of utility functions as shown in Table 4-5. In the table, rows represent the utility functions of enquiry features, while the columns represent the utility functions of offer features.

For some combinations of utility functions, the utility check is independent of the utility calculation and the utility calculation makes only sense in the case of successful utility check. For other combinations, the utility calculation has to take place before making the utility check. This fact makes the utility calculation to a part of the semantic matching for MCDM.

			Offer feature		
			Quantitative		*Qualitative*
			One point (max)UF	(Simple) Interval UF	Likert-scale (max)UF (one point)
Enquiry feature	*Quantitative*	Continuous UF	Utility value of offer point on the enquiry UF	Minimal utility value of the interval on the enquiry UF	Application\property dependent
		(Simple) Interval UF	Utility value = 1 (for simple interval UFs)	Utility value = 1 (for simple interval UFs)	Application\property dependent
		Multi-point UF	Utility value of the offer point	Utility value is always 0.	Application\property dependent
	Qualitative	Likert-scale UF (Multi points)	Application\property dependent	Application\property dependent	Utility value of the offer point

Table 4-5 Utility calculation for all combinations of utility functions on the enquiry side (rows) and on the offer side (columns)

4.4. Generalized matching process

The generalized matching process makes use of the semantic matching for MCDM (Section 4.3), by applying it to feature pairs, where a feature pair consists of an offer feature and an enquiry feature.

From a set of available offers, the generalized matching process selects the best available offer for an enquiry.

For each enquiry, the suggested description schema (section 3.2) distinguishes between “hard features” and “soft features”. Therefore, the matching process can be divided into two steps as shown in the next sections.

4.4.1 First matching step

The first matching step considers only the hard features. Working as a filter of offers: Offers fulfilling the hard features are considered as “acceptable offers” and they pass through the first matching step to the second step.

For each hard feature the process of “semantic matching for MCDM” tries to find a match from the offer’s features. The exact utility value is not important in this context.

The matching is regarded unsuccessful, when one of the hard features is found to be not fulfilled by the offer. The relation between the hard features is characterized by the logical relation “AND”.

After a successful matching of all hard features, one feature of the offer is announced as the matched feature for each of the hard features of the enquiry[6].

4.4.2 Second matching step

The second matching step accomplishes semantic matching and utility calculation for the soft features.

In contrast to the soft features, the soft features can be matched sequentially and the matching process has to cover all of them, independent of the matching results of the individual features.

A second difference to the matching of hard features is the utility calculation, which has to take place after (or before) a successful utility check[7].

The utility values of features, along with their weights (The weights are assumed to be given in the design time, as described in section 4.1.2.2) are then to be aggregated to an offer evaluation. The aggregation function has been described in section 4.1.2.5. The best offer is the one with the highest evaluation value.

The steps of the generalized matching process are mapped to autonomous units and/or to central in section 4.5, where the market scenarios are presented.

4.5. The Generalized matching process and the marketplace-oriented behavior

The generalized matching process suggested in section 4.4 consists of two steps, gathering the semantic aspects to the multi-criteria nature of decision making problems. In the absence of a central broker, this process has to take place in the autonomous units. On the base of the marketplace-oriented behavior as an upper behavior schema for the autonomous units, this section discusses the allocation of the matching steps to active and passive units.

Systematic design conditions are defined in section 4.5.1. A set of market scenarios emerged then in section 4.5.2.

Beside the distributed market scenarios, the possible commitment of a central broker completes the allocation possibilities and results in a central scenario.

4.5.1 Conditions on the allocation of the matching steps on autonomous units

For the selection of the best available offer, the evaluations of all offers must be available for the decision making unit. By its very nature, a passive unit is interested only in enquiries and in its own offer. Furthermore, a passive unit is not aware of offers coming from other passive units. For these reasons, the allocation of the second matching step (or at least the decision making) to passive units has to be avoided.

The suggested scenarios below assume the existence of a central common memory for communication purposes between the autonomous units. This central common memory is called “broker”, however the market scenarios can be considered as distributed, as long as no matching activities are performed by the central broker (the broker can be replaced in this case by a broadcasting mechanism). However, the allocation of matching activities to the

[6] The exact utility value is only available for some of the offered features, as a spin-off product of the utility check.

[7] The utility calculation of the hard feature in the first step is not always required to perform a utility check. Therefore, the second matching step has to consider the hard features again, and to calculate the not available utility values.

central broker has been also considered, and the resulting central scenario serves as a reference to evaluate the distributed scenarios.

Different scenarios may evolve depending on the initiator of an action. Passive units can initiate a market scenario by forming offers of their features and then sending them to the active units. Active units can also initiate the market scenario by sending their enquiries to the passive units. Taking the possibilities of centralized matching into account, three groups of market scenarios can be distinguished:

- **Enquiry-oriented scenarios**: The active units send enquiries to the passive units. Each passive unit sends its offer when its own features meet at least the hard features in the enquiry. The first matching step takes place in this case in the passive units. The passive unit can also address principally the soft features, and perform a part of the second matching step. Only the decision making (the selection) still always beyond the control of the passive units.
- **Offer-oriented scenarios**: The passive units send offers to the active units, while the active units try to choose acceptable offers, carrying out the whole matching process.
- **Mix scenarios**: Active and passive units send queries and offers, respectively, to a central "broker", which performs the general matching process (at least the first matching step).

The run-time performance of market scenarios, as well as their structure, is affected also by timing conditions of actions. Two principles characterize SeMCDM scenarios:

- The first matching step is automatically triggered when an enquiry meets an offer. This principle leads to the fact that the passive units perform the first matching step in enquiry-oriented scenarios and that the active units perform the matching process in the offer-oriented scenarios.
- To assure that an enquiry gets at least an offer (when available), the enquiry stays available on the marketplace for a predefined time period, i.e. expiration time of the enquiry. The second matching step will be only triggered after the expiration of the enquiry. All scenarios follow this principle with the exception of the 2nd scenario in its "B" version.

Moreover, an important rule of market completes the previous conditions: Direct negotiation, normally between active units about one offer, are not allowed and not required. A reservation mechanism is adopted in the next scenarios to avoid the need for negotiations. A reservation of an offer for a specific enquiry takes place after performing the first matching step with a positive result. The offer cannot be then considered by other units until it has been given free again, usually because the second matching step resolved that the offer is not the best available one. The reservation is not to be thought of as an ultimate solution for concurrence situations, but as a way to avoid negotiations between autonomous units. Although it could be a good way for optimizing the resulting contracts, the negotiations between units builds a higher degree of complexity and falls beyond the scope of this study (the game theory discusses such approaches [Ros09]).

4.5.2 Market scenarios

Taking the conditions in the last section into account, possible allocations of matching steps to the autonomous units evolve to a set of market scenarios. Table 4-6 shows the types of the market scenarios with their initiator, the type of messages exchanged between the units and the responsible unit of both the first and second matching step.

	Scenario 1	Scenario 2 (in two versions "2A" and "2B")	Scenario 3
Initiator	Active units	Passive units	Active and passive units
Type of messages	Enquiries	Offers	Enquiries and offers
1.st matching step	Passive units	Active units	Central
2.nd matching step	Active units	Active units	Central

Table 4-6 Possible market scenarios with their specifications

The next sections describe the market scenario in details, where each scenario is presented as a sequence diagram. A complete flow through the sequence diagram is called as "cycle". Because of concurrence between the autonomous units more than one cycle can be necessary to find a suitable offer for all wished components. Appendix A shows the market scenarios as sequence diagrams.

4.5.2.1 Market scenario 1

In this enquiry-oriented scenario (Figure 4-8), the active units send their enquiries (one for every looked-for component) to the broker, and wait for suitable offers. Every free passive unit reads an enquiry (randomly selected by the broker), checks the matching relation between its own features and the hard features in the enquiry. When a suitable match can be proved for all these features, the passive unit sends an offer, addressed at the specific enquiry. Every passive unit may send only one offer addressed to one enquiry. With this behavior the need for conversation between active units is avoided.
The broker stores the offers addressed to each enquiry, until the enquiry expires. At this time point, the broker sends the offers to the enquiring active unit and deletes the enquiry. The active unit performs the second matching step and chooses the best available offer. The active units then conclude a contract with the passive unit of the best offer. All other offers will be deleted and all other passive units are free again. Multiple offers can be directed to one enquiry, while other enquiries will not receive any offers. Also when suitable offers are available, the active units will generally need more than one cycle in order to detect them.
Scenario 1 is similar to the marketplace-oriented behavior meant by EvoArch (section 2.2.3), as well as by the Contract Net Protocol (section 2.3).

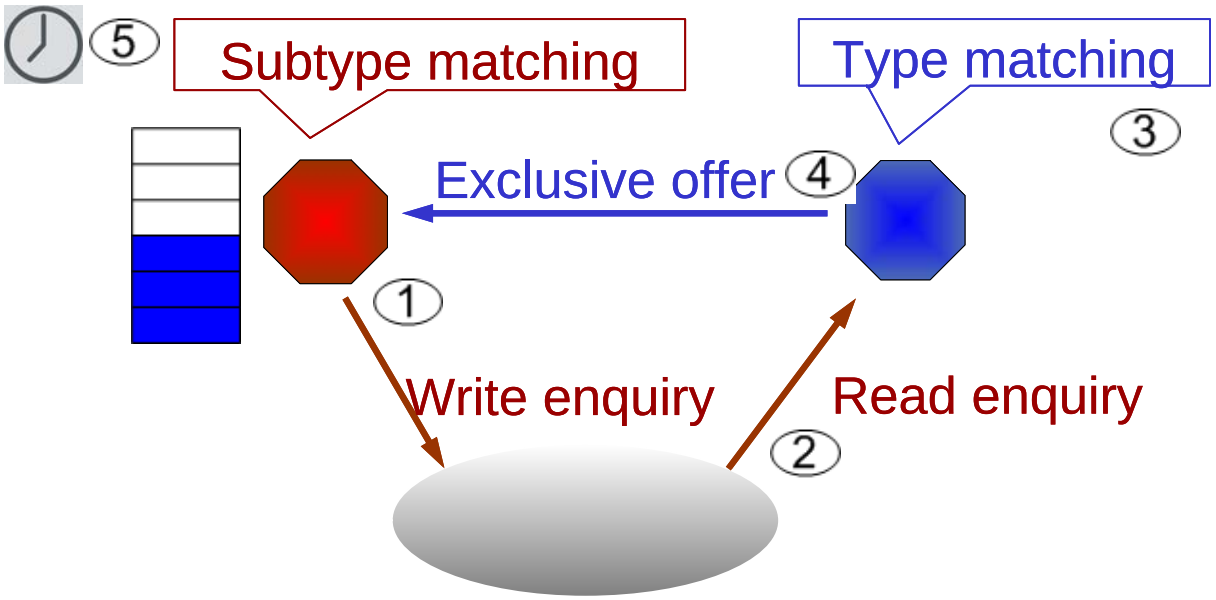


Figure 4-8 Scenario 1 is an enquiry-oriented scenario.

4.5.2.2 Market scenario 2A

In an offer-oriented scenario, the start point of the marketplace activities is the passive unit, which tries to find a partner, by putting its own features as an offer in the central broker. Each active unit can read an offer (randomly selected by the central broker) and decides about its suitability as one of the wished components (first matching step). In the positive case, the active unit tries to reserve the offer (the offer might be already reserved for another active unit). When the enquiry expires, the active unit calls its reserved offers from the broker, and performs the second matching step to choose the best available offer and to make a contract with the corresponding owner (passive unit). All other offers will then be released, and a new cycle can be started.

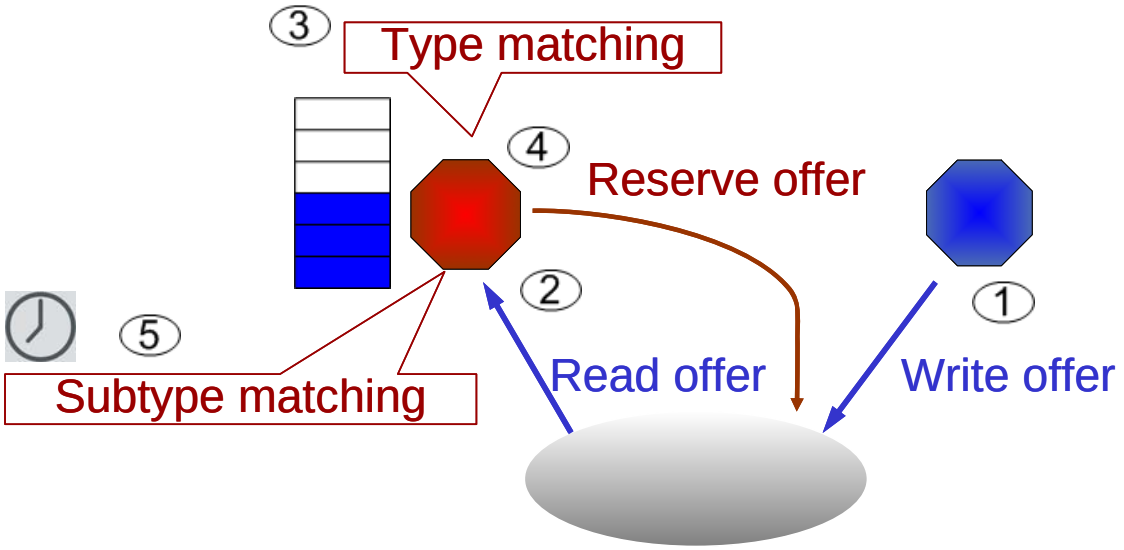


Figure 4-9 Scenario 2A is an offer-oriented scenario.

4.5.2.3 Market scenario 2B

Version “A” of scenario 2 has the disadvantage of reserving more than one offer for one active unit, waiting for the time-out of the enquiry and the end of the second matching step. This prevents other active units from making use of these offers in the same cycle. Version “B” of scenario 2 tries to avoid this disadvantage by performing the second matching step

directly after the first matching step, when the offer is proved to be acceptable. This way, only one offer will be reserved for the active unit, i.e. the one with the best evaluation, while other acceptable offers will be released "online".

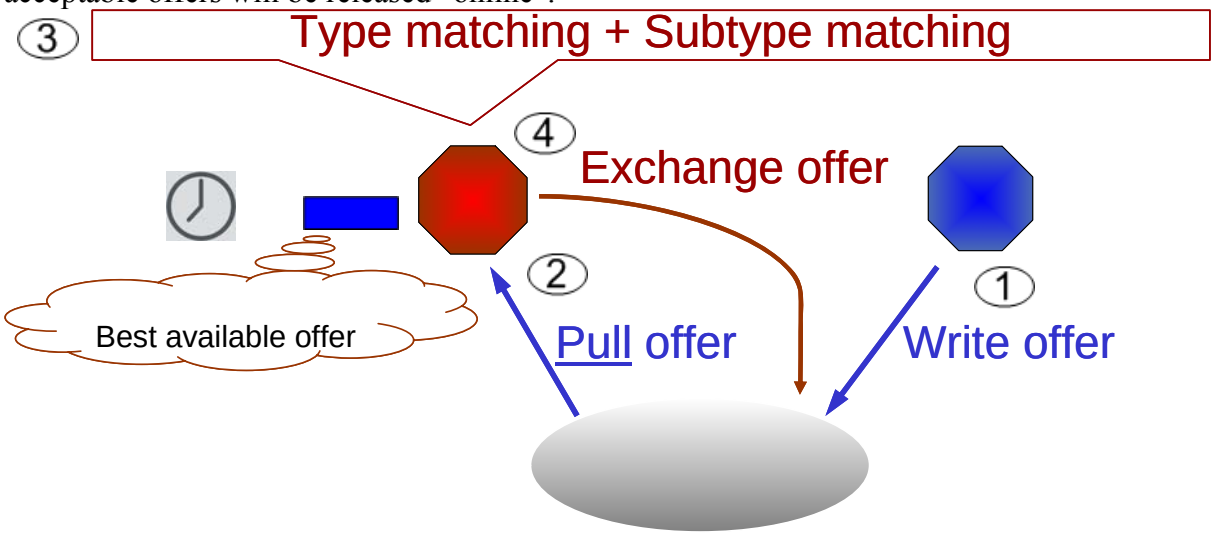


Figure 4-10 Scenario 2B is an offer-oriented scenario, which tries to overcome the disadvantages of scenario 2A.

4.5.2.4 Market scenario 3

This mix scenario relies more on a central broker, in order to perform the matching steps. In this case the autonomous units only have to send their enquiries or offers to the broker, which will find the best offer for each enquiry. Then it recommends the closing of contracts between pairs of active and passive units. All offers will be matched against all requests (first matching step). Upon time-out the offers' list of each enquiry will be processed by the broker (second matching step). As soon as the best offer has been selected, it will be reserved for the specific active units and it will be deleted from the offers' list of other enquiries. The broker tells the active units about the best available offer for each of their enquiries. Contracts are then concluded directly between the active and passive units.

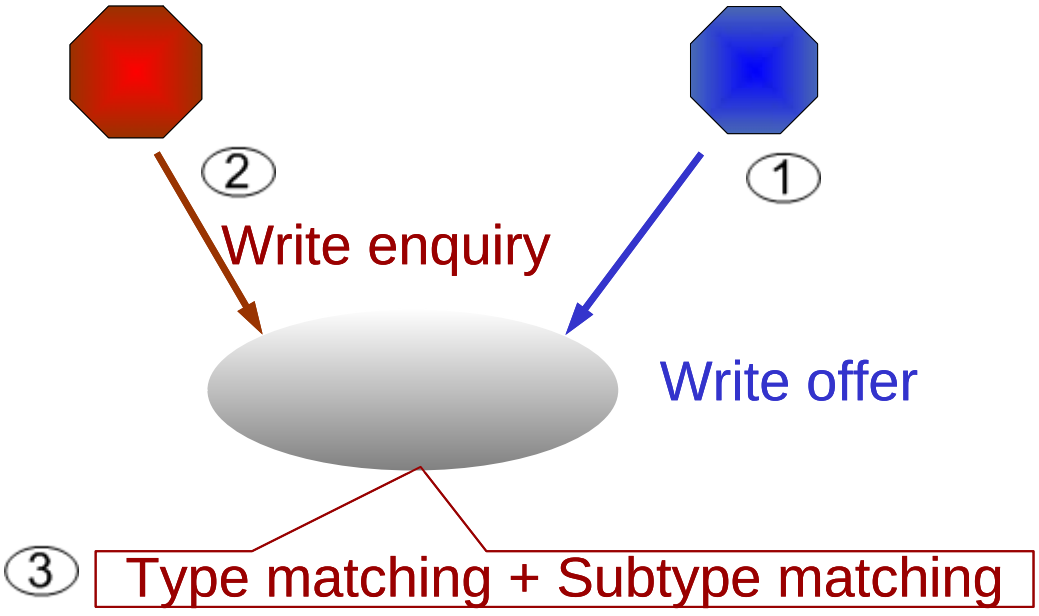


Figure 4-11Scenario 3 is a central scneraio.

4.5.3 Summary

The adoption of the marketplace idea leads to design questions about the allocation of matching steps to the autonomous units. Different possible market scenarios have been presented.

The Contract Net Protocol (see section 2.3) is very similar to market scenario 1. This makes the Contract Net Protocol to a special case of the market scenarios presented (and evaluated in following sections) in this thesis.

The selection of the most suitable one is not a trivial question because the nature of the application environment (in terms of number of available autonomous units and their types) is expected to affect the performance of these scenarios. Section 5.4 addresses this issue in more details.

4.6. Selection of automotive communication platforms

A communication platform builds the basis for any distributed architecture, like SeMCDM, This section proofs the adequacy of automotive communication platforms for the purposes of SeMCDM. The automotive communication platforms have been already reviewed in section 2.7. The requirements of the SeMCDM architecture on the communication platform are analysed in section 4.6.1. The following subsections apply the requirements on automotive communication platforms.

This section concludes with results about the most suitable communication platforms and gives some suggestions for a step-wise integration of autonomous computing in automotive systems.

4.6.1 Requirements on the communication platform

The autonomous units exchange two main types of messages through the communication framework: Messages on the organic level (i.e. offers and enquiries) and operational control messages (on/of-control, set value, get value, etc.).

The requirements depicted here are related only to the organic level. The communication on the operational level is beyond the scope of this thesis. However, the communication platform is expected to be shared between the organic and operational level. To this reason, the organic communication level would rely on available automotive communication platforms like CAN, LIN, FlexRay or MOST.

By analyzing the SeMCDM architecture and its application environment, the next requirements are expected to be supported by the underlying communication platform:

- **Dynamic extension**: SeMCDM is designed to be applied in open environments, where new devices (autonomous units) are supposed to be engaged or disengaged in run-time. Therefore, the communication network has to support the ability of such dynamic extension.
- **Bandwidth**: Offer and enquiry messages take the form of logical expressions with different strings lengths. Enquiry languages and logical expressions define the syntax of these messages. The lengths of the messages are also affected by the number of features (for offers) and required features (for enquiries), as well as by the coding format (usually ASCI). Especially in combination with URI, the length of the message can simply reaches some kilobytes. The communication framework has to offer "sufficient" bandwidth, taking into consideration the number of autonomous units and their parallel activities on the marketplace (according to the adopted market scenario).

- **Fair access to the communication framework**: It is supposed that all autonomous units have equal access rights and, consequently, the messages (offers or enquiries) have also equal priorities. The communication framework may not affect the results of activities on the organic level, for example, because different types of messages (or of their sender/receiver) are exposed to different priorities and wait times.
- **Support of centralized messages exchange**: Market scenarios rest on a shared memory, called as broker. The existence of this broker means that the messages exchange goes through a central node. For the distributed scenarios (the first three described in 4.5.2) the broker plays in reality the role of a common black board. Such black board can be replaced with broadcasting mechanisms. However, the centralized scenario (section 4.5.2.4) expects additionally specific computing capabilities from the broker. The communication framework is expected to support the centralized messages exchanges in all cases, without to build a bottleneck for the whole system.
- **Adequacy for the application environment**: The selection of the communication framework depends also on its suitability to the application and to the surrounding environment. For the automotive industry this means that the communication platform has to withstand the harsh environmental and electromagnetic conditions in the car.
- **Real time capability**: The capability to support real time systems with organic behavior depends partially on the real time capability of the communication framework.
- **Independence from special resources**: To avoid a system collapse because a single point of failure, the communication framework has to provide stable connections, which don't depend on particular resources, like a central node of communication.

4.6.2 Assessment of the CAN Bus as a communication platform for SeMCDM

This section proofs the adequacy of CAN Bus/CANopen as a platform for the SeMCDM architecture. In the following subsections the requirements defined in section 4.6.1 will be discussed in relation to the CAN bus.

4.6.2.1 Dynamic Extension

Devices on the CAN Bus have to be configured manually using special tools. Off-the-shelf plug-and-play functionality was not originally foreseen in CANopen. [Zel01] argues that solutions for this problem have to take place on the application level, without giving more details. However, dynamic extension is not the mainstream of applications based on CAN bus.

4.6.2.2 Bandwidth

The examples in section 2.7.1 show the transfer time of messages with different lengths. The block transfer is proved to deliver faster transfer that the segmented transfer, with the disadvantage of sending only one confirmation message for each block (in case of failure the whole block has to be transferred again).

Considering the communication load on the SeMCDM marketplace, the CAN bus will be apparently overloaded with high number of long messages (of multiple kilobytes) and the transfer time of these messages would be not acceptable for most of modern applications. The problem gets an additional dimension of complexity because of the unfair access to the CAN bus, as discussed in the following section.

4.6.2.3 Fairness of access

The CAN Bus rests on a strong prioritization of messages. The prioritisation leads to situations where messages with minimum priority have to wait for long time before they get access the communication platform, while another messages of highest priority can access the communication platform immediately. In such case the results on the organic level can be shifted towards offers or enquiries of higher priorities.

While some application can benefit from such feature, the SeMCDM requirements place special importance to the fairness of access.

However, a suboptimal solution can be here suggested. It is based on random prioritization of the nodes IDs (IDs of autonomous units). Then according to CANopen every node has a dedicated ID of 7 bits (i.e. CAN but supports maximally 128 units). These IDs affects the priorities of the messages, the receiver ID is a part of the identifier field of the messages, which decides about the priority of the message. When these IDs are randomly generated (instead of manual configuration) the messages priorities are then in fact randomized. This way the results on the organic level will not be shifted towards "expected solution". While this solution is acceptable for simulation purposes (statistical values), unknown effects can appear in real applications.

A solution on the scenarios level is also imaginable by using long dead times of enquiries, so that offers of low priorities can be also considered by the autonomous units. This solution has the disadvantage of pushing towards too long cycles (see section 4.5.2) and consequently towards longer times to reach a system configuration.

Therefore, a complete elimination of the negative effects of messages prioritization is not really possible.

4.6.2.4 Support of centralized messages exchange

CANopen suggests to use the ID of the receiver unit as a message identifiers in the CAN Bus frame. Through bit arbitration of messages identifiers, the message of the highest priority will be then selected. In the case that two (or more) messages addressed to the central broker would share the same ID (the ID of the broker) and consequently the same priority on the CAN Bus. This situation could guide the CAN controllers to a conflict, unless the bit arbitration takes place on all messages bits and it is not restricted to the identifier bits (there is no clear statements about this issue. Such details about the execution of arbitration seem to be dependent on the hardware design of the CAN controller).

As the market scenarios described in section 4.5.2 relies on centralized messages exchange, they will be affected by conflicting messages on the CAN bus. A clear example of the problem would happen in the first phase of each scenario, where a great number of enquiries and/or offers have to be sent at the same time to the central broker.

A solution for this problem will be suggested and discussed in the following few lines. By adding the sender ID to the message identifier, the identifier will gather the sender and the receiver IDs. For such arrangement, the 11 bits foreseen for addressing in the base frame format are not more sufficient (The 7 bits identifier could be sufficient only in rare cases, where the maximum number of units is restricted to 8 and consequently 3 bits can be used as a receiver ID and 3 bits as a sender ID). However, the extended frame format supports messages identifiers of 29 bits. Therefore, the suggested solution has to rely on the extended frame format. In order to enable this solution, the messages ID should be freely editable by the application, and the addressed receiver nodes shall recognize their address from the receiver part of the identifier. The suggested solution should take place on the CANopen

level. Therefore, CAN bus and CANopen in its original form are not able to support centralized scenarios.

4.6.2.5 Adequacy for the application environment

CAN Bus has been originally developed for automotive applications. Therefore, this requirement is surely fulfilled by the CAN bus.

4.6.2.6 Real time capability

CAN Bus is able to guarantees maximal transfer time for messages of high priorities. This gives it its name as a real time bus. But this capability is restricted to small messages (Process data objects: PDO) which are not longer than 8 bytes.

Usual offers and enquiries consists of more than 8 bytes, and can be only transferred in segments (as SDO mentioned above), and hence, they cannot benefit from the real time capability of the CAN bus. Even with the most available compression techniques it is not expected to reach as short messages as 8 bytes.

Beside the problem of messages lengths, the existence of high number of units, which want to access the communication platform at the same time, with equal priorities, makes it difficult to think about real time transfer on serial busses in general.

4.6.2.7 Independence from special resources

CAN Bus doesn't rely on any central instance, and hence, it is immune against a system collapse because of a single point of failure.

4.6.2.8 Conclusion

Because of its adequacy for the automotive environment and because of its wide spread since many years, the CAN bus is a natural candidate to be a communication platform of SeMCDM in automotive applications. It has also the advantage of being independent of special resources. However it has serious disadvantages: The static configuration by the designer (the lack of dynamic extension), the restricted bandwidth, and the prioritized message transfer. It serves as a real time bus only for very short messages.

The suggestions discussed above try to avoid some of these disadvantages, but they don't promise an ultimate solution.

4.6.3 Assessment of LIN as a communication platform for SeMCDM

LIN is designed as to form sub-networks from the CAN bus. Because of its very limited bandwidth and because of the resource restrictions on the participating nodes (like sensors), LIN cannot be looked at as a realistic communication platform for SeMCDM.

4.6.4 Assessment of FlexRay as a communication platform for SeMCDM

In comparison to the CAN bus, FlexRay is surely a better alternative regarding the bandwidth question and the fairness of access (Time Division Multiple Access - TDMA). Additionally, it has better fault tolerance as described in section 2.7.3. FlexRay is also designed for deployment in automotive systems. Its flexibility in sense of dynamic messages enables the transfer of long messages (although not in the real time transfer mode). However, according to FlexRay specification [Fle05] and to the confirmation from the FlexRay consortium (see email contact in [Ema08]), the problem of dynamic extension is still an obstacle before the vision of highly flexible plug-and-play extendable systems can be realized.

4.6.5 Assessment of MOST as a communication platform for SeMCDM

MOST provides high bit rates, dynamic extension of the system (plug-and-play), fair access to the transfer medium for all devices (channel reservation), and it supports central and distributed scenarios. MOST supports also real time transfer. Moreover, its adequacy to the application environment is guaranteed. However, the typical applications of MOST are related to multimedia and infotainment devices in automobiles. Additionally, the central administration of MOST represents a single point of failure.

4.6.6 Conclusion

Table 4-7 shows a comparison between the automotive communication platforms according to the requirements of SeMCDM. Because of its advantages, MOST seems to be the best available automotive communication platform for dynamic organic architectures.

Its restriction to multimedia and infotainment applications can be positively understood as a chance to test the concepts of organic systems. As such applications are not critical to the safety or to the kern functionality of automobiles, and can serve as test applications for SeMCDM. After a successful test phase, the concepts of SeMCDM and Organic Computing can be adopted for more critical applications.

Furthermore, MOST has an important advantage because of its predefined functions and function blocks. These standardized functions are documented and organized in a "catalogue" [May07b]. It is clear that the MOST community realizes the need for standardization and for a common language. A catalogue of well understood terms is an optimal base to develop a domain specific ontology. The standardization efforts made by the MOST community performed indirectly the required prearrangements to go a step further towards SeMCDM. With MOST, automotive systems and SeMCDM find their common base and their right start point To this reason, the example domain ontology presented in section 5.3.1.3 have been designed to represents a part of MOST standardized functions.

	CAN	FlexRay	MOST
Dynamic extension	-	-	+
Bandwidth	1Mbit/s	10-20 Mbit/s	22-150Mbit/s
Fairness of access	-	+	+
Support of centralized messages exchange	-	+	+
Adequacy for the application environment	+	+	+ Multimedia /Infotainment applications)
Real time transfer	+	+	+
Distributed solution (no critical central resources)	+	+	-
Predefined functions catalogue[8]	-	-	+

Table 4-7 Assessment of automotive bus systems

[8] The availability of a predefined functions catalog is not originally expected from a communication platform. To this reason, it is not part of the requirements presented in section 4.6.1. However, this requirement is inherited from the main aim of SeMCDM, trying to integrate ontologies within the application environment and its development cycle.

4.7. Design support

The integration of the selected MCDM methods within the ontology development process is one of the important steps towards putting the suggested methodology in practical use.

Tools for feature weighting and for utility assessment have been implemented as extensions to "Protégé", a widely used free and open source ontology development environment. Protégé enables the extension of its functionalities through plug-ins. Like Protégé itself, these plug-ins are based on Java, implemented and tested with Protégé version 3.3.1. Both the weighting tool and the utility assessment tool make use of the kernel ontology depicted in section 4.2.1. The utility assessment tool relies especially on the MCDM ontology presented in section 4.2.2.

4.7.1 Features' weighting tool OntoAHP

OntoAHP is a Protégé plug-in for calculating feature weights from the estimation matrix, which contains pairwise comparisons between the features. It calculates the weights according to the AHP approach (see the Analytical Hierarchy Process in section 2.5.3). Additionally, the consistency ratio CR will be also calculated. A consistent matrix has a CR value of 0. The calculated CR value appears in green showing acceptable consistency (CR < 0.1) or in red for inconsistent matrices (CR > 0.1). For more about the origins of the value 0.1 see section 2.5.3.2.

Figure 4-12 shows OntoAHP used to give weights to the soft features of a *WishedComponent* (see the kernel ontology in section 4.2.1). The user can choose and/or add a new feature to the feature list. Already defined features can be selected using the + symbol, while new features can be defined using the * symbol (as usually done by Protégé). The matrix dimension will be automatically adapted to the given number of features. The columns and rows of the matrix are headed by the number of the corresponding feature (feature names could be too long to be shown in the headers).

The diagonal cells (where column number equals the row number) have always the value 1.0, because they represent the comparison of each feature with itself. The user can enter any number in the non-diagonal cells. Automatically the reciprocal value will be calculated and shown in the diagonally symmetric cell.

By selecting the button "Calculate" the weights and the consistency value will be calculated. The new values will be saved to the ontology by selecting the "Load" button.

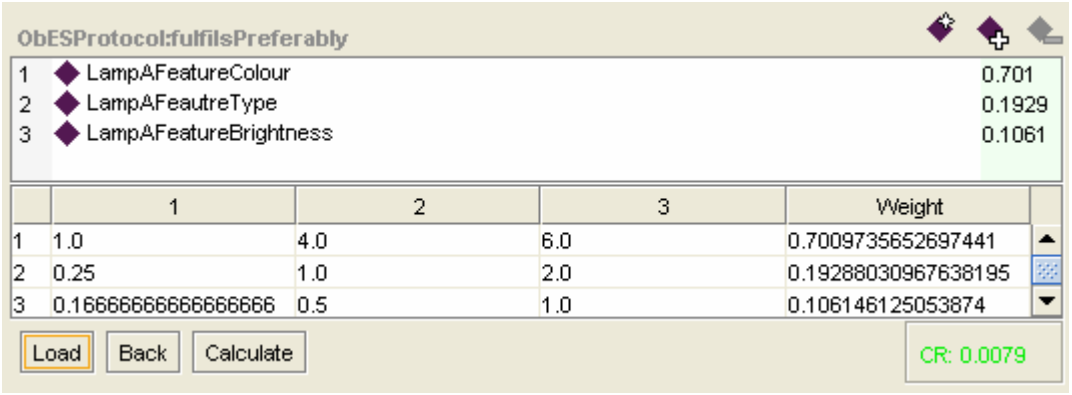


Figure 4-12 A capture of Protégé showing the weighting widget, the green value of consistency ratio CR indicates acceptable estimation matrix in terms of its consistency.

The usage of OntoAHP is not limited to the "*fulfilsPreferably*" property or to the suggested ontologies in general. Any Protégé user can benefit from it in relation to any property within the own ontology.

4.7.2 OntoUtil for the utility assessment of features

The designer can profit from the MCDM ontology (section 4.2.2) to assess the utility of his autonomous units' features. The usage of OntoUtil (Figure 4-13) follows the normal principles of ontology development: Instances of the utility function concepts can be parameterized and then assigned to features. As soon as the user selects the type of the proposed utility function (4.2.2.3), the adequate number of points appears, with empty values. The designer fills out the values, and presses the "Compute Parameters" button to get the parameters of his utility function. By pressing the "save" button, the parameters will be saved to the ontology.

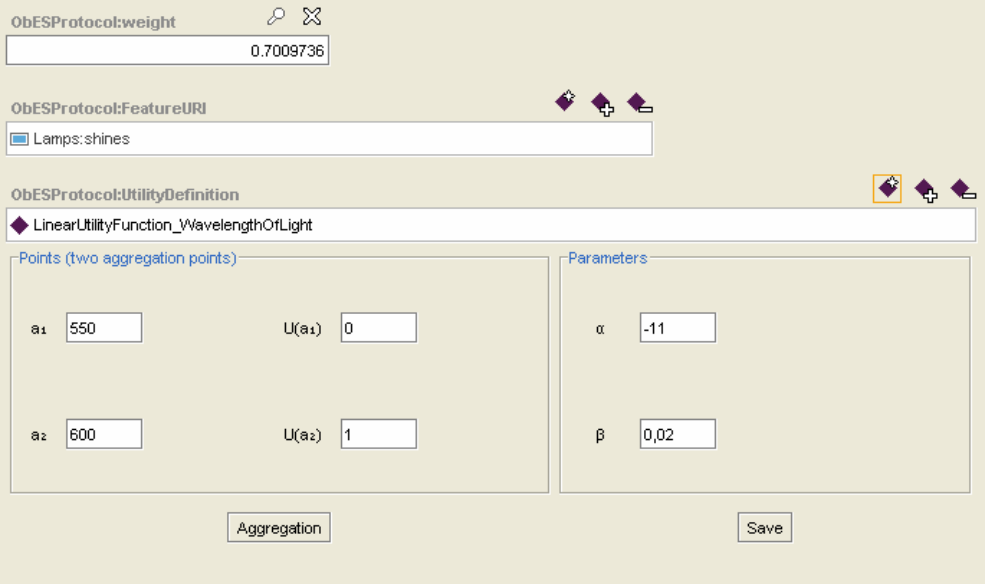


Figure 4-13 OntoUtil helps to parameterize the utility functions. The user can parameterize a linear utility function from the MCDM ontology by giving in two relevant points.

Figure 4-14 shows the role of OntoAHP and OntoUtil in the ontology-based description of the autonomous units.

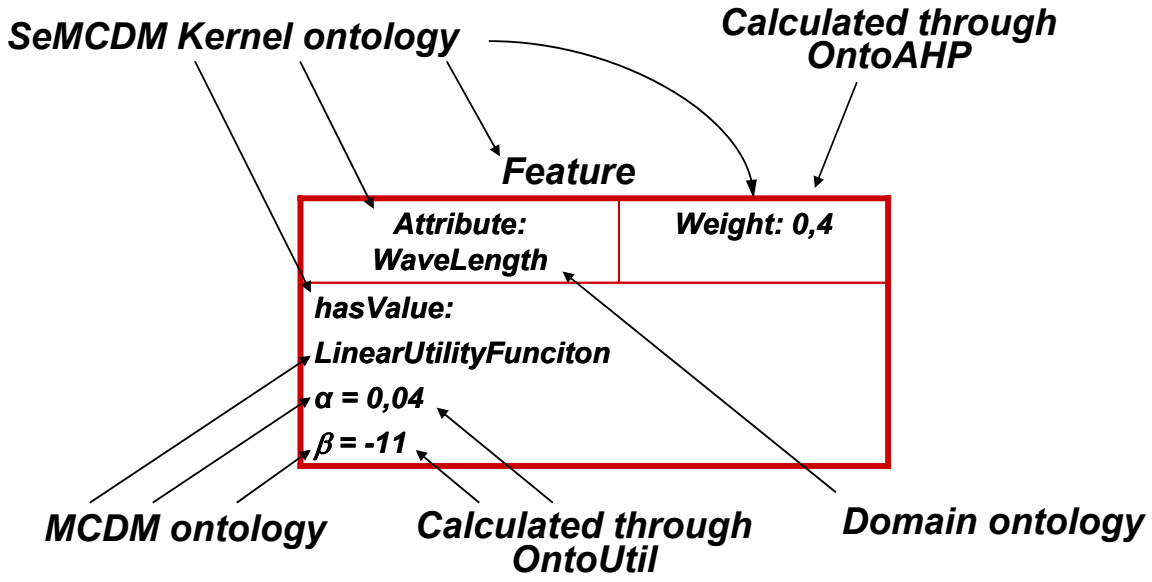


Figure 4-14 The description of autonomous units with the help of SeMCDM ontologies, OntoAHP and OntoUtil.

5. Evaluation

5.1. SeMCDM: A concept under evaluation

The SeMCDM concept has been suggested and specified a solution of the problem facing current approaches towards high adaptivity in automotive systems. Table 2-1 shows the SeMCDM in comparison with these approaches.

	DySCAS	**EvoArch**	**SeMCDM**
Components	Software tasks	ECUs	Autonomous units of any type
Behavior	Application dependent	Marketplace oriented	Marketplace oriented
Enabling technologies	- Policy-based computing - Selection through utility functions (only for quantitative values)	- Taxonomy selection (coarse selection) - No fine selection	- Ontology-based design and inference - Multi-Criteria Decision Making
Matching type	Syntactic matching between parameters (context parameters and rules parameters)	Syntactic matching between taxonomy nodes	Semantic matching
Efforts towards a common description language	No	Yes	Done
Development\Deployment support of a common language	No	No	Done
Development tools	- Policy editor (under construction) - No tools to set up the utility functions	UML editors	Extended ontology development environment to support MCDM
Implementation	A prototype on .NET	A prototype in Java	A prototype in Java

Table 5-1 SeMCDM in comparison with approaches of automotive systems.

By it design, the concept of SeMCDM has multiple advantages has multiple advantages in comparison to the other approaches: The abstraction of components to autonomous units, the common description language, the semantic matching and the supporting design tools.

The next sections will proof the functionality and the performance of SeMCDM from different points of view.

5.2. Methodology

The evaluation covers different aspects of SeMCDM. For each aspect, special methods and tools have been developed. The evaluation of SeMCDM combines the following types:

- **Functionality test**: A prototype of the autonomous units simulates their behavior on the marketplace. This prototype shows how a semantic multi-criteria decision making takes place on the base of the ontological description of autonomous units. In the functionality test the SeMCDM ontologies, OntoAHP, OntoUtil and application dependent rules have been used, in order to test the complete chain at design and runt-time. Benefits of the resulting MCDM-aware inference engine have been demonstrated on applications from the field of automotive industry, especially, on MOST (Media Oriented Systems Transport) applications. The selection of MOST as an example application is based on the argumentation from section 4.6.6.

- **Performance analysis**: The prototype of the autonomous units delivers information about the performance of the inference engine. Using the example ontology and a well-known inference engine the execution time of the SeMCDM-process has been measured. The results of this step provide the required values to perform a realistic evaluation of alternative market scenarios.
- **The evaluation of alternative market scenarios**: The run-time performance of the market scenarios (defined in section 4.5.2) can be vital to adopt one specific scenario. For the evaluation of market scenarios different evaluation metrics has been defined. The comparison between the market scenarios according to these metrics takes place on the base of simulation. To bring the simulation results close to real applications, characteristics of the application environment have been extracted. The complexity of the environment is not only reflected by the number of the autonomous units, but also by the diversity of their types and by the number and diversity of the looked-for components. The simulation results combine run-time performance of market scenarios in different environments. The evaluation of market scenarios helps to make recommendations for the engineer of self-organizing systems.

5.3. Simulation Environment

5.3.1 Prototype of the architecture

The model of the suggested architecture is designed to evaluate the applicability of the idea and to show the benefits of combining ontologies and inference to mechanisms of multi-criteria decision making.

Autonomous units have marketplace-oriented behavior (very similar to the market scenario 1 presented in section 4.5.2.1). Autonomous units are equipped with an inference engine. The used inference engine is Jena [HP08]. Jena is an open source Java framework for building Semantic Web applications. It includes a generic rule-based inference engine, besides an OWL API, and query languages (other inference engines and APIs are also supported).

The broker is implemented as a common memory space, on the based of Jada [Ros96], a Linda [Gel85] implementation. Jada is a package for Java that allows distributed Java applications to access a shared object space for coordination and data sharing purposes. The prototype deploys Jada only as a common memory space (The associative addressing capabilities of Jada allow only for syntactic matching).

The model has been implemented in Java.

5.3.1.1 Usage of the prototype

Before starting the model, autonomous units have to be defined in application ontologies. The kernel ontology, presented in section 4.2.1, can be considered as a start point for the user. The ontology can be edited for example by Protégé [Pro08]. Additionally, domain ontologies can always be gathered to the application ontology (as usual in Protégé).

Starting the application a graphical user interface (the main window of the application) appears as in Figure 5-1.

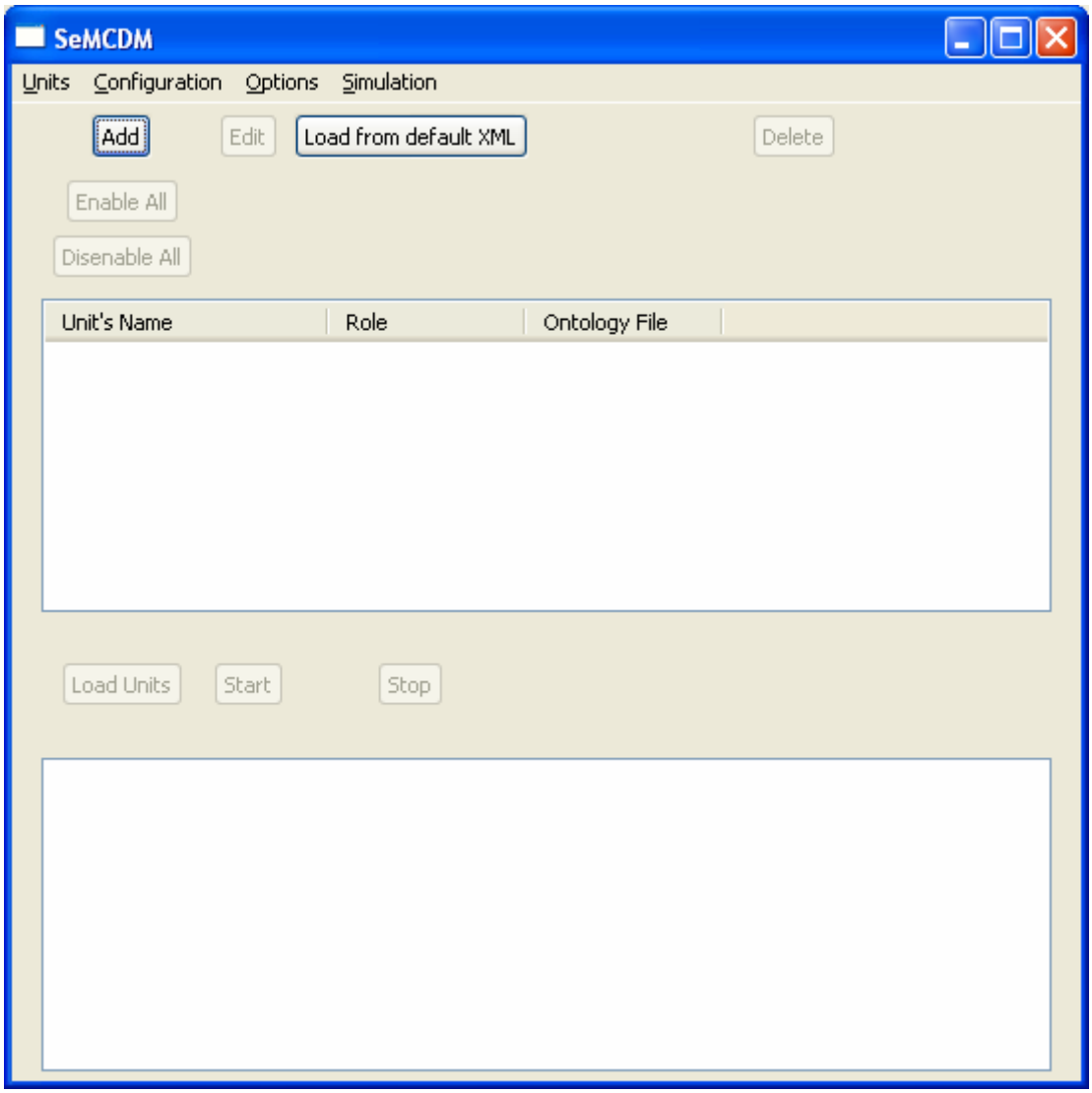


Figure 5-1 Main window of SeMCDM prototype.

The user can "add" autonomic units manually to the "Arena". In this case additional window opens, where the user can define the autonomic unit name, its role, and the ontology file name to import the units from (.owl file).

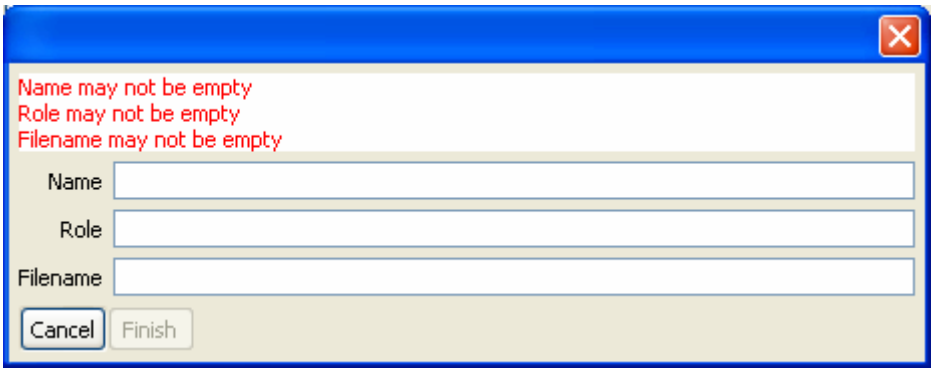


Figure 5-2 Adding an autonomous units manually.

An easier way to add autonomic units is to import the autonomic units directly from ontology files (Main window: Units->Read from->Ontology file). In this case all autonomic units defined in the ontology file will be added to the arena (the upper list in the main window). All mapped domain ontologies have also to be loaded (the user has to select the .owl files of these ontologies manually).

To avoid the manual addition of mapped ontologies, the prototype is equipped with the capability of reading "Units files". The units file is a XML file, where ontology files and their mapped (domain) ontology files can be listed. Figure 5-3 shows a typical example of units files.

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
 <OntolotgyFiles>
   <OntolotgyFile
       mainfile="C:\\MOSTUnits\\MOST_Amplifier_ST_1.owl">

      <importedFile namespace= http://www.sra.uni-
      hannover.de/rdf/Amplifiers.owl
       filename= "C:\\DomainOntologiesMOST\\Amplifiers.owl"/>

      <!–other mapped ontologies-->
   </OntolotgyFile>


   <OntolotgyFile
      mainfile="C:\\MOSTUnits\\MOST_Amplifier_ST_2.owl">

      <importedFile namespace= http://www.sra.uni-
      hannover.de/rdf/Amplifiers.owl
      filename="C: \\DomainOntologiesMOST\\Amplifiers.owl"/>

      <!–other mapped ontologies-->
   </OntolotgyFile>

   <OntolotgyFile
    mainfile="C:\\MOSTUnits\\MOST_Manager.owl">
      <importedFile namespace= http://www.sra.uni-
      hannover.de/rdf/SignalProcessingTechnologies.owl
      filename="C:\\DomainOntologiesMOST\\SignalProcessingTechnologies.o
      wl"/>
      <!–other mapped ontologies-->
    </OntolotgyFile>

 </OntolotgyFiles>
</root>
```

Figure 5-3 An example of a unit file

As soon as the autonomous units appear in the arena list, the user can activate/deactivate them, load them (instances of autonomous units will be generated by loading). The market place oriented behavior can be triggered by pressing the start button.

The user can stop the activities on the arena, by pressing the stop button.

The resulted configuration of the system (the contracts made between the autonomous units) can be seen as a "Tree", in the lower part of the main window (see Figure 5-1).

5.3.1.2 Implementation of utility check in inference rules

To equip the inference engine with the capabilities of Semantic Multi-Criteria Decision Making, a set of logic rules have been defined. These inference rules implement the utility check as a part of the generalized matching process (see Table 4-4).

Figure 5-4 shows the utility check rule between two multi point utility functions (the offer's utility function is usually of sub-type: *OnePointMaxUtilityFunction*. See the discussion in

section 5.2.3.1). The most important part of the rule is the equality check between two values. This equality check is supported by Jena.

```
[hasCommonValuesWithMultiPointsUtilityFunctionMultiPointUtilityFunction:
      (?a rdf:type pre:MultiPointUtilityFunction) (?b rdf:type
      pre:MultiPointUtilityFunction)
      (?a pre:point ?pa)(?pa pre:point_x ?pax)
      (?b pre:point ?pb)(?pb pre:point_x ?pbx)
      equal(?pax,?pbx)
      -> (?a pre:hasCommonValuesWith ?b)
      ]
```

Figure 5-4 Rule for utility check of two multi point utility functions.

The property *hasCommonValuesWith* gather two utility functions, and has the semantic: There are common values between them.

For other pairs of utility function, the utility check requires utility calculation beyond the standard mathematical support of Jena, and inference engine in general. It is clear that the logical inference engines are not designed to cope with the needs of utility check. But they offer the possibilities of adding user extensions. For Jena, these extensions are called as Built-Ins. Built-ins are java classes (with a special interface), which can be added while initialising the inference engine.

For purposes of utility check between different pairs of utility functions a set of built-ins and rules has been implemented to extend the capability of the inference engine.

Figure 5-5shows the rule for utility check between: A *MultiPointUtilityFunction* and a continuous function (*CubicUtilityFunction*).

```
[MultiPointUFhasCommonValuesWithCubicUF:
      (?a rdf:type pre:MultiPointUtilityFunction) (?b rdf:type
      pre:CubicUtilityFunction)
      (?a pre:point ?pa)(?pa pre:point_x ?pax)
      (?b pre:parameter_1_alpha ?alpha)
      (?b pre:parameter_2_beta ?beta)
      (?b pre:parameter_3_gamma ?gamma)
      (?b pre:parameter_4_delta ?delta)
      utilityOfxInCubicFunctionFallsBetweenNullAndOne(?alpha,?beta,?gamm
      a,?delta,?pax)
      -> (?a pre:hasCommonValuesWith ?b)
      ]
```

Figure 5-5 Rule for utility check between a multi point utility function and a cubic function.

The built-in *utilityOfxInCubicFunctionFallsBetweenNullAndOne* is an extension of the logical inference capability. Similar built-ins have been added for other pairs of utility functions.

Utility check between a multi point utility function and an interval utility function is supported by the rule shown in Figure 5-6.

```
[MultiPointUFhasCommonValuesWithIntervalUF:
      (?b rdf:type pre:IntervalUtilityFunction)
      (?b, pre:hasfunction, pre:unityFunction)
      (?b, pre:hasRange, ?rangevalue)
      equal(?rangevalue, pre:greaterThan)
      (?b pre:intervallimits ?intervallimitpoint) (?intervallimitpoint
      pre:point_x ?px)
      (?a rdf:type pre:MultiPointUtilityFunction)
      (?a pre:point ?pa)(?pa pre:point_x ?pax)
      ge(?pax,?px)
      ->
      (?a pre:hasCommonValuesWith ?b)
      ]
```

Figure 5-6 Rule for utility check between a multi point utility function and an interval utility function.

The utility check between qualitative values (Likert-scaled utility valued) is implemented by the help of the rule in Figure 5-7.

```
[LikertScaleUFhasCommonValuesWithLikertScaleUF:
      (?a rdf:type pre:LikertScaleUtilityFunction) (?b rdf:type
      pre:LikertScaleUtilityFunction)
      PrintingBuiltin(?a)
      PrintingBuiltin(?b)
      (?a pre:hasLikertScaledPoint ?likertscaledpoint_a)   (?b
      pre:hasLikertScaledPoint ?likertscaledpoint_b)
      (?likertscaledpoint_a pre:physicalValue ?aphysicalvalue)
      (?likertscaledpoint_b pre:physicalValue ?bphysicalvalue)
      equal(?aphysicalvalue,?bphysicalvalue)
      -> (?a pre:hasCommonValuesWith ?b)
      print(?a,?b)
      (?b pre:hasCommonValuesWith ?a)]
```

Figure 5-7 Rule for utility check between two Likert scaled functions (qualitative values).

5.3.1.3 Domain ontologies

As shown in section 4.6, the Media Oriented Systems Transport (MOST) standard provides a suitable background for first-steps towards ontological description. A special ontology for automotive communication platforms has been developed. Figure 5-8 shows this ontology and its relation to MOST devices.

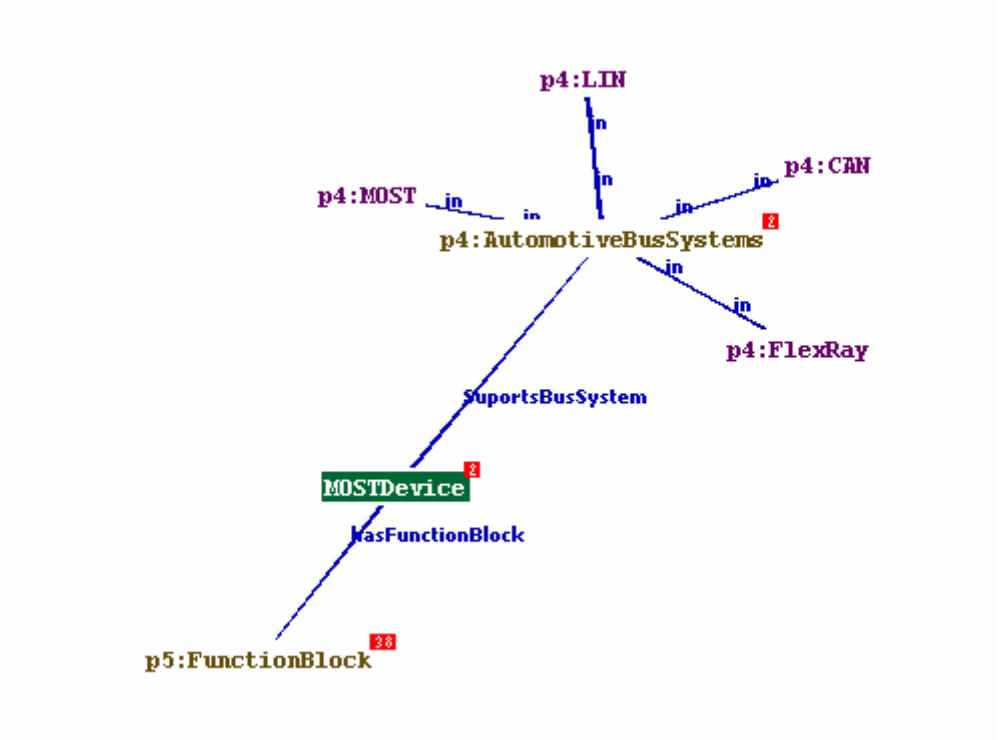


Figure 5-8 MOST device in relation to the functions blocks and to the automotive bus systems.

In addition to the general ontology of automotive communication platform, a special ontology for MOST has been designed to gather the MOST function blocks (as can bee found in MOST specifications [MC06]).

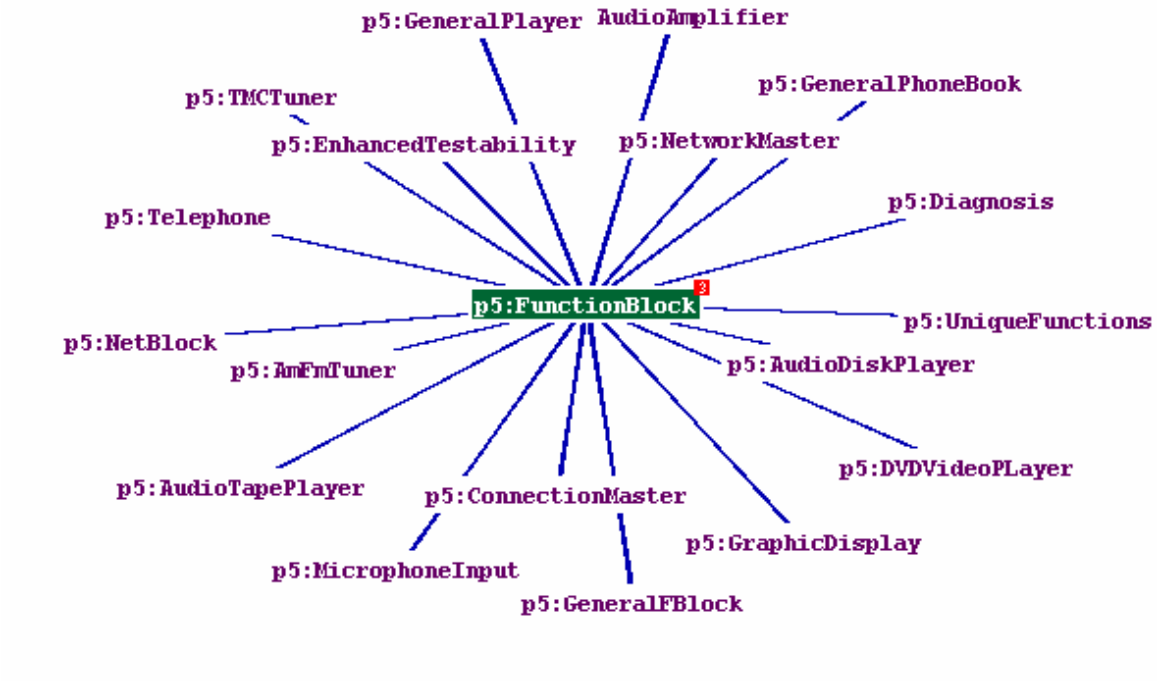


Figure 5-9 Ontology of MOST function blocks as defined for MOST devices

A set of supporting "background" domain ontologies has been also developed. Figure 5-10 shows a part of these domain ontologies. This set of ontologies contains an ontology of electronic technology, an ontology of signal processing technologies and an ontology of electronic devices.

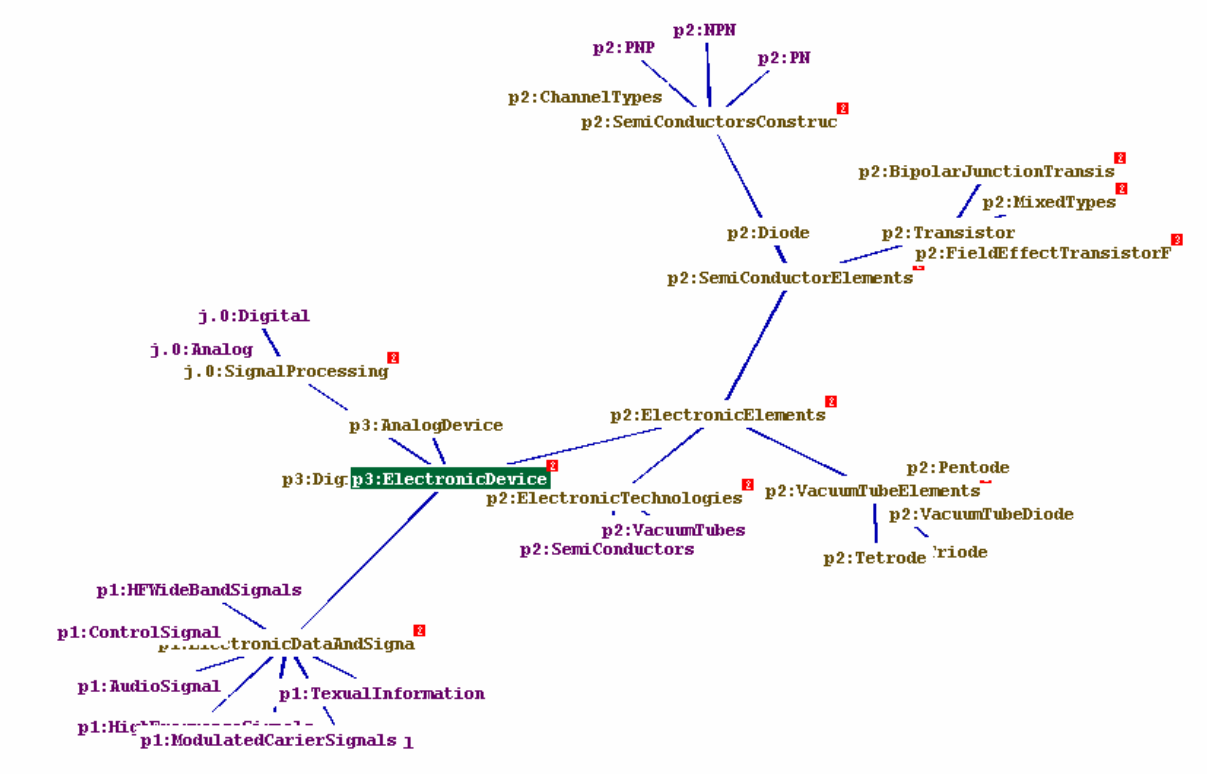


Figure 5-10 Part of the domain ontologies

On the base of the background domain ontologies, special domain ontologies have been built, like the ontologies of multimedia devices (amplifiers, tuners, displays, data storage devices, etc.).

These sets of domain ontologies (beside the MOST ontology) enable the description of MOST devices as autonomous units.

5.3.1.4 Application specific rules

The semantic matching can be extended by application specific rules. In addition to the semantic matching supported through the use of a common ontology, the user can define rules, which enhance the matching capabilities. An example of such rules is depicted in Figure 5-11.

```
[basedAroundSomeThingMeansItHasTheSameTechnology:
      (?a rdf:type ElectronicDevices:ElectronicDevice)
      (?a ElectronicDevices:basedAroundElement ?element)
      (?C rdf:type owl:Restriction), (?C owl:onProperty
      ElectronicTechnologies:hasElectronicTechnology)
      (?element rdf:type ?C)
      (?C owl:hasValue ?D)
      (?a rdf:type pre:Component)
      ->
      addFeatureToComponent
      (?a, ElectronicDevices:hasElectronicTechnology,?D)
]
```

Figure 5-11 Application specific rule discovers the technology of a MOST device on the base of the technology of its main element.

The rule enhances the semantic matching in the sense that it say: If a device is based around an element, which has a specific electronic technology, then the electronic device itself should

has the same electronic technology. An electronic technology is defined (in a special ontology) to be either a semiconductors technology or a vacuum tubes technology. Elements are semiconductors elements like diodes and transistors (Bipolar BJT, FET, or Mixed); or vacuum tubes elements (Triode, Pentode, Tetrode, etc.).

5.3.1.5 Examples of SeMCDM

Many examples and configurations of autonomic units have showed the benefits of the semantic MCDM matching process.

In Figure 5-12, a MOST manager (an active autonomous unit) looks for an amplifier (a passive autonomous unit). Two amplifier units are available on the marketplace. The MOST manager prefers an amplifier based on vacuum tube technology over a semi conductor amplifier (Likert scale of qualitative values), although both amplifiers are acceptable offers.

The fact that the preferred amplifier is based on vacuum tube technology is not asserted explicitly in the ontology. The only known fact about this amplifier is that it is based around "Pentode_6K6GT", an instance of the concept "Pentode". "Pentode" in its role is defined in the ontology of electronic technologies. Such matching result would not be possible, without an application specific logical rule. The rule presented in Figure 5-11, solves the matching problem, enabling optimal matching on the base on "better knowledge".

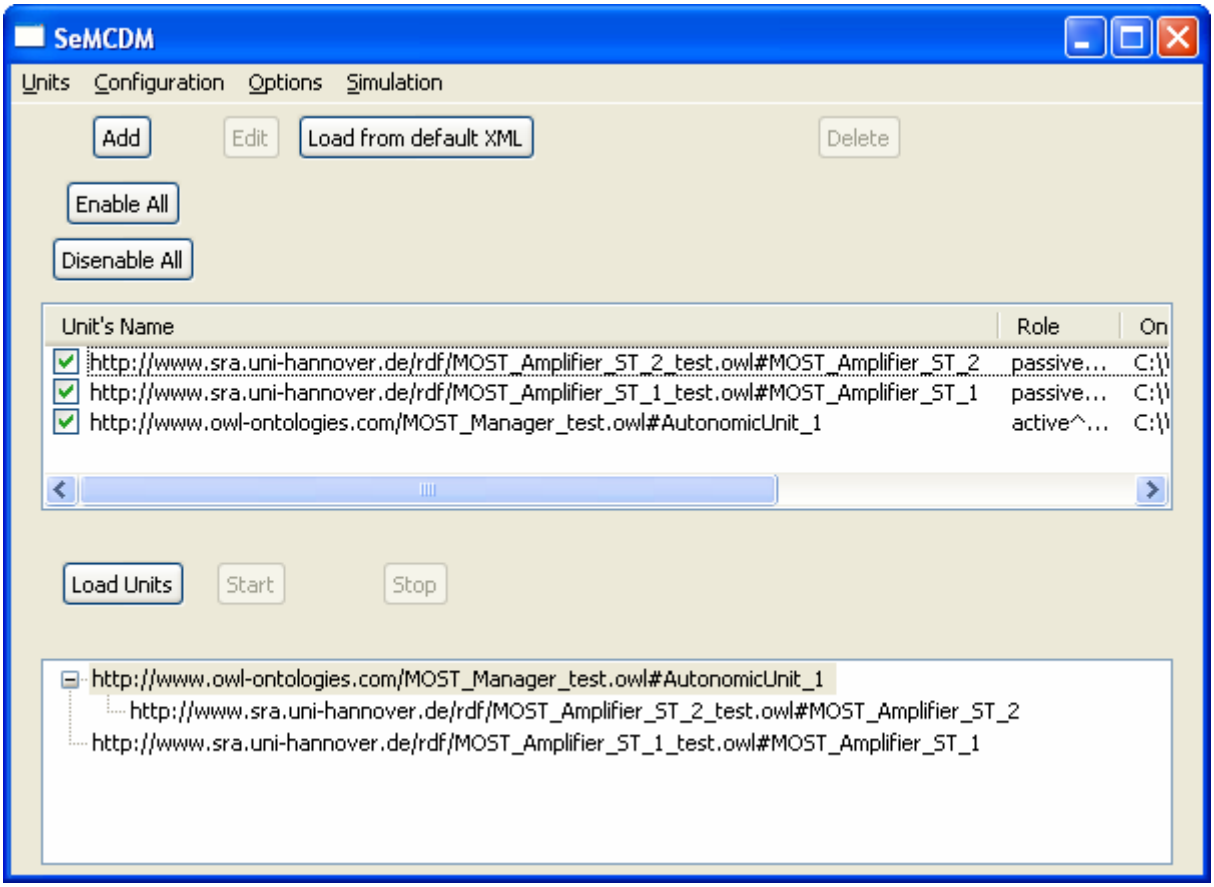


Figure 5-12 The configuration of an example system as a result of the semantic matching and application specific rules.

Different examples have been developed, where the multi-criteria decision making plays a main role of driving the matching results towards specific solutions. The examples gathers combinations of continuous utility functions, Likert scaled utility functions and interval utility functions (covering different matching rules between these utility functions).

The used example features are defined in the ontologies mentioned above. For example MOST amplifiers have features like: Gain, noise, its electronic technology, its signal processing technology (Analog\Digital), beside support of the MOST bus and of the "AudioAmplifier" function block (from the MOST otology).

A typical description of a wished component can be seen in Figure 5-13. Combinations of hard and soft features, as well as weights of the soft features have been a subject of experiments. Features like "SupportBusSystemFeature" and "hasFunctionBlockFeatures" are usually used as hard features. The electronic technology and the gain reflect fine preferences of the designer. The feature "SignalProcessingTechFeature" can play a key role, because it has a great influence on the ranking values (it is a hard feature. See section 4.1.2.5).

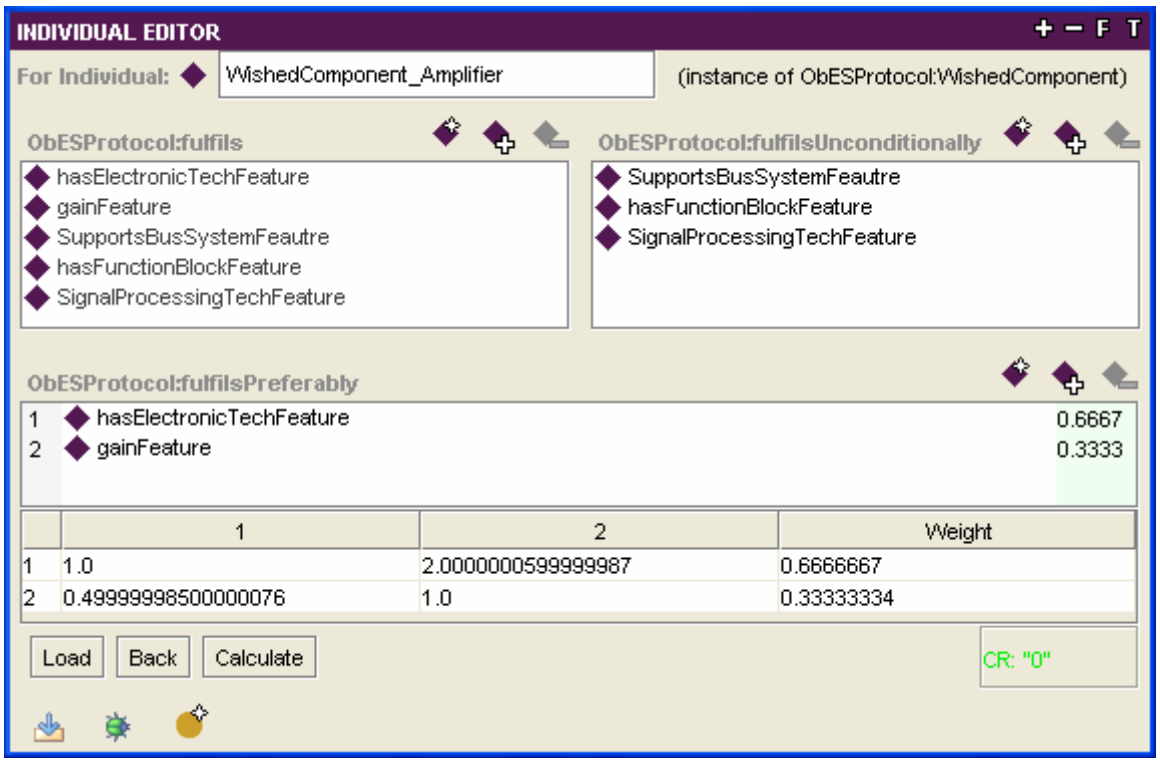


Figure 5-13 Typical description of a wished component.

5.4. Matching time

The prototype of the autonomous units delivers also information about execution time of the both matching steps. For this purpose, the SeMCDM ontology and domain ontologies from the field of automotive engineering have been deployed to define several autonomous units. The prototype is a Java application, which uses the Jena inference engine with SeMCDM matching rules.

The experiment has been performed on an AMD Athlon (tm) 64 Processor 3200+, with 1 GB RAM and Microsoft Windows XP operating system.

Throughout all experiments, the inferred ontology models (in Ontology Web Language OWL [13]) have the size of 3183 RDF statements (Resource Description Framework [14]) for the passive units and 3133 for the active units.

Passive units have been described with the help of 6 features, while enquires combined 5 features (3 hard features and 2 soft features). The features have been specified with different kinds of utility functions.

The measured execution time of the first matching step is about 31 ms, while the execution time of the second matching step is about 16,7 ms. These results are measured in nanoseconds and apply of each pair of (offer, enquiry).

The execution time of the matching steps are not to be confused with the preparation time (forward chaining) of the ontology models, which may last several seconds (4 - 5 s).

5.5. Evaluation of alternative market scenarios

The aim of the simulation-based evaluation in this section is to assess the performance of the suggested market scenarios (section 4.5.2) in different settings of the application environment. The simulation models of the market scenarios have been built in Java, so that every unit has the form of a thread, in addition to a broker thread. The fairness between the autonomous units is guaranteed through the equal sleep time of the threads and the random selection of offers/enquiries from the central broker.

Real applications of SeMCDM vary between the building of a whole system consisting of a high number of heterogeneous autonomous units and the search for a substitution of one failing autonomous unit a in a merely homogenous environment. In this context, the complexity of the application environment is not only reflected by the number of the autonomous units, but also by the diversity of their types and by the number and diversity of the looked-for components.

Therefore, the simulation model has been designed to supports the modification of the next parameters:

- Number and type of existing passive units
- Number and type of existing active units, in addition to the number and type of the components in their wish lists.
- Computing load resulted from the first matching step and from the second matching step. The computing load takes the form of delay time.
- Enquiry time out: The expiration time of the enquiry.
- Cycle time: Defines the start time of the next cycle.

In following sections, theses parameters will be categorized to form settings of the application environment (section 5.5.2) and settings of the computing load and timing (section 5.5.3).

However, the next section presents a set of assessment metrics of the market scenarios.

5.5.1 Assessment criteria of the market scenarios

Different assessment metrics have been defined and measured for each of the market scenarios: The success rate and the time to first solution, the number of matching attempts and the quality of solution. The next sections describe each of these metrics.

5.5.1.1 Success rate and time to first solution

A “solution” is a possible configuration of the application system. A configuration consists of a set of contracts between autonomous units. A first solution is reached when at least one suitable offer for every looked-for component has been found. This leads to the definition of the “success rate” as “the ratio of found components to the total number of looked-for components”. A success rate has the maximal value of 1, which means that all looked-for components have got passive units assigned.

In the simulation settings with only 15 passive units available, the maximum value of the success rate is limited to 0.5 (there are always 30 looked-for components). In other settings the maximum value of 1 can be reached. The concurrence between the active units on the available resources (passive units) and the reservation mechanism yield that one cycle is not always sufficient to reach the maximal possible success rate. The time to the first solution can therefore be evaluated as the number of the needed cycles to reach the first solution.

5.5.1.2 Number of matching attempts

Scenarios differ in the number of matching attempts carried out to reach the first solution. In the results' diagrams the total numbers of matching attempts are depicted for every scenario. The total number of matching attempts is the sum of matching attempts in all cycles.

5.5.1.3 Quality of solution

An optimal solution is that, where every looked-for component meets its corresponding optimal offer. The first solution found is not necessarily the optimal solution. This means that it contains acceptable offers, which differ from the optimal offer in their sub-types. The quality of solution is defined as the "distance" between the achieved solution and the optimal solution. To get a numeric impression of the quality of solution for each pair (enquiry, selected offer) the following formula has been used (subtypes are integer numbers), while the average value for all pairs represents the quality of found solution:

Quality (enquiry, selected offer) = 1 – (subtype of enquiry – subtype of offer) / total number of subtypes

5.5.2 Settings of the application environment

To cover the variety of application environments, carefully selected values of the simulation parameters have been gathered to 4 sets, representing 4 different application environments of different complexity degrees.

Through all tests the number of looked-for components is fixed to 30, while the number of passive units took one of three values: 15, 30 or 60. Table 5-2 summarizes all simulation settings.

	#of Types	# of passive units	# of passive units per type	# of components	# of components per type = # of sub-types	# of components per active unit	# of active units
Single I	1	15/ 30/ 60	15/30/60	30	30	1	30
Single II	1	15/ 30/ 60	15/30/60	30	30	5	6
Multi I	5	15/ 30/ 60	3 / 6 /12	30	6	5	6
Multi II	15	15/ 30/ 60	1 / 2 / 4	30	2	5	6

Table 5-2 Simulation settings of the application environments.

The following sections describe each configuration with more details. However, the simulation results are presented only for a sub-set of these environments settings. Section 5.5.2.5 justifies the selection of this sub-set.

5.5.2.1 Application environment "Single I"

Single I represents a very homogeneous application environment, where all passive units are of the same type[9]. Single I is characterized by the next facts:

- The passive units are of the same type: All passive units share a common set of features with the same feature values.
- Passive units of the same type differ in their sub-types: Each passive unit has additional features with specific values. The features values of these additional features differ between the passive units of the same type.
- Every active unit has only one looked-for component in its wish list.
- The looked-for components share the same type with the passive units: The common features of the passive units are exactly those counted under the hard features of the looked-for component.
- Looked-for components of different active units are of different sub-types: As they have different values of their soft features.

5.5.2.2 Application environment "Single II"

Single II is also a homogeneous application environment, similar to Single I. However Single I and Single II differ in the next point:

- Every active unit looks for five components of the same type.
- The looked-for components of one active unit have different sub-types.

5.5.2.3 Application environment "Multi I"

Multi I represents a heterogeneous application environment characterized by the next facts:

- There are 5 types of passive units and of looked-for components.
- Passive units of the same type have different sub-types.
- Every active unit is looking for 5 components
- Looked for components in each wish list are of different types.
- Looked-for components of the same type (in the wish lists of different active units) have different sub-types.

5.5.2.4 Application environment "Multi II"

Multi II is also a heterogeneous application environment, similar to Multi I, with the only difference that the application environment combines 15 types of passive units and of looked-for components.

5.5.2.5 Environment settings for better results comparability

For purposes of better comparability the simulation results will be presented only for environment settings with 30 and 60 passive units. In this sub-set of the simulated

[9] The word "type" is used to describe either a "partial set of the features and their specific values" when related to passive units or "the hard features with their specific values" when related to looked-for components. The word "sub-type" is used to describe either "all other features not ranked among the set of type's features" when related to passive units or "the soft features" when related to looked-for components. A sub-type is also specified with the values of its features' set.

environment settings, a maximal success rate of 1 is expected and there is at least one optimal offer for every looked-for component. This means that these environments are better comparable in relation to the metrics: Time to first solution and quality of solution.

Additionally, the effect of environment on the results can be seen clearly be considering the extreme settings: Single I and Multi II. Therefore, the simulation results will be presented for four comparable and useful environment settings:

- **Homogeneous environment**: A special case of Single I with 30 available passive units.
- **Easy homogeneous environment**: A special case of Single I with 60 available passive units.
- **Heterogeneous environment**: A special case of Multi II with 30 available passive units.
- **Easy heterogeneous environment**: A special case of Multi II with 60 available passive units.

5.5.3 Settings of the computing load and timing parameters

On the base of the measured execution time of both matching steps, computing load has been simulated for each pair (enquiry, offer) with the following values: The computing load of the semantic matching is set to 50 ms, and for the MCDM process to 25 ms.

These values are not equal to the measured values, because the measured execution time represents a subjective value and is strongly related to the different factors, like the complication of the ontology, the number of describing features (on both sides: The offer and the enquiry), the type of the utility functions, and the number of rules (matching rules between utility functions and application specific rules). However the ratio between the execution time of the first and the second matching step is still stable (the simulation result will show that the evaluation of the scenarios is not related to the execution time of the matching steps. See section 5.5.5).

The expiration time of the enquiry has the value of 20 s for all distributed scenarios, with cycle time of 30 s. The third scenario has exceptional values, because of the high number of matching attempts: An expiration time of enquiry of 120 s with a cycle time of 200 s.

5.5.4 Simulation results of the market scenarios

5.5.4.1 Success rate and time to first solution

The next sections depict the success rate diagrams and the time to first solution for all scenarios in each of the simulation settings (Section 5.5.2.5).

5.5.4.1.1 The homogeneous environment

Figure 5-14 shows the success rates of each scenario in relation to the time when running in the homogeneous environment. To reach the first solution 6 cycles of scenario 1 have to be carried out. The fact that more than one offer can be reserved for one enquiry means that other enquiries have to wait for a following cycle to get an offer (30 passive units face exactly 30 looked-for components). The homogeneous environment represents a worst-case for scenario 1 because all looked-for components and all passive units are of the same type.

The best scenario for the homogeneous environment is scenario 2B, which reaches the first solution within the first cycle. In comparison to 2B, scenario 2A delivers a moderate

performance (time to first solution of 3 cycles). This difference can be traced back to the fact that scenario 2B avoids the reservation of multiple offers for one enquiry.

According to scenario 3, contracts can be concluded only after achieving both matching steps for all possible pairs (offer, enquiry). This explains the late rise of the success rate values in comparison to other scenarios.

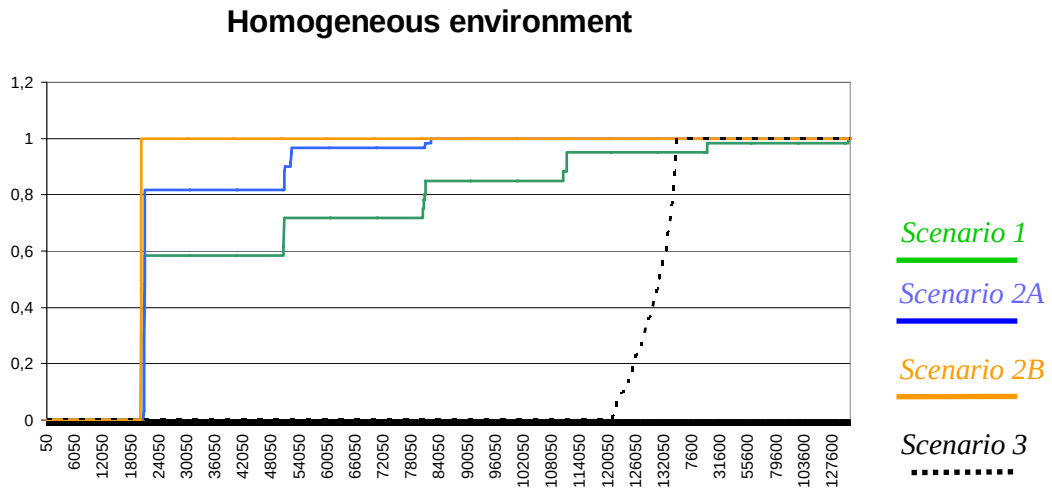


Figure 5-14 Success rates of the market scenarios in the homogeneous environment.

The success rate line of scenario 3 is not linearly increasing. This is due to the decreasing number of available offers for each enquiry along with the number of already processed enquiries.

5.5.4.1.2 Easy homogeneous environment

The easy homogeneous environment is relatively a friendly environment because of the high number of available passive units. Therefore, all distributed scenarios show shorter times to the first solution (see Figure 5-15).

Only scenario 3 requires a longer time to find the first solution in its second matching step. This indicates that a high number of candidates can be a real problem for the centralized matching.

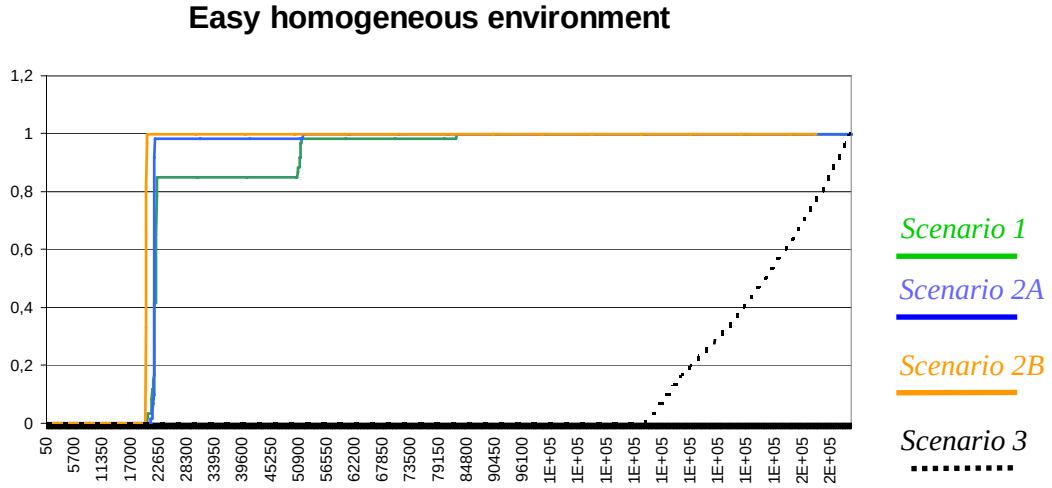


Figure 5-15 Success rates of the market scenarios in the easy homogeneous environment.

5.5.4.1.3 The heterogeneous environment

In contrast to its bad performance in the homogeneous environment, scenario 1 reaches the first solution in only 2 cycles (see Figure 5-16). The heterogeneous environment is not a friendly environment, but it is obvious that scenario 1 benefits from the distribution of the search processes (the first matching step) on a high number of passive units. The rarity of the looked-for components hides the disadvantage of scenario 1 (i.e. the offer reservation problem).

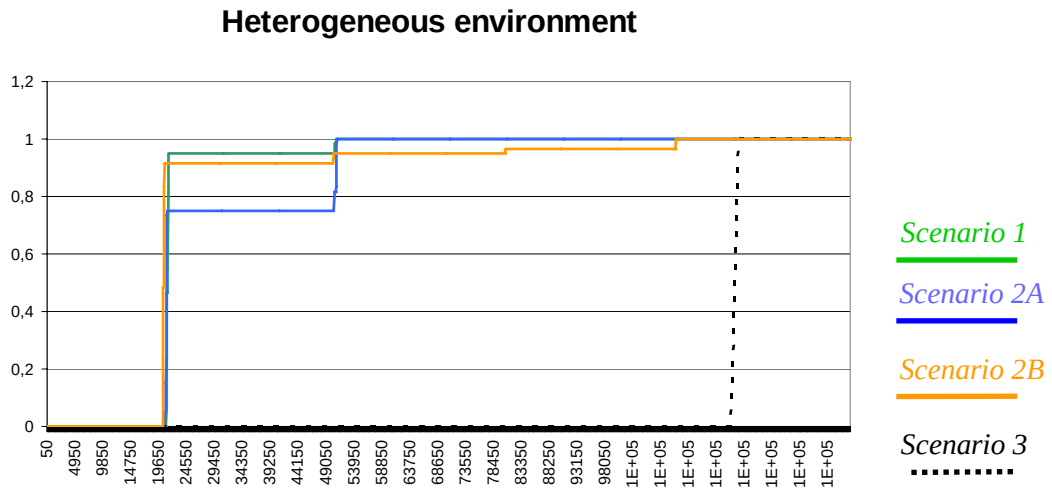


Figure 5-16 Success rates of the market scenarios in the heterogeneous environment.

Scenarios 2A and 2B show a weakness in the heterogeneous environment (both distribute the difficult search process on 6 active units). Scenario 2A shares principally the same difficulties with scenario 2B, but it can reach the first solution in two cycles (scenario 2B reserves offers for both matching steps while scenario 2A reserves them only for the first matching step).

The second matching step in scenario 3 shows a steeply ascending line (small number of alternatives).

5.5.4.1.4 The easy heterogeneous environment

In scenario 1, 60 passive autonomous units complete the first matching step in parallel. Due to the higher number of passive units, the success rate in the first cycle reaches a better value (near to 1) in comparison to its value in heterogeneous environment, and two cycles are sufficient to reach the first solution (see Figure 5-17). In spite of additional passive units, in comparison to the heterogeneous environment, two cycles are required anyway to reach the first solution by scenario 1.

Even scenarios 2A and 2B deliver better results, but only because of the higher number of available offers, where the reservation time plays only a limited role.

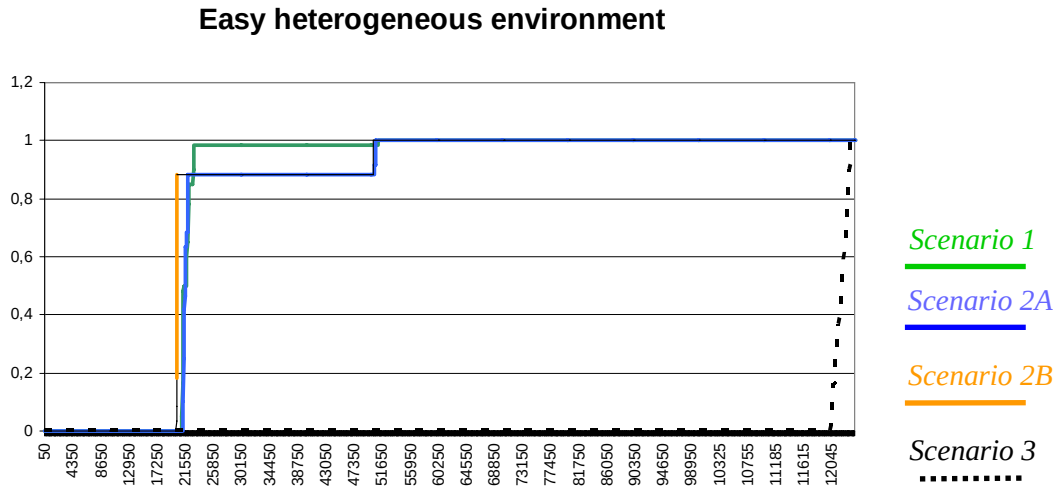


Figure 5-17 Success rate of the market scenarios in the easy heterogeneous environment.

The fact that more alternatives are subject to comparison in the second matching step can be seen from the less steeply ascending line of scenario 3.

5.5.4.2 Number of matching attempts

In the central scenario (scenario 3) all possible matching attempts are to be attended, and therefore this scenario is selected to serve as a reference for all other scenarios in all settings. In scenario 3 the central broker achieves (30 * 30 =) 900 matching attempts in homogeneous environment and in the heterogeneous environment In the easy homogeneous environment and in the easy heterogeneous environment the number of matching attempts in the central broker reaches the value of (30 * 60 =) 1800. The following section present the number of matching attempts of all scenarios as a percentage fraction of matching attempts in scenario 3.

5.5.4.2.1 The homogeneous environment

Figure 5-18 shows the number of matching attempts achieved by the market scenarios in the homogeneous environment. In relation to scenario 1, the passive units perform the first matching step 60 times (6,6%). In the first cycle 30 matching attempts are made by all 30 passive units in parallel. No additional attempts are required, nor possible, in the same cycle because all attempts return positive results (passive units and looked-for components are all of the same type in the homogeneous environment). The other 30 matching attempts are then performed in several additional cycles (see Figure 2-1 of the success rate).

The active units have to try the matching 108 time (12%) before reaching the first solution, when adopting scenario 2A. To read an offer from the broker without reservation (through the first matching step) means that the same offer can be a subject of matching in more than one active unit at the same time. This concurrence situation explains that in the first cycle 94,5 matching attempts have been carried out in scenario 2A (the number of matching attempts exceeds the number of available offers, although all offers are acceptable offers in this setting). Concurrence situations are discovered and solved in the same cycle according to scenario 2A, and in different cycles according to scenario 1. The amount of effort in each scenario is reflected in the achieved success rates shown in Figure 2-1.

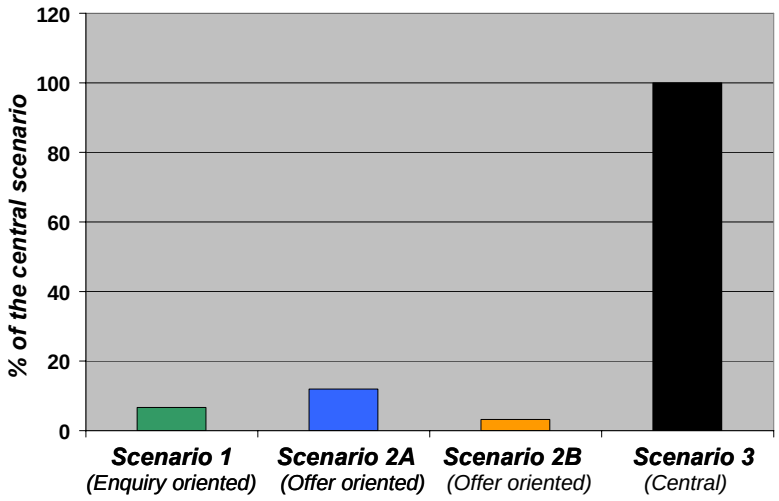


Figure 5-18 Number of matching attempts achieved by the market scenarios in the homogeneous environment.

Scenario 2B is designed to avoid the drawback found in scenario 2A, i.e. the reservation of multiple (acceptable) offers within a cycle. So it is able to get by with 30 matching attempts (3,3%) within only one cycle.

5.5.4.2.2 The easy homogeneous environment

Figure 5-19 shows the number of matching attempts achieved by the market scenarios in the easy homogeneous environment. In scenario 1, more passive units (60) attempt to find suitable enquiries, and therefore 60 matching attempts are carried out in the first cycle. In total 93 attempts (5,2% of possible matching attempts) are required to reach the first solution in the third cycle.

The number of carried out matching attempts in scenario 2A is directly related to the number of available passive units (a comparison between the homogeneous environment and the easy homogeneous environment).

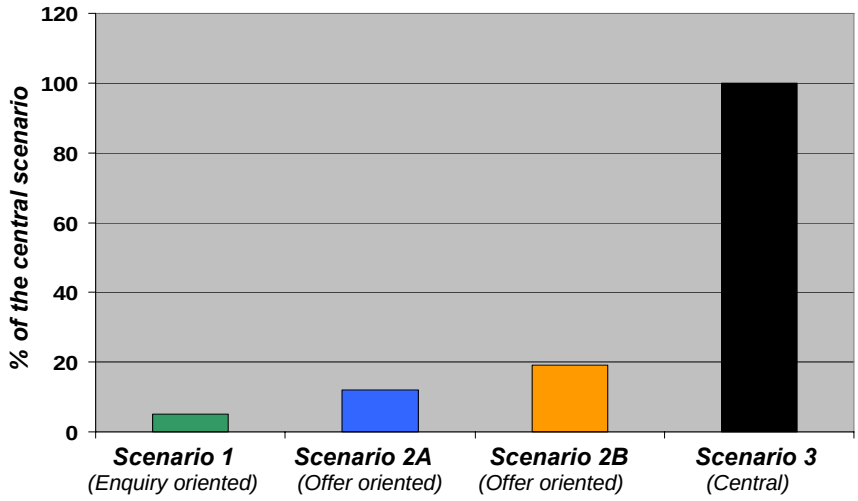


Figure 5-19 Number of matching attempts achieved by the market scenarios in the easy homogeneous environment.

Scenario 2B finds the first solution in only one cycle, but at a price of 345 matching attempts (19,2% of possible matching attempts). In scenario 2A, the offer reservation makes additional matching attempts only in a following cycle possible. In scenario 2B, free offers are always

available (in easy homogeneous environment), because each active units reserves only one offer.

5.5.4.2.3 The heterogeneous environment

Figure 5-20 shows the number of matching attempts achieved by the market scenarios in the heterogeneous environment. The rarity of the looked-for components is the main reason for higher numbers of matching attempts by all distributed scenarios. For scenario 1, 385 attempts (42,7%) have been achieved in two cycles (only 2 attempts in the second cycle), i.e. in order to find suitable offers for the last few enquiries, an enormous number of matching attempts has to be carried out. Keeping in mind that passive units try to find only one suitable enquiry and that there is actually a suitable offer for each looked-for component (according to the selected settings) one discovers that there is a drawback distributing the enquiries on the passive units. The distribution is performed by the broker in a random way. So the search is not as perfect as it could be, if the broker tried to choose enquiries with no available offers at first. An important indicator for this drawback can be found in the second cycle where the number of enquiries is much smaller, and the random selection is not any more a real problem. A similar problem can be expected also by other distributed scenarios, but it cannot be shown as clearly as in scenario 1, because other scenarios try making additional matching attempts.

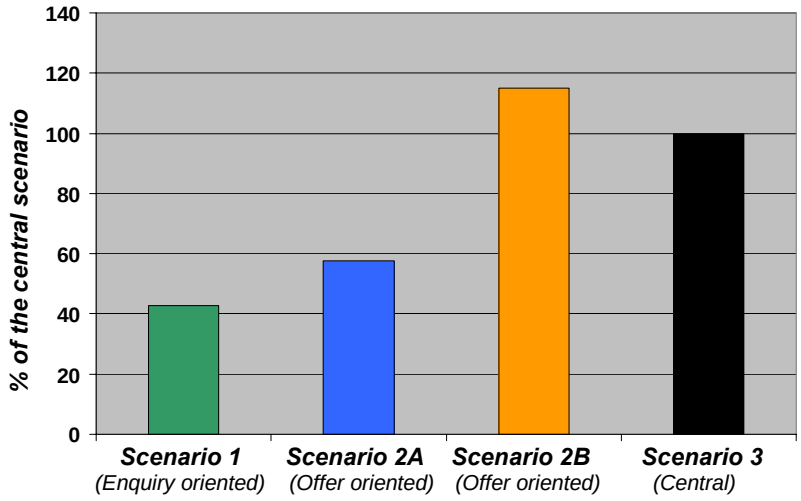


Figure 5-20 Number of matching attempts achieved by the market scenarios in the heterogeneous environment.

A dramatic worsening can be observed especially in scenario 2B, which reaches the first solution after 1035 matching attempts (115% of possible matching attempts). This high number of matching attempts has to be explained in correlation with the success rate diagram (Figure 5-16), which shows how scenario 2B needs multiple cycles to reach the first solution (because of the too long reservation time of rare offers).

5.5.4.2.4 The easy heterogeneous environment

Figure 5-21 shows the number of matching attempts achieved by the market scenarios in the easy heterogeneous environment. Scenarios 1 and 2A show similar results to those achieved in the heterogeneous environment. This reappoints the relation between the number of available offers (passive units) and the number of required matching attempts in both scenarios. Scenario 2B indicates the relative ability to deal with heterogeneous environments, when the looked-for components are not as rare as in heterogeneous environment.

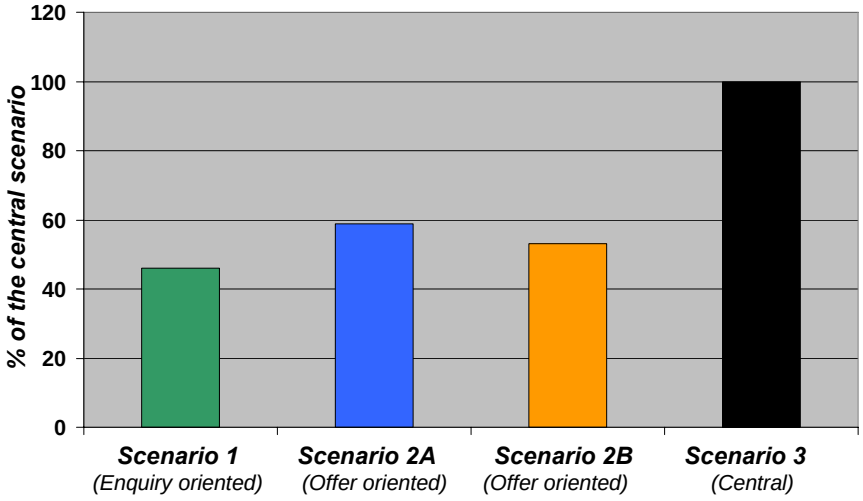


Figure 5-21 Number of matching attempts achieved by the market scenarios in the easy heterogeneous environment.

5.5.4.3 Quality of solution

Figure 5-22 shows the quality of solution achieved by the market scenarios in different application environments. The central solution proves its excellent ability of discovering the optimal solution, independent of the simulation settings. This goes back to the fact, that the central scenario verifies actually all possible pairs (enquiry, offer). Therefore, whenever an optimal offer is available, it can be found and assigned to the enquiry.

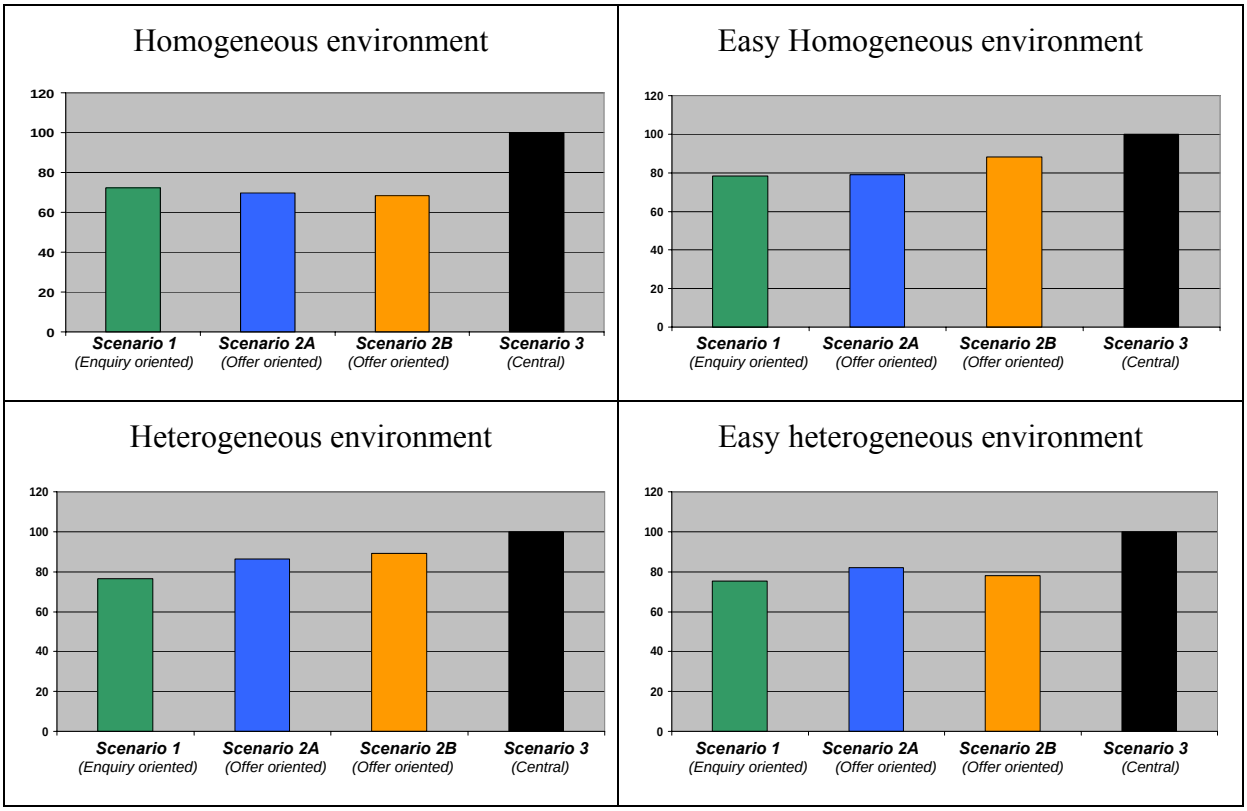


Figure 5-22 The quality of solution achieved by the market scenarios in different application environments.

The distributed scenarios show in general similar results, but there is almost always a correlation between the number of performed matching attempts and the achieved quality of

solution. This applies especially for the easy environments (both the homogeneous and the heterogeneous), where decision making takes place really between multiple alternatives (The homogeneous environment and the heterogeneous environment don't give enough margin for the decision making process).

5.5.5 Conclusion

Figure 5-23 shows the advantages and disadvantages of the market scenarios in different application environments. The time to first solution is represented as "Performance" of the market scenario, while the number of matching attempts is represented by the computing "Load".

From the figure it can be clearly shown that the evaluation results depend strongly on the application environment. Figure 5-23 proves the important fact that the market scenarios can only be evaluated in correlation with a specific application environment. The application environment is characterized by the diversity of its components (homogeneous vs. heterogeneous environments) and by the number of available offers per enquiry (the number of enquiry is constant for all studied environments, while the number of offers, i.e. of the passive units, changes).

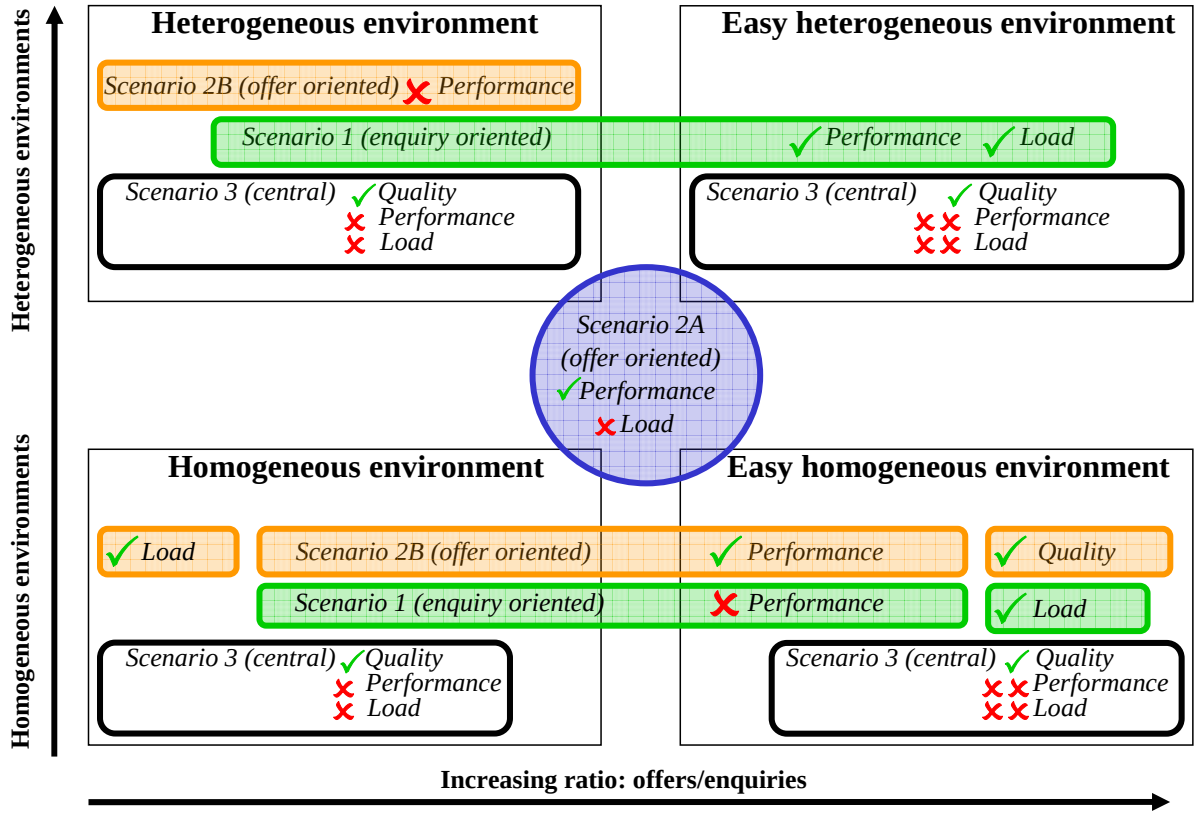


Figure 5-23 Evaluation of the market scenarios in four different environments.

The reference scenario, scenario 3, is a typical central scenario, because it promises to deliver the optimal results (quality of solution) at the price of centralized heavy-weight computation. In addition to scalability problems with higher numbers of autonomous units, the centralized solution entails the danger of total system breakdown because of a failure in the central computing unit.

In the heterogeneous environments it is recommended to use an enquiry-oriented scenario, like scenario 1, because of its short time to first solution. The enquiry-oriented scenarios have also an important advantage regarding the distribution of the computing load, so that they are

strongly recommended for a balanced computing load and high parallelism, especially for environments with high number of passive units. The small number of required matching attempts, which are performed on high number of passive units, means that the passive units can be designed for restricted computing capabilities.

Moreover, real application environments are expected to be heterogeneous environments and which involve more passive units than active units. In such environments the enquiry-oriented scenarios deliver better performance for a smaller number of matching attempts in comparison to the offer-oriented scenarios.

However, the offer-oriented scenarios promise high performance in homogeneous environments (especially with high numbers of active units, like the easy homogeneous environment).

The offer-oriented scenario 2A has an important advantage of delivering mid-level, but relatively stable, performance value in all environments, although it doesn't promise a small computing load on the active units.

Pushing towards a "one step matching" is not always a good idea. Scenario 2B faces real difficulties by looking for rare components. This is the case in the heterogeneous environment, where scenario 2B shows bad performance and high computing load. However, higher numbers of matching attempts promise better quality of solution. In this context, scenario 2B can be considered as an optimization scenario (the optimization affects only the subtypes) where enough time is available.

A very important result of the simulation is the fact that the evaluation of the market scenarios according to the defined metrics is not related to the execution time of the matching steps (as soon as all autonomous units have the same computing power). This means that the application of the market scenarios on powerful computers may reduce the overall execution time of the scenario, but it would not affect the number of required cycles to reach the first solution. The market scenarios are actually behavior patterns and their performance is determined by the reservation mechanism and by the conflicts between the autonomous units.

5.5.6 Summary

The adoption of a marketplace-oriented behavior in Organic Computing architectures, like SeMCDM, is not a trivial process. Different market scenarios have been defined in section 4.5.2 by allocating activities on available autonomous units. Reasonable market scenarios have been classified according to the kind of exchanged messages (offers or enquiries).

To prove the adequacy of these scenarios in different application environments, a prototype of market scenarios has been developed. Three evaluation metrics of the market scenarios have been defined: Success rate and time to first solution, number of matching attempts and quality of solution. A centralized scenario serves as a comparison reference for the defined scenarios. The application environment has been specified by different characterization factors like: The number of available units, the diversity of unit types and additional timing restrictions (computational load). Simulation results proved the effect of the application environment on the performance of market scenarios. Therefore, there is no optimal scenario for all typed of application environments. However, the discussion of the simulation results led to recommendations on the most suitable market scenario in different application environments.

The delay time of messages (offers and enquiries) is only considered indirectly in the form of thread sleep time. To study the effect of the communication platform in future work, it is possible to simulate it as a separate variable.

Additionally, autonomous units can dynamically change the market scenario in run time. Synchronization mechanisms and the management of such processes can be also a theme for further studies, for example, by the usage of an Observer/Controller architecture.

6. Conclusion and future work

The Organic Computing promise high adaptivity in complex engineering systems. Different approaches trying to incorporate (forms of) the Organic Computing principles in real engineering applications have been surveyed. An assessment has shown shortcomings, open questions and challenges facing the current approaches.

By studying these challenges, and through the abstraction of the open questions, similarities to challenges known in other research areas have been recognized. Corresponding solutions have been adopted to reach an interdisciplinary concept gathering Semantic Web technologies, especially the ontological expression and processing of knowledge, methods of Multi-Criteria Decision Making (MCDM), the marketplace-oriented behavior and the autonomy of self-organizing systems.

The concept is based on the idea that the autonomy given to engineering systems and to their components must be supported by balancing common, machine processable knowledge and by decision making mechanisms.

A comprehensive study of different technologies, mechanisms, and methods provided the underground for the novel concept of this thesis: Semantic Multi-Criteria Decision Making (SeMCDM).

The design of the goal architecture of SeMCDM has been achieved in a systematic way. On the base of a detailed requirements analysis, suitable Multi-Criteria Decision Making (MCDM) methods has been adopted.

The integration of MCDM methods and ontological knowledge processing techniques has been addressed in both phases: The design time and the runtime.

An ontology design tool has been extended to express the knowledge required for MCDM, and a specific MCDM-ontology has been defined. The ontology-based inference has been supported by rules to enable “Semantic Multi-Criteria Decision Making” in runtime.

The concept of SeMCDM has been studied in relation to marketplace-oriented behavior of autonomous system components. In contrast to other approaches, the marketplace-oriented behavior has been here carefully specified and a set of possible market scenarios has been discovered.

Applications of SeMCDM have been addressed in the automotive systems. The modern communication networks and design methodologies promise to be able to benefit from SeMCDM. However, the high priority of safety in automotives may delay the wide adoption of autonomy-based concepts, like SeMCDM, as a comprehensive design and operation methodology. Therefore, multimedia systems in automotives have been selected as first-step applications.

The example application proofed the functional validity of the new design tools and of the SeMCDM model. The SeMCDM model helped also to estimate the performance of SeMCDM. The measured performance is strongly related to many factors, like the size of ontology, the number of rules, and the complexity of the application.

To this reason, the performance of the market scenarios has been studies in different application environments. Three evaluation metrics of the market scenarios have been defined and measured on the simulated marketplace. The application environment itself has been specified by different characterization factors related to its complexity, in terms of size (scalability) and diversity. The results show that the market scenarios are merely behavior patterns, with conflicts-determined performance. The results show also that the performance of the market scenario depends not only on the size of the marketplace (the scalability

question) but also to the diversity of the system components. No market place can promise an optimal performance in all possible environments. A dynamic switching between the market scenarios may be an optimization solution in future work.

Such switching capability and the need for more trust in Organic Computing systems emphasises the need for a higher layer to observe the overall system behavior and to give an indicator about its "success". An Observer/Controller architecture may provide more trust and optimization-capable systems on the base of SeMCDM and the marketplace-oriented behavior.

Similar to the situation in the Semantic Web and in Multi-Agent systems, the distribution of ontologies on multiple components means that the problem of ontology mapping and inference in distributed ontologies is also an important field for future research. Within the scope of SeMCDM, efforts have been made to support inference in distributed ontologies [Abe06]. On the base of ant-colony-optimization, an inference engine in distributed ontologies has been designed and successfully implemented. However, a lack of standardized test environment for performance estimation of such inference engine can be registered.

The development of ontologies may appear as a difficult process in real applications, especially in areas of long history and high complexity, like the automotive industry. On the other side, the need for standardization pushes towards intensive involvement of modelling languages. The ontologies with their open nature, their human-machine readability and their applicability in the design time, as well as in the real time, are surely a better alternative. The modern automotive systems tend to provide a standardized "catalogue" of functionalities. Such catalog is a suitable entry application of ontological-knowledge. However, SeMCDM expect more information: SeMCDM expects a detailed expression of designer preferences. It is known that such process can be a difficult and error-prone one. Especially the question about the completeness of the given preferences cannot be easily answered. Therefore, the application of SeMCDM requires an iterative, simulation supported, development of the knowledge and of designer preferences, before loading them to the real autonomous system components.

The question about "How much knowledge and preferences are needed" is an important research field in future. While the suggested fact that "Suitable knowledge leads to suitable order" has been validated on the base of a useful methodology, it calls the definition of the word "suitable" into question. Mathematical measures for amount of knowledge (like the entropy), knowledge consistency, knowledge completeness, and for the suitability of the resulting order may provide a solution in future.

7. Bibliography

[A+07] Fabian Abel, Eelco Herder, Philipp Kaerger, Daniel Olmedilla and Wolf Siberski. Exploiting Preference Queries for Searching Learning Resources. In proceedings of 2nd European Conference on Technology Enhanced Learning (EC-TEL 2007)

[Abe06] F. Abel, "An Agent-based approach to Distributed Reasoning (in german)", Master Thesis applied to the Institute of computer and System Architecture, Leibniz University of Hannover, 2006.

[AE07] R. J. Anthony and Cecilia Ekelin, "Policy-driven self-management for an automotive middleware", at "First International Workshop on Policy-Based Autonomic Computing (PBAC 2007)" at the "Fourth IEEE International Conference on Autonomic Computing", in Jacksonville, Florida, USA, June 11-15, 2007.

[AH08] Grigoris Antoniou and Frank von Harmelen, "A Semantic Web Primer", Massachusetts Institute of Technology, second edition, 2008.

[ANT07] Richard J. Anthony, Agile Policy-Expression-Language, Policy Autonomics, http://www.policyautonomics.net , 2007.

[ARCJBE07] R. Anthony, A. Rettberg, D. Chen, I. Jahnich, G. de Boer, and C. Ekelin, „Towards a dynamically reconfigurable automotive control system architecture", in IFIP International Federation for Information Processing, volume 231/2007, Springer Boston, 2007.

[Aut03] AUTOSAR www.autosar.org, May 2003.

[Aut08] AUTOSAR GbR, "AUTOSAR Methodology", www.autosar.org, February 2008.

[BL05] Tim Berners-Lee, "Web for real people", (http://www.w3.org/2005/Talks/0511-keynote-tbl/), W3C 2005.

[BLHL01] Tim Berners-Lee, James Hendler and Ora Lassila, "The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities", Scientific American, May 2001.

[BW89] Beni, G., Wang, J. Swarm Intelligence in Cellular Robotic Systems, Proceed. NATO Advanced Workshop on Robots and Biological Systems, Tuscany, Italy, June 26–30, 1989.

[CAN08] CAN in Automoation, Controller Area Network (CAN), http://www.can-cia.org/, 2008.

[CHH92] Shu-Jen Chen, Ching-Lai Hwang, in collaboration with Frank P. Hwang, Fuzzy Multiple Attribute Decision Making, Methods and Applications, Springer-Verlag Berlin Heidelberg 1992.

[DCS99] ATLAS Detector Control Systems, "CANopen Service Data Objects", http://atlas.web.cern.ch/Atlas/GROUPS/DAQTRIG/DCS/LMB/PROFILE/cano-sdo.htm

[DS04] M. Dean and G. Schreiber. „OWL – Web Ontology Langauage Reference", W3C Recommendation, 2004. http://www.w3.org/TR/owl-ref/

[Dys07] DySCAS project web site http://www.dyscas.org/doc/DySCAS_D1.1A.pdf, "Dynamically Self-Configuring Automotive systems: Existing Technologies", specific targeted research project, 2007.

[Ehr05] Matthias Ehrgott, Multicriteria Optimization, Springer. Berlin-Heidelberg, second edition, 2005.

[Ema08] EMail from FlexRay consortium

[ES04] H. A. Eiselt, C.-L. Sandblom, “Decision Analysis, Location Models, and Scheduling Problems”, Springer-Verlag, Berlin, Heidelberg, 2004, with contributions by: J. Blazewicz, R. L. Church, A. Drexl, G. Finke, C. S. ReVelle.

[F+06] H. Fennel et. al., “Achievements and exploitation of the AUTOSAR development partnership”, Convergence, Detroit, USA, 2006

[FGE05] José Figueira, Salvatore Greco, Matthias Ehrgott, editors, Multiple Criteria Decision Analysis: State of the Art Surveys, Springer Science+Business Media Inc., 2005

[FIPA02] Foundation for Intelligent Physical Agents, “FIPA Contract Net Interaction Protocol Specification”, 2002.
[Fle05] FlexRay consortium, “FlexRay Communications System Protocol Specification”, Version 2.1, Revision A, 2005.
[Fle06] FlexRay consortium, http://www.flexray.com/, 2006.
[GC] Grid Computing Info Cetner, http://www.gridcomputing.com/
[Gel85] David Gelernter, “Generative communication in Linda”, ACM Trans. Programming Languages and Systems, 7(1):80{112, January 1985.

[Gru93a] Thomas R. Gruber, “A translation Approach to Portable Ontology Specifications”. Knowledge Acquisition, 5(2): 199-220, 1993.

[Gru93b] Thomas R. Gruber: Toward Principles for the Design of Ontologies Used for Knowledge Sharing. International Journal Human-Computer Studies 43, p.907-928, 1993.

[GWH89] Bruce L. Golden, Edward A. Wasil, Patrick T. Harker (Eds.), The Analytic Hierarchy Process, Applications and Studies, Springer-Verlag Berlin-Heidelberg 1989.

[H+02] Peter E. H. Hofmann et. al., DAIMLERCHRYSLER, „Evolutionäre E/E-Architektur, Vision einer neuartigen Elektronik-Architektur für Fahrzeuge“, 2002.

[HK04] Henze, N., Kriesell, M., “Personalization functionalitiy for the Semantic Web: Architectural outline and first sample implementations”, Semantic Web challenge 2005. In: De Bra, P., Nejdl, W. (eds.) AH 2004. LNCS, vol. 3137, Springer, Heidelberg 2004.

[HL05] Peter Hoffmann, Stefan Leboch (Daimler Chrysler AG), „Evolutionäre Elektronikarchitektur für Kraftfahrzeuge“. Information Technology 47 (2005) 4, Oldenbourg Verlag, 2005.

[HMS00] Stefan A. Hajkowicz, Geoff T. McDonald, Phil N. Smith, An Evaluation of Multiple Objective Decision Support Weighting Techniques on Natural Resource Management, Journal of Environmental Planning and Management, 43:4, 505 – 518, online publication date 01 July 2000.

[HP08] Hewlett-Packard Development Company, L.P., “Jena Semantic Web Toolkit” http://www.hpl.hp.com/semweb/tools.htm#jena, 2008.

[HY81] Ching-Lai Hwang, Kwangsun Yoon, “Multiple Attribute Decision Making, Methods and Applications: A State-of-the-Art Survey”, Springer-Verlag Berlin Heidelberg New York, 1981.

[ICP04] ICPDAS, “CAN: the new protocol to enhance your power”, http://www.icpdas.com/products/Remote_IO/can_bus/can_intro.htm, 2004.

[JHYPT05] J. L Hellerstein, Yixin Diao, Sujay Parekh and Dawn M. Tilbury, “Control Engineering for Computing Systems”, Industry experience and research challenges”, IEEE Control Systems Magazine, Volume 25, Issue 6, Dec. 2005.

[JP02] Juhasz, Z.; Paul, P., "Scalability Analysis of the Contract Net Protocol," Cluster Computing and the Grid, 2002. 2nd IEEE/ACM International Symposium on , vol., no., pp. 346-346, 21-24 May 2002.

[KC03] J. O. Kephart and D. M. Chess, "The vision of Autonomic Computing", Computer, IEEE, Volume 36, Issue 1, January 2003, pp. 41-50.

[KS06] Christine Kunzmann, Andreas Schmidt, "Ontology-based Competence Management for Healthcare Training Planning: A Case Study", in Proceedings of I-KNOW 2006, Graz, September 2006.

[Kue01] Ralf Kuesters, "Non-Standard Inferences in Description Logics", Lecture notes in computer scienc; vol. 2100: Lecture notes in artificial intelligence, Springer-Verlag Berlin Heidelberg, 2001.

[LD] Logical Decisions, http://www.logicaldecisions.com/

[LRLSM06] Jinwei Lu, Clive Roberts, Karl Lang, Alan Stirling and Keith Madelin, "The Application of Semantic Web Technologies for Railway Decision Support". In Jatinder N.D. Gupta, Guisseppi A. Forgionne, Manuel Mora T. editors, "Intelligent Decision-making Support Systems, Foundations, Applications and Challenges", pages 321-337, Springer Verlag, London, 2006.

[LSL] The Linnean Society of London, http://www.linnean.org/.

[LY99] Tim Lindholm, Frank Yellin, "Java(TM) Virtual Machine Specification", The 2nd Edition, The Java Series, httpS//java.sun.com/docs/books/jvms/second_edition/html/VMSpecTOC.doc.html, Sun Microsystems, 1999.

[Mal06] Andreia Malucelli, „Ontology-based Services for Agents Interoperability", doctoral thesis presented at the university of Porto, Portugal, 2006.

[Mar99] Jean-Marc Martal, Multicriterion Decision Aid: Methods and Applications, CORS-SCRO annual conference, Windsor, Ontario 1999.

[May06] Eugen Mayer, "Serial Bus Systems in the Automobile, Part 2: Reliable data exchange in the automobile with CAN", http://www.vector-worldwide.com/portal/medien/cmc/press/PTR/SerialBusSystems_Part2_ElektronikAutomotive_200612_PressArticle_EN.pdf, 2006.

[May07a] Eugen Mayer, "Serial Bus Systems in the Automobile, Part 4: FlexRay for data exchange in highly critical safety applications", http://www.vector-worldwide.com/portal/medien/cmc/press/PTR/SerialBusSystems_Part4_ElektronikAutomotive_200703_PressArticle_EN.pdf, 2007.

[May07b] Eugen Mayer, „Serielle Bussysteme im Auto, Teil 5: MOST für die Übertragung von Multimediadaten", Elektronik automotive 9/2007, http://www.vector-worldwide.com/portal/medien/cmc/press/PTR/SerielleBussysteme_Teil5_ElektronikAutomotive_200712_PressArticle_DE.pdf, 2007.

[MC06] MOST Cooperation, MOST Function Block NetworkMaster, Rev. 2.5.0, 12/2006, 2006.

[Mie06] Kaisa Miettinen, International Society on Multiple Criteria Decision Making, http://project.hkkk.fi/MCDM/intro.html, September 2006

[MMS06] M. Mnif, C. Müller-Schloer, "Quantitative Emergence", In Proceedings of the 2006 IEEE Mountain Workshop on Adaptive and Learning Systems (IEEE SMCals 2006), July 2006.

[Mos08] MOST Cooperation, “Motivation for MOST”, http://www.mostcooperation.com/technology/introduction/index.html, 2008.

[MS04] Christian Müller-Schloer, “Organic Computing - On the feasibility of controlled emergence”, International Conference on Hardware/Software Codesign and System Synthesis 2004. CODES + ISSS 2004, 8-10 Sept. 2004, pp. 2-5.

[Mur03] Niall Murphy, “A short trip on the CAN bus”, http://www.embedded.com/showArticle.jhtml?articleID=13000304, 2003.

[Muy08] Henry Muyshondt, „Consumer and automotive electronics converge: Part 2 – A MOST implementation“, Automotive Design Line, online edition http://www.automotivedesignline.com/howto/infotainment/198001031, 2008.

[OCI] http://www.organic-computing.de/

[Ols96] David L. Olson, Decision Aids for Selection Problems, Sprigner-Verlag New York, 1996

[OMG08] Object Management Group, “Unified Modelling Language: UML Resource Page”, http://www.uml.org/, 2008.

[OV04a] OSEK/VDX Portal, “System configuration, OIL: OSEK Implementation Language”, version 2.5, 1 July 2004.

[OV04b] OSEK/VDX Portal, “Communication”, version 3.0.3, 20 July 2004.

[OV04c] OSEK/VDX Portal, “Concept and Application Programming Interface”, version 2.5.3, 26 July 2004.

[OV05] OSEK/VDX Portal, “OSEK Run Time Interface (ORTI), Part A: Language Specification”, version 2.2, 14 November 2005.

[OVP03] OSEK VDX Portal, http://www.osek-vdx.org/, 2003.

[PCJ04] Paurobally, S., Cunningham, J. and Jennings, N. R. (2004) Verifying the Contract Net Protocol: a case study in interaction protocol and agent communication semantics. In: 2nd International Workshop on Logic and Communication in Multi-Agent Systems, 2004, Nancy, France.

[Pro08] Protege, http://protege.stanford.edu/, 2008

[PYM07] Zhou Pu-Cheng, Han Yu-Sheng and Xue Mo-Gen, “Extended Contract Net Protocol for Multi-Robot Dynamic Task Allocation”, Information Technology Journal 6(5): 733-738, 2007. ISSN 1812-5638. The Asina Network for Scientific Information 2007.

[RMBMSS06] U. Richter, M. Mnif, J. Branke, C. Müller-Schloer, and H. Schmeck. "Towards a generic observer/controller architecture for Organic Computing". In C. Hochberger and R. Liskowsky, editors, INFORMATIK 2006 -- Informatik für Menschen, volume P-93 of GI-Edition -- Lecture Notes in Informatics, pages 112--119, Bonn, Germany, Sept. 2006. Köllen Verlag.

[Ros09] Ross, Don, "Game Theory", The Stanford Encyclopedia of Philosophy (Spring 2009 Edition), Edward N. Zalta (ed.), URL = <http://plato.stanford.edu/archives/spr2009/entries/game-theory/>.

[Ros96] Davide Rossi, “Jada: multiple object spaces for Java”, http://www.cs.unibo.it/~rossi/jada/, 1996.

[Roy05] Bernard Roy, Paradigms and Challenges, in José Figueira, Salvatore Greco, Matthias Ehrgott, editors, Multiple Criteria Decision Analysis: State of the Art Surveys, Springer Science+Business Media Inc., 2005

[Sch05] Hartmut Schmeck, “Organic Computing- A New Vision for Distributed Embedded Systems”, in Proceedings Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2005), 18-20 May 2005, Seattle, WA, USA, pp. 201-203. IEEE, IEEE Computer Society 2005, May 2005.

[SK06] Andreas Schmidt, Christine Kunzmann, “Towards a Human Resource Development Ontology for Combining Competence Management and Technology-Enhanced Workplace Learning”. In proceedings of OntoContent 2006 (in conjunction with OTM Federated Conferences 2006), Springer, Lecture Notes in Computer Science, 2006.

[Smi80] Reid G. Smith, “The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver”, IEEE Transactions on Computers, Vol. C-29, No. 12, December 1980.

[SMS08] H. Schmeck, C. Müller-Schloer, "A Characterization of Key Properties of Environment-mediated Multiagent Systems", in Lecture Notes in Computer Science, Springer Verlag, Berlin / Heidelberg, 2008.

[Ste05] Theodor J Stewart, Dealing with Uncertainties in MCDA, in José Figueira, Salvatore Greco, Matthias Ehrgott, editors, Multiple Criteria Decision Analysis: State of the Art Surveys, Springer Science+Business Media Inc., 2005

[SZ04] Stephan Zelewski, „Kooperatives Wissensmanagement in Engineering-Netzwerken“, (Vorläufiger) Abschlussbericht zum Verbundprojekt KOWIEN, Arbeitsbericht Nr. 25, 2004.

[Tri00] Evangelos Triantaphyllou, Multi-Criteria Decision Making Methods: A Comparative Study, Kluwer Academic Publishers, 2000.

[Tur95] Turban E., Decision Support and Expert Systems: Management Support Systems, 4th edition, Prentice-Hall International, London, 1995.

[Uni07] The Unicode Consortium, “The Unicode Standard, Version 5.1.0, defined by: The Unicode Standard, Version 5.0 “, Boston, MA, Addison-Wesley, 2007. ISBN 0-321-48091-0), as amended by Unicode 5.1.0 (http://www.unicode.org/versions/Unicode5.1.0/).

[Van95] Vansnick, J.C., L'aide multicritère à la décision: une activité profondément ancrée dans son temps”, Newsletter of the European Working Group “Multi Decisions Aiding”, Series 2,6, Spring, 1-2, 1995

[W3C04] World Wide Web, “Resource Description Framework (RDF)”, (http://www.w3.org/RDF/), W3C 2004.

[W3C06] World Wide Web Consortium, “Extensible Markup Language (XML) 1.1 (Second Edition)”, W3C Recommendation, http://www.w3.org/TR/2006/REC-xml11-20060816/, 16 August 2006.

[Wik08a] Wikipedia, “Local Interconnect Network”, http://en.wikipedia.org/wiki/Local_Interconnect_Network , 2008.

[Wik08b] Wikipedia, http://en.wikipedia.org/wiki/Controller_area_network, 2008.

[Wik08c] Wikipedia, http://en.wiktionary.org/wiki/ophelimity, 2008.

[XW01] Lai Xu, Hans Weigand, “The Evolution of the Contract Net Protocol”, In Proceedings of WAIM 2001, LNCS 2118, pages 257–264, 2001.

[YA03] Yilmaz Alan, “Konstruktion der KOWIEN-Ontologie”, KOWIEN-Projektbericht 2/2003, 2003.

[ZAADW05] Zelewski, S.; Alan, Y.; Alparslan, A.; Dittman, L.; Weichelt, T. (Hrsg.): „Ontologiebasierte Kompetenzmanagementsystems, Grundlagen, Konzepte, Anwendungen“, Logos Verlag, Berlin, 2005.

[Zel01] Holger Zeltwanger, „CANopen“, VDE Verlag GmbH, 2001.

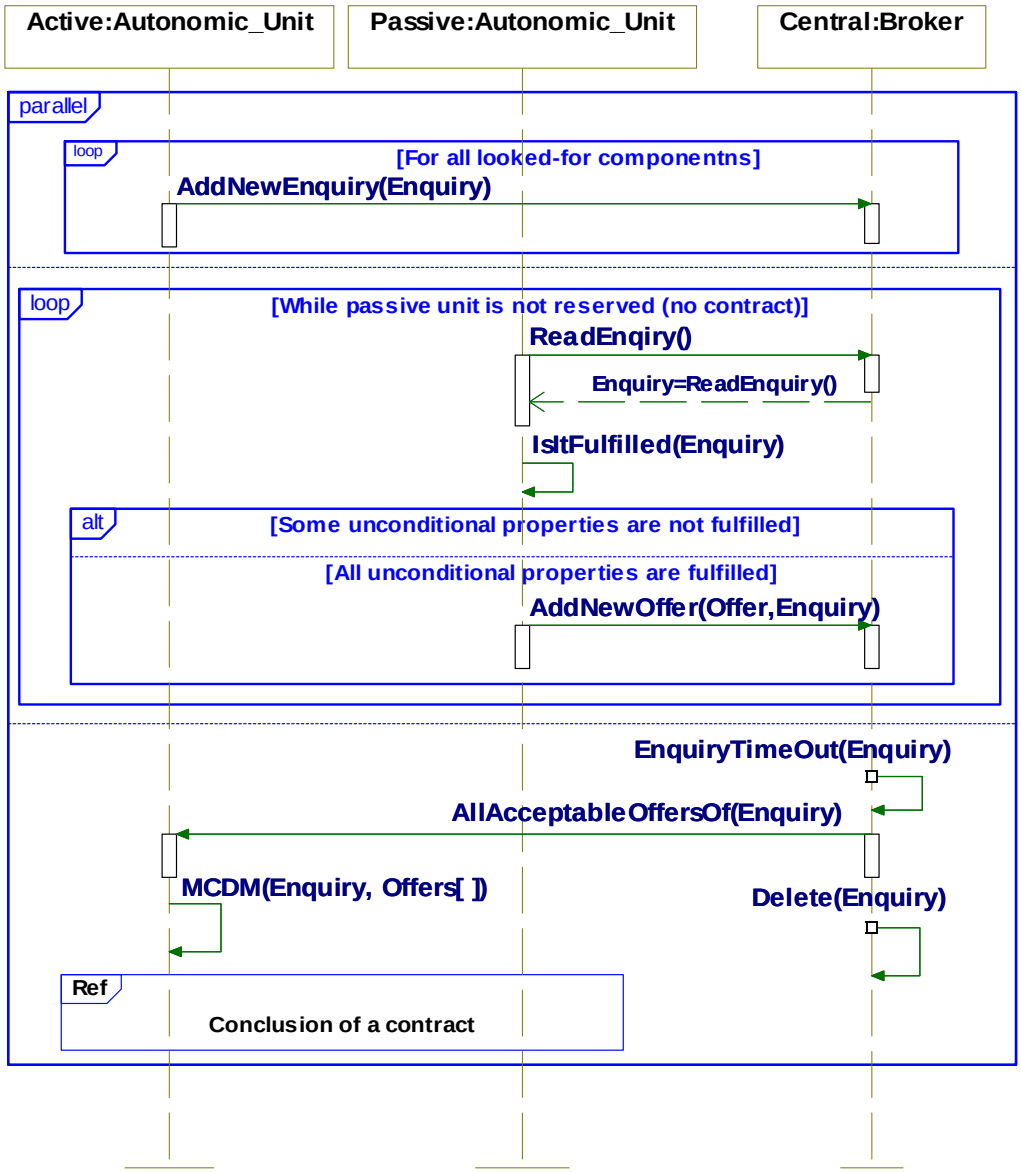
Index

Wissenschaftlicher Werdegang

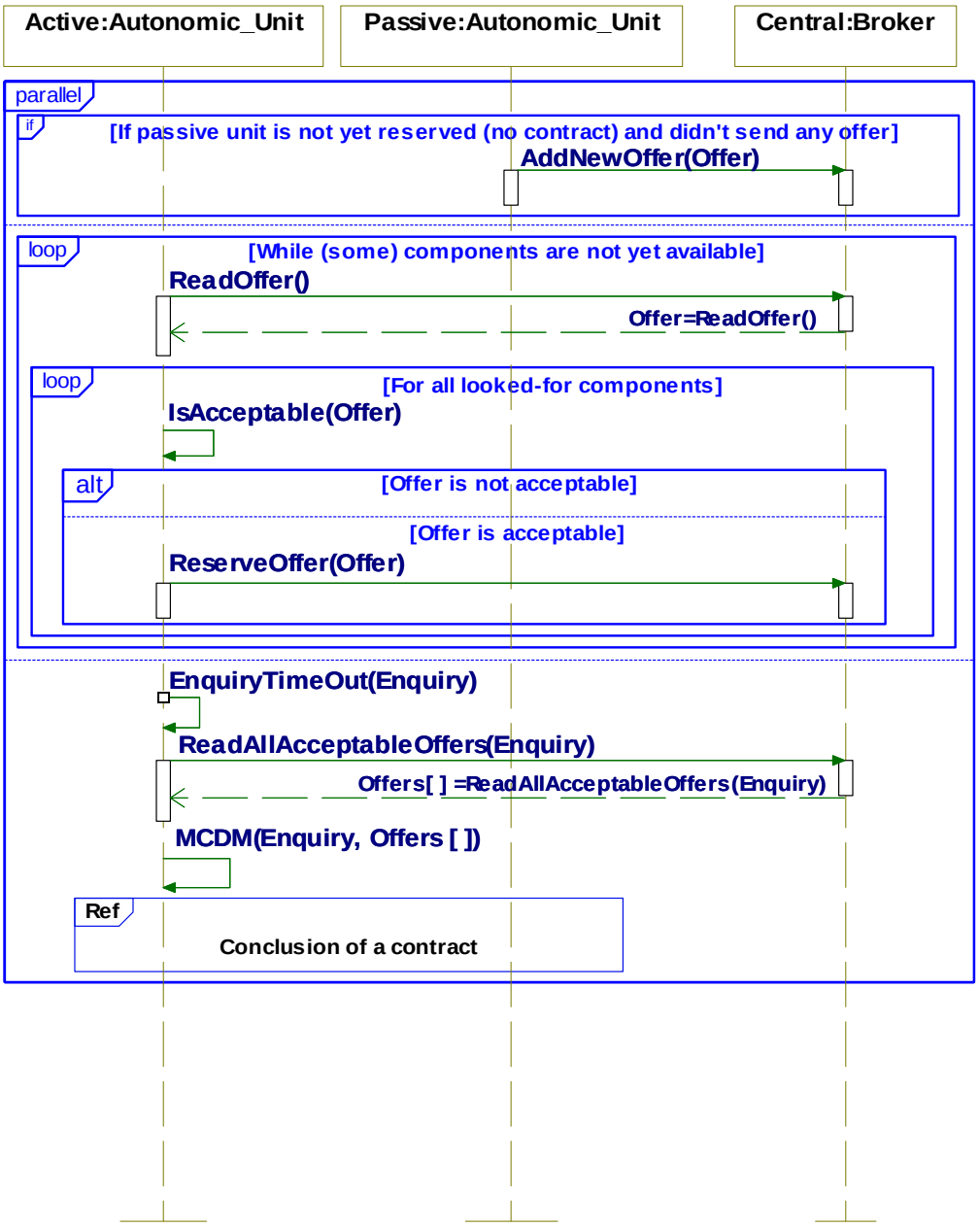
Name	Ghadi Mahmoudi
Geboren am	10.10.1975 in Latakia (Syrien)
Staatsangehörigkeit	Deutsch/Syrisch

1980 – 1986		Grundschule in Latakia
1986 – 1989		Mittelschule in Latakia
1989 – 1992		Oberschule in Latakia, mit Abitur
1992 – 1997	Bachelor	Universität Tischrin in Lattakia/Syrien Fakultät für Elektrotechnik Abteilung Elektronik Hochschulabschluss: „Sehr gut" Gesamtnote: 78 %
1997 – 1998	Diplom	Universität Tischrin in Lattakia/Syrien Fakultät für Elektrotechnik Abteilung: Rechneringenieurwesen und Automatisierung Prädikat: „Sehr gut" Gesamtnote: 83,5 %
1998 - 2000		Assistent an der Fakultät für Elektrotechnik der Universität Tischrin Abteilung für Elektronik (Rechnerstrukturen, Programmierung, Nachrichtentechnik, Automatisierung)
11.01.00 - 27.09.00	Sprachkurs	Deutschkurs im Auslandsinstitut Dortmund PNdS-Prüfungsabschluss (DSH)
15.11.00 - 14.01.03	Master of Science	Universität Hannover Fachbereich: Elektrotechnik und Informationstechnik Studienrichtung: Technische Informatik Prädikat: „Gut" Gesamtnote: 1,71
Ab 15.01.03	Promotion	Universität Hannover Institute für System- und Rechnerarchitektur Forschungsbereich: Organic Computing

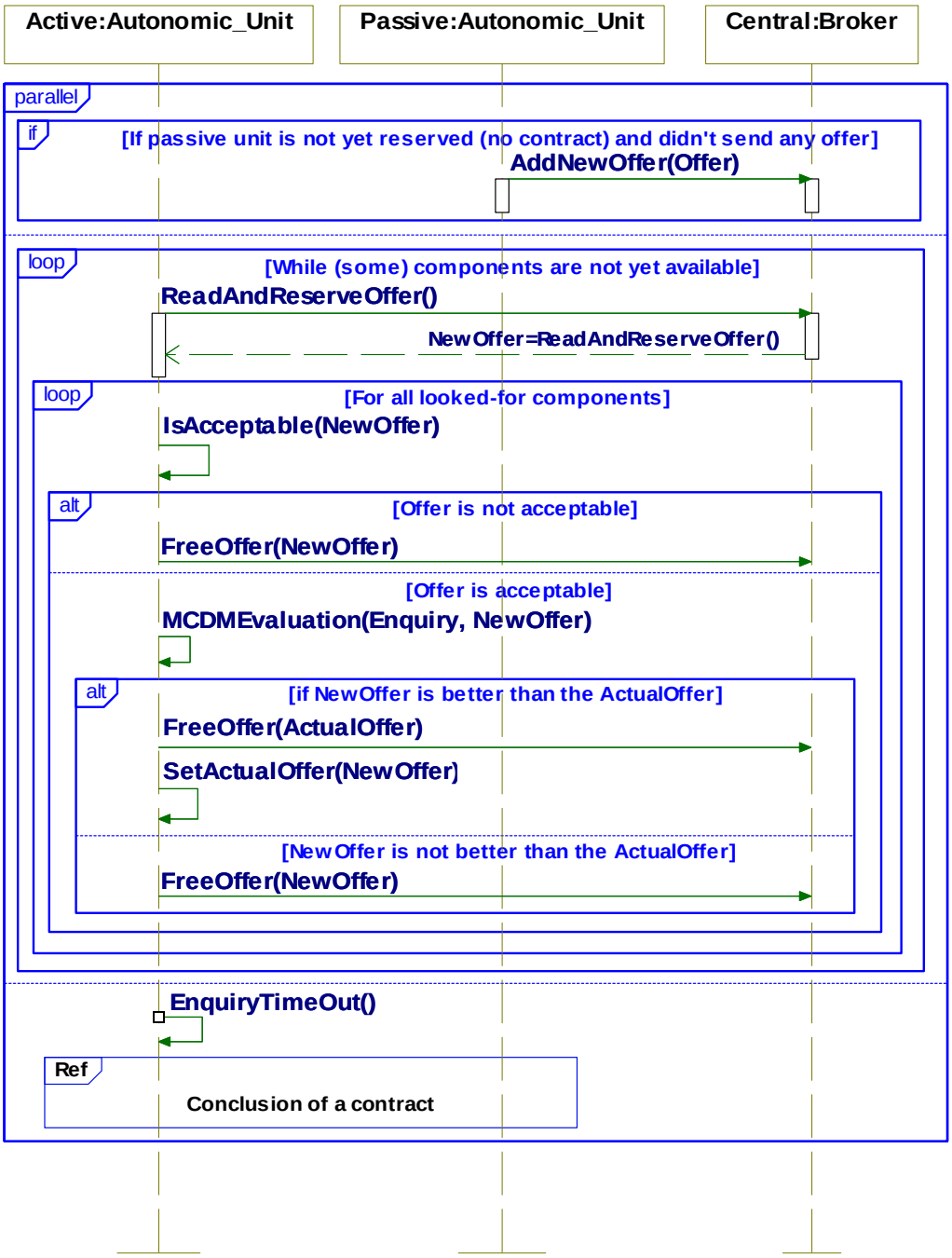
Appendix A



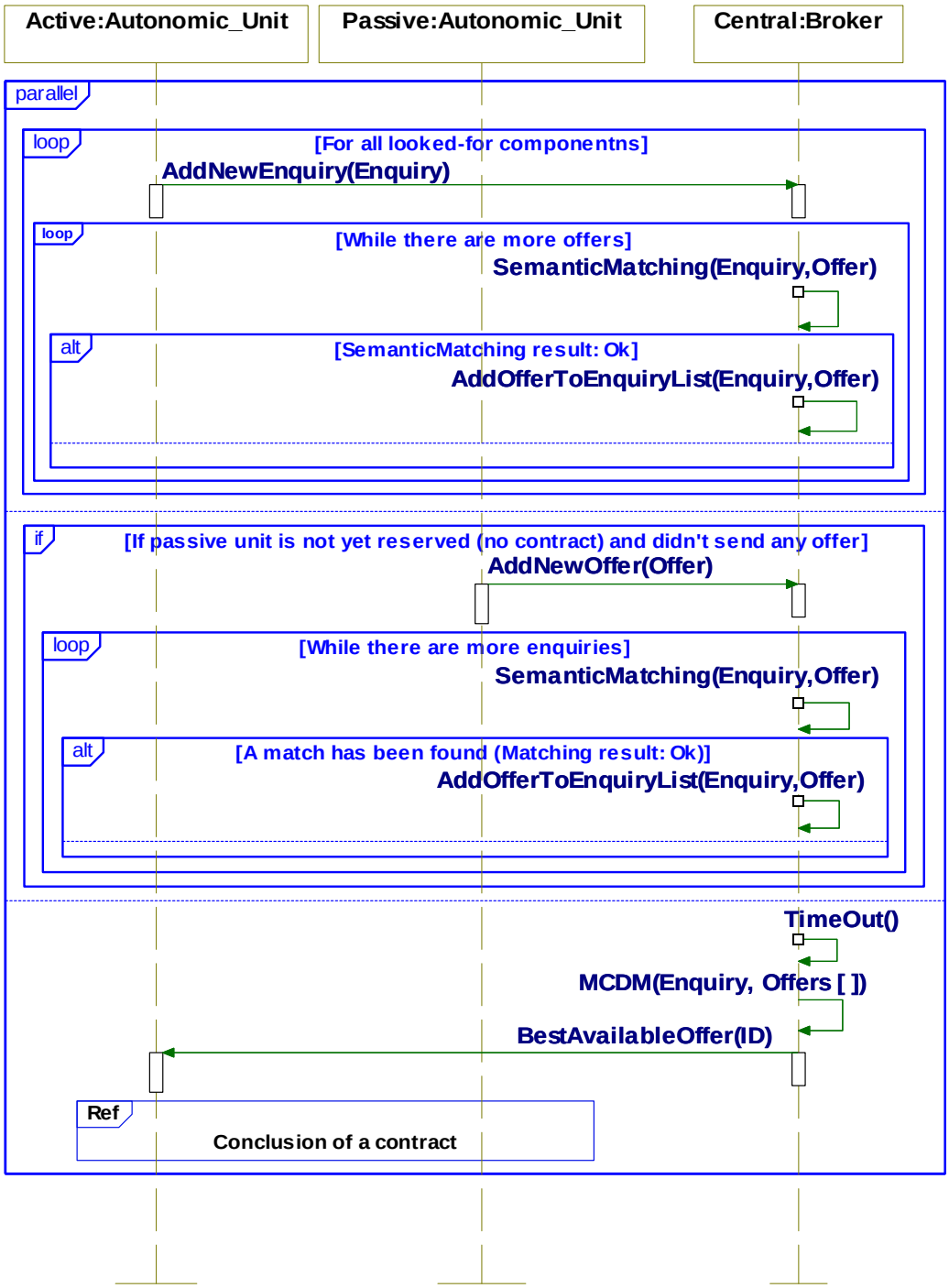
Sequence Diagram 1: Scenario 1. The method "IsItFulfilled" represents the first matching step. The second matching step is carried out in the „MCDM“ method.



Sequence Diagram 2: Version "A" of scenario 2



Sequence Diagram 3: Version "B" of scenario 2



Sequence Diagram 4 The central market scenario.