

DISSERTATION

Situation-dependent Behavior in Building Automation

ausgeführt zur Erlangung des akademischen Grades eines
Doktors der technischen Wissenschaften unter der Leitung von

O. Univ. Prof. Dipl.-Ing. Dr. techn. Dietmar Dietrich
Institut für Computertechnik

und

Ao. Univ. Prof. Dr. Ernst Schuster
Institut für Medizinische Computerwissenschaften

eingereicht an der Technischen Universität Wien
Fakultät für Elektrotechnik

von

Dipl.-Ing. Gerhard Russ
Matr.-Nr. 9525715
Liebhartsgasse 7/21
1160 Wien

Wien, am 1. Juli 2003

.....

Situation-dependent behavior in building automation

*Ein guter Spruch ist die Wahrheit
eines ganzen Buches in einem einzigen Satz.*

Theodor Fontane

Kurzfassung

Die Verbesserung verschiedenster Lebensbereiche ist ein grundlegendes Ziel in der Automation. Auf unterschiedlichsten Gebieten wird nach Möglichkeiten gesucht, Abläufe schneller, sicherer und effizienter zu gestalten. Das Streben nach diesen Faktoren stellt eine treibende Kraft für Entwicklungen in der Automation dar. Trotz der vielen Fortschritte auf diesem Gebiet ist die heutige Technik nicht immer in der Lage, mit den rasant steigenden Anforderungen und Bedürfnissen unserer Gesellschaft Schritt zu halten.

Der Fokus dieser Arbeit liegt in der Automation in Gebäuden. In diesem Bereich existiert bereits eine Vielzahl an fertigen Systemen, die für verschiedene Aufgabengebiete eingesetzt werden. Allerdings sind die meisten dieser Anwendungen in sich abgeschlossen und verhindern somit die Erstellung übergreifender Funktionen. Zudem besteht die heutige Gebäudeautomation aus rein reaktiven Systemen, das heißt, diese Systeme reagieren auf Inputs lediglich, wenn die Relevanz und der Einfluss dieser Daten explizit vordefiniert wurden. Andere Faktoren, die einer Anwendung nicht direkt zugeordnet sind, werden einfach ignoriert, d. h., unvorhergesehene Situation können zu Fehlverhalten oder unbeabsichtigte Aktionen führen.

Das Ziel dieser Arbeit ist die Entwicklung eines Modells, das die Forderung nach einem zuverlässigen System, dass die komplexen Situationen des modernen Lebens bewältigen kann, erfüllt. Dieses System soll eine „bewusste“ Situationserkennung und ein „vorausschauendes“ Handeln ermöglichen. Dafür ist es notwendig, neben der Einbindung verschiedenster vorhandener Technologien sehr große Informationsmengen effizient zu verarbeiten. Biologische Systeme und Prinzipien haben sich seit jeher als zuverlässige und geeignete Vorbilder für technische Entwicklungen bewährt. Daher liegt das Hauptaugenmerk meiner Arbeit in der Integration biologischer Konzepte in das Modell.

Als Abschluss dient eine auf diesem Modell basierende Implementierung eines Systems der Überprüfung der theoretischen Annahmen.

Situation-dependent behavior in building automation

*Ein guter Spruch ist die Wahrheit
eines ganzen Buches in einem einzigen Satz.*

Theodor Fontane

Abstract

The general aim of automation is to improve different areas of life. That is to say, the driving force in this field of research is geared towards managing processes faster, safer, and more efficient. Great advance has been achieved but techniques that are used today are facing their limitations when it comes to meet the increasing demands of our rapidly developing society.

The main focus of this thesis will be on the automation in buildings. Within this area a vast amount of systems coexist which are used in various fields of applications. Unfortunately, most of these applications are self-contained and therefore obstruct the establishment of system-spanning functions. Moreover, today's building automation consists of purely reactive systems, solely reacting to previously defined inputs. It follows that factors which are not directly and explicitly assigned to the application cannot be considered. Unforeseen situations may result in malfunctions and unintended actions.

The aim of this paper is to establish a model which is able to meet our increasing demand for a reliable system that is capable of dealing with complex situations in modern life. This system also needs to be capable of integrating various techniques as well as processing a large amount of data in order to provide a basis for "consciously" recognizing situations and "far-sighted" actions. Therefore one question has to be whether - and to which extend - existing systems can be interconnected and extended. Biological systems and principles have ever since proven to be reliable and qualified sources for technical development. I therefore concentrate on establishing a model architecture that embodies nature-like elements and biological concepts. Finally, an implementation of a system based on this model serves as an evaluation of the theoretical assumptions.

Preface

"But where shall I start? The world is so fast, I shall start with the country I know best, my own. But my country is so very large, I had better start with my town. But my town, too, is large. I had best start with my street. No, my home. No, my family. Never mind, I shall start with myself."

Elie Wiesel

The origin of motivation of this thesis was based on my work in the Smart Kitchen Project. Within the framework of this project, the idea was born to build a model structure of situation-dependent behavior.

Whereas the start of this work was intended to be a pure technical treatise, the fascination with biological concepts increased steadily in the course of the time.

I would like to express my gratitude and appreciation to Prof. Dietrich for his constant support, valuable inputs and help to stay on track during the long process of writing this doctorate thesis.

I would also like to gratefully acknowledge Prof. Schuster for furnishing the second expert opinion of this thesis.

A personal enrichment were my colleagues Clara Tamarit and Markus Falkner, who were also involved in the Smart Kitchen Project. Furthermore I owe many thanks to Kerstin Kugler and Felicitas Engeler for their critical proof-reading of the manuscript.

Content

Chapter 1 provides an overview of the topics of this work. Following a brief analysis of the goal the theoretical as well as the practical background is explained. Moreover, in this chapter the status quo in building automation is presented.

Chapter 2 describes the present state of technical realizations of biological concepts in this application field. Furthermore, a variety of biological research works is analyzed.

Chapter 3 explains the development of the intended model structure. It starts with the perception of information from the environment and describes step by step the processing of this data. The result of this chapter is the theoretical structure of the entire system.

Chapter 4 uses this model structure in a first prototype. It explains the realization of each of the layers and describes possible ways of implementation of the different concepts of chapter 3.

Chapter 5 shows the results of the analysis based on the realization. Within this chapter each layer of the system is tested separately as well as the behavior of the entire system is analyzed.

Chapter 6 summarizes the results of the work. Problems which occurred in the course of the work are specified and analyzed. Finally, a prospect of further work is given.

Table of Contents

Kurzfassung

Abstract

Preface

1	INTRODUCTION.....	1
1.1	HOME AND BUILDING AUTOMATION	2
1.2	AN EXTENSION	5
1.3	ANALYSIS OF THE GOAL	8
1.4	THEORETICAL AND PRACTICAL BACKGROUND	14
1.4.1	Communication	15
1.4.2	Testing environment.....	19
1.4.3	Databases.....	24
1.4.4	Methods for searching and comparing	24
2	SITUATION RECOGNITION	31
2.1	SCENARIOS.....	31
2.2	BIOLOGICAL SYSTEMS.....	35
2.2.1	Subsumption Architecture	36
2.2.2	Cognition	38
2.2.3	Structure of biological systems	40
2.2.4	Performance strategies of biological systems.....	45
2.3	TECHNICAL SYSTEMS	47
2.3.1	Technical realizations using situation recognition.....	47
2.3.2	Biologically inspired technical realizations.....	48
2.3.3	Conclusion of existing approaches	51
3	A MODEL FOR SITUATION-DEPENDENT BEHAVIOR	57
3.1	PERCEPTION	58
3.1.1	Sensors	59
3.1.2	Transformation	69
3.1.3	Perception Layer Interface	75
3.2	SITUATION RECOGNITION.....	77
3.2.1	Requirements for a representation	78
3.2.2	Representation of the current environment.....	83
3.2.3	Representation Layer Interface.....	89
3.2.4	Recognition of the current situation	92
3.3	REACTION CHOOSING	101
3.4	REACTION IN THE REAL WORLD	108
3.5	CONCLUSION: THE ENTIRE MODEL STRUCTURE	111

4	REALIZATION.....	116
4.1	GENERAL CONSTRAINTS	116
4.2	PERCEPTION OF THE ENVIRONMENT.....	120
4.3	REPRESENTATION OF THE ENVIRONMENT	123
4.4	RECOGNITION OF THE SITUATION AND REACTION	126
5	DISCUSSION.....	131
5.1	TESTING SEQUENCE	131
5.2	PERCEPTION OF THE ENVIRONMENT.....	132
5.3	REPRESENTATION OF THE ENVIRONMENT	134
5.4	RECOGNITION OF THE SITUATION AND REACTION	136
5.5	SYSTEM FOR SITUATION-DEPENDENT BEHAVIOR	140
6	CONCLUSION.....	145
6.1	ACHIEVEMENTS.....	145
6.2	OUTLOOK	147

Appendix

A	List of Figures
B	Abbreviations
C	Bibliography

Chapter 1

Introduction

Alice came to a fork in the road. "Which road do I take?" she asked. "Where do you want to go?" responded the Cheshire cat.

Lewis Carroll, Alice in Wonderland

In many different areas, efforts have been increased to manage processes faster, more efficient, and above all without human management, in areas like industrial processing, piloting an aircraft or just the heating control in their own homes – efforts to automate these processes.

The term “automation” dates from the Greek “automatos”, which means “occurs by itself” or “moving by itself”, and is used for machines, devices or technical constructions being able to work autonomous. The goal of automation is to run technical systems with a maximum of safety, economy and reliability, and to relieve humans as good as possible of monotonous, dangerous and also of strenuous activities.

One can find a variety of different automation systems on the market, systems that are tailored to specific tasks, depending on the application field of the manufacturer. So far, these technologies were sufficient to control simple applications, for example the light or the heating. In the meantime, the manufacturers have realized that for a further extension and improvement of their systems aspects like interoperability between different devices and systems is gaining significance and that it is necessary to define new concepts. Nevertheless, today’s automation applications still suffer from the rigid construction and the static and unalterable connections of the control circuits they are based on.

The global framework of this work is the field of home and building automation and the inclination to extend existing systems far beyond their today’s potential. This extension should offer new functionalities by using existing applications. The idea

behind is to apply biological concepts and methods, and to produce recognition of situations in rooms and buildings. By that, the system will obtain more global information, resulting in a kind of understanding of the current situation in order to find an appropriate reaction to it.

1.1 Home and Building Automation

Over a long period functions in buildings have been realized only by using centralized computers or programmable logic controllers (PLC) connected with extensive cabling. With fieldbusses a new technology of networks has been developed, a technology which allows decentralizing the intelligence of large systems using small units, i.e. nodes [Die97, Die98]. These are autonomic working units, containing a processor unit, memory and several interfaces. They are able to preprocess input information and send only the results of their calculations to the network. Hence, the amount of data on the transmission medium can be dramatically reduced.

In buildings the drawbacks of centralized systems play a very important part. For example, [Sch97] and [Ste95] explain that with increasing complexity a central system will require a great deal of wiring because of parallel cabling. It offers a lesser degree of availability because if a malfunction in the central part occurs, it will affect the entire system. Additionally, it offers just a limited flexibility because it will become increasingly difficult to change or extend a complex central system. Using the new technologies it should be possible to eliminate these shortages. However, [Die00] emphasizes that the reduction of cabling has only been a historical driving force behind the development of fieldbus systems and that aspects such as the reduction of the operating costs are the decisive factors for today.

[Die00] points out that nowadays the number of nodes in a building can already reach thousands and is still increasing. Thus, the required information can be found in a high number and can furthermore be located everywhere in the building. These facts alone suggest the usage of a decentralized system. Moreover, these technologies make it possible to use different physical media types for the data transfer [Die98].

THE ADVANTAGES...

Automation systems like control systems or regulations were centrally organized in the beginning. Today, they have become decentralized systems using autonomous acting sensor and actuator nodes. If there is now a malfunction in one of the nodes, it will have no effect on the overall system, and the system offers consequently a higher availability. The integration of many redundant sensors allows the implementation of built-in-test-functions, wrap-around systems (WAS), plausibility checks, etc. Furthermore, by using these functions, the reliability and the availability of the system can be improved. Such procedures and methods were already described in [Die84] and can be seen as state of the art in areas like aircraft construction or ASIC-development [Ste00].

Furthermore, the investment costs represent an important factor. Using components which are able to process data and to communicate will naturally result in higher costs for these units. But a node can be used several times for different tasks. Additionally, the reduction of energy consumption through the use of automation systems like heating control or light control or the automatical closing of windows in

case of rain through the use of rain sensor, and the subsequent preservation of the building and saving of money must be kept in mind. [LON02, Sch97].

These new technologies initiate new approaches in the field of home and building automation and new solutions to the given problems. There has to be a shift in thinking, beginning with the planning up to the startup of an installation: components are not longer referred to but the application and the functionality become the focus of attention. These systems are developed top-down: beginning with the goal of the application, you can derive the components necessary for it. In this sense, [Sou00] describes the usage of a top-down study in case of home and building automation for the first time.

Due to that policy another problem should have been solved: the division of the entire automation system into different industries.

Industry:

Large-scale Production; organized economy activity connected with the production, manufacture or construction of a particular product or range of products [Mic99].

In this thesis I make use of the term “industry” in equivalence to the German term “Gewerke”. By that, I understand application-specific fields, for example lighting, heating, ventilation, air condition (HVAC) or shading. This usage is analog to [Kab02].

Gewerke:

Mittelhochdeutsch; Handwerks-, Zunftgenosse, Teilhaber an einem Bergwerk; schon in mittelhochdeutscher Zeit wird die Bedeutung von Gewerke auf den Bergbau eingeschränkt, bedeutet deshalb bis ins 18. Jahrhundert „Gesamtheit der Inhaber eines Bergwerks“. Von da an wird es auch auf andere Berufe ausgedehnt [Gru99].

Before the development of fieldbus technology in the field of building automation, each industry was announced, planned, constructed and put into operation separately. “Togetherness” was neither possible nor necessary: each function was considered by a different industry section [Kab02, Die01, LON02, Sch97, Sch98]. In the course of time, the companies considered process automation for their own function but without considering the other functions of a building. Since all these functions were working independently, it was not necessary to use the technologies of other industry areas. Therefore, there was no reason for cooperative interaction.

The result of that practice is the existence of many different systems working in parallel. Each industry uses its own wiring, its own components and its own communication technique.

Since the demand of automation systems increased, there are no longer any reasons for a division into these industries. Components should be used by as many applications and systems as possible. In order to achieve common efficiencies and a

coordinated behavior, the algorithms used by the different industries have to interlock, to communicate and to share information.

Here was a gap between some systems and the fieldbusses had to fill it up. They should ensure a comprehensive usability.

...AND THE DRAWBACKS

Meanwhile, all these problems were well known and several producers and developers of fieldbus technology have started to deal with them. Hence, one could assume that the technology is well prepared for higher demands and applications where the cooperation of different applications is absolutely necessary. But in spite of the progressive way of thinking in the conception of some bus systems [LON30, Wal97, Sau01] and the encouraging comprehensive examples in [Kab02], it unfortunately seems that there still exist some weaknesses.

In case of LonWorks, the basic idea of combining all the different industries and systems is the use of hierarchical structures containing object and profile definitions and a specification of the communication. All real components should be based on these definitions and thus be able to cooperate. Consequently, they become interoperable [LON32].

However, this concept does not consider all factors. Someone has to define the profiles. And who else could make a definition of a component that will be used in a specific application in a better way than someone who is working in the industry area where the application is coming from? Therefore, companies have started to define the characteristics of units with regard to their specialized knowledge.

Subsequently, also the profiles themselves are conceived for certain industries [WWW2], which is why we are still confronted with problems as described in [Rau00]. Even within one industry area or a group of applications using the same technology problems can arise like described in [Rus01]. The reasons of these problems are industry-related profiles defined by manufacturers. They often include characteristics in their components which will be useful for a specific application [Wal97]. Using the components of these manufacturers means that additional features in exactly that application can be found. It will work perfectly until you have to replace a component with one of a different manufacturer which does not support that features ... Therefore, it is a reasonable assumption that also these new technologies are aimed at specific, self-contained applications.

Moreover, the problem solution has stopped with these profile definitions. [Rau00] describes a way to extend that hierarchy of objects and profiles, but until now the improvements make only slow progress. Many researchers like [Pos01] or [Sch98] have focused on the technical side. For instance, [Pos01] has tried to find a solution by specifying integrated management middleware architecture and by identifying all services and management mechanism which are necessary for combining established home and building networking technologies.

In contrast to these pure technical attempts to deal with this predicament, this work evaluates a different possibility. By using ideas adopted from biological systems a concept has been developed which allows the combination of different applications independent of their industrial origin and the information they are working with. Moreover, by merging the information provided by the different systems a global

representation of the environment is possible, and therefore new functions emerge, which use this extra information. Thus, the result of this work points out a new way in order to overcome the described drawbacks and to extend current systems to be able to master more complex tasks.

1.2 An extension

When speaking of an extension and an improvement of automation systems in buildings following the above mentioned biological concept, then that means that the previously independent systems and applications have to work together, and that they must be combined into one global system in order to fulfill superordinate functions. This includes on the one hand the use of already available systems and applications; on the other hand it presupposes that these systems and applications already have certain flexibility and the possibility of an extension.

The combination into one system must not be misinterpreted as return to a central system. As mentioned in [Sch98], building automation should use a virtually central but physically decentralized, distributed architecture. One logical control system combines all sensors and actuators of the complete installation in a single large, optimized, and intelligent control loop. Since this makes more information available for the different applications, new, more global acting functions and possibilities are enabled. The central control system is then implemented as a distributed program on a variety of microprocessors and intelligent sensors or actuators, which are connected by a large homogeneous data network.

Besides the possible advantages by combining existing applications, the further usage of these systems provides also economical benefits. In many buildings already a variety of automation systems are used which work satisfactorily for years and with well-known factors such as costs, expenditure, functionality, necessary maintenance etc. Correspondingly, the entire exchange to a new solution would involve a variety of elements of uncertainty. Therefore, a solution can only consist of the extension of these systems instead of the replacement by a completely new system. Hence, these systems have to satisfy certain requirements which will be discussed in detail later.

Starting from the totality of needs which can be met by automation systems we can divide these needs into four basic functionalities, as it has already been done in works like [Die01a], [Tam00] and [Sch98]. However, there often occurs an overlapping of these four sectors, and some applications cannot be assigned clearly to one of these four groups:

- Safety
- Security
- Energy management
- Comfort

In the following, there is a description of these four groups on the basis of already existing applications and their extensions. Furthermore, there will be an example of each of them, which will be used again in testing the realization of our system later. These examples are analyzed in more detail in Section 2.1.

SAFETY

Probably the most important subject is the need of sufficient safety of the family or the user, as well as of the house and the installations. First of all technical systems have to be used to protect humans from dangerous situations and to support the occupants in case of threatening or trying actions.

Most of the time current systems will react only if a dangerous situation has already taken place instead of preventing such a situation to occur. Safety systems can be found for instance in the form of fire or gas alarm systems. They will activate the alarm after detecting smoke, fire, gas etc. But they will not prevent these situations by closing the affected gas pipes or by cutting off the power line if they detect high temperatures, smoke, gas or any other unusual properties. Most of the time preventive measures are completely disregarded.

Hence, there are a number of opportunities for improving these existing systems and for developing new applications in this area. Concerning personal safety, on the one hand the environment around a person is to be checked for threats and, on the other hand, it should be possible to foresee future actions of that person in order to perceive further risks. Concerning the safety of the building itself the system has to inspect appliances, components, pipes, wires, etc. for unusual behavior: abnormal temperatures, high power consumption, moisture, smoke etc. are indicators of a malfunction. Additionally, the system has to observe circumstances like “the window is open and it starts to rain” or “a water pipe is broken” and to react in a proper way.

Example: *A child is in the kitchen, no adult is nearby, the stove is switched on and a plate is hot.*

Here, a preventive acting system could be used to identify the current situation as a dangerous situation and to react in an adequate way. It could give both optical and acoustic signals or messages respectively to the child. It could inform adults if they are in an adjoining room and additionally it could switch off the stove. This is a situation which can hardly be solved by using only conventional systems.

SECURITY

Security systems deal with the protection of rooms and buildings. They contain areas like the protection against housebreaking, access control, logging of events, or the identification of individuals (authentication).

Systems of that field can offer a variety of capabilities. They can check for open doors or windows if the user is leaving the room or building. If possible, they have to close them automatically or they have to inform the user. The same applies to the protection against housebreaking. Windows and doors can be checked by sensors for breaking, occupancy sensors and light barriers can detect intruders. As reaction the system can activate the alarm and inform the police or neighbors.

Example: *The system detects the breaking of a window.*

A reasonable reaction in that example depends on distinctly more information than only the detection of a breaking window; it depends on the entire situation. The system has to know whether the owner or some other allowed person is still somewhere in the building. If so, this person has to be informed first. Additionally, if there is a message to the police or someone else, they also have to be informed that there is still someone in the building. Besides, possible reactions may also be to report the location of an intruder if one is already inside the house or to switch on the lights.

ENERGY MANAGEMENT

In the field of energy management there are already existing applications. Again these applications are concentrated on particular tasks. For instance, we can find light control systems which will activate the lighting by using movement sensors and switch off the lighting again after a certain time. Heating controls are able to react according to temperature sensors and can be additionally controlled by programs. Control systems for blinds can contribute to the regulation of the brightness and the temperature in rooms.

All these systems focus on clear divided tasks. Again, there are numerous starting points of improvement. The heating control could use additionally a presence detection instead of a rigid programming that sets the timings for switching on and off, and when heating it could take open doors or windows into consideration. The basic idea of automatically switching off the light which is used by the light control could be extended to many other applications. All electrical appliances could switch themselves off if they are no longer needed: the fume cupboard, if there is no longer any steam or smoke; the stove, if there is nothing on the hot plates; devices, if they are using “standby”, could switch themselves off as long as there is no one present.

Example: *The fridge is switched on, it is opened by someone and everybody is leaving the room.*

In this case the fridge should – if possible – close automatically. Otherwise, the person has to be informed about the open fridge if the system detects that s/he wants to leave the room. Furthermore the system has to report in an acoustical or optical manner if the fridge has been open for too long.

COMFORT

The intention of this field is to organize the stay of an occupant or a user as pleasant as possible. To reach that, on the one hand personal preferences concerning brightness, temperature etc. should be taken into account. On the other hand, the system should take it upon itself to do mechanical jobs like turning off water if it is no longer used or the regulation of the volume of a radio or television set if the phone is ringing.

Especially in this field scarcely any existing applications can be found since the combination of different systems is a prerequisite in this area. For a “person-oriented influence” on these applications, presence detection has to be combined most of the time with other applications, which have worked completely independent so far.

Example: *The phone is ringing in a room where nobody is present. Someone is in the room next door.*

The system has to detect the ringing of the phone and the absence of a person in the room. Thus, it has to search and detect the person in the other room. Subsequently, the situation in this other room has to be analyzed (for example, the person may be asleep). Either the call is relayed to this next room, or the system has to look for someone else.

Only these four examples make it obvious that lots of information is needed in more than just one application. Therefore it will be necessary to create a common base for these applications to enable an information exchange. For example the information of the presence of persons as well as the location finding are used in all four examples, and the possibility of acoustical messages is used by three applications (Figure 1).

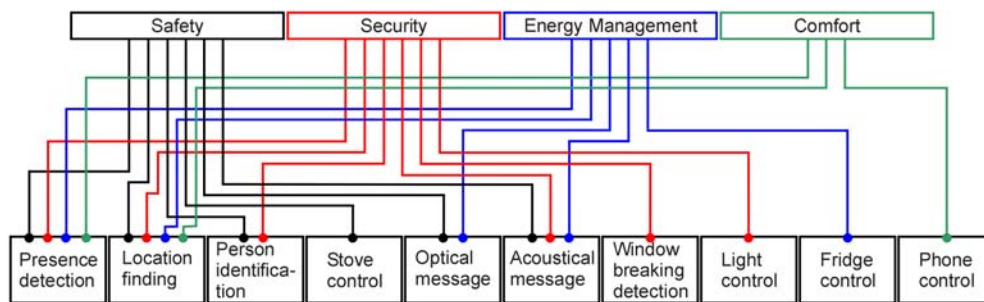


Figure 1: Multiple usages of functions

Up to now, these four areas were regarded individually; usually conventional systems can be assigned to only one of these areas. However, the goal is that *one* system combines these four fields of requirements and reunites the whole of functionality. Only then can it be possible to create completely new applications, which are not realizable using only present tools.

However, only connecting these areas will not be sufficient in order to achieve a preventive behavior of a system. For that, the system has to realize *and* to identify the entire current situation beyond the bare perceiving of information about the environment. Only if it is capable of perceiving situations and of interpreting them as a consequence, conclusions about future events can be made and a preventive behavior can be achieved.

1.3 Analysis of the goal

The goal of the work is the creation of a model for the construction of a system which is able to behave in an intelligent and preventive manner in a changeable environment. The application field of this work is the area of home and building automation.

In such a system, the first step is the ability to perceive the environment, to interpret the information, and to recognize the situation in a building. In doing so it will gain the necessary knowledge to react in a – for the user of the building and for the

building itself – helpful way, i.e. to avoid dangerous situations. An important feature in order to achieve appropriate, situation-dependent reactions is to act with foresight and to contemplate the consequences of its own reactions.

Furthermore, in order to check the design of the model, a system has to be developed founded on that model. Using the four situations described in Section 1.2 this application will be tested. The model has to be based on existing applications in home and building automation.

For a detailed description of the goal of the work, there will be a synopsis of the requirements concerning such a system in the following.

REQUIREMENTS CONCERNING SITUATION-DEPENDENT BEHAVIOR

In the course of the work, on the one hand a model for situation recognition and situation-dependent reactions has to be created. On the other hand, this model has to be verified using an application based on it. As already mentioned the model has to allow the extension of existing applications in home and building automation. Due to that, there are many different factors to be taken into consideration. Therefore these criterions have to be discussed in detail. They have to be investigated if and how they are to be realized or whether it actually makes sense to take them into account respectively. However, some of these factors will be imperative to acquire an intelligent and preventive acting system. Since biology represents the best examples of systems that are living and acting in a dynamical environment, I will repeatedly use these systems to identify the prerequisite of a technical system. To meet all requirements we will use approaches of biological systems as several researchers have done before e.g. [Alb96] or [Yut97]. These are only two examples where nature is the model for technical conversions. The former concentrates on the design of intelligent control systems with biological concepts in general. The latter takes animals as example of a technical system, since human qualities are too complex. More detailed descriptions of works dealing with biological ideas can be found in Section 2.3.1.

Interoperability

As mentioned above, one basic requirement is the usage of existing applications and the extension of these systems. Due to that, it is necessary to make connections between them. Up to now, they are usually working independently. But in order to improve these automation systems they have to be capable of communicating, they have to share information, and they have to be linked together. The information in one system has to be made available to other systems.

The development in home and building automation has established boundaries between the different industries. Therefore, boundaries between systems out of different industry areas exist as well. This problematic is described in [Kab02] in detail. According to this thematic, they divide the communication into:

- Intra-Industry-Communication: the information flow between components within one industry (within one Gewerke)
- Inter-Industry-Communication: the information flow between components out of different industries

In some technologies there are first attempts to deal with that division into industry areas and to solve that problem [WWW3, Rau00]. In this case we have to consider

these approaches in our system as well. For example, [WWW3] uses detailed definitions of templates for real devices, whereas [Rau00] pleads for the usage of a structure similar to a tree with abstract device-definitions in the nodes and the resulting, interoperable devices in the leaves.

Moreover, it will be necessary to examine the usability of the most popular technologies in home and building automation. Hence, we will have to define the requirements on these technologies.

However, since we will use already existing systems and devices, not all of them will be extendable in the desired way. If, for example, a prefabricated tool for sound recognition is integrated, it will probably not be possible to change this application or to apply a method as proposed by [Rau00] to it. Thus, it might be necessary to realize the interoperability on a higher, abstract level. As a consequence, the communication time between some devices or applications will rise and with that the processing time of the concerned functions. This fact will be an important criterion of many applications.

Sensory system

Biological systems possess an enormous number of sensory cells for perceiving the environment. Hence, if we want to adopt particular concepts from nature, our system has to use an extensive amount of different sensors to acquire as much information about the environment as possible. Another feature to be seen in biological systems is the usage of many different types of sensors, the usage of different senses. Again, it could be an advantage to adopt this characteristic.

However, awareness of the state of today's technology, which is far from the density of sensory units used by biological systems, is needed. It is possible to integrate more different sensor types of what one biological individual possesses, but the number in total will be only a fraction. Thus, one may ask whether it is at all possible to realize a situation-dependent behavior under these conditions. Yet, it has to be remembered that in the course of the time technology will evolve, devices will become smaller and cheaper, and the number of them will rise [Aeb00]. Although the mentioned restrictions make it improbable to identify every situation today, this might well be possible in the foreseeable future.

The realization of this work is based on approximately 100 data points consisting of more than 15 different sensor types. Therefore, many situations will not be detectable by this system. Hence, the demands have to be tailored to some selected examples of situations. In doing so, it will be possible to evaluate the principles of the resulting system despite the comparatively small number of sensors.

Structure

The dataflow of the information processing in human beings is organized in a way that, independently of the location of the sensory input, almost all information comes together in a central area called the cerebral cortex [Roh94]. Moreover, the information processing can be structured into several layers which are working in parallel and communicate among themselves. On the one hand, a decentralized structure means a higher maintenance effort because the parts of the system can be located everywhere in a building. On the other hand, it can offer a higher availability because a malfunction would not affect the whole system but just a part of it.

In the human body we can find processes like reflex actions¹, which are working decentralized, initiated by a specific sensory input and immediately led back to a corresponding actuator. [Hal92] has described them as unintentional actions. Deliberate actions can be found there as well, which are controlled by the brain. In the course of this work it has to be evaluated if it is possible to manage actions in home and building automation systems in the same way. It has to be analyzed if there are reactions comparable to biological reflex actions, and if they can be used for a fast response to dangerous situations. At the same time it must be guaranteed that the reflex action itself does not cause an unwanted situation, since it would act without a higher control.

Furthermore, the dataflow of the information processing will be an important aspect. Most of the processing takes place *after* a first handling of the perceived information (for example the transformation of the perceived stimuli, see Section 2.2.3). Tasks like merging of information, internal descriptions of momentary events or the identification of the current situation will have to deal with large amounts of data and will therefore need more processing time. This can sometimes account for directing the information flow to a central area, since with the increasing amount of processing the distances of communication gain more significance.

Concerning the decentralized structure, fieldbusses seem to offer sufficient prerequisites. They show many similarities to the biological nervous system and may therefore play the part of the spinal cord including the sensory system and the task of motor activity. Since there are many differences between existing fieldbus systems, the selection of the suitable technology has to be considered.

Data storage

In the human brain the memory and the structure of it play an important part. The 100 billions of neurons in the brain keep more than 100 quintillions of connections among themselves [Car99a]. Each of these connections has the potential to represent a part of a recollection. Thus, the storage capacity of our memory can be seen as almost unlimited. Nevertheless, it is possible to find and process information in a very short time.

The explanation of this is partly founded on the structure of the memory. [Mai97] describes the parallel search and processing of information in two separated areas, the short-term and the long-term memory (Figure 2).

Another division into timing-characteristics is mentioned in [Kie99]. There, the 3-memory-model of Atkinson and Shiffrin is described. At first it consists of sensory registers, which receive the incoming information and store them for only a few 100 milliseconds. Then the information moves to a short-term memory for some seconds. Only if the information is transferred into the third part, the long-term memory, it will remain for a longer period.

¹ Autonomic functions that are formed by reflex circuits in the central nervous system [Kan00g].

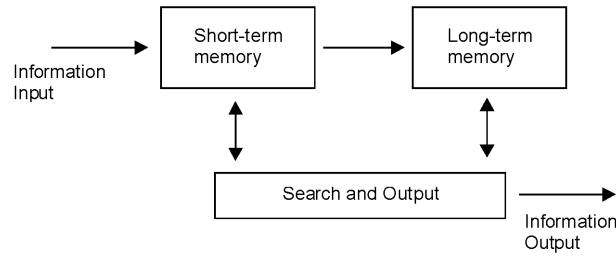


Figure 2: Hierarchy of forming the memory

L. R. Squire [Squ94] makes a subdivision into a declarative explicit memory, which contains information about objects, actions and events, and the nondeclarative memory for motor capabilities.

In Section 2.2 further research works concerning the structure are presented. All of them are related to a specific topic as for example memory, performance, information flow etc. Considering the uncertain outcome of results in various research areas it is not advisable to integrate all results in a technical system. In a considerable number of works, the authors try to disprove other researchers or state a high number of questions that are still open respectively.

Thus, it will be necessary to analyze the demands on a technical system with great care in order to find the right structure of it.

Inevitably, we will have to organize large amounts of data in that system. Since intelligence can be integrated in the devices (for example in fieldbusses), they become smart, flexible, and therefore interesting for a high quantity of applications. The devices will become better known and used; the number of nodes will increase and the price will fall dramatically, which will again increase the number of integrated devices. To handle that amount of data it could be necessary to separate the information in the same way as it was done in the biological studies. There are dynamic, changing information about the environment where it may not be necessary to store them for a longer period. There will be knowledge about users and objects which will be used for a long time. Here, a solution as in [Mai97] or in [Kie99] might be beneficial. The system has to store the “way” how to do something and how to react in a specific situation – perhaps the answer can be found in [Squ94].

However, some of these solutions may turn out to be wrong for the usage in this work since the resulting system must not meet only one of the requirements but be able to comply with all of them at the same time.

Flexibility

Already existing applications often suffer from their rigid structure and consequently from missing flexibility. Sensors as well as actuators are connected with specific applications. Thus, a device can be assigned to another application only if that connection was considered when designing the device. In addition the exchange of information between nodes which were not explicitly defined for that is often restricted or even impossible.

Biological systems have found methods to combine information of most different sensors. For example, as described in Section 2.2.3, shortly after a stimulus is perceived, it is transformed into an information item that can be merged with the information of other sensors. In doing so, there are no restrictions in using combinations of information of different sources; one is able to resort to a variety of sensor types perceiving the same objective, but delivering different information types (for example an individual is able to see a car driving past, to hear it and to feel the draft of it). This flexibility of biological systems is also a goal for the usage of devices in this work. The resources should be available in all applications and they should be combinable and usable in every conceivable way.

However, already existing applications should be used as the base of this work. Only few of them will be designed for the flexible usage necessary in this work. Therefore, new methods have to be developed and analyzed in order to overcome this problem. The aspect of interoperability will be an important factor for the flexibility within the system.

Not only will the structure require flexibility, but also the “logical content”, the knowledge of the system. It is obvious that the system has to store information about previously experienced situations (or simply predefined descriptions of situations) in order to be able to identify the current situation. If we want to enable the system to adapt to the environment according to biological examples, this knowledge must not be static. It must be possible to change the stored information about connections between different data, and therefore about descriptions of objects, events, reactions etc., dynamically in order to get an optimal adaptation.

Behavior

One of the main requirements for the system is to achieve an intelligent and preventative behavior.

An intelligent behavior means the right interpretation of situations in the environment in order to find appropriate reactions to them. Furthermore it requires the system to be able to “consider” the consequences of its actions. Each action and its possible effects have to be evaluated. By that, the reaction that fits best to the situation is chosen and put into action in the real world.

Preventative behavior means that the system supports humans in dangerous situations by preventing them. Once the situation is identified, the system has to analyze the possible following situations. It has to “foresee”, to look for events which can lead to problematic situations, and to react accordingly.

Both, intelligent and preventative behaviors are related to each other. If the system has to contemplate the consequences of its own actions, it has to have an idea about the situation following a reaction. The same knowledge is necessary for the detection of probable, future events. Thus, an important aspect of the behavior of the system will be the knowledge about sequences of events, about scenarios (Chapter 2).

In order to react in an appropriate way, it will be necessary sometimes to carry out a reaction to a specific situation immediately – similar to the biological reflex. And sometimes it can be acceptable to take more time to react, but to be sure of the consequences of that reaction. However, the system should be aware of the current situation all the time. It has to be avoided under all circumstances that our systems make a decision which leads to a dangerous situation for the occupants or the building

itself just because of a shortage of knowledge about the environment or undetected coherences between states and events. Though, even if there is more time available for well-considered actions, the time factor must not be disregarded. An approximate time period has to be kept. There can be several different situations at the same time, which have to be analyzed and handled; to each of them there can be several possible following situations. In a complex environment, the processing time will rise exponential. A factor that is even for humans sometimes difficult to handle (just think about driving in heavy traffic with many cars, traffic lights, pedestrians, etc.) might be almost impossible for a technical system. Hence, we have to look for mechanisms to overcome or to – at least – attenuate this problematic.

Another task, which is not directly connected with the behavior of the system but nevertheless important and necessary, is the definition of all biological terms used for a technical system, notions like intelligent behavior, awareness, etc. The importance of that task can be seen in [Mey00], where an analysis of characteristics of the term “intelligence” has been carried out and six degrees of intelligence have been defined afterwards.

We also have to consider the behavior concerning the starting up of the application. By using a memory structure like [Kie99], the system would possess knowledge about previously experienced situations (if it was the first starting a basic knowledge has to be predefined), but it would not know anything about the states of the current surroundings. Thus, it must be assured that the system has already a complete overview of the environment before it starts to act within it.

Performance

As already mentioned above, the performance of the system will be an important factor in achieving reasonable reactions to situations. It must be assured that the task of situation perception and recognition as well as the selection and application of a reaction takes place in time. If a reaction does not take place at the right time, it might even be better sometimes if it does not take place at all. For example, if a dark room is detected and the system decides to switch on the light but the user has found in the meantime his way through the dark room and has opened the blinds before the system is able to act, it would not make any sense to switch on the light afterwards (besides, such a performance would put the meaning of the entire system into question).

Biological systems use a variety of different mechanisms for an efficient acting onto environmental conditions (see Section 2.2.4). For example, the fact that not every aspect of the environment might be important at the current situation can be used. Therefore, one does not have to deal with every single detail, but it is sufficient to only focus on some important features as it is described for example in [Mai97] or [Alb96].

1.4 Theoretical and practical background

The development of systems for situation recognition and situation-dependent reactions in the field of home and building automation requires a fundamental knowledge in various areas. In the following there will be an overview of the required techniques and principles as well as of the used resources.

1.4.1 Communication

Many of the requirements concerning situation-dependent behavior of Section 1.3 are related to the field of communication. In this work existing applications and technologies have to be used for the sensory system and the actuators. Thus, by selecting these applications as the base of the aimed system, a large part of the principles of the communication is predetermined. For example, by using a fieldbus, the communication between the devices is already given. In case of enclosed applications as for example a pattern recognition tool no communication to other systems might have been considered at all, but the information is simply offered to the outside.

Consequently, the different features of the possible technologies with regard to a system for situation recognition and situation-dependent reactions are to be considered.

CONSIDERATION OF THE COMMUNICATION SYSTEM

For an immediate answer to particular situations this work will make use of functions comparable to biological reflexes. The main feature of these functions is the short response time with as low demands on the communication and processing as possible. At first, it requires the following aspects:

- It is necessary to enable a simple and fast information exchange between the devices of a reflex. The nodes and applications have to communicate with each other as directly as possible without making a detour over other parts of the system. Consequently, there is a high demand of interoperability between all the nodes.
- The previous aspect is closely related to the flexible usage of the devices. In order to achieve reasonable reflex actions, it will be necessary to handle the resources in a flexible way. For example, if a sensor of an enclosed system possesses important information, this information has to be available not only to the devices of *this* application, but also to other nodes which are not explicitly defined for that. By that it might be possible to realize reflex actions which respond faster than the original application since they do not require a complex processing. To clarify this point it is important to understand that the reflex action should not replace the original result of the application. It will only be based on some information and will result in a very fast and simple solution, for example in activating an alarm or in sending a message.
- By distributing the intelligence over the devices, the communication between them can be simplified. This allows to process information in parallel and to exchange smaller data packets. Moreover, analogous to the biological example, there is no need to include a central control of the reflex action. The nodes themselves can carry out all necessary processing.

However, not all reactions will be as simple as reflex actions to handle. Since the resulting structure has to allow an intelligent and preventative behavior, the integration of higher, more complex functions is required. These functions will be

based on a more general knowledge about the current environment; the limited information of a single application will not be sufficient. Instead, it will be necessary to collect information from many different applications and devices.

Though the possibility of a communication between all these systems has to be assured, it is far beyond the above mentioned interoperability between the devices. The higher functions of the system will have to collect data out of various systems, and to merge them. Thus, this communication has to be treaded on a higher level as well.

Nevertheless, there is also an aspect of higher functions which has influence on the choice of technology:

- In order to obtain a general knowledge about the situation in the current environment, it will be necessary to use a very high number of information sources. Thus, there are two factors which have to be considered.

First of all, the chosen technologies must be able to connect and to handle a large number of devices. In a situation-recognition system it will be required to install nodes in various places, for example contact sensors for doors, windows or cupboards, distance sensors for appliances, actuators for optical or acoustical signals for electrical devices, and many more. Therefore, the underlying technology should be able to use as many different media types as possible to reach every necessary location in the building.

Secondly, there is an economical factor as well: considering the quantity, the nodes have to be cheap. However, as already mentioned, exactly this high number of needed devices will decrease their costs.

The large number of nodes and the partly complex processing will probably result in a longer response time. However, it will hardly be possible to define exact values of the timing constraints since they depend on the desired functions. If an answer is expected “immediately” for reflex actions, it will be acceptable to wait “some time” for the result of a complex process. The throughput as well as the reaction time is moderate for systems in today’s building automation. Fieldbus systems generally try to guarantee short reaction times by allowing only small data packets. The chosen technology will therefore determine the limits of the timing constraints of our system. In case of reflex actions, the response time is almost directly connected to the response time of the selected technology since they will not need an expensive processing. On the other hand, for higher functions, these limitations will play a minor role since the majority of their response time will be used for the processing.

Summing up these considerations, it can be said that we have to deal with the following aspects of communication concerning a system of situation-dependent behavior.

- Interoperability
- Flexible usage of the devices
- Distributed structure
- Large number of devices

ANALYSIS

In the following, there will be a description of the three most important fieldbus systems in home and building automation [Fou01] as well as of enclosed applications for specific tasks.

EUROPEAN INSTALLATION BUS

Within Europe, the European Installation Bus (EIB) [Sau01] has gained widespread acceptance, especially in private and office buildings. In 1999, the Konnex Association was founded; its aim was a One-Single-Standard called KNX. EIB is a member of Konnex [WW1, Eib01], and is therefore named EIB/KNX.

EIB uses a variety of different objects which are specially tailored to the requirements of building automation and control; thus, EIB components from different manufacturers are able to communicate with one another.

A maximum sized EIB network consists of 15 areas, whereby each area has 12 lines each of which may connect 256 nodes at maximum. In total, the network can connect up to 57600 devices or nodes. An EIB network is divided into several areas, which again are divided in one or more lines. By that, the network has a hierarchical structure that generally corresponds to the structure of a building, which consists of floors, corridors, rooms, etc.

As physical media EIB uses almost solely shielded twisted pair (STP). The maximum throughput is 9600 bit/s, which is already barely sufficient for today's control systems, which controllers, sensors, and actuators communicate with few data only.

If only EIB/KNX is used as the base of the system, the restrictions with the physical media types as well as the small throughput may develop into drawbacks of future applications, which combine the various functions and the management level of building automation.

LONWORKS

LonWorks (LON) [Die98] has been developed by Echelon, and therefore is of no international standard (it uses the American Standard ANSI/EIA 709.1, in Europe CEN TC247 WG4 is in progress). Nevertheless, it is the direct competitor of EIB in the United States. Its primary application field is building automation as well.

One of the main features of LON is the interoperability by the use of an object/profile hierarchy. Similar to EIB, LON uses a hierarchical structure of the network. At maximum, approximately 32000 intelligent nodes, which are able to measure, process, control, etc. on-site, are partitioned into 255 subnets of 127 nodes each.

LON supports many different physical media types like Twisted Pair with for example standardized RS-485 interface as well as optical fiber for higher data rates. Additionally, the data rates further depend on the size of the network. Using STP wires, the rates vary between approximately 50 kbit/s, if the communication length is about 1.5 km, and 1.25 Mbit/s for shorter distances up to 300 m.

BACNET

The American Society of Heating Refrigeration and Air Conditioning Engineers has created an open standard called Building Automation and Control Network (BACnet)

in order to create a standardized method of interconnecting heating, ventilation, and air conditioning (HVAC) subsystems from different manufacturers.

BACnet defines multiple physical architectures to handle high and low speed networks, as well as point-to-point connections. This emulates the structure of most automation systems.

All features of a device are presented in a device's Protocol Implementation Conformance Statement (PICS). Each device must have some obligatory features and it may contain additional optional features. However, the PICS constitutes a problem, since it allows the manufacturers to diverge in their implementation of BACnet products, which means that devices from different manufacturers might not be interchangeable [Ten00, Zab02]. Furthermore, it was originally designed for HVAC systems; other applications were not contemplated for BACnet, and the protocol has not been optimized for their operation.

BACnet was originally designed to communicate over local area networks (LANs). Several different LANs were defined including point-to-point, LonTalk and Ethernet. Thus, there are almost no physical limitations for a BACnet. Its only limitation is the chosen network technology.

Another advantage of BACnet is the economical aspect: compared to other automation systems, it has lower installation costs and lower life cycle costs (service) [Ten00].

SPEECH RECOGNITION

A speech signal is complex and encodes far more information than can be analyzed and processed in real time [WWW7]. Since the late 1990s, computers have become powerful enough to make serious progress in the field of speech and sound recognition possible. However, today's technology is still far away from reaching a level, where we can make natural, unstructured conversation with a computer.

There are two basic functions in speech technology: speech recognition and speech synthesis.

- Speech recognition (or speech-to-text) tries to analyze spoken words. First, it captures sound waves, generates electrical impulses and digitizes them. Next, these digital signals are converted to basic language units or phonemes, which are merged to words in the next step. Finally, it analyzes the context of the words to ensure correct spelling of words that sound alike.
- Speech Synthesis (text-to-speech) uses the other direction and converts text into spoken language. Here, the text is broken down into phonemes and prosodic¹ symbols. It involves a special handling of text such as numbers, currency amounts, inflection, and punctuation. Finally, a digital audio is generated and converted into acoustical signals.

COMPUTER VISION

Computer vision is the combination of automating and integrating a wide range of processes and representations used for vision perception [Bal82]. It includes many techniques such as image processing or pattern recognition and classification.

¹ Prosodic is the metric-rhythmic handling of speech.

Computer vision is the construction of explicit, meaningful descriptions of physical objects from images. Image understanding is very different from image processing, which deals with image-to-image transformations and not with explicit description building. However, descriptions are a prerequisite for recognizing, manipulating and contemplating objects.

The first experiments in computer vision were conducted in the late 1950s; many essential concepts have been developed in the late 1970s. By then, various areas have arisen, such as artificial intelligence, computer graphics or image processing.

As well as the task of speech recognition, the processes in computer vision are very expensive to use. Thus, they have to be treaded similarly to the “intelligent nodes” of a fieldbus system: running on appropriate hardware, these applications have to process the perceived inputs and pass on only their results.

In case of the fieldbus systems each of them can offer particular strengths in different areas. Hence, one way to take advantage of these different features is by using a combination of them as researchers and manufacturers already have done before [Kan01, Kae00]. The bus systems have to be positioned in a way that it is possible to use the potentials of each of them.

Concerning the additional tools, it has to be considered that these applications, despite their enclosure, offer the ability to pass on the results of their processing. At best, they tolerate changes in their functions to achieve a better and more efficient adaptation to the entire system [Int00].

1.4.2 Testing environment

The Smart Kitchen (SmaKi), which was already object of several works [Sou00, Die01a, Sie02], serves as realization and testing environment of this work. In the course of the SmaKi-Project, this room was equipped with different systems and technologies in order to provide the physical base of this work.

THE WHY

There are varied reasons for choosing a kitchen as laboratory. First of all, it represents a central location in a building. There is a continuous coming and going, and people are doing many different tasks in the kitchen. Therefore, it is the perfect place for perceiving and testing a high quantity of situations with a varying number of volunteers. Moreover, the kitchen is the room in a building where almost 20% of all accidents in a home take place [Bau98]. Since the main objective of this system with situation-dependent behavior is the improvement of personal safety as well as the safety of the building by preventing problematic situations, the kitchen seems to be the right location of the first attempts.

THE WHAT

In the following, the chosen systems are described.

Fieldbus system

The fieldbus technology used in this testing environment is LonWorks. There are several factors for choosing this fieldbus system.

First of all, it offers interoperability between the devices right from the start. By that, the majority of the components used for this work are already able to work together in the required way. Consequently, it offers the right prerequisite of the realization of reflex actions for example.

Another factor is the flexibility when using this fieldbus. There are hardly restrictions in using any desired form of wiring. It is possible to use star, ring and linear structures as well as mixed configurations [LON02], as long as you fulfill the requirements of the used transceiver type [Die98]. That simplifies the necessary laying of the bus wiring. Since it is required to install many nodes in every possible location, this factor has had a strong influence on the amount of work involved in the installation.

Using a 5-MHz LON node, the average reaction time is about 20 ms [Die98]. Even though there is no guarantee for a specified delivery time, it should be fast enough for the aimed system. Thus, concerning the performance, one will not have to expect restrictions on the reflex actions.

Likewise, the possible number of nodes should be sufficient for the situation-recognition system. Though BACnet would offer better prerequisites concerning the number, our sensory system would still be far away from biological examples. Referred to the number and the density of sensors and actuators in biological systems, it will be impossible to construct a comparable technical system just by using current technologies. However, the possible number of nodes provided by a LonWorks network should be sufficient for the testing environment.

Speech recognition

There are two functions which have to be handled by using speech recognition:

- **Acoustic Control:** The user is able to control specific devices per spoken commands. Moreover, this function is used to identify the user.
- **Noise Recognition:** Additionally, the system is able to recognize a specific noise in the kitchen. In doing so it is able to identify for example a breaking window, the opening of the door or running water.

For the recognition of speech, the system is equipped with 3 microphones. As already mentioned, the task of speech recognition will need a powerful data processing. Thus, it is not possible to use just one single application to which all 3 microphones are connected. This application would have to do all the arithmetic functions necessary for the sound recognition. Because of this factor, the expensive processing of the acoustic signals of a sensor could influence or even block the central unit for the other sensors. Moreover, if this central application fails, none of the sensors is working.

Therefore, a distributed solution was chosen. Each of the three functions was integrated into one “intelligent” sensor. Each of those sensors contains a microprocessor which processes the acoustic signals depending on the implemented function, and passes the processed data to the overall system.

The tool is called VOICE EXTREME [Sen01] and consists of a Voice Extreme Module and a Software Development Kit.

The Voice Extreme Module is the central part of the “intelligent” speech-sensors. Each sensor consists of one of these modules, a microphone, a loudspeaker and the components, which are necessary for the connection to the LonWorks network.

The module supports “Speaker Dependent- ” and “Speaker Independent Recognition”, “Continuous Listening”, and “Word Spotting”. For the programming

the language “C” is used, containing some additional functions necessary for sound recognition.

- **Speaker Dependent Speech Recognition:** This type of technology requires the user to go through training exercises. Each recognition word is trained twice by the user in order to create voice “templates”. It can be used to store acoustic passwords for example. A flexible vocabulary in any language and any accent are possible.
- **Speaker Independent Speech Recognition:** This technology is the complete opposite of the Speaker Dependent Speech Recognition. No training by the user is required. It is designed for specific languages, sets of words, called “weight files”, which are created before and loaded into the application memory. Only words contained in these word sets can be recognized.
- **Continuous Listening:** Continuous Listening enables products to respond to specific, discrete commands without pressing a button or waiting for a prompt. Discrete means that the commands are preceded by silence. Both Speaker Dependent and Speaker Independent Speech Recognition can identify the commands.
- **Word Spotting:** Word Spotting technology is an advanced version of Continuous Listening. It offers the ability to extract a keyword or -sound from normal conversation.

Computer vision

Like the speech recognition, the computer vision has to fulfill two tasks:

- **Person identification:** The system has to identify persons by recognizing the faces.
- **Person count:** It has to calculate the total number of persons in the room.

For a first test, there were 4 cameras installed: 2 cameras of the type iBOT FireWire Web Cam [WWW8], 1 Sony CCM-DS250 [Fif94] and 1 Philips ToUCam XS [WWW9].

The main differences between the different cameras are in focusing, zooming, handling of brightness and colors, resolution, and the price. The first two types are using IEEE-1394 [Fif94], the last one the Universal Serial Bus (USB) [Gar96]. The reasons for using different models are on the one hand the varying requirements of the cameras. For example, a camera needs an image of the face of the user for the person identification, and therefore has to have a wide angle of view since we cannot expect the user to be directly in front of the camera. On the other hand, the different camera models were used because of the requirements of the software for computer vision.

Two products have been selected for the integration:

- **OpenCV**
It is an open source product for algorithms for image processing. The Intel Open Source Computer Vision Library was initiated in 1999 by the Visual Interactivity Group of the Intel Microprocessor Lab [Int00]. The main advantage of this product is the open source, that is, the software can be adapted to our requirements.
- **AdOculos**

The program AdOculus serves as a platform of the testing of image processing algorithms [Bäs99, WWW10]. Although it is not an open source product, it offers an interface for the development of extensions.

If there will be a final decision between these two tools only time can tell. Whereas AdOculus offers more functionality from the very begin, the advantage of OpenCV might be a better integration into the overall system.

Interface to the user

For the interface to the user as well as for remote control the i.LON web server is used [Ech00]. It is a product of Echelon and establishes a connection between two LonWorks networks or between LonWorks and the Ethernet. For that, it offers special tags for the access to variables which are connected with the network. These tags can be included into regular HTML-code. Via a webpage it is possible to read the current state of variables as well as to change it.

THE HOW

The network of the Smart Kitchen is divided into five separate channels which are connected via routers (Figure 3). One of them is used to connect the user interface (i.LON); the other four are used for specific application fields in the room.

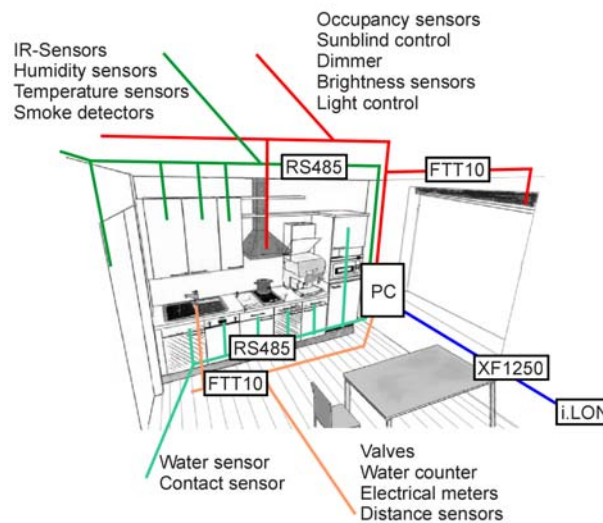


Figure 3: Physical structure of the testing environment

Three different transceiver types are used for these five channels, whereby each of them has specific properties [Die98].

- XF1250

This type is used for one channel and offers a high data rate (1250 kbit/s). It is used for the connection to the i.LON web-server [Ech00], for the user interface, and for the remote control.

- FTT10

The FTT10 transceiver offers a high flexibility in wiring with a data rate of 78 kbit/s. It is used for two of the five channels. The main tasks of the first one are the light control including dimming and brightness sensing, and the presence detection installed in the ceiling. The task of the second FTT10 channel is mainly water and energy management.

- RS485-78

Even though the use of this technology constitutes certain restrictions since this transceiver requires bus wiring, it has the advantage of very low costs. RS485-78 is used for two channels as well. The first deals with the localization of persons, mainly by using infrared sensors installed in the ceiling and collecting information about the environment like humidity, smoke, temperature and many more. The second one is responsible for a high number of contact and distance sensors for doors, cupboards, windows, devices, etc.

Once, the information about the environment is collected by the sensory system, these data have to be passed to the higher layers in a suitable form. Since the higher layers will run on a PC (or several PCs), a standardized interface between the fieldbus and the PC is needed. In the Smart Kitchen, OPC (OLE for Process Control) is used for that interface. The OPC specification is a non-proprietary technical specification. It is a standardized interface based on the Microsoft OLE/COM Technology [Had99, Opc98]. This interface allows the data exchange between nodes and applications of a fieldbus network and applications running on a PC.

An OPC server task is running on a PC, which is connected via a network interface card to the fieldbus. This server task holds a lot of variables which are bounded to host based variables of the network [Ech01]. It can either poll these variables or it can automatically receive changed values, which is inevitably better for the overall performance of the network. Then, the server makes these values available to other applications running on the PC.

The task of the devices connected to this fieldbus is to collect as much information as possible and also to act within the environment. However, as already explained, we will need kinds of information that fieldbusses are not designed to gather and to process.

- Speech recognition

The application of speech recognition is integrated into a hardware module that already contains components for a connection to the LonWorks network. Thus, no further conversion of the results of this application is necessary. Depending on the desired information, a suitable network variable type has to be selected, and will be therefore treated in the same way as a “normal” LonWorks node.

- Computer vision

One application, supported by the information of 4 cameras, has to identify the users as well as count the number of persons in the room. Since this task requires a very expensive data processing, a PC is used for it. Although the results of this tool are already inside the PC, they have to be adapted to the values provided by the OPC interface. In doing so, the sensory system that

supplies the higher layers with information is transparent – in any case, they will receive the data in the same form.

1.4.3 Databases

As already mentioned in Section 1.3, it will be necessary for an implementation to process and store large quantities of data. As well, the temporal behavior plays an important role apart from the basic possibility of the accomplishment of large volume of data. Likewise it must be considered that many accesses to the data can take place almost at the same time and, thus, data consistency has to be ensured.

A possible solution to these requests can be achieved by the application of a database. Nowadays databases offer global functionality with simultaneous high performance. Additionally, the problem of many simultaneous accesses is solved because they are handled by the database as well.

In this work, two different database systems have been available and tested: MySQL 3.22 and Microsoft SQL-Server 7.0.

In [Fal03] detailed measurements are listed. Both systems have been tested in the final environment under real-world conditions. That means, the tests were not running in a sealed testing environment where no disturbing influences occur. Because of that, each test has been repeated several times in order to achieve an average result.

First of all, the following aspects have been analyzed:

- Value type

In Section 3.1.2 the usage of a specified variable type for the symbols is explained. This type will be used for most of the processing. Therefore, the selection of the right type might have a decisive influence on the entire system. In the course of the tests, every possible type of value was used for each access to the database. Section 4.1 summarizes the results of these measurements.

- Large amount of data

In the tests, tables with more than 1000000 entries were used to analyze the performance of the systems. This large number was necessary since a high number of values can be expected despite several concepts of a reduction of information.

1.4.4 Methods for searching and comparing

An important process of the system, which is described later in this work, consists of finding the correct data in a large quantity of information, to interpret the information, and to make correct decisions.

As it will be described in detail later, a large quantity of data has to be reduced systematically to a smaller amount of information. Subsequently, this set of different information has to be looked up and found in a large pool of data. An important factor is that we have to research the task of learning together with the task of search. This is obvious if we consider a search without result, and we have to add – and therefore to learn – new information.

In the following, I will give a brief description of different learning mechanisms. The major goal of these methods is to make a prediction in case of a new, unknown

example. In this work, the focus is more on an efficient search mechanism than on the learning aspect. Nevertheless, learning and searching will use the same underlying information structure as we will see in Section 3.2.4. Therefore, we have to consider both tasks when creating the structure of the system.

SEARCH AND LEARN MECHANISMS

In the field of machine learning a major goal is the classification of new, previously unseen examples besides new search techniques. The system begins with a set of examples and learns to predict the class of each of them based on its features. Next, the user presents new examples to the system, and it attempts to classify them. If the predictions of the classes of new examples are more accurate than random guessing¹, it can be said that the system has learned to some extent to perform the task of classification. Researchers have investigated different methods in this field, for example rule induction techniques or instance-based learning.

Rule induction

The idea of rule induction methods is to generalize the training set into rules in order to classify new examples. There are different ways to represent these rules, like modular rules or decision trees. Rule induction systems evaluate the features of the training set and decide which one to use to distinguish between the different classes.

The following describes two popular rule induction systems.

- ID3

Quinlan developed the tree-inducing system ID3 while trying to find a method to compress data [Qui86]. ID3 reduces a set of input examples to a decision tree. The value of a single attribute determines the outcome of each decision node. It is used for the efficient construction of decision trees for the tasks of classifying and diagnosing. Thus, it can be applied to the automatic construction of knowledge bases in expert systems.

Figure 4 gives an example of a typical decision tree.

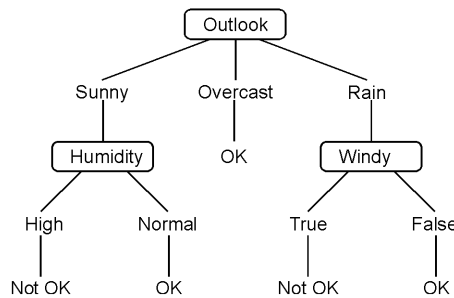


Figure 4: Example of a decision tree

ID3 produces top-down decision trees. Starting with the tree root, ID3 chooses the first attribute to discriminate upon, and produces a subnode to each value. If there is the same class for all examples with a particular attribute value, the node becomes a leaf node; otherwise another attribute is

¹ By a uniform distribution of the classes it means a probability of $1/\text{number of classes}$.

chosen to further differentiate between the classes. When all examples are represented by a leaf node, the tree is complete.

The tree should have the following properties:

- It should correctly classify the given examples.
- It should be able to classify new, unknown objects.
- It should be general and useful, and therefore able to handle as many unknown objects as possible.

In a large number of domains ID3 performs well; it produces compact decision trees with high classification accuracy. Problems may arise if there are dependences between attributes or if there are unknown or missing values. One of its main drawbacks is its sensitivity to noise¹, and subsequently the problem of overfitting².

- CN2

The CN2 algorithm is described in [Cla89]. It was designed to induce “if ... then ...” rules in domains where problems of poor description language or noise may be present.

The CN2 algorithm consists of two main procedures. First, there is a search algorithm which performs a search for good rules, and secondly a control algorithm for repeatedly executing the search.

During the search procedure, it has to evaluate the rules it finds in order to decide which one is best. Here the problem arises that it tends to select very specific rules covering only a few examples. This is caused by the likelihood of finding rules with high accuracy on the training data, which increases as the rules become more specific. In the original algorithm a significance test is used to avoid the selection of highly specific rules. In this way, many rules covering only a few examples are eliminated. Nevertheless, this test may fail in some domains since it only eliminates rules which are below a certain threshold.

Therefore, an improvement of CN2 was made in [Cla91]. There, the Laplacian error estimate is used as evaluation function. By that it is possible to achieve an improvement of the accuracy as well as of the performance of the algorithm.

Comparing decision trees with rule sets, it is obvious that the trees suffer from their restricted readability. A solution can be achieved by the extraction of classification rules (if ... then ...) of the tree. The way from the root to a leaf represents one rule for the class of the leaf. Furthermore, these extracted rules can be simplified and decimated in order to avoid redundancies, and to increase the accuracy of the predictions [Qui87].

Instance-based systems

¹ Unsystematic errors in the data.

² The examples contain wrong values because of noise. The algorithm constructs too complex trees since it distinguishes special cases that do not exist.

Instance-based systems try to store presented examples in such a way that new cases can be directly compared with them. In doing so, they are able to classify new examples, and therefore they are capable of learning. There are two quite different approaches. The first approach, called nearest neighbor, uses a distance function to measure the difference between the new example and those in memory. The case of the most similar example is then used to classify the new one. The second, case-based reasoning is a knowledge-rich approach that uses expert knowledge to link the examples in memory so that the system can quickly locate, and then search the relevant cases to find the most similar one.

- **Nearest neighbor**

Nearest neighbor is a method that originated in statistics. It was first considered for rule production by Fix and Hodges [Fix51], subsequently adopted as a Bayesian approach to non-parametric classification of two-class problems, and used in the field of pattern recognition since then.

A nearest neighbor learner uses a metric that measures the distance between a new example and a set of exemplars in memory. The new example is then classified according to the class of its nearest neighbor. A pure nearest neighbor system stores all examples in memory word for word. Next, it classifies new examples by finding the most similar case in memory and assigning the example to this class. The similarity is determined by using a distance function. In case of numeric attributes, usually a Euclidean distance is taken, where each example represents a point in an n -dimensional feature space. The idea behind is that for a given point in this feature space the surrounding area will share the same class. Furthermore, the Euclidean function assumes that all features are equally important. Thus, they share the same scale in the space, which is linear along each axis. The examples must be clustered into relatively few regions in feature space that share a common class for the Euclidean distance function to work well. If the examples are randomly distributed over the feature space, this conflicts with the assumption that nearby regions in feature space classify the same. The Euclidean distance metric would fail and result in the same classification as random guessing.

However, nearest neighbor methods regained popularity after Kibler and Aha [Kib87] showed that the simplest of nearest neighbor models could produce excellent results in a variety of domains. They tested some simple algorithms which used a normalized Euclidean distance function to classify new examples. The class for each example was chosen according to that of the single nearest neighbor. However, in [Aha92] it is shown that instance-based algorithms are robust incremental learners, but, depending on the domain, the general performance is much poorer than other classification methods.

A series of improvements was introduced in the algorithms IB1 to IB5, showing how the standard Euclidean distance metric is inadequate in many domains. The aim of the study was to overcome five objections to nearest neighbor systems, which were worked out in [Bre84].

- They are expensive due to their large storage requirements.
- They are sensitive to the choice of similarity function.
- They cannot work easily with missing attribute values.

- They cannot work easily with nominal attributes.
- They do not yield concise summaries of concepts.

IB1 uses a Euclidean distance function that classifies according to the nearest neighbor, saving all examples as they are introduced to it. The only variations from a pure nearest neighbor system are that attribute values are linearly normalized before examples are processed, and that missing values are handled by assuming that a missing feature is maximally different to that feature in all other examples. IB1 performs very poorly on domains characterized by noise values, missing values, and irrelevant features.

IB2 differs from IB1 in that it saves only instances that it misclassifies. By that, it reduces the number of examples required by storing only a single exemplar for each important region of feature space. Due to that, it offers an effective way to prune the exemplar database. However, if there are not enough examples of conflicting classes to describe the differences between a new example and the nearest neighbor, examples may be discarded which are important early in the learning process. Thus, the accuracy decreases. With an increasing number of stored exemplars, the accuracy of the model will also rise, and so the system makes fewer mistakes. Nonetheless, there are still problems with noise in the input data. IB2 is more likely to store noisy examples because the classification of them is poor. This will lead to an exemplar database where an excessive number of the examples contain noise.

IB3 overcomes these repeated misclassifications of new examples by pruning bad classifiers. It uses a statistic concerning the accuracy of the predictions of each of the stored examples to determine whether or not they should be used. IB3 keeps a record of the number of correct and incorrect classifications made by each exemplar. If the closest exemplar does not have an acceptable record, its statistics are updated but ignored. For that, IB3 uses the following decision: only if an example correctly classifies new exemplars with a significantly higher degree of accuracy than the frequency of its class, it is accepted for classification. Otherwise, it is rejected, which means it is deleted from the database.

IB4 shows another improvement by assigning weights to the attribute values. In numeric domains where all attributes have similar relevance the Euclidean distance function works well. In most domains, however, this is not the case. By dynamically updating feature weights, the relevance of each attribute may be learned incrementally. Aha proposes in [Aha92], that these weights should be concept-specific: an attribute may be important to one class but not to the others. IB4 weights attributes dynamically. It performs much better than IB1, IB2 and IB3. The presentation of irrelevant attributes has only a slight effect on IB4.

IB5 is able to handle novel attributes. It is an extension of IB4 and handles novel attributes by updating the weight of an attribute only when its value is known for both the instance being classified and the instance chosen to classify it.

- Case-based reasoning

Nearest neighbor methods try to define similarity and differences between historic cases by imposing a metric upon them, the Euclidean distance function. This metric is very rigid and does not represent for example irrelevant or missing features. As already explained, one solution is to add terms like feature or weight to the function. These weights are derived by the system from the input data – they are not formally related to the original distance function. Case-based reasoning methods are based on a different idea; they do not try to measure the similarity between cases numerically. Instead, they try to build a model of the relationships between the examples in memory. These relations may either be induced or supplied by a user. New examples are compared to the cases in memory by ascertaining how closely they match these relationships. Some methods use the relationships to generalize the cases into a hierarchy that is difficult to alter once it is established. Other methods that are more flexible retain all cases; they maintain relationships instead of forming threads that link similar cases. Case-based reasoning is of particular interest in the field of cognitive science. Although nearest neighbor is a successful way of classifying new examples, it provides little insight into the mechanism of human learning. The possibility to calculate the distance between new and previously experienced exemplars in order to draw conclusion about the processing of the brain is lacking. However, it is quite widely thought that people often recall past experiences when solving new problems. For this process, case-based reasoning is considered to offer a plausible model [Bar87].

An example of a case-based system is CYRUS (Computerized Yale Retrieval and Updating System) [Kol84]. The aim of the CYRUS project is to construct and test plausible ideas about how people organize their memories concerning events they participate in. It was meant to be a pure scientific model, not an expert system intended for any practical purpose. The resulting model is a semantic network where a particular node can only be accessed if the key to one of its paths is provided. That means CYRUS is only able to retrieve objects from memory if it can provide exact matches of features pertaining to them.

Chapter 2

Situation recognition

The essence of intelligence is skill in extracting meaning from everyday experience.

Unknown

In a continuously changing environment many different situations occur permanently. Biological systems possess the impressive capability of adapting to this changing environment. Provided there is enough time, nature will adapt to changed conditions in most cases. It seems that time is the key to that ability. But in contrast, a technical system works all the time and nevertheless it will probably fail in new situations. Thus, if time is not the key to that phenomenon, it has to be the way of information processing, and the way how biological systems extend and modify their knowledge of the environment.

However, even biological systems may fail under certain conditions. They still adapt to the environment if there are only slight changes, but not if the surroundings have changed completely. Every system expects particular situations in particular environments. It cannot be hoped to create a technical system which is able to handle any situation. Instead, we have to concentrate on the application field and the situations which can occur in this field.

2.1 Scenarios

We cannot start to deal with *Situation recognition* before defining the terms necessary for this task. In Section 1.3 the term *scenario* is mentioned, and now we deal with the term *situation recognition*. Hence, a differentiation between *situation* and *scenario* is needed firstly.

In [Yau02], the fact is criticized that there are no well-defined concepts of situations up to now. In the following, situations are defined as an expression of previous device-action record over a period of time. Thus, a situation is seen as a sequence of events or states of devices over time.

[Mor97] outlines a situation reactive system, which uses a series of situation transitions of tasks for estimating the future of tasks or the motion of targets in order to achieve appropriate actions. Hence, the main emphasis lies rather on the transitions within a series of situations than the descriptions of the single situations.

In contrast to this definitions, [May01] describes a situation as a unique moment which cannot be repeated; a moment which represents the immediate, concrete reality to everyone. Contrasting, a scenario is defined as the composition of possible sequences of events concerning a particular part of the environment.

[McC69] describes situations in the same way. To him, a situation is a snapshot of the part of interest out of the entire world in one specific moment, and is described by a set of logical formulas.

In the following, I will overtake the definitions of [May01] and [McC69], and treat them in more detail in Section 3.2.4.

To interact with the environment it will not be sufficient to react to particular situations. A situation represents a still frame, the current state of the surroundings, like a photo. Therefore, an image of a situation does not contain very much which is of significance concerning the happenings that have led to this particular situation. For example, if we see a picture of an open fridge, it could mean that someone has opened it right now to take something out. But it could also mean that someone has forgotten to close it and it is open for a long time. Thus, just by seeing the picture with the open fridge, we would not know for sure how to react. Instead, we have to look at an entire sequence of situations, at a *scenario*. A scenario describes the sequence of events of a specific period, and the result of a scenario is a particular situation. Let us illustrate this with an example: if you hear a single note, a single sound – it will not make much sense. Only in context with other notes we will be able to recognize a song. Hence, if we want to react to a situation, we always have to look at the images of the environment over some time in order to recognize the situation to which this sequence of images will lead. Only then will we be able to handle that situation in time.

Additionally, there is another advantage. If we follow several images of situations, we will probably have an idea about the next sequences, we will be able to estimate what follows next. Therefore, we can be prepared for the events, and can thus react even faster and more efficient. By that, we are able to fulfill the requirements of preventative behavior demanded in Section 1.3.

In the following, there will be a detailed description of the four examples of scenarios of Section 1.2.

Example of **Safety**: A child is in the kitchen, no adult is nearby, the stove is switched on and a plate is hot (Figure 5).

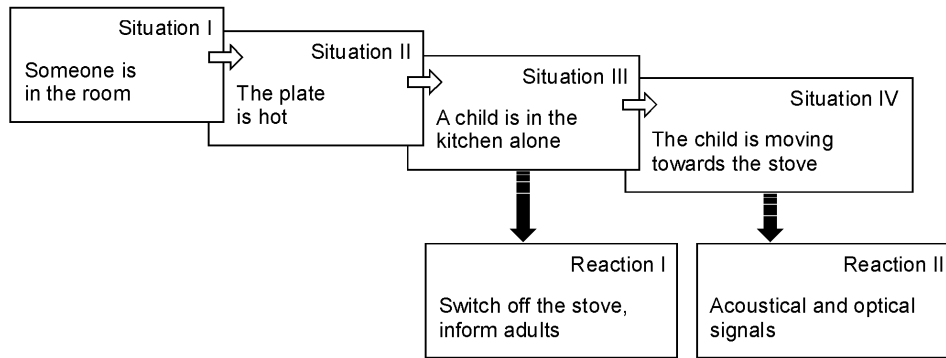


Figure 5: Scenario of Safety

After the second situation our system is prepared because it knows the dangerous situation which can follow. Situation III, where the child is in the kitchen and no adult is nearby, is sufficient for first reactions: the system has to switch off the stove and inform adults, if they are in an adjoining room. If the scenario continues, the system has to send optical and acoustical signals to the child, as is shown in Situation IV. This situation can be split up into several consecutive situations representing different distances between the child and the stove. Depending on the distance, the reaction could start with an acoustical warning and increase in volume with the diminution of the distance.

Example of **Security**: The system detects the breaking of a window (Figure 6).

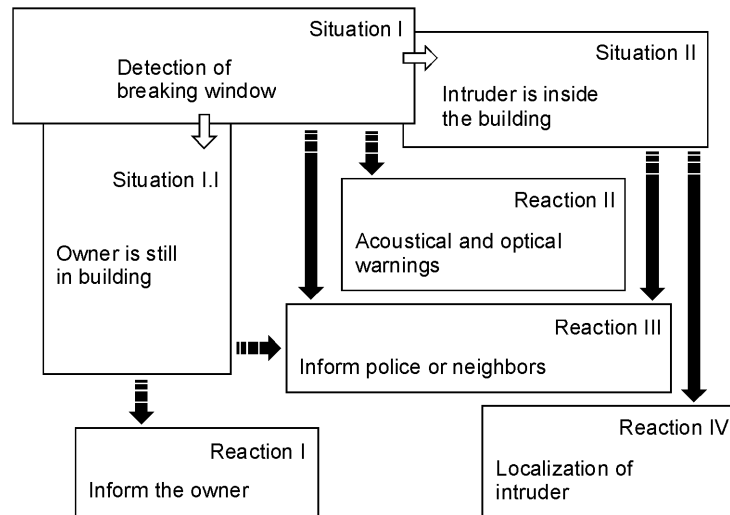


Figure 6: Scenario of Security

By detecting Situation I the system has to check for Situation I.I immediately. If the owner or some other allowed person is still somewhere in the building, they have to be informed about the breaking window. Additionally, the system has to give optical and acoustical warnings (switching on the light and raising alarm). Besides, it has to inform neighbors or some other persons about the broken window. If an intruder is

detected (Situation II), the police have to be informed. Moreover, the messages to the police or someone else have to include additional information about the presence of the owner and whether there is an intruder inside the building. Furthermore, Reaction IV indicates the localization of the intruder. By that, the system is able to include the position of an intruder into its messages as well.

Example of Energy management: The fridge is switched on, is opened by someone, and everybody is leaving the room (Figure 7).

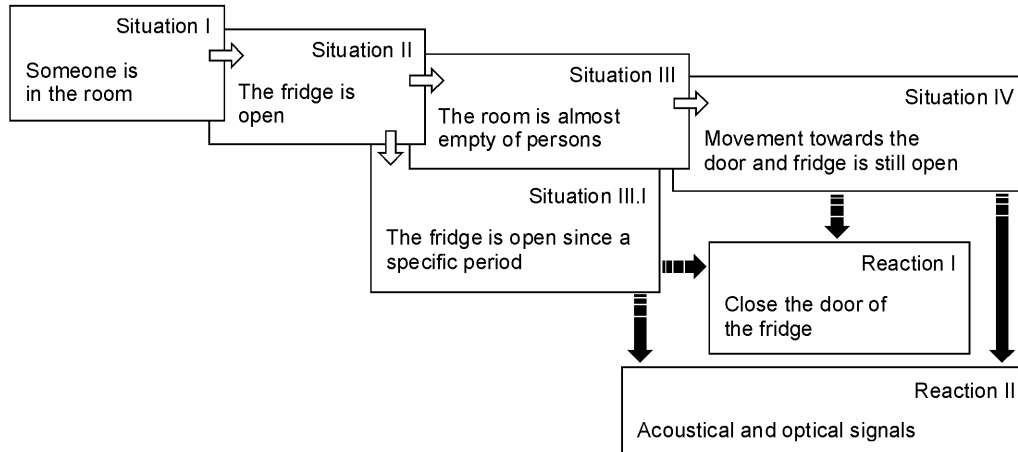


Figure 7: Scenario of Energy management

After Situation II the system can expect two different following situations: either everyone is leaving the room or a specific period goes by – in both cases the fridge is still open. If none of both happens, everything is ok (at least for this scenario). There are two ways of acting if the system has recognized a problematic situation: in Situation III.I as well as in Situation IV the system has to close the fridge by itself, if possible. Otherwise it has to inform someone. In case of Situation III.I it has to inform either someone in the kitchen or someone in an adjoining room by acoustical or optical signals. In Situation IV it has to remind the person that wants to leave the room of the open fridge.

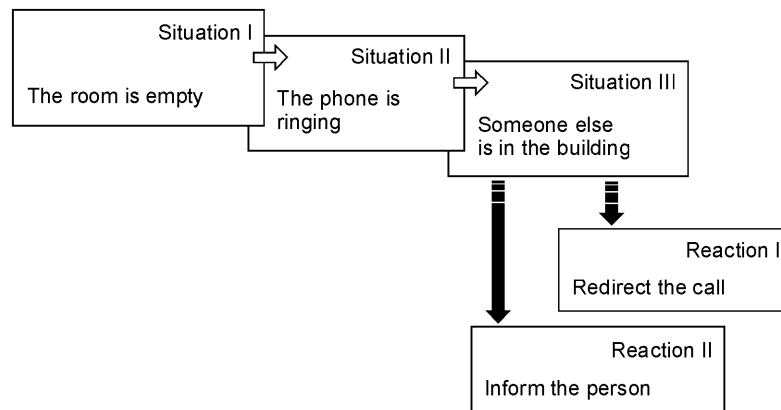


Figure 8: Scenario of Comfort

Example of **Comfort**: The phone is ringing in a room where no-one is present. In the room next door is someone (Figure 8).

In this example the system has to take the entire situation into account. If the phone is ringing in an empty room, it has to find the nearest person. Subsequently, it has to check the situation there. If the person is sleeping, the system has to continue the search. Otherwise, if possible, the system has to divert the call to this person or it has to inform the person about the call by acoustical or optical signals.

As we can see, there is more than only one possible reaction to one scenario. Sometimes, the reaction depends on the technical abilities of the system (closing the fridge automatically / inform person), and sometimes it depends on the state of the scenario, on how far the scenario has advanced (performing Reaction I in one situation, and carrying out Reaction II if the next situation is reached).

Additionally, we can see that the system, by using scenarios, is able to be prepared for the next situations. Hence, it can start to react already during the course of the scenarios instead of reacting only to the final scenario.

2.2 Biological systems

In the following, I will present a number of different works about highly diverse research fields. These works deal with the structure of biological systems, with differences between them and technical copies, they try to find answers to still unsolved questions, and try to explain functions in biological individuals which are not understood so far. In the course of the work I will refer repeatedly to these researchers and employ their results. Thus, we have to examine some of the concepts in extensive detail in order to be able to decide whether such a result might be a reasonable solution to this work. This might appear slightly exaggerated in a work in a technical application field, however, it is necessary.

Biological organisms are the best examples of systems which exist in and with a continuously changing environment. Despite partly rapid changes of the environmental condition, natural systems show an enormous robustness and a most extensive adaptability. Often even simplest organisms are able to act more clever and target-oriented than today's technical systems.

Compared to the computational power of today's technical systems the human brain is processing relatively slow. The stimulus conduction in nerve tracts is between 1 m/sec and approximately 130 m/sec. In contrast, signals are transmitted with approximately 100000 km/sec in electrical lines. The high transmission speed results from the circumstance that there are only electrons moving. In the biological nervous system, however, the signals are transmitted electrochemically by the shifting of ions. The fastest neurons in the cerebral cortex have switching times with a maximum of 1 Kilo Hertz. Nevertheless, a biological brain is able to manage many tasks like pattern or sound recognition much faster and more precisely than computers. Moreover, it is able to orient faster in a new environment, and to adapt better to new situations.

The reason for these apparent contradictions is founded not only on the massive parallelism by which the brain is built up. For example, [Hei98] describes a phenomenon named “selective spatial attention”. It is the ability to concentrate the mental, intellectual resources onto selected events, and to process these events more effectively by that. It is comparable with a spotlight, which makes the events in its cone of light brighter than the rest. In Section 2.2.4 additional reasons for the efficient information processing of the brain are stated.

Consequently, nature is to serve as a model for a technical system of situation recognition. In the following, different models, which are the results from studies of biological systems, are presented and examined for their usability regarding the achievement of the objective.

At first, we want to investigate the necessity of possessing a form of consciousness in order to interact with the environment in an intelligent manner. There are both supporters of the theory that consciousness is not necessary, and scientists who argue that in particular conscious perception has had an important influence on the evolution [Str00]. Therefore we want to analyze this aspect at first, and scrutinize the fundamental structure of biological systems later.

One of the best-known examples of a system which is working without any internal representation or conscious perception is the subsumption architecture.

2.2.1 Subsumption Architecture

There is a wide-ranging controversy about the way natural systems interact with the environment. Accordingly, there exists a high quantity of models and descriptions of these behaviors.

R. A. Brooks made an important contribution in this area. He has presented his work for the first time in [Bro86] and describes therein the behavior of low animals by means of the “subsumption architecture”, a description of the synergy between the perception and the acting in low animals as for example insects.

The behavior (he calls it “activities producing system”) is structured in different layers. The lowest layer thereby is as simple as possible, but nevertheless complete. It serves only for simplest behavior rules as for example basic movements or reflexes. However, it is fully functional on its own. Succeeding, a second layer is added. This layer offers a somewhat more complex behavior, without worrying thereby about the most fundamental rules. These two layers operate now in parallel, the one “knows” nothing about the other. Difficult functions are now processed by this second layer – all basic movements needed for that are still mastered by the first layer. Next, the system is extended by a further level. Again, this new layer is responsible for more complex functions than the previous one, and again all levels operate in parallel. Thus, there will be an incremental path from a very simple system developing towards a complex, autonomous, intelligent system in the course of time. An extending number of layers, which are increasingly intelligent but which priority in influencing the controlling of actions decreases, will be added.

A deciding point in this architecture is that there is no central, internal representation of the environment. Each layer operates for itself, without knowledge of the other levels. This point of view is also advocated by Charles Sherrington, who proposed in 1906 that simple reflexes, stereotyped movements, are the basic units for movement, and combining these simple reflexes can produce complex sequences of movements.

For much of the last century this view has been the guiding principle in motor physiology [Kan00f].

However, there is also a variety of different works, which rely on the assumption of an internal representation. In [Kie99] a list of different researchers is stated who are of the opinion that at first a sensory analysis of the surroundings takes place. Based on these perceived characteristics, a specific internal representation of the objects of the environment is created.

For a – at least partial – representation argues [Web98]. In this work it is pointed out that insects compare an internal representation of the environment with stored pictures in order to orient themselves. A large number of experiments has shown that many insects are able to find their way home by special features of the environment. Insects actually use such features not only for determining the positions of important locations but also to find entire routes. By using a type of image they are searching for a certain place. They detect places if a positive comparison between the received image and a stored image is achieved. Moreover, they can search these stored pictures for certain characteristics (positions, orientations, distances, colors etc.). Due to their enormous number of different sensors they are able to perceive most diverse types of information and interpret them, and take them into account for information processing. For example, it is well-known that insects are able to determine their direction by means of the position of the sun, by the polarized sunlight or also by a magnetic compass.

SUBSUMPTION ARCHITECTURE VS. COGNITION

Even if the use of the subsumption architecture alone may be sufficient for certain, very simple fields of application, it will be impossible to achieve a foresighted behavior which anticipates consequences. According to [Hal92], the system has to additionally possess the possibility of learning. This again is not possible in applications without cognition [Str00], since there are only direct and unalterable linkages between certain attraction classes and certain reaction types. Therefore, the system has to possess a cognitive character, which is described in [Str00] as following:

- Cognitive systems are integrated in an environment where they take action, and with which they exchange information.
- By representing system-relevant aspects they control their actions in a flexible and environmental-adapted manner.
- The information processing of cognitive systems is determined by the ability of learning and anticipation.

Cognitive functions enable the estimation of the course of actions and their consequences. Subsequently, decisions for action alternatives result from it. Thus they contribute substantially to the improvement of the verification of actions. From the evolution-theoretical perspective one can even say that cognition has arisen for the surviving on account of these helpful functions [Str00, Flo97].

Due to the above mentioned description of the cognitive character it is now obvious why the architecture of [Bro86] cannot be sufficient for a preventive behavior:

- There is no representation of the environment in the subsumption architecture
- Anticipation is possible due to cognitive functions

- Cognitive systems possess the capability of learning
- Regulations with no cognitive character are not able to learn
- Cognitive systems possess a representation of the environment

On the one hand, systems based on the subsumption architecture have no presentation of the environment, and they are not able to learn. On the other hand, these aspects are necessary for anticipation and for a preventive behavior. Since this work is dealing with foresighted and considered behavior, it is now apparent that cognitive systems require an investigation in more detail.

2.2.2 Cognition

In [Oxf94] we can find the following definitions:

Cognition:

(psychology) action or process of acquiring knowledge, by reasoning or by intuition or through the senses

Conscious:

Adj. Knowing what is going on around one because one is able to use various senses and mental powers. ~of sht conscious of something (/ that), aware, noticing

Consciousness is the knowledge of the own status with regard to the environment, and contains among other things the sensorimotor coordination and proprioception¹, object recognition, navigation, logical processing, and social behavior [Flo97]. “Being conscious”, Floreano describes in this work, “is the process by which an intelligent control system performs spontaneous self-monitoring of internal states (which could take into account not only neurally generated activity, but also physiological states) by putting them in relation to the external environment.” Thus, although consciousness is an internal reflexive activity, it is not purely subjective and disjointed from the external environment, but it is intimately linked to purposeful behavior. It allows comparison and processing of information coming from several sensory modalities, and it provides the system with the ability to make predictions in order to change its course of actions. By spontaneously monitoring, cross-correlation, and variously arranging several internal processing states in relation to what happens in the environment, one can anticipate different behavioral outcomes and act accordingly.

In [Roh94], the biological sensory system is divided into three functional areas (Figure 9):

1. Sensors which are specialized in particular modalities of inputs
2. Control center in the brainstem, the midbrain or the interbrain
3. The cortex – it can be divided into a hierarchical structure in primary, secondary, and tertiary centers

¹ Proprioception is an automatic sensitivity mechanism in the body that sends messages through the central nervous system in order to recruit the proper muscle groups needed to counteract any outside force.

Each sense supplies only specific frames. This mosaic of individual pictures is assembled to a unit only by thinking. Several sensory organs are necessary for the complete picture. The created picture of the environment does not represent an immediate, direct picture of the environment, but only selective stimuli from the environment. Nevertheless, in this model there exists an overall representation of the perceived world.

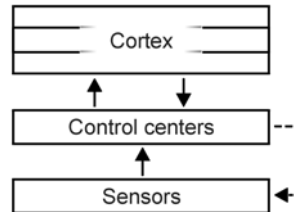


Figure 9: Biological Sensory System

Many different disciplines like philosophy, psychology, neurology, AI, engineering, computer science or mathematics are dealing with the understanding of the nature of the consciousness. Non-invasive brain imaging, lesion studies, and single and multielectrode studies on animals and humans have given enormous wealth of knowledge about the mechanisms that support consciousness.

Nevertheless, the adaptability of natural cognitive systems, perhaps the most superior ability and at the same time the one with the greatest differences to technical systems, is still neither reconstructable nor even explainable so far. Despite a huge number of biological studies and technical algorithms as for example in [Wro00], technical systems still lag behind biology.

Another aspect in that field is the awareness about the perceived information. Again, there exist theoretical studies without equivalent technical conversions. One reason for that situation can be found in the heavy demands made on such systems. For example, according to [Gla00], perceptive awareness includes the modification of stored, internal images by “learning”, regarding the achievement of goals. It is quite difficult to develop a technical system when even the theoretical base is not completely settled yet.

Cognition and learning are also the thematic of [Coe01]. They deal with the task of continuous and real-time learning in robotics and point out that it is difficult to find a technique for the integration and classification of a large number of continuously changing sensors, motor and cognitive signals. It is important to recognize the current situation and to learn appropriately for that context. That requires having a cognitive system as well as a large number of sensors and actuators. They analyze the cerebellum, the vertebrate brain structure, concerning sensorimotor coordination. Due to that, they investigate how the input layer of the cerebellum encodes contextual knowledge in a representation useful for coordination and for learning.

PURE COGNITION?

The fact that cognition is basically necessary has already been sufficiently described. Now, the question arises whether the use of pure cognition is enough to meet all requirements

[Roh94] describes a model in his work, in which sensors perceive information, process them and pass the results on to higher layers. In the cerebral cortex these results are assembled to a representation, to an overall image. However, there are also processes which take place exclusively in the lower layers, and therefore never reach into the higher layers¹.

In [Mai97] the central nervous system is described as a hierarchy of organized substructures with increasing size and complexity. This structure is starting with ions, molecules, cells and synapses. There are the first signal transmissions between the synapses. These form the base for the dynamic of the next layers: the neural networks, the topographical maps, and the subsystems of the central nervous system. Finally we achieve perceptive awareness, complex movements, thinking, and consciousness. Figure 10 shows this hierarchy with a chemical synapse, a local neuronal network for a cellular circuitry in the visual cortex, and subsystems of the visual cortex as examples.

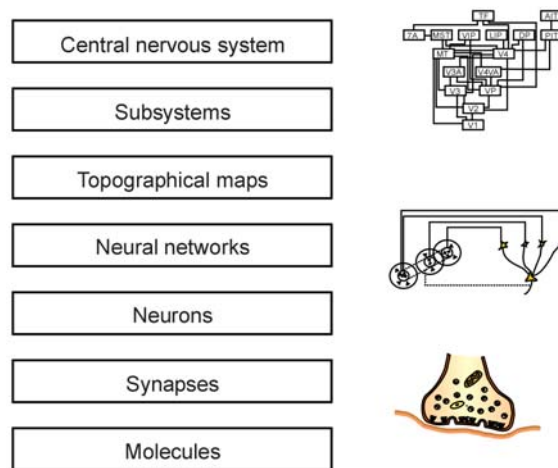


Figure 10: Structure of the central nervous system

Another work which is dealing with that thematic is [Str00]. It is explained that cognitive systems are no substitution for non-cognitive systems. Instead, they are an extension; they are based on non-cognitive parts. Sometimes, there can be reciprocal disturbances, if, for example, an intended action is interrupted or prevented by a reflex action of the non-cognitive part.

The results of these studies show that in natural systems both, consciously controlled higher functions as well as lower functions like reflex actions without any complex control, are used.

Another evident point is that many researchers use a kind of model or structure for describing natural systems.

2.2.3 Structure of biological systems

As well as in the controversy of cognition or non-cognition there exist different models for the structure of biological systems. On the one hand, they differ in the

¹ The senses in the viscera usually produce no conscious perception. Their information will not reach the cerebral cortex.

structure itself, and on the other hand, there are physical models as well as logical ones.

STRUCTURE CONCERNING THE INFORMATION PROCESSING

In the human body, the information processing takes place in different distributed areas (in the sensors) as well as there is the tendency to concentrate the nervous masses in the head [Roh94]. Thus, the information is decentralized in the sensors, just as it flows together in central areas such as the thalamus or the cerebral cortex, shown in Figure 11. Each sensor perceives information by using several neurons. It then processes these data and transmits the results to the higher layers. The representation of the environment is therefore dismantled into separate pictures and will not be reassembled until it reaches the cerebral cortex. As already mentioned, there are processes that only occur in the lower layers, as well as direct connections between periphery sensory cells and the cerebral cortex, whereas the layers between are skipped¹.

According to this description, [Car99] points out that in principle all sensory cells have the same task: they translate specific stimuli into electrical impulses, which are then passed on by afferent nerve tracts. However, the different types of sensory inputs are not differentiated. Instead they are rather adapted.

An analogous description can be found in [Foe02]. Von Foerster states that more than 150 years ago, the physiologist Johannes Müller had already formulated this observation, which he called the “principle of specific nerve energies”. He discovered that nerves of the different sense organs responded to different kinds of stimuli such as light, sound, and pressure in their own specific way. This process happens independently of the physical nature of the stimulus that triggers off the sensation.

The thalamus is described as the central part of the brain in [Jov97]. Nearly all information in the human body, which is received via sensory cells, reaches the central thalamocortical system. The thalamus serves in this connection as relays station and transmits the data to the responsible sectors in the cortex.

Therefore, the first data processing takes already place in the decentralized located sensory cells. This interpretation is also supported by researches in the field of experimental psychology [Kan00c]. In this work, among other things, the processing of perceived inputs is dealt with. It has been detected quite early that, in spite of the fact that all human senses are different in the way of data collection, all senses have three essential steps in common:

1. A physical stimulus occurs
2. The stimulus is transformed into nervous impulses
3. This impulse is followed by an answer in terms of a perception or a conscious experience of a perception respectively.

After the thalamus the combined data are separated again and passed on to different areas in the cortex. These sectors are responsible for different tasks, and work in parallel by using communication between them (Figure 11).

¹ In the olfactory system the sensory cells are connected with the cerebral cortex without interposition of the thalamus.

The question about the origin of a stimulus is a thematic in [Gel98]. The research area concerned the examination whether the location of a physical stimulus is also the location of the perceived stimulus. Tests with the sense of touch have shown that there are pseudo stimuli – without a real stimulus a touch was perceived by the subjects.

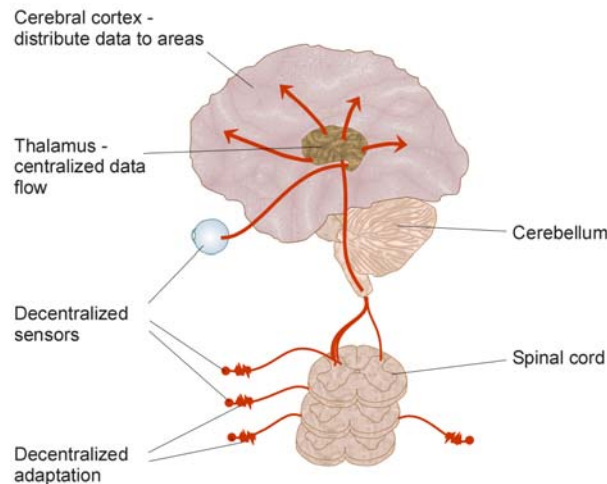


Figure 11: Structure of Information flow

Further tests with the visual sense have yielded the same results. For example, the researchers have demonstrated that there is a visual perception even in the blind spot of the eye. They emphasize that there are still many unsolved problems which allow only speculative answers. However, it seems obvious that stimuli do not arise at the locations of the physical stimuli but appear centrally, most probably in the brain.

STRUCTURE CONCERNING DATA STORAGE

L. R. Squire [Squ94] defines two kinds of memory. The declarative, explicit memory contains the knowledge about objects, processes, and events – about facts. This part is called “knowledge that”. The information stored in the nondeclarative part is motor abilities in general, for example the capability to do a particular task. Due to that, the nondeclarative memory is called “knowing how”. [Car99a] uses the same subdivision of data storage concerning the content.

Concerning the lifetime of stored information, [Mai97] describes two separated areas, the short-term and the long-term memory.

Other models concerning timing-characteristics are stated in [Kie99]. For example a 3-memory-model of data storage of the year 1968 is described, which uses in addition to the two areas of [Mai97] a sensory register which holds the incoming information for only a few 100 milliseconds. This 3-memory-model was criticized by other researchers in the following, and the short-term and the long-term memory were merged into one part. In the course of time, this new structure was followed again by a subdivision into two areas of short- and long-term information, succeeded by subdivisions of these two areas themselves, and new areas were introduced and rejected by other researchers afterwards.

To sum up the results of works concerning data storage of the brain, it can be said that most of the questions are still unsolved. Experiments have shown that the one or other subdivision seems to be reasonable; however, in many cases subdivisions are reduced to conjectures.

STRUCTURE CONCERNING FUNCTIONALITY

A remarkable characteristic of mammal brains is the partitioning of the cerebral cortex into several sectors, which are assigned to different functions [Sin98]. Due to today's knowledge a large part of the cerebral cortex can be divided into sectors equivalent to the human senses (Figure 12). The correlations between these areas and the senses are described in [Car99].

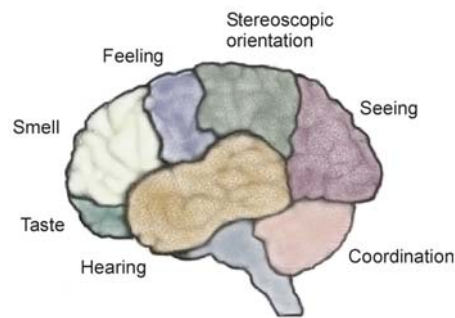


Figure 12. Human senses

Another structure is made by Johannes W. Rohan in [Roh94]. He has investigated the human sensorimotor system and has built a model for the working method of the system.

The sensorimotor function-circuits are responsible for the motor activity of humans. Afferent neurons¹ in the muscular system pass on stimulations from the periphery to the central nervous system. From there, the stimulations can be redirected to efferent neurons². The spinal cord is responsible for controlling simple motor functions. The more complex the movements, the higher are the layers in the brain, which are switched into the control loop of the spinal cord. The entire process can be divided into five functional systems (Figure 13).

¹ Afferent neurons perceive the information from the outside and pass it on to the inside.

² Efferent neurons work in the reverse direction and pass the information to the outside.

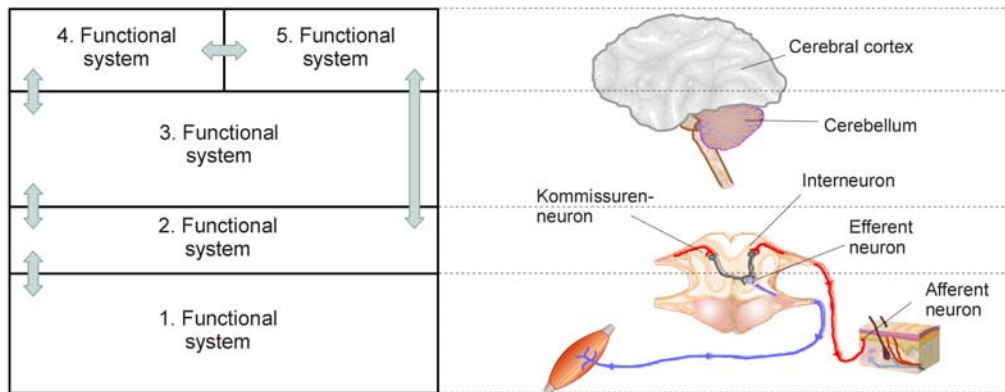


Figure 13. Functional system

The **1. Functional system** is called the basal sensorimotor system, since all other systems are based upon it. In the simplest form it consists of an afferent neuron for the transmission from a muscle to the spinal cord, and an efferent neuron for the transmission from the spinal cord to the muscle. Its task is, among other things, to keep the muscular system contractionable.

However, simplest movements are impossible without the **2. Functional system**. It contains the propriospinal tract of the spinal cord¹, and combines groups of muscles for specific single movements by using the first functional system.

The **3. Functional system** is located in the cerebellum. It receives information by the lower 2. system as well as by the higher 4. system. The task of this system consists of balance and harmonizing functions, hence it has to correct the received information. The results of the corrections are given back to the corresponding systems.

The **4. Functional system** is placed beside the 5. system, the pyramidal system. Both are closely connected. The 4. system is used for complex learned or automated movements.

A transmission arc from the spinal cord to the cerebral cortex forms the **5. Functional system**. It enables the realization of highly differentiated, conscious, and goal-directed intended movements. These movements can also be trained consciously.

With the term “intelligence” deals [Mey00]. In this work, the concept of this phenomenon is characterized and six degrees of intelligence are defined. Finally, they describe how to model intelligent systems based on hierarchies of Elementary Loops of Functioning (Figure 14).

¹ The propriospinal tract of the spinal cord consists of interneurons, commissural and association neurons

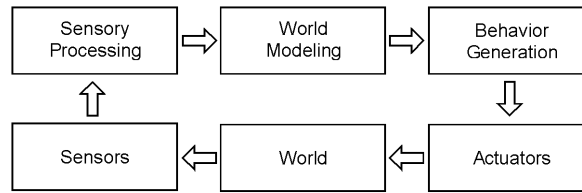


Figure 14: Elementary Loop of Functioning

2.2.4 Performance strategies of biological systems

With growing complexity of the system also the amount of data which has to be handled will rise. In dealing with a large amount of information in an efficient way, we can see a conspicuous difference between technical and biological systems. Whereas technical solutions rely on an increasing processing power, nature deals with this aspect in a cleverer way. We can find many different mechanisms in biology in order to achieve results in an efficient way despite a relatively low processing power. In the following, I will present some of these strategies, which have proven to be useful for this work.

One possible way of handling a big amount of data is achieved by reducing this amount.

PROCESSING IN SEVERAL STEPS

Again, there are different methods of this reduction, for example by the processing of data in several steps. [Kan00e] states that for example the task of visual perception is divided into several layers. Not every detail of the environment will be perceived at once, but in each layer is just some of the information. At first, we perceive only single line segments, put them together, and perceive the entire form in the next step. This process goes on separately in both eyes until it reaches a level where the information of the two eyes is merged.

[Sta97] explains our perception in the way that a description of a representation is multiscale: we work at first with “big pictures”, containing only main features, and then we go into more details. In Section 3.2.4 I will use the terms *coarse* and *detailed* for this classification. At first, we recognize only a coarse representation of the environment, containing only very few features, and in the course of several steps increasing information items are added.

By using this idea in the task of perception, the processing of the perceived information does not have to wait until the entire perception is finished, but can already start after some first information items are collected.

PROCESSING BY USING A FOCUS

Nevertheless, there even may be too many coarse information points in the environment. Thus, researchers as [Kan00d], [Hal92] or [Mai97] describe another feature of biological individuals. Not every single point of the environment is important at the moment, and therefore the attention is focused onto those few important objects or events. These points of attraction can mean changes in the

environment, movements, sounds, smells, colors and so on. For the handling of these points there are also different methods possible.

H. Heinze describes in [Hei98] the ability to concentrate the mental and intellectual resources onto these selected features. The processing of this information will be favored above all other perceived data.

In [Jov97] the way into the other direction is analyzed. Instead of favoring the important points, he describes the blocking of sensors which are not focusing onto relevant features. This process achieves a kind of filter which allows only the relevant information to pass.

These two basic concepts, the processing in several steps as well as the focusing onto relevant points, might offer big advantages in a technical system. It would be sufficient to perceive only some information about the environment at first. Depending on the importance of this information, the focus of attention is directed to one or another point. Then, only the information concerning these aspects has to be perceived and analyzed in detail, which can be done in several steps to enable an immediate starting processing.

PROCESSING BY USING SYMBOLS

There is another important area to which several researchers have given their attention: the usage of symbols. S. Starks states in [Sta97] that one of the reasons for the efficient work of the brain is the usage of symbols. We do not store or process the perceived information point by point as a computer normally does. Instead, we store and remember descriptions in a very compressed form by using, for example, symbols.

[Kan00b] points out that language, mathematic, or the reading of musical scores are obvious examples of the use of symbols, and that symbolic representation is an important component of virtually all human behavior.

The abstract representation of the perceived different stimuli is one of the research targets of [Kie99]. It deals with the semantic structure of the internal representation by using "objects" (comparable to the already mentioned symbols). These objects are structured into categories; meaning that some objects are more similar to others since they possess similar properties, and they are associatively connected to other knowledge contents of this memory structure. Thus, if the knowledge about one object is activated, you also have access to information that is far beyond the stored knowledge about the actual object.

[Car99] explains that the different types of sensory inputs are adapted. Though the sensory cells perceive most diverse stimuli, there is no differentiation between them later. Thus, there must be a common understanding of these different information types.

One possible solution to the construction of symbols is given in [Foe93]: sensor inputs are transformed into simple symbols by neuronal networks. He explains that after the perception a neuronal network is used to reduce the incoming values to symbols.

The usage of symbols instead of “real world values” might gain increasing significance in a technical system the more information it has to merge and process. On the one hand, it would offer the possibility to deal with different information types in the same way. On the other hand, it could reduce the number of information which has to be processed once more, since behind *one* symbol there can be the context of it and the relation to other symbols as well.

However, the problem will be to find the right transformation into symbols and the right base of the symbols themselves in this application field. The usage of symbols requires a common understanding of them, the knowledge of what a symbol is standing for. Thus, the benefit of data reduction by symbols will entail the demand for a system with a general knowledge of the meaning behind every symbol.

2.3 Technical systems

A variety of research works deals with systems which have to act within a specific environment. For this task, many of them rely on the use of situation recognition. Within this field, a further subdivision can be made: into the works based upon conventional techniques, and works that are influenced by biological concepts.

2.3.1 Technical realizations using situation recognition

The achievement of motion skills is the thematic in [Ota96]. In this work, the attempt to realize motion skills in motion planning of multiple mobile robots is made. In the course of the work, a concept of situation recognition of sensor information patterns is outlined. With this recognition it is hoped to achieve motion skills by using hierarchical neural networks. The behavior of the system is described by rules. Finally, simulations are made to show the effectiveness of the proposed algorithms.

[Uen99] deals with a cooperation of cognitive learning and behavior learning. It describes a system which can learn a state representation and a behavior policy simultaneously while executing a task. Concerning cognitive learning, it extracts situations out of inputs. The system can extract highly abstracted states – situations, whereby a state is defined as a set of input vectors. Concerning behavior learning, it creates a model of the environment and performs planning on that model. In each step, the system identifies the current situation by using the current input, makes a plan on the model, and activates a behavior module according to the plan.

[Hai99] presents a system called ROGUE that forms the tasks of planning and learning for a real mobile robot, Xavier. ROGUE is able to learn from its execution experiences. One goal of this project is to have the robot move autonomously in an office building, reliably performing office tasks. Due to that it has to be responsive to changes in the environment. In this work, a system with the ability to adapt to changes and continuously improving the performance is presented. Particular patterns are used for the identification of the current situation. Due to this identification the system needs to learn the correlation between features of the environment and situations. These correlations are described by using situation-dependent rules. Furthermore, events in the environment are stored in an event matrix, which will be mapped into situation-dependent knowledge. Here, regression trees are used as learning mechanism for the mapping.

Another example of a system based on situation recognition is [Mor97]. The goal is a technical system which is able to support human activity. For this purpose it is required to recognize human actions conforming to situations. For an appropriate support without an explicit instruction by the user to the system, it is necessary to understand the task flow, and to decide corresponding support for the next action. It is pointed out that in former robotics the term “situation” is used as a static state in most cases. In [Mor97] the term “situation” is used for a dynamic series of events, for the course of events. It is emphasized that without using the flow of event, it is hard to determine the direction of the current event, and to estimate the future in order to be prepared for supporting the next task.

While these examples use rather conventional methods of situation recognition in means of a technical base and mathematical or statistical calculations, there are also attempts of technical realizations of systems which are acting in and with a continuously changing environment by using biological approaches. R. A. Brooks has described in [Bro01] four reasons for the non-functioning of today’s models in these systems.

- We might be getting a few parameters wrong. That would mean we have modeled everything correctly, but are just unlucky or ignorant in some minor way.
- We might be building models that are below some complexity threshold.
- It might be still a lack of computing power.
- We might be missing something fundamental and currently unimagined in our models of biology. (This point was his favorite.)

In [Bro97] he has made an analysis of behavior-based systems. The result of that work shows that there are many different models for this application field, but most of the time they are just computational experiments.

However, in the following, we will have a look at several examples of technical realizations influenced by biological ideas. Current systems concerning situation recognition are dealing rather in the field of robotics than in building automation. Nevertheless, they can convey an impression of the strengths and weaknesses of today’s technologies in general.

2.3.2 Biologically inspired technical realizations

The first realization to be presented here is also by Brooks. In [Bro87] he has built a technical system by using an incremental path from very simple systems to complex autonomous intelligent systems. On the basis of tests he comes to the conclusion that the use of an internal representation of the surroundings is not target-oriented in the creation of an intelligent system. He argues that we cannot be sure about it, and that humans are using their perception for an internal representation. It is more likely to him that we are missing some important aspects, and since we are too blind to recognize this absence, we are building on a wrong base. The human body is too complex, and we are therefore not able to understand it. Hence, we try to represent only our misconception.

Furthermore, he has defined several requirements for his realizations (he called them Creatures).

- A Creature must cope appropriately and in a timely fashion with changes in its dynamic environment.

- A Creature should be robust with respect to its environment; minor changes in the properties of the world should not lead to total collapse of the Creature's behavior; rather one should expect only a gradual change in capabilities of the Creature as the environment changes more and more.
- A Creature should be able to maintain multiple goals and, depending on the circumstances it finds itself in, change which particular goals it is actively pursuing; thus it can both adapt to surroundings and capitalize on fortuitous circumstances.
- A Creature should do something in the world; it should have some purpose in being.

The human thalamocortical system is the object of research in [Jov97]. The goal of this work is to build a distributed, parallel real-time multiprocessor system, which should be able to focus onto relevant aspects of the environment. In contrast to that, sensors and actuators which are not dealing with these important points should be hampered (super consciousness).

An excellent example of a realization was done by J. S. Albus. In [Alb96] he compares his real-time control system (RCS), which is for the design of intelligent control systems, with biological concepts. Within this work he has provided programming tools and software templates for a variety of platforms in order to build such a system. The RCS itself is composed of several modules like behavior generation, sensory perception, world modeling etc. Each node and module is implemented as an augmented finite state machine, which runs asynchronously as a cyclically executing process.

In [Alb99], this architecture is used for the Demo II Experimental Unmanned Vehicle program of [WWW4]. In this work, an unmanned vehicle has to plan and carry out tasks in a natural terrain by itself.

The system uses a hierarchy of computational nodes each of which contains processes for behavior generation, world modeling, sensory processing and value judgment, and additionally a knowledge database. The behavior generation is able to hypothesize tentative plans. Then the world modeling predicts the probable results, and the value judgment evaluates the results of each tentative plan. Finally, the behavior generation selects the tentative plan with the best evaluation (deliberative behavior). Additionally, a feedback from the knowledge database can generate a reactive behavior. There are services for compensating errors and differences between planned and observed situations in the world. To improve the performance, there are, among other things, multiple levels of representations to limit the amount of detail, multiple levels of sensor information to minimize the feedback time delay, and precomputed planes in response to the recognition of objects and events to limit the amount of search required to generate plans.

In many works the task of situation recognition is used in the field of robotics. In this application field one can find the usage of biological concepts quite frequently. However, situation recognition is only one problem these works have to deal with. Beside that, biology is used to support tasks like moving around, determining the position, overcoming obstacles, and so on.

In [Wit00] it is emphasized that the request to use "biological inspiration" for machines has to be founded on knowledge about the basic principles. The original bionic approach led to a theory called "biomimikry". While the lack of adapted

materials is still prominent, the step to an improved technology just by simply copying natural constructions will fail in most cases. Thus, similarities between different individuals were established, and 10 principles were extracted which have to be transferred into a machine in order to overcome any technical limitations.

The motion control in individuals has inspired many researchers. [Ilg00] presents an adaptive control for a four legged walking machine named BISAM. The idea of this adaptive control is to learn sensor-based reflexes of posture control. The analysis of the biological locomotion of a goat is used as a basis for the machine motion.

The idea of [Kim00] is a quadruped robot which is able to walk dynamically on irregular terrain by the support of a neural system model. Since animals show excellent abilities in autonomous adaptation, it is pointed out that the biologically inspired control proposed in this study has advanced abilities for adaptation to unknown irregular terrain. Different biological reflexes such as stretch reflex or vestibulospinal reflex were analyzed and integrated into the technical system.

In [Tak00] the development of a quadruped robot is the aim as well. However, this time the goal is the ability to jump over a hurdle like a horse.

In [Yam00], the motion of a jumping cat is studied since the assumption is made that it may be an efficient mechanism of the vertical movement of robots. Thus, the purpose is to analyze and construct a control law of a machine's mimicking a cat's motion.

[Bis98] deals with vision-guided autonomous indoor vehicles. Thus, the situations in the environment of the robots have to be recognized. It is explained that today's robots show deficiencies in a variety of factors, for example they are not able to make complicated decisions or to adapt quickly to changes in their surroundings. It is pointed out that organisms are able to adapt easily to changes of their own conditions and of the environment. In this work, situation recognition is regarded as the key of a perception-action loop of a behavior-based robot.

As we can see, there is already a variety of works dealing with situation recognition and the integration of biological concepts into technical systems. However, the field of home and building automation seems to have remained uninfluenced by this development so far. If we examine projects concerning building automation as for example [Inh00], we can see that the term "smart building" is used with completely different aspects. In this project many different companies work together to analyze which applications could be useful, desirable, functional and, above all, payable. Thus, components such as telephone, audio, video, PC, Internet etc. are integrated everywhere in a house. Based on this concept, the scientists realize remote control facilities, test new user-interfaces, prepare market analyses and so on.

In contrast to the above mentioned approach, a system is developed in this thesis that uses biological mechanisms in order to achieve an intelligent and preventative behavior. Reviewing the above, it is apparent that due to technical restrictions and the absence of sufficient evidential knowledge about biological individuals, some of the planned functions will not work with the desired accuracy and efficiency. However, it is evident as well that in the course of time an increasing number of these restrictions will be eliminated. Hence, despite of today's borders, I will pursue the development of an intelligent and preventative acting system by means of situation perception and situation recognition supported by biological concepts.

2.3.3 Conclusion of existing approaches

In Section 1.3 a summary is given of all requirements to a situation recognition system for a preventive and intelligent behavior in home and building automation. By now we have seen the features included in today's systems and the possibilities offered by them. In the following, we will analyze how these systems treat the aspects stated in Section 1.3.

INTEROPERABILITY

The aspect of interoperability is of little interest in the presented examples. Since they are confined to specific, enclosed applications (for example robots), there is no need to deal with the problematic concerning the combination of different technologies. Most of the time, there is a central system where all sensors and actuators are connected. By that, interoperability between the integrated devices becomes unnecessary.

Hence, these examples do not offer solutions to the need of interoperability in our system. Up to now, there are only solutions based on pure technical concepts, as for example [Pos01]. In biology, the different sensor types are brought to a general base, and allow therefore a global communication [Car99]. Beside this aspect, the usage of a symbolic processing might offer additional advantages, as shown in Section 2.2.4.

SENSORY SYSTEM

The extensive usage of sensors in biological systems is widely ignored by current technical systems so far. Although different types of sensors are used, the number of sensors is still low.

This situation is partly founded on the insufficient technology nowadays. Until now, today's technical sensory systems are far away from biological systems concerning aspects such as the possible number of sensors, and size, density or accuracy of them.

Another reason for this drawback can be found in the application field of these systems. As already mentioned, systems for situation-recognition are used almost exclusively in the area of robotics. In this application field it is sufficient to collect very specific information on the one hand. Mainly information concerning obstacles in their way is needed. Robots do not need to know if there is someone in the room or if the light is switched on. They do not need to measure temperature or humidity in a room. On the other hand, they are restricted by factors like energy consumption or weight. A mobile robot is forced to transport not only the hardware with sensors and actuators, but also its own power supply. Therefore, the whole structure, including logic and hardware, suffers from the restrictions of weight and energy. It is difficult to escape this vicious circle: using a bigger source of energy will also increase the weight and therefore the stress on the construction. Thus, stronger motors are needed which are again heavier and will need more energy. Due to that a bigger source of energy is needed ...

Since we are dealing with home and building automation in this work, it should be possible to overcome restrictions concerning energy consumption and weight. It will rather be necessary to install as many sensors as possible to perceive the environment with sufficient precision. Similar to the biological senses, one characteristic, object or

event will be realized by several sensors, sensors of the same and of different types, in this work. For example, [Roh94] describes that each sensor conveys only a part of the entire world of stimuli. If we want to achieve fairly complete descriptions of the surroundings, we have to use a combination of different sensor organs. Though none of the stated technical research works offers practicable solutions to this task, we can find a variety of biological works dealing with this aspect as shown in Section 2.2.

STRUCTURE

In this scope there are enormous differences between biology and technology. While nature banks on decentralization, technical systems use very often a central processing unit. Most of the time the sensors and actuators are directly connected to this central unit, and all calculations take place there. As long as they use just a small number of devices there will be no problem at all, although this means a lower availability since a malfunction in the central part will affect the entire functionality. The idea of this work bases on the use of fieldbusses. At least the aspects concerning the distribution of the system can be handled by that.

Studies of natural systems provide a variety of approaches to the physical structure as well as to the logical structure. The significance and usability of these concepts have to be investigated in the course of the work.

- Central structure

In the human body, though the information collection takes place in distributed sensors, almost all of the information flows together in central areas such as the thalamus or the cerebral cortex [Jov97]. As [Roh94] has explained, nature tends to concentrate the elements which are involved into the processing and intermingling of these data in central areas.

- Decentralized structure

We possess an enormous number of sensor cells, which are located all over the body. In principle, all of them have the same task: they receive specific stimuli and translate them into electrical signals. While doing so, they adapt the different types of sensory inputs to each other [Car99]. Thus, the first data processing takes place in the decentralized located sensory cells.

- Hierarchical structure

The majority of studies on natural systems has shown that the course of behavior can be described in a hierarchical structure, e.g. [Cor93]. Following J. H. Jackson, the founder of modern British neurology, [Kan00b] describes the information processing in the cerebral cortex with a hierarchical model. Between the different layers, there is a continuous information exchange [Hal92], and the layers can have reciprocal influence [Str00]. Nevertheless, there is a clear functional differentiation between the far-reaching independent levels [Sac98].

DATA STORAGE

There exist already technical solutions which use biological concepts for organizing the data. For example, [Kie99] describes the information storage within a structure of

semantic relations. Though we are still far away from the efficient handling of the stored information of a biological brain, there are already several attempts in this direction.

The right way of storing data will be an important part of this work, since there will be a large amount of information as a result of the high number of used sensors. Therefore, we have to address this problem more in detail than it is necessary in the field of robotics (the efficient handling of data storage is mentioned in none of the presented technical works).

However, by using the presented biological methods, we have to bear in mind the usability of these concepts. For example, the separation into two kinds of memory, into the declarative, explicit memory and the nondeclarative memory may be reasonable in biological systems. But it could be a restriction in technical systems. We have to search for a suitable way of storing data. Let us assume that we will take a database for the storage. Then, the relations between the tables define the connections between the stored information, the “knowing how”, automatically. Therefore, it will not be necessary or even possible in some cases to make a clear division.

FLEXIBILITY

Restrictions concerning flexibility are mainly found in the field of building automation as the result of already existing applications with rigid structures. The presented technical systems are not affected by this problem because they are complete, independent solutions. They do not have to base on already installed systems; they do not force us to consider the extension of different technologies. Furthermore, a sensor has a particular task in these systems. There is only one application where the sensor is working in. In contrast there are a variety of different applications in building automation, and hence it should be possible to use a device in as many applications as possible. Thus, we will have to make direct use of biological concepts as described in Section 2.2.3. They offer methods for merging information of different kinds, and therefore a flexible handling of different sensors.

However, concerning learning, i.e. the flexible changing of knowledge, most of the presented works are dealing with this task. Though we have only insufficient knowledge about the processes of learning in the biological brain, existing approaches offer promising methods in that field. For example, [Uen99] describes a system that is able to learn new situations as well as appropriate behavior.

BEHAVIOR

Since none of the presented systems is working in the same application field as this work, none of them has tried to achieve an intelligent and preventative behavior in our sense. However, it will be necessary to identify situations to achieve the desired behavior, and at least for that we can take existing approaches into consideration.

Many researchers have stated that a system has to be cognitive in order to behave adaptively to the environment.

Cognitive systems are in general biological organisms or technical systems like robots, or even groups or mixtures of such systems [Str00]. In this connection, cognitive processes are understood as information processing, as calculations, and are therefore fundamental in biological systems as well as in technical ones.

The variety of works in Section 2.2 leads to the conclusion that a combination of a cognitive and a non-cognitive system, that is, a combination of conscious perception and processes without any internal representation, will show the best conditions for an adaptive behavior. The single use of a non-cognitive system would not be target-oriented, since we cannot achieve an intelligent preventive behavior by that. The single use of a cognitive system would not be reasonable as well, since we could not realize reflex actions by that. Even if there occurred just a small change in the environment, the system would have to recognize the entire situation in order to be able to react. Hence, cognitive systems are not a replacement for non-cognitive ones, but they act in addition, they are based on the non-cognitive parts [Str00].

[Bro97] describes 7 key issues of technical cognitive systems:

- **Bodily form:** The form of our bodies is critical to the representation that we develop and use. If we are to build a robot with human-like intelligence then it must have a human-like body in order to be able to develop similar sorts of representations.
- **Motivation:** The system needs to have some sort of motivation, which may vary over time, and this motivation must be able to reach some sort of expression in what it is that the humanoid does.
- **Coherence:** A humanoid robot has many different subsystems and many different low level reflexes and behavioral patterns. There must be some sort of coherent behavior, which has to orchestrate all these parts without a centralized controller.
- **Self-adaptation:** The system must be continuously self-adapting and thus self-calibrating.
- **Development:** Cognitive development is a completely new challenge for robotics, behavior-based or otherwise. In humans (and, indeed, most animals) there is a parallel development between cognitive activities, sensory capabilities and motor abilities.
- **Historical contingencies:** Excessive incorporation of everything that exists in the human system may well be a waste of time if the particular aspect is merely a historical contingency of the evolutionary process and plays no longer any significant role.
- **Inspiration from the brain:** In building a human-level intelligence, a natural approach is to try to understand all the constraints that we can from what is known about the organization of the human brain.

PERFORMANCE

Works, as for example [Alb96], show several methods for improving the performance of processing by using concepts that can be found in nature. The processing in several steps is used as well as the focus onto important features of the environment.

However, as a consequence of the relatively small number of sensors used in these stated examples, the aspect of performance is not that critical there as in the field of building automation. In our application field, the number of integrated information sources will be much higher and, therefore, the efficient handling of the information will be more significant. Here, we have to consider the handling of a large amount of information collected by the sensors, as well as the demand for short reaction times.

Chapter 3

A model for situation-dependent behavior

*A theory should be as simple as possible,
but not simpler.*

Albert Einstein

This chapter is an investigation in the conclusion of already existing systems in order to create a new model for systems with situation-dependent behavior. This new approach should use the strengths of existing systems, and at the same time remove their weaknesses. By that it should be possible to achieve the goal: an intelligent and preventive reacting system in the field of home and building automation.

On the basis of studies of biological systems it seems to be reasonable to make a division similar to the one we have come across in some of the studies (see Section 2.2).

- There is a physical stimulus,
- a set of events transforms the stimulus into nerve impulses and,
- there is a response to this signal in the form of a perception or conscious experience of sensation.

We acquire this division and extend it by the possibility of a reaction with which the entire behavior in a certain situation is covered.

- The conscious experience leads to a deliberative behavior which consists of a set of reactions
- Each of the reactions is transformed into impulses
- The impulses change the state of the real world by using actuators.

If we consider these steps, we can reduce them to a more general level:

1. **Perception:** Physical stimuli are perceived and transformed into impulses
2. **Situation recognition:** There is a response to these signals in form of a conscious experience and in the recognition of the current situation
3. **Reaction choosing:** An appropriate behavior, which consists of a set of reactions, will be chosen
4. **Reaction:** Each of the reactions is transformed into impulses, which change the state of the real world by using actuators.

These four steps should represent the base partition of this chapter. Following this pathway, we have to analyze each step and find a suitable structure to satisfy the requirements.

3.1 Perception

Humans use an enormous number of sensory cells for the perception of the environment. Only by this large quantity of acquired impressions it is possible to obtain a picture of the world which surrounds the human. This is also a central idea of our system: the system has to possess as many sensors as possible to be able to create a detailed image of reality [Hal92]. Though it will have to manage a large amount of data, it still has to provide sufficient performance in short reaction times.

This process way has however another important advantage: redundancy. On the one hand an item or a characteristic of the environment is perceived by many homogeneous sensors (for example the temperature in a room is measured by several temperature sensors), on the other hand a value can be registered also by several completely different sensors (for example the presence of a person in a room can be detected by movement sensors, cameras, light barriers, rangefinders etc.). In both cases measuring errors can be avoided, and furthermore a check of the plausibility is possible, since one information item is always covered by several sensors. Only by receiving information from different, from independent sources, can we check whether the received data correspond logically or whether contradictions arise.

Additionally, by redundancy one achieves advantages like the reduction of the probability of sensor failures or the improvement of the reliability of the entire system, as described in detail in [Die98].

SCOPE OF THE PERCEPTION

This part of the system has to perceive all characteristics of the environment and the state of the system itself. It has to manage the collected information and present the data to the higher layers in a suitable form.

Step I: Sensors

There are different physical stimuli from the surroundings. These inputs have to be received by using appropriate sensors. It is beyond question that we have to use an enormous amount of sensors in order to detect every detail of the environment. In general, sensors can be used in two different ways: either the sensor is already included in a system containing a variety of sensors (as for example a fieldbus network), or it is a stand-alone device. Since we are

dealing with the field of home and building automation, a fieldbus will be taken as example in the case of the former. This analysis will be based on the system presented in Section 1.4.1. There are many devices supported and used by one system, and a sensor is not just a simple device but already contains a processing unit, memory, communication interfaces, etc. The latter means an external sensor which is connected directly to a PC. An example of this is a camera, which is supported by pattern recognition software running on the PC. This kind of sensor represents an independent, demarcated system. However, as long as it is practicable to obtain usable information of the sensor or the application, and to provide the data to the entire system, it is possible to use this sensor for this work.

Step II: Transformation

The perceived information has to be translated into a suitable form in order to store it. Although this step is described briefly, it will be a very important step in the task of perception.

Step III: Interface

Finally, the transformed data of Step II has to be presented to the higher layers. This step describes an interface between the real outside world and the logical PC-internal information processing. This interface is a representation of the states of the current environment.

The scope of this part allows us the rough division into a few steps as above. In the following, we will analyze each of the steps in detail.

3.1.1 Sensors

The functionality offered by a sensor depends on the division which has been carried out before. A fieldbus-device can be seen as a fully functioning computer. Most of the time, the functionality of such a device is changeable, and therefore adaptable to the requirements of the system. In contrast, a stand-alone sensor which is controlled by appropriate software has a defined functionality, which can hardly be changed afterwards. Since the application is given in such a system, either a completely new program has to be developed in order to control the device or – if possible – just a module has to be generated, which is based upon the former application and offers additional functions.

BASIC FUNCTIONS

There is a variety of simple functions which can be met by the sensors themselves (Table 1). In the following, an overview over different functions with regard to the integration in the used sensor devices is presented.

The majority of fieldbus sensors offers the possibility to calibrate the device and change the states of it. This can be done either in the sensor itself or via the network by using simple commands and changeable parameters. At first, calculations of transforming the physical stimuli into digital values or sometimes even simple statistic calculations take place (for example in [Phi00]). As already mentioned, some restrictions can arise by using a sensor with a prefabricated application.

Often, several sensors of the same type are used in buildings to measure a value, for example the Kluczynski buildings, where about 200 sensors are used for occupancy and more than 400 nodes for measuring the temperature [WWW5], or the Kokstad building, where almost 100 nodes are used for detecting intruders [WWW6]. In a fieldbus, usually a controller node is used for multisensor valuations. It receives the transformed values of the sensors, and is therefore able to perform further calculations, and identify and correct errors. For example, in a room several temperature sensors can be installed in different places. They send their values to a controller node, and this node will check the plausibility of each of the received values, and is therefore able to check the functionality of the sensors. Additionally, it can calculate the average temperature in the room. By passing on just this calculated value, we can achieve an enormous data reduction. A stand-alone sensor will rarely be designed to support this feature. This can be achieved only by developing an additional software module.

Since a fieldbus network supports a variety of sensors as well as a variety of actuators, it is possible to execute individual designed actions already on the lowest level. This point will be discussed in detail later in this chapter (reflex actions). Again, a stand-alone sensor has to be defined explicitly in order to support the control of actuators.

In several studies we have seen that a kind of focusing the attention on important objects or events, a kind of prioritizing is used in technical systems as well as in biological systems. Hence, we have to consider the use of priorities. There are several ways to include priorities: they can be static or dynamic, there can be a priority for each node or the priority can depend on a composition of particular nodes, a priority for an object, an event, etc. We will return to this point in Section 3.1.2. At this point we can only deal with priorities for the nodes themselves or groups of nodes. It would be easy to include a priority in a device in a fieldbus. Either the node already offers this feature or the communication protocol of the network itself or the software in the device has to be extended by that. The disadvantage of doing so is obvious. It will work perfectly well as long as we are using only few nodes. But in a building we have to deal with thousands of devices. It would not be maintainable if, for example, a new sensor is added and the priority of this node interferes with existing priorities of other nodes, since we would have to make changes in the entire system, node by node. An improvement concerning the maintenance can be achieved by moving the task of prioritizing to controller nodes, which serve several sensors. It would be possible to assign a priority to each of the connected sensors as well as to defined groups of them. However, this improvement depends on the used fieldbus system. It will be difficult to find a standard node for the task of handling priorities. Therefore, a new application for a controller in compliance with the standards of this fieldbus has to be defined. Supposing that it is not possible to find or define a suitable node, we can direct the handling of priorities to the next higher level, explained in Section 3.1.2.

As already mentioned in Section 1.3, the system has to possess a memory, i.e. storage for predefined and previously experienced knowledge. In some cases it could be useful to access this stored information already on the lower levels in order to use this knowledge for calculations, comparisons, etc. For example, if the system detects movement in a room, the stored knowledge that the owner has left the building some hours ago, would have a significant influence on further actions. Since fieldbus devices support only a restricted access to storage, if any at all, we are forced to add an additional tool or module for that. Moreover, fieldbus networks are not designed

for sending long requests or for receiving a large amount of data. The only benefit by using stored knowledge at the base level is the improvement of performance, because then it is no longer necessary to involve higher levels into the process. However, this benefit would vanish because of the moderate performance of the network, and therefore it is not reasonable to generate connections between the nodes and the storage. In the case of the stand-alone sensor, it will again depend on the application of the sensor if this program already supports the access to a memory. Table 1 shows a summary of this comparison.

Table 1: Basic functions provided by sensors

Fieldbus sensor	Functions	Stand-alone sensor
Integrated in device	<ul style="list-style-type: none"> - Calculations - Calibrate - Parameterize 	Depends on predefined functions
Integrated in device	<ul style="list-style-type: none"> - Multisensor evaluation <ul style="list-style-type: none"> - Error identification - Error correction - Calculations (average...) - Data reduction 	Module is needed
Integrated in device	<ul style="list-style-type: none"> - Signals to actuators 	Module is needed
Either in <ul style="list-style-type: none"> - Sensor itself - Controller node - Higher system level 	<ul style="list-style-type: none"> - Priority 	Module is needed
Module is needed	<ul style="list-style-type: none"> - Access to storage 	Module is needed

In Table 1, the column *Functions* contains all stated functions. The possible functions integrated in a fieldbus device (column *Fieldbus sensor*) are compared with the functions in stand-alone sensor (column *Stand-alone sensor*).

Though fieldbusses offer a higher flexibility and easier handling, we still have to employ additional, external sensors. There are many application fields which are not or hardly covered by fieldbusses, application fields which are, however, necessary for situation recognition like pattern or sound recognition.

REFLEX ACTIONS

Even though this section deals only with the perception of the environment, there occur already decisions for first reactions in this scope. These reactions will be called “reflex actions” in the further work by analogy with [Kan00g]. Due to that, we first have to define the scope of a reflex action.

Definition 1. A *reflex action* is an unintentional, involuntary, automatic reaction of the system to an internal or external stimulus.

There are several aspects arguing for the use of reflex actions.

- The first reason can be found in nature. Since the desired system of this work should profit by the use of ideas of biological systems, and the functionality of these reactions has been described in several studies, reflexes should be included in the present system.
- The big advantage of this inclusion is the high performance, the short delay between a stimulus and the reflex action. Especially this aspect is very important for the behavior in dangerous situations. There, the system has to react as fast as possible.
- The reason for this short delay is easy to explain: it is not necessary to identify the entire situation in order to perform such simple reactions. Thus, there is just a short information flow. The data do not have to reach higher levels, and no complex calculations will take place in order to produce a reaction. Instead, reflexes have to be managed exclusively by the lowest layers of the system.

And here we can find the drawback of implementing reactions at this level of the system. Reactions are based on available information. Unfortunately, most of the time there will be only scanty knowledge of the environment and the entire state of the surroundings. A reaction without sufficient knowledge could lead to an even more dangerous situation. An answer to this dilemma is provided by nature itself. Even human beings or animals can get into dangerous situations through reflex actions, for example if there is a sudden noise or a sudden movement and an individual jumps out of the way without realizing if it was really necessary, and without checking the new situation whether this unintentional movement can lead to a dangerous situation. Nevertheless, the evolution seems to consider reflex actions to be advantageous – and nature has had a long time to check the usefulness of them. One can say that in general it is acceptable if a reflex leads to a problematic situation **sometimes**, if in exchange one is **always** able to react immediately to a dangerous situation.

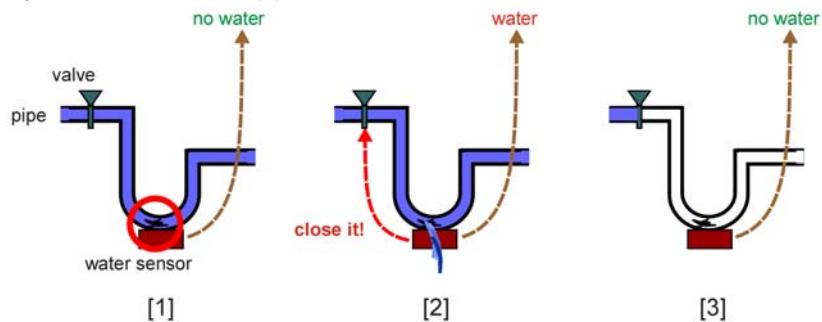
Preposition 1. *The only intention of a reflex action is to react to a dangerous situation.*

However, there is a technical answer to this problem as well. We have to keep the reflex actions as simple and safe as possible. For example, there will not be any problems by activating alarm or sending a message. Now we have to consider reactions where we have to anticipate the consequences. According to the definition, reactions like *switching on the light if someone enters the room* or *closing the fridge if it is open for too long* do not belong to reflex actions. Sometimes, side effects are acceptable if it is necessary to react immediately. But in higher functions, for example in the improvement of comfort, there must not be any fallout. There, the system has to be sure of the consequences; in exchange the system has enough time to calculate and check everything. Hence, we have to focus on reflex actions in dangerous situations, and the avoidance of possible resulting situations. We will use again the example of the sudden movement or noise, and the following jump out of the way. An improvement to this situation would be a second reflex action, initiated by the resulting dangerous situation, which stops the jump on time. Here, the technology can offer an important advantage: reflex actions are involuntary in humans, but not in a technical system. Thus, the entire sensory system is still active during a reflex action.

The only requirement: one has to define a large amount of possible reactions. Only then can we achieve that the reflex actions supervise each other.

In the simplest form, a reaction consists of a sensor and an actuator. The sensor is directly connected to the actuator device, and initiates an action there. In case it is necessary to convert the signals or to make an additional comparison or calculation, a controller node can be included in this task. However, it will be necessary to connect several devices. It is well possible that these devices come from different systems or even from different industry areas. Or it could be required to combine a non-fieldbus sensor with a fieldbus device or vice versa. It would not be reasonable to develop a specific module for every reflex action, which is using incompatible devices, especially with regard to the possibility of extension by new further devices. This topic will be discussed in detail later in this section.

A reflex means an action without thinking. Now the question arises what sort of information about the reaction should be passed on to the higher layers or whether these higher levels should receive information about reflexes at all. In the human body, at least some of the perceived information will reach the cerebral cortex, however we are aware of this information mostly only later on. But what does a reflex action mean to a technical equivalent? We will take a simple example to find an answer (Figure 15): a water pipe (1) has a leak; a sensor detects water (2), and closes the pipe by means of a valve (3).



**Figure 15: Sequence of a technical reflex action;
[1] leak in pipe, [2] water detection, [3] closed valve**

Since we have to consider that higher functions will need some time to handle the information provided by the sensor level, this is an example where the higher layers will probably never know this problem. The reflex action will start immediately and the message about the water leak will last only for a very short time. After that, the situation presents itself as usual – with the difference that there is still a leak in the pipe.

There are two ways to manage this situation: either the system finds out about the reflex action by itself, or it has to be directly informed.

- We can use a kind of memory, where we store changes carried out by reflex actions. Thus, the system will be able to “remember” these changes. However, it will be necessary to constantly check the stored reminiscences. Moreover, the question arises how many of these past events should be stored and, above all, where? The inclusion of memory into devices will be neither practicable (since we want to use existing approaches, it would mean to

change the origin applications) nor manageable (if there is more than one water sensor in the example above, we would have to check the memory of each of them or we would have to determine one of them as the one with the storage). Moreover, the usage of an external storage accessible by the devices has already been excluded earlier in this section. Hence, the remaining way is to hand the information up to a higher layer where the system is able to access storage.

- The second way is to pass the information about the reflex action up to the higher layers. Instead of the stored past changes, the higher levels would receive the information that there was a specific reflex action. In the former example, this would mean that the system would “see” the reflex action for closing the pipe. Somewhere it has to store the information about which devices are involved in that action.

The fact is that it is not sufficient to know merely the resulting state. The system has to know that there has happened something. If we compare the necessary resources of the two ways, we can see that both need additional memory: the former for storing past events, the latter to store connections. In case of performance, the first needs more time to find the reflex action. In contrast, the second one will know immediately about the reflex but has to search for the devices concerned. Let us compare this to nature: if you are burning your hand, you will immediately know that the hand is affected, but not necessarily the reason. Thus, a further investigation is required to realize the details of the situation. [Car99] and [Kan00a] describe examples of reactions without any conscious registration. In case of the knee jerk (Figure 16), if you neither saw the stretch reflex nor received information about the position of the leg by the sensor cells in the muscles, you would not know anything about it.

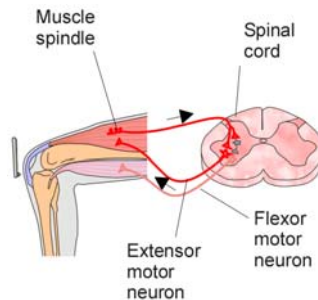


Figure 16: Knee jerk

Therefore, I come to following conclusions (illustrated in Figure 17):

1. The affected devices have to be known.
2. The higher layers have no explicit information about the reflex itself.
3. Sensory information has to be used to identify reflex actions.
4. Hence, there must be storage about the devices used by a reflex to allow the system to search for the reflex action.
5. The system has to know that a particular change has been caused by a reflex.

Figure 17 shows the exemplary handling of reflex actions. At the bottom, 12 devices are shown, named with A to L. The connections between some of the devices represent communication; for example, A and B send messages to C, and F is

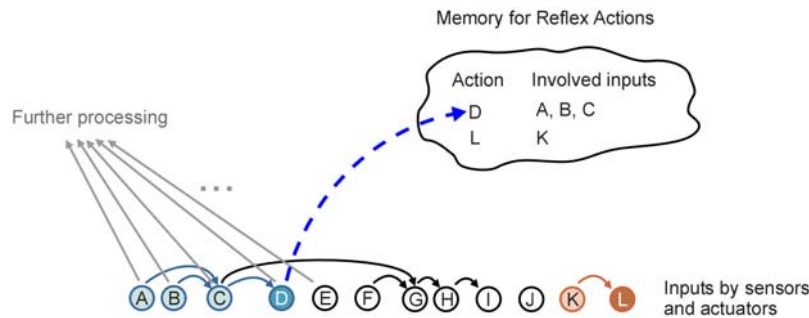


Figure 17: Handling of Reflex Actions

connected with G. Some of the devices are used for reflex actions: A to D is used for a reflex (where D is the actuator), and K and L are used for a second reflex (with L as actuator). The memory concerning these reflex actions contains the actuators D and L including the inputs involved with these actions. Beside that, collected information is passed on to higher layers for further processing. If a reflex action occurs (D), the memory is updated and the system is informed by that about the action.

TRANS-SECTORAL PROCESSING

One goal that has to be reached in this work is the integration of current technologies into the aimed system. Therefore, today's applications and devices have to be considered, adapted and merged in order to meet new requirements. By doing so, we are confronted with problems which result from different dependences.

Application:

Many manufacturers offer complete systems for specific tasks. Devices within such a system are defined in a way that they work together to manage their tasks. In some cases, they use a proprietary communication [Rus01]; most of the time, they are not prepared to cooperate with other applications. By using systems based upon fieldbusses, at least some of the problems can be solved, since there are given definitions of devices and of the communication itself.

Industry:

By using devices or systems out of different industry areas we have to face another problem: since there are different requirements in the different industries, the manufacturers have certain ideas of defining devices; they focus on diverse aspects of their systems.

Let us assume that we want to use one alert-device to inform us about too high temperatures, and we want to connect it with the cooling system of a fridge and the heating system of a room. The manufacturers interpret the values in different ways, they use different precisions, and they have their own range of values – nevertheless, there is a general understanding of high temperatures in both areas. This example can be extended by a temperature

controller. Again, the task of this controller is the same in both cases – it can regulate the temperature within a certain range. However, it cannot be used for both applications resulting from the differences stated before.

Technology:

Most of the devices used in this work are based on sensors and actuators in fieldbus networks. However, non-fieldbus systems like speech or pattern recognition tools are to be used in the overall system of this work as well. Hence, we have to consider the integration of such applications.

In the following, I will use the term *trans-sectoral* for systems which combine diverse applications, industries or technologies. One can see that for the most part the handling of trans-sectoral systems will depend on whether we are using fieldbus or non-fieldbus devices.

In the simplest case, the cooperation of different devices already works. Most of the time, these devices will come from one application or one industry area. Thus, we will name that cooperation “intra-work” (Figure 18). I have chosen that term following the definition of intra-industry in [Kab02]. However, intra-industry deals with the problematic within one industry field. Since this work is not based on fieldbusses alone but also on pure software applications (for example speech recognition) which are not classifiable in particular industry areas, a new naming has been required. In the following, this term will be used for fieldbus systems as well as non-fieldbus applications, and systems consisting of only one device.

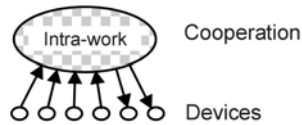


Figure 18: Intra-work

In the next step, we have to merge different intra-working systems. This applies to systems based on fieldbusses as well as to non-fieldbus systems or combinations of both. In general, there are two possibilities to solve this problem: we can either create the connections directly between the intra-working systems (inter-working systems) or we can convey the information of these systems into a global¹ “connection module” in order to merge them (Figure 19).

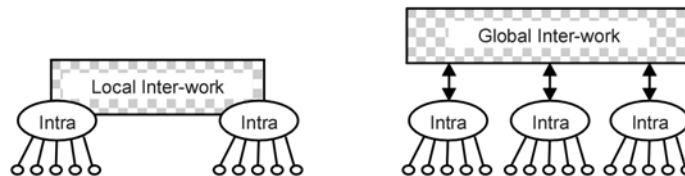


Figure 19: Local and global inter-work

The former uses specialized modules between the intra-working systems. Within fieldbus systems, there are, as mentioned above, already attempts like standardized

¹ Global in the sense of a system wide validity

communication formats or hierarchical structures to define the devices. Furthermore, for example in LonWorks, it is possible to create controller nodes for connecting different applications. In case of different fieldbus systems, there exists a variety of solutions on the market for connecting them. Therefore, this way of connecting is built in several layers (Figure 20). Still, it can be seen as a local inter-work since it connects directly to different systems – if you want to connect another system, you either have to integrate another connection module or you have to employ the latter method.

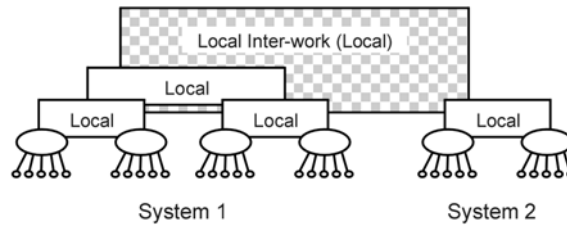


Figure 20: Several layers of inter-work

The latter method means that there is a global module to which all systems are connected to. By that, we have to transform all different systems into a common base. This will be discussed in detail later in this section.

Both solutions have their strengths and weaknesses. Table 2 compares both concepts. Depending on where we establish our priorities, the one or the other method is better. Comparing these aspects with the requirements of Chapter 1, we can see that this part of the system is decisive for three of the main requirements: structure, performance, and flexibility. Though proprietary or standardized solutions and maintenance are not imperative for the functionality of the entire system, they are still important for this part. Especially fault tolerance is a critical aspect of the entire application. And here, the local inter-work has the advantage that most of the parts are still able to work together, even if the highest connection part fails.

Therefore, the use of local inter-work has to be preferred. Since there are already many solutions on the market, they should be used to construct the connections in this layer. Otherwise, if no solution is available or advisable, the connection can be made in the global inter-working part (Figure 21). Details of this part will be discussed later in this section.

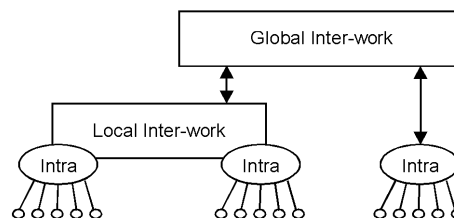
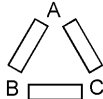


Figure 21: Global inter-work

The advantage resulting from a combination of these both options is that the weaknesses of each of them can be reduced. By using intra-work, many systems are

already connected; therefore, it will be easier to find a common base for the global inter-work. Additionally, the usage of the global inter-work removes the restrictions concerning the extensibility of the local inter-work.

Table 2: Comparison of Local and Global inter-working

Local inter-working	Global inter-working
<p>+ Performance:</p> <p>It is specialized in specific systems and all connection modules are working in parallel</p>	<p>– Performance:</p> <p>The conversion is not specialized in a specific system. Instead, it has to serve a variety of different systems, and therefore the conversion will be more costly.</p>
<p>+ Reliability:</p> <p>If one inter-working system fails, it will not affect the entire system. All other parts will still work.</p>	<p>– Reliability:</p> <p>To put all conversions into a single module means that if this part fails, the whole system is affected.</p>
<p>– Extension:</p> <p>In the worst case, if a connection between every pair of systems is necessary, for n different systems, we would need $\sum_{i=1}^{n-1} i = \frac{n}{2} * (n-1)$ inter-working modules. If we have the systems A, B and C, we would need 3 connections.</p>  <p>In this case, 2 connections would be sufficient, for example AB and AC. But this would mean that we have to make a detour over A for a communication between B and C, which would impair the performance.</p>	<p>+ Extension:</p> <p>Since all systems are brought to the same base, an extension can be made in one step. A new system needs to be adapted only once, and it will be able to communicate with all other applications.</p>
<p>– Proprietary solutions:</p> <p>This way makes use of many different modules. It will be difficult to find a standard, since they depend on the applications of systems which have to be connected.</p>	<p>+ Standardized solutions:</p> <p>The entire system uses the same conversion for combining different applications or systems.</p>
<p>– Maintenance:</p> <p>The result is a system which is difficult to maintain because of the distributed, different modules.</p>	<p>+ Maintenance:</p> <p>The maintenance is much easier than in the case of local inter-working, since the conversion of the different data types is located in one module.</p>

3.1.2 Transformation

Once the information about the environment is perceived, all data has to be stored in a suitable way. Here, the question may arise whether it is really necessary to store the values, or whether it is sufficient just to process them. This stage is the highest level of the trans-sectoral task. On the one hand, it has to complete the combination of the different systems; on the other hand, it has to provide information to the next higher layer. This information represents the states of the surroundings, and some of them may be valid for a long period. For the higher layers, it must always be possible to have access to the current states of the surroundings, and therefore we have to store them.

The transformation itself is also a necessary task. S. A. Starks describes in his work [Sta97] the necessity of symbols to achieve a reasonable performance in a system, which has to recognize or to identify something. He insists that we humans use information in semiotic form, for example by words and symbols, to process real-life data. And exactly this is an important goal of this work: real-life situations have to be identified. The use of symbols instead of discrete real-world values offers the possibility to include more information in one item.

Moreover, we will recommence the task of prioritization mentioned in Section 3.1.1.

Thus, we have 4 tasks in this stage:

- Standardization
- Completion of the trans-sectoral processing
- Storage of data
- Priorities

STANDARDIZATION

For this system, we will not only carry out a simple transformation of values of one system into values of another system, for example adapt the value for temperature of one system to the value type of the other one. In this work biological systems, and first of all the human being, are the examples, and humans do not work with discrete values. If one has to interpret a temperature, one will engage terms like *warm* or *cold* for it, depending on the context of the measurement. For example, if an individual enters a room, s/he will not sense the prevailing temperature as for example 23°, but as warm. Accordingly, one will not sense the brightness as 600 lux but as bright. Thus, this standardization has to be in a logical sense concerning the meaning and the context of the data: we have to construct symbols. A detailed description of the transformation task of the SmaKi project can be found in [Fal03].

By the use of symbols we can achieve an increase of the informational content. The symbol warm in the above example contains not only 23°, but a range of the temperature including the connection to a specific person and a specific location. This “increase in value” is a decisive factor, since by that it is possible to reduce many single independent values to few symbols. By that, the values can be processed more efficient.

In general, there is a variety of reasons pleading for a standardization of information:

- In order to merge information of different sensors and systems, data have to be adapted in an appropriate way. As already explained, this combination of information is important for an internal global representation of the environment; moreover, it is also an important step towards interoperability.
- An important factor is the reduction of information by using a certain range of input values for one symbol. By that it is possible to inform the higher layers just about important changes in the environment. For example, a brightness sensor will inform the higher levels of every slight change in brightness, and will cause a lot of unnecessary processing. By using a range of values for one symbol less information will be handed to the higher levels.
- Another reduction can be achieved by the filtering of noise. The measurements will fluctuate within a small range even under constant conditions. Since a range of values is used for one symbol, this fluctuation will have no effect. Thus, the handling of noise can already be implemented in the smart devices.
- A consequence of the information reduction in the higher layers is the smaller storage needed for the memory. In Section 3.2.2 the necessity of storing the history of changes in the environment is explained. Hence, the size of this memory concerning the history can be reduced.
- The data type used for the symbols is another factor of the improved storage management. The storage of different real-world values is no longer necessary. Instead, the type for the symbols is chosen that offers the best performance with the selected storage.
- Since we do not have to care about different data types, the handling of the values can be simplified. Values of different devices do not require a conversion.
- Besides this simplified handling, the usage of a uniform symbolic communication enables the cooperation of most diverse devices. For example, the information of a camera, which detects a person in a room, can be compared with the value of an occupancy sensor.
- In Section 6.2 a user interface for the aimed system is mentioned. For this task, the usage of symbols can be helpful in a simplified human-readable representation. For example, in the prototype described in Chapter 4 understandable names are used for the symbols. In this application, a brightness sensor would report *bright* instead of 716 lux.

Thus, we have to define a symbol stock of the common information base in order to achieve an unrestricted data exchange. This symbol stock has to ensure the correct transformation of information (syntax), and it has to represent the correct meaning of data (semantic).

Though there are several biological research works supporting the idea of the symbolic use, as for example [Kie99] or [Kan00b], it is difficult to attain a description of the symbolic structure. At least [Foe93] explains one biological solution to this task: sensor inputs by the optical sense are transformed into simple symbols by neuronal networks. Other works, such as [Car99], point out that the different types of

sensory inputs somehow adapt, but the way to this transformation into a uniform information item is left open.

In the course of this work, three possible ways of determining symbols are described and compared:

- Neuronal networks
- Fuzzy logic
- Rule bases

Neuronal networks

Neuronal networks are an artificial adaptation of the human brain. Their development has begun approximately 50 years ago by researchers as McCulloch and Pitts [Fau94]. The cooperation of several neurons creates a neuronal network analogue to the biological example. According to [Kin94], by using appropriate weightings functions they can be used as model for behavior, control systems, prognosis etc.

The selection of the activation function, which describes the behavior between inputs and outputs, plays an important role in an artificial neuronal network. According to [Fau94] and [Kra90], 3 kinds of functions are mainly used:

- Identity function

$$f(x) = x$$

In case of a linear activation the output is proportional to the sum of the inputs.

- Binary step function¹ (with threshold 0)

$$f(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

The characteristic of this function is the usage of a threshold.

- Sigmoid function

$$f(x) = \frac{1}{1 + e^{-\sigma x}} \quad \text{where } \sigma \text{ indicates the steepness parameter.}$$

This function is nonlinear and is used in most cases.

Concerning the weightings, [Kin94] points out that the network has to learn them according to the application field. By that, the neuronal network is able to adapt to an unknown environment. In works like [Kin94], [Fau94] or [Mit97] a variety of learning mechanisms is stated. Artificial neuronal networks, which employ such learning concepts, are useful for modeling tasks, classifications, data assignments, as well as for signal processing [Pet00]. The big advantage of using a neuronal network for the transformation task is the autonomous generation of symbols: the system itself can find a proper representation for given inputs.

However, this advantage entails some disadvantages too. Most of the time, a neuronal network works in one direction – an input vector produces an output vector. That is sufficient for the creation of symbols. However, the system will have to pass the symbolic reactions back to the real world (see Section 3.4). Thus, it will be necessary to use the transformation in both directions. [Kin94] describes such *bidirectional*

¹ The binary step function is also known as the threshold function or Heaviside function

*networks*¹, and explains that they offer poor results compared to *feedforward nets* with only one direction.

Another disadvantage is the knowledge representation in neuronal networks: the entire knowledge is stored in the net itself. Thus it is hardly able to combine its processing with an external storage. Consequently, it is not possible to use information of the processing in the net for other tasks or, conversely, to influence the processing. Another aspect that argues against this combination is the time factor: due to the large number of neurons the access to an external storage would result in an unacceptable delay for the transformation task.

Moreover, the mentioned learning concepts of neuronal networks entail numerous learning cycles in order to adapt to the environment [Pet00]. Each time a device, which uses a new communication with new values, is added, the system has to learn these values to integrate them into the overall communication. To surmount this problem, one can use a network that is already trained – however, by that one loses the ability of adapting to the environment.

Fuzzy logic

Fuzzy logic is another way to simulate human thinking. In the late sixties first theories of this kind of logic were developed [Zad65]. It can be used to obtain an exact mathematical notion for inaccurate, qualitative symbols.

[Sch98a] explains that in case of the classic, accurate logic, all elements of a set (crispy set) can be listed or described by set theory. For example, the set of *warm* can be presented as $warm = \{22, 23, 24, 25, 26\}$. By using a membership function $\mu(x): x \rightarrow \{0, 1\}$ we attain the following description:

$$warm = \begin{cases} 1 & \text{if } 22 \leq x \leq 26 \\ 0 & \text{otherwise} \end{cases}$$

Fuzzy logic allows values between 0 and 1 as well. For example:

$$warm = \begin{cases} 0 & \text{if } x < 21 \\ \frac{x-21}{2} & \text{if } 21 \leq x < 23 \\ 1 & \text{if } 23 \leq x < 25 \\ \frac{-x+27}{2} & \text{if } 25 \leq x < 27 \\ 0 & \text{otherwise} \end{cases}$$

By that, a temperature of 22° is still warm but not at the same level as 24° – which produces a description comparable to the human perception. The representation of a state value is named fuzzy set [Sch98a]. Figure 22 shows a possible fuzzy set of the input values of a temperature sensor.

¹ Bidirectional or recurrent neural networks

For the aimed system the usage of fuzzy logic offers an easy way to attain symbols according to human sensation. However, the user has to have basic skills in this field. First of all, s/he has to determine the membership functions of the different inputs. Moreover, mathematical functions such as negation or logical interconnections can be applied to the resulting symbols.

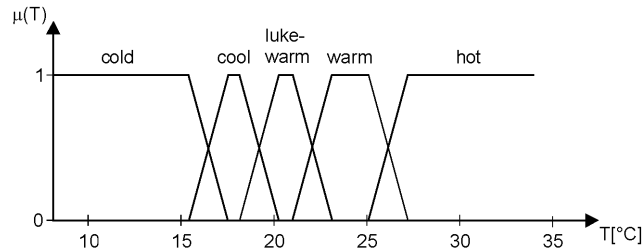


Figure 22: Fuzzy set of temperature

Rule base

A rule base means a more or less static assignment of ranges of values to a specific symbol. In contrast to fuzzy logic, these ranges are precisely separated. Therefore, the proper selection of these ranges and a reasonable number of symbols for an input will be decisive criteria for a suitable behavior of the system. [Daw03] summarizes recent developments in the field of data-analysis technique referred to as symbolization or symbolic time-series analysis. In this work several researchers are quoted who attempt the determination of the right number of symbols. The unanimous opinion is that when enough symbols and appropriate partitions are used, the entropy¹ is maximized, and the partition choice is “optimal”.

The usage of a rule base for the construction of symbols offers a readable and extendable method. It can easily be adapted by a user via an appropriate interface. However, the exact separation of the ranges might result in an incorrect behavior if the measured values are within this dividing line.

COMPLETION OF THE TRANS-SECTORAL PROCESSING

In biological systems, the adaptation of different subsystems is already conducted in the senses [Kan00c]. However, we want to use already existing technologies as base, thus we cannot carry out all the adaptation there. Technologies such as BACnet [Ten00] are developed to create a standardized method of interconnecting systems from different manufacturers. Since this interconnection is accomplished on a low level, it offers a high performance. Nevertheless, due to the demand of the integration of most diverse devices into the aimed system, which are not covered by these existing technologies, it is necessary to offer an additional interconnection at a higher level. Section 5.5 shows that this higher interconnection needs more processing and, therefore more time than the connection on lower levels. Thus, the low-level connections have to be preferred for the completion of the trans-sectoral processes.

Though the delays by the additional processing have to be considered, the advantage of the high-level connecting is obvious: it enables the definition of functions that are

¹ Entropy means the average information content of the elements of a certain set of elements (information theory)

not possible by today's systems. Devices which originally were not designed for this cooperation can be integrated into one application. For example, if the information of a speech recognition tool should influence a fieldbus application, we are forced to involve a higher level of interconnection.

It is evident that this completion of processes has to take place after the task of standardization. All devices have to use the same "language" to communicate. Otherwise, it would mean to design a communication module for every two devices, comparable to the local inter-work of Section 3.1.1. Whereas in that section the local inter-work is preferable, we have to favor the opposite in this case. In the hardware level the factors of performance and reliability are decisive. However, at the current stage we have already accepted a delay due to more processing. Moreover, the reliability of the interconnection is ignorable, since it is already based on an interface layer to the underlying hardware. Thus, if this connection to the hardware fails, it does no longer matter if the high-level interconnection is separated into several modules.

Therefore, the advantages of a standardized communication remain prominent. Values of all devices are translated into symbols. Descriptions of functions are constructed of symbols as well. Results of these functions, which are symbols too, have to be retranslated into values understandable by the devices. By means of that, there are no restrictions concerning the integration of new devices. Once a device is adapted to the symbolic communication, it can be immediately integrated into functions.

STORAGE OF DATA

The storage of perceived information is comparable to the short-term memory mentioned in Section 2.2.3, and will be necessary for the recognition of the current situation. Via the sensors, the system will get a large amount of information. However, it will neither be possible to process all of it at once, nor required to do so, since not every detail will be important in a particular situation. Thus, the extraction of the appropriate inputs and the recognition will take place in several steps – which requires the storage of the input values for a certain time.

Obviously, the storage has to be based on symbolized values. Data of different devices can be of most diverse types. This would mean that all these types have to be considered for the storage. In contrast, by storing only the transformed values one has to deal with just one data type. Moreover, one can take the type that offers the most benefits for the symbols in the chosen storage method.

PRIORITIES

In the Basic Functions in Section 3.1.1 we have decided to move the task of prioritizing the objects of the environment to a higher level, to a central control.

H. Heinze describes in [Hei98] that humans are able to concentrate their mental resources on selected events. He also raises the question how to find these selected events. How do we notice important aspects in the environment? Since this question still occupies current researchers, its answer remains unclear. However, even if the biological process in this question is unsolved, we can make some basic assumption for a simple technical equivalent.

In order to define the importance of particular parts of the surroundings, we have to make a differentiation first:

- Any information may have a priority.
- The same information together with another information part may have a completely different priority.

At this stage, we have no knowledge about the relations between the perceived values. Therefore, the latter aspect cannot be handled yet. Nevertheless, we can define a priority of some of the values already at this stage. As mentioned above, we have to make some basic assumptions concerning the importance of aspects. Thus, I have determined the following factors which will be treated with a higher priority – the order represents the importance, beginning with the highest priority:

- **Priority 6:** Particular information about dangerous factors, for example gas or fire;
- **Priority 5:** Persons;
- **Priority 4:** Information about an abnormal state of a device, for example a high temperature;
- **Priority 3:** Information which are marked as having been generated by a reflex action;
- **Priority 2:** Devices which are switched on;
- **Priority 1:** Changes in the environment, for example temperature changes or the opening of a window.

In this section, I have already described the usage of symbols. We can define a rule set for the priorities of some values, which assigns a specific priority to a specific symbol. By that, we are not forced to check the origin of the values. Instead, we deal only with the symbolic representation of them. For example: we have the symbol *too_high*, which indicates that the temperature is too high. It does not matter whether it is used for the room temperature or for the temperature in the fridge – though the value of the temperature itself will be different in both cases, the symbol to indicate this event will be the same: *too_high*. We can therefore say that this value – or better, this symbol – is important in the current situation. Thus, it is sufficient to assign priorities to the symbols.

However, we will handle the priorities for combinations in Section 3.2.2. At this stage enough knowledge about the relations will be available.

3.1.3 Perception Layer Interface

The communication between the task of *Situation Recognition* and *Perception*, as well as between the task of *Reaction* and *Perception* has to be defined in an interface, in the *Perception Layer Interface* (Figure 23).

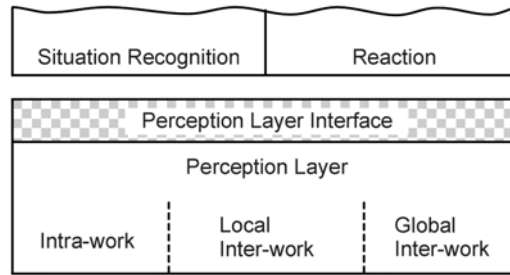


Figure 23: Perception Layer Interface

Thus, it is possible to construct different systems for the *Situation Recognition* as long as they are based on the interface provided by the *Perception Layer*. Additionally, it is possible to use other ways to perceive the environment as long as the result of that perception is in conformity with the definition of the *Perception Layer Interface*. The same applies to the task of *Reaction*.

SERVICES PROVIDED BY THE PERCEPTION LAYER INTERFACE

The *Perception Layer* represents a server task for the *Situation Recognition* and for the *Reaction*. Thus, it is the passive part of that connection. Both, *Situation Recognition* and *Reaction* have to register at the *Perception Layer* for getting information and changing states of the surroundings.

The following services have to be defined for the communication:

- **register_recognition()**
The task of *Situation Recognition* starts with a registration at the *Perception Layer*. The server task will answer either with a `handle()` or an `acknowledge()`.
- **register_reaction()**
The task of *Reaction* starts with a registration at the *Perception Layer*. The server task will answer either with a `handle()` or an `acknowledge()`.
- **deregister()**
Situation Recognition and *Reaction* end with a deregistration at the *Perception Layer*.
- **get_data(command_in)**
The task of *Situation Recognition* asks with `get_data()` for new values. Since it has to create an internal representation of the current environment, it is sufficient to attain only the changes of the surroundings instead of all perceived information. Thereby, the *Situation Recognition* adds a NEXT or a REPEAT as argument. NEXT means that the last value has been successfully perceived, and that it waits for the next change. REPEAT indicates that there is a problem with obtaining information and that the *Perception Layer* has to send the last change once more.

- **set_data(msg_in)**
This function is used by the *Reaction* task for changing a value of an actuator. The arguments are the required information for this change.
- **handle(handle_out)**
If the chosen communication technique requires for example a shared memory provided by the server, this can be achieved by sending handle() to the client as an answer to a registration.
- **send_data(msg_out)**
The *Perception Layer* detects all changes in the environment and offers them by using send_data() to the *Situation Recognition* task. Depending on the request by that task, the *Perception Layer* sends either a new change in case of NEXT or the last one in case of REPEAT.
- **acknowledge()**
The *Perception Layer* sends an acknowledge message to the task to notify the receipt of sent data, and to announce that all resources are ready for further processing.
- **repeat()**
If there is a problem with reading information from either the *Situation Recognition* or the *Reaction* task, the *Perception Layer* sends a repeat(). By that, the concerned client has to send its data once again.

3.2 Situation recognition

As already shown in Section 2.2, many researchers emphasize that for the recognition of situations, for a situation-dependent behavior, an internal representation of the surroundings is necessary. In [Str00], the representation is the base for control actions in a flexible and environmental-adapted manner, and [Flo97] puts a kind of representation, a monitoring of states, in relation with events in the environment to achieve consciousness. By that, we get a feedback from the surroundings. The perceived information is used for controlling the system in order to embark particular reactions. Next, the system will detect the changes caused by these reactions and again, this will affect the behavior of the system.

Studies in the field of psychophysics have focused on the relation of the physical characteristics of stimuli and the sensory perception. They have shown that there are qualitative differences between our perception and the physical properties of the stimuli, since our nervous system extracts only particular parts of the information of a stimulus and ignores the rest. Next, these extracted information are interpreted, depending on the internal structure of the brain [Car99]. We receive electromagnetic waves with different frequencies but we perceive different colors. We receive blasts of vibrating objects with different frequencies but we hear sounds, words and music. We receive chemical components in water and the air, but we perceive them as taste and smell. [Roh94] has pointed in the same direction: our perception does not reflect the surrounding world directly. We rather construct our perception internally, depending on the architecture of our nervous system and its capabilities. Therefore,

the internal representation does not show an immediate, direct picture of the surroundings but only selective stimuli.

Following the argument of internal representation, the task of situation recognition can be divided into

- the representation of the current environment, and
- the recognition of the current situation.

3.2.1 Requirements for a representation

[Sin98] describes two present-day hypotheses of the structure of an internal representation: the classical one and the alternative concept to it. The former is based on behavioristic positions. The process of information perception by the sensory system and the representation in the central nervous system is a stimulus-reaction-process. The brain is more or less passive - it is just a filter for the incoming signals. In the latter hypothesis, the brain plays an active part. It formulates hypotheses and solutions by means of predefined knowledge. The perception is used to confirm of these hypothesis. The author of this work emphasizes that experiments in that field lead to the conclusion that the structure is probably a mixture of both concepts, a filter as well as a creator of hypotheses and solutions.

We have to conduct some fundamental analyses in order to describe the environment.

- Static and dynamic elements are necessary

In order to identify the current situation, it will not be sufficient to use just a simple description of the surroundings. By using an image which consists exclusively of static data where no temporary constraints are included we will lose important information about the situation.

For example: if a picture shows someone lying on the floor, this can mean that the person is searching for something under a wardrobe, or the individual is playing, or even that this person is unconscious. This cannot be decided by merely looking at the picture. Additional information about the time the person is lying on the floor can be helpful in identifying the situation.

- A large number of data sources will entail a large amount of information

Due to the fact that biological systems are the examples for this work, and therefore a large number of sensors and actuators is used, the resulting "image" will contain an abundance of information. In order to handle all these collected information with a reasonable performance, it will be necessary to reduce this amount of data. For this reduction different factors are of importance.

- o Within the task of perception: a large number of values will be filtered out already in the perception layer

On the one hand, "intelligent" nodes are able to evaluate multisensors, i.e., for example, out of a number of brightness sensors, an average brightness value can be calculated, and only this value will be passed on. On the other hand, several different sensors will always perceive events or states. Again, the results of these sensors will be merged, and only the result will be forwarded.

- After the task of perception: there are several aspects which can be used for a further reduction, as for example that not all of the perceived information will be important in the current situation

B. Hallam describes in [Hal92] that animals extract important parts of various environmental features. When the animal senses that a significant feature may be present, it hears for example a noise that might indicate a predator, it stops doing other things in order to concentrate on the sensor concerned; in other words, it focuses attention on relevant stimuli.

But even by using a focus on few important aspects in the environment, a large number of detail-information will probably remain. Hence, we have to structure that information to handle it. That leads to the next fundamental aspect:

- The perceived information of the surrounding belongs to particular locations, objects or events.

A single value without context would make no sense – we always have to take the connections into account. For example, if we measure a temperature, we have to know the location of the measurement in order to be able to interpret the value. 5° Celsius would mean *very cold* in a living room, but *normal* in a fridge. Therefore, a possible structure has to preserve these connections.

In the following, I will state some possible representations in order to compare them and find their advantages and disadvantages. They are by far not the only possible representations – they are only examples in order to find factors which have to be integrated into the final solution.

Representation I concerning the location

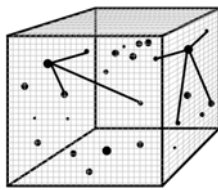


Figure 24: Location matrix

- A 3-dimensional matrix represents a room or an area
- Sensors as well as actuators are data points in that matrix. Thus, they have an exact location in that data-room
- It is a helpful representation for humans, since it is easy to see the spatial connections of events and situations. One can easily identify influences between data points. For example, one will see immediately the influence of an open window to the temperature near the window.
- However, it is more difficult to evaluate for technical systems, although there are different ways to achieve this evaluation. For example, one can use links between the data points to show relations. Furthermore, these connections can have weights to express the importance of this combination. Additionally, it could turn out to be helpful for measuring the distances between the data points. This representation is compara-

ble to the structure of IB4 and IB5 in Section 1.4.4.

Representation II concerning objects

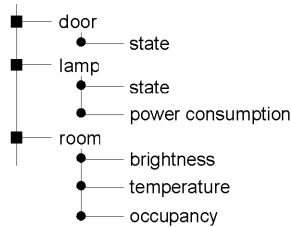


Figure 25: Object list

- All the surroundings are listed as objects with attributes. For example, we can have the object lamp, which has the attributes state (switched on or off), and the power consumption.
- For this representation a kind of rule base can be used to describe connections between objects.
- By using this list of objects it is more difficult for humans to interpret the environment.

Representation III concerning events

13:30:00 door opened
13:30:05 room entered
13:30:10 movement
 towards stove
13:30:12 stove switched on

Figure 26: Event list

- The environment is described by a list of events
- Each scenario is decomposed into single events
- There is an entry for each change in the surroundings, and each line of the list has a time stamp.
- For example, “someone enters the room and switches on the stove” can be divided into several events which are chronologically ordered.
- For humans, this list offers a readable description of the situations
- Technical systems can handle this list in an easy way as well. Situations can consist of combinations of some of these events. By assigning a time period to each of these situations to determine their validity, it is easy to extract them out of the list.

Now we have examined different ways for representing, and established a number of their strengths and weaknesses. In order to investigate which mode of representing the environment is best we will consider our 4 examples of Section 2.1.

1. Example of safety, where a child is in the kitchen, no adult is nearby, and the plate is switched on. The Situations III and IV have to be identified in order to create reactions.

Location matrix:

There must be a connection between the static data point *hot plate* and the dynamic point *child*. The system must be able to generate and delete data points dynamically, and also to make connections between these points and other (perhaps again dynamic) data points. In doing so it is possible to recognize Situation III.

If there is an additional connection between *child* and *stove* and we measure the distances, we can easily identify also Situation IV.

Object list:

Situation III can be represented by the two objects *child* and *hot plate*, whereby the later has the attribute *temperature* with the value *hot*.

For the next situation, we need either the position of the child or the distance to the stove. In both cases we have to use additional objects for those measurements. Since there are no spatial connections between the objects, we have to define the connections between stove and distance or position explicitly.

Events list:

Both situations are described in the event list as they are defined in Section 2.1. However, for an entry like “movement towards stove” we have, as shown with the object list, to define the spatial connections of measurements explicitly. Furthermore, it is expensive to describe the state of the hot plates by events: all changes of the temperature would result in entries in the list.

2. Example of security, where the system detects the breaking of a window. All three possible states of that scenario are key-situations. Each of them will lead to at least one reaction.

Location matrix:

The detection of a breaking window inclusive the location of that window is easy to realize with the location matrix.

The same applies to Situation II, where the intruder is detected inside the building. Since we have the exact location of the data points, we also have the position of the intruder by that.

To localize the intruder is the same as to localize the owner himself. Again, the only problem with that representation is the use of dynamic data points. For Situation I.I as well as for Situation II it would be necessary to have connections to dynamic points.

Object list:

Situation I can be identified by the object *window*. By using an appropriate name for the object, it would also be possible to identify the location of this window (for example *window_kitchen*).

The presence of persons can also be detected by the corresponding objects. However, the exact position cannot be extracted as long as there are no connections between the objects of the persons and the occupancy measurements.

Events list:

There will be the event *breaking window*, which will identify Situation I.

Also the Situations I.I and II can be detected by event messages. Again, connections between the persons and the occupancy measurements would enable us to generate messages related to the position of these persons.

3. Example of energy management, where a fridge is open.

Location matrix:

Again, this example shows the advantages and disadvantages of the location matrix. The movement towards the door in Situation IV can be detected by a

distance measurement. However, this measurement requires the usage of dynamic data points.

We reach the boundaries of this representation with Situation III.I. We have to detect that the fridge is open since a specific period. For that, we would need a kind of history, i.e. the knowledge about timing constraints.

Object list:

By means of the object list it is easy to identify the presence in the room and the open fridge. The movement to the door in Situation IV points to the same problem as already described in the example about safety.

Since this list offers only the current state of the objects of the surroundings, there is no information about the duration of Situation III.I

Events list:

In this example, the same applies to Situation IV as in the example about safety. The event “movement towards door” needs explicit spatial connections between the measurements.

For the detection of Situation III.I, the event list offers the time stamps of the entries in the list. By using this information about the occurrence of the events, we receive knowledge about the duration of the open fridge.

4. Example of comfort, where the phone is ringing in an empty room.

Location matrix:

The empty room, the ringing phone, and the presence of a person somewhere in an adjoining room can be represented by the location matrix. A problem could arise with representing the state of that person. However, this problem is not due to the mode of representation but to the sensory possibilities. The required knowledge whether this person is sleeping or not can be achieved by the position of the person, for example whether the person is lying on a bed.

Object list:

This case is similar to the location matrix. The states *empty room*, *ringing phone* and *occupancy* can be represented by the appropriate objects. For the *sleeping person*, we need a spatial connection between the bed and the person.

Events list:

All situations of the fourth example can be defined by the event list. The empty room is represented by the event of the last person leaving the room, the ringing phone by the corresponding event, and the sleeping person by using the time stamps of movement-events. However, without knowledge about the location of the person we will not know whether the person is sleeping or just sitting somewhere else.

REQUIREMENTS

As a result of these four examples it can be seen that there are only a few factors which are important for the description of situations. It must be possible to

1. use dynamic objects; objects, which are no fix elements of the environment

2. represent positions, movements and directions
3. know the time of the occurrence, or the duration of some events respectively
4. represent the state as well as the changes of data points

Our exemplary representations have mastered these four requirements with varying degrees of success.

In Table 3 we can see that none of our representations is able to handle all 4 requirements. However, for each of these requirements at least one representation is successful. This comparison shows that a combination of the location matrix and the event list would be sufficient to represent all four necessities. But it also shows that these two ways have nothing in common. Each of them is successful where the other one fails. Here, the object list could offer a connection between them. It combines

Table 3: Comparison of different representations

Requirements	Location matrix	Object list	Event list
1	– Connections to dynamic points are necessary	+ Dynamic objects are treated in the same way as static objects	+ Does not define objects at all
2	+ Easy to measure	– Explicit definitions of spatial connections between measurements necessary	– Explicit definitions of spatial connections between measurements necessary
3	– No time included	– No time included	+ Offers story with time stamps
4	+ Available in the data points	+ Available in the object definitions	– Possible, but expensive

advantages of the location matrix as well as of the event list: the representation of states and changes of objects in the environment of the former, and the usage of dynamic objects of the latter. A parallel usage of more than one representation is not reasonable, since each of them has some disadvantages, furthermore would it be expensive, and always a part of the two must be used for the identification of the situation. By that, I come to the conclusion that a reasonable representation has to include all 3 factors: objects with their state, the location of the objects, and the course of action of them.

3.2.2 Representation of the current environment

Since the object list offers already first attempts for a combination of the three representations, it is to serve as base for the final solution. Requirements number 1 (dynamic objects) and 4 (current state of objects) are met by the object list. Thus, we need the positions and the movement of the location matrix, and the timing constraints of the event list.

POSITION AND MOVEMENT

According to [Dom01], there are two groups of location models. The first group (1) deals with absolute specifications, for example grid based positions in form of $\langle x, y, z \rangle$. Models of the second group (2) are more abstract and therefore easier to adapt to automatic processing. There, locations are represented by sets, and located objects are referred to as members of these sets. The weaknesses of such a model are the expensive management of the dependences and the restricted spatial resolution. The combination of the two model types (3) is a so-called semi-symbolic model. A located object is represented by both: area coordinates and a membership in one or more location domains.

Furthermore, S. Domnitcheva states two different ways to associate any located object with a location in that work:

- Containment: The positions of objects are determined by identifying spatial regions, which contain those objects.
- Positioning: objects are tracked by reporting their coordinates; each object is represented only by its current position.

Now we have to contemplate about the use of these models in our object list.

Model (1): The absolute location of the object can be used as an attribute. Since we need the information about the position in the room, we have the attributes x , y and z .

Model (2): For the more abstract model, we can use a hierarchy of objects. That means one object may contain other objects. By that, we achieve a set of dependences, which can be used to identify the location of an object. We will not be able to name the exact position, but we will know about dependences, which are not available by the model (1). For example, if a distance sensor is mounted on a device, we can define by that hierarchy a unity.

Model (3): This means the combination of the first two models. We have the hierarchy of objects and additionally the absolute coordinates of the objects.

The four examples of the scenarios have shown that the absolute locations as well as the dependences can be necessary. Exact coordinates can represent the position of the intruder of example 2; for the movement towards the stove in example 1, the object *stove* contains the object *distance_sensor* and, we can therefore use the measurement of the distance in a simple way in this scenario. Thus, the resulting representation including the position has the structure shown in Figure 27.

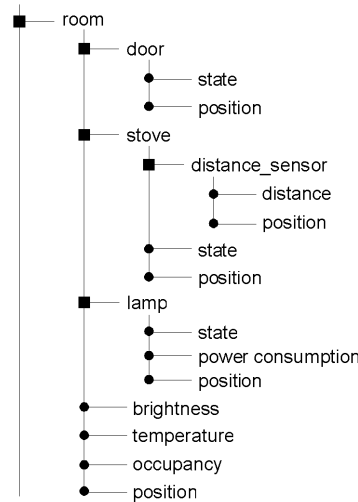


Figure 27: Object hierarchy

Again, I have to stress that the structure of Figure 27 is just one possible representation of the environment. As long as the chosen method offers the same features as the one presented above, any structure can be used.

TIMING CONSTRAINTS

Let us now regard the requirements on the timing constraints of our representation.

It can be necessary to know the exact time of the occurrence of an event. We will assume, for example, a building where the heating system is activated every day at a certain time. In order to detect a problematic scenario where the system is not switched on at time we will need the exact time information.

In the example about the energy management, it is not necessary to know the exact time, but the duration. In Situation III.I, we have to know that the fridge is open since a specific period.

We can meet both of these requirements by using time stamps as attributes. A change of the value of an attribute will also result in a new time stamp of that object. By that, we have the demanded exact time on the one hand. On the other hand, we can calculate the time periods in combination with the current time. If there is a change of the state of the fridge, i.e., the door of the fridge is opened at the time x , we can calculate the duration by $now-x$.

However, there are two weaknesses concerning time stamps.

- If there is more than one attribute for an object, we have to define the unity between attribute and time stamp, or rather we have to use a time stamp for each attribute.
- In Section 3.1.2, we advocate the usage of symbols instead of discrete values. By using time stamps, we would violate this demand – the system has to take care about that special type of the time stamps.

By using a method like the event list, we would overcome these weaknesses. We do not have to store time stamps for each of the attributes, since those connections are

given by the entries of the list. Furthermore, we would not violate the demand for the usage of symbols, since we do not use the time in the object hierarchy at all.

However, this solution would mean to generate a separate list beside the representation of the current states of the surroundings. Thus, if there are changes in the environment, we have to extract the states of the object hierarchy and additionally the entries of the event list.

So far, we have stated the requirements on the representation concerning the time constraints and their solutions, but each of them either shows weaknesses or the use is expensive. Therefore, in order to come to a decision or even to find a new solution, we investigate additional requirements on the representation.

In Section 2.1, we define the term scenario as follows: it is a composition of a set of single situations. By watching the images of the environment over time, we will be able to recognize the situation to which this sequence of images will lead. Using this requirement it will no longer be sufficient to store the time stamps in the objects. There would be no history, which can be compared with the sequence of situations in our scenarios. Hence, it is obvious that we have to operate the additional event list despite the expensive processing.

These considerations are conforming to studies of biological systems. In Section 2.2.3 I mention the division into the declarative and nondeclarative memory by Squire [Squ94]. He has furthermore divided the declarative memory into two parts: the semantic and the episodically memory. The semantic memory is responsible for the storage of facts. The latter one has to store their history. It is responsible for answering the question: what has happened when and where?

The system allows a flexible realization, depending on the needs and requests of the particular user, of the implementation of these parallel representations in order to optimize the performance and to reduce the efforts. I will state here one possible realization and assess its requirements.

Since it would not be reasonable to use two completely different ways to store the information, I suggest a combination of both. If we store the current states of the objects together with a time stamp, we do not require the event list. By that, no separate handling of the information about the history is necessary. However, it would result in enormous memory consumption to store all states about the environment all the time. At first, we can use the same principle for storage as in the original event list: we will store the states only if there occurs somewhere a change. A second problem is the storage of unnecessary static information. For example, if a window is closed, this state would be stored every time when there is for instance a change in the brightness of the room. To overcome this problem we reduce the stored information to the one changed value together with a time stamp. Thus, we almost achieve an equivalent to the event list. To find the states of some objects at a particular moment, it is necessary to move back in the history to the last changes of these values. Improving this, we can make use of a feature which will be discussed in detail later in this section: priorities. Biological systems use different mechanism to find important aspects of the environment. Adapting this idea to our system, we can find a way to achieve an improved representation of the states at a particular moment. If each of our objects has a priority to indicate the importance in the current situation, we can store these objects together with the changed one and the time stamp. Furthermore, in case of different levels of priorities, we can use this way to achieve

an additional feature. By that, we obtain a combination of the storage-efficient event list and the convenient but less storage-efficient storage of important objects. Let us assume we have a separate process, which is responsible for the administration of the history list, and let us name it History Administration Process (HA-Process). At first, we store all objects with their states when there is a change. With each change, a new entry is added. After a certain time, this process goes through this list, beginning with the oldest change. It deletes all objects with the lowest priority. After that, it moves to the next entry and continues the reduction. Thus, after some time, there will only remain the changed value in the older entries (which caused the storage and must not be deleted) and the time stamp (Figure 28.2). Furthermore, it means that every single entry, as long as it describes the near past, may contain many objects. But with time, it will be increasingly reduced (Figure 28.1).

By changing the interval when the HA-Process has to reduce the entries one is able to control the memory of the system.

- A short interval will result in a short list of entries with more than one object. Thus, a short interval means a low storage need, but also a limited memory concerning past representations of situations.
- A long time span between the cycles leads to a detailed long-term memory. On the other hand, it will need more storage, and the processing time to go through the list will become longer.

In general, the frequency of reduction should be adapted to the frequency of changes in the environment. The more changes there are in the surroundings, the shorter the

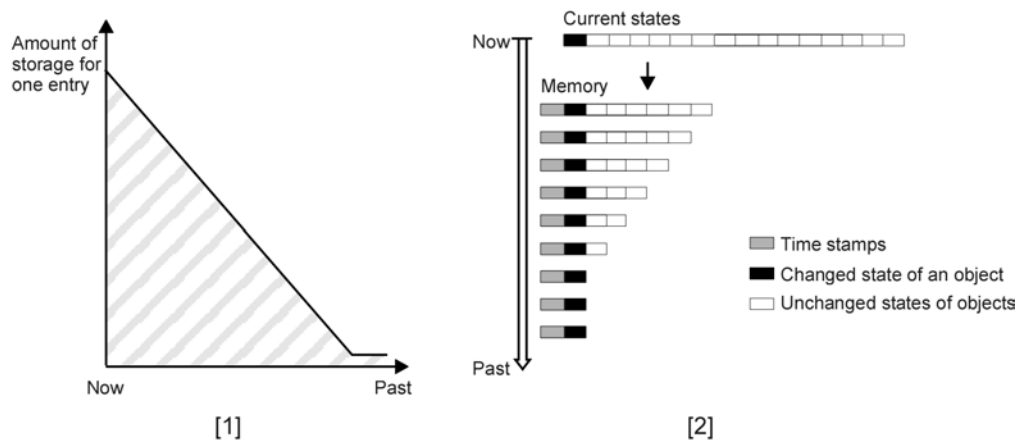


Figure 28: Storage of the history;
[1] Memory needed for one entry, [2] Structure of one entry over time

time between the cycles is to be set. A disadvantage of this rigid cycle time occurs if there are no changes in the environment for a longer time. The HA-Process would continue to reduce the entries, and if there are no new inputs, there would be nothing left than a list of time stamps with always only one object after a certain time. As a consequence, the question concerning the importance of old information arises. However, there can be no overall answer to that, since this factor strongly depends on the prevailing environment and the application field.

A similar approach is to define time periods in which we want to store a predefined amount of objects. For example, we store all objects for the first two hours, and every two hours we delete the objects with the lowest priorities. Again, we will lose our history list after a certain time if there are no changes for a long period.

To overcome the problem of rare new entries, we can use a different way to operate the HA-Process. For example, we can apply the events of new inserts as initiator. By that, the reduction takes place every time there is a change in an object. Thus, the lifetime of one entry is determined by the frequency of entries into the history list. A higher frequency will result in a shorter lifetime for each entry, whereas the objects of an entry will remain for a longer duration with rare changes. Additionally, this will not affect the entire amount of needed storage at all.

Furthermore, aspects like “delete always only one object or all objects with the lowest priority” or “is an older object with a high priority less important than a new object with a low priority?” can be considered.

PRIORITIES

As already mentioned above, the objects of the environment have to represent their importance in the current situation. In Section 3.1.2, we have assigned priorities to single symbols. The possible change of these priorities due to combinations of states of objects or of events is still unsolved.

At this stage, we have a complete description of all the surroundings, and we are therefore able to detect combinations of objects or events. There are two aspects that have to be dealt with: the system has to find a combination of objects that is important in the situation, and it has to determine the priority of it.

Combination of objects

The system has to know which objects are of special interest if they occur at the same time. There are different methods for attaining this knowledge.

For example, the importance can be explicitly predefined: the system possesses a kind of rule set which defines a specific combination as important. By that, if there is a change in the environment, it just has to check all the combinations containing this changed object.

The knowledge of important combinations can also be extracted of stored information about situations by the system itself. In Section 3.2.4, the method to store situations and scenarios is explained. These representations already contain all possible important combinations. Moreover, these situations and scenarios may also contain priorities for influencing the order of handling of them. This circumstance can be used to extract combinations of objects out of them.

Priorities of combinations

Once a combination is determined, we have to find the appropriate priority of it. Again, there are several aspects which have to be considered. We can assign the priority to every single part of the combination, or we can treat this combination as an object as well and assign a priority to this new object. One advantage of the treatment of combinations as objects is the simplified representation of situations: instead of describing a situation by many single objects, it can be done by few combined objects. On the other hand, the extraction of combinations of objects out of stored situations by the system itself (as mentioned above) is then no longer possible: since

we already use combinations of objects for the situations, we have to use an additional list that describes the structure of these combinations; this will lead to the first method of finding relations between objects, to the predefined rule set.

Accordingly, there are two different methods to define the priority itself:

- We can use predefined priorities.
- The system can calculate the priorities during operation time.

With regard to the maintenance of the system, the second method has to be preferred. So far, the definition of a priority can only be done in accordance with the priorities of the parts of the combination: it would make no sense to determine the importance of a combination of specific parts of the environment when ignoring the priority of each of them. This could lead to a situation where, for example, gas and fire at the same time are less important than only fire. Thus, we have to include the single priorities in the calculation. Possible solutions for the resulting priority are to consider the highest single priority or to increase each of the priorities by a specific factor. By that, even the proposed way by [Jov97] of blocking inputs is possible: we just have to define combinations for decreasing the priorities of objects.

Furthermore, I want to introduce here the “Priority List”.

In Section 3.2.4 we will discuss the requirements on the task of situation recognition. One of the major requirements is a sufficient performance of the system. In order to meet this necessity, our system has to possess some preprocessed lists of information. By that, it is possible to handle some tasks in parallel, and to work with reduced amounts of data.

The Priority List is a collection of all important objects of the environment. It is an additional structure to the already proposed one of objects with their states and the structure containing history. The detailed integration of it into the global system is described in Section 3.2.4.

3.2.3 Representation Layer Interface

A well-defined interface has to handle the communication between the layers of the representation of the environment and the recognition of the current situation on the one hand, and between the representation and the reaction task on the other hand (Figure 29).

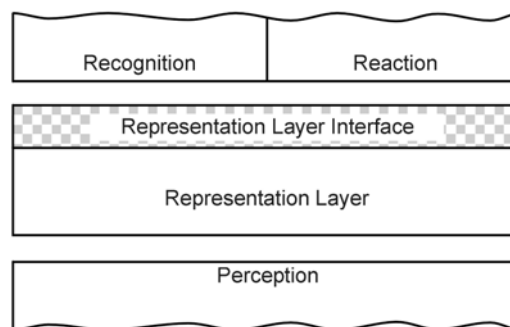


Figure 29: Representation Layer Interface

It is obvious that the *Situation Recognition* has to be based on the *Representation Layer*. This task has to analyze the offered description of the surroundings and identify situations in it.

The reason for the interface to the *Reaction Layer* is given in the evaluation cycle in Section 3.3. Since the system has to act intelligently by means of considering the consequences of its actions, it has to include the current environment in these reflections; it has to estimate the effects of a reaction on the surroundings.

SERVICES PROVIDED BY THE REPRESENTATION LAYER INTERFACE

Similar to the *Perception Layer*, the *Representation Layer* represents a server task for *Situation Recognition* and for *Reaction*. It is the passive part of this connection. At the same time, it acts as client task for the *Perception Layer*. Whereas it has to register at this layer in order to acquire information for constructing an internal image of the environment, both, *Situation Recognition* and *Reaction Layer*, have to register at the *Representation Layer* for receiving the required data for the analysis of the situation and the evaluation cycle of the reaction.

The following services have to be defined for the communication:

- **register ()**
Both, the task of *Situation Recognition* as well as the task of *Reaction* starts with a registration at the *Representation Layer*. The server task will answer either with a `handle()` or an `acknowledge()`.
- **deregister()**
Situation Recognition and *Reaction* end with deregistration at the *Representation Layer*.
- **get_data(msg_in)**
Both client tasks may ask for a specific element of the environment by using `get_data()`. The argument of this command identifies the required element in the representation offered by the *Representation Layer*.
- **get_history(command_in)**
In Section 3.2.2, the usage of a history list is explained. This list holds the states of some objects of the surroundings for a certain time span.
Both clients are able to inquire past events and changes in the surroundings by this command. They can use three possible arguments: START, NEXT and REPEAT. START is used for beginning a new query. NEXT means that the last value was perceived successfully and that the client is waiting for the next item. REPEAT indicates that there is a problem in receiving information, and that the *Representation Layer* has to send the last information once again.
- **get_priority(command_in)**
The priority list is introduced in Section 3.2.2. This list contains all important objects of the current environment. By using `get_priority()`, the clients are able to ask for these elements. Again, they can use three possible arguments: START, NEXT and REPEAT. START indicates the beginning of a new query. By sending a NEXT, the client notifies the successful transmission and

indicates that it is waiting for the next item. If there is a problem in receiving information the client uses REPEAT. By that, the *Representation Layer* has to send the last information once again.

- **get_EvaluationData(msg_in)**
For the evaluation cycle, the system has to apply changes in form of consequences to the internal representation in a virtual manner in order to analyze reactions. That means the origin representation must not be effected by these changes, since they represent no real events but only estimations. By using `get_EvaluationData()`, *Situation Recognition* asks for information about this image. The argument of the command identifies the required.
- **get_EvaluationPriority(command_in)**
Due to the changed representation for the evaluation cycle, we also have a priority list based on this image. The command `get_EvaluationPriority()` has, analog to `get_priority()`, three possible arguments: START, NEXT and REPEAT, which have analog meanings too.
- **handle(handle_out)**
If the chosen communication technique requires for example a shared memory provided by the server, this can be achieved by sending `handle()` to the client as an answer to a registration.
- **send_data(msg_out)**
The *Representation Layer* uses a structure as described in Section 3.2.2 as the internal image of the real world. The client is able to ask for a specific element of this structure by using `get_data()`. An answer is given by `send_data()`, whereby the demanded information item is offered as argument.
- **send_history(msg_out)**
Depending on the request of the client, the *Representation Layer* sends the first, the same or the next part of the history list.
- **send_priority(msg_out)**
Analog to `send_history()`, the argument of `send_priority()` depends on the request of the client: it either sends the first, the same or the next element of the priority list.
- **set_data(msg_in)**
This function is used by the *Reaction* task for passing the desired changes on to the representation for the evaluation cycle. The arguments are the required information about this change.
- **acknowledge()**
The *Representation Layer* sends an acknowledge message to the client to notify the registration.
- **repeat()**

If there is a problem with reading information from either the *Situation Recognition* or the *Reaction* task, the *Representation Layer* sends a `repeat()`. By that, the concerned client has to send its data once again.

3.2.4 Recognition of the current situation

Once we have a complete description of the entire situation, we can start with the recognition of it.

This task has already been analyzed by several researchers before. Two exemplary works are [Sac87] and [Hei98]. The former, O. Sacks, points out that the comparison of internal images with perceived information is necessary to obtain an idea of the environment, to make conscious experiences. The latter, H. J. Heinze, goes in the same direction. He advocates an internal representation as well, and emphasizes that we use this image of the environment as base for a conscious recognition, and consequently as base for thinking and reacting.

REQUIREMENTS ONTO SITUATION RECOGNITION

There are several requirements which have to be met by a system of situation recognition. First of all, it has to offer a high accuracy. This requirement comprises a spacious scope of duties, since it includes many dependencies. For example, a high accuracy will depend on the precision of the description of the environment as well as of the predefined scenarios, which have to be identified. The better the descriptions are the better will be the result of the search. However, sometimes it will not be possible to know every detail of a scenario or one simply forgets about a part of it. Thus, this task should also be able to handle incomplete descriptions. Naturally the system has to process the recognition as fast as possible – at least faster than the system receives new information about the environment. It would make no sense to identify a situation which has already changed in the meantime.

In Section 2.1 I mentioned that the definition of scenario by situations can be used to make predictions about the future. This circumstance can be useful in another requirement: the System is to be prepared for happenings in the future. By that, we can also achieve an enhanced performance of the recognition itself, since the system has already a rough idea about the next situation.

If the system is used in a new environment, the user will have to define everything in detail. Here, the ability of the system to adapt to the environment would be a considerable improvement. The system should be able to add new knowledge and change existing knowledge – it should be able to learn.

To fulfill these requirements, we can use different possible methods. However, the methods of representation of the environment, the definition of the predefined scenarios, and the task of the recognition are not separable – they belong together. Therefore, we have already set the course for the task of recognition with the structure of Section 3.2.2.

But, as already emphasized, the flexibility of the system allows for several representations, the sole limitation being that the chosen representation has to offer the required features. The same principle applies to situation recognition.

Whereas these possibilities are open, we still have to define the “environment” of them. The way of presentation and recognition may differ – the management of them will remain the same. We have to consider the infrastructure of these separate tasks, which combines them to one unit and which is open for other methods at the same time. We will reflect on this later in this section.

In Section 1.4.4 two different methods of machine learning are presented: rule induction and instance-based learning. Though these two ways differ in many factors, both of them are possible strategies to be used with our data structure in order to meet the tasks of searching and learning.

CN2, for example, can achieve the process of representation and recognition in a quite readable way by using “if ... then ...” rules. The only extension to the pure structure of objects and attributes are the logical operators for the connections between the relational expressions.

In case of instance-based learning mechanisms, IB5 offers many useful features for our system, for example the handling of novel examples or of noise. For the representation, it uses a n-dimensional coordinate system, where the axes are defined by the attributes of our objects. The identification of a situation is done by distance measuring between a new point in the data room (the current situation) and an existing one.

For this work, I have chosen the rule base, since the work aims at the model behind, the management of situation dependent behavior. Although IB5 may have advantages in some fields, it will be easier to explain this management by the use of a rule base – the processing in the IB5 algorithm is not evidently readable for humans.

However, we cannot start to handle the rule base before we have not done some definitions of the management first. We will analyze every aspect firstly in general, and secondly with regard to the use of a rule base and our structure consisting of objects.

SCENARIOS

First, we have to define what we want to recognize at the end: scenarios. Our system has to possess a set of predefined scenarios in order to be able to identify the current situation; it has to possess a memory. This is an important aspect: we want to recognize the current situation, and we want to do that by using scenarios. It does not matter whether this set of scenarios is predefined from the very first start of the application or the scenarios were learned one after the other.

In Section 2.1 it is mentioned that a scenario is more than just a simple representation of the current state of the surroundings. It has to contain the story which led to the current situation; it has to consist of a set of situations.

Definition 2. A *situation* is the representation of the current state of the environment, of the immediate and concrete states of the surroundings in one moment.

We can distinguish between a

global situation consisting of every single part of the environment, and a *partial situation*, which is limited to the information of a particular part of the global situation.

In order to establish the differences between situation and scenario, I present the definition of a scenario as follows.

Definition 3. *A **scenario** is described by a set of situations over the time.*

We can distinguish between a

***global scenario** consisting of a sequence of global situations, and a **partial scenario**, which consist of a sequence of partial situations.*

The differentiation between global and partial situations as well as global and partial scenarios makes sense in the way that we have now the base for the construction of scenarios in our situation recognition system. This will be explained in Figure 30 and in the following preposition.

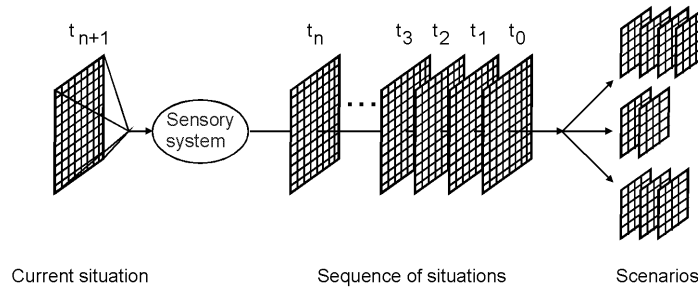


Figure 30: Construction of scenarios

Preposition 2. *A sequence of global situations may include the descriptions of more than one partial scenario.*

Figure 30 gives a rough outline of the creation of a scenario. It starts with the current environment. The lattice represents all data points which can be perceived via a sensory system. This lattice of data points will be stored – it represents one situation (global situation). In the course of time, we will have stored a sequence of situations, a sequence of snap shots of the environment. As already mentioned, not every aspect of the environment will be important. Therefore, we will take only some objects of each situation (partial situation; shown as the small lattice). By putting these partial situations together, we receive partial scenarios. We may obtain more than one partial scenario of a sequence of global situations.

To identify a specific situation, we have to analyze the development of it. This idea is analog to [Dav97], where a similar process is described. It is proposed that in order to recognize the behavior of a person, one has to compare characteristic features of events with stored information of a database.

We have to compare the history which led to this situation with the histories describing our stored scenarios. If we identify the history of the situation, we have identified the situation itself. Thus, situations and scenarios cannot be separated: the former is an element of the latter and at the same time can be the result of it. There can be situations with a longer history as well as scenarios which consist of only very few situations.

The internal representation of the perceived environment can adopt different states during the task of situation recognition.

- **Unknown:** The perceived situation is unknown to the system; there are no similarities to situations in the stored scenarios.
- **Possible:** Again, the perceived situation is unknown to the system. However, there are resemblances to situations in the predefined scenarios. The system is able to identify at least parts of the situation.
- **Identified:** The system has found a correspondence between the last perceived situations and a stored scenario including the final situation of that scenario.
- **Probable:** The situation is not yet identified. There are already correspondences between perceived situations and a stored scenario. However, there are still situations of the scenario missing. A probable situation is the next missing situation of the scenario – it will probably take place next.
- **Calculated:** In Section 3.3 we will see that the consequences of reactions have to be analyzed. During this process, the system has to calculate the situation which would follow after the current situation due to an action.

We can make some general regulations, mechanisms to improve the performance of the task of situation recognition also for managing these states.

If we assume that we have a very large number of data points (in our case objects with states) for the representation of the environment and also a large number of predefined scenarios, we can compare all the elements of these two lists with each other in order to identify the situation. However, this comparison will take a long time with an increasing number of elements. Hence, we have to use a very clever algorithm for this comparison, or we have to narrow down the choice (which does not exclude the clever algorithm). However, a preliminary selection will also speed up every mechanism for comparison.

PRELIMINARY SELECTION

Taking advantage of the priorities in our representation by means of the priority list, we can carry out a first preliminary selection; this structure is already introduced in Section 3.2.2. A situation of a scenario will consist of a few objects of the environment, whereby some of them will be of a higher importance, for example a breaking window, someone is in the room or the stove is switched on. This fact can be used to search at first the scenarios which contain at least one of the important objects of the environment. Thus, this selection enables us to sort out all *possible* scenarios. Again, there is no dividing line between situations and scenarios. If we have identified parts of a situation and found a *possible situation*, we have thus found a *possible scenario*, the scenario of which this situation is a part.

By using priorities in the representation, we take advantage of one of the requirements described in Section 3.2.1. Analogue to biological systems we will focus our attention on relevant stimuli.

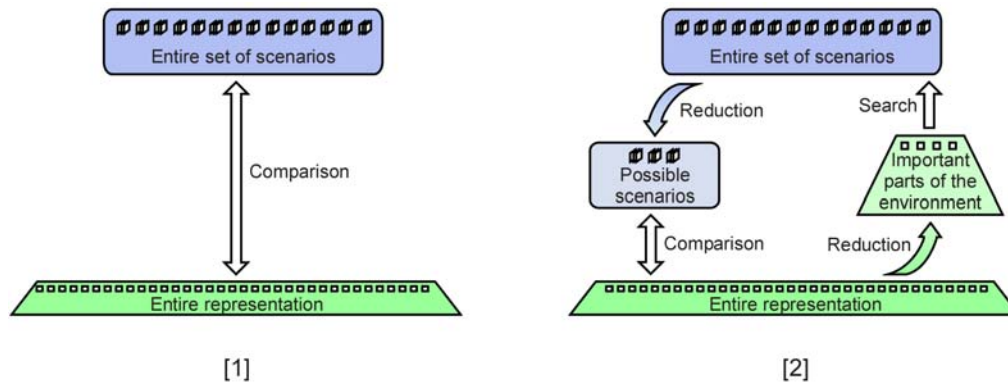


Figure 31: Preliminary selection of objects and scenarios;
[1] Comparison of entire sets, [2] Comparison of reduced sets

Figure 31.2 shows the two reduced sets which can be achieved by the use of important aspects of the representation. Compared to the entire set of predefined scenarios and objects of the representation (Figure 31.1), we can handle much smaller sets now.

In the first comparison we use a makeshift-representation, containing only the important aspects of the environment. In case of our object structure this can either mean an additional list containing only objects with high priority or we take only these objects out of the entire list. The chosen method will depend on aspects like storage used to hold the representation or the costs of either an additional list or an additional search for high priority objects.

In the next step, we carry out a search to find all scenarios which contain at least one of the important elements of the representation. Again, this can be done by means of an additional list of scenarios, which are partly identified, or by marking these scenarios in the complete list.

At this stage, I will introduce another differentiation of situations:

- **Coarse situations** and
- **Detailed situations.**

In [Sta97] this differentiation is explained in the way that a description of a representation is multiscale: we work at first with “big pictures” (coarse situations), containing only main features, and then we go into more detail (detailed situations).

At first, we will recognize only a coarse representation of the environment, containing only very few features. After this coarse situation is identified, we will move on to a more detailed description of it and verify this detailed situation. The differentiation can be exemplified as follows:

Again, we take a dark room and a person who enters this room. Our system has to detect this situation and change the brightness. It has three possibilities to carry out this task: it can switch on the light on the ceiling, it can switch on a small table lamp and it can use blinds. Let us assume that the system has already switched on the ceiling light; however, it is still too dark in the room. Thus, it has to use one of the other two possibilities. Now it is faced with two possible ways:

1. It is too dark, thus the system wants to switch on the ceiling light (the reason why it would start with that action is a time factor described in Section 3.3). Now it checks whether this light is already switched on, and consequently goes on to the table lamp. Again it checks whether it is switched on and, since it is not, it activates the lamp.
2. The system detects that it is too dark and that the ceiling light is on, whereas the table lamp is off and the blinds are closed. Consequently, it activates the table lamp. If this is not enough the system will detect that the new situation with the ceiling and table lamp switched on but the blinds closed is still too dark.

There is a big difference between these two ways. In the former, the system has several possible solutions to a situation. It comes to a conclusion and checks afterwards if that action would make sense. In the latter one, the situation is defined in detail at the very beginning. That means that there is a large number of scenarios and that there is only one solution to a scenario. In both cases the processing can be very expensive. Either we get solutions and have to recognize that they are not useful afterwards, or we have to work through a very large number of possible scenarios to receive a result.

The differentiation into coarse and detailed situations offers a solution to this problem. At first, we define very simple situations. In our example, the situation is merely described by the dark room and someone entering the room. The advantage is that the system will recognize this situation very fast. Sometimes even the priority list may be sufficient to identify such a simple scenario. In the next step, a set of detailed situations follows the identified coarse situation. Hence, we may ignore most of the information offered by the representation layer and concentrate on limited information (Figure 32).

Basically, this differentiation can be conducted several times, depending on the complexity of the situations. In every step, some features are added to the situation. In an extreme case, we add only one feature in every single step. This process is comparable to the search in a decision tree as described in Section 1.4.4.

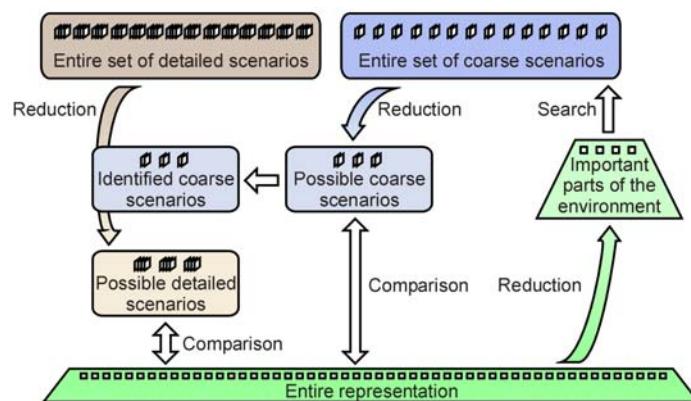


Figure 32: Coarse and detailed scenarios

SITUATION RECOGNITION WITH THE OBJECT STRUCTURE

As already explained, I will use a rule base for the description of this work. A situation in a scenario can therefore be described by a set of objects with particular states together with logical operators. An example of a situation of the scenario of Energy management is “[object]door [object]distance sensor [attribute]*near* [operator]AND [object]fridge door [attribute]*open*”.

Analogue to that, the representation of the environment which also consists of a structure of objects and attributes can be seen as a list of objects with attributes and their states. To look for a specific situation in this representation of the entire environment is therefore a simple search, depending on the method of storing data – for example, a *select query* in case of a database.

However, as mentioned above, the identification of one single situation will not be sufficient for an intelligent reaction. Beside the search in the momentary representation, we also have to inquire into the past representations stored in the history list (Section 3.2.2). Hence, once we have identified a situation of a specific scenario, we also have to search for the other situations of this sequence in the history list. Again, this search depends on the method for the storage.

Today’s databases are powerful enough to handle a large number of data – nevertheless, they can be assisted by the preliminary selection mentioned above. A list containing only important objects can be hold in parallel to the entire representation. A change of the priority of an object in the entire list will result in a change in the list with the prioritized entries. In the same way, the additional lists of possible, probable, calculated etc. scenarios can be created and used. All lists can be managed in parallel to the actual system. Instead of searching situations or scenarios respectively in the entire amount of stored information, this search is performed in several steps in parallel. The final search for the identification will be conducted on a small section of the entire knowledge base. Therefore, the amount of information which has to be handled can be reduced without impairing the task of situation recognition itself.

LEARNING

The memory of the system does by no means represent a constant quantity. In the course of time, many situations will occur which are still unknown to the system. Thus, these situations have to be added to the set of predefined scenarios. Hence, the number of stored scenarios will extend; at the beginning it will rise fast, but after some time this development will slow down since most of the scenarios will already be described and stored.

Again, this is a reason for using mechanisms as described in the previous section. After some time, it will become increasingly difficult and expensive to filter out the current situation due to the extended number of stored scenarios.

If we compare the learning behavior of the algorithms presented in Section 1.4.4 and Section 3.2.4, we notice that they use a different mechanism.

- Rule induction:

Eager Learning: They generalize beyond the training data before observing new examples. For CN2, this means this algorithm inductively learns a set of propositional “if ... then ...” rules from a set of training examples. After that, the learning phase is finished and it is able to classify new cases.

- Instance based learning:

Lazy Learning: They defer the decision to generalize beyond the training data until each new example is encountered. For IB5 this means that it does not need all learning examples in the very beginning.

This different learning behavior has of course influences on the computation time as well as on the task of classification itself. Lazy methods generally require less computation during the training, but more computational time during the classification. In case of IB5, nearly all computation takes place at classification time. On the other hand, lazy methods offer more flexibility with new cases. They are able to constantly change their knowledge. In eager learning methods, the knowledge base is committed at training time.

Thus, since each of these methods is restricted in some way, none of them can be used as the only learning mechanism. To overcome these problems, one can for example use combinations of different methods as shown in [Tin96].

In addition, we can use mechanisms which are in accordance with to the special scope of activity in this work. In the following, two mechanisms, learning by observation and learning by association, are presented.

Learning by observation

The first method can also be seen as a kind of imitation. The idea of it is to create connections between scenarios and observed actions of the user. For example, someone enters the room and switches on the light. Here, the system can make a connection between the current scenario and the activation of light.

In [Len99], a learning mechanism based on observation is presented, which uses a sequence of time steps with sensor inputs from the environment, any actions performed by the user, and annotations indicating goal achievement. A special form of rule format is used for the knowledge representation of their system.

Another example of a system based on observation can be found in [Wan94], who describes the usage of this learning mechanism in an agent-based environment. There, the observations of an expert agent consist of the sequence of actions being executed, the state in which each action is executed, and the state resulting from the execution of each action.

If our system detects an event, i.e. an action executed by the user, it means that the system either does not have the capability to do this action by itself, or that there is no appropriate scenario yet defined.

If an actuator also works as a sensor, i.e. if it is able to detect changes which were caused by external sources, the question concerning the capability is clarified. For example, a switch in LonWorks is able to detect a change and reproduce it as well. Furthermore, the action of the user can be detected without a connection to a necessary actuator. For example, if someone approaches a washbasin and turns on the faucet, an acoustic sensor can detect this. If there is another pipe which can be controlled via valves the system has to “know” that it can reproduce the detected information with theses actuators. Concerning this problem, I introduce the relation

“intention-reaction-consequence” in Section 3.3. The reason, the intention and the possible result of action must be defined for each reaction. This set of information can either be introduced by the user, or the system has to find the information by itself.

Finally, the system has to check whether there is already a scenario defined for this event, which has to be adapted, or whether there is none at all. However, in both cases the questions arise which elements of the environment are participating in this event and when was the beginning of the scenario, i.e., what is the first situation that indicates that the scenario begins?

To answer the first question we can use the priorities of our objects, whereas the latter questions requires the search for a past situation when there was a change of an important object of the current situation. However, either the system has to ask the user for confirmation of the definition of the new scenario, or it will result in an iterative process. If the system has to define the scenario by itself, it will probably not be able to perform this at first in the right way. Thus, next time it will have to adapt this scenario. This process will continue until the system is able to overtake the handling of the scenario.

Learning by associations

The second idea is based on results of biological studies. [Kan00b] describes association areas in the human brain. These areas consist of connections between different sensory inputs and motor outputs. Analog to that, [Roh94] emphasizes that the representation by associative connections is an important principle of the structure of the human cerebral cortex. [Mai97] analyses the associative learning, and distinguishes between different methods in this context. A well-known example of “self-made” connections is the classical conditioning, for example the Pavlov’s dog. There, chronological relations (associations) are created between stimuli and response.

The question arises now how we can apply these findings to this work.

In works like [Hab00] or [Kie99], brief descriptions of semantic networks are given. Such networks represent the structural relations of meanings, and have their name due to the net-shaped structure of this memory model, shown in Figure 33. These conceptual networks are assigned to the semantic memory. Again, we can find here a partitioning of the memory into two parts: the semantic memory and the episodic one. This is comparable to the partitioning described in Section 2.2.3, where Squire has defined the declarative and nondeclarative parts of the memory.

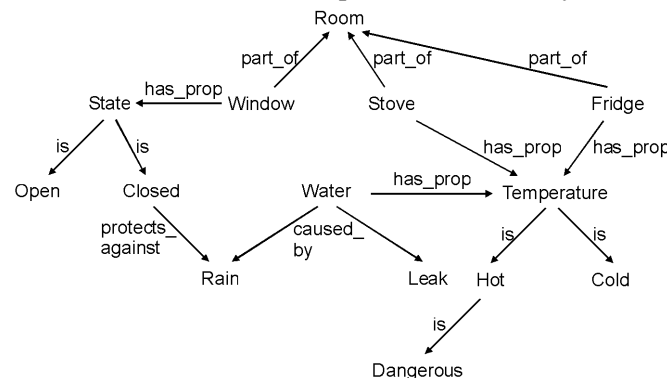


Figure 33: Semantic network

The idea behind the usage of a semantic network is to identify relations between parts, states and events of the environment in order to recognize the situation. If the system faces an unknown event, it can try to find the reason and the concerning elements by itself. For example, it is raining, the window is open, and a sensor near the window detects water. Even if there is no predefined scenario, the system can find the connections by itself. Moreover, as mentioned in [Kie99], this knowledge can be used in other, similar situations afterwards. If the system has made the connections once, it can use this knowledge not only in the same situation next time, but also for other situations which contain these connections.

Nonetheless, the network has to be extended by some additional features. For example, it needs to know the location of the elements. It could also be necessary to include chronological aspects.

However, the integration of learning mechanisms of this kind remains a challenge for further works.

3.3 Reaction choosing

So far, the system is able to identify the current situation with regard to the current scenario. Depending on this situation in connection with the scenario, the system has to react. For example, the situation is “the room is dark and someone is in it”. Without the story of this situation, we are not able to determine a reaction. It could be that the user has switched off the light or has just entered the room. In the first case, the light should remain switched off; in the second case the system has to activate it. Thus, the reaction depends on the entire scenario.

In each situation, the system may possess several possible reactions. Therefore, it has to find an appropriate reaction in a pool of possible ways for acting as a first step.

Humans can react to situations in a conscious or in an unconscious way. Unconscious reactions can take place due to reflexes, i.e. instinctive-automatic answers of the organism to an external or internal stimulus. By reflexes, humans are able to adapt quickly to changes in the environment. Conscious actions, however, will take more time since they include higher regions of the brain. In return, these reactions may offer the possibility to consider the consequences of them.

[Hei98] has described it in the way that the conscious human is able to put the “self”, the subjective experienced representation of oneself into any spaces of time and situations. By doing so, one is able to simulate the consequences of possible decisions without real effects. Due to this, one is prepared to handle the different requirements of life.

I will use the same way of processing in this work. Reactions which will need no complex calculations but have to take place immediately will be treated in the lower layers (Sections 3.1.1 and 3.1.2). There will be *no* identification of the current situation. It must be assured that there is no need for higher layers for these reflex actions. Complex reactions will happen only *after* the situation is identified. Therefore, the execution of them will need more time. As a countermove to that, we find here the possibility to integrate mechanisms to estimate the consequences of these actions.

A consequence of our request to establish reactions with foresight is that *one* scenario can lead to *several* different reactions. By simulating, the best one of these reactions, the one with the best result, should be selected and realized. Hence, we have to think about a way to find this “best” reaction, a way to estimate the result of a reaction.

In the sixties, the same was carried out by [McC69] by using formal specifications. In this work a formal description of a reaction in the world is used. This action converts a given situation into the next situation, the resulting situation. Again, this result is calculated in a formal manner.

Starting with this work, there was a variety of different methods for the prediction of factors after the execution of an action, as shown in [Sch00].

In [Aam94], this process is described by the CBR-Cycle (Case-Based-Reasoning), shown in Figure 34.

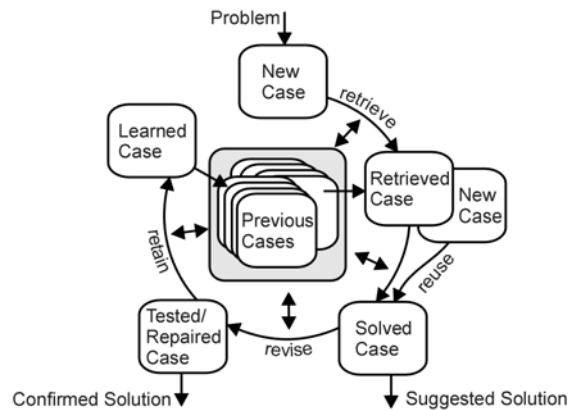


Figure 34: CBR-Cycle

After introducing a problem to the system, a suitable case is selected of a set of previous cases (retrieval phase). This case will be either accepted or adapted (reuse phase). In the revise phase, the solution will be tested for correctness and usability and, if necessary, revised. The final solution will be given back to the system in the retain phase – either by simple storage or by a learning mechanism.

Although a way of reaction choosing is described which seems to fit correspond to this work, the proposed solution is not unrestrictedly useable.

To select the best case of the previous cases (retrieve phase), the steps “identify features”, “search”, “initially match” and “select” are used. The idea behind is that each case is described by some features. Due to these features, the best case will be chosen. This may be done by using the system’s own model of general knowledge, or by asking the user for confirmation. In the revise phase, only this selected case is evaluated and, if necessary, adapted. This means that the selection itself is already finished when the evaluation starts.

In our case, this means as well that we would have to describe our scenarios and reactions very carefully and completely: we would have to consider all influences and interactions, and need to include these aspects. There will be many additional features which will be important in the case of a reaction, which are, however, not related to the origin problem. In other words: a reaction which is the right one once need not be the right one next time. For example, someone enters the room and it is dark; in

addition, there is a gas leak. If we assume that there is no additional control instance which would handle the gas immediately the description of the problem “someone enters the room and it is too dark” must not lead to the reaction “switch on the light”. The description would not be sufficient – it would need some additional information about the environment which is not directly related to the problem with the light.

However, one cannot expect that the creator of a system can consider everything. Thus, if we want to have a system which is able to consider the consequences of its reactions we have to change the order of this cycle.

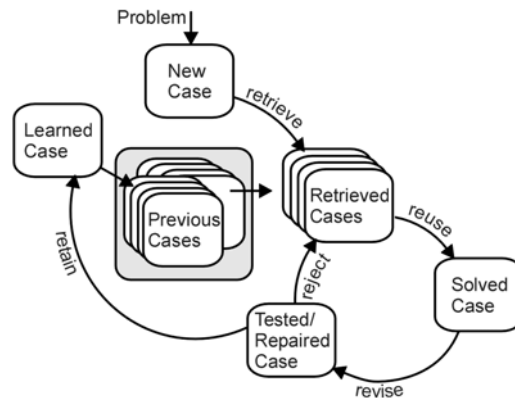


Figure 35: Adapted CBR-Cycle

Figure 35 shows the adapted CBR-Cycle. After introducing a problem to the system, a set of suitable cases is selected of a set of previous cases (retrieval phase). One by one, these cases are taken (reuse phase) and tested (revise phase). If the result of the test is successful, this solution will be taken (retain phase), otherwise it is rejected and the next retrieved case will be taken.

By changing the order of the original structure, we have now two cycles. The test of selected cases is integrated in a separate cycle.

Again, the selection of cases of the previous set is done only by identifying features of them. However, we will not consider only one case and ignore the rest, but take a set of them. We can choose the best one of this set. Supposing that there is no suitable situation among the selected ones, we can even make a new selection of the previous cases. Additionally, if there is no appropriate previous solution available, we can try to adapt one of them or ask the user for assistance.

However, this adapted CBR-Cycle describes only the basic steps that are necessary for this task. There are still open questions in this system. For example, it can announce that there are previous cases – are there connections to the problems (scenarios), and how are they defined? How to find the retrieved cases? How to test the selected solutions?

In order to find answers to at least some of these questions, we will start with a simple thought experiment. Let us assume you enter a room which is completely new for you. You can identify nothing, no switch, no light, no device. Nevertheless, there are many things around you in this room – you just have no idea how to use them and what they are used for.

Since you have to stay in this room for a longer time in our experiment, you have to know the functionalities of these devices. For example, if you would like to control

the temperature of this room (it is rather cold there), you have to identify the right button. But is it enough to know the control in order to regulate the temperature? As mentioned above, everything in this room is new for you. Thus, let us say that it is not a simple button but a foreign object, and you have no idea of its capabilities. Therefore, you have to know how to use it. Now you know how to handle that button – is that enough? Of course not, since you still do not know what will happen if you make use of it. Using it wrongly, you could turn the room immediately into a fridge. Thus, you have to know the effects if you bring this button into a specific position.

Now you have sufficient knowledge to control the temperature. Let us summarize the ideas of this experiment: if you want to do something, you have to know

- What to use it for,
- How to use it, and
- What will be the result when using it.

Now, you have two possibilities to obtain this information.

1. Someone tells you everything.
2. You have to try it out by yourself.

Some of the devices may be dangerous, some not. Hence, there should be someone who informs you about the dangerous ones. Concerning the rest, it does not matter if there is a supervisor or if you analyze them by yourself.

This example is comparable with the learning behavior of a child. The parents will explain the dangerous devices in the environment (for example the hot surface of a plate), and the child itself will discover the rest step by step.

As a consequence of these considerations, I come to the following conclusion.

Preposition 3. *The description of a reaction consists of 3 parts:*

- *The intention;*
- *The reaction itself;*
- *The consequences.*

The intention is the meaning of an action, the reaction the described way what to use and how to do it, and the consequences describe the situation after this action. For example:

Intention: Change of brightness;

Reaction: Switch into the state off;

Consequence: Less brightness.

RETRIEVAL PHASE

In order to find a solution to the retrieval phase, we have, first of all, to decide how the process after the situation recognition is going on. A situation (the resulting situation or, as shown in Section 2.1, a particular situation of an identified scenario) leads to a reaction – the question is now: how? I do not want to discuss how to find a reaction, but which steps are necessary for it.

A situation might either lead directly to a reaction, or the result of an identified situation is expressed by intentions to do or to change something. The succession

depends on the type of situation, if it is a coarse or a detailed situation. A reaction cannot follow immediately a coarse situation. This description would offer only the main factors of the entire situation. In order to perform a reaction, we will need much more information, for example the exact states of all actuators for the reaction. Hence, a coarse situation has to be followed by an intention. As already explained, we will need additional information for reactions. Therefore, we have to collect the data necessary for the chosen intention in the next step – the result of this collection is a detailed description of the situation. Thus, an intention has to be placed between a coarse and a detailed situation (Figure 36).

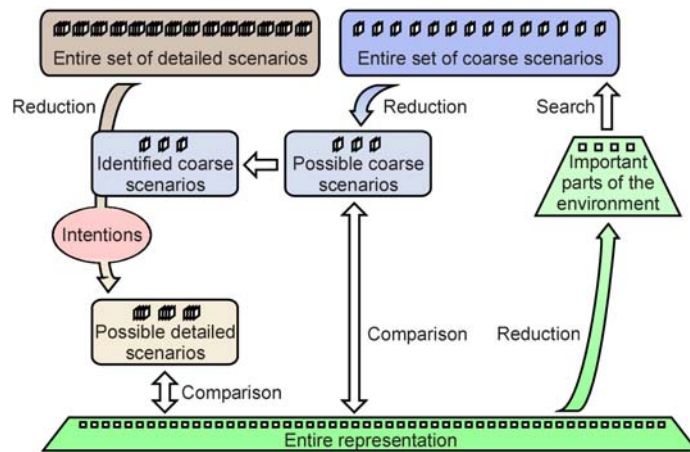


Figure 36: Usage of intentions for the situation identification

By that, we have again extended our identification of the situation. Now, it contains a block with intentions, a part of the reaction task. We can see that we can no longer separate the identification of the situation and the selection of a reaction. On the contrary, they influence each other in several steps. A first analysis of the situation leads to the intentions of a reaction, which, consequently leads to the next step of the situation analysis.

There are two possible ways to acquire the appropriate intentions for a specific situation (scenario). The relations between them are predefined by the user, or the system has to find them by itself. The latter method is, of course, more comfortable for the user. It can be realized, for example, by using the important objects of the situation. In our example with the dark room and someone entering this room, the system would be able to find the intention, since the low brightness would be decisive in that scenario. However, there is a problem with this idea: it will work only if the necessary actuators for the reaction exist in the scenario. Let us use the scenario of security of Section 2.1 as example. One intention should be to activate the alarm. But since the alarm giving system is no constituent of the scenario itself, the system will find no connections between them.

To overcome this problem, we can again use a learning algorithm for this task. By that, the system could learn the relations between scenarios and intentions. Here the same considerations and factors as in Section 3.2.4 are valid.

EVALUATION CYCLE

It is obvious that there must be an evaluation of the chosen reactions. However, the possible methods to achieve this evaluation remain open. One can use case-based reasoning methods as for example presented by A. Aamodt [Aam94]. Specific knowledge of previously experienced problem situations is used. A new problem is solved by finding the most similar past case and reusing it in the new problem situation. Another way is to make use of Proposition 3, where we already possess a description of the consequences.

Though, regardless of the taken method, all of them lead to the problem of evaluation in the end. Even by knowing the consequences of an action, we have to compare these consequences and choose the best one.

There are several aspects we have to consider for this task.

Correctness of the evaluation:

If one wants to use an additional simulator as listed in [Aam94], it has to be assured that it simulates processes in the environment in a correct way. The internal model of a simulator has to be strong enough to give correct feedbacks.

In case of using the descriptions of the consequences for the evaluation, these definitions have to be conducted very carefully; one has to consider every single effect of a reaction.

Duration of the evaluation:

Every evaluation will take some time to find the best solution. However, we want to use the system in a real environment, and there can happen something permanently. Thus, we have to consider that the current situation may have changed during the evaluation process.

Quality versus urgency:

In the section concerning the retrieval phase, I have already mentioned that there can be several possible reactions to one intention. Thus, if the system selects one reaction, there can still be another one which is better or more efficient. However, one has to weigh up the costs in order to find a better reaction if the selected one already fits to the problem.

For the evaluation, Aamodt states three possible methods: the evaluation by a teacher, in the real world, or in a model. The first one is not acceptable in our system, since the user cannot be expected to decide about every reaction. Also the second one, the evaluation in the real world, cannot be used in our application field. We cannot allow the system to perform a dangerous action in order to find out that it is dangerous. Thus, the only remaining solution is based on a model, on a simulation.

However, I will not use an additional simulator for the evaluation task in this work but use the system itself for it. Since it is able to recognize the situations in the current environment, it offers everything we will need for the evaluation.

Thus we have to deal now with the aspects described above.

Correctness of evaluation:

Since we will make use of the definitions of the consequences, we have to achieve the highest possible correctness of them. An important step is to decompose complex reactions into a set of simple reactions. Thus, it is much easier to consider the consequences of these simple actions. This method is also supported by our definition of scenarios. Instead of a complex reaction at the end of a scenario, we have several simple reactions during the entire scenario. Furthermore, this decomposition is the key to the solution of the next aspect.

Duration of evaluation:

By using short, simple reactions, which have only few effects, we can considerably reduce the time needed for the evaluation.

Quality versus urgency:

In order to balance the quality of an action and the urgency of it we need additional information about this urgency. A solution could be to include a time factor in the intention, which indicates the time span in which a reaction has to take place. The same effect could be achieved by using priorities or even a number of permitted additional search-cycles instead of the time factor.

In the example with the dark room it would be important that the light is switched on immediately. Therefore, the intention to switch it on would have a high priority or a short time factor respectively. In the next step, the brightness can be adjusted with a lower priority. Then the system has time enough to look for efficient alternatives, for example the rotation of the lamellas of the blinds, or to consider aspects like power consumption.

Another factor which can be important in this aspect is the time needed for the reaction itself. The switching on of the light would be much faster than a change of the blinds. If the intention has a high priority, since it is a problem situation, and the system has to find a solution very fast, it will also be reasonable to choose a reaction with a short processing time.

Thus, we have two time factors which have influence on the evaluation, as shown in Figure 37. In the previous section we have seen that an intention of a coarse situation is followed by the identification of a detailed scenario, which, subsequently, leads to a reaction. Therefore, within the time factor of the evaluation, we also have to handle the analysis of the detailed situation necessary for this intention.

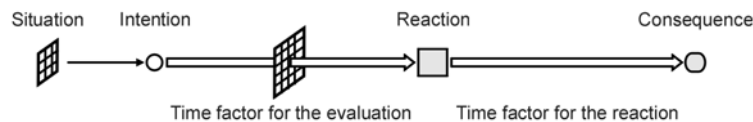


Figure 37: Time factors for evaluation

In the introduction of this chapter, we have already seen that there is a variety of different mechanisms for this evaluation task. I will use a very simple method in order to show the possible potential of the structure of this system. By inclusion of the

handling of the aspects described above, I use the following structure for the evaluation, shown in Figure 38.

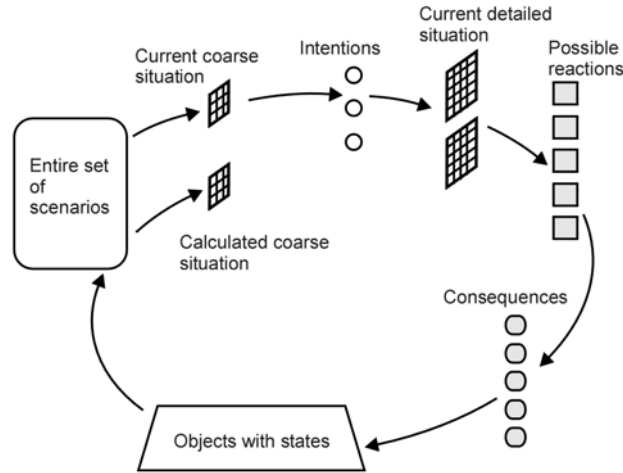


Figure 38: Evaluation cycle

This evaluation cycle is based on a weighting of the identified scenario. Each scenario contains a priority, which indicates the importance of it. The more dangerous a scenario is the higher is the priority of it. In the best case, meaning everything is ok, the weighting is zero. The evaluation in Figure 38 starts with the identified current, coarse situation. Via the intentions of this scenario we get at first a more detailed description of the situation and in the next step a set of possible reactions with defined consequences. Now we use these consequences to change the states of the objects in a “virtual manner”. That means, by using the consequences, we obtain a representation of the environment as it would be after a reaction – the system “imagines” the environment after reacting. Now the system has to find the scenario which would take place based on this imagined environment. We will attain a new situation which is not real but purely calculated. It represents the fifth possible state of a situation as shown in Section 3.2.4. In the last step, we only have to compare the weightings of the identified scenarios. If the priority of the virtual scenario is less than the priority of the real one, it would mean that we have found the right reaction. One can continue this cycle in order to obtain also calculated, detailed situations. However, this step is not necessary since the importance of a situation can be already determined in the coarse stage.

Again we can include the considerations about the aspect “quality versus urgency”, described above. However, we will finally have a proper reaction, and are able to continue to the last step, the realization of the chosen reaction.

3.4 Reaction in the real world

One of the most important factors of the realization of reactions in the real world is the factor of time. It will take some time until the system will detect effects depending on its actions. Thus, although the system has already initiated a reaction to a situation,

it will still perceive the same situation the next time. For example, if the system wants to adjust the brightness by using the blinds: first it will need some time to rotate the lamellas and, second, it will need some time until the brightness sensor detects the change.

Thus, we have to prevent the system performing the same reaction once more in the meantime. To overcome this problem, we can use the time factor for the reaction, as described in the previous section. The system, or better a part of it, has to wait for a certain time span before it can expect a result and may perform the next reaction.

Now the question arises which part or which task has to wait. Our structure offers three possible stages where the break can occur: the selection of the scenario, the selection of an intention of a scenario, and the selection of a reaction. These are the three main tasks in performing a reaction. In Table 4 we compare the different aspects offered by using one of them.

By comparing the advantages and the disadvantages of these three possibilities, the usage of the intentions for the break offers the best result. On the one hand, we are able to handle a scenario completely; on the other hand, we will not have the problem with the overlapping of reactions. If there is no further intention left, the situation is already completely handled, and the system can start to treat the next situation.

Now we will move to the next step. Let us assume, an intention has been blocked for some time and this time is now over, and let us say that nothing has changed. There has been no or only a slight effect by the reaction. The situation is still unchanged, thus, the system has to “know” that it makes no sense to perform the same reaction again if it recognizes the same situation once more. For example, there are two equivalent lamps in a room and the system is free to choose between them. It switches on the first one, but the lamp does not work. Next time, the system identifies the same situation again and now it should know that it has to activate the second lamp.

However, if we contemplate this process carefully, we will recognize that the situation will not be the same in every detail. One lamp was activated (Figure 39.1), and we will therefore get a new situation. That means, even if there is a choice between several reactions – by taking one of them, the situation has already changed and the system will not encounter the exact same choice again (Figure 39.2)

Table 4: Comparison of the main parts of the reaction performing

	Advantages	Disadvantages
Scenario	As explained in Proposition 2, there may be more than one scenario in a sequence of global situations. If one scenario is blocked, the system has the resources to handle another scenario.	A scenario may need several intentions to be handled completely. Therefore, there can be a change to a new scenario, though the last scenario is not yet finished. If it is a dangerous situation, the system may change to a non-critical scenario without a complete handling of the dangerous one.
Intention	There may be more than one intention of a scenario. Thus, if one intention is blocked, the system is able to handle	An intention may lead to several reactions. By blocking an intention, we also prevent the system from looking

	the next one.	for a better reaction than the actual one.
Reaction	In Section 3.3 it is explained that there may be alternatives to the chosen reaction, possibly better and more efficient actions. By blocking a reaction, the system is still able to start an alternative action immediately.	Since only a reaction is blocked, the system would start with another reaction at once. However, there may be an overlapping of these actions, too many actions for a specific task – possibly some of them are not necessary and have to be undone. For example the scenario of energy management: if the system is able to close the fridge by itself, it makes no sense to inform a user, just because the system did not wait for the closed fridge.

Moreover, we can make an additional extension at this stage: after the time factor of the blocked intention has expired, we compare the effects in the real world with the predefined consequences of this reaction. By that, we are able to detect abnormal behavior (for example if a lamp does not work), and also wrong or insufficient definitions of consequences.

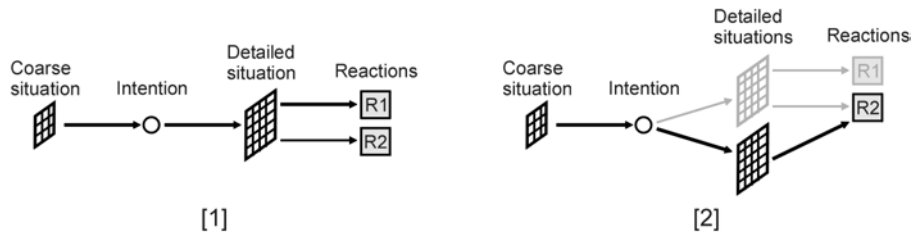


Figure 39: Re-handling of a situation;
[1] Selection of one reaction, [2] Changed situation

WAY OF PROCESSING

In order to realize a reaction in the real world, the data have to be passed down the designed structure of the system. The processing way we use forms the real world until the reaction choosing has used the following steps: Perception – Transformation – Perception Layer Interface – Representation Layer – Representation Layer Interface – Recognition of Scenarios – Reaction choosing. We can take a shortcut for the information flow for the processing in the opposite direction. Naturally, we can skip the task of situation recognition. Also the representation layer can be omitted since by changing the internal representation, our situation recognition would be based no longer on the real world. Thus, we can hand the reactions directly to the *Perception Layer Interface*. As already described in Section 3.1.3, this interface offers the necessary capabilities to receive the data and pass them to the transformation task. Since the system operates with symbols (Section 3.1.2), the values have to be retransformed into real world values. Here, the same mechanisms are used as described in Section 3.1.2. Finally, the values are passed on to the underlying physical structure.

3.5 Conclusion: The entire model structure

By now, we have treated the complete information flow beginning with the perception of the environment, followed by the transformation into symbolic values, the internal representation, the recognition of the current situation, the choosing of a proper reaction to that situation, and finally the realization of that reaction. Figure 40 shows the final structure with all necessary tasks and their assignment to the particular layers.

The following will summarize the several steps of the resulting model of a system with situation-dependent behavior.

PERCEPTION LAYER

In principle, the *Perception Layer* itself can be divided into two layers: the *Real World Perception* and the *Transformed Perception*. First, values about the environment, the real world, have to be collected. A large number of different sensors has to be used to acquire this data. Via several steps, different types of information will be merged. Next, the data may go back to an actuator in order to realize a reflex action, and/or the data is passed on to the *Transformed Perception*. The information is passed on via particular interfaces, depending on the chosen technology. In case of fieldbusses, OPC can be used for this connection. In other applications, such as the computer vision system chosen for this work, an equivalent interface to the higher layers of the system has to be created. These interfaces represent the border between the physical devices and the logical part of the system.

Since the “higher” processing is based on a homogeneous representation of the information, the information is transformed into symbols in the *Transformed Perception*. There are two additional tasks of this layer. First, we have to complete the trans-sectoral processes, which could not be handled within the physical part because of the different nature of the information. And second, some of the symbols obtain a priority, which indicates their importance in the current environment.

Via the *Perception Layer Interface*, the perceived and transformed values are provided to the next layers.

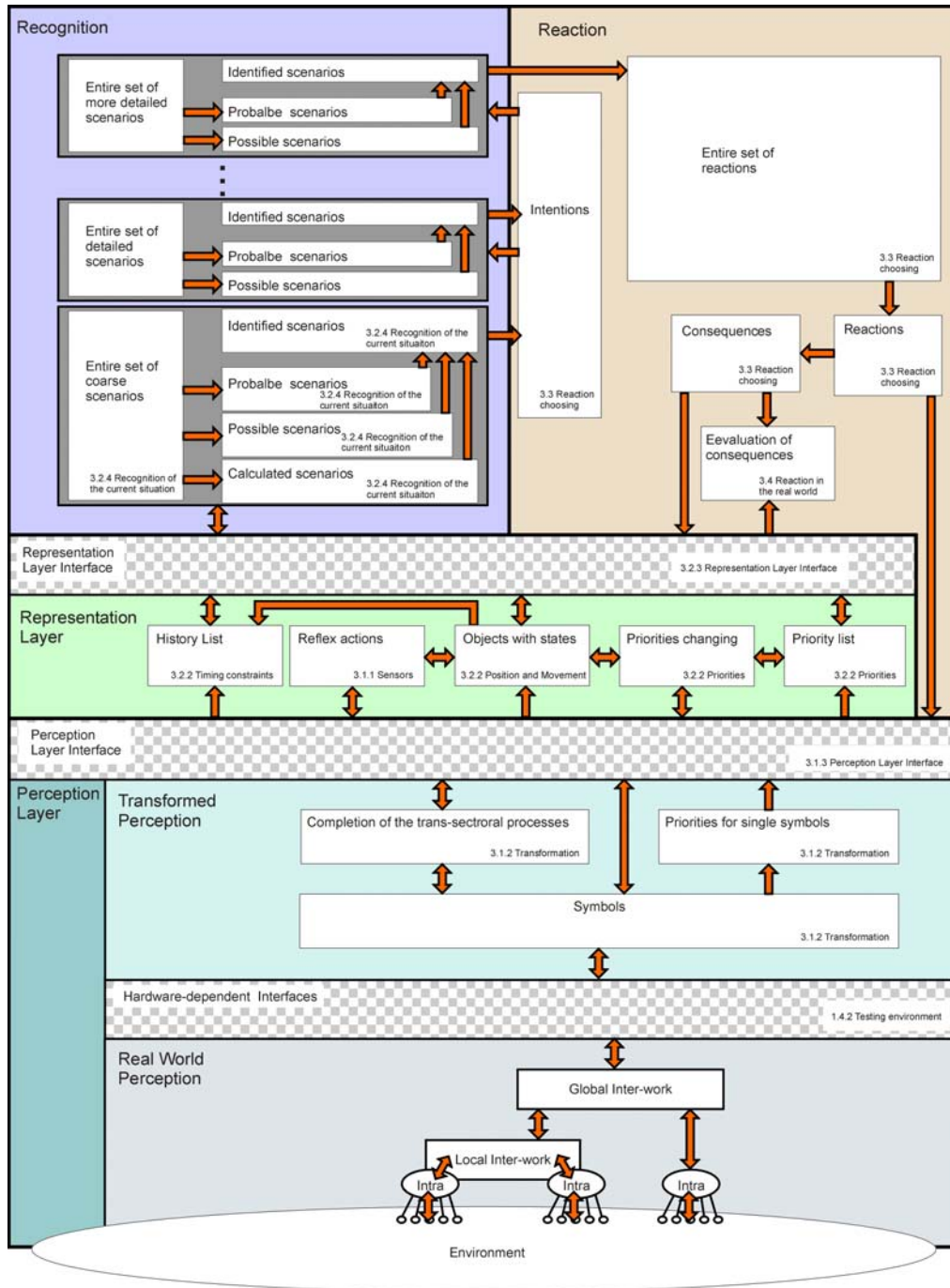


Figure 40: Resulting model structure

REPRESENTATION LAYER

In this layer, the information perceived by the *Perception layer* has to be organized and an internal representation of the real environment has to be constructed. In general, the representation consists of three parts:

1. The surroundings are described by a structure of objects with attributes.
2. The history of changes, events, etc. has to be stored.
3. In order to focus on important aspects of the environment, the system holds a list with all high-prioritized objects.

Furthermore, there are two more tasks in this layer which are, however, not directly related to the internal image.

The first one deals with priorities: a combination of objects may change their importance in the current situation. Just an open window will be less important than an open window together with rain. Therefore, we have to analyze in this layer if specific combinations of objects occur at the same time.

The second one has to inform the system about reflex actions. It is necessary to know which parts of the system are involved in a reflex arc. Since this kind of reaction is handled only by the *Perception Layer*, we have to assure that the higher layers receive knowledge about them in order to be able to consider this knowledge in the next planned reaction.

Again, an interface, i.e. the *Representation Layer Interface* offers the organized information to the higher layers.

RECOGNITION LAYER

Based on the internal image of the surroundings, in this layer the system has to identify the current situation. It possesses sets of predefined scenarios and situations in a varying level of detail. By using the Priority List of the *Representation Layer*, a preliminary selection out of the entire set of predefined scenario can be done. This selection contains only very coarse descriptions of scenarios. Now the system uses the possible intentions (provided by the *Reaction Layer*) for these coarse scenarios, and performs by doing so a preliminary selection of detailed scenarios. This step can be executed several times, depending on the complexity of the scenarios. During these analyses, different states of scenarios such as *calculated* or *possible scenarios* can be used.

Finally, we have an identified situation which has to be handled by the next layer.

REACTION LAYER:

One part of the reaction, the intention, has already been dealt with in the *Recognition Layer*. Thus, there is no clear separation of these two layers.

However, we have a scenario and most of the time several possible reactions to this scenario. Hence, we have to find the right one of them. In an evaluation cycle, the consequences of a reaction are applied to the internal representation. By that, the task of situation recognition starts again and identifies this time a virtual situation. This mechanism allows us to simulate a reaction and assess the situation after a reaction. If the evaluation offers a satisfactory result, the reaction has to be realized in the real world. It is passed down the hierarchy to the *Perception Layer Interface*, is re-transformed from symbols into real world values, and transferred to the appropriate devices.

Additionally, we have the task of evaluating the consequences in this layer. That means, after the reaction has been realized and we can expect the result of the reaction, the predefined consequences are compared with the real consequences

shown by the internal representation. This evaluation may show differences between the prediction of consequences and the real effects as well as malfunctions of devices.

Chapter 4

Realization

*The only limit to our realization of tomorrow
will be our doubts of today.*

Franklin D. Roosevelt

In this chapter, a first realization based on the model structure constructed in the previous chapter is presented. This realization should show the usability of the model structure in principle and, due to that, offer a pattern for further applications.

In the next sections, the four scenarios presented in Section 2.1 will serve as testing examples: the system will be installed in the Smart Kitchen and has to identify these scenarios under real world conditions.

4.1 General constraints

The overall system consists of several independent applications which are able to work in parallel. By that, there is no need to wait for the completion of a specific task; for example, the *Perception* does not have to wait for the analysis of the previously collected information by the *Recognition task*. However, in some services a situation may occur where the operation by one application cannot be completed until an operation by another application has been performed. This can happen when some tasks are producers and others are consumers – the consumers may have to wait until a producer has supplied the information in question. For example, the *Representation Layer* has to build the structure of the internal image, and the *Recognition task* has to use this structure. Thus, it must be assured that there is no busy waiting of the consumer task. A detailed description of the handling of that problematic can be found in [Cou98].

COMMUNICATION

The communication between the independent applications uses *Sockets*, the threads within one application communicate via *Named Pipes*. Reasons for these communication techniques are given in [Kha01] and [Mic01]. It is explained that in a fast local area network (LAN), Sockets and Named Pipes clients are comparable in terms of performance. However, in slower networks such as wide area networks (WANs), the performance difference between these technologies becomes apparent. This is due to the different ways of interprocess communication (IPC). Network communications are typically more interactive in Named Pipes. One process does not send data until another one asks for it. A network read typically involves a series of messages which can be very costly in a slow network and cause excessive network traffic. However, if the processes are running locally on the same computer, the local Named Pipe runs in kernel mode and is extremely fast. In case of Sockets, data transmissions are more streamlined and have fewer overheads. In general, Sockets are preferred in a slow network environment, whereas Named Pipes can be a better choice working locally as they offer more functionality, ease of use, and configuration options.

DATA PROCESSING

Because of the large number of information that has to be handled, and the high performance required for that task, a database is used as memory of the system. Although today's database systems offer a fast handling of large data amounts, this work can be assisted by considering aspects such as the data types which have to be stored and processed.

Analyses of the processing of the application have shown that more than half of the requests onto the memory are used for lookups:

58%	searching for information
18.7%	updates of stored values
15.5%	inserting of new data
7.8%	deleting of information

Hence, the performance for lookups concerning a specific data type will have a significant influence on the overall performance of the application.

For the database chosen in this work, (MS SQL Server 7.0), a series of tests was done in the same environment where the final application will have to work in order to determine the best data type. One query of each type (select, update, insert, delete) was executed many times on a table containing 1000000 records. It was used with always one of 8 different data types offered by the database. The average time needed for each request was calculated on the basis of these test series.

The following charts show the results of each type of request. The x-axis shows the different data types used for the test. The y-axis, which represents the time, is without numbering. The tests were not performed in order to compare the four types of requests but to compare the different data types within one query.

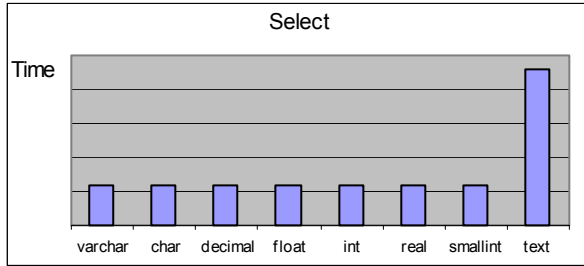


Figure 41: Select

The search itself was done in several different ways: by using "=", "<" and "<>".

For example:

```
SELECT id FROM test
WHERE integer = 100
```

In case of the type *text* the comparison was done with *like*:

```
SELECT id FROM test
WHERE text LIKE 'word'
```

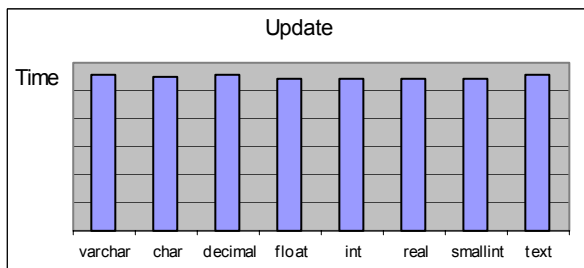


Figure 42: Update

The request to update a record was the most expensive one of the four types. The reason for that is that there must be a search and an update or even a delete and an insert as explained in [Kli99].

For example:

```
UPDATE test SET integer
= 99 WHERE integer = 100
```

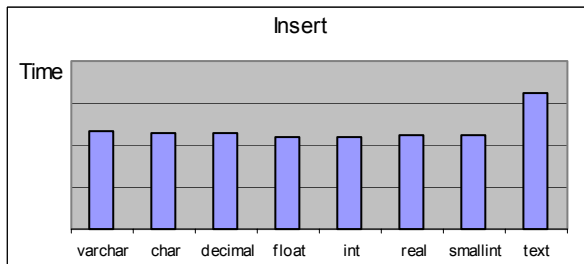


Figure 43: Insert

The insert command appends a new, empty record to the table and inserts the values in this record.

For example:

```
INSERT INTO test VALUES
(100)
```

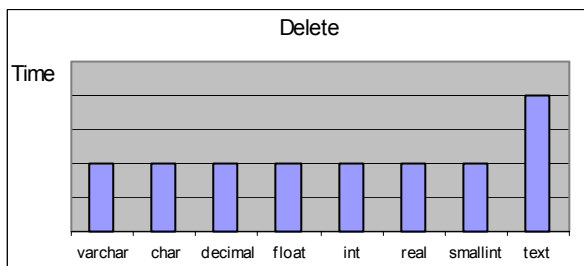


Figure 44: Delete

Similar to the update, the delete requires a search at first (except of deleting the entire content).

For example:

```
DELETE FROM test WHERE
integer = 100
```


With the exception of *text* each of the types is usable; *int*, *real* and *smallint* show a slightly better performance. Therefore, integer values are used as standard types in this work.

DATA STORAGE

Figure 45 shows the memory structure used in the aimed system. The overall application (shown as Situation-dependent Behavior Module) is supported by a variety of different memory modules.

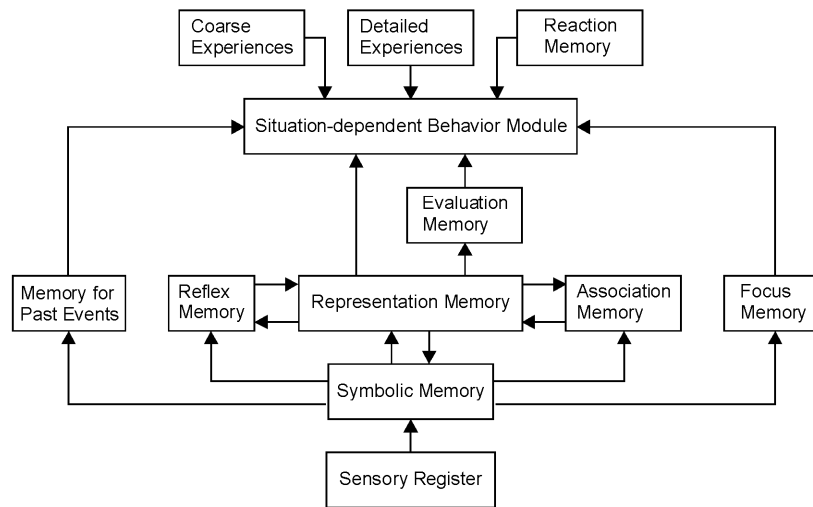


Figure 45: Memory structure

At the bottom, the memory starts with the Sensory Register in analogy with Atkinson and Shiffrin [Kie99]. It consists of the memories of single devices which store the incoming stimuli only for a very short time. The Symbolic Memory holds a set of possible symbols, which are used instead of the real-world values for processing. This memory supplies data to five further memory modules: the Memory for Past Events, the Memory for the descriptions of Reflexes, the Representation Memory for describing the environment, the Association Memory for storing relations between objects, and the Focus Memory for important features in the surroundings. While Reflex and Association memories are connected to the Representation, the other three modules are used by the overall system in order to recognize the situations. Beside the direct connection between the Representation memory and the Situation-dependent Module, we have another connection with the Evaluation memory placed within. This memory is used for “virtual”, imaginary information necessary for the evaluation cycle. Finally, three more memory modules, two for stored experiences and one for reactions, are needed to complete the Situation-dependent Behavior.

EXTERNAL DESCRIPTION OF THE ENVIRONMENT

A factor which has been more or less disregarded so far is the way how the system obtains the necessary knowledge about the environment. In Section 3.3 I have mentioned two ways in order to receive this information: either someone tells the

system everything, or it has to find out everything by itself. Whereas the autonomous exploration of the environment is desirable in many cases, in problematic situations a supervisor is required. For example, it would not be reasonable if the system analyzes the different states of fire-fighting equipment or it will not be able to create messages by itself.

Moreover, one has to consider that the autonomous exploration will take some time. This is the reason for using a predefined description of the environment for the testing phase. Due to the large number of changes during the testing phase, it would not be manageable to wait for an analysis by the system. Hence, this prototype makes use of an external description which defines every detail of the surroundings. This description has to contain the following elements:

- Objects with their attributes
- Possible symbols for a specific attribute
- Symbols for a specific object/attribute combination
- Ranges of real-world values for symbols
- Which inputs are connected with the same objective
- Compositions of trans-sectoral functions
- Compositions of combined objects
- Compositions of situations and scenarios

The external description of the environment is explained in more detail in [Fal03].

4.2 Perception of the environment

At the start of the project Smart Kitchen, a LonWorks network has been installed. Thus, the first realization is oriented mainly towards this fieldbus technology. Nevertheless, the integration of additional systems has been already considered.

The task of perception is handled by two applications: the transformation of values into symbols and the preprocessing of these symbols. The applications are connected via Sockets, shown in Figure 46.

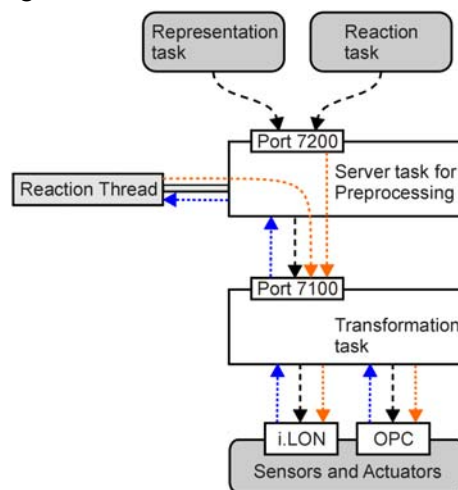


Figure 46: Structure of the Perception Layer

As interface between the fieldbus and the transformation task running on a PC, the i.LON web server and OPC are used.

TRANSFORMATION

The transformation task is a single application that communicates on the one hand with the server task for preprocessing, and on the other hand with the underlying sensor and actuator level.

In Section 3.1.2 several concepts of the construction of symbols are stated. Following biological concepts, the autonomous creation of symbols such as by neuronal networks seems to offer a high potential. Nevertheless, in this implementation, a static assignment of input values to symbolic representations is used. The reason for this choice is the possibility to adapt a static symbolizing very fast to changed conditions – no learning time or training time is necessary. Since this realization has to deal with a relatively small number of input information, and the conditions changed several times due to different testing assemblies, the static assignment seems to offer the best prerequisites for the prototype.

For each object/attribute combination a specific pool of symbols is defined, whereas each symbol is used for a specific range of real-word values. The range taken for one symbol depends on the device and the usage of it. For example, for the attribute *temperature* we use the symbols *cold*, *cool*, *lukewarm*, *warm* and *hot*. In case of inputs by a temperature sensor in a living room, we use the following ranges: up to 17°: *cold*, 17° – 19°: *cool*, 19° – 22°: *lukewarm*, 22° – 26°: *warm* and above 26°: *hot*. In contrast, for the plates of a stove we use *cold* for up to 25°, 25° – 45° is *warm* and above 45° means *hot*. Thus, not all of the possible symbols for an attribute have to be used for a specific input in a specific environment, and the range of values of a symbol is also varying.

This means in the overall system that if we want to integrate a new device, the definitions of the transformation have to be done first. By this adaptation we are able to use every possible device in our system.

Together with the symbols, the static priorities described in Section 3.1.2 are assigned to the object/attribute combinations.

PREPROCESSING

In the task of first processing of the symbolic inputs, a server task in combination with one thread is used. The server task is responsible for the communication with other applications and the thread, and has to deal with the symbols perceived by the transformation task. The connection between server and thread is handled via a Named Pipe, while the communication with other applications is performed with Sockets.

Server task

Based on the symbolic values, first processing is done in this task. This is due to the fact that one information point in the environment is perceived by several sensors. Once a change in the inputs is detected the system looks for all data sources concerning the information point of the changed input. For example, the system measures the temperature of a plate by using a camera (S1) and two different temperature sensors (S2 and S3). The camera offers the value *hot* if the plate is red,

otherwise it states the value *cold*. The first temperature sensor has the values *hot*, *warm* and *cold*, and the second one *hot*, *warm*, *lukewarm*, and *cold*. If there is a change in the camera from *cold* to *hot*, the system looks for other sensors concerning the temperature first. In this example, it will find the two temperature sensors S2 and S3. If we assume, S2 has the value *warm* and S3 measures *lukewarm*, we have three independent sensors measuring the same information item but offering different values (maybe there is a malfunction or the sensors have a too long reaction time and offer already outdated values).

In this realization, a method following the ideas of Fuzzy-Logic is used to deal with the situation [Sch98a]. A value of one sensor correlates not only with the same symbol of another sensor, but with a range of values. The range of one state of the camera can be calculated as following: the number of possible states of S2 divided by the possible states of S1. The result is interpreted in the way that *hot* of S1 includes 100% of *hot* of S2 and 50% of *warm*. Figure 47 shows the ranges of all values in this example.

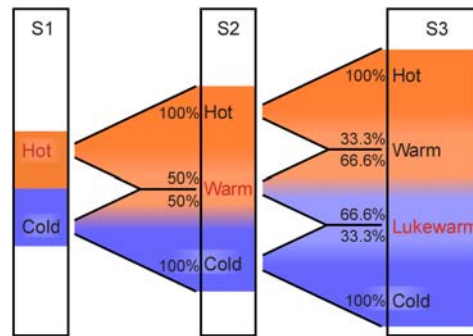


Figure 47: Exemplary handling of different sensors

In the given example, where $S1 = \text{hot}$, $S2 = \text{warm}$ and $S3 = \text{lukewarm}$, the correlation of *hot* (S1) in S2 is calculated at first. The first half of *warm* is within the range of *hot* of S1 and therefore, *warm* in S2 is valid for *hot* in S1. Next, the range of *warm* (S2) in S3 is calculated. The range includes $\frac{2}{3}$ of *warm* and $\frac{1}{3}$ of *lukewarm*, which is valid too. However, in this calculation we have to consider the fact that only the first half of *warm* in S2 is available. Consequently, only the first half of S3, which means the $\frac{2}{3}$ of *warm*, is valid. Therefore, the combination of the states in the example would not be true. In the first realization, these cases are stored in an external file; however, there is not yet further processing of them implemented. Nevertheless, this stored information can be used to detect problems in the sensory system. For example, if a sensor is logged in this list very often, it would be a sign of either a defect of the sensor or the wiring respectively, or the processing time of the device is too long for the aimed system.

Now, in order to analyze the other case, let us say, the value of S3 is *warm*. This time, *warm* in S3 is valid for *warm* in S2. By that, we arrive at the next advantage of this preprocessing: the reduction of redundant information. Since all three sensors show the same measurement result, it is sufficient to pass on only one value. The chosen value is the one of the sensor with the largest number of possible states, since it is the most exact one. In this case, S3 has four possible states and the result is therefore *warm*. By passing on the value, a further abstraction takes place: until now, we have

dealt with values of sensors; now we have reduced these sensory values to the state of the temperature of the plate! For the higher layers it is completely transparent where the values come from – they just deal with states of objects.

Furthermore, reactions are handled in this application. There are two possible kinds of reactions: performed by the higher layers or the completion of trans-sectoral processes. The former kind is simply passed on to the transformation task. Reactions produced by the higher layers are already considered and defined in very detail, and can be directly handed over to the transformation in order to “retranslate” the symbols into real-word values. The completion of the trans-sectoral processes is treated by the reaction thread.

Reaction thread

The reaction thread uses a list of definitions of possible trans-sectoral processes. This list consists of rules that describe actions without influence of the higher layers. Since these reactions and their causes may contain most varying devices, which are therefore not able to work together on the hardware level, we have to complete them after the transformation task. For example, if we want a camera which detects a problem to cause an optical alarm by switching on a light, it will be difficult to realize this control loop since a different communication is used. However, the same language is used after the transformation task, and they can thus be combined.

In parallel to the reaction thread, the transformed inputs are passed on to the higher layers in order to enable a comprehensive processing as well.

4.3 Representation of the environment

The representation of the current environment is a central part of the entire application. On the one hand, it has to reflect the real word as correct as possible. On the other hand, it should extract additional information (for example priorities) and offer them to higher layers in order to enable a faster processing.

STRUCTURE OF THE REPRESENTATION LAYER

The representation task consists of a server and three client threads, shown in Figure 48.

Server

The server is first of all responsible for the communication with other applications. It contains a Client Socket for the *Perception Layer* and a Server Socket for *Recognition* and *Reaction Layers*. For the communication with the client threads it uses Named Pipes. New values of the layer below are passed to the appropriate client threads via these pipes.

Concerning the connection to the higher layers, it is listening at a specific port and is able to handle multiple connections. A unique number is assigned to each connection when an application sends a `register()` command, and by using this number, the client is served.

Object thread

The main client thread is called object thread. It has to build and maintain the internal structure of the environment. It receives all changes of the surroundings by the server and updates the internal states. Beside that, it is supported by information received from two additional tasks: Reflexes and Associations.

Concerning changed states in the internal image, the need arises to check whether the change has been caused by a reflex action. Furthermore, we have to analyze whether this new value in combination with other already existing states of objects might cause a change of the priority.

In Section 3.2.2, two methods are stated for assigning priorities to combinations of objects: either the priorities are assigned directly to every single object, or to the combination itself. In this realization I have chosen the latter method. Using the Association Memory, combinations of objects may result in a new combined object. Let us assume we have a building with 30 windows and our system has to be able to detect a scenario where a window is open when it starts to rain (remember: we are using coarse and detailed situations). There are several ways to deal with this situation. For example, we can define one scenario of each window – if we use the same method for open doors, for lights that are switched on, for each device etc., we will obviously obtain a very large number of scenarios, and the performance benefits of using coarse and detailed situations are lost. To overcome this problem, we can use a controller node in the fieldbus level, which uses a combined value of the 30 windows. In doing so, we can have a coarse situation which only says that there is an open window, and in the next step, the system can deal with this situation in more detail. The combination of objects in the *Representation Layer* is based on the same idea: it is used for all sensory inputs where it is not possible to integrate controller nodes for combined information.

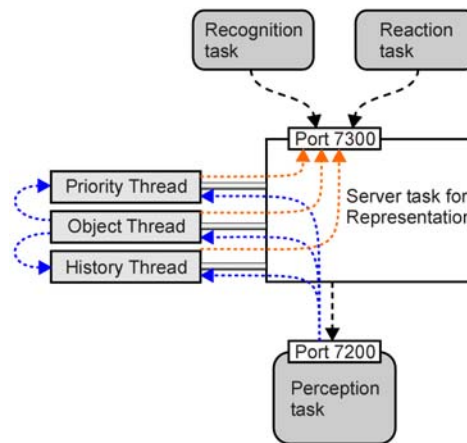


Figure 48: Structure of the Representation Layer

Priority thread

The changes offered by the *Perception Layer* are also perceived by the priority thread. With each change, the priority is checked, and the thread analyzes whether the information item is already in the priority list. If so, and if the priority of the new value is above a certain threshold, the entry in the list is updated; otherwise, if it is not

in the list yet, it is inserted. In case of the priority being below the threshold, the item is deleted of the list.

Beside the inputs of the *Perception Layer*, the priority thread is connected to the object thread. If this thread has detected specific important combinations of objects, this information is also inserted in the priority list.

History thread

Finally, the history thread is the last one of this application. Like the other two threads, it perceives the new changes in the surroundings. Additionally, it has access to the current representation of the environment by means of the object thread. The change-event is used to start the storage of a new history record. After it has saved the current states of the internal image, it reduces the previously stored descriptions. In each row, the entries with the lowest priority are deleted – except the one that has caused the storage.

DYNAMIC DATA POINTS

In Section 3.2.1 the demand for dynamic data points to cope with the location of persons is explained. In this realization, I have solved this requirement by using persons as attributes of particular sensor objects. For example, the computer vision system which is responsible for the identification of persons offers the attribute child. This attribute indicates that there is a child present in the room – information that is necessary for the scenario concerning safety in Section 2.1.

EVALUATION CYCLE

Section 3.3 describes the evaluation of effects of chosen reactions: predefined consequences of actions are “added” to the internal representations in order to recognize the following situation in the next step. Since this evaluation must not conflict with the current representation of the real world, I have decided to use a second representation in parallel. Before starting an evaluation, this second image is updated according to the main representation. While this additional image is used for the identification of a virtual, future situation, it is evident that we can operate the history list for the identification of a virtual scenario.

As a consequence of the usage of a second description, we also have to employ a second priority list, which is only based on the states of the virtual representation. A change in this list causes the *Situation Recognition* to start an analysis based on the imaginary representation. In this process, it operates the command `get_EvaluationData()` and `get_EvaluationPriority()`. By that, the active part of managing the evaluation is the *Situation Recognition*. It knows about the evaluation and asks explicitly for these values. Another possible method would be to pass the control to the *Representation Layer*. Thus, we could use the command `get_data()` for the evaluation cycle as well. However, since the recognition task has to be informed whether it is just a simulation or not (it has to pass its results to the recognition task either as real situation or as imaginary one), the command `get_data()` has to be extended by a flag in this method.

4.4 Recognition of the Situation and Reaction

The task of *Situation Recognition* reacts to changes in the priority list: it makes a recalculation of the current situation by using a memory containing predefined images of situations. Furthermore, since the recognition of the environment is connected to the task to react to it, the two tasks are managed by one overall application.

STRUCTURE OF RECOGNITION AND REACTION

The structure is similar to the one of the *Representation Layer*. Again, we have a server and three client threads, as shown in Figure 49.

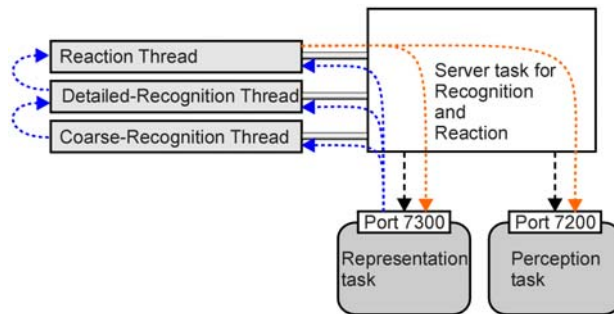


Figure 49: Structure of Recognition and Reaction

Server

The server is again responsible for the communication with other applications. This time it possesses two Client Sockets. The first one is connected to the *Representation Layer*, which is listening at a specific port (7300) – the command register() establishes the communication with this task. The second Socket communicates with the *Perception Layer* via the port 7200. This connection is used for passing changes of states to the real world.

The server deals with three client threads by using Named Pipes for the communication. As explained in Section 3.2.4, the identification of the situation can be done in several steps. At first, only a very coarse situation is identified, and with each step an increasing number of details is added. I have decided to use two steps according to the relatively low complexity of the exemplary scenarios of Section 2.1. Even the usage of two processes means an overhead for a small number of scenarios – they could easily be recognized in one step. However, the realization is meant to show the usability of the ideas behind, and therefore I use two threads for the recognition in order to realize the concept of coarse and detailed situations.

Coarse-recognition thread

Figure 50 shows the structure of the data storage and processing for the recognition of coarse situations. The entire process is started by changes in the priority list. Using the command `get_priority()` the content of the list is passed to the coarse-recognition thread. This information is used to make a preselection of possible situations: coarse situations that contain elements of the priority list are selected. In the next step, these possible situations are analyzed by using the representation offered by the *Representation Layer* (`get_data()`). Matches are added to the list for identified coarse

situations. In Section 3.2.4 five different kinds of situations are stated. In this realization I have chosen two of them: the possible and the identified situations. A further differentiation between them would mean an unnecessary overhead in this application field. Once we have identified situations in the current environment, we have to look for appropriate scenarios: all coarse scenarios which contain at least one of the identified situations are added to the list of possible scenarios. For the identification of the right scenario, we use the history list of the *Representation Layer* (`get_history()`). Finally, the identified situations together with the identified scenarios result in conforming intentions.

Since the threads are able to process in parallel, the system tends to deal at first with important situations and scenarios in order to offer them to the next thread as soon as possible. Consequently, situations as well as scenarios are equipped with a priority that indicates the importance for the environment. Concerning the priorities, the classification of Section 3.1.2 is used.

Additionally, a second order for the situation list based on, for example, the number of elements which are found in the priority list can be used.

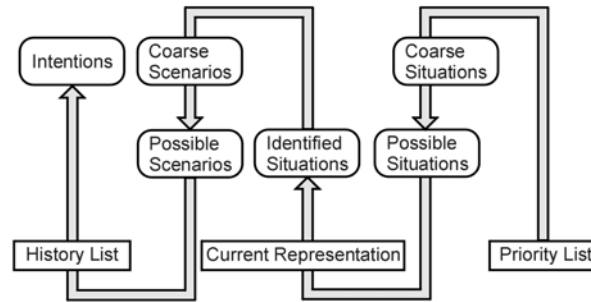


Figure 50: Recognition of coarse situations

Detailed-recognition thread

The intentions concerning the coarse situations launch the recognition of detailed situations. As shown in Figure 38 in Section 3.3, intentions are the connections between situations of varying detail level.

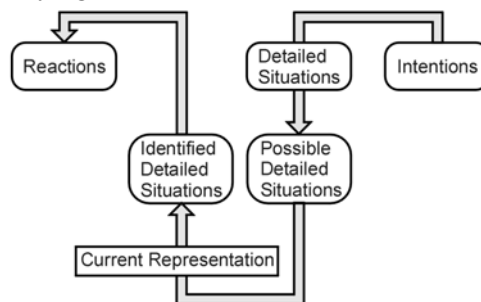


Figure 51: Recognition of detailed situations

Thus, intentions result in some detailed situations, which are added to a list of possible detailed situations (shown in Figure 50). Analog to the coarse situations, the current representation is analyzed in order to identify the possible ones. Next, the

system could analyze detailed descriptions of scenarios by using the history list. However, the results of the realization have shown that in the current testing environment it is sufficient to base on the identified coarse scenarios. The additional step concerning detailed descriptions would offer a higher accuracy of the recognition task; however, the requirements of the application field have to be considered and the adaptation of the processing must be in accordance to that. Therefore, this realization uses the identified detailed situations in combination with the identified coarse scenarios, and obtains by that some possible reactions.

Reaction thread

Based on the identified detailed situation, the system can find several possible reactions. With each reaction there exists a detailed description how to realize it. For example, if the reaction has to switch on the light in a specific room, it is explicitly defined which object and symbolic value it has to use for that. Additionally, the consequences of an action are predefined in each item of a reaction. If the reaction says to switch on two lights in a room, for each of them the consequences are given. Each of these actions results in a change of the state of the switch and in a change of the brightness in the room. Furthermore, consequences can be absolute or relative: the switch will be in the state “on”, while the brightness will increase in some degrees. These definitions are necessary for the evaluation cycle. The system takes the first possible reaction, looks for all consequences of it, and updates the representation of the evaluation with the consequences. In Section 4.3 I have explained that there is an additional internal representation of the environment. When starting the evaluation, in each cycle this second image is brought to the actual states of the main representation. The same is done in the additional priority list. If the change in the consequence is absolute, the value is written in the appropriate state of the image. Otherwise the state is increased or decreased by the given degree.

When all changes are handed over, a new analysis of the current situation starts – this time based upon the imaginary description of the environment. After two steps of recognition (in this realization), the detailed-recognition thread presents an identified, imaginary situation. Since situations possess a priority describing their importance, the priorities of both situations, the real and the imaginary one, are compared. If the new priority is less than the real one, it means that the reaction would improve the current situation. Otherwise the reaction is rejected, and the cycle starts again with the next possible action.

Once a proper reaction is found, all changes necessary for it are passed to the *Perception Layer*.

Chapter 5

Discussion

*What we call results
are beginnings.*

Ralph Waldo Emerson

The final realization described in the previous chapter, and the four exemplary scenarios in Section 2.1 are the base for a comprehensive analysis.

I will use a similar subdivision as in the Chapter Realization and discuss the main tasks of the entire system, the perception of the environment, the representation, the recognition of the situation and the reaction choosing in separate. Finally, the behavior of the overall system is analyzed.

However, it must be considered in this discussion that, due to technical restrictions, some of the results are obtained only by simulations.

5.1 Testing sequence

During the testing phase, all elements required for the four scenarios took place. Since not all of the demanded information was offered by the sensory system, some of the information sources were simulated.

- Person identification: In the scenario of safety, the identification of a child is demanded. Due to problems with this recognition, the information is simulated.
- Building: In all four scenarios more than only one room is involved. For example in the scenario of security, the entire building is included, while in the scenario of energy management the system must look for someone in the adjoining room. However, the testing environment has been limited to one room.

- Breaking window: In the scenario of security a breaking window is detected. Again a simulated sensor has to supply this information.
- Number of sensors: Since the number of data points is relatively small for the desired system, the total number has been virtually multiplied. Each value perceived by the environment is passed to the system several times. By that it has been possible to test the application with a larger amount of data.

The testing sequence itself is in a way that the light-cycle of day and night has been maintained. A test subject is entering the room several times under varying lightning conditions. The person is walking around and after some time leaves the room. For the demands of the energy management example, the person is opening (and closing) the fridge several times. Beside that, actions like opening the window, opening a cupboard, switching on the stove or other appliances have taken place. The phone is ringing at a time when the room is empty and when the test subject is in the room. As already mentioned above, information about the child, the breaking of a window and information concerning other rooms are simulated.

5.2 Perception of the environment

The perception is based on information of the following devices:

- 16 contact sensors for the door, windows, appliances, furniture, etc.
- 12 relays for different electrical devices like lights, stove, blinds, microwave, etc.
- 1 humidity sensor
- 1 dimmer
- 4 electricity meters
- 3 cameras
- 3 brightness sensors
- 3 smoke detectors
- 8 infrared-sensors
- 4 water counters
- 2 occupancy sensors
- 1 switch for blinds
- 4 valves for the water pipes of the sink unit, dishwasher and coffee machine
- 12 water sensors
- 13 controller nodes
- 1 distance sensor
- 8 temperature sensors

For capturing values of these devices, two different concepts are used: the i.LON web server and OPC.

In case of the i.LON a web-interface is used. For reading values of the physical network a web page containing these variables is interpreted. For writing to the network, the new values are posted to the server.

Both, reading and writing, requires the web server to process a web page. Either the perceived values of the physical network have to be integrated into a page by using specific tags, which, in the next step, have to be interpreted for displaying them. Or, changes have to be posted again by using these tags – next, the server has to interpret them in order to pass them to the concerning devices.

Thus, the communication via the i.LON has offered only a poor performance. This constitutes the reason for the integration of a second interface based on OPC.

[Bur98] describes a variety of performance and throughput tests of OPC. The tests are run where the client and the server are running on the same computer as well as on different physical computers. Moreover, varying numbers of clients are used. For the communication, different variable types have been taken. The results have show that in general performance and throughput are highly dependent on the hardware configuration. For our aimed system the delay due to this interface is negligible. The decisive factors are the way of the values from a device to the OPC server and the processing of this value after the OPC client in the higher layers.

Transformation

As in the analyses of the other tasks of the application, the communication with the database is the major factor for the transformation. The average times for the entire task is between 20 and 190 ms. In case of 20 ms, 17 ms are used for the database; in case of 190 ms the database access needs 184 ms. That means, up to 97 % of the time required for the transformation is spent for lookups in the memory.

Furthermore, the tests show that the internal communication between the different parts of the applications (threads via Named Pipes and applications via Sockets) is negligible – it needs only 1.8 % of the total time.

Trans-sectoral task

In this task, the system possesses a list of possible functions. Each input value has to be compared with the descriptions in this list if it might cause a reaction. The search in this list takes an average time of 25 ms. Although it is easy to get an average value of finding a trans-sectoral function, it is more difficult to determine the time of the entire process of this function. The values have to pass server stages, where the time in each of them depends on a variety of factors. The way of a value from a device to the verification task via the interface is depending mainly on the traffic of the network. In the best case it is about 13 ms. Together with the verification and the transformation the entire processing of a trans-sectoral function takes about 100 ms in the best case up to more than half a second – both results depend on the assumption that the involved devices react immediately.

Verifications

As already explained, in the verification task several input values concerning the same objective are compared. The performance of the analysis described in Section 4.2 strongly depends on the number of values which have to be compared. During the testing phase up to 10 different inputs are involved at the same time, which results in processing times of 15 to 90 ms.

The average number of errors detected by the verification task could not be analyzed during the tests. Since the installed devices have been chosen for exactly this application, their processing times have already been adjusted. Thus, the perceived values are always within valid ranges.

The benefit of data reduction is remarkable, though it has not been a decisive factor. The overall reduction is 1:1.78, which means one abstract object for “1.78” real inputs. The reason for this slight reduction is given in the next section.

5.3 Representation of the environment

The exemplary internal representation is built by 40 objects; in combination with attributes the structure offers 132 information points.

Data reduction

Compared to the number of data points in the real word – which is 208 – we can notice a considerable reduction. This reduction is even more remarkable if we consider that 15 out of the 132 information points are used for combined objects. That means firstly that the real number of abstract points is 117. Secondly, the usage of combined objects will show only little effect in this restricted environment. For example, there are two windows in the testing room, which are combined into one object. Thus, there is just a slight reduction – real benefits will be achieved only with large numbers of objects which are combined by one single object. In case of the used testing environment, it would even have been better to work without combined objects.

Management of the internal structure

Once the structure of the internal image is established, we have to use only *updates* in case of changes in the environment. In Section 4.1 it is mentioned that updates are the most expensive commands for processing the information in the memory of the system. The average time needed to update a change in the internal image is about 20 ms – independent of the size of the structure.

Also the management of the priority list has a considerable influence on the performance of the representation task. In this realization, all elements with a priority higher than zero are stored in this list. By that, the average number of elements in the list has been 25 during the tests. The decision about a threshold value of the priority list has proven to be important for the performance of the entire system. It determines the number of elements in the priority list, and consequently affects the search for scenarios. As already explained in Section 4.4, the priority list is used to sort out possible coarse situations. Thus, a higher number of important elements will result in a more expensive search as well as in a higher number of possible situations. Also the identification of the situation will be more expensive since a higher number of possible situations has to be analyzed.

An increase of the threshold to a value, where only elements with abnormal states, persons and dangerous elements are stored in the priority list, decreases the average number to 6 elements. This reduction has improved the performance of the search for situations; however, the scenario of comfort could no longer be detected since the elements required for the situations could not reach the priority list.

Thus, when increasing the threshold of the priority list, one has to consider that each coarse scenario contains at least one element above this threshold.

Comparing the time needed for the object structure and the priority list, we can see that the management of the list with about 120 ms is 6 times more expensive than the handling of the main structure. A reduction of elements in the list has, however, only slight affects to that.

The usage of a threshold implies another question: what is the reason for priorities which are not able to reach the priority list. Would it not be easier to set all priorities below the threshold to zero? There are two answers to that:

First, the usage of several different priorities below the threshold value guarantees that they will be reduced step by step instead of all at once. It would be better to include even more priority levels in order to slow down the reduction.

Second, we are still dealing with the feature of combined objects: a combination of several states in the environment at the same time might be followed by a change of the priorities of the single states. Thus, an element which has originally a priority below the threshold might acquire an additional importance by such a combination, and might reach the priority list by that. However, this feature has proven to be not a good idea for the small number of elements in the testing environment. This mechanism is mainly useful in combination with the usage of a threshold value higher than zero for the priority list. The idea of this method is to push elements in the priority list which are not yet in the list although they are of importance in the current situation. Therefore, if we use no threshold in the priority list, each element with a priority higher than zero is already in the list. However, in the testing environment of this work, the usage of this feature is more expensive than the renunciation of the threshold value of the priority list. Due to the relatively small number of data points in this test, it is faster to work with a larger number in the list instead of a reduced list but an additional task for changing the priorities.

History list

The assumption of Section 3.2.2 concerning the memory needed by the history list has proven to be wrong. Figure 52 shows the development of the memory needed by a specific entry over the time in the tested examples.

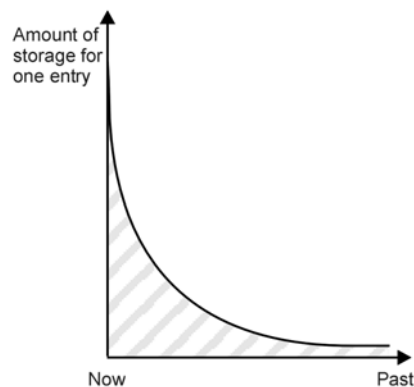


Figure 52: Memory needed in history list

The reason for this change can be found in the distribution of the priorities in the internal image, shown in Figure 53.

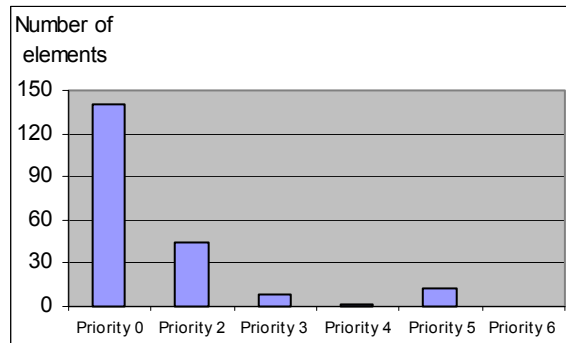


Figure 53: Priority distribution

- Most of the elements of the internal representation are without priority – they are shown as “**Priority 0**”.
- **Priority 1** of Section 3.2.2 is used for changes in the environment. Since these elements will not be deleted by the reduction of the history list, they do not show in this figure.
- Activated devices which show a normal behavior use **Priority 2**.
- Reflex actions according to some of the scenarios are marked with **Priority 3**.
- **Priority 4** is used for abnormal states of devices. In the given examples, it is only used for the breaking window.
- Detected persons have **Priority 5**.
- **Priority 6** marks dangerous elements such as gas or fire. During the testing phase none of these elements occurred.

This result leads to the conclusion that a reduction with every new entry is not reasonable. It would require an evenly distribution of the priorities, which will rarely occur in a real environment. Thus, a new solution to the reduction has to be found. For example, if the task starts only after a specific number of changes, we would be able to level off the curve.

This step would have two more advantages. If the past states are available for a longer time, it will improve the accuracy of the recognition task, since necessary elements could otherwise be deleted. Moreover, the tests have shown that the reduction task takes about 140 ms. This time would increase a little by the additional data in the list, but it would no longer be needed with every change.

Furthermore, it has been revealed that a complete forgetting will be necessary. Even with only one remaining element in an entry of the history list, the list will increase with each change in the environment. Hence, there must be a mechanism to judge how long a value will be needed.

5.4 Recognition of the Situation and Reaction

Although the task for situation recognition is connected to the reaction, we will analyze the two tasks separately when possible. The relation between them is treated later in this section (evaluation cycle).

SITUATION RECOGNITION

A basic concept for the recognition of scenarios is the processing in several steps. The aim of this mechanism is the reduction of information that has to be processed in one step. Instead of dealing with all available data points of the internal image and all possible descriptions of scenarios at once, we do it step by step. An important aspect to take advantage of this mechanism is the usage of parallel processes. Once there is a first result in one step, this result can be immediately transferred to the next stage for further processing. At the same time, the previous stage goes on with its own work. By using the priorities of situations and scenarios it can be assured that the most important elements are processed at first.

Although this is a mechanism which shows its strength in handling of larger data amounts we can already see the benefits with our small number of inputs. In this realization, the first coarse recognition has to analyze only 6 scenarios. If the system selects one of them, it receives at least three possible detailed scenarios for one coarse scenario, which have to be analyzed in the next step. The advantage is obvious: by this method we have to compare (in the worst case) about 9 scenarios in order to analyze the current situation (the six simplified and the three detailed descriptions). If we perform the recognition in one step, we would have to analyze more than 18 ($6 \cdot 3$) detailed scenarios.

Furthermore, a coarse scenario consists of an average of 1.5 situations and a small number of elements in each situation. Thus, it can be processed very fast. By contrast, a detailed scenario consists of approximately 4 situations, whereby the number of elements in one coarse situation to a detailed one is in a ratio of about 1:3.

Beside the confirmation of the concept of processing in several steps, the analysis of the realization has shown three main aspects.

Number of steps

In the same way as we have reduced the 18 scenarios to 9, we could reduce the 9 to, for example, 8 by adding another step (Figure 54). The first search has to deal with three scenarios, each of them can lead to two more detailed ones, and again each of them can result in three detailed descriptions.

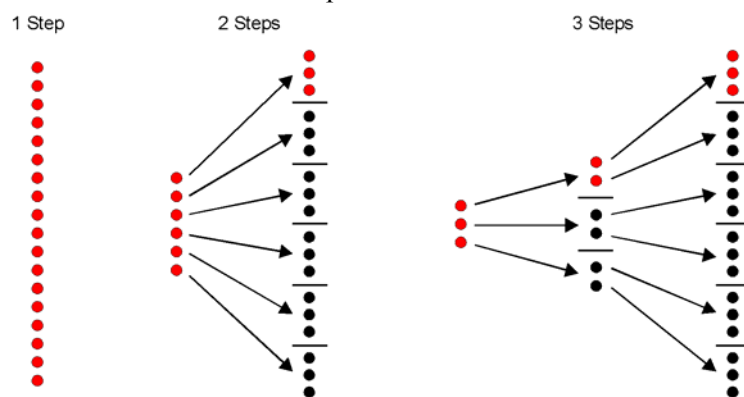


Figure 54: Different number of processing steps

However, the adding of additional steps will not be a long-term solution: so far, we have considered only the worst cases. This means that we were assuming that we

have to compare all possible scenarios at each step since the right one is always the last one. But let us examine the best case: in each step the first scenario we take is valid. If we have a processing in only one step, we have the right scenario with only one comparison. With 2 steps we need 2, and with 3 steps 3 comparisons. Thus, an improvement of the worst case means at the same time a change for the worse of the best case.

Usage of already available knowledge

In the course of this realization, several scenarios are described in detail. Of these detailed descriptions the most important features are extracted in order to form the coarse scenarios. During that process, an important aspect has been ignored: since the detailed scenarios are the coarse scenarios together with additional details, we have the coarse information twice. Consequently, the analyzing of detailed images takes once more the time for the coarse recognition plus the time for the additional elements. This mistake is reflected in the measurements: the recognition of detailed scenarios is about 8 times more expensive than the coarse one (about 2.7 times more situations per scenario and 3 times more elements per situation). By using the fact that some of the elements of a detailed description are already identified, we can reduce this ratio to 1:7.

Skipping of processing steps

During the tests it has been confirmed that it can be sufficient to base on identified coarse scenarios instead of using detailed ones. This is done by using a combination of coarse and detailed descriptions. Once the coarse situation and scenario are recognized, the system analyzes the detailed situation. Next, instead of dealing with the detailed scenarios, the already identified coarse scenario is used in order to find a reaction.

This method can dramatically improve the performance of the entire system, since we skip the expensive search for detailed scenarios. However, this improvement is based on the expense of the accuracy. There is no guarantee that the identified detailed situation fits to the identified coarse scenario. There is always the possibility that some features are ignored by the coarse scenario. Thus, this method requires more detailed descriptions of the coarse scenarios – by that it is a combination of processing in one and in several steps.

REACTION

The reaction is more or less a part of the evaluation cycle. Therefore, it will be treated in more detail in the next section.

For the selection of a reaction and the passing of the elements of that action to the *Perception Layer* the system needs an average time of 120 ms – however, the evaluation cycle is not included.

The verification of the consequences is not yet implemented in this realization. However, one can state already now that the major problem of the verification will be the right timing. The system has to know when the expected affects of the reaction should be reflected by the real environment. Moreover, all consequences will not take place at the same time. For example, if the system switches on the light, there will be immediate power consumption but the brightness sensor will need more time to return

the new value. Thus, there must be a process waiting for every single affect of the system's actions in order to compare them with its own expectations.

EVALUATION CYCLE

At first, I will give a short summary of the evaluation cycle. The cycle starts once a reaction is selected. At first, a copy of the internal image as well as of the priority list is made. Next, the predefined consequences of the selected reaction are used to adapt these copies. Finally, a new situation recognition has to be conducted.

The following values are measured by using a P3-550Mhz for the situation recognition application. The PC is connected by a 10Mbit to a MSSQL-Database running on a P3-800Mhz. First, I have to mention that the tests have shown that most of the processing time is used for the database access. Only 2 – 10% of the time is used in the application itself.

To copy the internal representation an average time of 280 ms has been measured; for copying the priority list 10 ms. The adaptation of the copied image by the elements of the consequence of the selected reaction takes about 80 ms – depending on the complexity of the affects; the adaptation of the copied priority list about 120 ms. Thus, after 490 ms the system is prepared for the evaluation and starts a new recognition. In Table 5 the timings of that task are shown.

Table 5: Timings of the recognition task

Current situation	ms		ms	Past situations	ms	ms
Search for possible coarse situation by using the priority list	100					100
Identification of coarse situation by using the internal image	400	Search for coarse scenario containing the identified coarse situation	400	1.5 * 400 - 400 (A coarse scenario consists of an average of 1.5 situations)	200	1000
Identification of detailed situation	1600	Search for detailed scenario containing the identified detailed situation	400	4 * 1600 - 1600 (A detailed scenario consists of an average of 4 situations)	4800	1600 (6800)

2700

The situation recognition starts with the search for possible coarse situations. This search is conducted by using the elements out of the priority list. Once a possible situation is found, the system compares it with the internal representation of the environment, which takes about 400 ms. According to the identified coarse situation, the system looks for a scenario that contains this situation. A coarse scenario consists of an average of 1.5 situations. Since one of situations is already identified (the current one), we can reduce the number of searches by one. For the comparison of past situations, the history list has to be used. In case of coarse scenarios, the comparison with the history list takes about 200 ms. As already explained, coarse situations are simplified descriptions of detailed situations. Thus, in the next step the detailed situation has to be identified. It consists of about 4 times more elements than

the coarse one; the identification takes therefore about 1600 ms. The search for detailed scenarios and the comparison with the history list would take about 5200 ms. However, in the realization this task has been skipped, and instead the coarse scenario has been used. By that, the detailed identification is reduced to 1600 instead of 6800 in total. Hence, the entire recognition task takes about 2700 ms.

By that, the problematic is obvious: the recognition of the current situation takes about 2700 ms. If we want to evaluate the possible reactions, each cycle needs about 3190 ms (490 ms for the preparation and 2700 ms for the recognition of the virtual situation). Even though most of the time is wasted in communication with the database, we have to observe this time very carefully. Otherwise, the system will find the perfect reaction for a situation which is already past.

In Section 3.2.4, among other aspects, calculated situations are introduced. They are constructed by the evaluation cycle by analyzing the virtual internal image. In case of acceptance of the reaction, this virtual situation can be used for the next situation recognition. Its usability will strongly depend on the correctness of the predefined consequences of reactions. However, the integration of this mechanism needs the verification of the consequences – which is not yet implemented in this realization.

5.5 System for situation-dependent behavior

This section is on the one hand a summary of the analysis results of the entire system. On the other hand, these results are compared to the requirements of a system with situation-dependent behavior, as stated in Section 1.3. Therefore, I will use in this section the same structure as in the description of the requirements.

Interoperability

One aim of the system is to deal with interoperability problems on different levels: first, on the hardware level by using appropriate devices, and secondly on a logical level by using symbolic communication. With these two mechanisms it is possible to combine very different devices into one function. However, the need for a transformation of the communication elements into symbols takes much longer than the direct communication between devices. As already mentioned, the measurements are depending on a variety of factors – however, functions using only direct communication between devices have shown to be 5 up to 25 times faster.

Nonetheless, as already explained, as much as possible of the communication has to take place in the hardware level. Considering this aspect, the benefits of the ability to combine all possible devices remain evident.

Sensory system

The requirement on the sensory system is the integration of a large number of devices in order to be able to detect every detail of the environment. As mentioned above, the testing environment offers only about 100 input sources. Thus, it is not possible to perceive all necessary information. Hence, some of the demanded sensory inputs are simulated in the testing phase. Furthermore, most of the real sensory values have been reused several times in order to confront the system with a larger data amount. Yet, this redundant information has already been filtered out at the lower layers, as expected.

Structure

The usage of a decentralized structure is one of the key concepts in the aimed application. Consequently, the system is based – as far as possible – upon fieldbusses. Thus, it is based on distributed, smart devices connected via a bus system. By that, another idea can be realized: a decentralized processing in the periphery of the system, and additionally a concentration of the information flow in a central localized area. The measurements during the testing phase have shown the necessity to keep information paths as short as possible. Therefore, in case of comprehensive complex processes, the involved tasks have to work centrally localized.

The concept of using simple control loops according to reflex actions has turned out to be a valuable mechanism for fast first reactions. However, one has to consider two aspects: it must never occur that a reflex action, which is not controlled by higher layers, leads to a dangerous situation. Secondly, the time needed for the reflex action has to be taken into account. If the action involves different devices which are not able to communicate at the hardware level one has to consider the additional time required for the transformation into symbols.

Beside these – more or less uncontrolled – reflex actions, there are still simple functions realized only by the hardware level or in combination with the task of the completion of trans-sectoral processes explained in Section 3.1.2. Since these functions do not belong to the category of reflexes, they have to offer the possibility of being influencing by the higher layers.

Data storage

Many results have pointed out the importance of the right data storage of the system. First of all, large data amounts have to be stored. Nevertheless, the aimed application requires a fast access to this memory. Moreover, since the overall system uses a variety of tasks that work in parallel, the storage has to handle many accesses to the information at the same time. Consequently, aspects such as the consistence of data have to be assured. Based on these demands, a database is chosen for the representation of the memory.

However, exactly this data storage has proven to be the bottleneck in the overall system. Most of the time (90-98%) is used to retrieve information or to change it respectively. This shows the need for a new concept of data storage. A first attempt could be to involve the local main memory. So far, only the database has been used for holding information. All queries and updates are executed directly on this memory. Depending on the size of the main memory, parts of the database could be stored locally. Though, by that, everyone has to use their own mechanisms for assuring the consistency of the data.

The usage of the main memory would extend the used 3-memory-model by a fourth part: a volatile memory. This means, an additional mechanism has to transfer the volatile information into the permanent parts of the database.

Concerning the proper data types of the symbols, integer values have been selected. They have offered the best performance in most of the demands onto the database. However, as we have seen, the choice of the right data type is not enough for sufficient performance of the data management.

Flexibility

Concerning the flexibility of using different devices, extending the system etc. the resulting application is not restricted in any way. The high flexibility is achieved by the usage of symbols for the communication between different devices or applications. Due to this reason, a value of one device can be made available to all other members of the system.

Moreover, the flexibility concerning the logical content, the knowledge of the system, is demanded. In the first implementation the knowledge is predefined. However, although there are no learning mechanisms integrated in the realization so far, the structure used for the system offers a variety of starting points.

Behavior

The overall aim is a system with an intelligent and preventative behavior. Intelligence means here that it the system able to identify the current situation and react accordingly. Moreover, it should be able to contemplate its own actions. Preventative acting means to foresee events and be prepared for them.

The model structure developed in the course of the work offers both kinds of behavior. Using a variety of mechanisms for collecting and managing information, it is able to identify the current situation and find relations to past ones. Based on that, it selects possible reactions. In an evaluation cycle, the effects of a reaction are analyzed. However, each evaluation cycle takes more time than the origin recognition process. Thus, one has to consider the delay of the realization of an action by the evaluation of it.

The same mechanisms are used to achieve preventative behavior. Once the current situation is identified, the system is able to estimate the next situation by using descriptions of scenarios. Applying this knowledge, it is possible to define reactions at every possible state of a scenario in order to prevent problematic future situations.

Another aspect the system has to manage is the fact that several situations may take place at the same time. By using a processing in several steps, the system is able to deal with this aspect. It extracts all important features of the environment and searches for all stored descriptions of situations using these features. Once a situation is identified, this information is passed on to another task. Beside that, the search for situations goes on. This next task analyzes the situation in more detail and, if it is valid, passes it again on to the next task. The entire process is comparable to a pipeline in a processor unit: each stage deals only with a certain part of the entire process. Once this part is finished, the result is passed on to the next stage, and it deals with the next part offered by the previous stage. The length of this pipeline will have an important influence on the performance of the overall system (Section 5.4).

Performance

The first implementation based on the resulting model structure is not intended to offer an optimized performance. Nevertheless, the measurements during the testing phase have shown new aspects that influence the overall performance and need to be taken into account. Moreover, it is possible to see timing relations between the different parts of the system.

Where it is not intended to optimize the performance, the focus is on the integration of several mechanisms for an efficient handling of information. The benefits of these mechanisms are already discussed in Section 5.3.

Figure 55 shows the measured results of the entire application. As already mentioned, in the application itself a P3-550Mhz, and in the database a P3-800Mhz are used. The PCs are connected via a 10 Mbit network; the PC with the database is additionally connected to the LonWorks network via a network interface card.

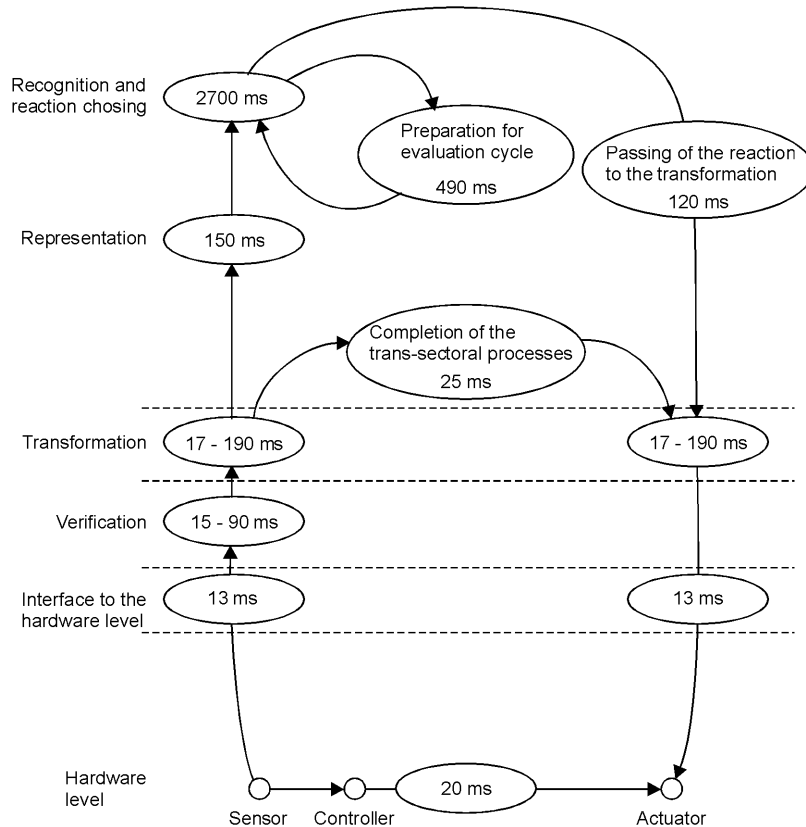


Figure 55: Measurement results

Figure 55 shows clearly the timings needed for the particular tasks. For example, a reflex handled at the hardware level takes about 20 ms in the best case. The time will rise if there is a lot of traffic on the network or if we use complex applications for the function. However, if it is necessary to involve a transformation of a value, the time rises steeply since the value has to pass several additional processes.

The internal representation of the environment needs a relatively short time. The updating of the structure takes about 150 ms. As shown in Section 5.4, the handling of the history list, which is also a part of the representation, will take 420 ms (280 ms for the copy of the current states and additional 140 ms for the reduction of the list). However, the recognition of the current situation will only need the internal image and will take about 500 ms in total. Thus, during this time, the history list can be handled in parallel.

A more expensive part is the recognition and the following evaluation of the reaction – whereby we must keep in mind that most of the time is used for the database access. Nevertheless, each evaluation cycle obviously delays the reaction by more than the time needed for the recognition task.

Finally, another 120 ms are needed to pass the elements of the reaction to the transformation. This is a good example for showing the expensive data management by using the database. After the evaluation cycle, the selection of the reaction is finished. Nevertheless, the measurements have shown an average of 120 ms just for retrieving the elements of the reaction and delivering to the next task.

Recognition rate

During the testing phase, the system has been confronted with information concerning the four exemplary scenarios of Section 2.1. The recognition rate varies between 0 and 100 % depending on several factors. First of all, it depends on the correctness of the descriptions – inaccurate definitions result in no or wrong identifications. However, humans tend to forget some details sometimes. Thus, a learning mechanism for new descriptions of situations and scenarios will be one of the most urgent further works.

A decisive factor is the reduction of the history list. By applying the reduction to every change in the environment, the system is not able to find past situations since the past descriptions are deleted too fast. Thus, it is able to identify the current situation but not the scenario which led to it. Hence, the reduction process has been deactivated in further tests.

Another crucial factor is the management of the priority list. As already explained in Section 5.3, the proper threshold value of the priority list is imperative for the identification of the current situation. If this value is too high, the system might miss necessary elements.

Finally, a problem which has been ignored during this realization occurred in the parallel tasks. The implementation possesses almost no mechanisms for controlling the parallel working. Parallel access to the memory is handled by the database, and the evaluation cycle is able to reserve the tasks of situation recognition and reaction. Beside these mechanisms, no further regulation is implemented. For example, the search for a scenario is able to start even if the history list is not yet finished. Thus, the recognition rate during the tests is not a constant slope or invariable, but it has the look of Figure 56.

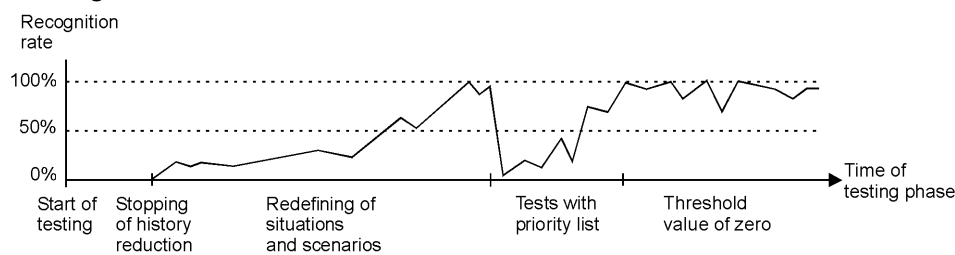


Figure 56: Recognition rate during testing phase

Chapter 6

Conclusion

*Imagination is more important than knowledge
because knowledge is restricted.*

Albert Einstein

In the course of this work, a model structure of a system with situation-dependent behavior has been developed. Basically, this structure is based on abstract objects which represent information items of the environment. Each object may possess a variety of attributes which describe the properties of the object and can have numerous different states.

Whereas the most important aspects have been coped with in this work, there are still some aspects that have to be treated in further works.

6.1 Achievements

Although the resulting model of this work has been designed and tested in the field of home and building automation originally, it is usable in every application field that requires intelligent, automatic acting.

In the course of the work a variety of biological concepts are analyzed and integrated in the final application. Though some of the processes give the impression to be too excessive for the selected testing examples, the resulting work offers the potential to deal easily with a larger number and more complex examples.

Concerning the performance of processing of a large data amount, mainly three concepts come to use: focusing, symbolic values and processing in several steps.

Focusing means the concentration on important features of the environment. This is achieved by assigning priorities to specific states or objects – the idea of this prioritizing and the relation between features of the surroundings and importance are

described in Section 3.1.2. Moreover, the idea of influencing these priorities by combinations of objects has been followed up: if several specific states in the environment are true at the same time, this combination might be more important than each single state itself. Whereas the former, the prioritizing, has proven to be necessary for a reduction of the perceived information, the latter, the changing of priorities, strongly depends on the application field. In the exemplary environment of the realization of this work, it would have been sufficient to integrate only the first concept (Section 5.3).

The usage of symbolic values offers on the one hand the advantage to increase the information content of each item, and, on the other hand, a simplified handling of the perceived information. Symbolic values allow a uniform communication between different devices and homogenous storage of the communication items. In the realization of this work, integer values are chosen as symbols. In doing so, it is possible to profit from the handling of this variable type in the selected database: tests have shown that integers offer most benefits concerning the demands of the exemplary application field.

The third main concept is the processing of information in several steps. Instead of dealing with all perceived inputs at once, the processing of information is conducted step by step. In each step, an increasing number of information is added, and a more detailed processing is started based on the results of the previous step. In the chosen examples, a processing in two steps has been used – although this concept would not have been necessary in this case, the advantages are evident (Section 5.4).

Beside strategies concerning the performance of the system, a variety of other biological concepts are analyzed and integrated. For example, in Section 2.2.3 different models of the memory structure are explained. The storage used for this work can be compared with the stated 3-memory-model. The first memory is located in the devices themselves and stores information only for a very short time. The second one is represented by the system's internal image of the environment. It is continuously updated by changes in the surroundings. And finally, the third part holds the long-term information: stored descriptions of situations, scenarios and relations between items of the environment.

Furthermore, we come again across the subdivision into the declarative and nondeclarative memory of Section 2.2.3. The knowledge of stored objects, attributes, situations etc means the "knowing that". The "knowing how" is characterized by the relations between the stored items – in this case mainly by the relations between tables of the database.

Biological systems rely on a very large number of sensory inputs and therefore on lots of redundant information. This is also a basic concept of this work. In Section 4.2 I have presented a method to deal with information of several input sources, with redundant data about the same objective. The explained concept offers several advantages: first, it is possible to verify the inputs; if there are differences between the perceived values, the validity of the inputs are analyzed. Next, we receive the resulting value of the different perceived ones, which is passed on. Finally, the redundant information is reduced to one abstract information item, which is moved to the higher layers.

Beside these achievements, it has become evident that there are some necessary requirements for such a system as well.

For example, it is required to integrate a large number of sensors, and here, above all, intelligent devices. The application field of this work is situated inside buildings. Hence, it can be installed in manifold environments, using most diverse input sources and dealing with a variety of different situations. Compared with a mobile robot that has to deal with its environment as well, the system of this work has to be omnipresent: a robot analyzes its own state to the surroundings and acts according to that; the robot itself is the active part. In case of the aimed application, the system itself is more or less passive; its behavior is determined by actions of persons in the building or by events in the environment. This system has not only to deal with its own states, but at the same time with all incidents in the building. Thus, it needs a large amount of sensors in order to perceive the necessary information. Here, the intelligent devices come into play: it would neither be manageable to handle all these inputs, nor reasonable to define every simplest behavior in the higher layers. Hence, intelligent devices have to take over this task. They are still part of today's applications, collect information, process it and act accordingly. By that, a very large number of simple situations is handled by the devices themselves. Whereas these simple situations need only local, restricted knowledge, the higher layers of the system are responsible for everything where comprehensive knowledge is required. In Section 5.2 it is pointed out that reduced important information is sufficient for the assessment of situations.

Nevertheless, in spite of this data reduction, one has to expect an extensive amount of data. Thus, it is necessary to use an appropriate storage solution to the memory of the system as well as sufficient processing power.

6.2 Outlook

Since the application field of this work is very extensive, there are still various aspects which are worth to be analyzed. The first realization based on the model structure has allows for research in various areas.

Learning mechanisms

The structure used in this work offers the ability to use learning concepts as stated in Section 1.4.4. It has to be analyzed which of them offers most benefits. Beside these conventional methods, one can use aspects offered by the specific application of the learning task. By watching changes in the environment, the system could extract connections between objects, events, changes etc. (described as "learning by observation" in Section 3.2.4). Values of the current description of the surroundings (short-term memory) together with entries of the history list (long-term memory) can form new descriptions of situations and scenarios. In Section 5.4, first attempts of this learning mechanism are analyzed.

In Section 3.2.4 another method is mentioned: the learning by association. By using a semantic description as shown in that section, the system could be able to find connections between objects and events that have not been explicitly defined before.

In this context, the usage of spatial and temporary distances can be decisive factors. Most of the time events will take place either within a certain spatial range or within a certain time. Let us use an example where a window is open and, because it is raining,

it is wet near the window – everything can be detected within a certain range around the window. If someone enters a room and, because it is too dark, switches on the light, all changes take place within a short time. Although the structure of the internal image offers the possibility to extract information concerning location and time, these factors are still neglected.

Moreover, the learning of proper reactions is still unresolved. So far, we have used actual reactions and evaluated the best one in the evaluation cycle. However, this evaluation has not yet any influence on the next search. Although the selected reaction this time need not be the right one next time due to the changed states in the environment, it has to be analyzed if an influence might improve the behavior of the system.

Forgetting ability

In the human brain the forgetting of data represents an important factor. [Nat50] points out that if we are not able to forget, “our brains will be impossibly burdened with a wealth of useless information”. Tests have shown that we readily forget things which are of no particular importance. Moreover, the connection between important features in the environment and the task of forgetting is clarified as well as the necessity of both. Test subjects, who are not able forget anything, are also not able to single out what is most important. As a consequence of this, they have problems in understanding, for example, a passage of a text.

The same “feature” should be used in a technical system. Though a variety of mechanisms to reduce the amount of data are integrated in the system, still lots of information have to be stored. After a certain time, these stored elements may become increasingly irrelevant. Additionally, the environment may have changed in the meantime and some stored information is no longer useful or valid.

Therefore, the system has to possess the ability to “forget” information after a certain time. There are numerous possibilities to integrate such a mechanism.

In Section 3.2.2 I have introduced the history list and the process dealing with it. Each change in the environment will cause a new entry into this list. However, the amount of stored data will be reduced by each change as well. Thus, this process can be seen as an example of the ability of forgetting. As explained in Section 5.3, the history list needs an extension for the complete forgetting of past events. Although the used mechanism shows good results with diminishing the entries, the remaining elements have to be handled and deleted for good after some time.

Concerning stored descriptions such as situations or scenarios, a kind of timer which indicates the last usage can be included. This additional information can support the decision whether a stored item is still needed. However, one has to consider that dangerous situations like fire or gas escape will probably never take place – nevertheless, they have to remain in the memory of the system. To overcome this problem, the system may delete unused information only if it is below a certain priority.

Handling of malfunctions

So far, the handling of malfunctions has been ignored in the realization. However, there are already several attempts to deal with this thematic.

In Section 4.2 the logging of problematic sensor inputs in an external file is mentioned – but there is no further treatment of this file up to now.

A mechanism for detecting wrong descriptions of consequences is explained in Section 3.4: after an action is realized in the environment, the results of it are compared with the predefined consequences of this action. Again, this feature is not treated in the realization.

Beside these examples, there are additional starting points for the detecting and handling of malfunctions. For example, we can have several possible reactions where the best one is searched by the evaluation cycle – it is still unsolved how to proceed if each of them leads to a problematic situation. However, this problem does not indicate a problem of the model itself, but of insufficient descriptions of internal images. If descriptions of situations and scenarios are wrong or inaccurate, they will lead to wrong actions since reactions are based on them. Furthermore, it could be a sign for too few predefined reactions.

User interface

A thematic that has been disregarded in this work is the user interface. Since the behavior of the system depends on the quality of internal descriptions, an appropriate interface of the definitions will be an important factor.

This work has been initiated in the course of the Smart Kitchen project. One of the next steps in this project will involve the construction of a user interface for the system built in this work.

List of Figures

Figure 1: Multiple usages of functions.....	8
Figure 2: Hierarchy of forming the memory	12
Figure 3: Physical structure of the testing environment	22
Figure 4: Example of a decision tree.....	25
Figure 5: Scenario of Safety.....	33
Figure 6: Scenario of Security.....	33
Figure 7: Scenario of Energy management	34
Figure 8: Scenario of Comfort	34
Figure 9: Biological Sensory System	39
Figure 10: Structure of the central nervous system	40
Figure 11: Structure of Information flow	42
Figure 12: Human senses	43
Figure 13: Functional system	44
Figure 14: Elementary Loop of Functioning.....	45
Figure 15: Sequence of a technical reflex action; [1] leak in pipe, [2] water detection, [3] closed valve.....	63
Figure 16: Knee jerk.....	64
Figure 17: Handling of Reflex Actions	65
Figure 18: Intra-work	66
Figure 19: Local and global inter-work.....	66
Figure 20: Several layers of inter-work.....	67
Figure 21: Global inter-work.....	67
Figure 22: Fuzzy set of temperature.....	73
Figure 23: Perception Layer Interface.....	76
Figure 24: Location matrix.....	79
Figure 25: Object list.....	80
Figure 26: Event list.....	80
Figure 27: Object hierarchy.....	85
Figure 28: Storage of the history; [1] Memory needed for one entry, [2] Structure of one entry over time	87
Figure 29: Representation Layer Interface.....	89
Figure 30: Construction of scenarios.....	94
Figure 31: Preliminary selection of objects and scenarios; [1] Comparison of entire sets, [2] Comparison of reduced sets	96
Figure 32: Coarse and detailed scenarios	97
Figure 33: Semantic network	100
Figure 34: CBR-Cycle.....	102
Figure 35: Adapted CBR-Cycle	103
Figure 36: Usage of intentions for the situation identification	105
Figure 37: Time factors for evaluation.....	107
Figure 38: Evaluation cycle.....	108

Figure 39: Re-handling of a situation; [1] Selection of one reaction, [2] Changed situation...	110
Figure 40: Resulting model structure.....	112
Figure 41: Select.....	118
Figure 42: Update	118
Figure 43: Insert.....	118
Figure 44: Delete	118
Figure 45: Memory structure	119
Figure 46: Structure of the Perception Layer.....	120
Figure 47: Exemplary handling of different sensors.....	122
Figure 48: Structure of the Representation Layer.....	124
Figure 49: Structure of Recognition and Reaction.....	126
Figure 50: Recognition of coarse situations.....	127
Figure 51: Recognition of detailed situations	127
Figure 52: Memory needed in history list.....	135
Figure 53: Priority distribution	136
Figure 54: Different number of processing steps.....	137
Figure 55: Measurement results.....	143
Figure 56: Recognition rate during testing phase	144

Abbreviations

ASIC	Application Specified Integrated Circuit
BACnet	Building Automation and Control Network
CBR	Case-Based-Reasoning
CV	Computer Vision
CYRUS	Computerized Yale Retrieval and Updating System
EIB	Europäischer Installationsbus
HA-Process	History Administration Process
HTML	Hypertext Markup Language
HTTP	Hypertext Transfere Protocol
HVAC	Heating, Ventilation, Air Condition
IPC	InterProcess Communication
KNX	Standard by Konnex Association
LAN	Local Area Network
LON	Local Operating Networks
OPC	OLE for Process Control
PC	Personal Computer
PICS	Protocol Implementation Conformance Statement
PLC	Programmable Logic Controller
RCS	Real-time Control System
SQL	Structured Query Language
STP	Shielded Twisted Pair
TCP	Transmission Control Protocol
TP	Twisted Pair
USB	Universal Serial Bus
WAN	Wide Area Network

WAS Wrap-Around System

Bibliography

- [Aam94] Aamodt, A., Plaza, E.: Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications. IOS Press, Vol. 7: 1, 1994, pp. 39-59.
- [Aeb00] Aebischer, B., CEPE, ETHZ, Huser, A., Encontrol GmbH: Vernetzung im Haushalt. Auswirkungen auf den Stromverbrauch. Schlussbericht im Auftrag des Schweizer Bundesamtes für Energie, November 2000.
- [Aha92] Aha, D. W.: Tolerating noisy, irrelevant and novel attributes in instance based learning algorithms. Int. J. Man-Machine Studies, vol. 36, no. 2, 1992, pp. 267-287.
- [Alb96] Albus, J. S.: The Engineering of Mind. Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior: From Animals to Animats 4, Cape Code, MA, September 1996.
- [Alb99] Albus, J. S.: 4-D/RCS Reference Model Architecture for Unmanned Ground Vehicles. Proceedings of the SPIE AeroSense Technical Conference 3693, Orlando, FL, April 1999.
- [Bal82] Ballard, D. H., Brown, C. M.: Computer vision. Prentice-Hall, Inc. Englewood Cliffs, New Jersey, 1982, ISBN 0 13 165316 4.
- [Bar87] Bareiss, E., Porter, B.: Protos: An exemplar-based learning apprentice. In Proceedings of the Fourth International Workshop on Machine Learning, San Mateo, CA, 1987, pp. 12-23.
- [Bäs99] Bässmann, H., Besslich, P. W.: AdOculos. Digital Image Processing. International Thomson Publishing, Book and Access edition, 1999.
- [Bau98] Repräsentative Haushaltsbefragungen in den Jahren 1988-1992. 1996/1997 Bundesanstalt für Arbeitsschutz und Arbeitsmedizin (BauA), Dortmund, Stand: Februar 1998.
- [Bis98] Bischoff, R., Graefe, V.: Machine Vision for Intelligent Robots. IAPR Workshop on Machine Vision Applications, Makuhari, Tokyo, November 1998, pp. 167-176.
- [Bre84] Breiman, L., Friedman, G. H., Olshen, R. A., Stone, C. J.: Classification And Regression Trees. Wadsworth International Group, Belmont, California, 1984.
- [Bro01] Brooks, R. A.: The Relationship Between Matter and Life. Nature, Vol.

- 409, January 2001, pp. 409–411.
- [Bro86] Brooks, R. A.: A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2, April 1986, pp. 14-23.
- [Bro87] Brooks, R. A.: Intelligence without representation. *Artificial Intelligence* 47, 1991, pp 139-159.
- [Bro97] Brooks, R. A.: From Earwigs to Humans. *Robotics and Autonomous Systems*, Vol. 20, Nos. 2–4, June 1997, pp. 291–304.
- [Bur98] Burke, T. J.: The Performance and Throughput of OPC. White paper of Rockwell Software Inc., <http://www.opcfoundation.org>, 1998.
- [Car99] Carter, R.: Atlas Gehirn. Entdeckungsreisen durch unser Unterbewusstsein. Kapitel 5, Schneekluth Verlag, 1999, ISBN 3 7951 1738 0.
- [Car99a] Carter, R.: Atlas Gehirn. Entdeckungsreisen durch unser Unterbewusstsein. Kapitel 7, Schneekluth Verlag, 1999, ISBN 3 7951 1738 0.
- [Cla89] Clark, P., Niblett, T.: The CN2 Induction Algorithm. *Machine Learning*, 3(4), 1989, pp. 261-283.
- [Cla91] Clark, P., Boswell, R.: Rule induction with CN2: Some recent improvements. *Machine Learning - EWSL-91*, Springer-Verlag Berlin, 1991, pp. 151-163.
- [Coe01] Coenen, O., Arnold, M., Sejnowski, T., Jabri, M.: Parallel Fiber Coding in the Cerebellum for Life-Long Learning. *Autonomous Robots* 11, 2001, pp. 293-299.
- [Cor93] Correia, L., Steiger-Garção, A.: Behavior Based Architecture with Distributed Selection. *NATO-ASI The Biology and Technology of Intelligent Autonomous Systems*, Trento, Italy, March 1993.
- [Cou98] Coulouris, G., Dollimore, J., Kindberg, T.: *Distributed Systems, Concepts and Design*. Second Edition, Addison-Wesley, 1998, ISBN 0 201 62433 8.
- [Dav97] Davis, J.; Bobick: The Representation and Recognition of Action Using Temporal Templates. MIT Media Lab Technical Report 402, 1997.
- [Daw03] Daw C. S., Finney C. E. A., Tracy E. R.: A review of symbolic analysis of experimental data. *Review of Scientific Instruments* 74(2), 2003, pp. 915-930.
- [Die00] Dietrich, D.: Evolution potentials for fieldbus systems. *IEEE International Workshop on Factory Communication Systems, Proceedings, ISEP, Porto, Portugal, September 2000*.
- [Die01] Dietrich, D., Fischer, P.: *LonWorks-Planerhandbuch für Planer, Architekten und Betreiber*. VDE Verlag, 2001, ISBN 3 8007 2599 1.
- [Die01a] Dietrich, D., Russ, G., Tamarit, C., Koller, G., Ponweiser, W., Vinczen, M.: Modellierung des technischen Wahrnehmungsbewusstsein für den

- Bereich Home Automation. e&i, Vol. 11, 2001, pp. 545-555.
- [Die84] Dietrich, D., Kessel, W., Konhäuser, W.: Prozesssicherheit mit parallelen Systemen. Elektrotechnik, 66, H. 5, März 1984.
- [Die97] Dietrich, D., Schweinzer, H.: Feldbustechnik in Forschung, Entwicklung und Anwendung. Springer Verlag Wien, 1997.
- [Die98] Dietrich, D., Loy, D., Schweinzer, H.-J.: LON-Technologie. Verteilte Systeme in der Anwendung. Kapitel 1, Hüthig Verlag Heidelberg, 1998, ISBN 3 7785 2581 6.
- [Dom01] Domnitcheva, S.: Location Modeling: State of the Art and Challenges. Proceedings of the Workshop on Location Modeling for Ubiquitous Computing, Atlanta, Georgia, September 2001, pp. 24-30.
- [Ech00] Echelon: i.LON 1000 Internet Server User's Guide. Version 1.0, Echelon Corporation, 2000.
- [Ech01] Echelon: LonMaker User's Guide. Release 3.1, Echelon Corporation 2001.
- [Eib01] EIB. Konnex Association. Including News of the EIB Association. EIB Journal, Edition July 2001.
- [Fal03] Falkner, M.: Symbolisierung zur effizienten Verarbeitung von Sensordaten Diploma Thesis, Institute of Computer Technology, Vienna University of Technology, 2003.
- [Fau94] Fausett, L. V.: Fundamentals of Neural Networks: Architectures, algorithms and applications. Prentice-Hall, 1994, ISBN 0 13 334186 0.
- [Fif94] Fife, J.: IEEE 1394 Brings bandwidth, simplicity, lower costs to machine vision, scientific imaging. White Paper.
http://bssc.sel.sony.com/Professional/docs/whitepapers/ieee_whitepaper.pdf, Sony Electronics Inc.
- [Fix51] Fix, E., Hodges, J. L.: Discriminating analysis. non-parametric distribution. Technical Report 21-49-004(4), USAF School of Aviation Medicine, Randolph Field, Texas, 1951.
- [Flo97] Floreano, D.: Ago Ergo Sum. In Mulhauser, G., editor, Evolving Consciousness. Benjamins Press, New York, 1997.
- [Foe02] Foerster, H. v., Poerksen, B.: Understanding Systems. Conversations on Epistemology and Ethics. IFSR International Series on Systems Science and Engineering, Volume 17, Carl-Auer-Systeme Verlag Heidelberg, 2002.
- [Foe93] Foerster, H. v.: Wissen und Gewissen. Versuch einer Brücke. suhrkamp taschenbuch wissenschaft 876, Herausgeber Schmidt, S., Suhrkamp Verlag Frankfurt am Main, 1993.
- [Fou01] Founder, J. H.: Ideal World, Ideal Protocol, Ideal System. Article in AutomatedBuildings, www.automatedbuildings.com, November 2001.
- [Gar96] Garney, J.: An Analysis of Throughput Characteristics of Universal Serial Bus. White Paper. <http://www.usb.org/developers/whitepapers/>

- bwpaper2.pdf, 1996.
- [Gel98] Geldard, F. A., Sherrick, C. E.: Raum, Zeit und Tastsinn. Biopsychologie. Beiträge aus Spektrum der Wissenschaft. 1998, ISBN 3 8274 0219 0.
- [Gla00] Glasersfeld, E. v.: Radikaler Konstruktivismus. Suhrkamp-Verlag, 3. Auflage, 2000.
- [Gru99] Gruyter, W. d.: Etymologisches Wörterbuch der deutschen Sprache. Kluge Verlag, 1999.
- [Hab00] Habel, C., Strube, G., Konieczny, L., Hemforth, B.: Kognition. Handbuch der künstlichen Intelligenz, Kapitel 2, 3. Auflage, Oldenburg Verlag München, Wien, 2000.
- [Had99] Hadlich, T., Szczepanski, T.: OPC - Making the Fieldbus Interface Transparent. White paper, <http://www.opcfoundation.org>, 1999.
- [Hai99] Haigh, K., Veloso, M.: Learning situation-dependent costs: improving planning from probabilistic robot execution. Robotics and Autonomous Systems, Vol. 29, 1999, pp. 145-174.
- [Hal92] Hallam, B., Hayes, G.: Comparing robot and animal behaviour. University of Edinburgh, Dept. of AI, 1992.
- [Hei98] Heinze, H. J.: Bewußtsein und Gehirn aus: Gene, Neurone, Qubits & Co, Gesellschaft Deutscher Naturforscher und Ärzte, Hrsg. Detlev Ganten, Berlin, 1998.
- [Ilg00] Ilg, W., Albiez, J., Witte, H., Dillmann, R.: Adaptive posture control of a four-legged walking machine using some principles of mammalian locomotion. In Proceedings of the International Symposium on Adaptive Motion of Animals and Machines, Montreal, Canada, August 2000.
- [Inh00] inHaus. Innovationszentrum Intelligentes Haus Duisburg. Presseartikel, <http://www.inhaus-duisburg.de>.
- [Int00] Intel Corporation: Open Source Computer Vision Library. Reference Manual. Intel Corporation Order Number: 123456-001, December 2000.
- [Jov97] Jovanov, E.: A Model of consciousness: An engineering approach. Brain & Consciousness, Proc. ECPD Symposium, 1997, pp. 291-295.
- [Kab02] Kabitzsch, K., Dietrich, D., Pratl, G.: LonWorks - Gewerkeübergreifende Systeme. Neue Wege in der Planung. VDE Verlag Berlin und Offenbach, 2002, ISBN 3 8007 2669 6.
- [Kae00] CAN ist schon lange industrieller Standard. 1.KK-Fachtagung, Kälte & Klimatechnik, Nr. 3, 2000.
- [Kan00a] Kandel, E. R., Schwartz, J. H., Jessell, T. M.: Principles of neural science. Chapter 12, Fourth edition, McGraw-Hill Companies, 2000.
- [Kan00b] Kandel, E. R., Schwartz, J. H., Jessell, T. M.: Principles of neural science. Chapter 19, Fourth edition, McGraw-Hill Companies, 2000.
- [Kan00c] Kandel, E. R., Schwartz, J. H., Jessell, T. M.: Principles of neural

- science. Chapter 21, Fourth edition, McGraw-Hill Companies, 2000.
- [Kan00d] Kandel, E. R., Schwartz, J. H., Jessell, T. M.: Principles of neural science. Chapter 25, Fourth edition, McGraw-Hill Companies, 2000.
- [Kan00e] Kandel, E. R., Schwartz, J. H., Jessell, T. M.: Principles of neural science. Chapter 27, Fourth edition, McGraw-Hill Companies, 2000.
- [Kan00f] Kandel, E. R., Schwartz, J. H., Jessell, T. M.: Principles of neural science. Chapter 36, Fourth edition, McGraw-Hill Companies, 2000.
- [Kan00g] Kandel, E. R., Schwartz, J. H., Jessell, T. M.: Principles of neural science. Chapter 49, Fourth edition, McGraw-Hill Companies, 2000.
- [Kan01] Kanngiesser, U.: Automatisierungstechnik. Die Entwicklung der Automatisierung. „de“ – Der Elektro- und Gebäudetechniker, Nr.21, 2001.
- [Kha01] Khambatti, M.: Named Pipes, Sockets and other IPC. Case study, April 2001.
- [Kib87] Kibler, D., Aha, D. W.: Learning representative exemplars of concepts: An initial case study. In Proceedings of the Fourth International Workshop on Machine Learning, 1987, pp. 24-30.
- [Kie99] Kiefer, M.: Die Organisation des semantischen Gedächtnisses. Ereigniskorrelierte Potentiale bei der Kategorisierung von Bildern und Wörtern. Verlag Hans Huber, Bern, 1999, ISBN 3 456 83221 4.
- [Kim00] Kimura, H., Fukuoka, Y.: Biologically Inspired Dynamic Walking of a Quadruped Robot on Irregular Terrain - Adaptation at Spinal Cord and Brain Stem. In Proceedings of the International Symposium on Adaptive Motion of Animals and Machines, Montreal, Canada, August 2000.
- [Kin94] Kinnebrock, W.: Neuronale Netze: Grundlagen, Anwendungen, Beispiele. Oldenbourg Verlag, 2. Auflage Auflage, 1994.
- [Kli99] Kline, K., Gould, L., Zanevsky, A.: Transact-SQL Programming. Chapter 20, O'Reilly & Associates, 1999, ISBN: 1 56592 401 0.
- [Kol84] Kolodner, J.: Retrieval and Organizational Strategies in Conceptual Memory: A Computer Model. Hillsdale, NJ, Lawrence Erlbaum, 1984.
- [Kra90] Kratzer, K. P.: Neuronale Netze: Grundlagen und Anwendungen. Carl Hanser Verlag, 1990.
- [Len99] Lent, M. v., Laird, J.: Learning hierarchical performance knowledge by observation. Proceedings of the 16th International Conf. on Machine Learning, Morgan Kaufmann, San Francisco, CA, 1999, pp. 229-238.
- [LON02] LON Nutzer Organisation: LonWorks Installation Handbook. LonWorks in Practice for Electrical Technicians. VDE Verlag GMBH, Berlin und Offenbach, 2002.
- [LON30] LONMARK Interoperability Association. LonMark Layer 1-6 Interoperability Guidelines, Version 3.0.
- [LON32] LONMARK Interoperability Association. LonMark Application Layer

Interoperability Guidelines, Version 3.2.

- [Mai97] Mainzer, K.: Gehirn, Computer, Komplexität. Springer Verlag, 1997, ISBN 3 540 61598 9.
- [May01] Mayers großes Taschenlexikon. Bibliographisches Institut & F.A. Brockhaus Setzerei GmbH, 2001, ISBN 3 411 11308 1.
- [Mcc69] McCarthy, J., Hayes, P. J.: Some Philosophical Problems from the Standpoint of Artificial Intelligence. Machine Intelligence 4, Edinburgh University Press, 1969, pp. 463-502.
- [Mey00] Meystel, A., Messina, E.: The Challenge of intelligent systems. Proceedings of the 15th IEEE International Symposium on Intelligent Control (ISIC2000) Rio, Patras, Greece, 2000.
- [Mic01] Microsoft: Named Pipes vs. TCP/IP Sockets. Optimizing Database Performance (SQL Server), MSDN Library, January 2001.
- [Mic99] Microsoft: Encarta. World English Dictionary, Microsoft, 1999.
- [Mit97] Mitchell, T. M.: Machine Learning. Chapter 4, McGraw-Hill Press, 1997, ISBN 0 07 042807 7.
- [Mor97] Mori, T., Sato, T., Mizoguchi, H.: Situation Reactive Handiwork Support through Behavior Understanding. Proceedings of the 1997 IEEE International Conference on Robotics and Automation, Albuquerque, New Mexico, April 1997, ISBN 0 7803 3612 7 4.
- [Nat50] Nathan, B. P.: Neurobiology. http://www.ux1.eiu.edu/~cfbnp/ZOO_4950/index.htm.
- [Opc98] OPC Task Force: OPC Overview 1.00. <http://www.opcfoundation.org>, 1998.
- [Ota96] Ota, J., Arai, T., Yoshida, E., Kurabayashi, D., Sasaki, J.: Motion Skills in Multiple Mobile Robot System. Robotics and Autonomous Systems 19, 1996, pp. 57-65.
- [Oxf94] Oxford university: Advanced Learner's Encyclopedic. Dictionary. Oxford university press, 1994.
- [Pet00] Peterson, K. L.: Artificial Neural Networks and Their Use in Chemistry. In: Reviews in Computational Chemistry. Wiley-VCH, 2000.
- [Phi00] Philips Lighting Controls: Datasheet of LRI 8133/10 - Multi-Sensor, Helio System.
- [Pos01] Posta, R.: Integrated Management Middleware for Building- and Home-Automation Systems. Dissertation an der Fakultät für Elektrotechnik der Technischen Universität Wien, 2001.
- [Qui86] Quinlan, J. R.: Induction of Decision Trees. Machine Learning, 1, 1986, pp. 81-106.
- [Qui87] Quinlan, J. R.: Generating Production Rules from Decision Trees. Proceedings of the 10th International Joint Conference on Artificial

- Intelligence, Milan, Italy, 1987, pp. 304-307.
- [Rau00] Rauscher, T.: SIIA, Standard for the Implementation of Interoperable Applications. Version 1.0, Mai 2000.
- [Roh94] Rohen, J. W.: Funktionelle Anatomie des Nervensystems. 5. Auflage, Kapitel 5, Schattauer Verlag 1994.
- [Rus01] Russ, G.: Interoperabilität standardisierter LON-Komponenten im Home-Bereich. Diploma Thesis, Institute of Computer Technology, Vienna University of Technology, 2001.
- [Rus02] Russ, G., Dietrich, D., Tamarit, C.: Situation Dependent Behavior in Building Automation. Workshop Proceedings of EurAsia-ICT 2002 Advances in Information and Communication Technology, Shiraz, Iran, 2002, pp 319-323.
- [Sac87] Sacks, O.: The Man who mistook his Wife for a Hat. Summit Books, Simon & Schuster, Inc., New York, 1987.
- [Sac98] Sacks, O.: Stumme Stimmen. Sachbuch rororo, 1998.
- [Sau01] Sauter, T., Dietrich, D., Kastner, W.: EIB. Installation Bus System. Publicis Corporate Publishing, Erlangen, 2001.
- [Sch00] Schneeberger, J.: Planen. Handbuch der künstlichen Intelligenz. 3. Auflage, Kapitel 13, Oldenburg Verlag München, Wien, 2000.
- [Sch97] Schickhuber, G., McCarthy, O.: Distributed fieldbus and control network systems. Computing & Control Engineering Journal, Volume: 8 Issue: 1, February 1997, pp. 21 –32.
- [Sch98] Schürmann, B.: Structure and Design of Building Automation Systems. Proceedings of the International Conference on New Information Technologies in Science, Education, Telecommunications and Business, Crimea, Ukraine, 1998.
- [Sch98a] Schildt, G.-H., Kastner, W.: Prozeßautomatisierung. Springer Verlag, 1998, ISBN 3 211 82999 7.
- [Sen01] Sensory, Inc: Voice Extreme. Speech Recognition Controller. Data sheet. <http://www.sensoryinc.com>, 2001.
- [Sie02] Siemens: Technik denkt. Was künstliche Intelligenz bringt. hi!tech. Das Zukunftsmagazin von Siemens Österreich, Nr.2, 2002.
- [Sin98] Singer, W.: Das Bild im Kopf - ein Paradigmenwechsel. aus: Gene, Neurone, Qubits & Co, Gesellschaft Deutscher Naturforscher und Ärzte, Hrsg. Detlev Ganten, Berlin, 1998.
- [Sou00] Soucek, S., Russ, G., Tamarit, C.: The Smart Kitchen Project – An Application of Fieldbus Technology to Domotics. International Workshop on Networked Appliances (IWNA'2000), New Brunswick, NJ, USA, December 2000.
- [Squ94] Squire, L. Declarative and nondeclarative memory: Multiple brain systems supporting learning and memory. In Memory Systems, Schachter, Tulving, Eds., Cambridge, MA: MIT Press, 1994, pp. 203-

232.

- [Sta97] Starks, S. A., Kreinovich, V., Meystel, A.: Multi-resolution data processing: It is necessary, It is possible, It is fundamental. Proceedings of 1997 Int'l. Conf. on Intelligent Systems and Semiotics, Gaithersburg, MD, 1997, pp. 145-150.
- [Ste00] Steininger, A.: The Testing of Fault Tolerant Computers. Habilitation, Institut für Technische Informatik, Vienna University of Technology; 2000.
- [Ste95] Steiner, J.-L.: Distributed Architectures In Automation Systems With CAN Bus. Selectron Lyss Ltd., INSTRUMENTATION & AUTOMATION, Issue 4, 1995.
- [Str00] Strube, G., Habel, C., Konieczny, L., Hemforth, B.: Kognition. Handbuch der künstlichen Intelligenz, 3. Auflage, Kapitel 2, Oldenburg Verlag München, Wien, 2000.
- [Tak00] Takeuchi, H.: Development of MEL HORSE. In Proceedings of the International Symposium on Adaptive Motion of Animals and Machines, Montreal, Canada, August 2000.
- [Tam00] Tamarit, C.: Automatizaci3n de la vivienda a traves de la tecnologia LonWorks. Diploma Thesis, Institut of Computer Technology, Vienna University of Technology, 2000.
- [Ten00] Tennefoss, M. R.: Implementing Open, Interoperable Building Control Systems. Article in AutomatedBuildings, www.automatedbuildings.com, January 2000.
- [Tin96] Ting, K. M.: The Characterisation of Predictive Accuracy and Decision Combination. International Conference on Machine Learning, 1996, pp. 498-506.
- [Uen99] Ueno, A., Takeda, H., Nishida, T.: Cooperation of cognitive learning and behavior learning. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 99), Vol. 1, 1999, pp. 387-392.
- [Wal97] Walter, A., Hellmann, H.: Dezentrale Automatisierung mit IEC 1131 und CANopen. Feldbusteknik in Forschung, Entwicklung und Anwendung, Springer Verlag Wien, 1997, pp. 325-332.
- [Wan94] Wang, X., Carbonell, J.: Learning by Observation and Practice: Towards Real Applications of Planning Systems. In: Planning and Learning: On to Real Applications, Papers from the AAAI Fall Symposium, AAAI Press, Menlo Park, California, 1994, pp. 156-169.
- [Web98] Weber, K., Venkatesh, S., Srinivasan, M. V.: Insect-inspired robotic homing. Adaptive Behavior, 1998.
- [Wit00] Witte, H., Hackert, R., Ilg, W., Biltzinger, J., Schilling, N., Biedermann, F., Jergas, M., Preuschoft, H., Fischer, M. S.: Quadrupedal Mammals as Paragons for Walking Machines. In Proceedings of the International Symposium on Adaptive Motion of Animals and Machines, Montreal,

- Canada, August 2000.
- [Wro00] Wrobel, S., Morik, K., Joachims, T.: Maschinelles Lernen und Data Mining. Handbuch der künstlichen Intelligenz, 3. Auflage, Kapitel 14, Oldenburg Verlag München, Wien, 2000.
- [Yam00] Yamakita, M., Omagari, Y., Taniguchi, Y.: Jumping Cat Robot with kicking a Wall. In Proceedings of the International Symposium on Adaptive Motion of Animals and Machines, Montreal, Canada, August 2000.
- [Yau02] Yau, S. S., Wang, Y., Karim, F.: Development of Situation-Aware Application Software for Ubiquitous Computing Environments. Proceedings of 26th IEEE International Computer Software and Applications Conference (COMPSAC 2002), Oxford, UK, IEEE Computer Society Press, Los Alamitos, USA, August 2002, pp. 233-238.
- [Yut97] Yuta, S.: Biologically Inspired Approach to autonomous Systems. IFRR International Foundation of Robotics Research, Eighth International Symposium of Robotics Research, Shonan, Japan, 1997, pp. 433-438.
- [Zab02] Zaban, T.: BACnet. A Marketing Perspective. Article in AutomatedBuildings, www.automatedbuildings.com, December 2002.
- [Zad65] Zadeh, L. A.: Fuzzy sets. Information and Control, 8(3), 1965, pp. 338-353.

Internet references

- [WWW1] <http://www.konnex.org>
- [WWW2] <http://www.lonmark.org/>
- [WWW3] <http://www.lonmark.org/products/fprofile.htm>
- [WWW4] <http://www.tacom.army.mil>
- [WWW5] <http://www.tridium.com>
- [WWW6] <http://www.simonsen.no>
- [WWW7] <http://www.microsoft.com/speech/evaluation/techover>
- [WWW8] <http://www.orangemicro.com/ibot.html>
- [WWW9] http://www.pc-cameras.philips.com/manuals/english/win/pcvc720k40_730_740k/index.html
- [WWW10] <http://www.theimagingsource.com/prod/soft/adoculos/adoculos.htm>

*'Begin at the beginning,' the King said gravely, 'and go on till
you come to the end: then stop.'*

Lewis Carroll, Alice in Wonderland