DISSERTATION

Interpretation of Situations in Buildings

Submitted at the
Faculty of Electrical Engineering, Vienna University of Technology
in partial fulfillment of the requirements for the degree of
Doctor of Technical Sciences

under supervision of

o. Univ. Prof. Dr. Dietmar Dietrich
Institute number: 384
Institute of Computer Technology
Vienna University of Technology

and

o. Univ. Prof. Dr.-Ing. habil. Martin Wollschlaeger
Industrial Communications
Technical University Dresden

by

Dipl.-Ing. Wolfgang Burgstaller
Mat. Nr. 9226391
Bergengasse 4/5/12
1220 Wien, Austria

Vienna, November 17, 2007

Abstract

The thesis at hand describes a system for the recognition of predefined situations in buildings. The definition of the situations is based on a hierarchy of symbols, which are also predefined. The archetype for this design is the human brain. Based on the findings of neurologists, neuro-psychologists, and neuro-psychoanalysts, a technical model of hierarchies of symbols and valuation mechanisms has been developed. In the presented approach, data from the periphery are collected, divided into known pieces, and compared with known patterns in order to perceive situations. In another project, the perceived situations are used as basis for decision-making in the interaction of the whole system with its environment.

Modern building automation uses fieldbus systems in order to connect sensors and actuators. On the market, several different standards of fieldbus systems are successful. These standards cannot be directly connected and offer different interfaces for applications. Therefore, this thesis discusses approaches of abstraction of data, which are also suitable for fieldbus nodes with limited resources.

The system presented here can be seen as a further development of building automation systems. Nowadays, such systems fulfill mainly control tasks. But in future it will be possible to fulfill by far more complex tasks as for example the recognition of an individual falling down, a child being in danger, or detecting the present position of people. We discuss here a prototype implementing these applications. In order to ensure that the defined scenarios are recognized, the building has to be equipped with a large number of redundant and diverse sensors. Since this is not feasible at the moment, beside a showcase in one room, a simulator has been used for the proof of concept. The simulator generates sensor values based on a defined virtual environment and scenarios. Although the presented system is concentrated on building automation, the concepts can be used in all areas of automation.

Kurzfassung

Die vorliegende Arbeit beschreibt ein System für die Erkennung von vordefinierten Situationen in Gebäuden. Die Definitionen der Situationen erfolgen auf Grundlage von mehreren Abstraktionsschichten von ebenfalls vorbestimmten Symbolen. Als Vorbild für diesen Aufbau dient das menschliche Gehirn. Basierend auf den Erkenntnissen von Neurologen, Neuropsychologen und Neuropsychoanalytikern wurde ein technisches Modell von Symbolhierarchien und Bewertungsmechanismen entworfen. Wie beim Menschen werden daher auch bei dem hier vorgestellten Ansatz die Daten von der Peripherie gesammelt, in bekannte Einheiten eingeteilt und mit bekannten Mustern verglichen, um daraus Situationen zu erkennen. In einem weiterführenden Projekt werden die erkannten Situationen als Entscheidungsgrundlage für die Interaktion des Systems mit dessen Umwelt verwendet.

In der modernen Gebäudeautomation werden Feldbusse zur Vernetzung der Sensoren und Aktuatoren eingesetzt. Am Markt haben sich verschiedene Standards von Feldbussystemen durchgesetzt. Diese können jedoch nicht direkt miteinander verbunden werden und bieten unterschiedliche Schnittstellen für Anwendungen. Deshalb werden in dieser Arbeit Ansätze einer Abstraktion von Daten diskutiert, die auch auf Feldbusknoten mit limitierten Ressourcen eingesetzt werden können.

Das hier vorgestellte System kann als Weiterentwicklung der Gebäudeautomation gesehen werden. Heutzutage werden vorwiegend Regelungs- und Steuerungsaufgaben von derartigen Systemen erfüllt. In Zukunft wird es aber möglich sein, weitaus komplexere Anwendungen zu realisieren. Die Anwendungen, auf die sich diese Arbeit konzentriert, sind z. B. eine Erkennung, ob ein Mensch stürzt, ob Kinder in Gefahr sind, oder die Erkennung, wo sich Menschen aufhalten. Diese Anwendungen wurden als Prototyp implementiert. Um zu gewährleisten, dass die definierten Szenarien erkannt werden, ist es notwendig, Gebäude mit einer großen Zahl von redundanten und diversitären Sensoren auszustatten. Da dies derzeit noch nicht sinnvoll zu realisieren ist, wird neben einem realen Aufbau in einem einzelnen Raum auch ein Simulator für die Überprüfung des Konzepts verwendet. Mit dem Simulator ist es möglich, Sensordaten, beruhend auf einer definierten virtuellen Umgebung und Szenarien, zu generieren. Obwohl das hier vorgestellte System auf die Gebäudeautomation ausgerichtet ist, sind die Konzepte prinzipiell in allen Bereichen der Automatisierung anwendbar.

Acknowledgement

I would like to express my gratitude and appreciation to Professor Dietmar Dietrich for his constant support, valuable inputs and help to stay on track during the long process of writing this thesis. I would also like to thank Professor Martin Wollschlaeger for his expert opinion of this thesis. My colleagues at the ICT, especially from the ARS Team, were a personal enrichment.

Furthermore I owe many thanks to my parents who supported me in all I have done in my life. Special thanks go to my partner in life Andrea for loving and supporting me for almost a decade.

Preface

Since building automation systems are available, engineers have been trying to find applications and solutions how sensor data can be used to ease the life of the user and the owner of a building. The ARS (Artificial Recognition System) project continues with this attempt, as is described in the introduction of this work. In the application of the project, we are trying to use data from the building to obtain an overview of the present situation in the building. On the basis of this representation, decisions are made on how the system should interact with its environment. Nevertheless, such a system can also be used in other automation areas where it is possible to perceive and interact with the environment. Such application areas could be autonomous robots, e.g. soccer playing robots, or software agents, e.g. for energy management systems.

The difference to most other approaches to the development of applications is the design of the system, which is the most important aim of the project. The design uses the functions of the human brain as archetype. The central theme of the ARS project is to use the results of the study of neuroscientists, psychoanalysts, and neuro-psychoanalysts as basis for our research work and therefore also for this work. The research work of the ARS model in general and in particular the ARS-PC (Artificial Recognition System - PerCeption), which is the focus of this work, is described in the second chapter. As relevant to the ARS-PC part, there follows a description of the neuro-scientific models from which a technical model is derived. The main concepts that where used are the layered perception model and the basic emotion systems as evaluation system of the perception. Since the main application area of this work is scenario recognition in buildings, an interface for classical building automation had to be defined. The third chapter presents a new approach, which allows data abstraction of the major fieldbus systems in building automation.

The forth chapter gives a definition and description of the application fields of types of scenario recognition. These definitions are the basis for the symbols defined in the sixth chapter. To ensure the universal application of these symbols, the fifth chapter presents a framework for symbolic processing in ARS, which includes the formal definition of symbols including an evaluation system, a communication system for symbols, i.e. the so-called SymbolNet, the provision of data either from building automation or from a simulator, and databases for sensor values and perceived symbols.

As mentioned above, the sixth chapter gives a definition and description of the symbols for scenario recognition. At the end of the chapter, there is an illustration of the joint symbol tree. The seventh chapter describes the prototype of the ARS project. It includes the different types of data provision, namely direct connection to the sensors, and access to the data points of different building automation system via a data point abstraction layer and with a simulator for data points. Further methods for the generation of symbols will also be discussed in this chapter. The eighth chapter describes the realization of a prototype of the

ARS system. It describes the techniques which have been used and the resulting outcomes. The last chapter presents a conclusion and an outlook for future work.

The implementation of the presented work here was part of a technology demonstration on the ENF (Engineering and Neuro-Psychoanalysis Form) [ENF] in July 2007. The ENF was organized as a joint workshop of the International Neuro-Psychoanalysis Congress (N-PSA 2007) [NPSA07], and the IEEE International Conference on Industrial Informatics (INDIN 2007) [INDIN]. In combination with the simulator, which is described in Chapter Seven, the scenario recognition has been shown, as described in Chapter Eight.

Abbreviations

AAL	Ambient Assisted Living
ANSI	American National Standards Institute
ARS	Artificial Recognition System
ARS-BASE	ARS Building Assistance system for Safety and Energy efficiency
ARS-PA	ARS Psycho Analysis
ARS-PC	ARS PerCeption
ASN.1	Abstract Syntax Notation One
BACS	Building Automation and Control System
BIBB	BACnet Interoperability Building Blocks
CEA	Consumer Electronic Association
CEN/TC 247	Comité Européen de Normalisation / Technical Committee 247; Building Automation, Controls and Building Management
CNP	Control Network Protocol
DER	Distinguished Encoding Rules; data encoding for ASN.1
DPAL	Data Point Abstraction Layer
EIA	Electronic Industry Alliance
FCL	Fuzzy Control Language
FIS	Fuzzy inference system
FT10	Free Topology
HVAC	Heating Ventilation and Air Conditioning
ICT	Institute of Computer Technology (of the Vienna University of Technology)
IP	Internet Protocol
ISO	International Organization for Standardisation
ITU-T	International Telecommunication Union - Telecommunication Standardization Sector; formerly known as CCITT (Comité consultatif international téléphonique et télégraphique; International Telegraph and Telephone Consultative Committee)
NPDU	Network Protocol Data Unit
NV	Network Variable
oBIX	Open Building Information Xchange
OPC	Openness, Productivity, Collaboration; formerly: OLE for Process Control (OLE: Object
OSI	Open System Interconnection
SDBM	Sensor-Driven, high-resolution Building Models
SNVT	Standard Network Variable Types
SOAP	Simple Object Access Protocol
TCP	Transmission Control Protocol

W3C	World Wide Web Consortium
XML	Extensible Markup Language
XPath	XML Path Language
XSL	eXtensible Stylesheet Language
XSL-FO	XSL Formatting Objects
XSLT	XSL Transformations

Table of Contents

1 Introduction

For years Building Automation and Control Systems (BACS) have been used in buildings and homes. These systems are usually used to control processes within certain areas such as so-called industries. For example, the HVAC (Heating Ventilation Air Conditioning) industry is built of a group of sensors, controllers, and actuators which are necessary for controlling the room temperature. Within this group communication is well defined. The devices are tested with each other to ensure a high degree of interoperability. But if a device of a certain industry is not used within its designated industry, it cannot be guaranteed that it adheres to the rules of that particular industry. The main reason for this separation into industries is to reduce complexity. The cost of the installation of such a system can be reduced, because the control processes just cover one area (or even only certain parts of one area). This situation is even worse if different BACS standards are used. Different BACS usually mean that different data types, service types and application-specific objects are used [Kab02].

However, the approach to define industries and devices for a special purpose is suitable for rather simple applications. Hence, the mathematical description of the control applications is rather simple too. Assuming that from additional sensors additional information can be generated, the functionality of a system can be increased by using this additional information. But in this case the mathematical description of the control application becomes increasingly complex and is no longer feasible.

One way of dealing with that complexity is not to describe the control application mathematically. Artificial neural networks [Arb95, Roj96, Sch97, and Cal03] use that approach. The sensors of a system are connected to the network, which is then trained. During the training phase, it usually needs an interaction with humans, who "tell" the system which information can be derived from the sensor inputs and what should be the reaction to them. Another way is to use sensor-driven, high-resolution building models (SDBM). That approach uses a model of the building and models of the physical processes together with the sensor values of the building in order to obtain a virtual image of the building. [Sut03, Mah06] describes a system for a simulation-assisted lighting control.

Although such systems are used in some special area with great success, these systems are usually greatly specialized. The complexity grows with the number of interdependent control circuits. This approach, first proposed in [Die00, Die01], tries to avoid growing complexity by using a bionic approach. This proposal was incorporated in the Smartkitchen project [Rus03, Tam03]. The project ARS (Artificial Recognition System) [ARS06] has been started as a successor program on the ICT (Institute of Computer Technology) of the Vienna University of Technology. The project is split into three major parts, ARS-PC (PerCeption), ARS-PA (Psycho Analysis) and ARS-BASE (Building Assistance system for Safety and Energy efficiency). The ARS-BASE part uses the statistical methods to analyze sensor data in order to be able to detect abnormal sensor values [Bru07]. The ARS-PA part develops a technical model which supports the decision process of an intelligent autonomous system. It uses the ARS-PC part to obtain environment-relevant information.

This work focuses on the provision of data from the field, and the perception of scenarios based on sensor data. Unified system architectures, especially designed for embedded systems in building automation, are proposed to provide the data. Such a unified representation of sensor data from the field is necessary as basis for the ARS-PC project. The ARS-PC project focuses on the perception of scenarios based on the symbolic processing of information from sensors in an environment. As a basis it uses the model proposed in [Pra06], which takes the human brain as an archetype. It tries to identify already known information in the stream of information received from the sensors of a human. These known items of information are so-called symbols. In a multi-stage process, the brain combines known combinations of symbols to new symbols. The weight of the symbol grows in every stage, whereas the amount of information is reduced. Such a system is able to deal with a high amount of redundant and diverse sensors.

The model proposed in [Pra06] and [Pra07] has been extended by a valuation system, which was inspired by the basic emotional systems of the mammalian brain. The symbols, which are associated from the sensor inputs, are valuated with emotions. In addition the definition of scenarios has been extended and improved. Still, [Pra06] does not address learning in the model. The system can only recognize objects or situations, which are pre-defined. The scenarios, that are to be recognized, are such typical for scenarios in buildings as tracking people and monitor their activities.

2 The ARS Model

The ARS project of the ICT has been founded in order to enhance automation systems. The number of data points even in current building automation systems is in the area of 100000, as for example in the Parliament Buildings in Berlin [BAC04]. However, a growing number of data points sees the effort of installing and extracting information increase. The idea of the ARS project is to use the human brain as archetype for automation systems. The human brain is able to deal with a great deal of information from the environment via the sensors of the body, and is able to react to them. To reduce complexity within the ARS project, the project is split into three parts (see Figure 1), namely ARS-PC (PerCeption), ARS-PA (Psycho Analysis), and Execution. Sensor values from sensors in the environment comprise the input of the ARS-PC part. In this part, the values are interpreted in a way that predefined symbols and scenarios are recognized. The output of the ARS-PC module is a world representation, which comprises in turn the input for the ARS-PA part. Using associations, the ARS-PA module arrives at decisions about actions that have to be taken in a certain situation. The Execution transforms these decisions into actions, which influence the environment via actuators. As mentioned above, this work deals for the main part with the ARS-PC part.

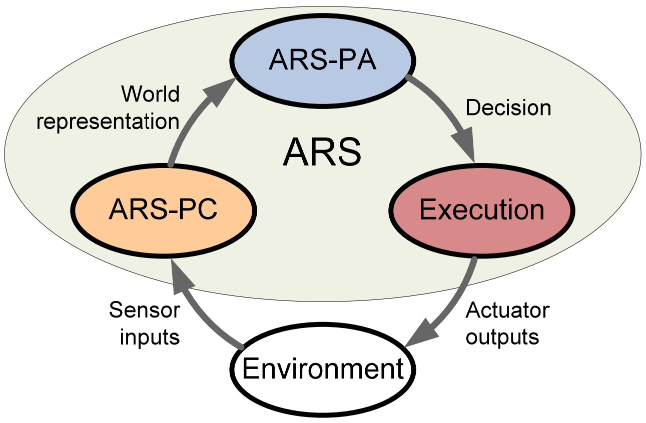


Figure 1: Overall view of the ARS project

The following chapters give a rough overview of models of the human brain, which is used as archetype for the technical model of the ARS project. The advice of neuro-psychoanalysts, psychoanalysts, pedagogues, and neurologists has guided the decision which models of the human brain to use for this project.

2.1 Archetype Brain

The models of the human brain form the basis of the technical models used in the ARS project. This chapter describes how perception is performed in the human brain, which is of particular interest to the ARS-PC project. The ARS-PC model is based on theories by the neuropsychologist and psychoanalyst Mark Solms [Sol02], Jaak Panksepp [Pan98], a neuroscientist working in the field of emotion, and Aleksandr Romanovich Luria[1] [Lur73], often claimed to be the founder of neuropsychology. Both Mark Solms and Jaak Panksepp are co-chairmen of the International Neuro-Psychoanalysis Society [n-psa], which promotes the inter-disciplinary work between the fields of psychoanalysis and neuroscience.

The main task of the ARS-PA project is decision making. The models used as archetype are based on the theories by Mark Solms [Sol02], António Rosa Damásio [Dam99], Sigmund Freud [Fre03], and Rainer Krause [Kra00]. Some of António Damásio's main areas of research are the human consciousness, the philosophy of mind, and cognitive science. Sigmund Freud was one of the founders of the psychoanalytic school of psychology, and is also known as the father of psychoanalysis. The used concept of complex emotions was developed by Rainer Krause [Bur07].

However, the human brain is an organ of the human body, which is interposed between the inner world (internal milieu of the body) and the outer world (external environment). According to Solms [Sol02], the inner world refers to respiration, digestion, temperature control, and the like. The information about the internal state is routed via the hypothalamus and the limbic system to the cortex. The sensory apparatus and the motor apparatus of the body are the interface to the outer world. The following chapter gives a brief overview of the cortex. The cortex is the area of the human brain that processes information from the sensors. Here, information from the outer world is combined with information from the inner world. Based on this combined body of information, the control of the motor apparatus of the body is managed.

2.1.1 The cortex

The theory of the human brain has changed in the last century from a mechanical model to a complex model [Lur73]. Following is a brief description of the parts of the brain that are according to Solms [Sol02] and Luria [Lur73] relevant for the perception and representation of the external world. The physical features of the external world are transformed by the sense organs (e.g. eye, ear, etc.) into nerve impulses. Via different parts of the brain (mainly the thalamus, which plays a major role generating basic emotions; see also 2.1.3) the nerve impulses are routed to dedicated areas in the lobes of the cerebral cortex. The different areas in the cortex are responsible for analyzing, processing, and storing information from the sensors, as well as creating the impulses necessary for taking actions.

Figure 2 shows the main areas in the cortex. The visual information is processed in the occipital lobe (red), the acoustical information in the temporal lobe (green). The parietal lobe (yellow) receives the somatic sensations, which are not only the sense of touch, but also other senses like temperature sense,

[1] Александр Романович Лурия; in this work the English transcription Aleksandr Romanovich Luria is used; the transliteration is Aleksandr Romanovič Lurija

vibration sense, or muscle and joint position sense. The frontal lobe (blue) in Figure 2 is the area that connects the motor organs of the body.

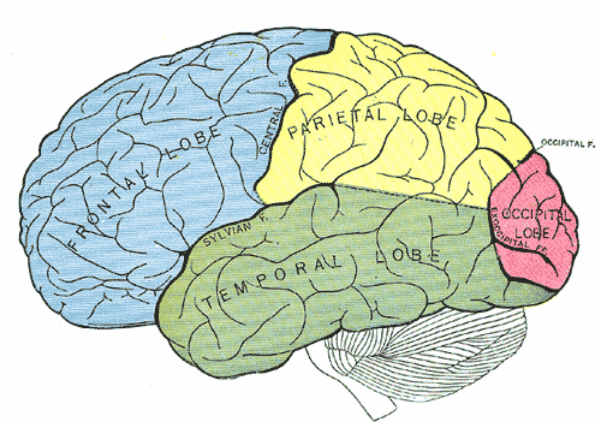


Figure 2: Principal fissures and lobes of the cerebral cortex viewed laterally [Gray728]

In the frontal lobe, information from the outer world (processed via the posterior lobes) and information from the inner world (processed via the hypothalamus and the limbic system) are used to program, regulate, and verify the actions of the person. The following chapter gives a brief overview of the functional units of the brain.

2.1.2 The functional units of the brain

The above sketch describes the main areas of the brain and where they are located. Aleksandr Luria defined a model of the functions of the human brain [Lur73]. This model has three basic function modules as represented in Figure 3. One module is responsible for the control of tone and waking, one module controls the obtaining, processing and storing of information arriving from the outer world, and the third module manages the programming, regulating and verifying mental activities. It should be stressed here that these functional units can be associated with regions of the brain, but not with small areas or even with single neurons.

The human consciousness comprised all three modules – tone, reception and action module. The tone and waking unit is essential for the human mental process. Only under optimal conditions of waking can the human brain receive information, analyze it, and direct its own actions. The reception module receives its stimuli from the sense organs and divides them into a huge amount of parts. These parts are then assembled into dynamic-functional structures. The action module is not only responsible for the reaction on the inputs from the reception module but also for planning and coordinating the action of the human with his aims. It controls the motor activity of the body and monitors whether the actions fit the aims.

The reception and the action unit are part of the cortex. The reception unit includes the parietal lobe, the temporal lobe, and the occipital lobe, and the action unit contains the frontal lobe. The cortical tone is not generated in the cortex, but below in the subcortex and brainstem.

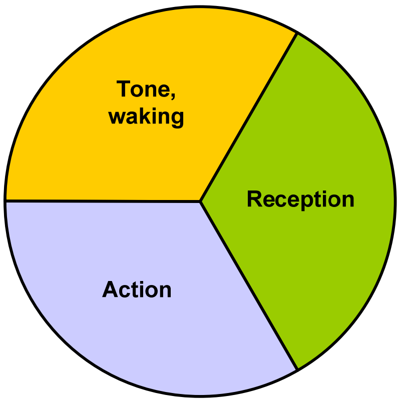


Figure 3: Model of the human brain

Since this work focuses on perception, we need to take a closer look on the reception module and the identification and description of its sub modules. According to Luria [Lur73], the module for reception, analysis and storage of information has a hierarchical architecture. This architecture has at least three layers. The layers of the reception module are shown in Figure 4.

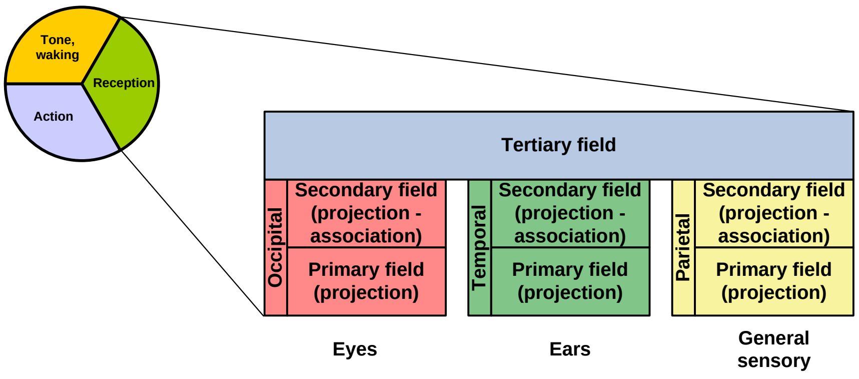

Tone, waking
Reception
Action
Tertiary field
Occipital
Secondary field (projection - association)
Primary field (projection)
Temporal
Secondary field (projection - association)
Primary field (projection)
Parietal
Secondary field (projection - association)
Primary field (projection)
Eyes
Ears
General sensory

Figure 4: The reception module

The primary field of the reception unit, which is also called projection field, consists of highly specialized neurons. It reacts only to very specific patterns. In the visual primary field such patterns are e.g. chromaticity or a direction of movement. It processes the information that comes in from the sensors. The next hierarchical field is the secondary or so-called projection-association field. The neurons of these fields are not as specialized as the ones in the primary fields but still have a relatively high degree of specialization. The secondary field has synthesizing functionality.

Incoming stimulations from the primary field are associated with patterns. The combination of information from all secondary fields is completed in the tertiary field. The specialization of the neurons of that field is even lower than in the secondary field. In the tertiary field, the transformation from specific perception into abstract thinking takes place. Based on the structure of the brain as described above, Luria derives three basic laws [Lur73], which are illustrated in Figure 5:

1. The first law is that the structure of the cortical zones is hierarchical. The higher fields control the lower fields[2].
2. The second law is the law of the diminishing specificity of the hierarchically arranged cortical zones that are composing it. The primary fields (occipital, temporal, and parietal) of the cortex contain a very large number of neurons, which have highly differentiated modally specific functions. In the secondary fields the associative neurons dominate, whose specificity is lower than in the primary fields. In the tertiary fields of the reception unit, the modal specificity is represented to an even lesser degree.
3. The third law is the law of the progressive lateralization of functions. The primary fields of both cerebral hemispheres have identical roles, whereas in the secondary and tertiary field one hemisphere is dominant.[3]

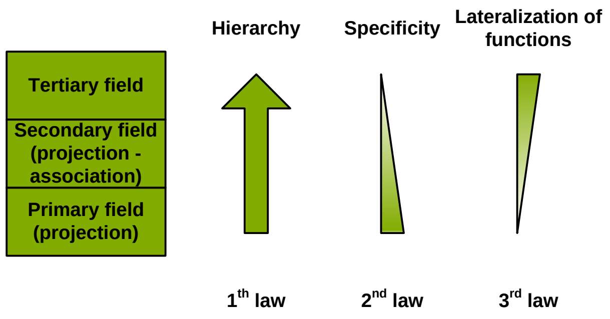

Hierarchy
Specificity
Lateralization of functions
Tertiary field
Secondary field (projection - association)
Primary field (projection)
1th law
2nd law
3rd law

Figure 5: Basic laws of the architecture of the human brain

Additional to the perception of the environment, the human brain uses basic emotions as a valuation system of the sensory input. The following chapter describes this valuation system and the basic emotions.

2.1.3 Basic Emotions

The literature is rich with definitions of basic emotions. An anthology of these definitions can be found in [Ort90]. Despite different views on basic emotions by different theorists, there is consent about the existence of a set of basic emotions. Mark Solms uses in [Sol02] the nomenclature of basic emotions according to Jaak Panksepp [Pan98]. I will use the same nomenclature in this work in order to achieve consistency in the description of the human brain. It should be noted here that according to [Sol02] and [Dam99], the term emotion is used to refer to emotions which are unconscious. In contrast, conscious

[2] This applies to the brain of an adult. During the ontogenetic development, the higher fields depend on the lower fields. After the secondary and tertiary field are developed, the dependencies change according to the first law.

[3] For right-handed persons the left cerebral hemisphere is dominant.

beings can register their emotions as feelings of emotions. I will focus here exclusively on unconscious emotions and use therefore the term emotion.

The basic emotions are the same in all mammals. These basic emotions are hardwired and allow mammals to evaluate situations, and in particular situations of biological significance (e.g. danger). With adhering to what these basic emotions propose, an individual can increase the likelihood of the survival of its organism. Therefore, Panksepp suggests in [Pan98] that the basic emotions should be thought of as e-motions – elocutionary emotions. Figure 6 shows the interaction of the emotional systems in the brain according to [Pan98].

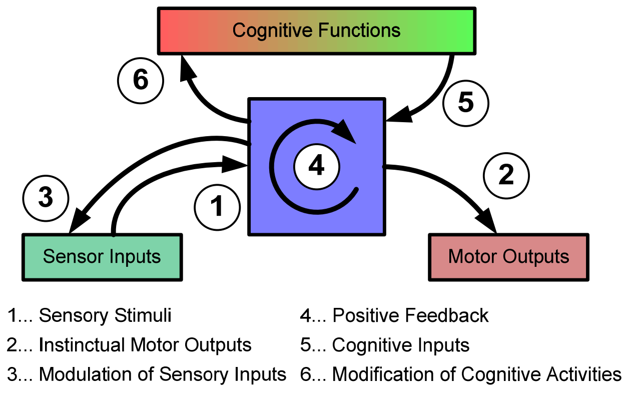


Figure 6: Interaction of the basic emotional systems with other functions of the brain [Pan98]

The basic emotion systems are typically triggered by events in the outer world (1). The emotions can trigger instinctual motor outputs (2) and modulate sensory inputs (3). With a positive feedback loop (4) emotions are sustained also after their trigger has passed. The emotional system can be modified by cognitive inputs as well as the emotional system modifies and channels the cognitive activities. The basic emotions defined by Panksepp [Pan98] are seeking, rage, fear, and panic. Figure 7 shows these basic emotions and their dependencies on the inner and outer world.

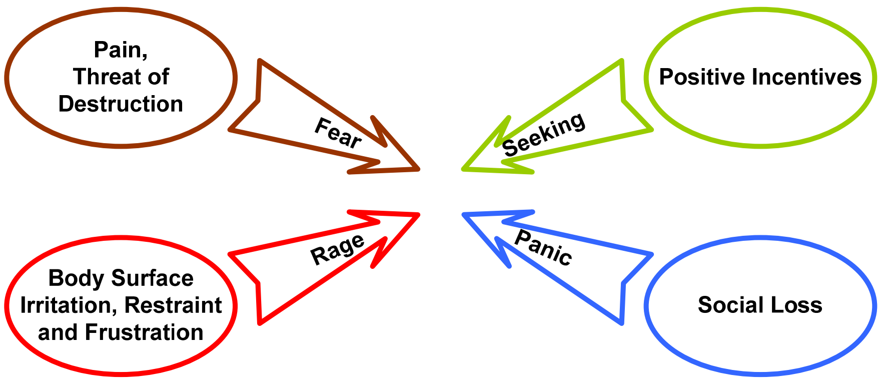

Pain, Threat of Destruction
Fear
Positive Incentives
Seeking
Body Surface Irritation, Restraint and Frustration
Rage
Panic
Social Loss

Figure 7: Basic Emotions [Pan98]

The seeking system is influenced by positive incentives (e.g. food, water, social contacts, etc.). The system is responsible for activating the organism’s interests in the outer world. The panic system is triggered if the individual loses social contacts (e.g. lost child). The rage system is activated when goal-directed actions are thwarted. The fear system is triggered in case of threat of destruction.

2.1.4 Résumé

As outlined in the previous chapters, the model of the brain according to Luria and Solms comprises on the one hand reception of information from the body's sensors (sense organs, outer world). This information is directed to the dedicated parts in the brain, namely the cortex. Within the cortex, the information is then split into a great amount of components and clues, which are coded and synthesized afterwards. On the other hand, the internal world is projected to the frontal lobe of the cortex, where it is combined with information about the outer world.

Additional to the dataflow described above the basic emotion systems valuate the inputs from the inner and outer world. I use here Panksepp's model of basic emotional systems. The basic emotion systems cause instinctive actions and influence the perception and higher thinking activities of the human. These major functional systems of the brain for perceiving the environment are used in the following chapters to derive from them a technical model which has similar properties as describe above.

2.2 Derived Technical Model

Before deriving the technical model from the presented functional systems of the brain, two research projects, i.e. Blue Brain [BBP06] and Ccortex™ [Cco06], are described briefly in order to demonstrate why it is difficult or even impossible to simulate functions of the (mammal) brain by simulating the functionality of neurons. Although only parts of the functions of the brain are simulated, enormous computational power is required. The Blue Brain project uses the Blue Gene supercomputer to simulate a neocortical column of a rat in real-time. It contains of about 10000 neurons and about 10^8 synapses. A neocortical column of a human consists of about 60000 neurons, has a diameter of about 0.5mm and a height of about 2 mm. According to the list of the top 500 supercomputers[4] [TOP500], the IBM eServer Blue Gene Solution system, installed at IBM's Thomas Watson Research Center where the Blue Brain project is simulated, has a performance of 91.20 Tflops (Linpack).

The goal of the Ccortex project is to simulate the human cortex, basal ganglia, thalamus, and hippocampus. For this purpose, $20*10^9$ neurons and $20*10^{12}$ synapses [Cco06] have to be simulated. The supercomputer used in the Ccortex project is expected to reach a theoretical peak performance of 4.800 Gflops. However, the last update of the site was done in September 2005 and no recent publications could be found. Therefore, I have to assume that the project was stopped.

Although these numbers are imposing, the total number of neurons and synapses is not reached by some order of magnitude. The neocortex of the human has about 10^{10} neurons and the estimated number of synaptic connections is 10^{13} [Pan98]. It does not seem possible in the near future to achieve enough computational power to simulate the human neocortex even by using the most powerful supercomputers

[4] 28th TOP500 List was released in November 2006

available. Comparing the numbers of neurons and synapses of the Blue Brain project, it has to be emphasized here that 10^6 less neurons and 10^5 less synapses are simulated than are in the human neocortex.

However, to apply models of the human brain to applications like Ambient Assisted Living (AAL), the approach of neuron simulation is not feasible. Moreover, there are no models available at present which can explain how higher functions of the brain like consciousness are derived from the functionality of neurons. Therefore, models of higher brain functions cannot use a simulation approach. Hence, the ARS project chooses a different path. It does not simulate neurons in the brain, but it uses neuro-psychoanalytic models. These models describe the functionality of the brain and the mind of humans, but do not refer to single neurons in the brain. Using this approach, we can expect following benefits.

Neuroscientists, psychoanalysts, and pedagogues like Solms or Panksepp have developed their models in the last decades based on theories by Freud and Luria. Therefore, engineers do not need to create new models from scratch, but can use the results of research by experts in the field of neurosciences. The engineers for their part can give feedback and thus add to the improvement of the models. So both neuroscientists and the engineers collaborate to obtain better models of the brain.

The mammalian brain, and especially the human brain, is the brain which has the highest cognitive abilities known to scientist. Using neuro-scientific models of the human brain, there is a high chance that related technical models have similar functionalities. In order to reach this ambitious target it is necessary to unit models of the brain which can then be transferred into a unitary technical model [Die07].

2.2.1 Interaction of ARS-PC and ARS-PA

The separation of the ARS project into the subprojects ARS-PC (see Chapter 2.2.3) and ARS-PA (see Chapter 2.2.4) has two main reasons. The first and pragmatic reason is to reduce complexity. Using simulators, it is not necessary to implement the whole system at once or to develop and test all parts of the system. Figure 8 illustrates the division of the project. On the left side of the functions of the overall ARS system are depicted, symbolized with small ellipses. The functions which are based on psychoanalytic models and are therefore higher level functions are grouped in the blue ellipsis, the hierarchical lower functions in the orange ellipsis. The symbolized functions in the orange ellipsis are based on neuro-scientific models. In order to aid the development and testing of all ARS functions, ARS is split into ARS-PC and ARS-PA as shown on the right hand side of Figure 8. In the ARS-PA part, the lower functions of the brain are replaced with a simulation of perception. This project part is therefore concentrated on the functionality of decision making, based on the psychoanalytic model. Complementary, the research work in the ARS-PC part is concentrated on the perception of the environment. However, the interface between the projects is defined as the world representation of the outer world (see also Figure 1).

The second reason is the use of different designs. The ARS-PC uses a bottom-up design where symbols are used as building blocks, which are assembled according to world representation. The ARS-PA project uses the top-down approach, transforming the psychoanalytic model into a technical model [Deu06].

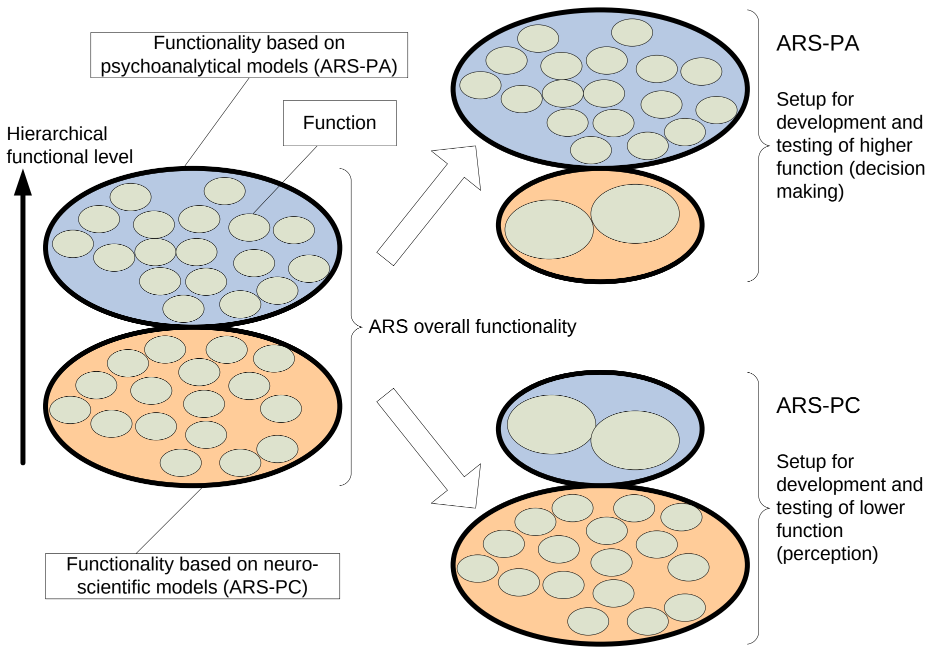

Functionality based on psychoanalytical models (ARS-PA)
Function
Hierarchical functional level
ARS overall functionality
Functionality based on neuro-scientific models (ARS-PC)
ARS-PA
Setup for development and testing of higher function (decision making)
ARS-PC
Setup for development and testing of lower function (perception)

Figure 8: Reducing complexity in the ARS model with splitting

Nevertheless, both projects are designed to work together. Figure 9 shows the simplified modular view of the ARS project as proposed in [Pra06] and [Deu06]. The ARS-PC project has only the module Symbolization. This module compares sensor data from the environment with predefined situations. If a predefined situation matches with the situation perceived by the sensors, the situation is identified as a known situation. The symbolization uses hierarchy of symbols to define situations. Similar to situations, the symbols are predefined as well. Higher-level symbols depend on symbols which belong to a lower hierarchical level. The symbols of the lowest hierarchical level, the so-called microsymbols, are composed of the sensor values. The interface between the ARS-PC and ARS-PA project is the world representation, which is described with representation symbols. A detailed description of the ARS-PC part follows in Chapter 2.2.3.

The ARS-PA project, as described in [Deu06], uses world representation and internal states to make decisions. The color gradient from red to green in Figure 9 is to point out that the decision making process is not a straightforward approach, but is multi-layered and contains a number of feedback loops. The internal state is derived from semantic and episodic memory, from the so-called superego, and from drives and emotions (basic and complex emotions). According to the psychoanalytic model, decisions are made as a result of the contrast between superego and drives. Further, the module Decision is responsible for changing the inner state according to input from the world representation and the expected response from the taken actions. The module Execution transforms the decisions from the module Decision into commands to the actuators of the system. The first implementation of the ARS-PA is the so-called Bubble Family Game [Deu07]. Detailed description of the ARS-PA part follows in Chapter 2.2.4.

In the current ARS model [Pra06] and [Deu06], the basic emotions are part of the ARS-PA project. The basic emotions do not influence the world representation. However, according to [Pan98], basic emotions in the mammalian brain also influence the perception of the outer world. Therefore, the ARS model was modified in this work as described in the following chapter. The modification of the ARS model is based on the neuro-scientific model presented in Chapter 2.1.

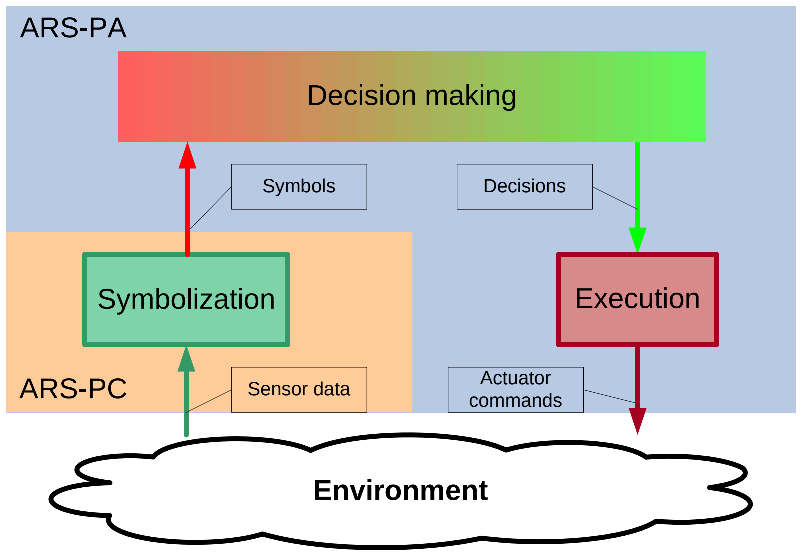


Figure 9: Modular view of the ARS project

2.2.2 Modified ARS model

The current ARS model described above is missing a concept of basic emotions as described in Chapter 2.1.3 and illustrated in Figure 6. The basic emotions are only part of the higher brain functionality (ARS-PA), and do not influence the perception (ARS-PC) but are influenced by it. Since the emotional systems in a mammalian brain influence other subsystems of the brain and are themselves influenced by these subsystems, the emotional system cannot be modeled as a single functional module. Every subsystem, which is dependent in the emotional system, has to implement interfaces to interact with the emotional system [Bur07]. Therefore, I modified the ARS model in order to integrate the basic emotional systems into both the ARS-PC and ARS-PA model.

The emotional systems influence the modules in the ARS-PC project as well as in the ARS-PA. A modular overview of the ARS project concerning the integration of the basic emotion systems into ARS-PC and ARS-PA is shown in Figure 10. It uses the same colors and numeration as Figure 6, which depicts the interactions and functionality of the basic emotional systems of the mammalian brain, as well as Figure 9, which illustrates the modular view of the ARS project.

The e-systems are in the centre of Figure 10. The term e-system stands for the technical model of the functionality of a basic emotional system as described in [Pan98]. It consists of a number of e-systems, where each system represents a basic emotion. In order to have a flexible system, which can be adapted to particular applications, the number and type of basic emotions are not defined or limited in the concept of the ARS system. The functionality of the characteristics of the mammalian brain is partly implemented in the e-systems, and partly in the other modules of the ARS model. Introducing the e-systems as module into the ARS model, the world representation and therefore the perception of the ARS system is influenced by the internal states of the system.

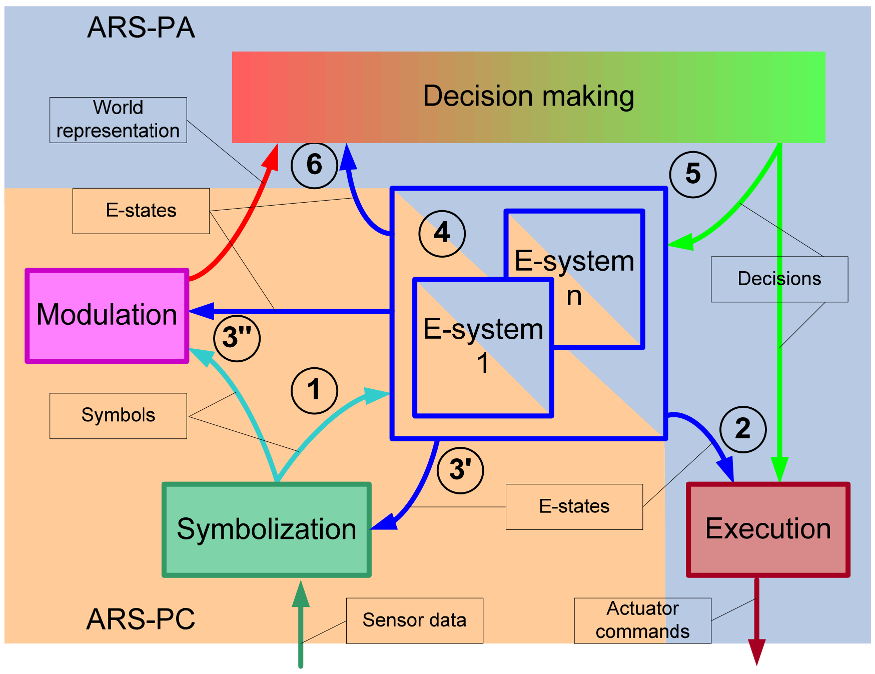


Figure 10: Introduction of e-systems into the ARS model

The module Symbolization has beside sensor values from the field an additional input from the e-systems (3'). This input influences the symbolization of the sensor values. The symbols of the module Symbolization are sent to the module Modulation (3'') and the e-systems (1). The module Modulation valuates the symbols of the module Symbolizations according to the values of the e-systems. The output of the module Modulation is then the world representation of the outer world. The modules Symbolization and Modulation form together the ARS-PC project.

The ARS-PA project comprises the modules Decision and Execution. The module Decision has two inputs, the valuated symbols of the module Modulation (world representation) and the input from the e-systems (6). Decisions are generated from the input and knowledge within the module. Theses decisions influence the e-systems (5). The module Execution controls the actuators, which influence the environment. Additional to the decisions of the module Decision, the module Execution receives input from the e-system (2). Both inputs can cause actuator commands.

The e-systems are influenced by the sensor data (1) as well as by the decision of the module Decision (5). Within the central modules, the positive feedback loop (4) is implemented. Together with the inputs of the e-systems, the current values of the e-systems are determined. Via the interfaces of the modules Symbolization (3'), Modulation (3''), and Decision (6), perception and decisions are influenced. The interface of the module Execution (2) is used to model instinctive behavior.

The modified ARS model constructs the functionality of the basic emotional system as described in 2.1.3. It is triggered by events from the outer systems (1). The emotions can trigger (instinctual) motor outputs (2) and influence the word representation (3) of the perception. The influence on the world representation was separated into two parts, namely how the environment (3') is perceived and what is perceived (3'').

Within an e-system, a positive feedback loop (4) makes sure that an emotion is sustained even after the trigger for it has passed. As in the ARS-PA model, emotions control decisions (5) as well as decisions influence emotions (6). In the following chapters, the ARS-PC and ARS-PA model are described in more detail.

2.2.3 ARS-PC model

Gerhard Pratl proposed in [Pra06] the ARS-PC model. I adapt this model here according to the neuro-psychoanalytical model as described in Chapter 2.1 primarily with the introduction of basic emotional systems. The second major adaptation is the reworking of symbolization according to the neuro-scientific principals set-up by Luria as formulated in Chapter 2.1.2. This chapter illustrates the rework of the layered symbolization model of the ARS-PC project according to the principals Luria.

The ARS PC project should be seen as the technical implementation of the functionality of the reception module, which is one of the three main functional units of the cortex of the human brain. As shown in Figure 10, the ARS-PC model consists of two main modules, i.e. Symbolization and Modulation. The e-system modules are part of both the ARS-PC and ARS-PA project. The module Symbolization implements the perceptive functionality of the cortex. This includes the processing of sensor data from the periphery as well as a valuation of the outer world according to the values of the e-systems. The module Modulation can be seen as filter, which filters the world representation depending on the e-systems. But before the relationship between the neuro-psychoanalytical model and the ARS-PC model is described, we need to define the term symbol as used in our approach.

The term symbol is used in many areas and in each of them it has at least a slightly different meaning. It originates from the Greek word σύμβολον (symbolon), which is consists of the words συν (sym-) meaning “together” and βολή (bole) meaning “throw”. Literally, the word symbol has the approximate meaning of “to throw together” or “the joined”. The Latin word is symbolum. The most basic meaning of the term symbol is a conventional representation of a concept, idea, quantity, object, etc. Within this work and the ARS project, a symbol is seen as a representation of a collection of information [see Pra06 and Goe06]. Since learning is not addressed here, symbols are pre-defined. This means that the system is only capable of recognizing known information patterns of the sensors.

The relationship of the neuro-psychoanalytical model and the ARS-PC model is shown in Figure 11. The reception module in the neuro-psychoanalytical model is divided into three parts, the primary (projection), secondary (association) and tertiary field. In the ARS-PC model [Pra06] the module Symbolization is equivalent to the reception module of the neuro-psychoanalytical model. It is also divided into three layers, the microsymbol layer, the snapshot symbol layer and the representation symbol. In the primary field are highly specialized neurons, which react only to special patterns. In the ARS-PC model, this layer corresponds to the so-called microsymbol layer. In this layer, microsymbols are generated from sensor values.

The secondary field in the reception module of the cortex combines the features which have been extracted in the primary field. In the technical model, the microsymbols are synthesized to new symbols. In [Pra06] the second layer is called snapshot symbol layer. It has to be mentioned here, that in [Pra06] snapshot symbols are defined as symbols which represent the world at one point of time (or a short time span, respectively). This definition does not cover its use a hundred percent here. According to [Lur73] synthesis is the key task of the secondary fields. As a matter of course, this represents part of the world, but in this work here, snapshot symbols are seen as building blocks used by the next higher layer, the representation symbol layer.

Another reason for the slight change in the definition of snapshot symbols is the fact that the primary and secondary fields are separated and do not join the sensory information together. A representation of (a part of) the world would then be limited to one sense. In our work, it is the tertiary field that has this ability of combining symbols from all senses, thus creating a world representation.

The neuro-psychoanalytical model is limited to the senses of a human. The technical model of the ARS-PC project is not limited in this regard. For every type of information connected to the model, a microsymbol layer and a snapshot symbol layer can be attached to the model. It is the key task of the representation layer to link the (snapshot) symbols of the different areas of information together in order to create a consistent world representation. The ARS-PC model in Figure 11 is an example of this. In any implementation, a visual sub-system or acoustic sub-system can but does not necessarily have to be integrated.

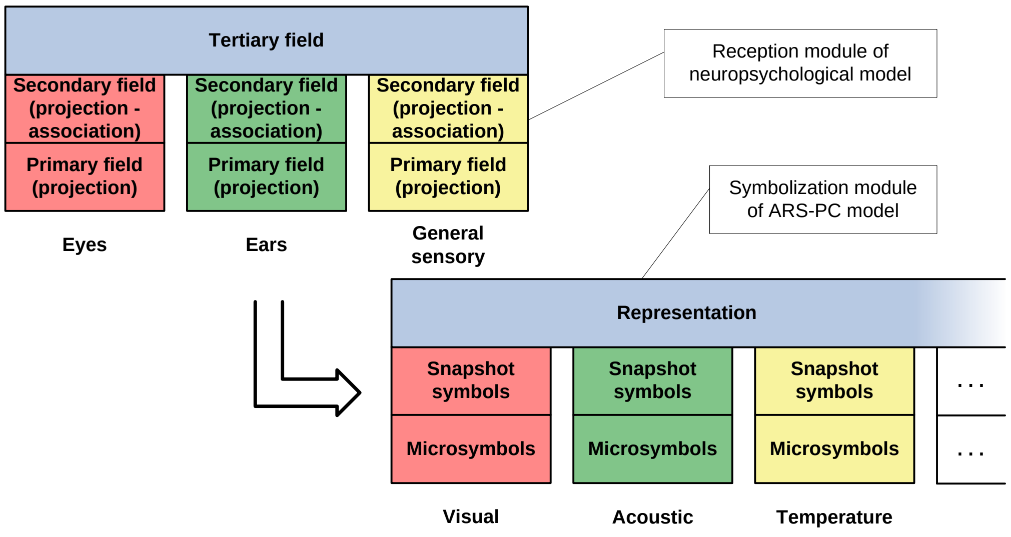

Tertiary field
Secondary field (projection - association)
Secondary field (projection - association)
Secondary field (projection - association)
Primary field (projection)
Primary field (projection)
Primary field (projection)
Eyes
Ears
General sensory
Reception module of neuropsychological model
Symbolization module of ARS-PC model
Representation
Snapshot symbols
Snapshot symbols
Snapshot symbols
...
Microsymbols
Microsymbols
Microsymbols
...
Visual
Acoustic
Temperature

Figure 11: Relationship of the neuro-psychoanalytical model and the ARS-PC model

Two of the three basic laws defined by Luria [Lur73] and briefly summarized in Chapter 2.1.1 apply also to the technical model of the human brain. The first law says that the cortex has a hierarchical structure. The technical model is also designed as a hierarchical model. The symbols of the microsymbol layer are the basis for the synthesis of the symbols of the higher layers. The same applies to all other symbol layers.

The second law says that the specificity of the higher layers is lower than the specificity of the layer below. In the ARS model the microsymbols are only generated from of one kind of sensors, e.g. temperature, and represent a very specific pattern in the sensor data. The higher-layer symbols are synthesized out of the microsymbols. Higher-layer symbols therefore have a lower specificity than the symbols of the layer below. They have integrative functionality.

The third law says that especially the higher functional layers are processed only in one hemisphere of the brain. For an implementation of microprocessors the spatial distribution of the computing power is not so important. Contrary to the human brain, the number of processors and the computing power of a single processor can be increased. One could interpret the third law as a strategy to distribute the tasks of the model to different processors in a multiprocessor system. However, in the ARS model the distribution of the computational power is not modeled in this work.

The equivalents of the basic emotional systems of the human brain are the e-systems and the modules, which are influenced by the basic emotions. The two characteristics of the basic emotional systems, sensor stimuli and modulate sensory inputs, are modeled in the ARS-PC project and therefore part of this work. In the technical model the stimulus from the sensors is not modeled as a direct input from the periphery, but the symbols generated in the module Symbolization are used as the stimuli for the e-systems. It has to be pointed out here that not only representation symbols can influence e-systems, but also snapshot symbols and microsymbols.

The second characteristic of the basic emotional systems of the neuro-psychoanalytical model is the modulation of the sensor inputs. In the technical model this characteristic is split. Both the module Symbolization and the module Modulation are influenced by e-systems. Thereby, the module Symbolization creates all symbols which are included in the world representation. The module Modulation does not create additional symbols itself, but can modify symbols or even filter them out completely, according to the e-status of the e-systems.

2.2.4 ARS-PA model

The ARS-PA model covers the higher cognitive functions of the human brain. It uses the psychoanalytic model according to [Sol02], [Dam99], [Fre03], and [Kra00]. [Bur07] gives a rough overview of the main modules in the ARS-PA (see Figure 12), which is described below. This model is a simplification of the model presented in [Pra07].

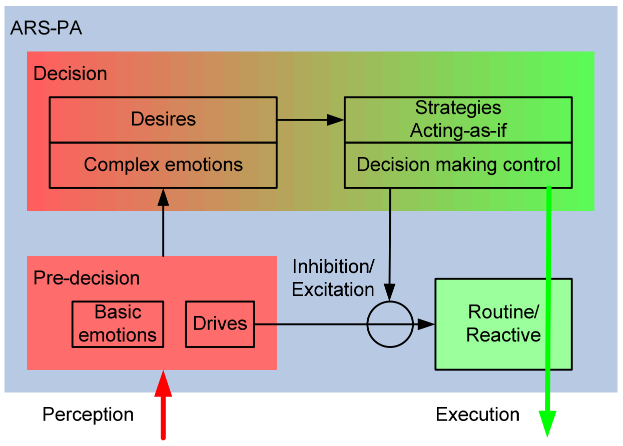


Figure 12: ARS-PA model [Bur07]

Figure 12 shows the ARS-PA part including basic emotions. It has to be emphasized here again, that the basic emotions are part of both the ARS-PC and ARS-PC part. However, basic emotions together with drives form the so-called pre-decision module. The pre-decision module can cause very fast reactions if the path to the reactive module is activated. The decision module obtains valuated information about the environment from the pre-decision module. The decision module is responsible for long term decision making.

In the decision making module, complex emotions and desires are used. Typical human complex emotions are hope, disappointment, joy, or gratitude. The goal of the decision-making unit is to satisfy the needs of an individual. Desires are experienced situations that lead to the satisfaction of a need. These desires push the individual to reproduce such experienced situations. The strategies module plans the actions or reactions to a certain situation. It simulates what would happen if a certain action were to be taken in the current situation (Acting-As-If). The decision making control module chooses one action from different actions planned by the strategy module[5]. It chooses the action which best satisfies the needs.

The decision making control module has the ability to inhibit reactions which were triggered by basic emotion systems. The routine module controls action series which have been learned by the individual. Typical examples of routine actions by humans are walking, cycling, or swimming. Once it has learned how to do it, the brain stores the actions necessary for the execution of these actions.

It has to be stressed here that the decision making module does not only react on input from the perception module (ARS-PC) but has also the ability to take actions without input from the external world. The drives and desires can push the individual to take actions. Therefore, the ARS project has to be seen as a loop, as mentioned in the beginning of Chapter 1 and depicted in Figure 1.

2.2.5 Résumé

As seen in the previous chapters the ARS model consists of two main parts, the ARS-PC part and the ARS-PA part. For both parts the functionality of the human brain serves as model. The ARS-PC part uses a neuro-psychoanalytical model; the ARS-PA part uses a psychoanalytic model. Since neither psychoanalysis nor psychology, nor neurology have models which describe the functionality of the human from low-level functions up to high-level functions, models of these disciplines were taken which can be combined for a consistent and unitary ARS project. In particular in this work, the technical model of the ARS-PC part has been derived from the neuro-psychoanalytic model presented in Chapter 2.1.

The output of the ARS-PC part is a world representation which uses the ARS-PA part as basis for decisions. Both parts are influenced by a valuation system of the so-called e-systems. Each of the e-systems, which are comparable to basic emotional systems of the brain, is influenced by the ARS-PC and ARS-PA parts. An overview of the main tasks of the ARS system is illustrated in Figure 10 above.

As mentioned previously, this work deals mainly with the ARS-PC part. Based on sensor data it creates a world representation of the outer world by using symbolization. The symbolization is split into three

[5] For the evaluation of the planned action/reaction the emotional systems are used (basic and complex). The planned actions are transformed into expected perceptions which are then evaluated. This feedback is not included in Figure 12

hierarchically ordered layers, the microsymbol layer, the snapshot symbol layer, and the representation layer as outlined in Chapter 2.2.3. Each layer is represented by its own symbols. In this work a symbol should be understood as a collection of information. Figure 13 shows an example of a tree of symbols.

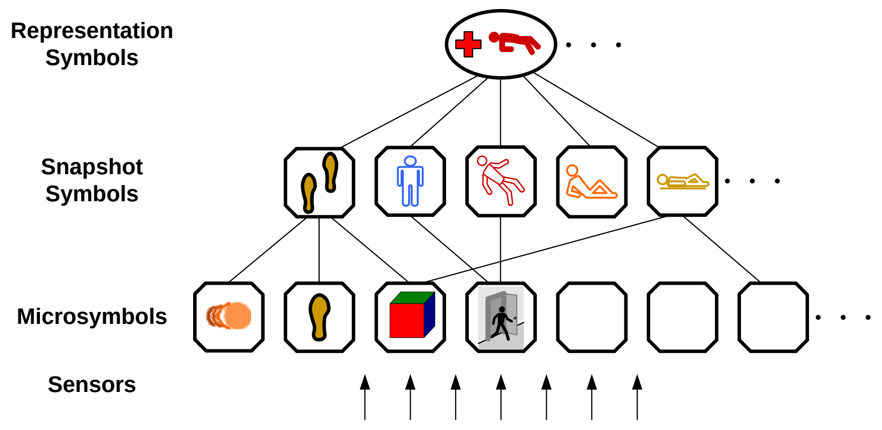


Figure 13: Tree of symbols in the ARS-PC project

In the hierarchical model the lower-layer symbols are building blocks of the higher-layer symbols. The microsymbols are generated from sensor data. A snapshot symbol is a synthesis of microsymbols. The representation symbols have an even higher abstraction, which allows the description of simple scenarios. Symbols, hierarchy concept, and scenarios of the ARS project are defined in Chapter 4 and Chapter 5. The next chapter describes how data from sensors in a building can be accessed in case more than one building automation system is used.

3 Provision of Sensor Data

It is intended that the ARS system is an extension of Building Automation and Control Systems (BACS) as defined in CEN TC247 (European Committee for Standardization Technical Committee 247 “Building automation, controls and building management”) [TC247]. In [Tam03] it is proposed to expand[6] the ISO/OSI model [OSI94] in order to implement the Perceptive Awareness Model (PAM), which is depicted in Figure 14. The current fieldbus technology is based on the ISO/OSI model. The basic access functions are interoperable functions to access the sensors and actuators of the fieldbus systems. Above the data abstraction layer, which is described in this chapter, the functions of the ARS model (perception, recognition and reaction) are illustrated. The main task of the data abstraction layer is to hide the specific representation of data and services of the underlying communication protocols.

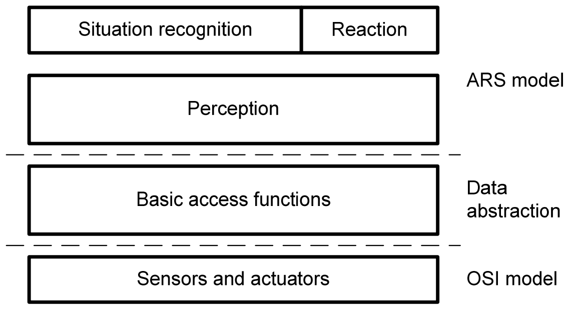


Figure 14: Perceptive Awareness Model according to [Tam03]

In the last years the fieldbus systems KNX [ISO14543], CNP [EN14908-1 and EN14908-2] and BACnet [ISO16484-5] were standardized by the international standardization. Within these standards it is possible to work more or less seamlessly [Bur05] because standard data types, standard services, standard semantics and profiles are the main elements to ensure interoperability [Kab02]. Unfortunately, the interoperability is only addressed within one standard. The lack of interoperability between different BACS leads to additional costs in installation and maintenance [Die01a]. Moreover it is not possible to develop applications that are based directly based on different BACS.

The classical approach to overcome the missing interoperability between different BACS is to use gateways. Their task is to “translate” between different protocols. This translation usually involves a certain degree of loss. However, the fieldbus systems have been developed for special application areas (field

[6] The ISO/OSI model is an abstract description of communication network protocols. The extension in [Tam03] covers high data processing and does not refer to the layers of the ISO/OSI model.

level, automations level, and management level[7]), and therefore data types and services usually differ between the fieldbus systems. An application designer normally develops an application for a fieldbus system, and therefore used its special data types and services. However, applications like the ARS system need data from all hierarchical levels, and therefore usually from different types of fieldbus systems.

This work develops two approaches, which avoid the problems caused by the use of gateways. The first approach describes a gatewayless communication between CNP and BACnet [Bur04]. The second approach is an abstraction layer, which hides the fieldbus specific data representation and services.

Other approaches have been developed for industry automation. [Sau07, Lob06, and Sau05] give a technology survey how fieldbus can be integrated into IP-based networks. A discussion of different approaches can be found in [Wol01, Wol02, Lob02, and Cal03]. However, [Sau04] maintains that no ideal solution for all occasions has been found yet. The following chapters describe the fieldbus systems CNP and BACnet. Further, the ISO standard for the mapping between CNP and BACnet is described briefly in order to develop the gatewayless communication approach and the data abstraction approach, which is needed to overcome the use of gateways.

3.1 Control Network Protocol CNP

CNP [Loy01] uses as transport protocol the LonTalk protocol, which is standardized by the American National Standards Institute (ANSI) in the standard EIA/CEA-709.1 (Electronic Industry Alliance/Consumer Electronic Association) [EIA709] as the so-called Control Network Protocol (CNP). Further it is specified in the European standard series EN 50090 [EN14908-1] and [EN14908-2]. Additional to this standard the Interoperability Guidelines for layer[8] one to six [LIGP05] and the Interoperability Guidelines for the application layer [LIGA05] define standards of configuring, managing and communicating components on a CNP network in an interoperable way.

In the CNP, two types of communication objects are introduced, the Network Variables (NV) and the Explicit Messages. The NVs can be used within an application program similar to variables known from programming languages like C. If a new value is assigned to a NV within the application, the new value is propagated automatically to all NVs that have a so-called binding with that NV. NVs can have bindings with NVs from the same node as well as with NVs from other nodes. The protocol stack automatically generates the messages exchanged between the nodes.

Explicit Messages have to be managed by the application program. This means that the messages exchanged between nodes have to be composed by the application program. That includes addressing as well as the type of service. Explicit Messages have a message code, which defines how the explicit message is processed by the protocol stack. In the CNP the range of the message code between 0 and 62 is defined for application messages, which can be used by the application program. The range between

[7] The levels are defined in ISO16484-2.

[8] The layers correspond to the ISO/OSI reference model (International Organization for Standardization/Open System Interconnection) [OSI94].

128 and 255 is reserved for NVs, and is managed by the protocol stack. The range between 64 and 78 is used for foreign frames, which can be used by other protocols to tunnel messages through a CNP network. In the BACnet standard the CNP is defined as a transport protocol using these explicit messages (see Chapter 3.2).

The communication on a CNP network according to the Interoperability Guidelines is based on NVs. To ensure that all partners interpret the NVs in the same way, the Standard Network Variable Types (SNVTs) has been defined. These communication objects are published in the SNVT Master List [LSML03]. For CNP a various number of transport media are defined. Widespread media are TP/FT10 (Twisted Pair/Free Topology), which allows a free topology cabling, twisted pair or transmission over the Internet Protocol (IP). Therefore, all media that are defined for IP can be used as transport media as e.g. “Ethernet” [ISO8802].

3.2 BACnet

BACnet was first defined as an ANSI/ASHRAE (American National Standards Institute / American Society of Heating, Refrigerating and Air-Conditioning Engineers) standard [ASH135] and is now also standardized by ISO in [ISO16484-5]. It was developed mainly for the management level, but can also be used in the automation/control level or field level. Data are represented as objects, which are characterized by their properties. To increase interoperability, standard objects are defined. Standard objects have mandatory properties and optional properties. An example of mandatory and some optional properties of the Analog Input Object are presented in Table 1. An Analog Input Object would be used to represent the value of analogue sensor, e.g. temperature sensor. The name of the property is listed in the first column of the table, and the data type of the property is shown in the second row. CC in the third column stands for Conformance Code. It describes whether the property is required (R) or optional (O).

Table 1: Example of properties of the Analog Input Object

Property Identifier	Property Data type	CC
Object_Identifier	BACnetObjectIdentifier	R
Object_Name	CharacterString	R
Object_Type	BACnetObjectType	R
Description	CharacterString	O
Device_Type	CharacterString	O
Present_Value	REAL	R
Status_Flags	BACnetStatusFlags	R
Event_State	BACnetEventState	R
Out_Of_Service	BOOLEAN	R
Units	BACnetEngineeringUnits	R

In the following, a short explanation of the concept of the BACnet objects is given. The only optional properties presented in Table 1 are the properties Description and the property Device_Type. They can contain a general description and a description of the connected sensor. All other properties in Table 1 are mandatory. The properties Object_Identifier, Object_Name and Object_Type are used to identify a BACnet object within a BACnet network. The property Present_Value represents the value, which is measured by the sensor. In case of a temperature sensor this would be temperature. The unit of the value is defined by the property Units. In the BACnet standard a list of units is defined. The property Status_Flag, the property Event_State, and the property Out_Of_Service describe the status of the BACnet object.

To access the properties of a BACnet object, so-called BACnet services are defined. Depending on the application of the BACnet device, different services are offered. The services follow the client-server architecture where the one device initiates the communication (request, BACnet type A) and the other node replies (response, BACnet type B). Table 3 shows the example of a BACnet Read Property service. On the left side the arguments of the request are shown. The second row shows whether the argument is mandatory (M) or a user option (U), which may not be provided. The next rows show the arguments of the response and the occurrence of the arguments.

Table 2: BACnet Read Property Service

Request		Response	
Argument	M	Object Identifier	M
Object Identifier	M	Property Identifier	M
Property Identifier	M	Property Array Index	U
Property Array Index	U	Property Value	M

If the value of a temperature should be read by e.g. an operator workstation, this device would act as the client (BACnet type A) and would send the Read Property Service (type A) to the sensor node where the temperature sensor is attached. To read the value of the temperature value the Object Identifier and the Property Identifier of the BACnet object associated with the temperature value have to be addressed. The node with the attached temperature sensor acts a server (BACnet type B) and sends the Read Property Service (type B) as response. If the value could be read, the arguments from the request and the value are returned, otherwise an error is returned.

In addition to the Read Property Service, BACnet offers a Write Property Service to change the value of a property (e.g. the value of a ventilator output). Multiple read and write services allow reading or writing more than one property at once. The so-called change-of-value (COV) service allows an automatic notification when the value of a property changes. Event and Alarm services allow the notification in case of an event or alarm.

Although BACnet defines its own transport protocol and media, namely MS/TP (Master Slave / Token Passing), it uses other communication protocols. These protocols are IP and “Ethernet” [ISO8802] and the CNP. It uses the foreign frame concept of CNP as briefly described in Chapter 3.1. This allows communication between CNP and BACnet without a gateway. Gateways are the concept which is used to map data points between different communication protocols as explained in the next chapter. The BACnet standard defines a mapping between CNP and BACnet as well as a mapping between BACnet and KNX. These mappings should ease the work of system integrators in case different communication protocols are used and the data points have to be mapped between these protocols.

3.3 Mapping CNP to BACnet using gateways

In building automation systems different communication protocols may be used. In order to allow communication between the different protocols gateways have to be used. A three-layered hierarchical model is proposed by the international standard for building automation and control systems [ISO16484-2]. The task of the two lower layers, the field level and the automation or control level, is to control the application in the field. The top level is the management level, which interconnects several control networks and allows the operator to manage the building. In the standard [ISO16484-2], there is no assignment of fieldbus systems to hierarchical levels. Nevertheless, BACnet is mainly used in the management level, whereas CNP is used in the automation and field level. Figure 15 shows a traditional approach to connect CNP and BACnet.

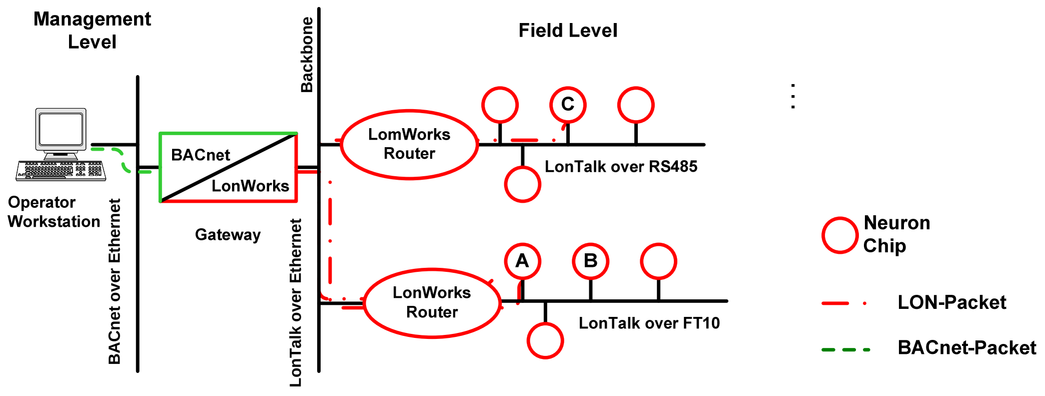


Figure 15: Communication with gateways

In the management level the BACnet is used. In the field level the CNP with different physical media (RS 485 and FT 10) is used. These two subnets are connected with CNP routers with the CNP backbone. Messages between nodes within a subnet are blocked by the CNP router and therefore not forwarded to the CNP backbone. If node A (subnet FT 10) and C (subnet RS485) of Figure 15 have to communicate with each other, the messages are routed via the CNP backbone from one CNP subnet (RS 485) into the other subnet (FT 10).

The BACnet/CNP gateway allows communication between the two protocol worlds. If the BACnet operator workstation has to poll a value from the CNP node A, the BACnet request has to be translated by the BACnet/CNP gateway into a CNP poll request. Via the CNP backbone and the CNP router this message is routed to the FT 10 subnet. The answer of the node takes the same way back and again has to be translated in the BACnet/CNP gateway before it reaches the BACnet operator workstation.

This approach is used nowadays in building automation systems. Nevertheless, it has some disadvantages, which originate from the use of gateways. The gateway is a very complex device and therefore expensive not only to purchase but also in operation. It needs an application-specific configuration and has to be adopted with every change in the network. Since the different protocols use different representations and types of data, usually there is a loss of information and precision during protocol conversion. If complex data type conversion is necessary the performance of the installation may suffer under high load.

Another major problem is that the gateway is a single point of failure. If the gateway is out of order, no communication between the management level and the field level is possible. Using a redundant gateway, which can ensure that communication is possible even if other gateway is out of order, increases the costs for installation, configuration, service, and operation.

To reduce or even eliminate these problems, communication without gateways can be used, provided that one protocol standard can use the transport protocol of the other standard. The next chapter describes how that this can be done with the protocols BACnet and CNP.

3.4 Gatewayless Communication between CNP and BACnet

As mentioned above, the concept of gatewayless communication between two different communication protocols is only possible if one of the communication protocols can use the transport protocol of the other communication protocol. In the BACnet standard [BAC04] this possibility is defined in clause 11. This approach minimizes or even eliminates the drawbacks of using a gateway, because of the lack of the gateway. It is clear that still a protocol conversion is necessary, but this conversion is done in the nodes of the field-level.

Such a node is a so-called NeuBAC node. A NeuBAC node is a node which implements a CNP stack and additionally the necessary parts of the BACnet protocol stack. In Annex L of the BACnet standard [BAC04] profiles of BACnet standard devices are defined. In these profiles it is defined, which BACnet Interoperability Building Blocks (BIBB) have to be implemented in a device. The BIBBs are defined in Annex K and describe a certain BACnet service and whether it is the client side implementation or the server side implementation of the service. For the BACnet Smart Sensor only the server side implementation of the read property service has to be implemented. For the BACnet Smart Actuator the server side implementation of the write property service has to be implemented in addition. A NeuBAC node therefore should at least implement the read property service and the write property service of the BACnet standard. Figure 16 shows again the network topology as used in Figure 15 and shows a network without a gateway.

Figure 16 differs from Figure 15 is several points. The BACnet operator workstation and the Ethernet backbone are left unchanged, but in the backbone not only BACnet messages are forwarded, but also CNP messages. The BACnet/CNP gateway and the Ethernet backbone for the CNP are omitted. Combined CNP/BACnet routers replace the CNP routers. A Combined CNP/BACnet router is a router that can route CNP messages as well as BACnet over CNP messages. Since the messages are routed independently from each other, such a device could also be composed out of a CNP router and a BACnet router. However, it should be mentioned here, that in the Ethernet backbone the BACnet messages either conform to the BACnet over ISO 8802.3 standard or conform to the BACnet/IP standard. The CNP messages conform to the EIA 852 (IP/Ethernet channel for CNP) standard [EIA852].

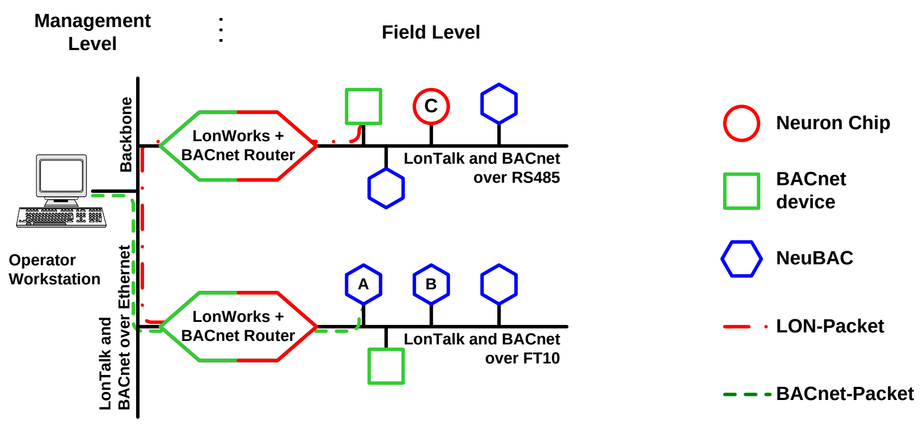


Figure 16: Gatewayless communication between CNP and BACnet

In the field-level, pure BACnet nodes, NeuBAC nodes, and/or pure CNP nodes can be used, even simultaneously. It is clear that a CNP node can communicate only with other CNP nodes or NeuBAC nodes, and a BACnet node can only communicate with other BACnet nodes or NeuBAC nodes. If the node A and C have to communicate with each other, the CNP messages are used as defined within the CNP standard. A request from node A is sent to the CNP router, which routes the message into the Ethernet backbone as an EIA 852 message. The CNP router for the RS 485 network translates the packet back into a CNP message and sends it into the subnet where the CNP node C is located. The answer to the request is routed the same way back.

If the BACnet operator workstation has to poll a value from node A, which is in this approach a NeuBAC node, the communication differs from communication with a gateway. The BACnet operator workstation sends out a BACnet message into the Ethernet Backbone. This message is routed by the BACnet part of the CNP/BACnet router into the subnet, where node A is situated. The message in the subnet is still a BACnet message, following the BACnet standard. The NeuBAC node A receives the message and can interpret it as request for a certain value. The response is sent as a BACnet message back following the same way to the BACnet operator workstation.

Using this approach, the gateway as a single point-of-failure is eliminated. The configuration of the network is easier, because no gateway has to be configured. The additional configuration effort within the routers can be reduced using learning algorithms within the routers. To configure the gateway the network integrator needs knowledge about the applications in order to configure a gateway. Using NeuBAC nodes, this problem is delegated to the application designer, who has this knowledge in the first place. He can design the NeuBAC node in a way that the node provides all necessary communication objects for both protocols. Therefore, the loss of information due to protocol conversion is zero. In Annex H of the BACnet standard a mapping between CNP and BACnet is defined which can ease the work of the system integrator for assigning the addresses and identifier.

A general-purpose gateway has to fully implement the stacks of both protocols. A multi-protocol node like a NeuBAC node only implements the functions required for the application. This reduces costs during development, functional testing, conformance testing, and certification. A NeuBAC node, which is used as a BACnet smart sensor device, only has to implement the read property service. Since one backbone and the gateway are omitted, the response time of the system can be increased. A gateway usually has a high delay time because the payload of the message has to be converted.

A NeuBAC node can also be used in pure BACnet installations, which do not use the CNP protocol as an application protocol, but only as transport protocol. In such a case already existing CNP devices can be used as BACnet devices without changing or redesigning the hardware.

There are also some limitations and restrictions when using NeuBAC nodes. The maximum length of the Network Protocol Data Unit (NPDU) is limited to 228 Bytes. If ARCNET[9], MS/TP, point-to-point is used the maximum NPDU length is 501 Byte, with ISO 8802.3 (Ethernet) or BACnet/IP 1497 Bytes allowed. But again, the payload limit is not an issue for simple devices. In BACnet analogue values are transmitted as single precision floating-point values according to ANSI/IEEE Standard 754-1985 [IEEE754]. One value requires therefore four bytes plus protocol overhead.

Typically, a CNP node uses the Neuron Chip hardware [Mot97], which has very limited resources. A typical node has a RAM ranging from 1 to 4 Kbytes; the EEPROM is available up to 62 Kbytes. Nevertheless, the hardware architecture and the programming model of the Neuron Chip are very specialized, so it is possible to meet the requirements of simple devices. However, if the resources of a Neuron Chip are not sufficient, alternative hardware based on the LC3020 [LC3020] can be used.

As described in Chapter 3.2, BACnet uses the concept of foreign frames defined in CNP. A CNP node usually ignores such messages, unless the application programmer develops a program, which deals with these messages. According to the CNP standard not only BACnet can use the message code 0x4E but also other protocols or applications. This may lead to malfunction of the system if beside the NeuBAC nodes other devices send CNP messages with the message code 0x4E. Therefore, measures have to be taken to ensure that only BACnet messages use this message code.

The probably most restrictive limitation for practical use is that this concept is only suitable for BACnet and CNP, but not for other protocols like KNX. If a more common approach has to be used, an abstraction of data as presented in the following chapters could be used.

3.5 Abstraction of Data

Usually, interoperability within a system can only be achieved, if only one (communication) standard is used. Several projects have tried to overcome this problem. The NOAH project (Network Oriented Application Harmonization) [DiS00, Mel99] tried to harmonize the device profiles of the EN 50170 [EN50170]. Other standardization bodies also tried to harmonize the communication objects, like the CEN TC247 [Fis99]. CEN TC 247 [TC247] has also been working on the BACnet onto CNP mapping mentioned in Chapter 3.4.

The OSGi (Open Services Gateway Initiative) has carried out a high-level data point abstraction of EIB data points [Kas04]. It is based on multi-threaded environments like Java. In [Vel03] a model is described for Internet-based access on control and automation data using virtual industrial devices and a virtual

[9] Attached Resource Computer NETwork, is a local area network protocol, defined in ATA/ANSI 878.1-1999

industrial protocol. They are based on SOAP (Simple Object Access Protocol) and XML (Extensible Markup Language). Such implementations are only suitable for some nodes within a system like an operator workstation because Java SOAP or XML-based protocols need a degree of computational power that usually cannot be provided by embedded systems. However, in the field-level only nodes can be integrated which are cost-efficient and therefore they have limited computational power. To overcome this problem a proposal is made here and in the following chapter for data-point abstraction suitable also for embedded systems like hardware based on the LC3020 [LC3020].

The abstraction of data of communication systems is necessary, since systems like the ARS system need access to all data available within the system. But also in modern building automation applications like energy saving applications it is necessary to combine data from different industries and automation levels in order to find optimal strategies for energy distributions [Pal03]. But also applications like shadowing, lighting, and climate control belong to different industries. In a conventional building automation system, shadowing and lighting would be realized within industry lighting, whereas climate control would belong to the HVAC (Heating, Ventilating, Air-Conditioning) industry. But physically these two industries are linked together. Sunlight heats and illuminates the rooms where it shines into. So the sunblind can influence both light control and climate control of the room. CEN/TC 247 prepares a standard for energy performance in buildings [prEN15232], which shows the interconnection between solar gains, heating, cooling and lighting energy for buildings.

However, applications which have to access data-points in more than one industry or communication protocol can only be developed efficiently if the data point representation and the services of the fieldbus standard are abstracted. The abstractions have to be done in a way that the application can be developed independently from the underlying communication protocol. The abstraction of data points requires firstly the analysis of different representations of data-points of different communication protocols in order to define a representation which is suitable for a data abstraction for embedded systems.

3.6 Data point representation

In the area of building automation there are different fieldbus and communication systems using their own characteristic representation of data and services to transfer the data. In Chapter 3.1 and Chapter 3.2 the communication protocols CNP and BACnet are briefly described. Additionally the protocols SOAP/XML, oBIX (Open Building Information Xchange) [oBIX05], BACnet/WS (Web Services) OPC XML-DA [OPC05] were investigated in this work in order to develop a data abstraction model.

The present value of a data point is its main characteristic. Additional properties describe the value in more detail. They are classified into three main groups. The first group is value characteristics, which describe the meaning of the value. The second group describes the application characteristics. Properties of this group describe parameters of the application. User characteristics comprise the third group. They describe, mostly verbally, characteristics of the data point and are usually only used by the user of the system. Table 3 shows the classification and properties assigned to the group.

The properties of the value group describe how the present value is represented. The data type of common communication protocols for building automation are Boolean, integer (8, 16 and 32 bits), and floating point (e.g. single or double precision of the ANSI/IEEE floating-point standard [IEEE754]). The data type normally determines whether the data point is an analogue or a binary data point. Non-numeric values like strings have not been investigated in this work.

Table 3: Groups of value properties

Group	Properties
Value	Data type, Range, Unit, Resolution, Precision, Timestamp
Application	Direction, Status, Poll cycle, Change of value interval
User	Name, Description, Unit Text

The data type on the one hand can limit the range of the present value, and also the application. For example, a sensor might have only a certain range or the value is not meaningful as it would be for clock. Assuming the time is transmitted as an array of integers, where the first value represents the hours, the second the minutes and the third the seconds, the meaningful range of the first value would be 0 to 23, and 0 to 59 in case of the second and third value. Additional communication protocols may define special values within the range of the data type, which indicates that the value is not valid. CNP e.g. defined for some SNVTs, which use 16-bit integer as data type, the value 0xFFFFh as invalid value. In the ANSI/IEEE 754 standard for floating point data types, so-called NaNs (Not a Number) are specified, which define invalid values. However, whether explicit or implicit, the range of the value has to be defined for every data type.

The property unit describes the physical unit of the data point. The metric system is standardized in the International System of Units SI (from the French: Système international d'unités) in ISO 1000 [ISO1000]. But almost all communication protocols investigated in this work allow also non-SI units. Similar to the property range, the properties resolution and precision describe how the present value has to be interpreted. The property resolution defines the smallest interval e.g. measured by a sensor, the property precision describes the number of the meaningful digits of the present value. The timestamp indicates when the value was generated or updated.

The application specific characteristics are classified in the group application. Usually in communication protocols there is a distinction between input (the value is only readable) and output (the value is readable and writeable). Additionally, calculated values (readable and writeable), which have no physical representation in the field, can be seen as a separate category. The property status describes the "health" of the data point. Finally, the investigated communication protocols provide status information, which distinguishes between "normal" and "alarm". However, this information can also be encoded in the present value of the data point, as mentioned above (see property range).

The property poll cycle defines the time between two poll requests to a data point. Several factors can limit the minimum poll cycle, like the available bandwidth on the fieldbus, the resources on the fieldbus node, or the conversion time of the analogue digital converter for analogue sensors. On the other hand, applications like controllers or visualizations have an upper limit for the poll cycle in order to fulfill their tasks properly. If the communication protocol provides only asynchronous communication services, this parameter can be used to emulate synchronous communication services. If a change of value (COV) service is provided by the communication protocol, the property COV interval specifies the hysteresis for analogue data points.

The third group comprises the user-specific characteristics and is usually a verbal description. It can be distinguished between the property name, which is mostly a unique name of the data point within the system, the property description, providing a verbal description of the data point and the property unit text, which defines the physical unit of the data point value.

Not all of the properties described above can be found in the investigated communication protocols. Properties might be only optional properties, which means that it depends on a certain type of implementation whether the property is present or not. If the property is available, different ways to access it have been identified. Either it is transmitted with every message (like the timestamp of the present value), or it can be found out with a certain request (e.g. description), or it is not accessible from the data point, but from other sources like a configuration database or file.

Using the properties identified in this chapter, a data point abstraction layer can be developed for centralized applications like visualizations or gateways, but it is especially suitable for decentralized applications like the ARS system, which need the ability to access all data available in the building, regardless of the type of communication protocols used in the building. The next chapter describes a data point abstraction layer for embedded systems.

3.7 Abstraction of data for Fieldbus Nodes

An abstraction of data points for fieldbus nodes allows developers to design decentralized applications independent from the communication protocol. This means that applications can be developed without having any knowledge of the used fieldbus system. In this work a hardware based on the LC3020 [LC3020] was used for the fieldbus node. This platform provides as transport media 100base-T Ethernet and various CNP transceiver variants (among FT 10, RS 485), which allows the use of IP-based communication protocols like EIA-852, BACnet/IP, KNX/IP, and MODbus/IP. Since this platform provides all standardized communication protocols for building automation in Europe (BACnet, CNP, and KNX), it is highly suitable for the implementation of an abstraction as proposed in [Tam03].

The results arrived at in Chapter 3.6 are used here to develop a model for data point abstractions suitable for fieldbus nodes. An overview of the modules and interfaces is given in Figure 17. It shows the applications and configuration modules on the highest layer, which use the data access services and the configuration services of the Data Point Abstraction Layer (DPAL). The DPAL uses the DPAL native system services of the adaptation modules. For each communication protocol a dedicated adaptation module is necessary, which provides these DPAL native system services. For their part, these modules use the services of the stacks of the communication protocols.

DPAL plays the central role in this architecture. It is responsible for representing the values of the data points and provides the appropriate services for accessing them. Instead of developing a completely new model, the BACnet model is used and modified in order to meet the requirements defined in Chapter 3.6. BACnet is already a proven model in many building automation installations. The BACnet standard already includes mappings that can be adopted and reused for the DPAL. Annex H defines the mapping between BACnet and KNX and BACnet and CNP, and Addendum C of the BACnet standard defines the mapping between BACnet and BACnet/WS.

As in BACnet, the data points are represented as objects. The following definition of the model uses as explanation the analogue, binary, and multi-state objects. However, the model is designed to implement also other objects like string objects, data-time objects or node objects. Analogue objects are used to represent analogue values, binary objects to represent Boolean values. The multi-state objects can represent a defined number of discrete values as it would be used for a switch with more than two possible positions. String objects are used to represent character strings, e.g. for a display text, and data-time objects are used to represent a date and/or time. The node object is not a data point with a value. It is used to structure the data point as described later in this chapter.

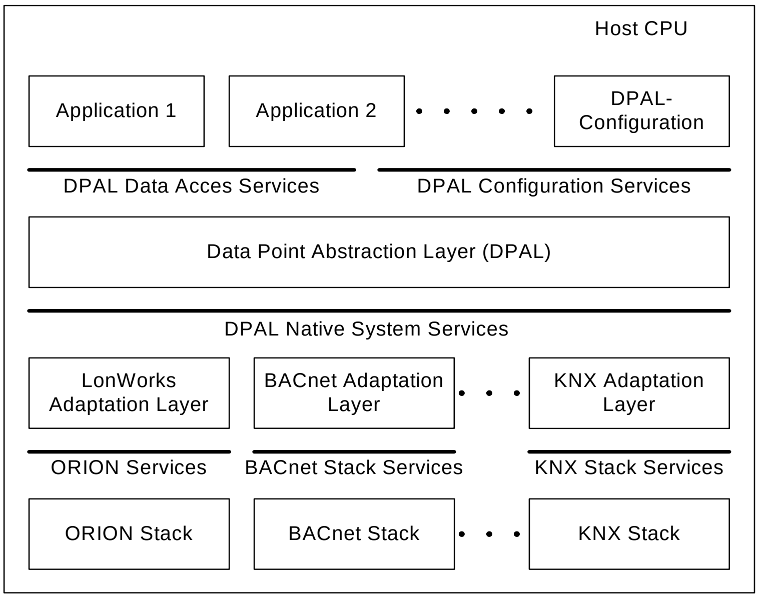


Figure 17: DPAL (Data Point Abstraction Layer) overview

DPAL uses properties similar to BACnet. But different from BACnet, these are divided into general static properties (see Chapter 3.7.1), general dynamic properties (see Chapter 3.7.2) and extended (static) properties (see Chapter 3.7.3). The general static properties and the general dynamic properties consist of properties which are common for all data point objects. The extended properties are specific for the type of the data point object. This splitting of properties into three groups has been done to structure the properties to ease implementation and handling of the data points. In the following chapters these properties are described.

3.7.1 General Static Properties

The general static properties define the behavior of the data point. In Table 4 the general static properties are listed. The general static properties are defined during the creation of the object and are fixed, except for the poll cycle, which can also be changed during runtime (see later in this chapter). The property object type defines the type of the data point. Table 5 shows the object types defined for DPAL.

Table 4: General Static Properties

Property	Fixed/changeable
Object Type	Fixed
Direction	Fixed
Data Type	Fixed
Poll cycle	Changeable by configuration instance
DPAL name	Fixed
Native name	Fixed
Description	Fixed

Table 5: DPAL object types

Object Type	Description
Analogue	Value is an analogue value
Binary	The value can be either zero (FALSE) or one (TRUE)
Multi-state	Represents objects with a defined amount of discrete states
Node	Has no value, can be used to be a parent object of other objects
User	The type of the value is not known
String	The value is a string
Date/Time	The value is data and time

The analogue object type has a value represented as an analogue value. A typical data point for such a type might be a temperature sensor. To represent a simple switch as a data point, the binary object is suitable, because its data type is Boolean. The multi-state object should be used if more than two discrete values are necessary as e.g. a multi-state switch. A node object has no value at all; it is used to group data points, as described later in this chapter. The data type of the user object type is not defined within DPAL. This object type can be used for data points, which do not fit to any other object types. The string object type is intended to be used for data points which have to represent strings, as with devices with displays. The date/time object type represents date and time and is therefore suitable for data types like time sources within a system.

The property direction of the general static properties can be input, output, value, or node. If the property direction is input, the data point is only readable. Therefore, this direction should be used for sensors, where the value of the sensor can be read, but does not change. Actuators should use the direction output. This value is readable and writeable. The direction value is also readable and writeable. The difference from the direction output is that the value is intended to represent a calculated value, which has no physical representation (actuator) in the field. The direction node is used for the object type node. It indicates that there is no value and therefore can neither be read nor written.

Table 6: DPAL data types

Data Type	Description	Possible Object Types
Double	IEEE 754 double precision float point (64 bit) [IEEE754]	Analogue object
Unsigned	unsigned integer (32 bit)	Multi-state object, analogue object
Integer	signed integer (32 bit)	Multi-state object, analogue object
Boolean	Boolean (1, 0)	Binary object
String	string of ASCII characters	String
Date Time	Time information including date and time	Date/Time

Table 6 shows the defined data types in DPAL. Defining data types independent from the object type allows a more flexible configuration for analogue and multi-state objects. The data type of an analogue object may be a double, an unsigned integer, or a signed integer. Multi-state objects have either signed integer values or unsigned integer values. Especially in fieldbus nodes with limited resources the flexible use of data types can reduce the conversion effort between different data types. Assuming that the output of a sensor has only integer values, as well as the input of an actuator depending on that sensor, the control loop can be calculated with integer values instead of floating point values.

The property poll cycle of the general static properties is only used if the underlying communication protocol does not provide synchronous services. In that case, the data point is updated periodically with the update rate defined with the property poll cycle. The poll cycle is the only general static property which can be changed during runtime. The property DPAL name is a unique name of the data point within the DPAL implementation, and the native name is the name of the data point of the underlying communication system. The property description is intended to hold a verbal description of the data point.

3.7.2 General Dynamic Properties

In contrast to the general static properties described above, the general dynamic properties are not fixed. They are updated continuously. In Table 7 the general dynamic properties are listed.

Table 7: General dynamic properties

Property	Fixed/Changeable
Timestamp	Changes automatically when the value of the data point changes
Status	Changeable

The property timestamp is updated whenever the value of the of the data point changes. Data points received from the native communication protocol have two possibilities to receive a timestamp. If the communication protocol provides a timestamp, that timestamp should be used. If such a timestamp is not available, the DPAL implementation should assign the timestamp of the system at the point of time when the new value has been received. For values generated by a DPAL application, the timestamp has to be handled similarly. If the application provides a timestamp, that timestamp should be used. If not, DPAL should use the system time at the moment when the application sends the value to DPAL. The general dynamic property status gives information about the “health” of the data point. Table 8 shows the defined values of the property status.

Table 8: DPAL status

Status	Description
Not configured	The DPAL data point is not configured
Normal	DPAL point is in normal operation
Out of Service	DPAL point is out of service
Overridden	The value is overridden
Invalid	Value is invalid
Offline	No value set

The general dynamic status not configured is assigned to the data point’s properties during configuration and when the data point is not configured properly. If the data point is in normal operation, the property status is normal. The status out of service indicates that the data point is configured properly, but for some other reason the data point is not in operation. Such a reason might be that communication is faulty due to an error in the cable system. If the value of a data point that has been received from a native communication protocol has been changed by an application, the status overridden is used to indicate this.

If the value is invalid the status invalid is used. A value might be invalid for several reasons. Protocols like CNP (see Chapter 3.1) use special values for indicating invalid values. If a conversion of the value between data types or units is necessary, it might happen that the value range is not suitable and therefore the value cannot be converted. The data point can be set offline by the application. In that case, the status is set to the value offline.

3.7.3 Extended Properties

The extended properties hold the properties, which are specific to the object type. Except for the node object type all object types defined in Chapter 3.7.1 have extended properties. In the following chapters these properties are briefly described.

Extended Properties of the Analogue Object

The extended properties of the Analogue Object are listed in Table 9. They define the value of the data point itself and how this value has to be interpreted. The property value of the extended properties of the analogue object holds the value of an analogue data point. The data type of the value is defined in the general static properties of the data point (see Chapter 3.7.1). Since the value of the data point can change, it is clear that this property has to be changeable. The property unit defines the unit of the data point. It uses the seven exponents of the SI base units [ISO1000]. With dimensional analysis, every physical unit can be expressed in this way. To have the possibility to distinguish between dimensionless quantities like radian, percent, or decibel an additional enumeration is used. For better usability a text string may be added, holding the unit as characters. In Table 10 examples are given for velocity [v], power [W] and percent [%].

Table 9: Extended Properties of Analogue Object

Property	Fixed/changeable
Value	Changeable
Unit	Fixed
Resolution	Fixed
Precision	Fixed
High limit	Fixed
Low limit	Fixed
COV increment	Changeable

Table 10: Examples for dimensional analysis

Unit	m	kg	s	A	K	Mol	cd	Enum	text
v	1	0	-1	0	0	0	0	SI	“m/s
W	2	1	-3	0	0	0	0	SI	“W”
%	0	0	0	0	0	0	0	PERCENT	“%”

The unit velocity [v] is defined as m/s. Therefore, the factor of meter (m) is 1 of the time (s) is -1. All other factors are zero. SI in the Enumeration field indicates that the unit is defined with the SI units. The text field holds the unit as character and is therefore “m/s”. The same applies to the power [W]. The factor of meter (m) is 2, for mass (kg) and time (s) is -3. In the Enumeration field SI indicates that the power is expressed in SI units, the text field holds “W”. Percent [%] is dimensionless. Therefore all factors for all SI base units are zero. PERCENT in the enumeration indicates that the value shall be interpreted as percent. The text holds the sign for percent “%”. Since the unit of a data point does not change, the property is not changeable.

The extended property resolution of an analogue object describes the smallest possible increment of the value. The property precision describes the number of meaningful digits of the value. The properties high limit and low limit describe the maximum and minimum of the value. These four properties are specific for the data point and will not change during operation. The property COV increment defines the delta of the value, which causes a COV (change of value) notification. For details on COV-notification see Chapter 3.9. This property can change during operation and is therefore changeable.

Extended Properties of the Binary Object

The extended properties of the binary object are listed in Table 11. They define the value of the data point itself and how this value has to be interpreted. The property value holds the value of the data point. According to Table 6 the only data type possible for the binary object is Boolean. The properties active text and inactive text are a description of both possible values. For a simple switch the active text might be “on”, the inactive text “off”.

Table 11: Extended Properties of the Binary Object

Property	Fixed/changeable
Value	Changeable
Active Text	Fixed
Inactive Text	Fixed

Extended Properties of the Multi-state Object

The extended properties of the Multi-state object are listed in Table 12. They define the value of the data point itself and how this value has to be interpreted. The property value holds the value of the data point. According to Table 6, the data type of the multi-state object can be either unsigned integer or signed integer. The property number of states defines the number of states of the data point. The state text is the description of the states. Both number of states and state text are assigned during the configuration phase and cannot change during runtime.

Table 12: Extended Properties of the Multi-state Object

Property	Fixed/changeable
Value	Changeable
Number of states	Fixed
State text	Fixed

Extended Properties of the String Object

The extended properties of the string object are listed in Table 13. They define the value of the data point itself and how this value has to be interpreted. Both properties value and length are changeable. The property value holds a character string; the property length defines the length of the value string.

Table 13: Extended Properties of String Object

Property	Fixed/changeable
Value	Changeable
Length	Changeable

Extended Properties of the User Object

The extended properties of the user object are listed in Table 14. They define the value of the data point itself and how this value has to be interpreted. Similar to the string object, properties, value and length are changeable. Since the user object has no defined data type, the value cannot be interpreted without additional knowledge. The property length defines how much memory is used by the value of the data point.

Table 14: Extended Properties of Multi-state Object

Property	Fixed/changeable
Value	Changeable
Length	Changeable

Extended Properties of the Date/Time Object

The date/time object has just one extended property, the value property. As defined in Table 6 only the date/time data type is allowed for that value.

3.8 Organizing the Data Points

The previous chapter described the architecture of DPAL data points. These definitions allow an application designer to describe values in the field. However, in order to associate data points with each other, a DPAL repository is introduced into the DPAL concept. Following is a brief explanation on how the grouping of values is implemented in CNP. Subsequently follows the presentation of the DPAL repository which has been developed for this work. In CNP, a data point (network variable) may have one or more values. An example is the SNVT_switch [LSML03]. This network variable type has two values as follows:

```
typedef struct {
    unsigned value;
    signed state;
} SNVT_switch;
```

The first value defines the intensity as percentage of full scale (resolution 0.5). Therefore, the range is between 0 and 100. The second value (state), which can be either 0 or 1, indicates whether the switch is on or off. In the CNP protocol these two values are seen as one data point. But in the DPAL model presented above, every data point has exactly one value. Therefore, a repository of the data points is introduced into DPAL, which allows the applications to browse the data points and find out which data points are correlated. A graphical representation of the DPAL repository is shown in Figure 18.

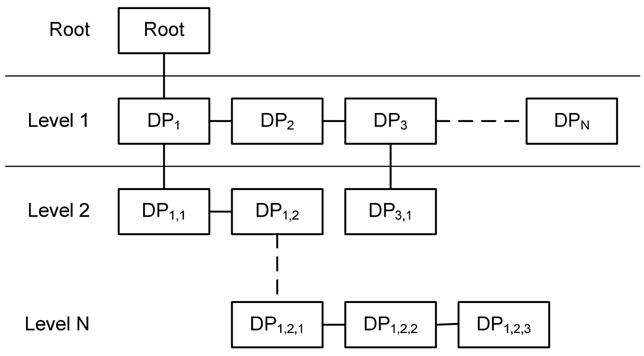


Figure 18: DPAL repository

The root object is a DPAL point with the object type node and therefore has no value. The reason for introducing it is to have a well-defined starting point for the browsing functions of the repository. All data points in the first level are children of the root object. Additionally, they are linked in their hierarchy level. Every data point can have child objects, as e.g. DP_1 and DP_3 in Figure 18. These child objects are in the next lower level. The child objects of DP_1 are labeled $DP_{1,1}$ and $DP_{1,2}$, and the ones of DP_3 are $DP_{3,1}$. Because the data points $DP_{1,1}$ and $DP_{1,2}$ are both child objects of the data point DP_1 they are linked together in the second hierarchy level. But the child object of DP_3, $DP_{3,1}$, is not linked with the other data point of hierarchy level 2, because it has not the same parent object as the other data points. This tree can be expanded to any number of levels as long as the resources of the hosting system are sufficient. Returning to the example above, there are two possibilities to map a CNP data point of the type SNVT_switch to the DPAL repository. Both are depicted in Figure 19.

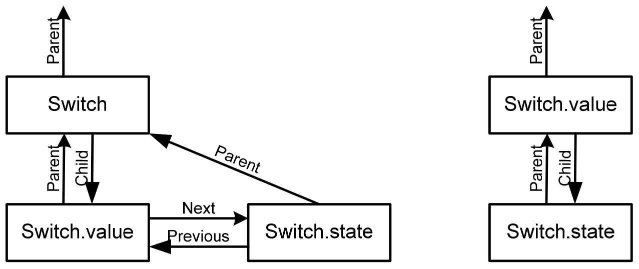


Figure 19: Mapping SNVT_switch to DPAL data points

The variant on the left-hand side uses a node object, which is labeled switch. It has no value and is only used to group the data points holding the two values of the CNP data point. The child object of the node object switch is the first value of the structure as defined in [LSML03], which is the percentage value. It is labeled switch.value and is an analogue object. The DPAL data point switch.state is mapped to the second value of the SNVT_switch structure, the state. Since the state has a binary value, the DPAL object type binary can be used for this object. As shown in Figure 19, the DPAL node object switch is the parent object of the data points switch.value and switch.state. Since these two data points are in the same hierarchy level and have the same parent object, they are also linked together.

The right-hand side of Figure 19 shows the second possibility. No node object type is used. The first object of the structure is used as the first DPAL data point. In the case of the SNVT_switch this is the data point switch.value. The next objects, in this case only the data point switch.state are child objects of the first value of the data structure. This method of mapping data structures to DPAL data points is more resource-efficient because it uses one element less, but does not reflect the correlation of the elements of the data structure. The data points holding the values of the structure (in this example switch.value and switch.state) are not in the same hierarchy level.

The DPAL repository can also be used to optimize the handling of dynamic data points. Dynamic data points are data points which can be created and deleted during runtime. In the CNP protocol these data types are called dynamic network variables [LIGA05 p 110f]. Especially for fieldbus nodes the creating and deleting of data points might be not very time-efficient. To avoid this, once a dynamic data point has been created, it may not be deleted completely from the system but just removed from the repository. A data point than can be inserted into the DPAL repository is very resource efficient.

The DPAL repository can not only be used to reflect the data structures of data points as shown above, but also to represent geographical or logical grouping of data points. For example the first hierarchy level can represent buildings, the second level floors, the third rooms and so on. The next chapter describes how the values and properties of data points can be accessed.

3.9 Data point access

As already mentioned above, the DPAL repository is intended to organize the DPAL data points. For accessing the value and properties of the DPAL data points, the use of dedicated interfaces is proposed. These interfaces are introduced to implement a simple right management of DPAL. As shown in Figure 17, there are applications and the DPAL configuration instance. In contrast to applications, only the DPAL configuration instance can access the DPAL configuration interface. This interface provides the functions to create and delete DPAL data points, to insert and remove data points into and from the DPAL repository. However, if a data point is manipulated, the applications using the data point are informed about the changes. Moreover, all properties of the DPAL data point are readable in every application.

Although the value of the data point is a property of the data point, it is treated in a special way. In principle there are two approaches to access the values of the native systems. The first approach is to cache all data in the DPAL. The applications have access to the cache and can read the value from it. A write operation on the data point first changes only the value in the cache of the DPAL data point. It is up to the system integrator which type of service of the underlying communication protocol is used to propagate the change of the value on the network. The advantages of this approach are that the applications can access the value of the data points very fast (it is only a read and write access in the memory of the host CPU). Further, it can be implemented with every underlying communication system and the characteristics of the underlying communication system can be completely hidden. DPAL responds automatically to requests from the native communication system.

The drawbacks of this caching method are that the data in the cache are not necessarily consistent with the data in the field. Further, when powering up the system it may take quite some time until the cache is updated and DPAL can be set into operational mode. If the underlying communication system only provides synchronous services, it can not be ensured that every change of the value in the cache is propagated to the network. It might happen that the application changes the value faster than the value is updated via the network.

The second approach is that DPAL “translates” every data access function directly to the services of the underlying communication protocol. On the other hand, every request or update from the native communication system has to be processed by the application. The advantages of this approach are that it guarantees the most accurate value of the DPAL data point and there is no dead time when powering up the system. The drawback is that the applications can only use services which are provided by the underlying communication protocol. The access of the data points is slower, because the value is no longer read from the cache but from the native communication system.

For DPAL a mixed approach was chosen, which combines both variants described above. It implements a cache, which caches the values as described in the first variant. For reading the value of a data point, two functions are introduced, one which reads the value from the cache, the other which translates the request to the native system in order to get the most accurate value. However, if the DPAL cache is not valid (e.g. at powering up), a read request from the cache is translated to native read request. If the value of the DPAL data point is changed by the application, the value is updated in the DPAL cache as well as being propagated via the network. If the underlying communication system does not provide asynchronous services, it is emulated by the DPAL polling the value periodically.

Beside the read/write value functions, there are also read/write multiple value functions. These functions are used to read or write more than one value at once. These functions are necessary to read/write values from different (native) data points, but especially to read native data points which have a structure of different values as the SNVT switch mentioned above. Like BACnet [BAC04] the DPAL provides also a change of value notification (COV). If an application uses this service, it obtains a notification of the change of value from DPAL. In case of an analogue object, the hysteresis can be configured (see Chapter 3.7.3).

3.10 Mapping BACnet to DPAL

The services in the BACnet protocol (see Chapter 3.2) comprise two types, the server side and the client side. The server side provides data, whereas the client side sends request for reading or changing the data. The following chapters describe the mapping between these two types and DPAL.

3.10.1 DPAL as application of BACnet server

One possibility is to use DPAL as an application of a BACnet server. From the network side, the node behaves like a BACnet server. This means that a BACnet input object is typically read and that it sends out COV notification messages in case of implementing a service as shown in Figure 20.

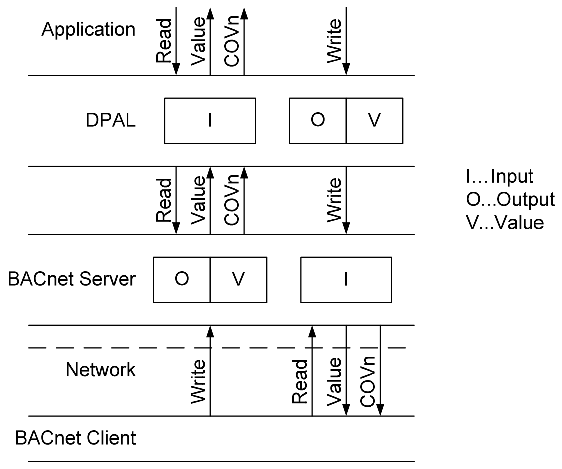


Figure 20: DPAL acts as BACnet server

The value of the BACnet output object and BACnet value object can be written via the BACnet network. DPAL maps the BACnet output and value objects to DPAL input objects. It uses the read property service and the COV service of the BACnet server stack to synchronize the value of the BACnet output and value object with the DPAL cache. The application can access the DPAL input object with the read functions and receives COV notifications if subscribed.

The BACnet input object is either mapped to DPAL output or DPAL value object. The application writes the DPAL value and output objects. DPAL propagates the new values using the write service of the BACnet server stack.

3.10.2 DPAL as application of BACnet client

In case of implementing DPAL as an application of a BACnet client stack, within the BACnet stack there are no BACnet objects, as shown in Figure 21. A read request on a DPAL input object is translated into a BACnet read request. The BACnet client stack sends out a BACnet read request to the BACnet server, which holds the value. The response of the server is sent back by the network to the BACnet client, which forwards the value to DPAL. DPAL updates the cache and sends the new value to the application. If the application has subscribed for the DPAL COV notification, it is informed whenever the value of the DPAL object changes. The value of the DPAL object can change when DPAL has subscribed (via the BACnet client stack) to the COV service of the BACnet server on the remote node or when DPAL periodically sends out read requests to the BACnet client stack in order to emulate a synchronous service.

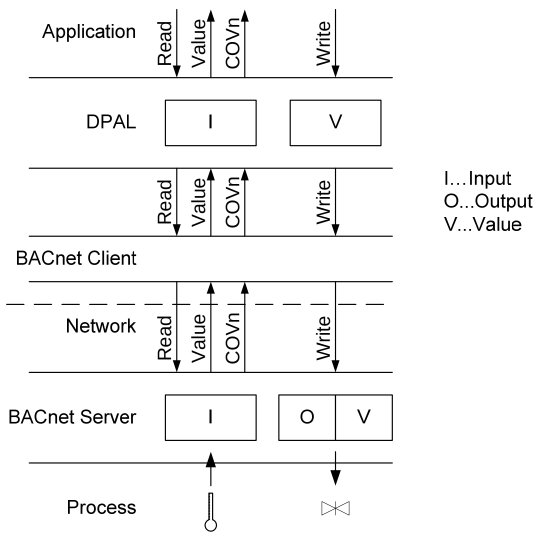


Figure 21: DPAL acts as BACnet client

If the application uses the write service of DPAL, the cache of the DPAL output or value object is updated. DPAL uses the BACnet client stack to write the new value via the BACnet network to a BACnet server.

3.11 Mapping DPAL to CNP

There are two possibilities to map the DPAL data points to CNP communication objects. On the one hand, the communication objects in CNP can be implemented as NVs as described in Chapter 3.1. On the other hand, the CNP stack under DPAL does not use NVs but poll and update requests in order to read and write NVs on remote nodes. In the following chapters these possibilities are described in detail.

3.11.1 DPAL as application on a CNP stack with NVs

This variant of mapping CNP to DPAL is intended to use for devices that have NVs in the CNP stacks, which are visible on the CNP network, as shown in Figure 21. A controller node could be such a device. In this case, DPAL input objects are mapped to CNP input NVs. In case of a read request of the application, DPAL sends the read request to the CNP stack. In case the NV has a binding with a remote output NV, the value is updated automatically by the CNP protocol. Therefore, no request has to be sent by the network. The CNP stack returns the value immediately. In case there is no asynchronous service available, the value is polled by the CNP stack from the remote NV.

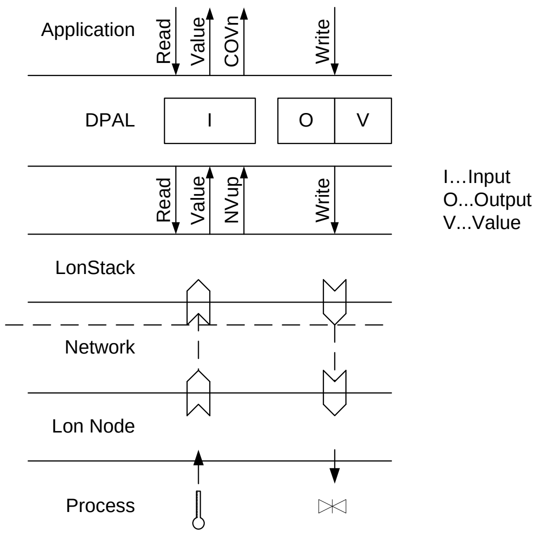


Figure 22: DPAL acts as CNP application with NVs

In case the application writes the value of an output or value object, the new value is changed in the corresponding output NV of the CNP stack of the node. If the NV is bound with a remote input NV, the value is automatically propagated via the network, otherwise the remote node has to poll the value.

3.11.2 DPAL as application on a CNP stack without NVs

Beside the communication with NVs the CNP protocol offers communication using so-called explicit NVs [Loy01, Mot97]. To poll or update the value of remote NVs CNP network management commands are used. Explicit NVs are not visible from the network side. Devices which serve a huge amount of data points like visualizations or gateways us that possibility. Explicit NVs are more flexible, but require a higher configuration effort. Figure 23 shows a configuration where DPAL acts as application a CNP stack without NVs.

If an application reads the value of a DPAL input object, DPAL sends a request to the CNP stack which sends out a poll request to the network. The response with the current value of the remote node is used to update the value in the DPAL cache and respond to the application. Using explicit NVs, only a synchronous service is available. To emulate an asynchronous service, DPAL has to request periodically the value NV of the remote node.

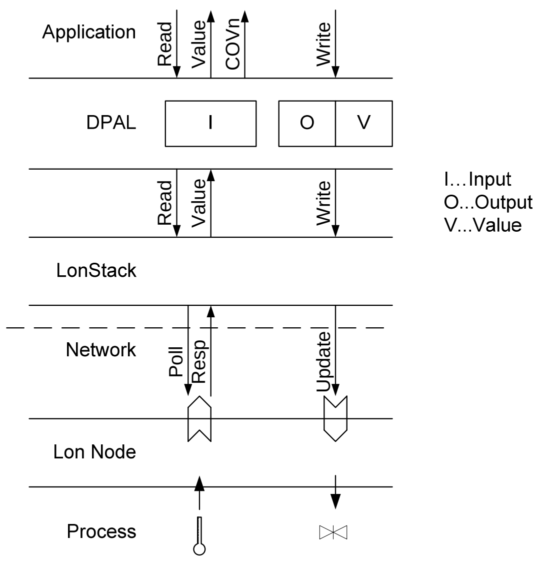


Figure 23: DPAL acts as CNP application without NVs

A write command of the application of output or value objects is forwarded by DPAL as a write command of the CNP stack. The CNP stack uses the network management command update to change the value of the NV of the remote node.

3.12 Résumé

The presented approaches above, i.e. gatewayless communication and DPAL, are both methods to provide applications with data origin from two or more communication protocols. Both have the advantages that a gateway does not exist as a separate device, which has to be configured by the system integrator. The conversion is carried out on the nodes where the application runs. Therefore, the protocol conversion is also done on the node and is configured by the application designer, who has the best knowledge about the applications. The gateway as a single-point-of-failure is completely eliminated from the system.

The first approach, the NeuBAC node, is only suitable for communication protocols which rely on each other, as e.g. BACnet, which uses the LonTalk protocol of the CNP protocol. Nevertheless, this approach needs only a CNP stack and the BACnet Application layer and the BACnet Network Layer. The lower layers are provided by the CNP stack. Existing CNP devices can be extended with the BACnet interface without changing the hardware, in case enough computational resources are available on the node. Since the approach assumes that a node is used which provides a full CNP stack, the applications on such NeuBAC nodes are most probably designed for the CNP protocol and its services, because the communication objects (NVs) and services of the stack are used for the CNP part. The BACnet communication objects and services are then mapped on the CNP objects and services. However, applications which only “observe” what happens in the field like visualization or a system like the ARS system can use this interface to access data. Due to the limited resources of a NeuBAC node, it is not intended to host complex applications.

The second approach, DPAL, is a more common approach than the first one. Nevertheless, it is still suitable for fieldbus nodes like the LC3020 [LC3020]. Using DPAL applications the stacks of every supported communication protocol has to be implemented on the node. Since the data points of the field as well as the services of the underlying communication system are completely abstracted, an application on top of the DPAL is developed independently. If a service of the DPAL is not available by the native system, it is emulated with other services of the system. Therefore, this approach is also suitable for applications which control processes in the field.

4 Applications

The ARS system is designed as a system, which is not limited to a special purpose. Nevertheless, its implementation on the ICT (Institute of Computer Technology) is focused on home and building automation and the possible application of the ARS system in these fields. The applications are based on symbols as proposed in [Pra06]. The applications in this work are based on home and building automation sensors. DPAL (Data point Abstraction Layer) defined in Chapter 3.7 can be used to hide the specifics of the communication systems used in the building. Additionally to the building automation system, cameras can be used to provide visual data.

4.1 Kitchen as an Environment for Applications

The applications of the ARS system can be demonstrated in the kitchen of the ICT. Another possibility is the use of a simulator, which was designed to generate sensor values based on a virtual environment. Such a simulator is described briefly in Chapter 7.1.3 and in detail in [Har07].

The layout of the kitchen [Goe06] is shown in Figure 24. Entering the room on the left-hand side there is a kitchenette with sink, dishwasher, stove, coffee machine, fridge and cupboards. Behind the door on the right-hand side there are a cupboard and copier. Near the window on the left-hand side is a cupboard, in the middle of the room a table and on the right-hand side a bookshelf. The room is equipped with a pressure sensor field, which covers almost the whole floor. Each individual sensor has the dimensions 17 cm x 40 cm [Goe06]. Three motion detectors, two on the side of the kitchenette and one on the opposite side are installed to detect motion in the kitchen. A shock detector is mounted on the coffee maker to register its usage. Two cameras are installed above the window. Distance sensors are mounted on the door and at the stove. The following applications are defined in this work of the setup in the kitchen on the ICT:

- People tracker
- Recognition of a person putting down an object
- Recognition of a child approaching a hot stove
- People meeting round the table
- Person making coffee

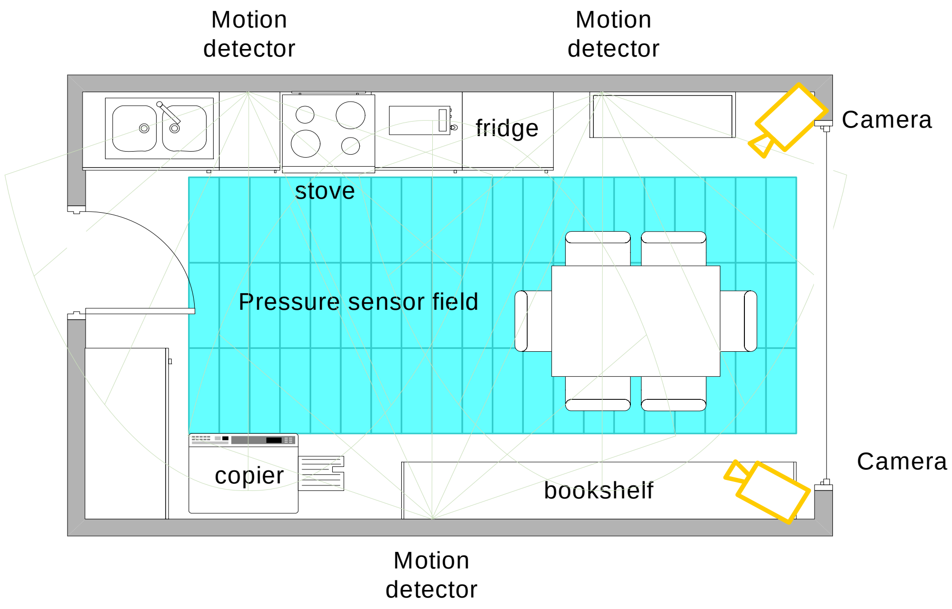


Figure 24: Layout of the kitchen in the ICT

The people tracker is one of the most important applications of the system since applications that involve people are based on it. The main task of the people tracker is to identify the position of people in the room and to record their walk path through it. The second application recognizes a person putting down an object onto the floor and leaving the place without taking the object. Recognizable objects are suitcases, packets or boxes. If a child is approaching a hot stove it is usually a dangerous situation, since the child might get injured. So the system should recognize this dangerous situation, but should also recognize whether there is an adult near the child. The fourth application is the recognition of meetings. A meeting is assumed to take place when two or more people sit around the table. The last scenario recognized in the kitchen of the ICT is a person making coffee.

All these applications are designed for the kitchen in our department. They were chosen because the events which trigger them occur quite frequently in our kitchen. Therefore, they can easily be put to the test. Of course the child safety application is not regularly used at university, but it was chosen to show other application fields of the system. Further applications developed for the ARS system but not implemented in the ICT kitchen and therefore in the prototype are:

- A person falls and lies on the floor
- Danger of fire
- A person takes reading material from the shelf
- Audio system self-test

Similar to the safety application for children mentioned above, the scenarios of a person falling or lying on the floor are suitable for detecting dangerous situations for elderly people. An example of recognizing dangerous situations with no people involved is the detection of flammable objects near a fire. A system could also have the ability to test its own functions. This does not mean that it is able to detect software failures but rather that it detects failures on the periphery. The third application recognizes whether someone takes reading material from a shelf. This application sounds the alarm if the material is carried out of the room. If the system interacts verbally with a user (e.g. tests whether the user reacts to a certain question), microphones should be able to detect the output of the system; otherwise there is a failure in the audio system. In the following chapters, the applications mentioned above are described in more detail.

4.2 People tracker

The people tracker has the task to identify the position of people within a scenario. Nowadays, people trackers usually use one or more cameras and image processing methods [Ham05] combined with other methods like Bayesian estimation [Bal05]. Here not only cameras but also other sensors like pressure sensors, distance sensors or light barriers are used. They are not directly combined (e.g. in the image processing module), but are connected through different snapshot symbols.

The position of a person is taken from the property *position* of the representation symbol *person*. The history of positions is stored in the symbol database (see Chapter 5.8). Therefore, the walk path of the person can be extracted by tracing back the property *position* of the symbol *person*. The representation symbol *person* is based on other symbols (for a detailed description of the representation symbol *person* see Chapter 6.1.5), which are based on diverse sensors. These sensors are pressure sensor on the floor, distance sensors on various positions, motion detectors, and a camera. Using geometric algorithms and classification numbers for the reliability of the different sensors, the position of a person cannot be identified with higher accuracy than using just one type of sensors. The output of the application is a periodical update of the position of a person.

The people tracker is not only used as a stand-alone application, but also as input for the applications in the following chapters. Since these applications do not need the identity of a person, it is not planned that the people tracker application identifies a person. This reduces the impact on the privacy of the user of such systems. The symbols which are defined within the ARS system used for people tracking are described in Chapter 6.1.

4.3 Recognition of a person putting down an object

The sequence of actions in this scenario is that a person carries a packet into the room, places it on the floor and leaves the room without taking it out of the room. To reduce the complexity in this work, the packet is not necessarily detected when the person carries it into the room. The application people tracker (see Chapter 4.2) is used to recognize the position of the person. To detect the packet, the symbol *object passed* is used (see Chapter 6.1.2), which is triggered in this case by the pressure sensors. In future, a camera could be used in addition to detect the packet. The camera would then generate the snapshot symbol *packet*. If RFID (Radio Frequency Identification) is used to identify packets, RFID read at the door could also be used not only to detect a packet in the room, but also to determine its contents. The output of this application is the notification of a predefined person of someone having left a packet in the room. The symbols used in this application are described in Chapter 6.4.

4.4 Recognition of a child approaching a hot stove

This scenario shall be applied to the dangerous situation of a child near a hot stove without the supervision of an adult. The child is detected with the application people tracker (see Chapter 4.2). But in this case information on the position of the person has to be extended by information on his or her age. Although there is no possibility to derive the age of a person from known sensors, a person's height is used as an indicator. It is assumed that in this application it is acceptable to derive the age of a child from the child's height[10]. Another possibility to estimate the age of a person would be voice analysis as shown in [Schö04]. The drawback of this solution is that a person would have to speak before the age can be estimated.

Similar scenarios are the recognition of an unattended child near stairs or in the lift. These scenarios are similar because the system has to be distinguished whether the people it has recognized are adults or children. The necessary child safety symbols are defined in Chapter 6.2.

4.5 Meeting

The meeting scenario recognizes when more than two people sit around the table. It is not taken into account what the people do. This scenario would also be triggered if two people sit at the table and read. Nevertheless, additional sensors like microphones or cameras could ensure that the scenario meeting is only detected when people talk to each other or look at each other.

However, in the prototype of the ARS-PC system the meeting scenario application is based on the people tracking application, which detects the position of people in a room. Certain objects like a table or a stove are represented as a special symbol, e.g. symbol table within the system. If two or more people, represented by the symbol person, are in the vicinity of the symbol table, the scenario meeting is detected. Chapter 6.4 shows the relationship between the symbols used in this scenario.

4.6 Person makes coffee

The scenario person makes coffee is divided into two similar scenarios. Both detect a person making coffee with the coffee maker. The difference is that a distinction is made whether the person is an adult or an unattended child, similar to the scenario child approaches a hot stove (see Chapter 4.4). Using the age estimation of the people tracking application, a person is identified as child or adult. The symbol *coffee machine* (see Chapter 6.2.20) represents the coffee machine and its status. The distance between child, adult, and the dangerous object is used for deciding whether a particular situation is classified as dangerous or not. Additionally to the person making the coffee, the amount of coffee made by the person is registered. The symbols and the dependencies between the symbols used in the application person makes coffee are described in Chapter 6.2.18.

[10] The drawback of this application is that a small person would be classified as child, and thus causes alarm.

4.7 Person falls and lies on the floor

This scenario is intended for geriatric care. It is defined as a scenario where a person falls down and does not stand up, which is usually a sign of a dangerous situation. As mentioned above, this application is not implemented in the ARS system on the ICT. Nevertheless, the scenario is defined as follows: A person, who is detected with the people tracker, falls down. The fall can be detected with image processing or pressure sensors. These sensors are combined to the symbol *person fell* (see Chapter 6.3.7). If the person does not get up, the symbol *person lies* (see Chapter 6.3.9) is generated. To improve the scenario, the severity of the fall could be estimated by detecting the force of the impact[11]. The speed of getting up could also be used to determine the severity of the fall. Since a fall can have severe consequences in any case, and especially in the case of elderly people, the output of this application should always be the notification of the caregiver. Chapter 6.3 defines the symbols which are used to detect harmless and harmful falls.

4.8 Danger of fire

Similar to the previous scenario, this scenario can be used in geriatric care. It is defined as detection of objects like candles or cigarettes as well as flammable objects (e.g. bed or curtain). If flammable objects are near open fire, an alarm will sound. The detection of a cigarette or candle could be achieved with image processing using either a camera that is sensible to the visible light spectrum, or a camera that is sensible to the infrared spectrum. There are flame detectors available now, but no sensors that are also capable of detecting the position of a fire. To avoid or at least minimize the occurrence of false alarms, it is essential to determine the distance between fire and flammable object.

4.9 Recognition of a person taking reading material from a shelf

The kitchen of the ICT also has a bookshelf with journals. Members of the institute can read them but should not remove them from the kitchen (without signing a list). This scenario is therefore defined as follows: A persons enters the room and takes a journal from the bookshelf. Before the person leaves the room, the journal has to be returned. The output of this application is the notification of a predefined person of a person leaving the room without having put back the journal into the bookshelf. The people tracker application (see Chapter 4.2) is used to determine the position of the person in the room. Image processing is used to detect that someone takes a journal. Other or additional possibilities are the use of distance sensors in each shelf or RFIDs in each journal. Chapter 9.2 discusses a possible implementation.

[11]This could be done e.g. by measuring the force with pressure sensors, which is not possible in the ARS project since pressure sensors have only a binary output detecting whether there is pressure above a certain threshold or not.

4.10 Audio system self-test

If a system interacts with the user verbally, it has both audio input and audio output components. If the system sends a message to the user, the microphones in the room should be able to record the message and the speech recognition system should be able to recognize the message. If the recognized message is the same as the emitted message, the system has no malfunction. The output of this application is sending an error message to the maintenance department. Similar mechanisms could also be used for other inputs and outputs of the system. Since there is no audio system in the ARS project, this self-test is not implemented either.

However, this application shows that not only observing task are possible applications for the ARS system. As mentioned in the Chapter 2 the ARS system forms a loop. The ARS-PC part observes the environment and the ARS-PA part uses the perceived world representation in order to make decisions how to interact with the environment. This loop is essential for this application. The system generates an output and has to be able to perceive its own interaction with the environment. This loop is also necessary to implement higher cognitive functions like planning or learning. Nevertheless, these are functions of the ARS-PA part and therefore out of the scope of this work. The next chapter describes the ARS-PC system as it is used to implement the applications described above.

5 ARS-PC System

The system in the ARS project uses symbols as defined in [Pra06] and modified in this work according to the model by Luria and Solms (see Chapter 2.2.3) and extended with the concept of basic emotional systems according to Panksepp (see Chapter 2.2.2). As already mentioned, learning is not addressed in this work, so the symbols used for the applications defined in the previous chapter have to be predefined. In order to do so and to process the symbols, a framework is necessary. The next chapter defines this framework.

5.1 ARS Framework

The symbols are used on the one hand to consolidate data (from one ore more sources), and on the other hand to perform plausibility tests (from different sources and/or prediction). Figure 25 gives an overview of the ARS framework. It consists of the definition of symbols, transport mechanism (SymbolNet) and the representation of the symbols. Additionally, a database for symbols and a database for sensor values are included.

The framework of the definition of the symbols is used to define the specific properties of a symbol. A symbol represents people/objects, actions or states, which are derived from the sensor inputs of the system. As described in [Pra06], a symbol belongs either to the inner world or the outer world. This classification is based on [Sol02]. In the human brain, the representation of the inner world is the representation of the inner milieu of the body. Parameters like blood pressure or temperature are used to arrive at these symbols. The outer world representation is the representation of sensations from the environment. The sensors on which these symbols rely are e.g. eyes, ears or the nose. If sensors then recognize known patterns (e.g. an edge), these patterns are represented as outer world symbols.

Symbols in the ARS system have a symbol class and a symbol name, which both identify the semantic meaning of the symbol. The name is a string of characters whereas the symbol class consists of numbers. The name of the symbol was introduced to have both a human-readable and machine process-able version of a unique identifier. This definition of the symbol determines the number and types of properties. Similar to the symbols, the properties also have two unique identifiers, the property class and the property name. They define the semantic meaning of a property. A property has at least one schema. A schema defines how the data are stored. A detailed description of the definition of symbols using XML-files can be found in Chapter 5.3. The graphical representation of the symbols and the dependencies between them are defined in Chapter 5.2.

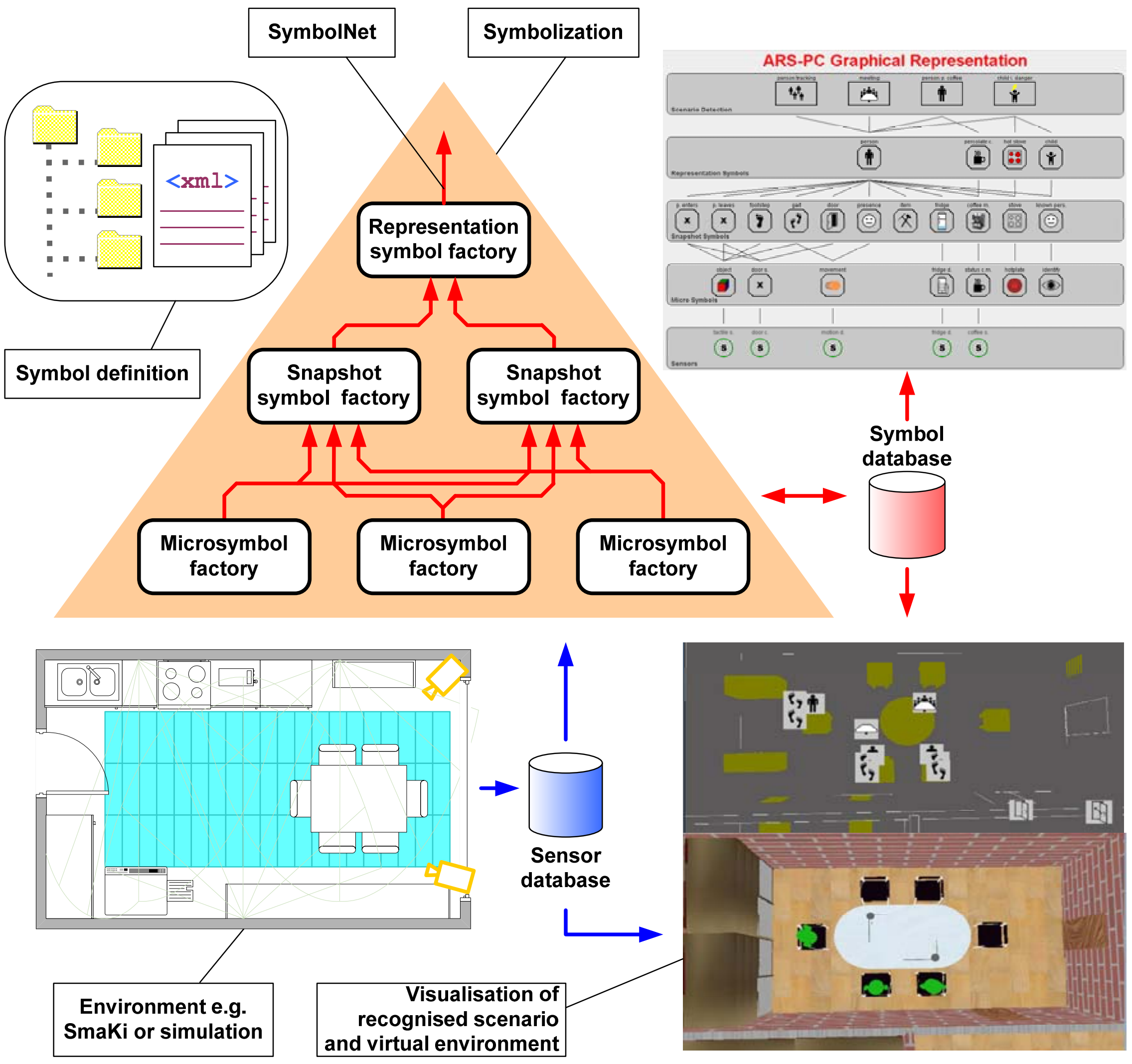


Figure 25: ARS framework

SymbolNet is the transport mechanism for symbols. It was designed to distribute symbols especially between the symbol factories of the symbolization. A detailed description of SymbolNet is given in Chapter 5.5, the usage of SymbolNet and the symbols in Chapter 5.6. The symbol database is used as a repository for the symbols. In this repository both the actual symbols with their actual properties and the expired symbols and properties are stored (see Chapter 5.8). Similarly, the sensor database is used to store the sensor value from real sensor implementations or simulations (see Chapter 5.7). The databases have been introduced to ease the implementation of the system. They have no functionality in terms of a memory as it would be found in the human brain. The ARS-PC system is designed to work without the sensor and symbol databases as well. In the lower right side of Figure 25 two views of visualization are shown. In the upper part of the visualization the recognized symbols and scenarios are shown, in the lower part the virtual environment and the objects and persons in it. The various types of visualizations for the ARS project are described in Chapter 5.9. Not explicitly shown in Figure 25 are the valuation systems, the so-called e-systems. The value of the e-system is represented by the e-status as described in Chapter 5.4. The term e-status stands for the technical representation of the basic emotional status.

5.2 Implementation of Symbols

A symbol consists of properties which describe it, as shown in Figure 26. This chapter gives a rough overview of the concept of symbols as used in the ARS project. A more detailed description is given in Chapter 5.3. Every symbol has the mandatory properties (which are not shown in the graphical representation) Identifier, Type, Class and Timestamp, the optional property Lifetime and a list of properties which define the specific properties of the symbol and are therefore shown in the graphical representation of the symbol.

The Identifier has to be unique in all symbols in a system. The Type specifies which "world" and representation hierarchy the symbol belongs to. The world to which the symbol belongs can either be the inner or outer world. Three representation hierarchy levels are defined, the microsymbol for the lowest level, the snapshot symbol for the middle representation level, and the representation symbol for the top level. The Class property of a symbol is a combination of both world qualifier and hierarchy level qualifier. The Timestamp specifies the time when the symbol was created or changed. The optional property Lifetime can be used to define how long a symbol should exist before it is automatically deleted.

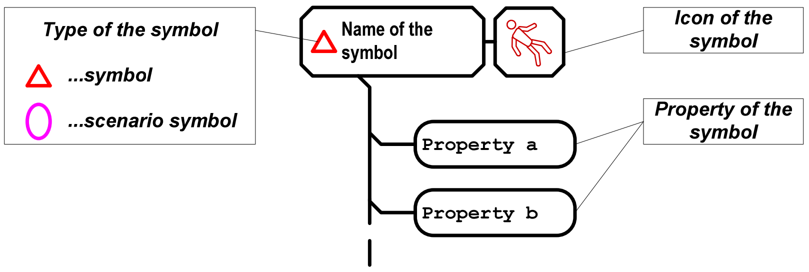


Figure 26: Representation of a symbol with properties

The List of Properties defines the individual properties of the symbol. These properties are all optional. A property is defined with its property class, which describes the semantic meaning. The schema how the data are stored is defined in the Property Schema. The following example should help to clarify this concept. The microsymbol Footprint (see Chapter 6.1.1) has also the property classSymbolicPostion to describe the position of the footprint and the property classSymbolicDirection, which describes the direction of the footprint. Although these properties have different semantic meanings, they both have the same property schema, namely schemeSymbolicPosition.

In this work, a symbol is illustrated as shown in Figure 26. The left frame on the top gives the name of the symbol. To distinguish between symbols and scenario symbols a red triangle or a pink ellipse is used. The right frame on the top shows the icon of the symbol. This icon is used in graphical representations of the symbols in this work and in the visualization within the ARS project. Along the vertical line the properties of the symbol are arranged.

The graphical representation of applications and the dependencies on symbols are illustrated as shown in Figure 27. The name of the application is in the green frame. The application is usually on the top of the tree of symbols. The symbols on which the application depends are in a black frame containing the icon of the symbol. It is the same icon as used in the representation of symbols with properties. The dependencies of an application on symbols or of symbols on other symbols respectively are depicted as red arrow. Sensors are usually drawn as blue circles. An exception is the camera, which is drawn as a camera symbol. The dependencies of a symbol on a sensor are shown as blue arrows.

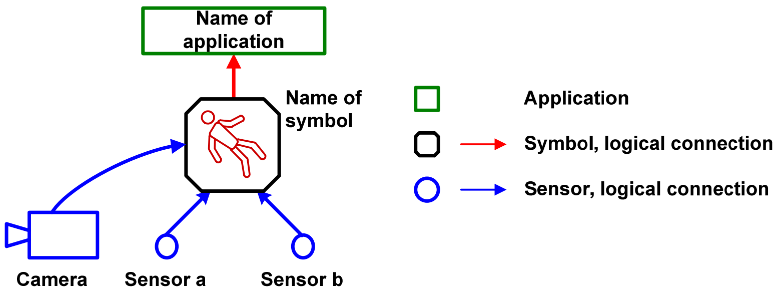


Figure 27: Graphical representation of application, symbols and sensors, and their dependencies

The application is responsible for setting all necessary parameters of the involved symbols. These parameters are like templates to recognize characteristics as well as the focus attribute, which exists in every symbol and shows that there are one or more applications which listen to this symbol or a symbol that depends on this symbol.

The symbols themselves have the “knowledge” of how they are generated, updated and terminated. They have attributes which describe their “meaning”. Inherent to symbols is the attribute activity, which represents the reliability and/or intensity of the symbol. The next chapter describes how the symbols and the relationships between the symbols are formally defined using XML-files.

5.3 Formal Definition of Symbol and Symbol Trees

In the pervious chapter the graphical representation of the symbols and the relationships between the symbols were defined. This chapter describes the formal definition of symbols and their relationships using XML-files. XML was chosen because it is both readable for humans and machines. Although not implemented now, the possibility of automatically generating code templates, which can be used to implement applications, has been taken into account. Additionally, a verification tool for symbols is possible. The task of this tool is to inspect symbols on the SymbolNet connections in order to find symbols which do not fit their definitions and dependencies. The following chapters describe how symbols, properties of symbols, schemes and symbolic values are defined.

5.3.1 Definition of a Symbol

Exemplarily, the symbol gait (for a detailed description see Chapter 6.1.4) is used to explain the most important tags of the file. The symbol gait represents a sequence of footprints which can be connected to the gait of a person. The specification of the symbol gait is defined as follows:

```
<symbolTemplate>
    <symbolClass>0.0.0.4</symbolClass>
    <symbolName>ossGait</symbolName>
    <symbolType>snapshot</symbolType>
    <symbolWorld>outer</symbolWorld>
    <properties>
        <property>
            <propertyClass>0.0.0.1</propertyClass>
        </property>
        <property>
            <propertyClass>0.0.0.8</propertyClass>
        </property>
        <property>
            <propertyClass>0.0.0.9</propertyClass>
        </property>
        <property>
            <propertyClass>0.0.0.6</propertyClass>
        </property>
        <property>
            <propertyClass>0.0.0.3</propertyClass>
        </property>
        <property>
            <propertyClass>0.0.0.4</propertyClass>
        </property>
    </properties>
    <icons>
        <icon>gait.png</icon>
        </icons>
    <dependencies>
        <dependence>
            <symbolClass>0.0.0.1</symbolClass>
        </dependence>
        <dependence>
            <symbolClass>0.0.0.2</symbolClass>
        </dependence>
        <dependence>
            <symbolClass>0.0.0.10</symbolClass>
        </dependence>
        <dependence>
            <symbolClass>0.0.0.11</symbolClass>
        </dependence>
    </dependencies>
        .
        .
</symbolTemplate>
```

The symbol is defined within the root element symbolTemplate. The first elements define the mandatory properties of a symbol. The element symbolClass defines the class of the symbol. It has to be unique within the system. The symbol gait has the unique symbol class 4. The element symbolName was introduced to increase the human readability of the symbol definition. However, the name of the symbol gait is ossGait. Oss stands for Outer (world) Snapshot Symbol as proposed in [Goe06][12]. The name of the symbol is combined by its type (world and hierarchy level) and name. This allows us to use similar symbols on different hierarchy levels using the same name. Nevertheless, the name of the symbol is not intended to be processed within the system. The elements symbolType and symbolWorld define the hierarchy level and world of the symbol as also coded in the symbol name.

[12] Further prefixes are oms for Outer world MicroSymbol, orp for Outer world RePresentation symbol, and osc for Outer world SCenario.

The distinction between inner and outer world in the ARS project is not necessary for the ARS-PC part, because the task of this part is only to process sensor data from the environment. The ARS-PA part (see Chapter 2.2.4) then uses the symbols and evaluates them, using concepts like complex emotions and associations [Bur07] and [Deu06]. However, all symbols generated by the ARS-PC part are symbols of the outer world.

The list of properties is capsulated within the element properties. The element propertyClass defines the property class. For each defined property class a XML file is defined as illustrated below. The properties specified for the symbol gait are position, velocity, acceleration, orientation, weight, length, and width. However, a detailed description of the symbol gait can be found in Chapter 6.1.4. The icons used in the visualization can also be included in the definition of the symbol. For this purpose a list of image files can be included in the element icons.

The element dependencies holds all symbols, on which the symbols depend. To identify these symbols the class of the symbol is used. For this work it is defined that the symbol gait depends on the microsymbol footprint, step, object, and movement. Using this information, a tree of symbols can be built. Not shown in the XML-code segment above are additional elements like verbal description, version numbering, change logs and so on.

5.3.2 Definition of a Property

The following gives a XML-code segment defining the class symbolic position is given:

```
<classTemplate>
    <propertyName>classSymbolicPosition</propertyName>
    <propertyClass>0.0.0.1</propertyClass>
    <schemaId>0.0.0.1</schemaId>
                    .
                    .
</classTemplate>
```

The class symbolic position is used to describe a position within a room. Instead of numeric values it uses symbolic values which describe areas. The root element of class definition is classTemplate. Similar to the XML-definition of the symbols above, a name of the property is introduced, defined in the element propertyName. In this case the class is called classSymbolicPosition. Again, the name is not intended to be processed in normal operation of the system. The element propertyClass holds the unique class identifier of the property class. In this case it is 0.0.0.1. The element schemaId then defines, which scheme is used for the representation of the value. The scheme schemaSymbolicPosition is used for the property class classSymbolicPosition. The definition of that scheme is as follows below. Additional to the element shown in the code segment above, elements like description, change logs and versioning are added.

5.3.3 Definition of a Scheme

An example of the code segment for the definition of the scheme schemaSymbolicPosition is shown here in order to explain the most important tags of the XML-file:

```
<schemaTemplate>
    <schemaId>0.0.0.1</schemaId>
    <schemaName>schemaSymbolicPosition</schemaName>
    <values>
        <value>
            <name>valSymbolicPosX</name>
            <index>0</index>
            <class>5</class>
            <description>The symbolic position in x axis</description>
            <type>32 bit signed integer</type>
            <unit>none</unit>
        </value>
        <value>
            <name>valSymbolicPosXReliability</name>
            <index>1</index>
            <class>13</class>
            <description>reliability of position of x axis</description>
            <type>32 bit signed integer</type>
            <unit>none</unit>
            <behaviourSpecification>
                <minVal>0</minVal>
                <maxVal>100</maxVal>
            </behaviourSpecification>
        </value>
        <value>
            <name>valSymbolicPosY</name>
                       .
                       .
            <name>valSymbolicPosYReliability</name>
                       .
                       .
        </value>
        <value>
            <name>valSymbolicPosZ</name>
                       .
                       .
            <name>valSymbolicPosZReliability</name>
                       .
                       .
        </value>
    </values>
        .
        .
</schemaTemplate>
```

The schema schemaSymbolicPosition defines how a position can be described. It assumes that a position is described using three orthogonal axes like the Cartesian coordinate system. The difference is that no numbers are used to define a point, but a symbolic description of the position. This additional abstraction allows on the one hand defining an area around a position instead of an exact position, on the other hand the definition is not limited to classical mathematics. A description like *close to object* could be used instead.

The element schemaTemplate encapsulates the definition of the schema. The element schemaId is used to identify the schema. In this case it is 0.0.0.1 for the schemaSymbolicPosition. As is the case with the definitions above the schemaName is only introduced for better readability by people. The element values holds a list of values which define the position. A single value has a name which is again only introduced for better readability by people. The element index is used to identify the value within the list of values of the property. The valSymbolicPosX (symbolic value of the position in the x-axis) is on the index 0. The element class defines the description of the symbolic value. The definition of the symbolic value for positions is given below. The data type is defined with the element type. The type of the valSymbolicPosX is a 32-bit signed integer. The unit of the value is defined with the element unit. A symbolic value has no unit and is therefore defined as none.

Additionally to the symbolic value its reliability is defined. This value is used to describe the reliability of valSymbolicPosX. A definition of the reliability for every axis is useful if the position can be determined in one or two axes, but not in all three axes. For the axes where the position can be determined, the reliability is high, for the others it is low. The elements name, index, class, type, and unit have the same meaning as the valSymbolicPosX described above. The name is valSymbolicPosXReliability, the index 1, the class 13, the type is a floating point and the unit is dimensionless. The definition of the valSymbolicPosXReliability has the additional element behaviourSpecification, which is used to define the size of the increment and limits of the value. With the elements minVal and maxVal the minimal and maximum value (in this case 0 and 100) are defined. Equivalent to the x-axis the position and the reliability of the value of the position of the y- and z-axis are defined.

5.3.4 Definition of a Symbolic Value

Following is an example of the definition of a description of a symbolic value.

```
<symbolicValueTemplate>
    <id>0.0.0.5</id>
    <uniqueName>valSymbolicPosition</uniqueName>
    <behaviourSpecification>
        <minVal>0</minVal>
        <maxVal>7</maxVal>
    </behaviourSpecification>
    <values>
        <value>
            <name>symbolicPosition0</name>
            <description>position is in the point of the origin of the axis</description>
            <index>0</index>
        </value>
        <value>
            <name>symbolicPosition1</name>
            <description>position is between the point of origin and including 0.5m on the
axis</description>
            <index>1</index>
        </value>
        .
        .
        <value>
            <name>symbolicPosition7</name>
            <description>position is grater than 3.0m on the axis</description>
            <index>7</index>
        </value>
    </values>
        .
        .
</symbolicValueTemplate>
```

As with all other definitions above, the XML-file contains an element for a unique identifier (id) and a unique name (uniqueName), where the identifier is intended to be used for automatic processing and the name to increase the readability for people. The element behaviourSpecification defines the limits of the used values. For the valSymbolicPosition eight values (0 – 7) are defined. The element values encapsulates the values of the symbolic values. The first value has the index 0 and the name symbolicPosition0. The element `description` contains the verbal description of the symbolic value. The value symbolicPosition0 is defined as position between the origin and 0.5 m of the axis. The value symbolicPosition7 (`index` 7) is defined as a position greater than 3.0 m of the axis.

Using symbolic values to describe positions of e.g. a person in a room can be defined as a position which describes an area. It is also possible to define symbolic values like the position of a certain place. An example in the kitchen of the ICT would be in front of the stove, at the door or nearby the window. However, a symbolic position can always be expressed in a Cartesian coordinate system. The next chapter describes how XSL (eXtensible Stylesheet Language) can be used to increase the readability of the XML definitions for people.

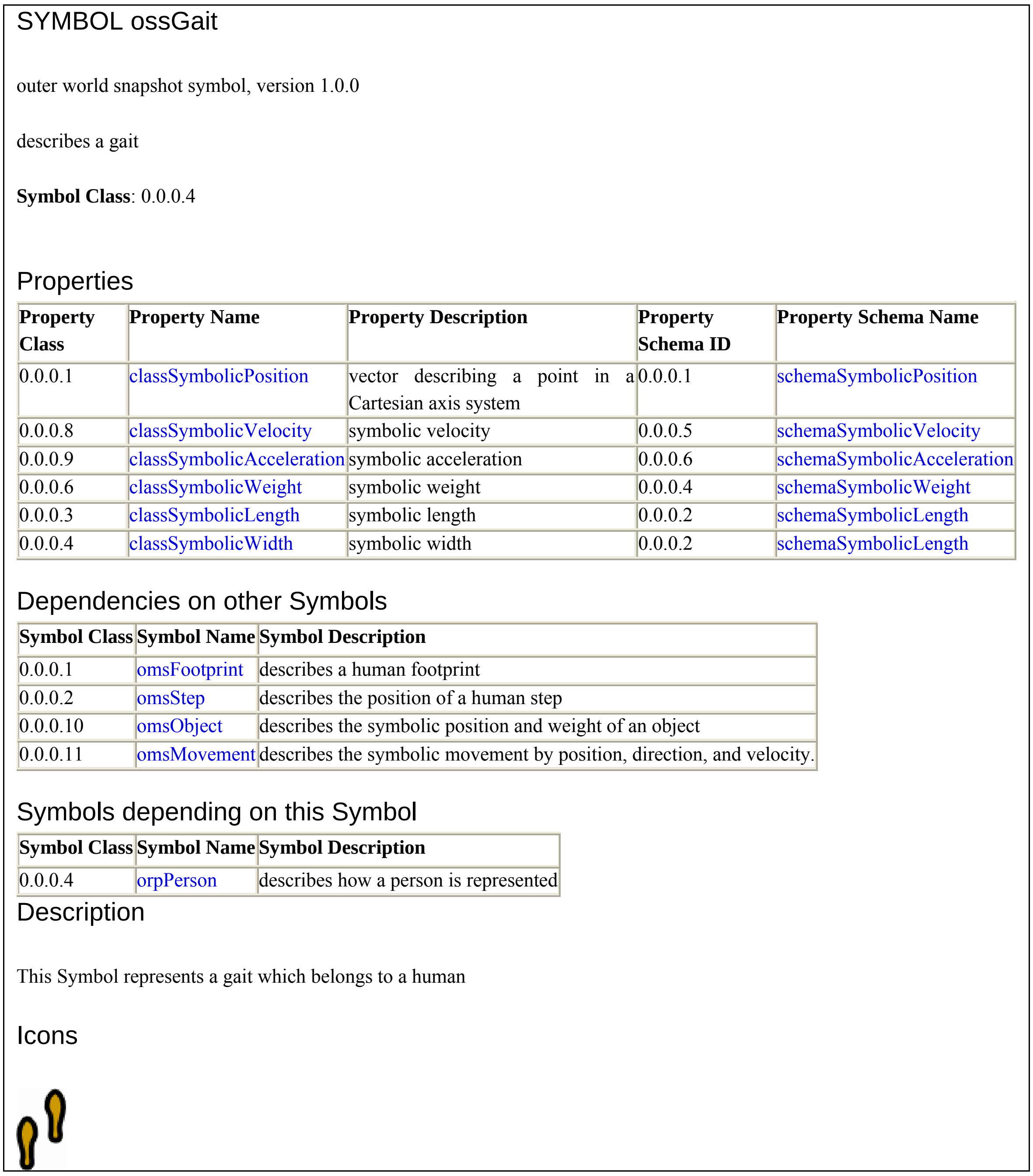
5.3.5 Representation of Symbols using XSL

In order to increase the readability for people and to introduce error checking in the definition of symbols, properties, schemes and symbolic values, the XSL (eXtensible Stylesheet Language) is used. XSL [XSL] is a W3C (World Wide Web Consortium) [W3C] recommendation for defining XML document transformation and representation. It consists of three parts, XSL Transformations (XSLT) [XSLT], XML Path Language (Xpath) [XPath], and XSL Formatting Objects (XSL-FO) [XSL-FO]. XSLT is a language for transforming XML; Xpath is an expression language used by XSLT to access or refer to parts of an XML document as well as being used by the XML Linking specification. Theses two parts of XSL are used to format the XML definitions given above. The third part, XSL-FO, which is mentioned here for completeness, is an XML vocabulary for specifying formatting semantics. Figure 28 shows the view of the example symbol gait as described in the previous chapters using a web browser.

The mandatory properties of the symbol are in the top. The list of properties is listed in a table. The table holds the property class as defined in the XML definition of the symbol (see Chapter 5.3.1). Using Xpath, the definitions of specific properties and the scheme of the property are included in the view of the symbol. Following the links, the property or schema can be viewed.

The symbols on which the viewed symbol depends are depicted in the second table. Additionally to the symbol class, which is displayed in the first column and taken from the definition of the viewed symbol, the name and the description of the referred symbols is attached.

The symbols that depend on the view symbol are not explicitly defined in a XML file. Instead, with XSL the dependencies element of all XML symbol definitions are inspected in order to find the symbols which depend on the viewed symbol. The third table displays the symbol class, name and its description.



SYMBOL ossGait

outer world snapshot symbol, version 1.0.0

describes a gait

Symbol Class: 0.0.0.4

Properties

Property Class	Property Name	Property Description	Property Schema ID	Property Schema Name
0.0.0.1	classSymbolicPosition	vector describing a point in a Cartesian axis system	0.0.0.1	schemaSymbolicPosition
0.0.0.8	classSymbolicVelocity	symbolic velocity	0.0.0.5	schemaSymbolicVelocity
0.0.0.9	classSymbolicAcceleration	symbolic acceleration	0.0.0.6	schemaSymbolicAcceleration
0.0.0.6	classSymbolicWeight	symbolic weight	0.0.0.4	schemaSymbolicWeight
0.0.0.3	classSymbolicLength	symbolic length	0.0.0.2	schemaSymbolicLength
0.0.0.4	classSymbolicWidth	symbolic width	0.0.0.2	schemaSymbolicLength

Dependencies on other Symbols

Symbol Class	Symbol Name	Symbol Description
0.0.0.1	omsFootprint	describes a human footprint
0.0.0.2	omsStep	describes the position of a human step
0.0.0.10	omsObject	describes the symbolic position and weight of an object
0.0.0.11	omsMovement	describes the symbolic movement by position, direction, and velocity.

Symbols depending on this Symbol

Symbol Class	Symbol Name	Symbol Description
0.0.0.4	orpPerson	describes how a person is represented

Description

This Symbol represents a gait which belongs to a human

Icons

Figure 28: Definition of the symbol gait as it is displayed in the browser

5.4 E-state and Symbols

In this work the basic emotional systems (e-systems) is introduced. The e-systems (see Chapter 2.2.2) follow the archetype of the mammalian brain with its basic emotional systems (see Chapter 2.1.3). Basic emotional systems valuate the current situation in which the individual is and influence the individual's state and actions. Since the ARS-PC part is the focus of this work, only the parts of the e-systems are defined which are integrated within this model. However, e-systems have to be seen as a system which correlates with the whole ARS system (see Figure 10).

Figure 29 depicts the part of the e-systems and their influence on the modules of the perception. The input from the sensors in the environment is first processed in the symbolization module, where the generated symbols are evaluated according to the situation perceived from the environment and the current state of the e-systems (3'). The perceived and evaluated situations influence for their part the state of the e-systems (1). The modulation module works like a filter, which generates a world representation according to the perceived situation and the state of the e-systems (3''). The e-systems are responsible for providing e-states of the entire system. Each e-state represents the status of an e-system. The e-status depends on the e-status itself (4) and the perceived situations.

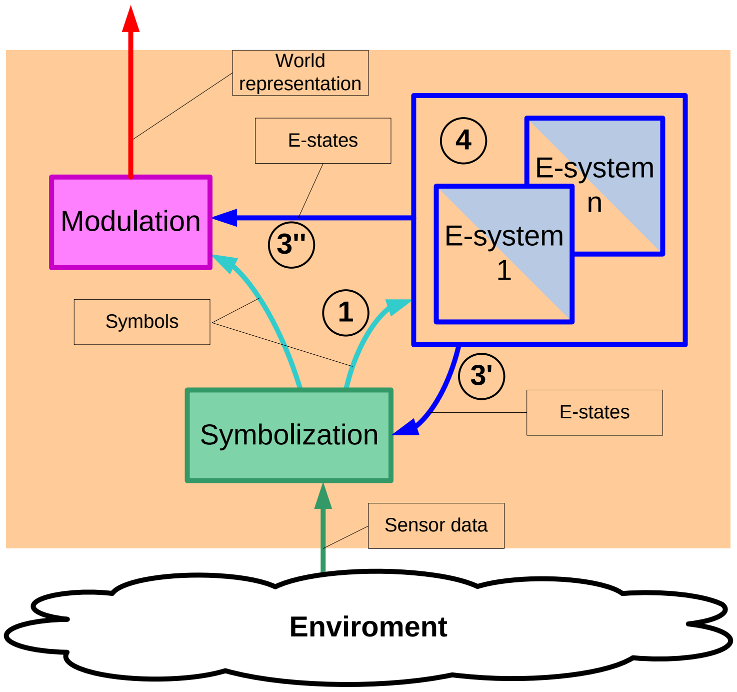


Figure 29: Perception and e-systems

In the ARS system the e-states are treated similarly to symbols. The e-state is used to propagate the global value of the state of an e-system to the modules. The global value is calculated from all symbols which are evaluated with the e-system and the value of the e-status in the past. Figure 30 shows the representation of an e-status by symbols. As described in Chapter 5.2 symbols have mandatory properties and a list of properties which are specific to it. Therefore, the e-state symbol has to have all mandatory properties. Since an e-state never expires, the lifetime of an e-state symbol is not set. The only property in the list of specific properties is the property e-status itself. It holds the system-wide value of the e-status.

Symbols (micro-, snapshot and representation) and scenario symbols are evaluated according to the e-systems. Since these symbols are not required to have all e-states used within the system, the symbol-inherent e-state is represented as a property within the list of symbol-specific properties. The value of the symbol-inherent e-state is calculated from the current situation and the current system-wide e-status. In the following chapter the transport mechanism of symbols is described.

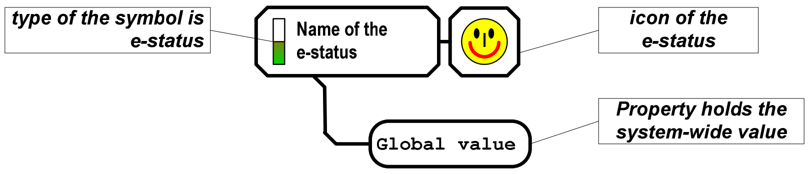

type of the symbol is e-status
Name of the e-status
icon of the e-status
Global value
Property holds the system-wide value

Figure 30: Representation of an e-status

5.5 SymbolNet

SymbolNet is a communication framework of the ARS project. [Pra06] briefly describes SymbolNet. It has been designed to distribute symbols as defined in this work. Therefore, it has to provide the necessary services to transmit the properties of the symbols. It has to be mentioned that SymbolNet is designed for the temporary processing of symbols. Since memory and processing power are limited in a computer, the main task of SymbolNet is to distribute the active symbols and their properties. SymbolNet does not implement memory like semantic memory or episodic memory as used in the ARS-PA model. In the following chapters the design of SymbolNet, its services and the included symbol repository are described.

5.5.1 Model of SymbolNet

SymbolNet provides functionality for generating, distributing, synchronizing and expiring symbols within the ARS system. In principle, symbols in SymbolNet are either stateless or stateful. If a symbol is stateless, the symbol cannot be changed after it has been created. It expires automatically. Symbols that are stateful can be changed during their lifetime. If a symbol is changed by a module within the system, the new values of the properties are propagated automatically by SymbolNet to all instances of the symbol. Figure 31 shows the possible lifecycles in SymbolNet.

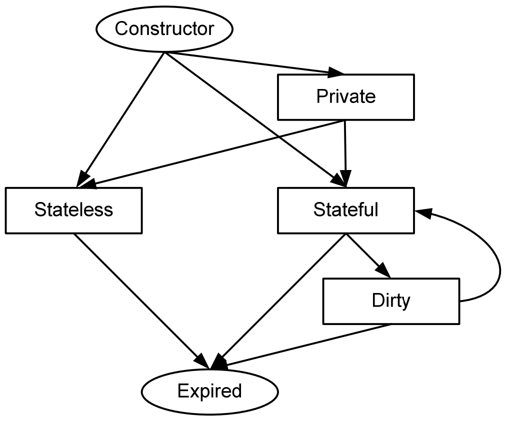


Figure 31: Lifecycle in SymbolNet

After a symbol has been created, it is private, stateful or stateless. A private symbol means that the symbol and its properties are not distributed to other modules in the system. It only exists in the module which created the symbol. However, after the properties of the symbol are set, the symbol is either stateful or stateless. SymbolNet sends the symbol automatically to all modules within the system. As mentioned above, stateless symbols cannot be updated and expire automatically at the end of their lifetime.

If a symbol is stateful, its properties can be updated. In case a module changes one ore more properties of the symbol, the symbol in this module is in the state *dirty*. This means, that its properties have only been changed within the module, but the changes were mot propagated to the other modules within the system. After SymbolNet has updated the symbol in all modules, the state of the symbol is set back to *stateful*. The lifetime of stateful symbols can be undefined. This means that the symbol does not expire automatically. To remove such a symbol from the system, a special message, the so-called expire message, has been introduced. The messages in SymbolNet are described in the next chapter.

5.5.2 Messages in SymbolNet

SymbolNet messages are used for the communication between the modules. SymbolNet provides three messages for handling symbols, i.e. new symbol, update property, and expire. Additionally, a so-called heartbeat message has been introduced to provide a timeframe for the whole system. After a symbol has been created and is in either the stateful or stateless state (see Chapter 5.5.1) of the module in which it was created, a new-symbol message is sent to all other modules. When the other modules receive this message, the symbol is created in theses modules. To save computational resources, the symbol is only sent to the modules that registered for it, which means that the module processes the symbols. If a symbol is not needed in a certain module, no messages are sent to it.

In case the symbol propagated with the new-symbol message is a stateless symbol, no further message is sent which refers to that symbol. If the new symbol is a stateful symbol, update-property messages are sent in case one or more properties of the symbol have changed. For stateful symbols a lifetime may be specified. If the lifetime is not specified, the symbol does not expire. However, whenever an expire message is sent, the symbol expires immediately despite its property lifetime.

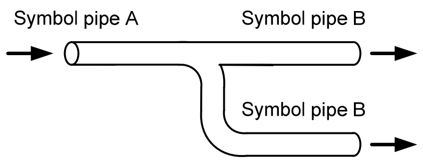


Figure 32: Pipe concept of SymbolNet

Figure 32 is an illustration of the pipe concept of SymbolNet. The pipe needs two types of ends, i.e. type A and B. Type A is the source of symbols. In SymbolNet a pipe can only have one end of type A. The type B is the sink of the symbol. A pipe in SymbolNet can have more than one end of type B. SymbolNet messages sent into the pipe (type A) are duplicated if the pipe has more than one end of type B.

This concept has been introduced in order to provide two properties of the communication framework in the ARS project. SymbolNet has to be portable to a broad range of systems with varying CPUs, operating systems and programming languages supported for the application using the communication framework. The second reason is to create a communication framework that is able to form a tree of symbols. Such a tree is necessary to form symbol trees as described in the following chapters.

The concept of the pipes as described above meets these requirements. Using TCP/IP as underlying communication protocol, SymbolNet can be used on a number of platforms, especially on PCs with various operating systems and embedded systems. A TCP/IP stack is available for a broad range of platforms. In order to have a representation and encoding of the data which is independent of the representation and encoding of data on different platforms, ASN.1 (Abstract Syntax Notation One) is used. ASN.1 [X.680] is a joint standard of ISO (International Organization for Standardisation) and ITU-T (International Telecommunication Union – Telecommunication Standardization Sector), which defines rules for describing the structure of objects that are independent of machine-specific encoding techniques. The Distinguished Encoding Rules (DER) [DER02] are used to encode the SymbolNet messages. DER is part of ASN.1 standard. Chapter 5.5.4 defines the symbols according to ASN.1 for SymbolNet. It has to be mentioned here that SymbolNet only propagates the representations of symbols but not the associations between them. The relationships between the symbols are shown in the graphical representation of symbols and symbol trees (see Chapter 5.2) and the formal definition using XML-files (see Chapter 5.3).

5.5.3 Structure of SymbolNet

A principal structure of how SymbolNet can be used is shown in Figure 33. The modules shown in Figure 33 are also called symbol factories. Symbol factories are intended to create new symbols, modify symbols, and expire symbols. They are used to build logical modules for handling symbols. A symbol factory can be used for every layer as proposed in Chapter 2.2.3. Thus, a possible arrangement of modules would be a microsymbol factory, a snapshot symbol factory, and a representation symbol factory.

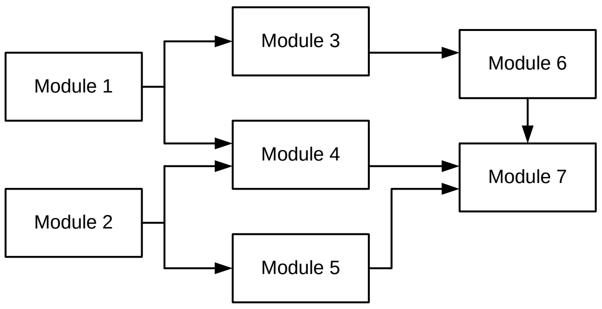


Figure 33: Principal communication structure of SymbolNet [Pra06]

However, SymbolNet supports also more complex structures than depicted in Figure 33. In this case, the symbol factories are split up again so that the microsymbol layer consists of two (independent) symbol factories. The second hierarchy (snapshot symbol layer) is split into three symbol factories. One symbol factory (module 4) has inputs from both microsymbol factories. The representation layer is also split into two modules, module 6 and module 7. Module 6 gets input from snapshot symbol factory module 3. Module 7 gets input from the snapshot symbol factories module 4 and module 5 and additionally from the representation symbol factory module 6.

It is important to note that the system designer has to monitor the stability of the system. SymbolNet allows building loops so that symbols in lower hierarchy levels depend on symbols of higher levels. If the higher symbols also depend on hierarchically lower symbols, the system might become unstable. So when using methods for creating the symbols (both the higher and lower level symbols) one has to take into account that they are part of a loop[13].

5.5.4 Symbol and its Properties

ASN.1 is used for the representation of symbols in SymbolNet. In principle, the formal definition of symbols using XML-files (see Chapter 5.3) could also be used with some extensions (a unique identifier for each instance of the symbol). Nevertheless, the XML definition was introduced into this project when the first version of SymbolNet was already in use. Moreover, the ASN.1 approach allows a more efficient encoding than XML, and therefore less data have to be transmitted. However, the ASN.1 XML Encoding Rules (XER) [X.693] specify rules for encoding values of ASN.1 types using XML.

The predefined properties of a symbol are identifier, type, class, timestamp, and optionally the property lifetime. A list of properties specific to every symbol is used to specify the specific properties. The predefined properties of a symbol are defined as follows:

```
Symbol ::= SEQUENCE {
    id           Int64,
    type         SymbolType,
    class        Int32
    timestamp    Int64,
    lifetime     Int32 OPTIONAL,
    properties   SEQUENCE OF Property OPTIONAL
}
```

The first property id is a system-wide unique identifier of the symbol. The data type of the id is a 64-bit integer. The value is generated during the creation of the symbol. In the XML definition of the symbol this value is not present. The type of the symbol defines the world (inner or outer) and hierarchy level (micro, snapshot, or representation) to which the symbol belongs. The data type of the property type is SymbolType. In the XML definition of the symbol the type is split up into a hierarchy level (`symbolType`) and the world to which the symbol belongs (`symbolWorld`). The property class specifies the symbol, and therefore which properties are defined in the list of individual properties. The data type is a 32-bit integer. The equivalent in the XML-definition is the element `symbolClass`.

[13] In this context loop means a loop in the perception part of the ARS system (ARS-PC). As shown in Figure 1, the ARS system forms a loop together with the environment. This loop always exists because the ARS system is intended to obtain feedback from the environment.

The property timestamp[14] is a 64-bit integer value and defines whether the symbol has recently been changed. Together with the lifetime it defines how long the symbol is valid within the system. The property lifetime is an optional property. If it is not defined, the symbol does not expire automatically. The symbol can only be removed from the system with an expire message from SymbolNet (see Chapter 5.5.2). In the XML-definition the time information is not included. The property *properties* holds a sequence of properties which are specific to the symbol. As mentioned above, the property class defines which properties are within the list of specific properties. In the XML-definition the specific properties of the symbol are encapsulated in the element `properties`. The specific properties of ASN.1 are defined as follow:

```
Property ::= SEQUENCE {
    class              Int32,
    schema             Int32,
    confidence         REAL,
    value              SEQUENCE OF CHOICE {
        pInt64         Int64,
        pInt32         Int32,
        pInt16         Int16,
        pInt8          Int8,
        pIeeeDouble    IeeeDouble,
        pIeeeSingle    IeeeSingle,
        pNull          NULL
    }
}
```

Every property within the property list is identified with the property class. It defines the semantic meaning of the property of the symbol. The schema defines how the data are represented. In case there is only one representation schema for a property, the schema is redundant information because the definition is already included in the property class. Nevertheless, introducing the property schema has two advantages. The representation of the value of the property can be changed without changing the symbol itself because the property class does not change. And secondly it allows the reuse of code for different symbols using the same representation of data. The following examples are given to provide further clarification.

Let us assume two symbols have a property position. The semantic meaning of this property is the current position of the symbol within the system. Therefore, the property class is identical. But the position can be represented in different ways, e.g. different coordinate systems, data types, with or without confidence information, or two- or three- dimension ional. For every representation a schema is defined. So although the symbol property class of two symbols is identical, the representation of the data may differ based on different schemas.

In the second example, it is assumed that a symbol has two properties, one describing the current position, and one giving the starting point. Since the properties have different semantic meanings, the property

[14] Since SymbolNet is a technical implementation suitable for modern processors, a mechanism had to be introduced which allows a chronological ordering of symbols. The concept of timestamp is a commonly known concept for the chronological ordering of data structures, and therefore supported by programming languages and existing software like databases.

class differs. Nevertheless, the schema of these two properties can be identical if the same representation is used for both positions. Therefore, the code for calculating the current position can also be used for calculating the starting position. Further, and especially in the case of geometric information, the schema can be used to determine whether a transformation of data is necessary. If the distance between the starting position and the current position has to be calculated, the property schema is used to determine whether the two positions use the same representation.

The property confidence is intended to describe the confidence of the property. Confidence means here the confidence of the property defined with the property class. Additionally, confidence can be defined for every value in the value field. A position in Cartesian coordinates could define a confidence for every coordinate axis. If for example the position of an object is determined with a light barrier, two axes of the position can be determined with a high confidence, whereas the third dimension can not be determined. Therefore, the confidence of the two axes which can be determined is high, the confidence of the third axis is low. The confidence of the property is also high because the method to determine a position of an object with light barriers has a high confidence.

The value of the property is a list of values. The number of values and their semantic meaning are defined by the property schema. As mentioned above, the semantic meaning of the property of the symbol is defined with the property class. The data type of the value can either be an 8-, 16-, 32-, or 64-bit Integer, an ANSI/IEEE 754 [IEEE754] single or double precision floating point value, or null. Null is used to set the value of a property to a value which represents an undefined value. In the XML definition the elements of the list of properties are defined in various files as described in Chapter 5.3.

A symbol defined according to the schema presented above is suitable for propagation with SymbolNet (see Chapter 5.5). Nevertheless, in order to define the relationship between symbols (building symbol trees), an additional definition is necessary. The principles of how SymbolNet and the symbols can be used are discussed in the next chapter.

5.6 Usage of Symbols

The chapters above present the definition of symbols and their transport mechanism, SymbolNet. SymbolNet provides three functions to manipulate, generate, update, and expire symbols. The symbol factories use these functions to associate the sensor values and symbols with each other in order to perceive types of situation and scenarios.

In a microsymbol factory, microsymbols are processed. Correspondingly, snapshot symbols are processed in the snapshot symbol factories and representation symbols in the representation symbol factories. They form a hierarchical structure as shown in Figure 25. In general, the rules for the generation of a symbol are implemented in the symbol factories. In principal, it is possible to use every kind of decision making process for the manipulation of symbols. In the ARS-project, different strategies are implemented. A rule-based approach is shown in [Pra06] and [Goe06]. This approach ensures a predefined behavior. The approach using Fuzzy Logic in [Ric07] allows the definition of rules in a more flexible way. Statistical models (Hidden Markov models) are used in [Bru07]. This approach uses no rules, but extracts a characteristic behavior from inputs and can be used to detect unusual behavior. Further, it is planned to use an artificial neural network and fuzzy inductive reasoning [Ayy06 and Cel91] in order to manipulate symbols. This approach will make it possible to introduce learning into a symbolic approach.

[Pra06] proposes that the symbol trees are hierarchical and exactly three layers are used, i.e. microsymbol layer, snapshot symbol layer and representation layer. This fits to the model derived from the human brain as shown in Chapter 2.2.3. Nevertheless, [Pra06] proposes that the microsymbols and snapshot symbols are the representation of the world at one moment in time or a short period of time, which does not fit to Luria's neuro-psychoanalytic model. According to this model, the second level is not the world representation of a short period of time but combines the lower-layer symbols (of one sense) to more complex symbols. In the next higher layer, i.e. the representation layer, a world representation is generated.

The implementations mentioned above ([Goe06 and Ric07]) experienced a number of problems in their attempt to accommodate [Pra06]'s constraint of the definition of snapshot symbols. An example of this is the symbol gait. As described in more detail in Chapter 6.1.4, the symbol is a snapshot symbol which depends on the microsymbols step and footprint. The microsymbols represent a short time span, but the snapshot gait is valid for as long as a person can be tracked. Therefore, the definition in [Pra06] according to the time constraints of symbols should be reduced to the integrative functionality. This means that a symbol of higher hierarchical levels is composed from symbols of lower levels.

The hierarchical structure of the layers ensures a stable system because the symbols depend only on symbols of lower hierarchical levels but not on symbols of higher levels. However, the implementation phase showed that a strict hierarchical model is not always suitable. The people tracking application, described in detail in Chapter 6.1, can also use cameras for tracking people, as shown in Figure 36. The output of the (complex sensor) camera is used by the snapshot symbol gait and the representation symbol person. In a strict hierarchical structure of symbols, only microsymbols and no higher-layer symbol depend on sensors. However, if a strict hierarchy of symbols is not observed, the designer of the symbols has to take into account that loops within the symbol tree may lead to instable systems. Chapter 6 will show how symbol trees can be used for special applications.

5.7 Sensor database

The sensor database is used to store data from real implementations like the kitchen on the ICT and from virtual environments generated with the simulator (see Chapter 7.1.3). In Figure 34 the tables of the sensor database are depicted. The table ARS_SEN_TYPEDEF defines the data types which can be used for the values of the sensors. In the current implementation sensors with binary values, integer values and floating point values can be used. The type of sensor is defined in the table ARS_SEN_DEFINITION, e.g. tactile sensor or temperature sensor. In the table ARS_SEN_PROPDEF properties of the sensor are defined like the position where the sensor is mounted or where the active area is.

The table ARS_SEN_ENVIROMENT is used to define the environment, e.g. the building in which the sensor is located. Within that environment objects can be defined. These objects are e.g. rooms and are stored in the table ARS_SEN_OBJECT. A sensor is defined in the table ARS_SEN_SENSOR. In this table the sensor id, the object id, the environment id, and the sensor definition id are defined. The sensor definition id refers to the table ARS_SEN_DATADEF. This table defines the value type (e.g. Boolean) of the sensor. The properties of the sensor (e.g. position) are defined in the table ARS_SEN_PROPERTY_INT.

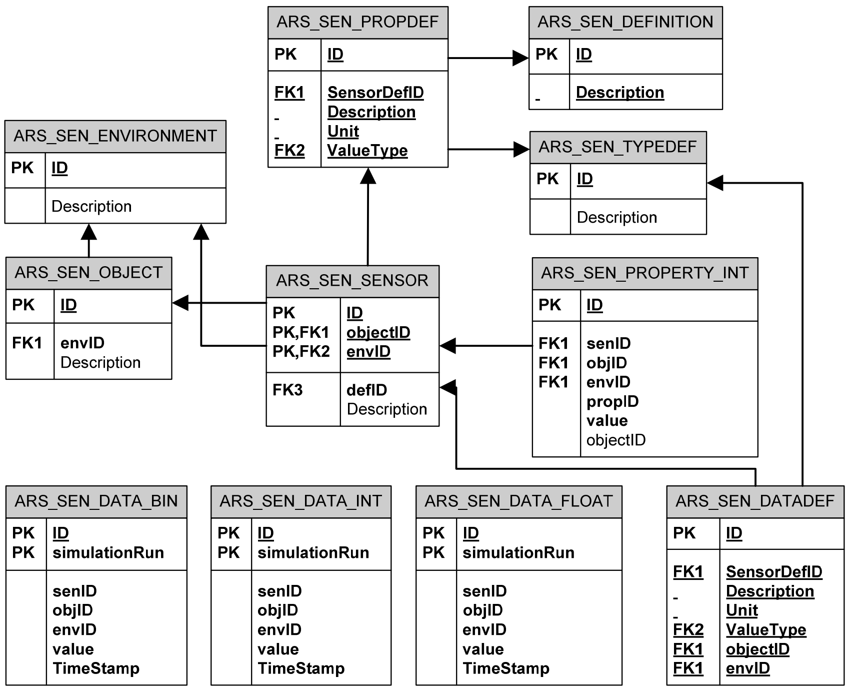


Figure 34: Sensor database tables

The tables described above are used to define a sensor, its properties, and the environment in which it is mounted. Although this structure is quite complicated, it is a very flexible way to store the required data for the ARS system. However, the sensor values are stored in the tables ARS_SEN_DATA_BIN (binary values), ARS_SEN_DATA_INT (integer values), and ARS_SEN_DATA_FLOAT (float values). Beside the value, the unique definition of the sensor (sensor id, object id and environment id) and the timestamp, of when the value was generated are stored in these tables. Additionally, the value simulation run id has been introduced. This id is intended to be used by simulators. In order to store different simulation runs of the same environment, only the simulation run id has to be changed to generate unique sensor values, and the timestamp of a simulation can always start with the same value.

The sensor database is used to store the input data of the ARS-PC system. The output of the ARS-PC system is also stored in a database as shown in Figure 25, namely the symbol database. This database is described in the next chapter.

5.8 Symbol Database

Similar to the sensor database, the symbol database has been introduced into the ARS framework to ease development. The database is not necessary for the functionality of the system. Nevertheless, the symbols generated by the ARS system can be stored in the database in order to have a logger of the events recognized by the system. In the following, the tables of the database are briefly described.

The tables of the symbol database are shown in Figure 35. Like the sensor database the symbol database allows the storage of data from different (simulation) runs which have the same timestamp. In the table Run, a simulation is described with the unique id, the username of the person who generated the run, date and description. The table message holds the messages received from SymbolNet. The message is characterized by a sequence number, the time (from the database) when it was received, the timestamp of the SymbolNet message, the type of the message, and the type of run it belongs to.

In case the received message is a heartbeat message, the message is stored in the table Heartbeat. If a new symbol message is received, a new entry is generated in the table Symbol, identified with the symbol id, the class and type of the symbol, and the lifetime if available. If an expire message is received, the reference to the expire message is inserted in the entry mentioned above. The properties of the symbol are stored in the table PropertyInstance. The class and schema together with q reference to the symbol to which the property belongs identify the property. Additionally, a reference to the message (new symbol message or update property message) is stored as well as the confidence of the value. The value itself is either stored in the table PropertyValueInt in case of an integer value or in the table PropertyValueFloat in case of a floating point value.

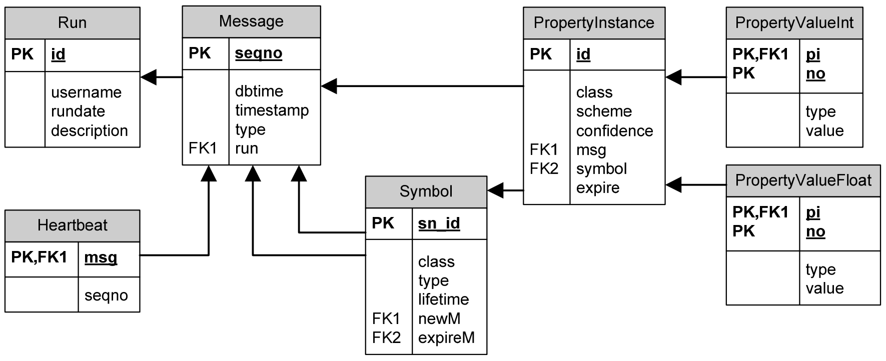


Figure 35: Sensor database tables

The database is mainly used to log the symbols created by the ARS-PC system, but also to buffer the symbols for the three-dimensional visualization of the simulator as described in Chapter 7.1.3. The implementation of the database is described in Chapter 7.4. The recorded symbols can be visualized with various visualizations. These visualizations are described briefly in the next chapter.

5.9 Visualizations

The ARS Framework offers various types of visualizations, i.e. a logical visualization, a two-dimensional visualization and a three-dimensional visualization. The logical visualization shows the symbol tree of the system. Within this tree it is shown which symbols and how many of each kind of symbols are currently active. Additionally, the dependencies between the symbols are shown. This type of visualization gives an observer a good view of the system.

The two-dimensional visualization is based on the floor plan of the kitchen of the ICT as depicted in Figure 24. In this floor plan, the sensor activities and symbols can be displayed. In order to get a clearer view, the visualization can be manifold. A filter can be defined for each type of visualization. For example, the sensors and microsymbols are shown on one floor plan, the snapshot symbols are on the second and the representation symbols on the third plan. Thus, it is possible to get an impression of the spatial relationship of symbols and sensor activities. The implementation of the two-dimensional visualization is described in Chapter 7.5.

The three-dimensional visualization is based on the simulator. It shows the virtual environment used in the simulator. Symbols can be displayed in this three-dimensional representation of the virtual environment. As is the case with the two-dimensional visualization, a filter can be used to mask symbols which should not be displayed. With these three different types of visualization an observer gets different views of the system. This allows the analysis of the association process of the system, and therefore its optimization. The implementation of the three-dimensional visualization is described in Chapter 7.1.3.

6 Definitions of Symbols for Applications

In this chapter, the symbols for the applications as defined in Chapter 4 are introduced. The applications are grouped into the areas where they are intended to be used. The people tracking application is the basis for all other applications and determines the position of the persons in buildings. The child safety application was developed to detect dangerous situations for children e.g. danger of burning at a hot stove. Fall detecting scenarios of the geriatric care application are intended to be used in buildings for elderly people.

The definition of the symbols follows their description, representation (see Chapter 5.2) and transport mechanism SymbolNet (see Chapter 5.5). Furthermore, these definitions are intended to be used with e-systems (see Chapter 5.3). The sensor values used to generate the (micro-)symbols are intended to be provided from a simulator, form a special installation as shown in [Goe06], a fieldbus system, or DPAL as proposed in Chapter 3.5.

6.1 People Tracking

Carrying out the research for this work, it emerged that a key application of the ARS project needs to be the tracking of people. In almost every application defined within the project, the position (or at least the presence) of people has to be detected. This chapter describes how the people tracking application is defined within the ARS project, and therefore how the tracking of people in buildings could be carried out in the future.

Approaches to track individuals are visibility of the individual, ground reaction force caused by individual, sound caused by individual (by voice or movement), different temperature from the surrounding environment, or tags like RFID tags. A substantial amount of research work has already been carried out in the area of detecting people with a camera. One example of an open-source implementation of computer vision is OpenCV library [OCV06]. Amongst others, face and gesture recognition, motion tracking, and object identification are implemented in this library.

Another biometrical feature of an individual is his or her ground reaction force, which can be measured with pressure sensors in the floor. [Orr00] and [Add97] explain how the recognition of footsteps on a sensor pressure field and their assignment to an individual is possible. [Orr00] claims that the system has a recognition accuracy of 93% for a user who is known to the system.

In addition to the footsteps, e.g. the human voice can be used to identify and localize a person. By using microphone arrays, it is possible to localize the sound source, and it is even possible to infer face-to-face conversations [Bia04] from this. As well as cameras sensitive to the visible spectrum, infrared cameras can be used to detect people in a room. Even simpler types of infrared sensors in connection with an analysis system (e.g. based on Bayesian networks) are enough to track persons [Tai05].

However, the methods mentioned above, focus on one type of sensor. If in an environment the sensor fails, e.g. it is to dark that a camera can take pictures, the recognition is not possible any more. Therefore the ARS system is designed that it can recognize situations based on diverse and redundant sensors. The symbols and sensors, which are used for the application *people tracking* in the ARS, are shown in Figure 36.

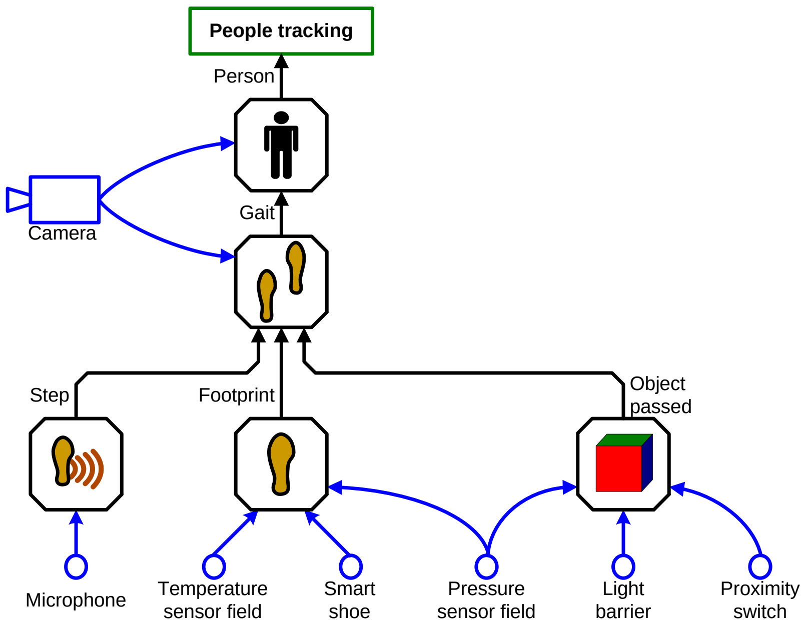


Figure 36: Dependencies of symbols and sensors for the people tracking application

Starting at the bottom of the figure, the microsymbol *step, footprint,* and *object passed* are generated from different kinds of sensors like light barriers, pressure sensor fields, Smart Shoes[15], microphones, cameras, or temperature sensor fields. The microsymbols *step, footprint,* and *object passed* are combined to the symbol *gait*. The camera is not treated as the other sensor where information output forms microsymbols, but as an object creating symbols. Thus, the output of a camera could be used to verify the symbol *gait*. The symbol *person* depends on the symbol *gait*. Again, a camera is used to recognize a person and to validate the results from these symbols. The following chapters describe the symbols used for the application *person tracking*.

[15] Smart Shoe is a shoe, which sends out position signals (e.g. GPS signals or RFID reader together with the carpet RFID-Tags http://www.heise.de/newsticker/meldung/70782) for every step made with it. This signal can be received and used for the localization of the wearer of the Smart Shoe.

6.1.1 Symbol *footprint*

The symbol footprint describes an individual's footprint. In our context, footprint means not only a step, but the sequence of imprints beginning with the heel, and followed by the whole foot and lastly the toes, as shown in Figure 40. Figure 37 shows the properties of the symbol footprint.

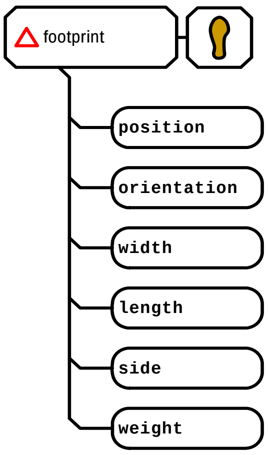


Figure 37: Symbol *footprint*

Figure 38 shows how the dimensions of a footprint are defined. The position of the footprint is in the origin of the coordinate system. In the symbol *footprint* this position is represented with the property *position*. The property *orientation* defines the orientation of the footprint (y-coordinate of the footprint) within the reference system (e.g. room). The property *length* is the length between the toes and the heel the property *width* is the broadest part of the footprint. The origin of the footprint is in the middle i.e. half length and width.

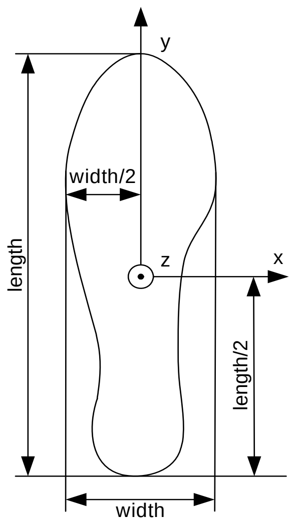


Figure 38: Orientation and dimension of a footprint

The property *side* describes whether the footprint is the left or right foot of the person. The weight of the person is given in the property *weight*. It is assumed that the person is walking, and when a step is made the weight of the person weighs down on the one foot that has ground contact. The dynamic change of the weight (more precisely the force which is measured by pressure sensors) is not taken into account with the property *weight*. If the person stands still, his or her weight is equally distributed on both feet. In this case, the property *weight* represents half of the person's weight.

A typical curve of the value of the property *weight* is shown in Figure 39. In this figure, the y-axis shows the normalized value of the property *weight*, the x-axis the time. In case the person stands still, the value of the property weight is half of the value of the person's weight. When the person is walking, the value of the property *weight* alternates between zero and the value of the person's weight.

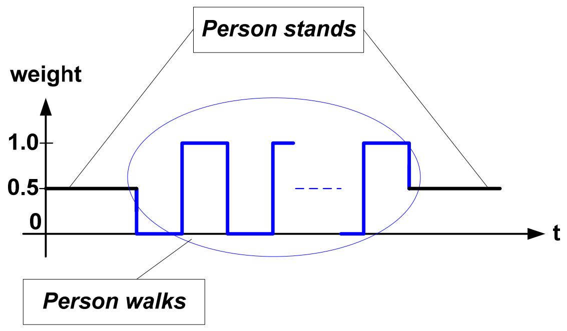


Figure 39: Behavior of the value of the property *weight*

The pressure and/or temperature sensor field or Smart Shoes can be used to generate the symbol *footprint*. If a pressure sensor field or a temperature sensor field is used, the detection of the symbol *footprint* follows the principle shown in Figure 40. When the footprint starts, the person first activates the sensor A with the heel. Next, the toes touch the floor and trigger sensor B, as shown in the middle of Figure 40. When the person continues to walk, s/he first raises the heel, which deactivates sensor A. If this sequence of activation of sensors is detected, the symbol *footprint* is generated. The property *position* is calculated from the position of the sensors; the property *orientation* is derived from the spatial and temporal sequence of the step.

Using pressure sensor fields, the weight of the person can be determined from the force measured by the sensors. If the resolution of a sensor field is high enough (e.g. an area of 1 cm² per single sensor), also the width, length, and side of the footprint can be determined.

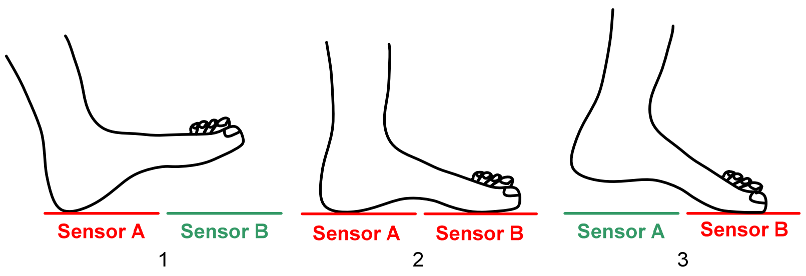


Figure 40: Detection of a human footprint

The third possibility mentioned above to determine the properties of the symbol *footprint* is a Smart Shoe. A Smart Shoe is a shoe which is able to determine its position. There are different possibilities to measure the position within a building. One of them is the usage of RFIDs situated in the floor. Each of the RFIDs has its position stored within the coordinate system. If the smart shoe touches the floor, it reads out this information from the RFID. Sensors within the shoe can measure the weight of the person. The length, width, and side of the shoe are known parameters and can be sent together with a unique id to the module, dedicated for the generation of the symbol *footprint*. This module sends the unique id together with the position of the RFID in the floor, and therefore obtains the position of the footprint. The disadvantage of this approach is that the person always has to wear the shoe; otherwise footprints would not be detected within the system.

The symbol *footprint* is updated when there are significant changes to the weight. Such changes can be caused by a person standing still before starting to walk. On the foot which remains standing, the weight is then approximately double of what has previously been measured. On the contrary, the weight is cut in half when the person stops. The symbol footprint expires when the foot is raised and no sensor is activated.

6.1.2 Symbol *object passed*

The symbol *object passed* indicates that an object has passed a defined place. Figure 41 shows its properties. The difference from the symbol *footprint* (see Chapter 6.1.1) is that the symbol *footprint* represents only the footprint of an individual, whereas the symbol *object passed* appears always when one or more sensors within the system detect an object.

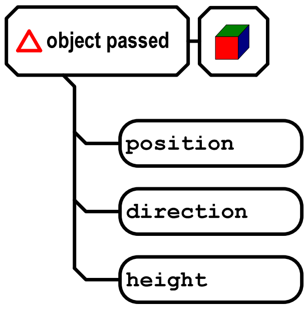


Figure 41: Symbol *object passed*

The property *position* is the position where the object is recognized. The property *direction* is the direction of the object[16]. The property *height* is the height of the object. The symbol *object passed* can be derived from pressure sensors, light barriers, or distance sensors.

If a pressure sensor on the floor detects a change of the pressure, the x- and y-coordinates of the position where the object has been detected can be determined. Every time the symbol *footprint* (see Chapter 6.1.1) is generated, the symbol *object passed* is generated as well. Light barriers are sensitive in one direction and have a very small beam angle. Therefore, it can be assumed that the object passes along a

[16] A property *velocity* could also be introduced but is not necessary for the ARS system. It can be assumed that within this system only people move (or move something), and therefore the velocity would not differ in a wide range.

line. If a distance sensor is used instead of a light barrier, the object with the smallest distance to the sensor can be determined.

The examples in Figure 42 show how these sensors can be used to derive the height e.g. of a person who passes a door. On the left-hand side, light barriers are mounted in the doorframe, and their sensitive area is horizontal to the opposite post. Depending on the height of a person, the light barriers are interrupted on a certain level. Therefore, the height of the person lies between the height of this level and the next higher light barrier. The distance between the light barriers limits the resolution of this method. The application in the ARS system uses the height of an object primarily to determine whether a person is a child or an adult (see Chapter 4.4). For this purpose, a resolution (distance between sensors) of 20 cm is sufficient.

If a higher resolution is necessary, more light barriers can be used (it might be sufficient to use more sensors in a special area e.g. between 160 cm and 190 cm, if it is necessary to get a higher resolution of the height of adults). Another possibility is to use distance sensors which are mounted on the lintel of the door as shown on the right side of Figure 42. The sensitive area is vertical down to the floor. If a person passes the door, the lowest distance measured by the sensor is used for further calculation. The highest point of the person is the height where the sensor is mounted minus the lowest distance measured by the sensor. To increase the probability that the head is the highest measured point, it is proposed to mount several distance sensors in the specific area. For a normal door (e.g. 90 cm width), three sensors are proposed as shown in Figure 42.

It is clear that both methods can lead to faulty results, especially if the person passing the door is not cooperative, i.e. raises the hand or bends the head when passing, which would lead to an incorrect measuring of his or her height. But even in the case of a person being cooperative but carrying something (e.g. a long object), the height could be measured incorrectly. Therefore, it is proposed to use additional methods to measure the height. One possibility is computer vision. Another possibility is the use of tags which identify a person and access his or her height from a database.

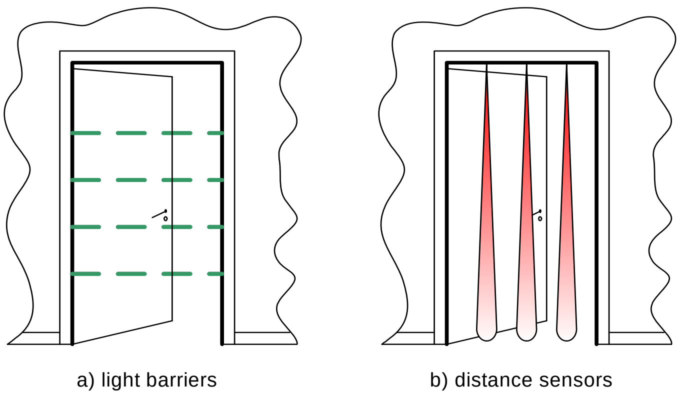


Figure 42: Examples for mounting sensors on a door for detecting the height of a person

The property *direction* of an object can be determined if at least two sensors (both either light barriers or distance sensors) are used, which are mounted on two planes parallel to the closed door, marked as (1) and (2). On the left side of Figure 43, it is shown that both light barriers on both planes have their own sensitive area. On the right side, three distance sensors are illustrated, one sensor (1) has its active area on the first plane closer to the room, and two sensors (2 and 2') on the second plane closer to the corridor. The direction of the person can be derived from the temporal relationship of the trigger of the sensors. If a person enters the room from the corridor (direction: entering the room), first the sensors in plane 2 are activated and then the sensor in plane 1. Leaving the room, a person first triggers the sensor in plane 1 and then the sensor in plane 2.

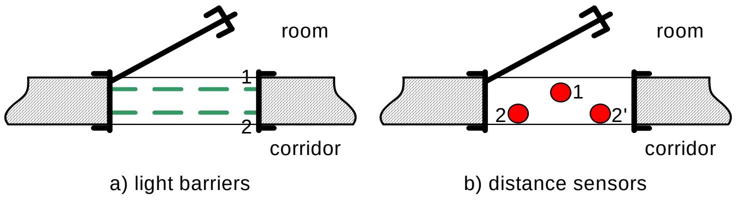


Figure 43: Example for mounting sensors on a door for detecting the direction of a person's motion

The symbol *object passed* expires when the stimuli from the sensors, which caused the generation of the symbol, are again inactive. If more than one sensor causes the generation, and one or more continue to be active, the symbol is not deleted.

In case the sensors detect an object at a special place like the door, additionally to the symbol *object passed* the symbol *person enters* is generated. This symbol is identical to the symbol *object passed* (in terns of properties and ways of generating it) but indicates with its type the special place where it was generated. The symbol *person enters* has been introduced into the system in order to ease filtering of symbols. For example for the generation of the symbol *person upright* (see Chapter 6.3.6) it is necessary to recognize when a person enters a room. If the symbol *object passed* is used instead of the symbol *person enters*, all symbols *object passed* within the system would need to be checked to determine whether this object passed a special position like the door.

6.1.3 Symbol *step*

The symbol *step* describes the step of a person. The difference from the symbol *footprint* (see Chapter 6.1.1) and *object passed* (see Chapter 6.1.3) is that it only occurs if a person's step has been recognized acoustically. Figure 44 shows the structure of the symbol *step*.

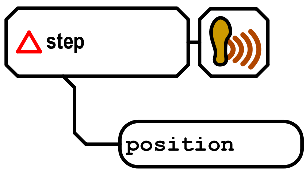


Figure 44: Symbol *step*

The property *position* describes where the step was recognized. It is generated when a step has been recognized acoustically, e.g. with a sound localization system as described in [Bia04]. The property *position,* and therefore the symbol *step* are never updated. Since a single step of a person can only be recognized within a fraction of a second, the symbol expires immediately after it has been generated. It just has to be guaranteed that the system is able to recognize the symbol. For such cases, SymbolNet (see

Chapter 5.5) offers the possibility to define a symbol's lifetime at the time of its generation. It is ensured that every module subscribed to a certain symbol receives this symbol, even if, a symbol has already expired at the time of reception due to delays within the system.

6.1.4 Symbol *gait*

The symbol *gait* describes the gait of a person. It is assumed that the person to whom the gait belongs walks in a regular fashion, meaning s/he does not limp, jump, dance, or do something similar. If more than one person is in a room, it is assumed that they do not collide. Figure 45 shows the structure of the symbol *gait*. The properties *position*, *velocity*, *acceleration,* and *orientation* define the current kinematics of the gait. Figure 46 shows the frame of reference (room) and the reference point of the gait. The vectors for orientation, position, velocity, and acceleration of the gait are within the frame of reference of the room. The vectors originate at half length and half width between two footsteps and therefore move within the frame of reference of the room when the person moves. The property *weight* is the weight of a person. The properties *width* and *length* describe the length of the last step and the width between the last steps according to Figure 46[17].
The symbol *gait* is usually generated near special points (e.g. front door). Only the property *weight* is a quasi-static property[18] all other properties are dynamic. These properties may be updated by the symbol *footprint* (see Chapter 6.1.1) or symbol *step* (see Chapter 6.1.3). Alternatively a symbol generated with a computer vision system could be used.

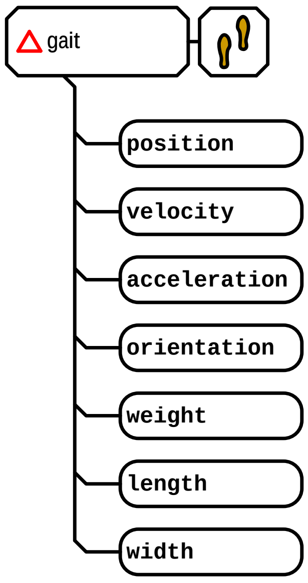


Figure 45: Symbol *gait*

[17] It is assumed that the last two steps recognized are a left and a right footstep. If not, the properties *width* and *length* should not be updated because it would not fit into the definition of Figure 46.

[18] This assumes that the person does not carry anything. The change in a person's weight can therefore be used to detect whether a person carries something or not.

If one of these symbols is available and it fits to a current symbol *gait*, that symbol is being updated. In this context fit means that the position of the new symbol has to be within a certain range. This range can either be defined statically (e.g. the maximum step length) or be calculated from past behavior. Thus, if a person has always made short steps in the past, it is assumed that the next step will also be a short step.

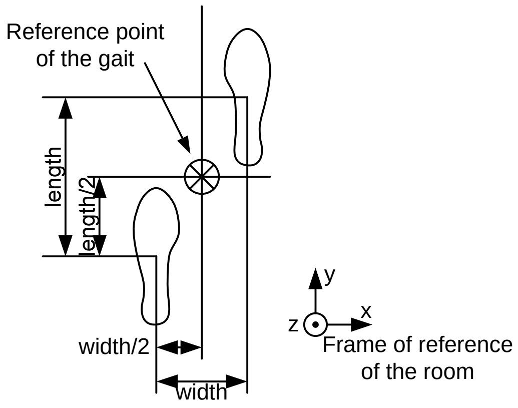


Figure 46: Orientation and dimension of the gait

The activity is calculated from the activity of the symbol which triggered the update and the plausibility of the parameters length and width of step. If the symbol *gait* is updated, the symbol which caused the update is inserted into the reference list. To reduce the calculation power, only footprints “near” the current position shall be observed. The symbol expires if the person is outside the scope of the system (e.g. leaves the building).

6.1.5 Symbol *person*

The symbol *person* describes a person and his (physical) properties. Figure 47 shows the properties of the symbol *person*. The properties *position, velocity, acceleration,* and *orientation* define the current kinematics of the person. Figure 48 shows the frame of reference (room) and the reference point of the person. The reference point of the person is in half the person’s height. The property *height* is the person’s size. The property *weight* is the person’s weight. If an application needs additional (physical) properties of a person, they can be added to the symbol. In principle, non-physical properties like mood can also be added to the symbol.

The property *type* defines whether the person is a child or an adult. If necessary for other applications, other categories like the person’s role (caregiver, patient…) can be added. The property *action* describes what the person is doing. The actions stand, sit, walk, and usage have been defined in [Pra06]. The property *action* is not used in this work. However, for the geriatric care application (see Chapter 6.3) symbols have been defined, which indicate the action of the person.

However, in this work human activities applications are introduce in Chapter 6.4 instead. The reason was that with additional applications the action property would get additional actions. But more values of the property *actions* will cause more SymbolNet update messages since it has to be assumed that the value of the property changes more often. This means, that every symbol dealing with the symbol *person* gets a SymbolNet update message when the action of the person changes. That leads to a waste of computational resources, in case the value of the property *action* is irrelevant. Further In case a person performs more than one action simultaneously, the property *action* as introduced is not suitable since SymbolNet allows only one value for a property.

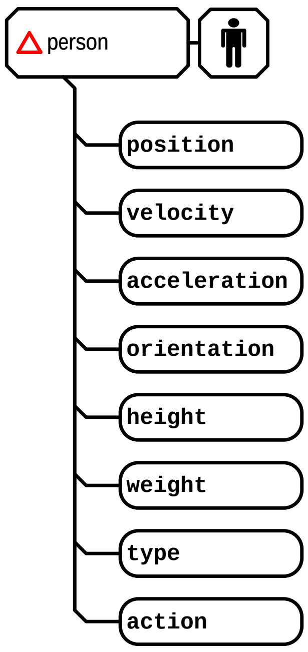


Figure 47: Symbol *person*

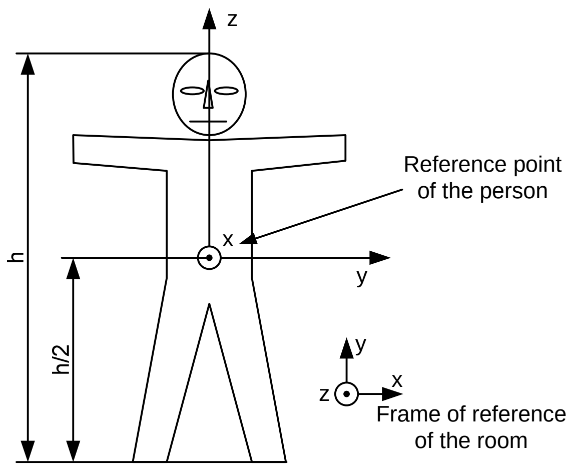


Figure 48: Orientation of the person

The generation of the symbol will usually be triggered at a special place, e.g. the front door. At this place sensors could be concentrated to determine most of the properties of the symbol. Dynamic properties are of course updated periodically, whereas the static properties will not change during the lifetime of the symbol. Since the ARS system assumes that only people can move themselves, a new symbol *person* is generated if a symbol *gait* (see Chapter 6.1.4), *step* (see Chapter 6.1.3), and *object passed* (see Chapter 6.1.2), or the symbol from the camera cannot be assigned to any symbol *person* in the system. If such a symbol can be assigned to the above-mentioned symbols, the properties are updated. The symbol will be destroyed if the person it represents leaves the observed area.

6.2 Child Safety

Of course there are various sources of danger in homes and buildings, especially for children. The following example describes a situation where a child without an adult nearby approaches a stove with hot plates. Since children are curious, they are attracted to usually busy places like stoves. They observe their parents cooking there and want to inspect that place. Unfortunately, this thirst for knowledge can lead to burns on the child's hand or all over its body, in case there is for example a pot with hot water on the stove. Therefore, the application *child safety* has been designed to prevent such dangerous situations for children.

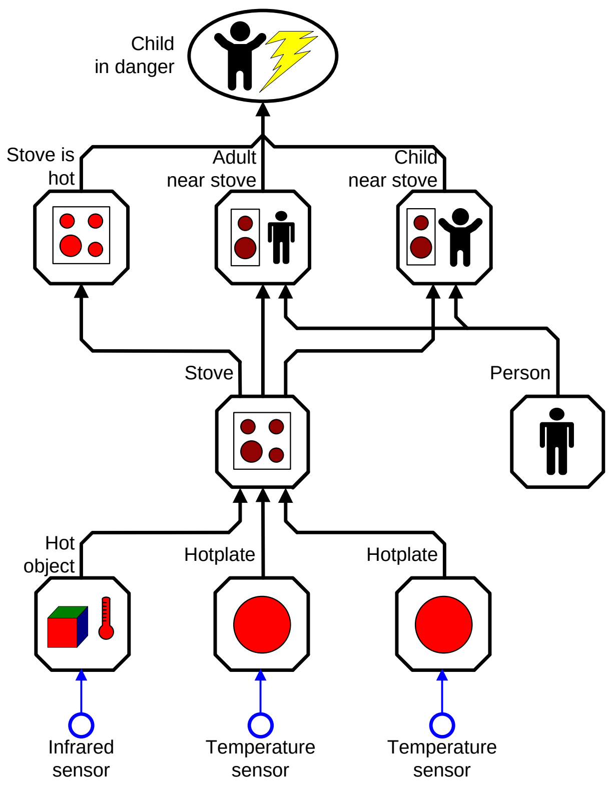


Figure 49: Child safety application

The symbols used in the application *child safety* are shown in Figure 49. The application uses the symbol *person* (see Chapter 6.1.5) as also used in the application *people tracking* in Chapter 6.1. For simplification, the symbols on which the symbol *person* depends are left out in Figure 49. The symbol *stove* depends on the microsymbols *hotplate*, which on its part consist of the values from temperature sensors and the microsymbol *hot object*. The symbol *hot object* is derived from infrared sensors in order to recognize hot objects. Each hotplate of the stove is represented by a separate symbol. From the symbols *person* and *stove* the symbols *adult near stove*, *child near stove,* and *stove is hot* are generated, which are used by the application to recognize possible danger for a child.

6.2.1 Symbol *hotplate*

The symbol *hotplate* describes a hotplate of a stove. Such a symbol is created for each hotplate of a stove. Figure 50 shows the properties of the symbol *hotplate*.

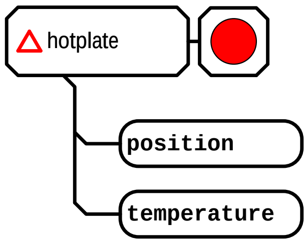


Figure 50: Symbol *hotplate*

The property *position* describes the position of the hotplate. It is assumed that the hotplate has the shape of a circle. The position therefore defines the centre of the hotplate. Further, it is assumed that the stove and thus the hotplate are at fixed points in the kitchen and the position of the hotplate is static. This means that no sensor is required for the position of the hotplate.

The property *temperature* represents the current temperature of the plate. As shown in Figure 49, it is derived from temperature sensors, which have to be mounted near the hotplate (e.g. at its bottom side) in order to deliver fast and accurate measurements. Since a hotplate exists even without the usage of a stove, a symbol *hotplate* is generated for each hotplate in the observed area at on the starting time of the system. Since the property *position* is assumed to be static, the only property of the symbol that is updated is the property *temperature*. The update is done when the temperature of the hotplate changes. Given that the hotplate does not disappear from its place, the symbol *hotplate* does not expire.

6.2.2 Symbol *hot object*

The symbol *hot object* describes a general hot object. The difference from the symbol *hotplate* (see Chapter 6.2.1) is that it is not static in position and time. The symbol *hot object* is dynamic and can appear in different places. Figure 51 shows the properties of the symbol *hot object*.

This symbol has been introduced in the system to detect hot objects near the stove, which have not been heated with the hotplates of the stove. In such a case, the temperature sensors mounted on the bottom side of the hotplate would not measure an increase in temperature, or would measure it only after some time has elapsed[19]. Thus, if someone heats water in a microwave and then puts the pot on the stove, the property *temperature* of the symbol *hot plate* would not be updated.

[19] Especially if the hot object that was put on the stove is not on a hotplate, the temperature sensors of the hotplate would not be influenced.

Values form sensors that are sensible to the infrared spectrum can be used to derive the symbol *hot object*. Since the position of the object has to be determined as well, an infrared camera in combination with a computer vision system could be used. Since such a camera can only detect differences in temperatures only objects which are colder or warmer than the surrounding environment can be detected.

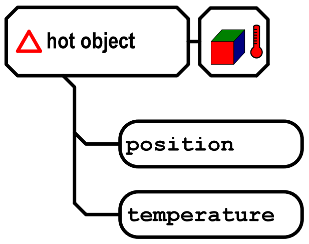


Figure 51: Symbol *hot object*

The property *position* describes the centre of the hot object. The temperature of the object is described with the property *temperature*. The symbol *hot object* is generated, when an object that is warmer than the surrounding environment is detected. If a pot is heated on the hotplate, the symbol *hot object* is generated later than the property *temperature* of the symbol *hotplate* is updated. But after a while, the property *temperature* of both symbols has a higher value than the surrounding environment. The symbol *hot object* is updated when the position of the object or the temperature of the object changes. If the object cannot be detected anymore, the symbol *hot object* expires.

6.2.3 Symbol *stove*

The symbol *stove* represents a stove. Similar to the symbol *hotplate* (see Chapter 6.2.1) it is static; the position of the stove does not change. Figure 52 shows the properties of the symbol *stove*.

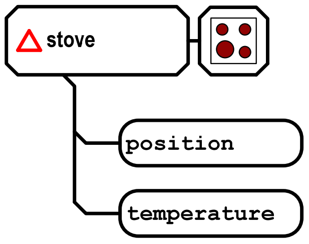


Figure 52: Symbol *stove*

The property *position* describes the position of the stove. It is assumed that the position is static and does not change. Therefore, no sensors are installed to measure the position of the stove. There are two possibilities to obtain the position of the stove. Firstly, it can be configured statically, as is the case with the property *position* of the symbol *hotplate* (see Chapter 6.2.1). The second possibility is to derive the position of the stove from the position of the hotplates.

The property *temperature* of the symbol *stove* describes the temperature of the stove. Since this symbol is used in the application *child in danger*, the property *temperature* describes the highest temperature of the stove. The property is derived from the symbols *hotplate* (highest temperature of all hotplates, which belong to the same stove) and the symbols *hot object* (highest temperature of all hot objects, which are close to the stove).

Since the symbol *stove* is static, the symbol is generated once for each stove in the observed area at start-up time of the system. The symbols never expire. The only property of the symbol *stove* that is updated is the property *temperature* as described above.

6.2.4 Symbol *stove is hot*

The symbol *stove is hot* represents a hot stove. The difference from the symbol *stove* (see Chapter 6.2.3) is that it is only generated when the temperature of the stove has reached a level that might be dangerous for a child. As Figure 53 shows, it has just the property *position*.

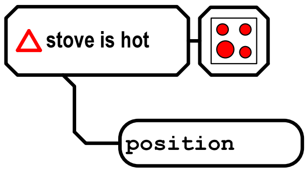


Figure 53: Symbol *stove is hot*

If a dangerous temperature level is reached, the symbol *stove is hot* is generated. The only property *position* has the same value as the symbol *stove*. Since the position of the stove is given as a static property, the symbol *stove is hot* is never updated. If the temperature of the stove is below the dangerous level, the symbol expires. This symbol has been introduced to represent the fact that the temperature of the stove has reached a dangerous level and therefore points out a possible danger for a child.

6.2.5 Symbol *adult near stove*

The symbol *adult near stove* represents an adult who is near the stove. It is derived from the symbol *person* (see Chapter 6.1.5) and the symbol *stove* (see Chapter 6.2.3). The properties of the symbol *adult near stove* are shown in Figure 54.

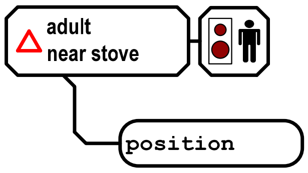


Figure 54: Symbol *adult near stove*

The symbol *adult near stove* has only the property *position*. It describes the current position of an adult. The property *type* of the symbol *person* is evaluated in order to determine, whether the person is an adult. If this is not defined in the symbol *person*, the symbol *adult near stove* should not be created since the symbol is used in a child safety application.

Whether the adult focus his or her attention on the child cannot be measured, at least not with simple sensors. However, one indication of attention on a child near the stove is that the property *orientation* of the symbol *person* points to the direction of the stove. Therefore, the symbol is generated when a person is near the stove and faces the stove. If the property *orientation* of the symbol *person* is not defined, it has to be assumed that the person does not face the stove and the symbol *adult near stove* should not be generated.

The property *position* is updated when the person moves and the property *position* of the symbol *person* changes. The symbol *adult near stove* expires, when the person leaves the area near the stove or does no longer face the stove. Figure 55 gives some examples, when the symbol *adult near stove* exists.

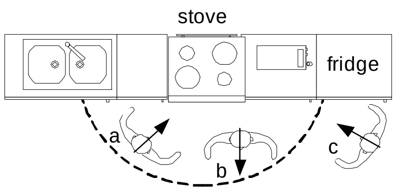


Figure 55: Examples for the symbol *adult near stove*

The person marked with *a* is the only person who would be represented with the symbol *adult near stove*. S/he is near the stove and faces it. The people marked with *b* and *c* are not represented with the symbol *adult near stove*. The person *b* does not face the stove, and person *c* is too far away from it.

6.2.6 Symbol *child near stove*

The symbol *child near stove* represents a child who is near a stove. Similar to the symbol *adult near stove*, it is derived from the symbols *person* (see Chapter 6.1.5) and *stove* (see Chapter 6.2.3). It has the properties shown in Figure 56.

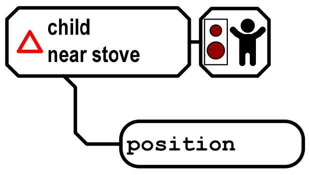


Figure 56: Symbol *child near stove*

The symbol *child near stove* is generated when a person is near the stove who is not an adult. To determine whether a person is an adult or not, the property *type* of the symbol *person* is used. As previously mentioned with the symbol *adult near stove,* the symbol *child near stove* is generated when the type of the person is not defined since the symbols are used in a child safety application.

The same safety concept applies to the orientation of the person. If orientation has not been defined for the person it has to be assumed that the child faces the stove, and therefore the symbol *child near stove* has to be generated if the child is within a certain area around it. If the child does not face the stove, the symbol is not generated.

The property *position* is updated when the child moves and the property *position* of the symbol *person* changes. The symbol *child near stove* expires when the child leaves the area around the stove or does no longer face it.

6.2.7 Application *child in danger*

The application *child in danger*[20] detects when a child is near a hot stove and no adult takes care of the child. It uses the symbols *stove is hot* (see Chapter 6.2.4), *child near stove* (see Chapter 6.2.5) and *adult near stove* (see Chapter 6.2.6). These symbols are only generated, when the conditions for a dangerous situation are given. Therefore, the rules for detecting dangerous situations are reduced to check whether the symbols exist or not. If more than one stove is within the system, it is not sufficient to check only the existence of the symbols. The property *position* of the symbols has to describe positions around a certain stove.

6.2.8 Scenario *danger child falling down stairs*

The application *danger child falling down stairs* detects when a child is near a staircase and no adult takes care of it. Figure 57 shows the symbol hierarchy of the application. It uses the symbols *person* (see Chapter 6.1.5) and the symbol *stairs* (see Chapter 6.2.9). From these symbols the symbols *adult near stairs* (see Chapter 6.2.10) and the symbol *child near stairs* (see Chapter 6.2.11) are derived.

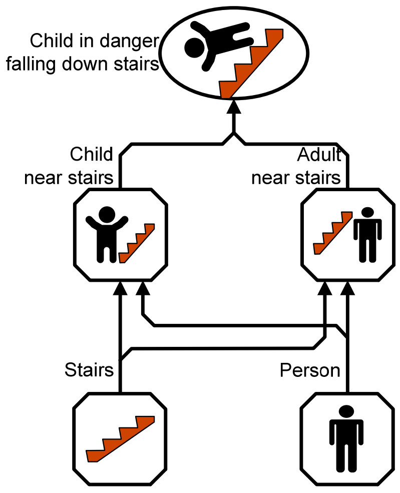


Figure 57: Scenario *danger child falls down stairs*

6.2.9 Symbol *stairs*

The symbol *stairs* represents stairs in a building. This symbol is not derived from sensor values, since stairs in a building are static. It makes no sense to detect them instead of configuring them statically. The properties of the symbol *stairs* are shown in Figure 58.

[20] The name child in danger for this application was chosen early in the project. At this stage of the project, more than one scenario is defined which represent that a child is danger. Nevertheless, the name was not change to avoid inconsistence with earlier works of the ARS project.

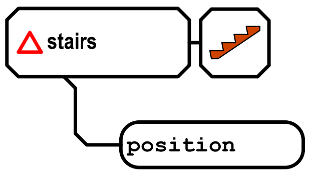


Figure 58: Symbol *stairs*

The symbol *stairs* has only the property *position*. It describes the position of the stairs in the building. The symbol *stairs* is generated at start-up of the system and does not expire. The symbol is never updated because it depends on no sensors or symbols.

6.2.10 Symbol *adult near stairs*

The symbol *adult near stairs* is similar to the symbol *adult near stove* (see Chapter 6.2.5). It defines the presence of an adult near the stairs and the orientation of the adult to the stairs. The properties of the symbol adult near stairs are shown in Figure 59.

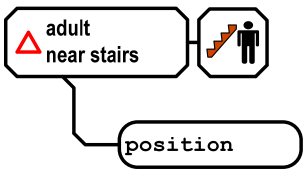


Figure 59: Symbol *adult near stairs*

The symbol *adult near stairs* has only the property *position*. It describes the current position of an adult. The symbol is generated when an adult is near the stairs and is facing them. Since the symbol *adult near stairs* is used in a safety application, the symbol should not be generated, if the type of the person cannot be determined as an adult or if the orientation of the adult is not known.

The property *position* is updated whenever the position of the adult changes but is still near the stairs. If the person's orientation does not point to the stairs, or the person is too far away from the stairs, the symbol *adult near stairs* expires.

6.2.11 Symbol *child near stairs*

The symbol *child near stairs* describes a child who is near stairs similar to the symbol *child near stove* (see Chapter 6.2.6). Figure 60 shows the properties of the symbol *child near stairs*.

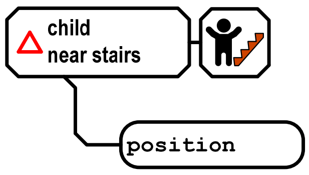


Figure 60: Symbol *child near stairs*

The symbol *child near stairs* has only the property *position*. It describes the position of the child who is near the stairs. The symbol is generated when a child is near the stairs. The property *position* is updated when the position of the child changes. When the child walks away from the stairs, the symbol *child near stairs* expires.

6.2.12 Application *child in danger falling down the stairs*

The application *child in danger falling down the stairs* is derived from the symbols *adult near stairs* (see Chapter 6.2.10) and *child near stairs* (see Chapter 6.2.11). If only symbol *child near stairs* exists, the child might be in danger and the application has to give alarm.

6.2.13 Scenario *unattended child in lift*

Usually it is forbidden for children to use lifts on their own. Nevertheless, the operation of a lift is simple enough for children to operate it. The scenario *unattended child in lift* recognizes the presence of an unattended child in the lift and raises alarm. Figure 61 shows the symbols and the dependencies between the symbols used for this application. It uses the symbols *person* (see Chapter 6.1.5), *lift* (see Chapter 6.2.14), *adult in lift* (see Chapter 6.2.15), and *child in lift* (see Chapter 6.2.16) that are derived from the symbols *person* and *lift*.

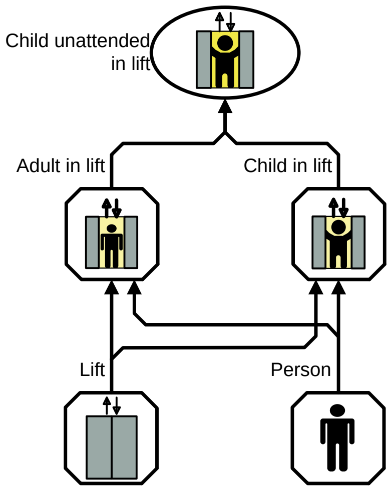


Figure 61: Scenario *unattended child in lift*

6.2.14 Symbol *lift*

The symbol *lift* describes a lift in a building. The properties of the symbol *lift* are shown in Figure 62. The symbol *lift* has only the property *position*. For the symbols depending on the symbol *lift* only the position in the x-y plane of the building is relevant but not the floor on which the cabin is at a given moment, therefore there is no need to define the z-coordinate of the property *position*[21]. Because the x-y coordinates of the lift do not change, the position is static, similar to the symbol *stairs* (see Chapter 6.2.9). The symbol is generated at start-up of the system. The property *position* is never updated and the symbol *lift* never expires.

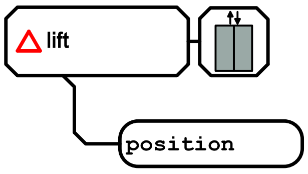


Figure 62: Symbol *lift*

6.2.15 Symbol *adult in lift*

The symbol *adult in lift* represents an adult, who is in a lift. It depends on the symbols *person* (see Chapter 6.1.5) and *lift* (see Chapter 6.2.14). Figure 63 shows the properties of the symbol *adult in lift*. The symbol *adult in lift* has only the property *position*. It describes the current position of the adult and has the same value as the property *position* of the symbol *lift*. It is generated when an adult is in the lift. As with all child safety applications, this symbol should only be generated when the property *type* of the symbol *person* defines the person as an adult. The symbol *adult in lift* expires when the adult leaves the lift.

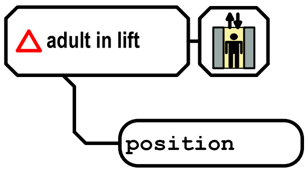


Figure 63: Symbol *adult in lift*

6.2.16 Symbol *child in lift*

Analogously to the symbol *adult in lift*, the symbol *child in lift* represents a child, who is in the lift. It is derived from the symbol *person* (see Chapter 6.1.5) and the symbol *lift* (see Chapter 6.2.14). The properties of the symbol *child in lift* are shown in Figure 64. The symbol *child in lift* has only the property *position*. It describes the position of a child in the lift. It is generated, when a child is in the lift. The property *position* of the symbol *child in lift* is identical with the property *position* of the symbol *person,* which represents the child. Whenever the position of the child in the lift changes, the property *position* of the symbol *child in lift* is updated. When the child leaves the lift, the symbol *child in lift* expires.

[21] If necessary for another application to have the z-coordinate of the lift, this information can be derived from the controller of the lift.

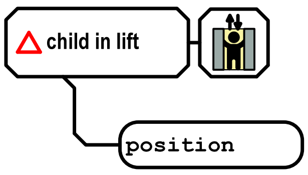


Figure 64: Symbol *child in lift*

6.2.17 Application *unattended child in lift*

The application *unattended child in lift* monitors whether a child is unattended in a lift. If it is, an alarm is raised and the lift will not react to the inputs from the control panel. The application uses the symbols *adult in lift* (see Chapter 6.2.15) and *child in lift* (see Chapter 6.2.16). If only the symbol *child in lift* exists, the application *child unattended in lift* is active. Whenever the symbol *adult in lift* exists, it is assumed that the adult monitors the child, and the application should stay inactive.

6.2.18 Scenario *person makes coffee*

The scenario *person makes coffee* is on the one hand an application for child safety. On the other hand, it is also used to detect human activities (see Chapter 6.4). Therefore, it is split into two scenarios, i.e. *child makes coffee* and *adult makes coffee*. The *child makes coffee* scenario detects an unattended child making coffee with the coffee maker, which is a dangerous situation since the child might burn itself with hot water. The scenario *adult makes coffee* is detected when an adult makes coffee. The symbols used in both scenarios are the same as shown in Figure 65.

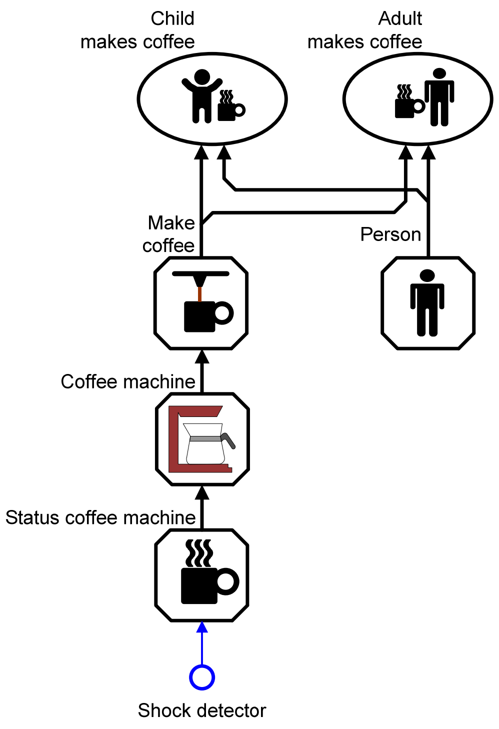


Figure 65: Scenario p*erson makes coffee*

From a shock detector the microsymbol *status coffee machine* and the usage of the coffee maker is derived and represented in the symbol *coffee machine*. In case the coffee machine is in use, the symbol *make coffee* is generated. Depending on the properties of the symbol *person*, which is generated with the people tracker application (see Chapter 6.1), either the scenario *child makes coffee* or the scenario *adult makes coffee* is generated. Similar to the application *person makes coffee,* the application that detects an unattended child near a hot stove (see Chapter 6.2.7) can be extended to detect human activities. In the following chapters the symbols used for this application are described.

6.2.19 Symbol *status coffee machine*

The symbol status coffee machine describes the status of a coffee machine. In this application, only the machine's usage is detected. Additionally, other parameters could be set to detect e.g. the filling level of water or coffee beans, or the connection of the coffee machine to the power supply. However, here only a shock detector is used for this application. Figure 66 shows the properties of the symbol *status coffee machine*.

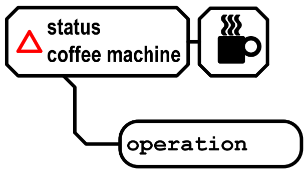


Figure 66: Symbol status coffee machine

The symbol *status coffee machine* has only the property *operation*. It indicates whether the coffee machine is making coffee or not. The symbol is generated at the start of the system and never expires.

6.2.20 Symbol *coffee machine*

The symbol *coffee machine* represents a coffee machine. It is derived from the symbol *status coffee machine*. The properties of the symbol *coffee machine* are shown in Figure 67.

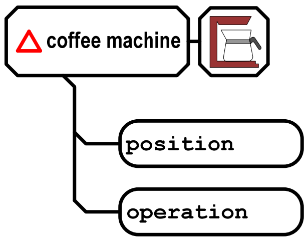


Figure 67: Symbol coffee machine

The symbol *coffee machine* has two properties, the property *position* and the property *operation*. The property *position* describes the place where the coffee machine is situated. The position of the coffee machine is defined statically, since it is assumed that the place of the coffee machine does not change. The property operation is derived from the symbol *status coffee machine* (see Chapter 6.2.19). The symbol is generated at start-up of the system and does not expire. Since the position is defined statically, only the property *operation* may change.

6.2.21 Symbol *make coffee*

The representation symbol *make coffee* describes the usage and the place of the coffee machine. It is derived from the symbol *coffee machine* (see Chapter 6.2.20). The properties of the symbol are depicted in Figure 68.

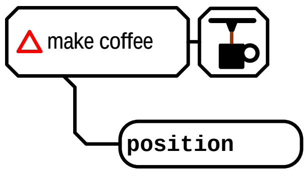


Figure 68: Symbol *make coffee*

The symbol *make coffee* has only the property *position*. It defines where a coffee is made. The symbol is generated when the property *operation* of the symbol *coffee machine* indicates the usage of the coffee machine. When the usage of the coffee machine has ended, the symbol *make coffee* expires.

6.2.22 Application *child/adult makes coffee*

As mentioned above the symbols and the dependencies of the symbols are equal for the application *child makes coffee* and *adult makes coffee*. As shown in Figure 69, both scenario symbols have the same properties. However, the scenario symbol *child makes coffee* is generated when an unattended child makes coffee, the scenario symbol adult makes coffee is generated when an adult makes coffee. Both symbols are derived from the symbol *make coffee* (see Chapter 6.2.21) and the symbol person (see Chapter 6.1.5).

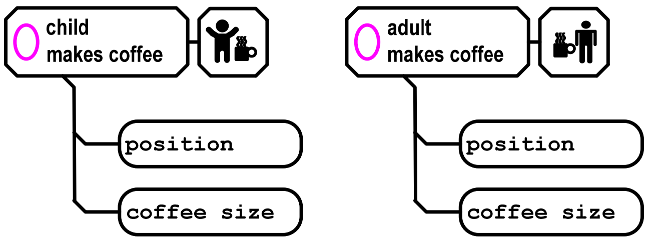


Figure 69: child/adult makes coffee

Both symbols have the property *position* and the property *coffee size*. The property position describes the position where the coffee is made. The property *coffee size* describes the size of the coffee. The duration of the symbol *make coffee* indicates how much coffee has been made. When the person (child and/or adult) leaves the room, the symbol *make coffee* expires.

6.3 Geriatrics

A routine job for the staff in homes for aged people is to monitor when the residents fall down. Here a scenario is described which is able to detect the fall of a person. Figure 70 shows the used symbols and their hierarchy. For a better overview, the connections between the sensors and the microsymbols have been left out. These dependencies are described in chapters that deal with the microsymbols shown in Figure 70. The sensors used for the scenario *person falls* are microphones, cameras, pressure sensors, or motion detectors. They generate the microsymbols *upright*, *fall*, *lie down*, *lie,* and *stand up*. These microsymbols are used together with the symbol *person* (see Chapter 6.1.5) to generate the symbols *person upright, person fell*, *person lying down*, *person lying,* or *person standup up*. From these symbols, the scenarios *harmful fall* and the *harmless fall* are generated. In the following chapters, these symbols and scenarios are discussed.

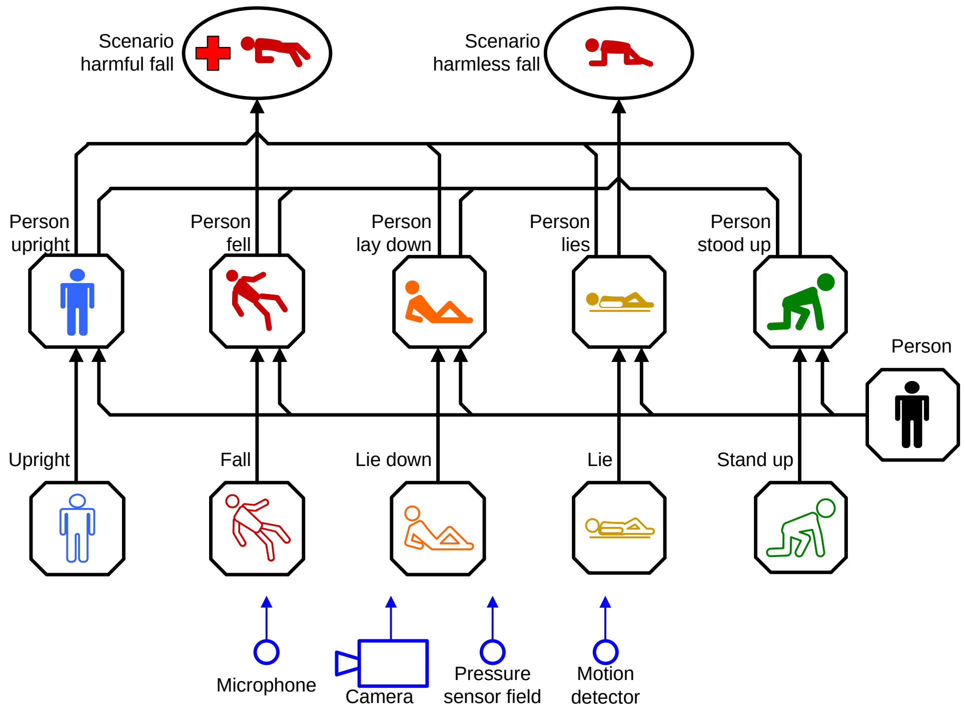


Figure 70: Geriatrics application

6.3.1 Symbol *upright*

The symbol *upright* describes a person who is upright. It does not matter whether the person stands still or moves. It can be derived from a camera, pressure sensor fields, microphones, orientation sensors, accelerometers, or motion detectors. Figure 71 shows the properties of the symbol *upright*. The symbol *upright* has only the property *position*. It describes the position of a person who stands or moves in the upright position. The symbol is generated whenever a person is detected who is upright. It is updated when the position changes. When the person leaves the room or is no longer upright, e.g. falls or lies down, the symbol *upright* expires.

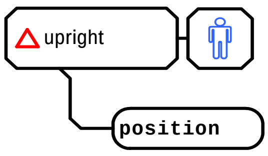


Figure 71: Symbol *upright*

6.3.2 Symbol *fall*

The symbol *fall* describes the fall of a person. It can be derived from a camera, pressure sensor fields, microphones, orientation sensors, accelerometers, or motion detectors. Figure 72 shows the properties of the symbol *fall*.

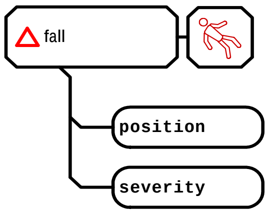


Figure 72: Symbol *fall*

The symbol *fall* has two properties, *position* and *severity*. The property *position* describes the position, where the person fell, the property *severity* the severity of the fall. To detect a fall a broad spectrum of detection systems are currently developed, which can all be used in principle to detect a fall, and therefore can be used to generate the symbol fall. In the following, these methods are briefly described.

Excursus fall detection

[Dou00] gives a classification of fall detection systems as shown in Figure 73. Fall detection systems are divided into primary detection systems, which detect the falls (and usually only falls) directly and secondary detection systems, which detect a fall indirectly (a fall results from a departure from normal patterns of activity). Both types of systems can be further divided into worn devices and built-in devices.

Worn devices of the primary fall detection techniques are shown on the left-hand side of Figure 73. They are devices with orientation sensors, impact sensors, or a combination of sensors, the so-called smart multi-stage fall detectors. The orientation sensors measure the orientation of the person[22]. If the person falls, the orientation changes within a short amount of time and usually by 90 degrees from an upright to a horizontal position. Impact sensors are accelerometers, which detect the shock of the impact of the body at the end of the fall. The retardation which is detected during the impact is higher than retardation measured during usual movement. Smart multi-stage fall detectors use both types of sensors, orientation and acceleration sensors combined with an algorithm. The algorithm decides whether the values from both sensors indicate a dangerous fall or not.

[22] More precisely, the orientation of the fall detector is measured. But it is assumed that people who have a high risk of falling wear the device.

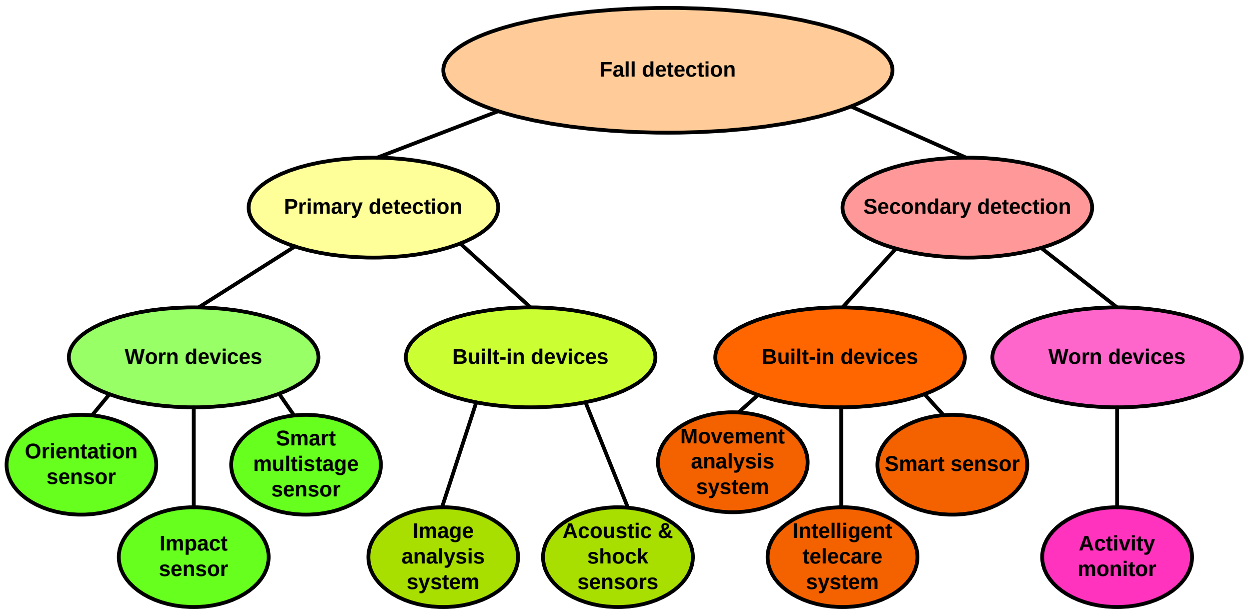


Figure 73: Classification of fall detection systems [Dou00]

The built-in devices of the primary fall detection systems are image analysis systems and acoustic and shock detectors. Image analyzing systems use either video cameras or infrared cameras as data source to detect the fall of a person as demonstrated in [CHA04]. Acoustic sensors are used to detect the impact of a fall, whereas shock detectors are used to detect the vibrations in the (wooden) floor [ALW03].

As with the primary fall detection techniques, the secondary fall detection techniques are divided into worn devices and built-in devices (see the right hand-side of Figure 73). Worn devices (e.g. on chest, wrist, and waist) measure physiological parameters, which are used as an indicator for the activity level. The built-in devices derive the activity level from the movement of the people who are measured e.g. with cameras as shown in [CHA04]. For both worn and built-in devices, the level of activity is then used to detect unusual situations. These secondary fall detection techniques are therefore not very reliable and fast because it usually takes at least some minutes until an unusual situation can be detected. Further, it is hard to identify the reason and the consequences of an unusual situation, and thus it is difficult to determine whether a fall occurred.

Fall detection in the ARS system

To generate the symbol *fall*, it is proposed to use a combination of the fall detection techniques mentioned above and pressure sensor fields. The position of the fall of a person can be determined with camera and vision systems, pressure sensor fields, and microphones. Additionally, worn devices can be used to increase the reliability of the overall system. When a person falls in the sensitive area of a motion detector, the detector has to detect the motion.

If pressure sensor fields are used, a fall is detected when the number of activated pressure sensors increases in the vicinity of the last known position of a person as shown in Figure 74. On the left side a person is shown who triggers only one pressure sensor. The right side shows the person lying on the floor, triggering five pressure sensors. If the sensors are also able to measure weight, instead of only activation, the sum of the measured weight of all activated pressure sensors has to be equal to the weight of the person.

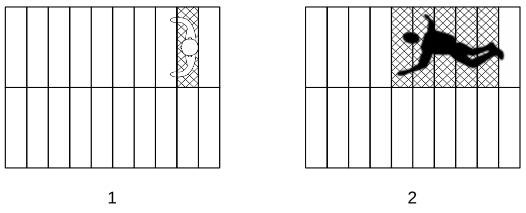


Figure 74: Detecting a fall with pressure sensors

The property *severity* of the symbol *fall* can be derived from accelerator sensors in the worn devices, the speed of falling with vision systems, and the shock and noise measured on impact of the body. Pressure sensor fields, which measure the dynamic behavior of the impact, can also be used to determine the severity of the fall, similar to the methods used in the worn devices.

The symbol *fall* is generated, when the fall of a person is detected. Since the property *position* and the property *severity* are determined when the fall occurs, the symbol is not updated. The symbol *fall* expires immediately after it has been generated.

6.3.3 Symbol *lie down*

The symbol *lie down* describes that a person is lying down. The difference from the symbol *fall* (see Chapter 6.3.2) is that the person wants to lie down and does not tip over. Since the ARS system is not capable to detecting the intentions of a person, indirect methods have to be used to decide whether a person wants to lie down or has tripped over. Such indirect methods are the position where the person assumes a horizontal position and the manner of how s/he gets into this position. Dedicated positions for lying down are e.g. a bed or sofa. In the ARS system, it is assumed that the floor is not a dedicated place for lying down. Further, it is assumed that a person who wants to lie down, will first sit down or sit on the knees before lying down. In principle, the same sensors as for the symbol *fall* are used. Figure 75 shows the properties of the symbol *lie down*.

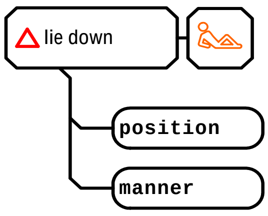


Figure 75: Symbol *lie down*

The symbol *lie down* has the property *position* and the property *manner*, which describe the position where a person lies down and the manner how the person lies down. The same methods as described for the symbol *fall* can be used to determine the properties of the symbol *lie down*, except the pressure sensor field since it is assumed that a person does not want to lie down on the floor.

The symbol *lie down* is generated when the fall of a person is detected. Since the property *position* and the property *manner* are determined when the fall occurs, the symbol is not updated. The symbol *lie down* expires immediately after it has been generated.

6.3.4 Symbol *lie*

The symbol *lie* describes a person who is lying. It does not matter how the person came into the horizontal position. Rather, the symbol is generated after the symbol *fall* (see Chapter 6.3.2) or the symbol *lie down* (see Chapter 6.3.3) has occurred. Figure 76 shows the properties of the symbol *lie*.

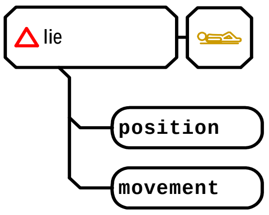


Figure 76: Symbol *lie*

The symbol *lie* has the property *position* and the property *movement*. The property *position* describes the position, where the person is lying. It can be detected with cameras, accelerometer, pressure sensor field, and orientation sensors. The property *movement* describes how the person is lying, whether s/he is calm or not. For detecting how the person moves while lying, cameras and accelerometers can be used.

The symbol *lie* is generated after a fall has been detected or a person lay down. It is also generated if neither a fall nor a person lying down were detected but a lying person is detected. The property *movement* is updated when the person changes his or her behavior while lying. Since the person might move when lying, the property *position* is updated. The symbol *lie* expires when the person stands up.

6.3.5 Symbol *stand up*

The symbol *stand up* describes a person standing up. Figure 77 shows the properties of the symbol *stand up*. The symbol *stand up* has the property *position* and the property *manner*. The property *position* describes the position where the person stands up. It can be detected with cameras, accelerometer, pressure sensor field, and orientation sensors. The property *manner* describes how fast the person rises. Cameras and accelerometers can be used for this.

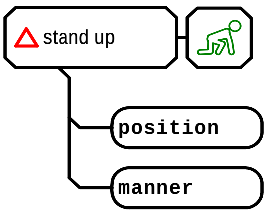


Figure 77: Symbol *stand up*

The symbol *stand up* is generated when the symbol *lie* (see Chapter 6.3.4) expires. Additionally, the symbol is generated when the features extracted from the sensors fit to the pattern of a person standing up. Since the properties of the symbol do not change, the symbol is never updated. The symbol *stand up* expires immediately after it has been generated.

6.3.6 Symbol *person upright*

The symbol *person upright* describes a person who is in the upright position. It is derived from the symbols *upright* (see Chapter 6.3.1) and *person* (see Chapter 6.1.5). Figure 78 shows the properties of the symbol *person upright*.

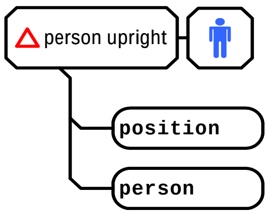


Figure 78: Symbol *person upright*

The symbol *person upright* has two properties. The property *position* describes the position of the person who is upright and is composed of the position of the property *position* of the symbol *person* and the property *position* of the symbol *upright*. The property *person* defines a person who is upright and is a link to the symbol *person*, which is associated with the upright position.

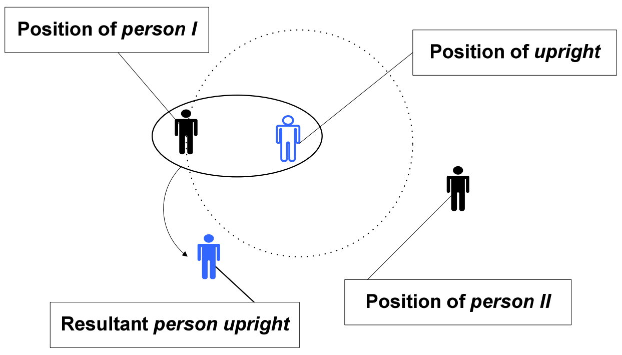


Figure 79: Spatial relationship between symbols *person* and *upright*

The association between the symbols *person* and *upright* can be conducted spatially and/or logically. Figure 79 shows the spatial relationship, while Figure 80 shows the logical relationships, which lead to the generation of the symbol *person upright*.

The spatial relationship between a symbol *person* and a symbol *upright* is derived from the shortest distance between the positions of both symbols (see Figure 79). The position of the symbol *person I* (black symbol on the left side) is closer to the position of the symbol *upright* (blue-framed symbol in the centre) than the symbol *person II* (black person on the right). Therefore, the symbol *upright* is associated with the symbol *person I* and the symbol *person upright* is generated.

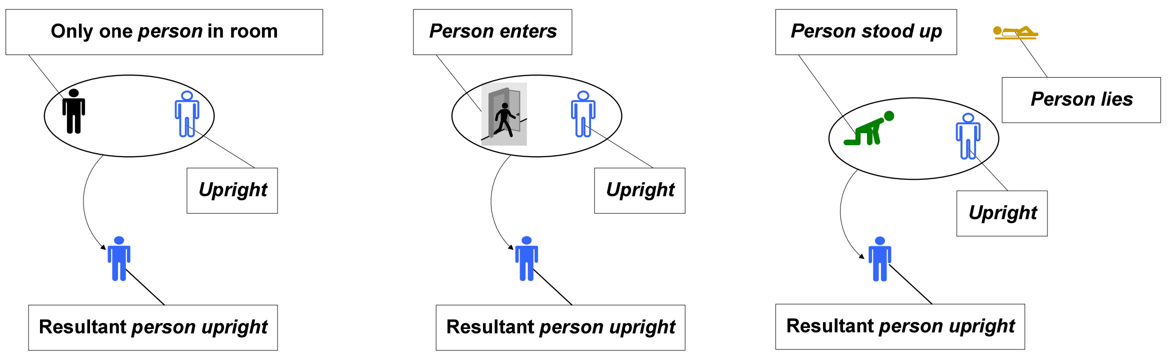

Only one *person* in room
Upright
Resultant *person upright*
Person enters
Upright
Resultant *person upright*
Person stood up
Person lies
Upright
Resultant *person upright*

Figure 80: Logical relationship for generating symbol *person upright*

Figure 80 shows three logical relationships which lead to the generation of the symbol *person upright*. On the left, only one person is in the room. Therefore, the symbol *upright* is associated with the only existing person in the room. In the middle, a person enters the room, represented by the symbol *person enters* (see Chapter 6.1.2). The symbol *upright*, which is generated almost at the same time and the same place (near the door), is therefore associated with the symbol *person enters* and thus with the person, who is associated with the symbol *person enters*.

On the right, two persons are in the room. The person in the upper right-hand corner is lying, represented by the symbol *person lies* (see Chapter 6.3.9). The other person in the middle on the left-hand side is standing up, represented with the symbol *person stood up* (see Chapter 6.3.10). The symbol *upright* is therefore associated with the symbol *person stood up*, because s/he can be upright if s/he first stood up, and not with the symbol *person lies*.

The symbol *person upright* is generated when the conditions for its generation are fulfilled, as described above. Since the person can move when s/he is upright, the property *position* of the symbol *person upright* is updated. The symbol expires when the person who is associated with the symbol leaves the room or is no longer upright.

6.3.7 Symbol *person fell*

The symbol *person fell* describes a person who fell down. It is derived from the symbols *fall* (see Chapter 6.3.2) and *person* (see Chapter 6.1.5). Figure 81 shows the properties of the symbol *person fell*.

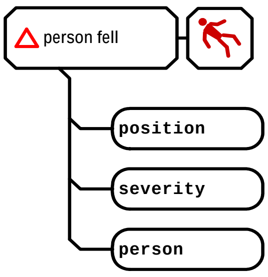


Figure 81: Symbol *person fell*

The property *position* of the symbol *person fell* describes the position where the person fell. Since the symbol *person fell* is a combination of the symbols *person* and *fall*, the value of the property *position* of the symbol *person fell* is taken from the symbols *person* and *fall*. If the value of the property *position* of the symbols *person* and *fall* are not identical, the value of the property *position* is taken from the symbol where the value has a higher reliability.

The severity of the fall is represented by the property *severity*. Since no additional information is available, the property *severity* of the symbol *person fell* is the same as the property *severity* of the symbol *fall*.

The property *person* defines the person who fell down. The value of the property *person* is a link to the symbol *person*, who is associated with the fall. For the association between the symbols *fall* and *person,* a spatial relationship, logical relationship (see Figure 82) and/or a statistical relationship (see Figure 83) can be used. Analogously to Figure 79 with the symbol *upright,* the symbols *person* and *fall* are associated with each other when the distance between the positions of the symbols is shorter than the distance of another pair of symbols.

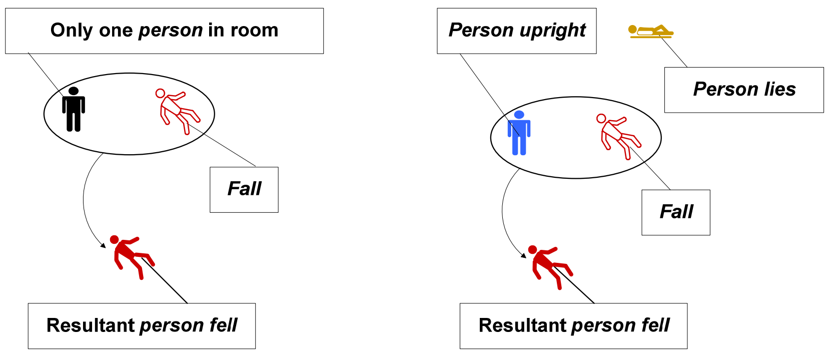


Figure 82: Logical relations between symbol *person* and symbol *fall*

Two examples of logical relations between the symbols *person* and *fall* are demonstrated in Figure 82. On the left-hand side only one symbol *person* is in the room where the fall is detected. Therefore, the symbol *fall* is associated with this symbol *person* and the symbol *person fell* is generated. On the right-hand side, two people are in the room, illustrated as the symbol *person upright* (blue symbol on the left) and the symbol *person lies* (yellow symbol on the upper right). Since it is assumed that the person who is lying cannot fall, the symbol *fall* is associated with the symbol *person upright* and the symbol *person fell* is generated.

Assuming that more than one person is in the room, the association between the symbols *person* and *fall* can be conducted by using statistics of a person's risk of falling down. On the left is *person I,* who has a high risk of falling, as indicated by the bar on the right. The second person in the room, represented with the symbol *person II*, has a low risk of falling as the bar shows. Therefore, the symbol *fall* is associated with the symbol *person I* because the person has a higher risk of falling.

The symbol *person fell* is generated when a symbol *fall* is available within the system. If an association of the symbol *fall* with a symbol *person* is not possible because no symbol *person* is available in the room, the symbol *person fell* is generated anyway, but the property *person* is left blank. The property *position* and the property *severity* are taken from the symbol *fall*. Since it is assumed that the properties of the symbol *person fell* do not change, the symbol *person fell* is never updated. The symbol expires immediately after it has been generated.

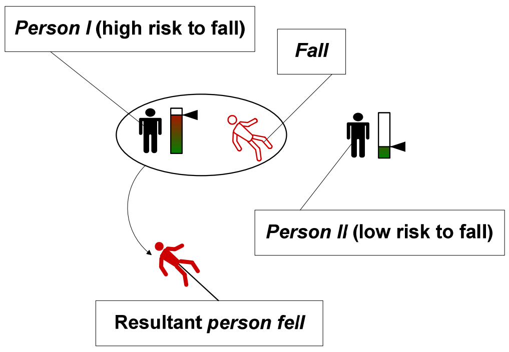

Person I (high risk to fall)
Fall
Person II (low risk to fall)
Resultant *person fell*

Figure 83: Statistical relations between symbols *person* and *fall*

6.3.8 Symbol *person lay down*

The symbol *person lay down* describes a person, who lay down intentionally, meaning he did not fall. It is derived from the symbols *person* (see Chapter 6.1.5) and *lie down* (see Chapter 6.3.3). The properties of the symbol *person lay down* are shown in Figure 84.

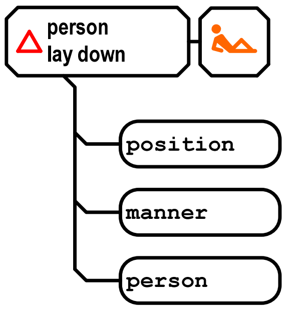


Figure 84: Symbol *person lay down*

The symbol *person lay down* has the properties *position, manner* and *person*. The property *position* describes the position where the person lay down. The property *manner* describes how the person lay down. The person who lay down is referred to with the property *person*. The relationship between the symbols *person* and *lie down* can be spatial, logical, and/or statistical. The spatial relationship is similar to the symbol *upright* illustrated in Figure 79. The symbol *person*, which is next to the symbol *lie down*, is combined with the symbol *person lay down*.

Corresponding to the logical relationships to generate the symbol *person fell* the symbol *person* is associated with the symbol *lie down*, when only one person is in the room as shown on the left of Figure 85. In the second example, the symbol *lie down* is associated with the symbol *person upright* because the person represented by the symbol *person lies* is already lying and is therefore is not able to lie down.

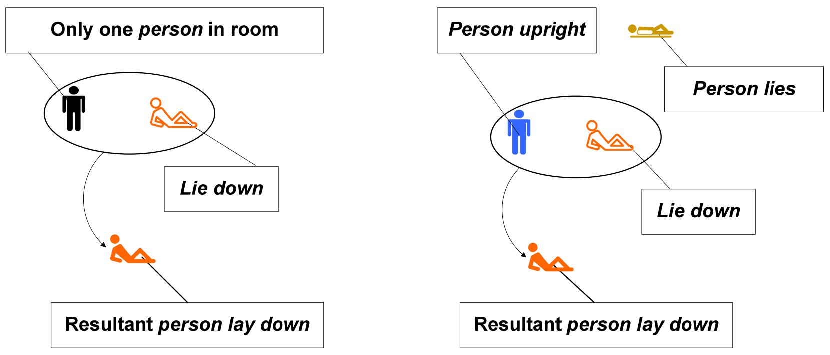

Only one *person* in room
Lie down
Resultant *person lay down*
Person upright
Person lies
Lie down
Resultant *person lay down*

Figure 85: Logical relationship leads to generate the symbol *person lay down*

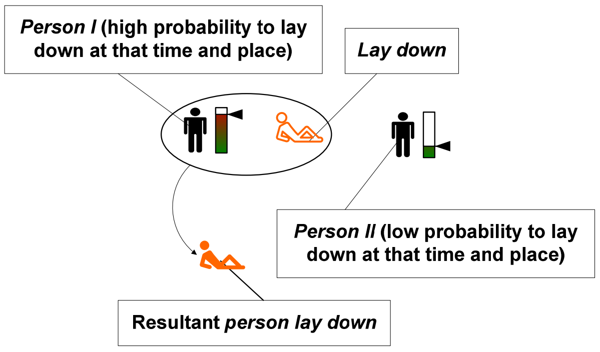

Person I (high probability to lay down at that time and place)
Lay down
Person II (low probability to lay down at that time and place)
Resultant *person lay down*

Figure 86: Statistical relationship to generate the symbol *person lay down*

A statistical method to associate the symbol *lie down* with a person is to use knowledge about a person's habits. If a person goes to bed at the same time (almost) every day and sleeps in the same bed, the probability is high that s/he will do so also in future.[23] The person on the left part in Figure 86 has a high probability to lying down at the time and place when the symbol *lie down* is active. The other person (symbol *person II*) usually lies down at another time and/or at another place. Since the probability is higher that person I is the person who is lying down, the symbol *lie down* is associated with the symbol *person I* and the symbol *person lay down* with a reference to symbol *person I* is generated.

The symbol *person lay down* is generated when a symbol *lie down* is available within the system. If an association of the symbol *lie down* with a symbol *person* is not possible because no symbol *person* is available in the room, the symbol *person lay down* is generated anyway but the property *person* is left blank. The property *position* and the property *severity* are taken from the symbol *fall*. Since it is assumed that the properties of the symbol *person lay down* do not change, the symbol *person lay down* is never updated. The symbol expires immediately after it has been generated.

[23] The probability may change depending on the situation (e.g. day of the week).

6.3.9 Symbol *person lies*

The symbol *person lies* represents a person who is lying. It is derived from the symbols *person* (see Chapter 6.1.5) and *lie* (see Chapter 6.3.4). The properties of the symbol *person lies* are shown in Figure 87.

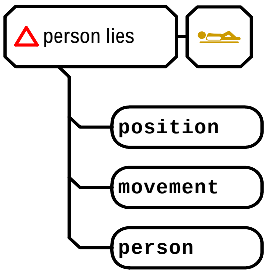


Figure 87: Symbol *person lies*

The symbol *person lies* has the properties *position*, *movement* and *person*. The property *position* describes the position of the lying person. As with the symbol *lie* the property *movement* describes whether the person lies quietly or is restless. The property *person* is a link to the symbol *person*, which is associated with the symbol *person lies*.

The relationship between a symbol *lie* and a symbol *person* can be spatial, logical and/or statistical. With the spatial relationship, the symbols *person* and *lie* with the shortest distance between them are associated with each other.

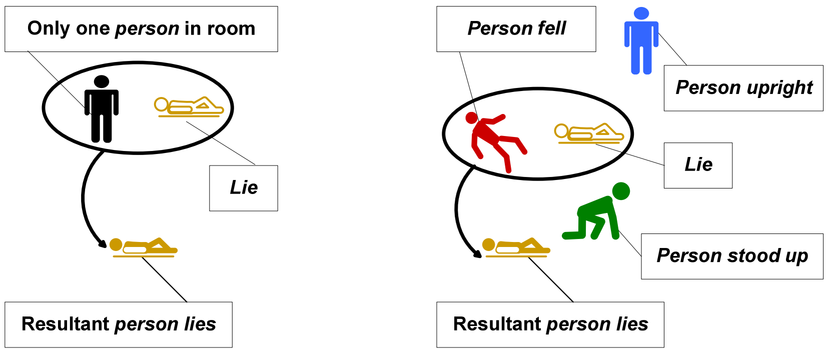


Figure 88: Logical relationship leading to generate the symbol *person lies*

Figure 88 shows the logical relationships between the symbol *lie* and other symbols, which lead to the generation of the symbol *person lies*. If only one person is in the room, the symbol *lie* is associated with the symbol *person* which represents this person. In the second example the symbol *lie* is associated with the symbol *person fell*, because it is assumed that after a fall the person is lying. The symbol *person upright* and the symbol *person stood up* are not associated with the symbol *lie* because a person who is upright or standing up cannot lie without falling or lying down in-between.

Similar to the symbol *person lay down* the habit of a person can be used for a statistical association between the symbols *stand up* and *person*. As is the case with going to bed, people often stand up at the same time[24].

The symbol *person lies* is generated when a symbol *lie* is available within the system. If an association of the symbol *lie* with a symbol *person* is not possible because no symbol *person* is available in the room, the symbol *person lies* is generated anyway but the property *person* is left blank. The property *position* and the property *severity* are taken from the symbol *lie*. Since it is assumed that the properties of the symbol *person lies* do not change, the symbol *person lies* is never updated. The symbol expires immediately after it has been generated.

6.3.10 Symbol *person stood up*

The *symbol person stood up* describes a person who is standing up. It is derived from the symbols *person* (see Chapter 6.1.5) and *stand up* (see Chapter 6.3.5). Figure 89 shows the property of the symbol *person stood up*.

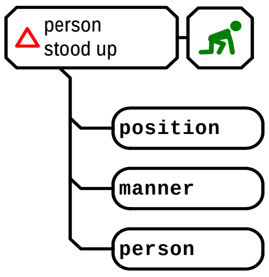


Figure 89: Symbol *person stood up*

The symbol *person stood up* has the properties *position*, *manner* and *person*. The property *position* defines the position where the person is standing up. The property *manner* describes the way, how the person is standing up. The property *person* is a link to the symbol *person,* which is associated with the symbol *person stood up*. Again the association between a person and the symbol *stand up* can be done spatially, logically and/or statistically.

In case more than one symbol *person* is in a room, a symbol *stand up* is associated with the symbol *person* who is closest to it (spatial relationship). Statistical information about the time and place when a certain person stood up can be used to associate the symbols. Examples of situations that lead to the generation of the symbol *person stood up* are given in Figure 90.

On the left part is a situation where only one person is in the room. Therefore, the symbol *stand up* is associated with the symbol *person* and the symbol *person stood up* is generated. On the right-hand side the symbol *stand up* is associated with the symbol *person lies* because the other person in the room, represented with the symbol *person upright,* is upright and therefore cannot stand up.

[24] The probability of a person standing up at the same place as he fell down or lay down is assumed to be equal to 1.

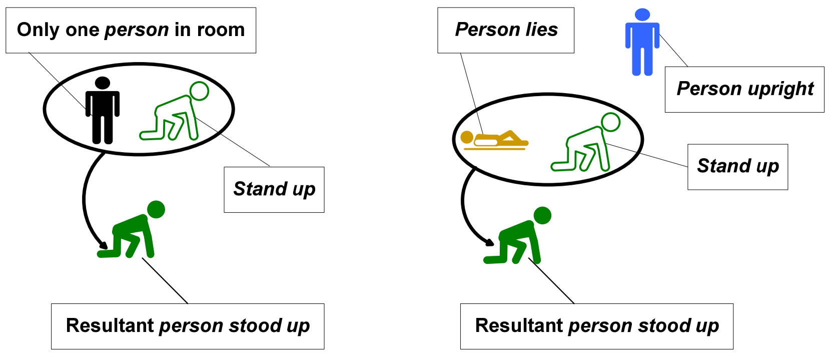

Only one *person* in room
Stand up
Resultant *person stood up*
Person lies
Person upright
Stand up
Resultant *person stood up*

Figure 90: Logical relationship leading to generate the symbol *person stood up*

The symbol *person stood up* is generated whenever a situation occurs in the system as described above. The property *position* is derived from the property *position* of the symbols, which are associated. The property *manner* receives the same value as the property *manner* of the symbol *stand up*. The value of the property *person* is a link to the symbol *person,* which is associated with the symbol. The symbol *person stood up* is not updated, since the properties do not change. The symbol expires immediately after it has been generated.

6.3.11 Scenario symbol *harmless fall*

The scenario *harmless fall* describes a scenario in which a person falls and stands up immediately after the fall without requiring any help. It consists of symbols *person fell* (see Chapter 6.3.7), and *person stood up* (see Chapter 6.3.10). The properties of the scenario symbol *harmless fall* are shown in Figure 91.

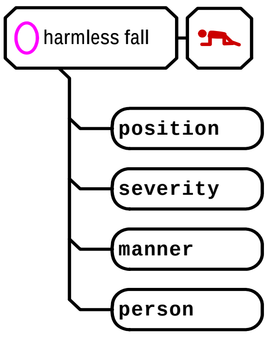


Figure 91: Scenario symbol *harmless fall*

The scenario symbol *harmless fall* has the properties *position*, *severity*, *manner*, and *person*. The property *position* describes the position where the fall happens. The value of the property is taken from the value of the property *position* of the symbol *person fell*. The property *severity* describes the severity of the fall. The value of this property is also derived from the symbol *person fell*, but from the property *severity*. The property *manner* describes the manner how the person stands up after the fall. The value for the property is taken from the property *manner* of the symbol *person stood up*. The property *person* is a link to the symbol *person*, which represents the person who fell.

The association between the symbol *person fell* and the symbol *person stood up* can be done logically and spatially. A harmless fall can only be detected when both the symbol *person fell* and the symbol *person stood up* are detected. The value of the property *person* of both symbols has to be equal. If they are not equal, the symbol *harmless fall* does not have to be generated. Assuming that the person stands up where s/he fell down, the property *position* of both the symbol *person fell* and the symbol *person stood up* has to be the same.

If the association between the symbol *person fell* and the symbol *person stood up* is carried out as described above, the following conditions have to be fulfilled in order to generate the scenario *harmless fall*. These conditions are also illustrated in Figure 92. First the person is upright, symbolized with the symbol *person upright*. At the time t_{fall} the person falls down which is illustrated with the symbol *person fell*. The property *severity* of the symbol *person fell* must not be higher than a certain value (medium). This value should represent the severity of the fall. It can be assumed that the person is not hurt. If the property severity is higher than the threshold level the scenario symbol *harmless fall* must not be generated. If the scenario harmless fall should be recognized, the person has to stand up at the time $t_{stood\ up}$, represented with the symbol *person stood up*.

The time between *person fell* (t_{fall}) and *stood up* ($t_{stood\ up}$) should not be more than a certain amount ($t_{max\ lie}$). This period of time represents the amount of time a healthy person usually needs to stand up after a fall. The manner how the person stands up, represented by the property *manner* of the symbol *person stood up*, represents a value which indicates that the person standing up is not injured (medium).

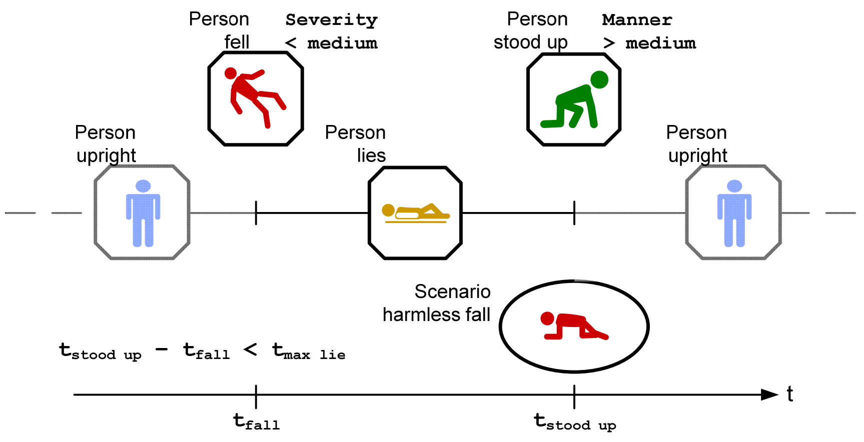


Figure 92: Conditions for generation of scenario *harmless fall*

After the person stood up the symbol *person upright* is generated. If all the conditions mentioned above are fulfilled, the scenario *harmless fall* is recognized. The properties of the scenario symbol are defined at the time of generation and do not change after that. The scenario symbol expires immediately after it has been generated.

6.3.12 Scenario symbol *harmful fall*

The scenario *harmful fall* describes a scenario in which a person falls and cannot stand up without any help or stands up slowly. It consists of the symbols *person fell* (see Chapter 6.3.7), *person lies* (see

Chapter 6.3.9) and *person stood up* (see Chapter 6.3.10). The properties of the scenario symbol *harmfull fall* are shown in Figure 93.

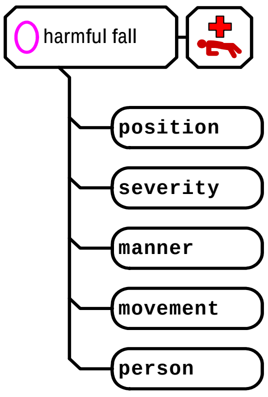


Figure 93: Scenario symbol *harmful fall*

In addition to the properties *position*, *severity*, *manner* and *person* as described with the scenario symbol *harmless fall* (see Chapter 6.3.11) the scenario symbol *harmful fall* has the property *movement*. This property describes how the person lies on the floor. The value is taken from the property *movement* of the symbol *person lies*.

The association between the symbols *person fell*, *person lies*, and *person stood up* is logical and spatial. Different from the scenario symbol *harmless fall* the scenario symbol *harmful fall* is also generated if the association of the symbols is not possible. The logical association uses the property *person* of the symbols. If the properties link to the same symbol *person*, the symbols should be associated with each other. The spatial association uses the property *position* of the involved symbols. If the property *position* of the symbols describes positions within a certain area, the symbols can be associated with the scenario symbol *harmful fall*.

The time when the scenario symbol *harmful fall* is generated depends on the conditions described below and illustrated in Figure 94 and Figure 95. On the left, the property *severity* of the symbol *person fell* has a value which is higher than the value medium. The value medium describes the severity of the fall of a person, which is so high that it has to be assumed that the person might be injured. Nevertheless, if the symbol *person fell* can be associated with the symbol *person lies*, the scenario symbol *harmful fall* is generated.

On the right-hand side of Figure 94, a scenario is illustrated where a person fell down at the time t_{fall}. The property severity of the symbol *person fell* is low enough to let us assumed that the person was not injured, so the scenario symbol *harmful fall* is not generated. But the period of time during which the person lay on the floor is longer than the maximum time ($t_{max\ lie}$). Therefore, at the time $t_{harmful\ fall}$ the scenario symbol *harmful fall* is generated.

Figure 95 shows a scenario similar to the scenario harmless fall. The difference from Figure 92 is that the person stands up too slowly, which is represented by the property *manner* of the symbol *person stood up*. Therefore, the scenario symbol *harmful fall* is generated at the time $t_{stood\ up}$ although the person stood up.

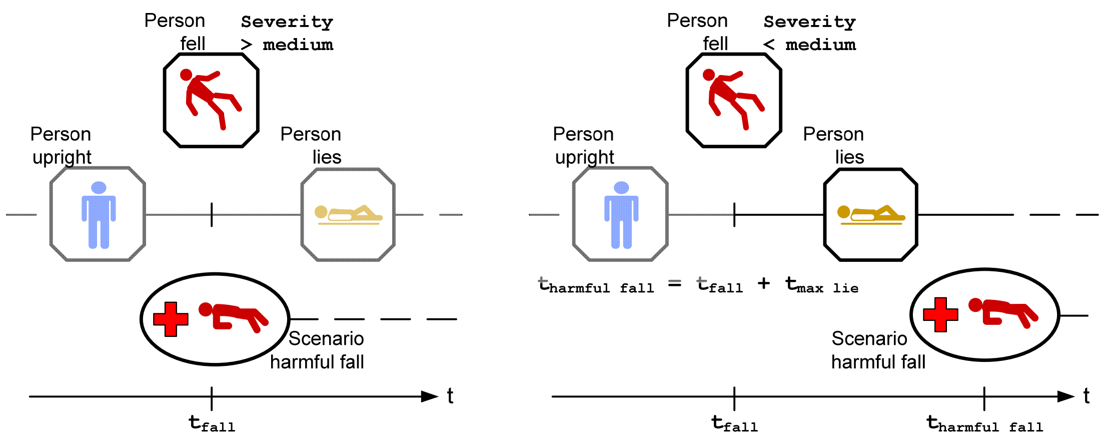


Figure 94: Conditions I for generation of scenario symbol *harmful fall*

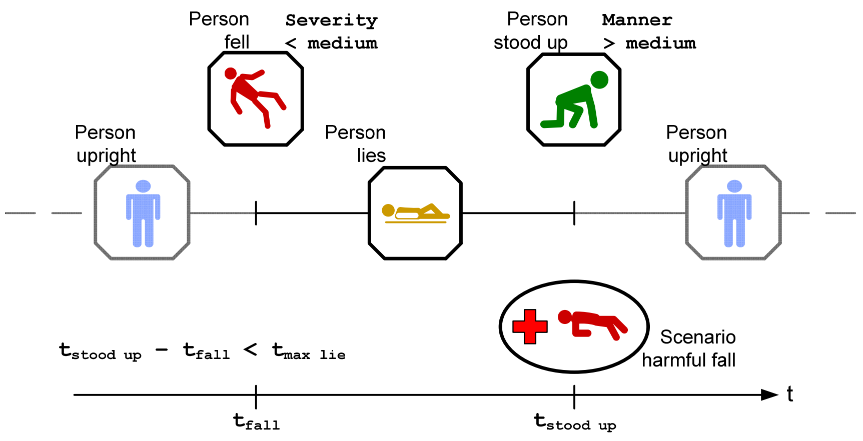


Figure 95: Conditions II for generation of scenario symbol *harmful fall*

Whenever the scenario symbol *harmful fall* is generated as described above, the property *severity* and the property *position* are taken from the symbol *fall*. The property *manner* is taken from the symbol *person stood up*. The property *movement* is taken from the symbol *person lies*. The value of the property *person* is a link to the symbol *person,* which is associated with the symbol.

6.4 Human Activities Application and Joint Symbol Tree

The applications described above form together a symbol tree (see Figure 96),which shows the symbols and the relationships between them as used in the applications described in the previous chapters. Additionally, the scenario *meeting* and the scenario *person and object* are introduced which can be seen as human activities applications. They have not been described separately because they are quite simple scenarios.

The scenario *meeting* is described in detail in Chapter 4.5. In it, it is detected when two or more people sit around a table. For this application, the symbols *person*, generated by the people tracking application (see Chapter 6.1.5) and the *table* are used. The symbol *table* is defined statically and describes a table and its position. If more than one person is detected at the position of the table, the scenario symbol *meeting* is generated. The property *position* of the symbol is identical with the position of the table. The symbol *meeting* expires when less than two people are detected.

The scenario *person and object* is described in Chapter 4.3. It recognizes when someone puts down an object in a room and then leaves the room. The scenario symbol *person and object* is generated, when on the same place a symbol *person* and a symbol *object* are generated and the object does not change its position, but the person moves away from the place. The same place means that the property *position* of the symbol *person* and the symbol *object* are located nearby. The scenario symbol expires when the object is taken away. In principle this could also be seen as scenario on its own.

The gray field on the left part of Figure 96 shows the geriatric care applications. Two scenarios, the scenario *harmful fall* and the scenario *harmless fall* are detected as described in Chapter 4.7. A harmful fall is detected when a person falls down and does not stand up again. A harmless fall is recognized when a person falls down but stands up quickly.

The green, field in the upper middle part of Figure 96 shows examples of the human activities applications. These scenarios are *adult makes coffee* (see Chapter 6.2.18) and the scenarios described above i.e. scenarios *meeting*, and *person and object*. However, all other symbols that represent a person and his or her action can be seen as a human activity application.

The blue field in the lower middle part of Figure 96 illustrates the people tracking application. This application is the basis for all scenarios where people are observed, it recognizes the position of persons. The application is described in detail in Chapter 4.2, and the used symbols and the dependencies of the symbol are listed in Chapter 6.1.

In the red field on the right part of Figure 96 the child safety applications are depicted. These are the scenarios *child makes coffee* (see Chapter 6.2.18), *child unattended in lift* (see Chapter 6.2.17) and *child in danger* (see Chapter 6.2.7). Due to reduced complexity, the scenario *child in danger falling down the stairs* is not shown in Figure 96. It is constructed analogously to the scenario *child unattended in lift* as described in Chapter 6.2.8. All these scenarios represent a situation where a child is near a dangerous object or device. Being near a hot stove or a coffee maker, the child is in danger of getting burned. In the near of stairs, the child can be hurt when falling downstairs. In a lift, children are usually not in danger of getting injured, but it is forbidden that unattended children operate a lift.

The bottom of Figure 96 shows the sensor on which the symbols are based. The microphones, cameras, pressure sensor field, and motion detectors are mainly used in the people tracking application and the geriatric care application. The shock detector is used in the *coffee making* scenario, and the infrared sensor and the temperature sensor in the *child in danger* scenario. However, as is the case with people, the assignment of a sensor to an application is invariant. Good examples of this variance are the human activities applications. As described above and illustrated in Figure 96, a range of symbols represent the action of a person. And these symbols are practically all derived from sensors used in the system.

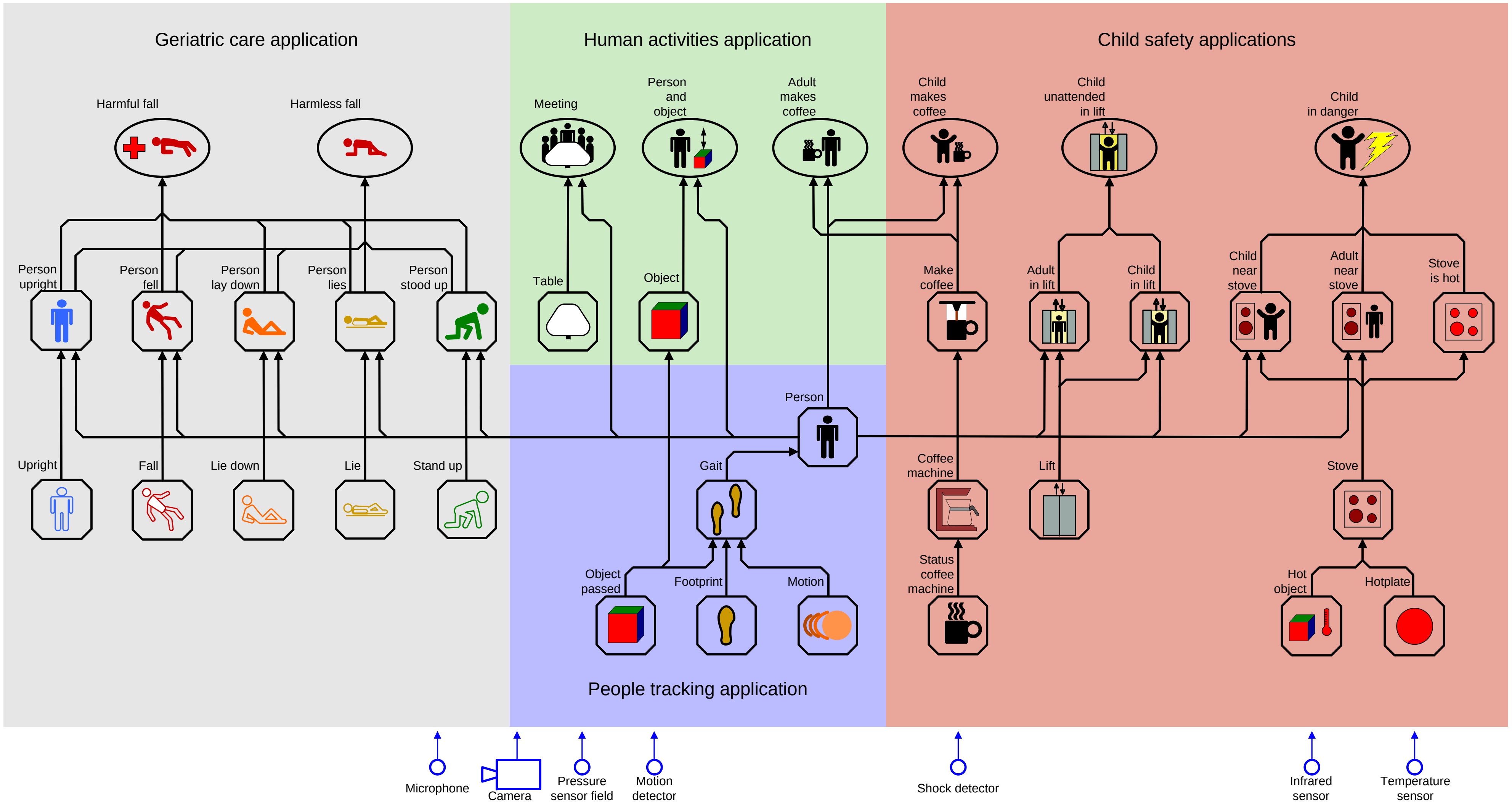


Figure 96: Joint symbol tree of ARS-PC

7 Prototype of the ARS project

In this chapter an implementation is described which uses the model described in the previous chapters. It uses data from the kitchen of the ICT and is based on the ARS framework (see Chapter 5). The layout of the kitchen and the used sensors, and the implemented applications are described in Chapter 4. An embedded system is used to collect the sensor data [Goe06]. A prototype of proposed DPAL (Data Point Abstraction Layer, see Chapter 3.5) has been implemented in order to provide data independently from a building automation system.

Alternatively, the data can be provided with a simulation tool [Har07]. This tool allows simulating scenarios in a virtual environment. The output of the simulator consists of sensor values of the virtual sensors. At present, pressure sensors, light barriers, and motion detectors are implemented. It is planned to add temperature sensors, cameras, vibration sensors, and others. An editor of the scenarios allows their definition, which is simulated by the simulator.

The sensor values of both the real implementation and the simulation can be stored in the sensor database or can be fed directly into the ARS system. A rule-based symbolization and a fuzzy-logic based implementation can be used as implementation for the ARS-PC module.

In [Goe06], a rule-based approach for a coffee making scenario is described. The coffee making scenario is defined as a person entering the room, taking a cup from the cupboard, using the coffee machine, taking milk from the fridge (optional), and leaving the room. This implementation was intended as a proof-of-concept prototype of the ARS architecture. The recognition rate of the scenario has reached about 90%.

[Ric07] presents a fuzzy logic-based implementation, which is based on this work and the work of [Pra06] and [Goe06]. It was designed to track people, recognize whether a person is making coffee, when someone is having a meeting, when someone puts down an object and leaves the room, and when an unattended child is near the stove. This software was also tested with the simulation tool described above. In the following chapters, the implementation of the concept presented in this work is described in more detail.

7.1 Provision of Sensor Data

In principle various methods for the provision of data are possible. In this work, DPAL (Data Point Abstraction Layer) is proposed for a fieldbus-independent abstraction of data points. In [Goe06], hardware with direct access was implemented, which allows an easy access to sensor values. A virtual environment including objects and sensors was described in [Har07]. Virtual people can interact with the environment and thereby trigger the sensors, which generate values similar to a real implementation. In the following chapters, these possibilities are described in more detail.

7.1.1 Implementation of the Data Point Abstraction Layer

In Chapter 3 methods for data point abstractions without explicit gateways are presented and discussed. The proposed Data Point Abstraction Layer (DPAL) (see Chapter 3.7) was implemented to CNP and BACnet. Since the used protocol stacks (CNP stack and BACnet stack) use an operating system service interface, it was also used for the adoption layers, DPAL, DPAL configuration instance and test applications (for the layout of the modules see Figure 17). This interface provides an abstraction for operating system services like threads, file access, and inputs/outputs. It is available for the operating systems Windows NT 4.0™, Windows2000™, Windows XP™, Linux, and RTEMS (Real Time Executive for Multiprocessor Systems). Therefore, the DPAL implementation can be used in principle on any hardware that supports one of the above operating systems and has access to the fieldbus network interfaces.

A prototype has been developed for the verification of the concept. In the meantime this approach has been developed further and is used in commercial products. However, a solution was chosen for the implementation in the ARS project without using a fieldbus system. Due to our limited budget, only one room (kitchen on the ICT, see Chapter 4) was equipped with sensors. This approach is described briefly in the next chapter.

7.1.2 Implementation of Direct Access

The sensors are connected to an embedded system which provides the sensor values of the ARS system [Goe06]. Via an I^2C (Inter-Integrated Circuit) [I2C] interface the analogue input modules are connected with the processor of the system. Furthermore, the system has an ISO 8802.3 interface (Ethernet) [ISO8802] and an IP-Stack [RFC791]. The values of the input modules are polled cyclically. In case a value has changed, the new value is propagated via the TCP (Transmission Control Protocol) [TAN02]. The cycle time of reading the inputs is 75 ms, including a noise reduction mechanism.

On the basis of this real world implementation, a virtual environment has been developed. This virtual environment is able to simulate people and their actions. Actions are detected by virtual sensors, which lead to sensor values that are similar to the sensor values measured in the real-world implementation. The following chapter describes this simulator briefly.

7.1.3 Implementation of the Simulator

[Har07] describes the simulator of a virtual office environment. The simulator was designed to simulate sensor values in order to perceive scenarios in this virtual environment. The reasons for simulating the sensor values are on the one hand cost reduction of such installations, on the other hand, the simulator enables the evaluation of which sensors are necessary and where they should be mounted.

As described in [Har07] and depicted in Figure 97, the simulator is split into three parts. The first part is the scenario editor, which allows defining the virtual environment and the scenarios within this environment. The second part is the simulation engine. The simulation engine calculates the sensor values based on the virtual environment and the scenarios defined by the scenario editor. The third part is the visualization. On the one hand the virtual environment and the scenarios are displayed. On the other hand the visualization also has an interface to the ARS system. Using that interface, the scenarios and symbols recognized by the ARS system can be displayed simultaneously with the virtual environment.

The scenario editor is used to define the physical properties of the environment, e.g. the building or room. Within that environment, not only objects like table, stove, etc. can be placed but also people who are referred to as objects in the simulation environment. Unlike the environment, objects are part of scenarios, which means that they can be involved in actions and can cause a change of values. Sensors are the third group of entities which are defined in the environment editor. From these definitions scenarios are compiled and stored in the scenario database. The sensor properties are also stored in the sensor database of the ARS system (see Chapter 5.1) because the ARS system needs the sensor properties to interpret the sensor values.

The simulator imports the scenarios from the scenario database. The sensor output simulation uses the mathematical models of the simulator to generate sensor values from the scenarios. The simulated sensor values are stored in the ARS sensor database. In the ARS part, scenarios are recognized from the values of sensor database by using symbols as described in the previous chapters.

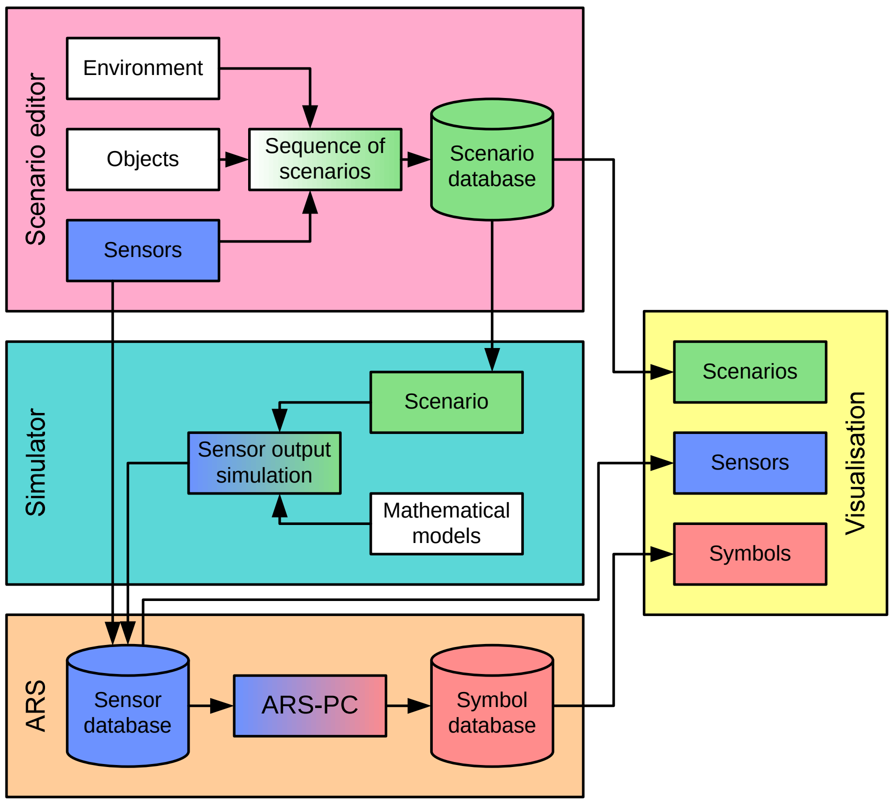


Figure 97: Concept of the simulator [Har07]

With the visualization a three-dimensional virtual environment can be displayed as well as scenarios as defined in the scenario editor and stored in the scenario database. Additionally, the sensor values from the sensor database can be visualized in the virtual environment. The symbols and recognized scenarios are imported from the symbol database of the ARS system into the visualization. On a separate screen these symbols are displayed. With this visualization, the user can compare the simulated scenarios and the recognized simulated scenarios. Chapter 8 shows the results of this comparison by means of the scenarios *meeting* and *child in danger*.

As described in [Har07], the scenario editor, the simulator, and the visualization are implemented in C++. They use DirectX 9, which is an application programming interface for handling tasks related to multimedia form Microsoft platforms. Therefore, a Microsoft operating system like Windows 2000 or Windows XP is needed. A special hardware requirement is, that a graphic card has to be used which supports DirectX 9.

7.1.4 Implementation and Integration of the Sensor Database

As described in Chapter 5.1 and depicted in Figure 25, the sensor data can be stored in the sensor database. In Chapter 5.7 the tables of the database are defined and shown in Figure 34. The sensor database is intended to store the sensor values. Therefore it does not matter whether the data come from a real-world implementation (DPAL or direct access) or from a virtual environment. The database is designed to store data in an efficient way but flexible enough to store data from different sources.

The code for the generation of tables of the database as well as the code to access the sensor data is written in SQL92 (Structured Query Language) [SQL92], which can be used for a broad range of currently available databases. In the implementation described in this work an Oracle database was used.

Introducing the database into the ARS project has eased the development of the components of the ARS-PC part tremendously. Scenarios of the real-world implementation have been recorded and they used the decision-making algorithms as described in Chapter 7.2. Furthermore, the database allows training statistical methods in reasonable time since the data from the sensors of the kitchen have been recorded from the beginning of the project. If the sensor values are simulated with the simulator (see Chapter 7.1.3), the sensor database is required to buffer the sensor data, since the simulator does not generate the sensor values in real-time.

7.2 Methods for Decision Making

Methods to provide the ARS system with data are described in the previous chapters. To generate and manipulate symbols, decision making-algorithms have to be used. Using SymbolNet (see Chapter 5.5), the following decisions have to be made: creation of a symbol, update of a symbol and expiring of a symbol. To make these decisions, the following approaches have been discussed. A rule-based approach has fixed rules, usually expressed by an if-then relationship. Stochastic methods calculate the reparability of different decisions, usually based on the Bayes' rule. Artificial neural networks are trained to bring similar results in similar situations. Knowledge-based systems, also called expert systems, make decisions based on knowledge of the system (rules) and the analysis of current information (input to the system). Fuzzy Logic uses membership values and fuzzy sets to make decisions. In the following chapters these methods are briefly described.

7.2.1 Rule-based approach

The rule-based approach is the most intuitive approach for decision making. The (fixed) rules of decisions are stored in a rule base (e.g. if-then clauses). A rule associates the value range of the variables with decisions. The value range is fixed. [Goe06] has chosen this approach. In the kitchen on the ICT, scenarios like the coffee-making scenarios have been implemented. In particular, the rule-based approach is suitable for sensors set up with binary sensors (light barrier, pressure sensors, door contacts…) since binary values consist of only two values.

However, in some cases the fixed definition of the value ranges is a disadvantage, e.g. in case of human gait, the distance between two footsteps is used as criterion to decide whether two recognized steps belong to the same person. Steps smaller than the defined maximum length of a step are associated with belonging to the same person, whereas steps longer than the defined length are not. Yet, our experience in day-to-day life tells us that an exact (maximum) length of a step is hard to define.

7.2.2 Stochastic methods

[Bru07] describes how to use stochastic methods for building automation. Here, the system learns from sensor values which values (and changes of values) have a high likelihood. After this learning phase, the system associates similar sensor values with the learned states. In [Bru07], a system is described, which uses sensor values from an office building to train a Hidden Markov Model. However, in many cases the learned states can be interpreted by people but there are also states which are meaningless for people.

In systems like the ARS system, the symbols are predefined and meaningful for people. This does not match the training phase because the symbols cannot be predefined. However, in future learning will be introduced into the ARS project. Therefore, there is currently research work being conducted on the ICT to combine Hidden Markov Models and symbol trees[25].

7.2.3 Artificial neural networks

Artificial neural networks [Arb95, Roj96, Sch97, and Cal03] consist of layers of artificial neurons. They were designed as an emulation of natural neuronal networks, which can be found in the spinal cord and brain. Artificial neural networks usually have to be trained. However, it is impossible to predict whether a similar situation as the one that has been trained will be recognized by the network. Therefore, input data of an artificial neural network have to be pre-processed.

Moreover, some symbols of the symbol tree used in ARS cannot be trained in a real installation, e.g. the child-in-danger scenarios. Such scenarios could only be trained using simulators (see Chapter 7.1.3). The

[25] However, Hidden Markov Models are not appropriate to simulate learning in a high level sense according to psychoanalytical model. If Hidden Markov Models are suitable for the learning symbols without contradicting the neuro-psychoanalytic model has to be checked in that research work. Nevertheless, from my point of view Hidden Markov Models can only be used for learning of low level symbols if the basic emotional systems are considered.

configuration of artificial neural networks has to be done very carefully, and therefore experienced stall is required.

7.2.4 Expert systems

Expert systems, which are also known as knowledge-based systems, try to emulate the knowledge and skills of a human expert [Jac99]. Such systems usually consist of a knowledge base and an interference engine. The knowledge base contains expert knowledge (problem dependent), and the interference engine is used to operate on the knowledge base (problem-independent).

The separation of knowledge and the problem-solving method facilitates the operation of these systems as well as their configuration and maintenance. Changing and growing knowledge of a certain problem type can be included. However, expert systems are usually specialized for a problem domain. Therefore, I did not concentrate the research work on the system itself rather than on the methods on which the system is based. A widely used method is the Fuzzy Logic as described in the next chapter.

7.2.5 Fuzzy Logic

In contrast to Boolean logic, fuzzy logic does not represent the values true and false but rather membership of truth. Membership is usually expressed as a value between zero and one. This allows creating systems that can deal with fuzzy information, as people are used to. Linguistic rules are used to map the input of the system to the fuzzy output values [Kru95]. These linguistic rules can be formulated similarly to a verbal description of a problem that needs solving. Therefore, fuzzy logic is suitable for the description of systems which cannot be mathematically described or can only be presented with very complex models [Bon92].

7.3 Used Decision-Making Algorithms

Some restrictions have been defined for the decision-making algorithm within the ARS project. At the current status of the project, learning should be excluded. Therefore, the stochastic methods and the neural networks cannot be used. Nevertheless, learning has to be implemented in future implementations. However, in the ARS-BASE branch of the project stochastic methods (Markov Models) are used to identify unusual situations [Bru07], but without identifying the situation itself.

The rule-based approach is implemented in [Goe06]. This approach is straightforward in its implementation and very well suited for binary sensors. However, this approach is not always suitable for higher-layer symbols (snapshot and representation symbols). The following example this should be clarified. The human gait has several characteristics, which are interdependent. [Heg00] lists the relationship between relative step length[26] and relative velocity[27]. In the current implementation, the step

[26] Length of (double) step in relation to body height

[27] Walking speed in relation to body height

length is measured. To decide which steps belong to the gait of a particular person, the length and velocity of a step have to be taken into account. Simple if-then rules do no longer apply. As described above, the algorithms that have to be trained prior to their application are not used in the first part of the project. Therefore, artificial neural networks and stochastic algorithms are not used.

The decision to use fuzzy logic in the project was made because it meets the needs of the project. It makes it possible to describe problems in a verbal manner as people would do. Another constraint was that open-source implementation should be used because of the limited budged of the project.

The ARS-PC system was developed in Java because a SymbolNet implementation was already available [Pra06]. Further, a fuzzy logic package is available in Java. The open source package jFuzzyLogic [JFL] is published under the GNU License GPL [FSFE]. It implements a complete Fuzzy Inference System (FIS) as well as Fuzzy Control Language (FCL) according to IEC 1131 [IEC1131-7]. The target platform of the ARS-PC system is a standard PC with either a Microsoft Windows ® or Linux operation system. In [Ric07], the implementation of the ARS-PC system is described. This implementation uses the symbols as defined in Chapter 6 in order to recognize the scenarios defined in Chapter 4.

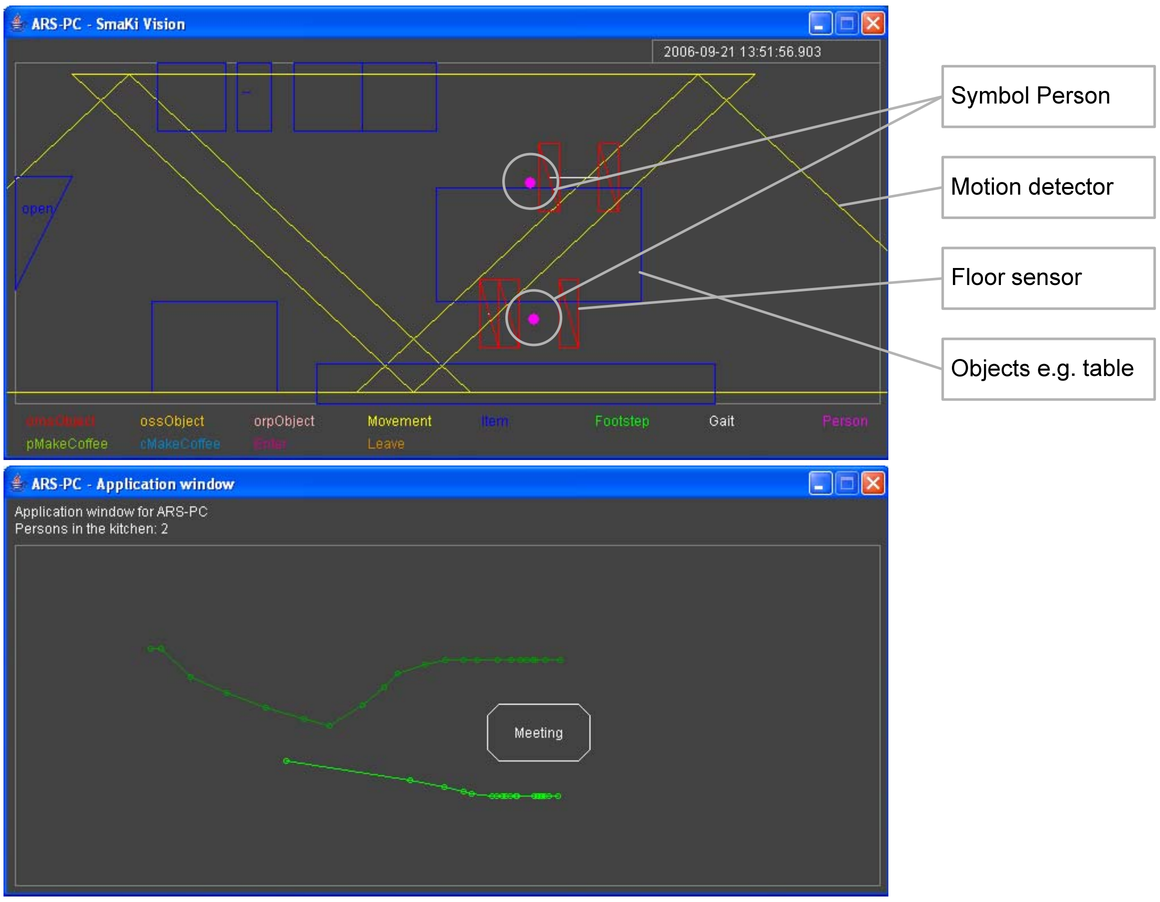
7.4 Implementation of Symbol Database

Within the ARS framework (see Chapter 5.1) a database for storing symbols is introduced. In this symbol database, all messages received from the ARS-PC system via SymbolNet are recorded. The design of the tables of the database is described in Chapter 5.8. Similar to the implementation of the sensor database (see Chapter 7.1.4), the symbol database uses the SQL 92 standard for accessing the tables [SQL92]. Similar to the sensor database, a broad range of database products can be used for the implementation of the symbol database. For the implementation described in this work an Oracle database was used.

Since SymbolNet (see Chapter 5.5) has been designed to distribute symbols and their properties within the ARS-system, an interface to the symbol database had to be developed. Using this interface, the symbols of the ARS-PC system can be recorded in the database. The three-dimensional visualization of the simulator (see Chapter 7.1.3) and the two-dimensional visualization of the kitchen on the ICT (see Chapter 5.9) read the data from the symbol database and display the recognized symbols and scenarios.

7.5 Implementation of Visualizations

As described in Chapter 5.9, a two-dimensional and a three-dimensional visualization are integrated in the ARS framework. The implementation of the three-dimensional visualization is part of the simulator and described in Chapter 7.1.3. The two-dimensional visualization uses the SymbolNet as communication interface. It makes it possible to obtain the symbols either directly from the ARS-PC system or from the symbol database. The visualization is implemented in Java. Therefore, it can be used with standard PCs together with a Microsoft Windows ® or Linux operating system. Figure 104 shows a screenshot of the two-dimensional visualization. Additionally to the two-dimensional visualization of the scenarios and symbols recognized in the kitchen on the ICT two visualizations were implemented in [Ric07] which have direct access (i.e. without SymbolNet) to the symbols within the ARS system. These visualizations have been introduced to facilitate the design of the ARS-PC system. One visualization type gives a spatial overview like the two-dimensional visualization, and the other gives a logical overview of the symbols within the system. In Figure 98 two spatial views of a meeting scenario are shown. In the upper screenshot the (active) sensors and the symbol *person* are visible, and the lower screenshot shows the recorded gait and the recognized scenario *meeting*.



Application window for ARS-PC
Persons in the kitchen: 2

Figure 98: Integrated visualization

The logical view of the symbols in the system is depicted in Figure 99. It is divided into five layers, i.e. sensor layer, microsymbol layer, snapshot symbol layer, representation layer and scenario detection layer. In the bottom layer, the sensor layer, the types of sensors used in the system are shown. In the implementation in the kitchen of the ICT the sensors are tactile sensor, door contact sensors, motion detectors, fridge door contacts, and a sensor in the coffee maker. The next higher layer is the microsymbol layer. In this layer the microsymbols implemented in the system are shown. The lines between sensor layer and microsymbol layer indicate which sensor type is associated with a microsymbol.

In general, the symbols are visualized in frames. In case the frame is black, no symbol of this type is currently active in the system. In case the frame is red, at least one symbol of this type currently exists in the system. If more than one symbol exists, the number of symbols is shown in the lower right-hand corner of the frame. The association between the symbols is indicated with lines. Black lines mean that currently no association between the symbols is active, red lines indicate active associations.

In the representation layer, snapshot symbol layer and microsymbol layer the symbols are displayed according to their membership. The detected scenarios in the implementation in the kitchen are the recognized scenarios of a person putting down an object (see Chapter 4.3), a meeting (see Chapter 4.5), a person making coffee (see Chapter 4.6), a child approaching the hot stove (see Chapter 4.4), and a child making coffee (see Chapter 4.6). If any of these scenarios is recognized, the black frame changes its color to green. In Figure 99, the logical view of the meeting scenario is depicted. The spatial view of this scenario is shown in Figure 98.

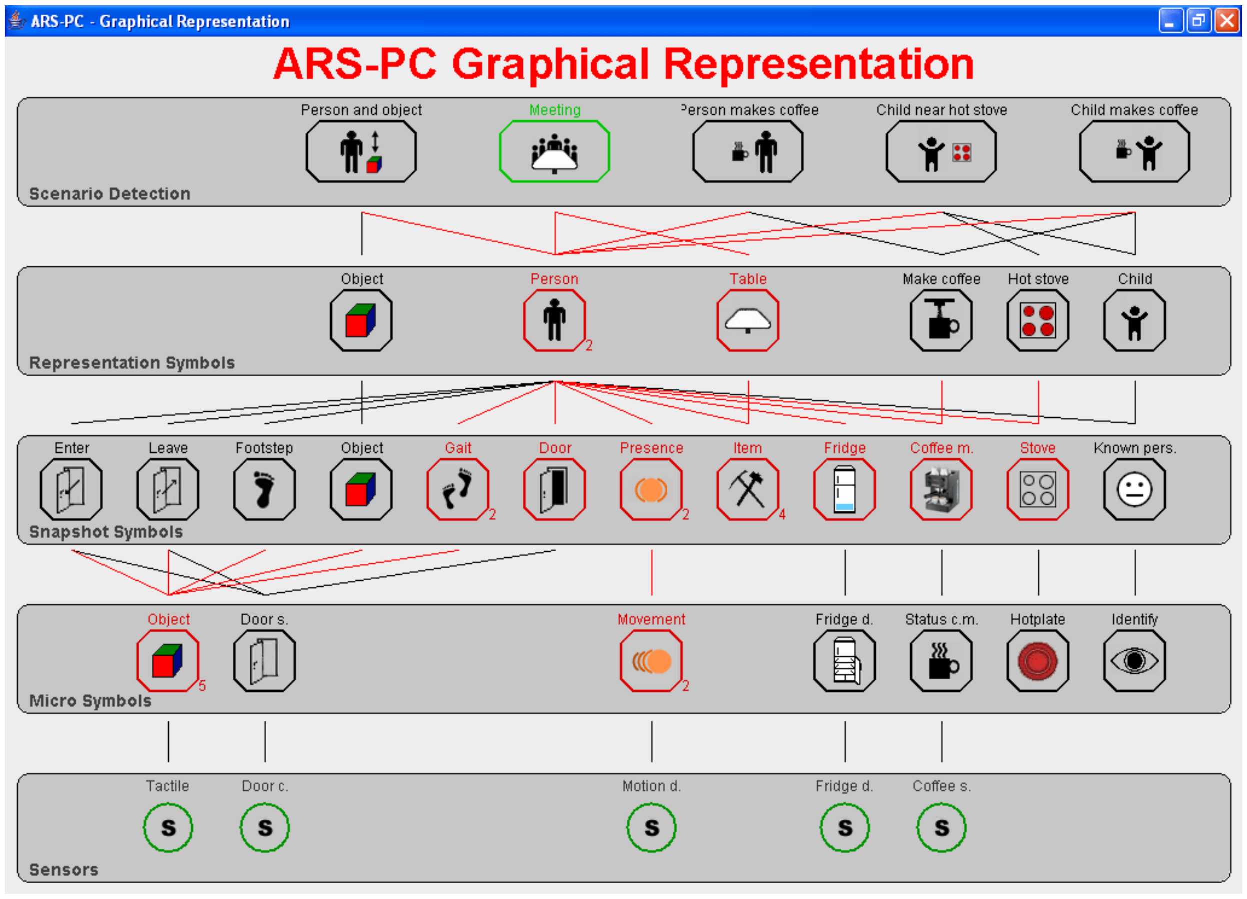


Figure 99: Screenshot of logical view of the symbols

Since these visualizations have direct access (embedded within the Java application) to the ARS software, it is possible to display the process within the system in more detail than with the two-dimensional visualization mentioned above. Nevertheless, these visualizations do not have user interfaces, which can be used to parameterize the displayed items. The parameterization has to be done within the Java source code. Therefore, the two-dimensional and three-dimensional visualizations are intended to be used by the user of the system. However, the logical view also gives the user a good impression of the mechanism of the ARS-PC system. Using the tools described in the previous chapters, the concept of this work is verified as described in the next chapter.

8 Realizations and Results

The system described in the previous chapter has been evaluated in order to realize the concepts described in this work. The concepts have been developed in order to gather sensor data from the environment and find predefined situations and scenarios. To extract these situations and scenarios from the sensor data, a symbolic approach has been chosen based on the brain models by Aleksandr Luria and Mark Solms. In the following chapter, the results of these tests are presented.

8.1 Data Provision

As described in Chapter 7.1 three variants for data provision are designed in the ARS system, i.e. the direct access of the sensor via a specialized hardware, the DPAL (Data Point Abstraction Layer) to access data in an abstract way from (different) fieldbus systems, and the simulator which generates data based on a virtual environment and predefined scenarios. However, the generated data can be stored in the sensor database. Since the sensor database holds besides sensor values also additional information about the sensor (active region, units, etc. - see Chapter 5.7), the access of the database is necessary for the operation of the ARS-PC system.

In this work, the direct access of the sensors and simulator has been used for the provision of data in the ARS-PC system. Additionally, a sensor database was used because on the one hand, it provides the properties of the installed sensors (mounting position, active area, units, etc.) and on the other hand it stores sensor values. The outline and sensors of the kitchen are shown in Figure 24.

The simulator [Har07] of the ARS system was designed to simulate sensor values of a virtual environment. [Pra06] has proposed the outline of such a virtual environment as depicted in Figure 100. It consists of four rooms and a corridor. In order to model the corridor, it was split into two rooms. This had to be done because according to the definition by [Pra06] a room consists of four rectangular walls. However, this virtual environment was used as outline for the simulation of sensor values.

In the scenario editor (see Chapter 7.1.3), this virtual environment was defined in order to be used as outline for the simulator. Room 1 is equipped with a table and six chairs and is intended as meeting room. In room 2 there is a kitchen with stove and coffeemaker. The rooms 3, 5, 6, and 7 are office rooms with desks and chairs. The hallway (room 4 and 8) is empty. The exact equipment of the rooms can be seen on the left of Figure 103. The following scenarios have been defined in the simulator, meeting in room 1 and adult making coffee, child making coffee, and child approaching the hot stove in room 2. In the other rooms people either walk around or sit at their desks.

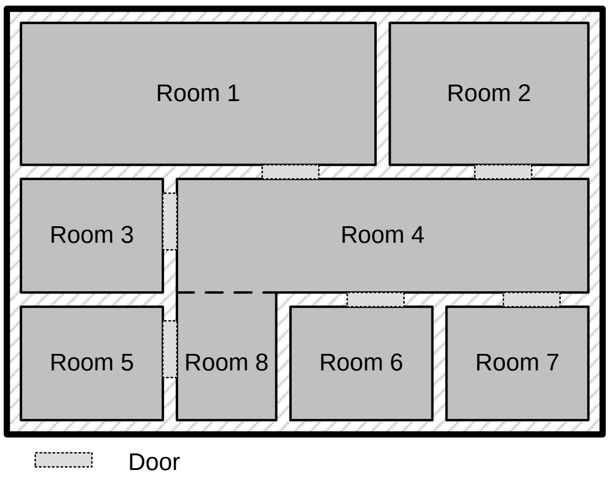


Figure 100: Outline of virtual environment [Pra06]

Both the direct access to the sensors of the real implementation and the simulated sensor values of the described virtual environment are used as input data of the ARS-PC system. In the following chapter, the test based on these two data sources is described.

8.2 Recognized scenarios

The ARS-PC system is designed to recognize predefined scenarios, independently from the environment. In this work, the system has been tested with the real implementation in the kitchen of the ICT and with the virtual environment of the simulator. However, the simulator was not available during the development of the ARS-PC system, so the test with the simulator had to be done after the work on the ARS-PC prototype was finished. Therefore, the applications of the ARS-PC system rely only on the sensors that were available in the real implementation (see Chapter 7.5). To test the concepts proposed in this work, the following applications have been tested: recognition of a person putting down an object (see Chapter 4.3), meeting (see Chapter 4.5), person making coffee (see Chapter 4.6), child approaching the hot stove (see Chapter 4.3), and a child making coffee (see Chapter 4.6). However, after its completion, the ARS-PC system has been tested with the simulator.

8.2.1 How to test scenarios

The scenarios defined in Chapter 4 are described verbally. It is not defined how persons exactly have to behave. People who reproduce such scenarios will always act differently, and therefore trigger different sensors. Even the same person acts differently if s/he is asked to reproduce a scenario twice. Furthermore, people interpret scenarios differently. If a recorded scenario (or simulated scenario) is to be evaluated by different people, the results may differ. Therefore, in principle it cannot be assured that a scenario will always be recognized by the system as it would be recognized by a human observer.

Beside that principal problem of testing scenarios, the sensors used in the real implementation or the simulated sensor data can limit the possibility of scenario recognition. As with the recognition of a person taking a journal from a shelf, a person falling and lying on the floor, danger of fire, and audio system self-tests, no sensors were available to recognize these scenarios. In principle it would be possible to use the simulator to generate such sensor data. But to do so the physical properties of people and objects have to be modeled first in order to obtain realistic sensor values.

Nevertheless, testing the ARS-PC system is possible. On the one hand, tests have been carried out during the development phase to make the definitions of fuzzy sets and fuzzy rules possible. Figure 101 shows the test setup. The left shows the two sources of sensor data, the real implementation in form of the kitchen on the ICT and the virtual environment from the simulator. The ARS system uses the sensor data in order to recognize the predefined scenarios described above. On the right side of the figure the visualization of the recognized symbols is shown.

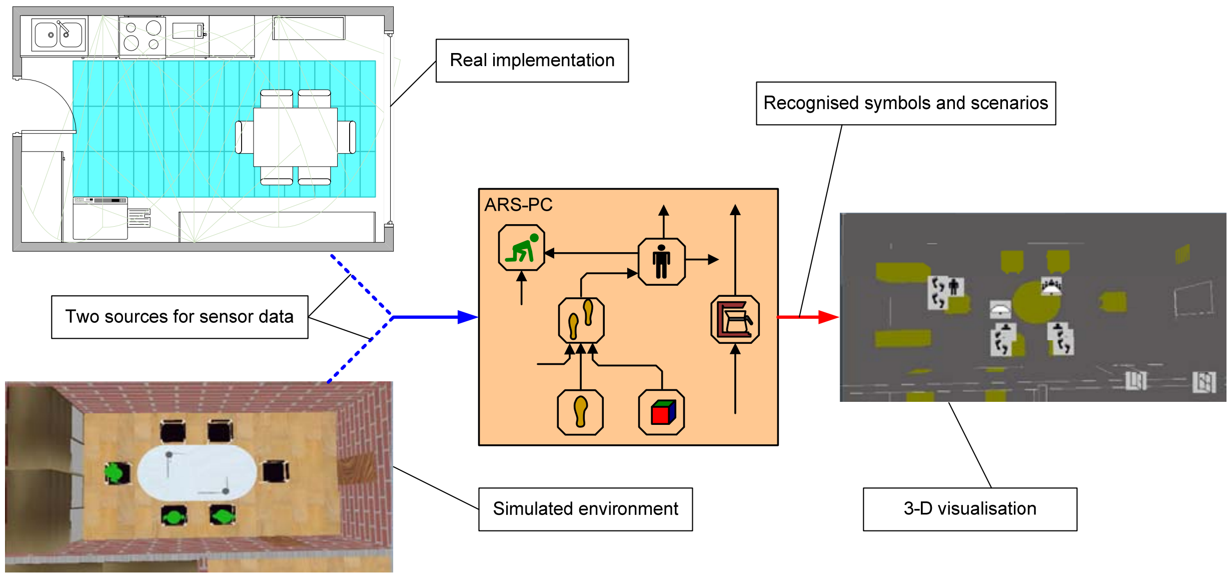


Figure 101: Setup of the test environment

It has to be mentioned that the software of the ARS-PC system, which is shown in the middle of the figure, was changed minimally in order to be used with the simulator instead of the real implementation. The adaptation is only related to changed dimensions, which means the symbols of the ARS-PC system have not changed. After its implementation, the system has been tested during permanent installation in the kitchen of the ICT. However, the tests in the real implementation have only been carried out with people were cooperative. This means that they showed normal behavior. The ARS-PC system is not designed to replace e.g. alarm systems, at least not as long as the hardware is not secured. The results of the tests are presented in the next chapters.

8.2.2 People tracker

The application people tracker is described in Chapter 4.2. It is the basis of the other applications, in which scenarios are detected. The task of the people tracker is to find people in a room and determine their position. In the implementation in the smart kitchen, this task can be seen as successfully applied. Nevertheless, the system has weaknesses in detecting people leaving the room. As shown in Figure 24, the floor sensors do not cover the whole room. Around the door no floor sensors have been mounted because the floor sensors need to be mounted in the floor and covered with an additional floor coating, and the space between door and floor is too small for mounting the pressure sensors.

To overcome this problem, several mechanisms have been implemented in the system. It is assumed that a person leaves the room when no pressure sensor or motion detector is active. But if more than one person is in the room, the system cannot detect one person leaving the room. Another assumption is that someone who goes straight to the door will leave the room. But considering Figure 24, it can be seen that due to the dimensions of the pressure sensors it is difficult to determine whether a person walks straight to the door or not. Another strategy to detect a person leaving the room is to consider the status of the door. If the door is closed, someone wanting to leave the room needs to open the door first. Thus, it can be assumed that a person opening the door will also leave the room, especially if afterwards the door is closed again and no one can be detected in the room anymore.

However, the sensor configuration mentioned above does not allow a reliable detection of someone in the room. Therefore, additional sensors are needed especially in the area around the door. Such sensors can be pressure sensors, light barriers as proposed in Chapter 6.1.2, or camera systems with person detection algorithms. This problem does not exist in the virtual environment of the simulator. On the one hand, the floor pressure sensors cover the whole room. On the other hand, the neighboring room, into which the person goes when leaving the first room, has sensors which allow the detection of the person. Therefore, there are no “blind” areas especially around the door.

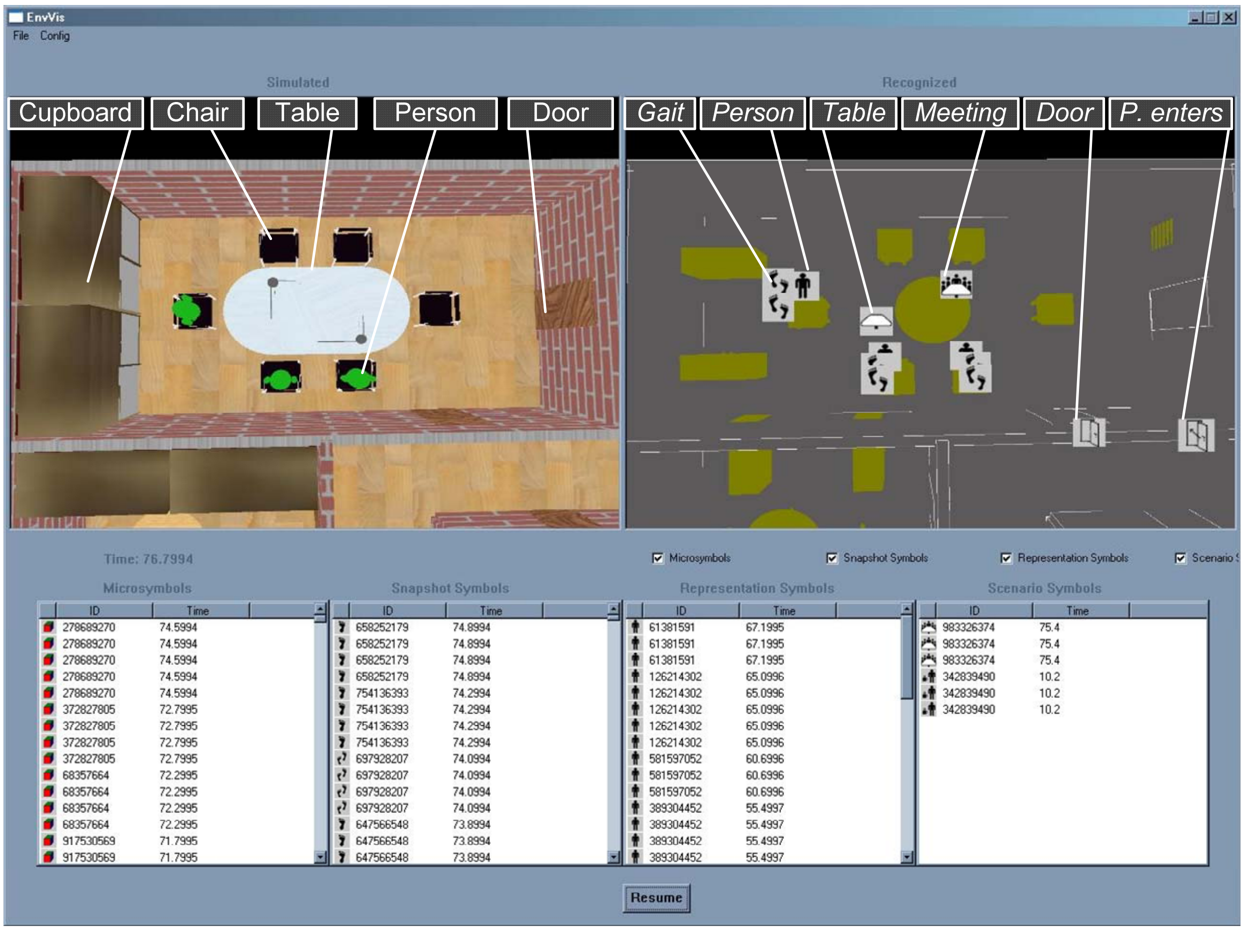
A similar problem as the problematic detection of a person in the area around the door is the detection of people in the area around the table. Although the entire area around the table is covered with pressure sensors, people cannot be tracked well because the table activates some of the sensors permanently. If someone is standing on such a pressure sensor, no new microsymbol object is generated, and therefore the gait detection fails. However, in such a case the person is still detected, just not in the right position. If the person leaves the area around the table, the property *position* of the symbol *person* is again updated. The system could be improved using pressure sensors with an analogue output instead of a binary one. The output of the used pressure sensors gives only information about the activation of the sensor (activated or not) but not about the pressure on it, in which case it would be possible to generate a microsymbol *object* whenever the pressure changes. In this case, a footstep could be detected even if a table is placed on the same pressure sensor. Another solution would be to increase the resolution of the pressure sensor field on the floor.

Another weakness of the system is the recognition of the exact number of people in a room. Depending on their position, up to seven people have been detected correctly. However, if people stand too close to each other, the detection mechanisms may fail. In such a case either two people are detected as one, or the system detects more people. To overcome this limitation, a higher resolution of the pressure sensors or additional sensors which allow people tracking, e.g. computer vision, might be used. Pressure sensors which are able to detect the weight of a person could be used to increase the accuracy of the tracking algorithms. Nevertheless, in both the real installation and the simulator, the people tracking application has been extensively tested. It has been tested explicitly and also implicitly during the tests of the other applications, since the people tracking application is the basis of all scenario recognition applications. The results of the scenario recognition application are presented in the next chapter.

8.2.3 Results of scenario recognition

As described in the previous chapter, the people tracker application is the basis of the scenario recognition applications of the ARS-PC system. Therefore, all problems connected with the weakness of the people tracking application can also cause problems with the detection of scenarios. The scenarios which can be detected with the implementation of the ARS-PC system are described at the beginning of Chapter 8.2. As with the people tracking application, the scenario recognition applications have been tested in both the real implementation and with sensor data from the simulator. Exemplarily, the scenario *meeting* and the scenario *child in danger* are described.

Meeting Scenario: The scenario *meeting* is described in Chapter 4.5. The scenario meeting occurs when two or more people sit around a table. A screenshot of the three-dimensional visualization of the scenario is depicted in Figure 102. The left side shows the visualization of the virtual environment. It shows a table and six chairs. The green objects are people, who sit on three chairs. On the left side of the room are cupboards, and there are doors in the opposite wall and the lower wall.



Time: 76.7994

Microsymbols

ID	Time
278689270	74.5994
278689270	74.5994
278689270	74.5994
278689270	74.5994
278689270	74.5994
372827805	72.7995
372827805	72.7995
372827805	72.7995
372827805	72.7995
68357664	72.2995
68357664	72.2995
68357664	72.2995
68357664	72.2995
917530569	71.7995
917530569	71.7995

Snapshot Symbols

ID	Time
658252179	74.8994
658252179	74.8994
658252179	74.8994
658252179	74.8994
754136393	74.2994
754136393	74.2994
754136393	74.2994
754136393	74.2994
697928207	74.0994
697928207	74.0994
697928207	74.0994
697928207	74.0994
647566548	73.8994
647566548	73.8994
647566548	73.8994

Representation Symbols

ID	Time
61381591	67.1995
61381591	67.1995
61381591	67.1995
126214302	65.0996
126214302	65.0996
126214302	65.0996
126214302	65.0996
126214302	65.0996
581597052	60.6996
581597052	60.6996
581597052	60.6996
389304452	55.4997
389304452	55.4997
389304452	55.4997
389304452	55.4997

Scenario Symbols

ID	Time
983326374	75.4
983326374	75.4
983326374	75.4
342839490	10.2
342839490	10.2
342839490	10.2

Figure 102: Test scenario *meeting*

The right side of the Figure 102 shows the recognized symbols and scenarios. To increase the clarity, only recognized symbols are shown. The green shapes represent objects in the room. In this case, the objects are cupboards, chairs, and a table. Special objects like table or door are also visualized with a symbol. However, these symbols and shapes are static and do not change. They are generated at the start of the system and never expire. When a person enters the room, the symbol *person enters* is generated. In the screenshot, this symbol is still active and therefore displayed. It was generated when the third person entered the room. The symbols, generated when the other two entered, have already expired and are therefore no longer displayed.

When a person walks around, it triggers the floor sensors which generate the microsymbol *object* and the microsymbol *footstep*. These symbols are not shown in Figure 102. However, the symbol *gait* is generated from these symbols which can be seen in Figure 102 as well as the symbol *person*. When two or more people sit around the table, the scenario symbol *meeting* is generated. The icon with a table and persons around it represent this scenario symbol. In the lower part of the screenshot, four tables can be seen, one for each type of symbol. Within the tables the recognized symbols are shown.

The scenario *meeting* was always detected when the simulator was used as data source. In case of data from the real environment, the scenario *meeting* has only been detected correctly when the people could be detected properly. The challenges for the detections of people in a real environment are described in Chapter 8.2.2.

Child in danger scenario: The scenario *child in danger* is described in Chapter 4.4. The scenario occurs when an unattended child is near a hot stove. In Figure 103, a screenshot of the three-dimensional visualization of the simulator is shown. Differing from Figure 102, only the simulated environment and the recognized symbols and the scenario are depicted, the tables of the recognized symbols have been omitted.

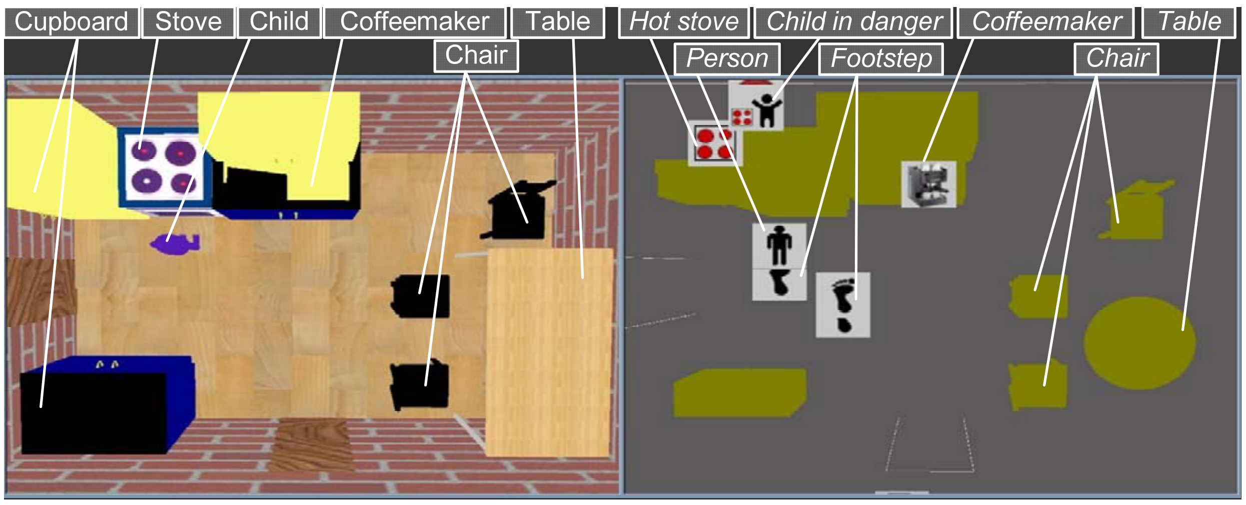


Figure 103: Test scenario *child in danger*

The left side of Figure 103 shows a kitchen environment with cupboards, stove, coffeemaker, a table, and chairs. The purple shape in front of the stove depicts a child. In contrast, adults are depicted as green shapes (see Figure 102). On the right side of Figure 103, the recognized symbols are shown. Special equipment like stove and coffeemaker have their own symbols and are therefore visualized with special icons. The chairs, table, and cupboard are green. The symbol *footstep* was generated when the child approached the stove. The symbol *person* shows the currently recognized position of the child. The red hotplates of the stove show that the stove is hot. Therefore, the scenario is recognized and the scenario symbol *child in danger* is present.

Not shown in Figure 103 is the scenario with an attended child. In this case, the scenario is not detected as it is defined. Since up until now no child detection has been implemented in the implementation (a possible solution is proposed in Chapter 9.2), the child is detected by a button in the graphical user interface of the ARS-PC system. If the button is pressed and a person is recognized near the kitchen door, this person is recognized as child. However, the recognition was successful in the simulator, provided that an adult pressed the button when a child is was in the room.

In the real implementation no child detection is implemented either. To test the scenario, the same mechanism is used as described above. Therefore, adults triggered the scenario *child in danger*, since every person is recognized as child when a user presses the child button of the ARS-PC system. However, as is the case in the meeting scenario, the correct detection of the position of people is necessary. Regarding the layout of the real implementation of the kitchen on the ICT (see Figure 24), it can be seen, that the stove is close to the door, where no sensors are mounted. Thus, someone entering the room might be detected quite late, especially if the person enters the room very quickly. Improved person detection, as described in Chapter 8.2.2, would also help to improve the detection of the scenario *child in danger*.

9 Discussion

The thesis at hand presents the technical model of a system which is able to perceive predefined situations in buildings. Although the applications are designed for buildings and the data provision of proposed data abstraction is designed for fieldbus systems in building automation, the concept of the scenario recognition presented in this thesis is not limited to building automation, which has been chosen as environment here because the scenarios are easier to define, and it is simpler to integrate sensors into a building with showcases. Other applications might be service robots, e.g. for homes of elderly people, or in principle any other kind of automation. In the following the thesis is discussed and possible future work is presented.

9.1 Conclusion

The situations defined in this work can be categorized roughly into the applications people tracking, child safety, geriatric care, and human activities. The people tracking application is the basis for the detection of scenarios in which people are involved. Examples for child safety are the detection of an unattended child near dangerous objects like a hot stove, or near stairs or lift. Examples of geriatric care application are the detection of a fall and its possible impact on the person. The category of human activities detection is very general and includes therefore the activities of the child safety and geriatric care applications. An additional example of human activities is the detection of a person making coffee. For the applications described in Chapter 4, the symbols and their relationships are defined in this work.

Symbols are the building blocks of the ARS-PC (Artificial Recognition System - PerCeption) system. They represent people, objects, and actions. The mammalian, respectively the human brain is the archetype of the technical model of these symbols. The results from neuroscientists, psychoanalysts, and neuro-psychoanalysts, namely Panksepp, Luria, and Solms, were taken as basis for this technical model. The main features of the technical system are symbolization, hierarchy of symbols, and the valuation system. The reason for choosing this approach is that the ARS-PC model is the basis for the ARS-PA (Artificial Recognition System - PsychoAnalysis) model, which implements the higher cognitive functions of the brain. Together, the ARS-PC and ARS-PA parts should form a unitary model which, like the human brain, is able to fulfill the functions of receiving sensor data from the periphery to the high-level decision-making process.

In order to fulfill the task of the ARS-PC part, i.e. the recognition of people, objects and scenarios, this work proposes a framework, which includes a database for storing sensor values, the ARS symbol tree, SymbolNet as communication protocol for symbols and the definition of symbols in XML, and a symbol database for storing perceived symbols. Around this framework, different kinds of data provisions and visualizations are presented. For the provision of data, a concept of abstracting sensor values in different fieldbus systems is introduced, which is suitable to be the basis of the ARS system. In the showcase of the kitchen, a direct approach was chosen in order to collect the sensor values. A specially developed simulator for scenario simulation is the third possibility for the provision of data. Two-dimensional and three-dimensional visualizations can be used to visualize the results of the work presented here.

In order to demonstrate the concept of the model presented here, a showcase in the kitchen of the ICT (Institute of Computer Technology of the Vienna University of Technology) was installed. In this showcase, several scenarios, as defined in this work, can be detected. Additional to the real implementation, scenarios have been simulated. The simulated sensor values can be used as an input for the ARS-PC system. The results of the test, as described in the previous chapter, showed that the approach proposed in this work is suitable for the scenario detection in buildings. The defined scenarios can be detected in the real implementation and in the simulation. During the tests in the real implementation, it turned out that especially for the people tracking application the positions of sensors is of high importance, e.g. near the door is essential to obtain accurate data on whether a person is leaving the room or not. Therefore, the outlook of this work lists improvements for the system which are partly related to the model and partly to the implementation and the used sensors.

Although in both the simulator and the real implementation a lot of sensor and sensor types are simulated or installed, respectively, a lot of sensor types are of course still missing. Beside the vision and audio system as described below for example smell or smoke detectors could be integrated, especially if the system is used with safety applications. Nevertheless, all sensors I have mentioned so far are used to obtain information about the outer world, the environment. But for us human beings the inner world also plays a major role for our perception of the world as described in Chapter 2.1. The inner world is practically ignored by the current model and its implementation. The reason for this is that we need a body or at least the simulation of a body that can have an inner world. In [Pra06], it is proposed to see the hardware (network, sensors, CPU, etc.) as the inner world of a technical system. However, it is also stated there that in fact the system itself cannot satisfy the needs of the hardware itself. Therefore, it is proposed to see maintenance (by a human operator) as the satisfaction of a need of a technical system. Another possibility is proposed in the Bubble Family Game (BFG) as presented in [Deu07]. Here, a software agent has to interact with its environment in order to get energy from energy sources.

But still, if such an approach is taken, it still would not solve how the outer world (environment) influences the inner world (body). One example would be the basic emotional systems. Although I presented a model for the technical implementation of the basic emotional systems, it is not integrated into the prototype presented in this work. As described in Chapter 2.2, the internal states of a basic emotional system are also determined by the sensor values from the inner and outer world. In a technical sense, this means that a relationship between sensor values and change of the emotional states exists. And that leads to the question: How do sensor values of the environment influence the internal world of a technical system? The seeking system of mammalians pushes the individual to observe the environment for objects which will satisfy its needs. In the simulation of the BFG, the bubbles have to do the same in order to survive. But in an environment like a kitchen or a building, a system usually does not have to take any actions to have enough energy or be maintained. However, robots, e.g. service robots for elderly people, have to take their needs into account in order to serve properly. Another possibility to use the evaluation system is proposed below for the installation of sensors.

9.2 Outlook

The implementations presented in this work are still in their prototype state, and therefore a lot of improvements and additions can be made. Some ideas of how the current implementation and ARS model can be enhanced are presented in the following.

Vision System: Currently, a master thesis is testing the integration of cameras into the ARS project. It is planned and partly implemented that the symbols *person* and *object*, and various symbols *item* (fridge, coffee machine, etc.) are also based on camera information and not defined statically, as proposed in this work. But the major innovation of this work is the integration of computer vision into the ARS system. The computer vision software has direct access to the ARS system with SymbolNet in order to improve both, the ARS system and the computer vision algorithms. The ARS system gets additional symbols of recognized objects and properties of symbols. The computer vision system uses the symbols that are depending on other sensors to increase accuracy. For example, the position of a person is (indirectly via different symbols) determined by the pressure sensors, light barriers, motion detectors, etc. Using the symbol *person* and its property *position* in the computer vision algorithms, only one camera can determine the height of a person. As described in Chapter 6.1.2, a person's height can be used to decide whether the person is a child or an adult. Another enhancement is the calibration of the cameras. For example, objects like the door, which are represented with the symbol *door*, are known objects in the building. Besides its position also the shape and possibly even the color of the door are known. Using this information, the position of the camera can be determined in case a known object is detected by the computer vision algorithm.

Also in this work, the scenario *person taking reading material from a shelf* as described in Chapter 4.9 will be implemented. A screenshot of the two-dimensional visualization of this scenario is shown in Figure 104. Additionally, the scenario *meeting* is detected. A camera is used to observe the bookshelf. After a person, represented with the symbol *person*, has been detected in the vicinity of the bookshelf, and the person has again disappeared, computer vision algorithms help to detect whether a journal was taken from the bookshelf or not. On left side of Figure 104, the three layers of the visualization are shown. On the first layer, active sensors and the recognized microsymbols are shown. The screenshot shows the microsymbols *object* and *motion*. In the snapshot symbol layer the objects are shown. In the kitchen, the special objects are fridge, coffee maker, stove, and door. The other objects like table, bookshelf and copier are shown as colored shapes. Since the people do not move, no gait or footsteps are detected. In the third layer, the representation and scenario symbols are shown. In the Figure 104 two people are recognized. Since they sit at the table, the scenario *meeting* is active. In the bookshelf, the symbol for the missing journal can be seen, which was generated by the computer vision.

On the right side, the visual output of the vision system is depicted. In the upper left picture, the bookshelf is shown as it was before a person approached it. As the person left the bookshelf, the picture on the upper right side was taken. The information of a person being near the bookshelf is taken from the ARS-PC system. The picture on the lower left side is the result of the subtraction of the first two pictures.

The difference is the missing magazine. On the lower right side, the position of the magazine is detected, shown by the red square in the magazine. The blue squares visualize the positions where no pictures have been taken.

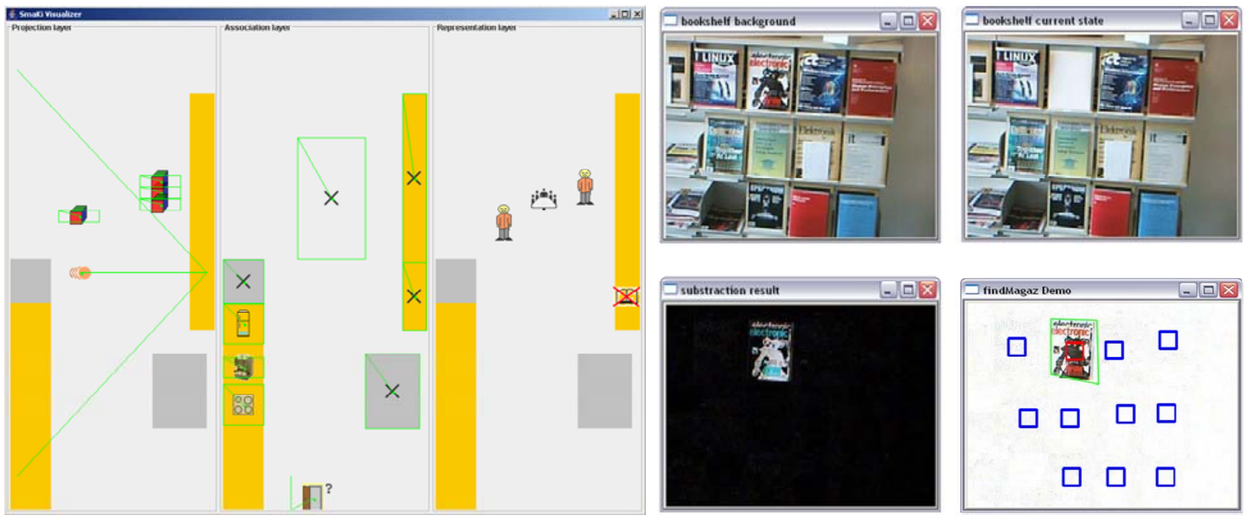


Figure 104: Meeting scenario and person tales reading material scenario

Similar to the scenario described above, the scenario *person puts down an object* can be improved. This scenario is detected when a person puts down an object in a room, leaves the room and the object remains in the room, as described in Chapter 4.3. With computer vision algorithms, both the person and the object can be detected and hence the symbol *person* and symbol *object* can be detected or the accuracy of the object's properties can be increased. However, all applications defined in Chapter 4 except the audio system self-test (see Chapter 4.10) can be improved by using additional information from one or more cameras in the room.

Audio System: As with the vision system described above, an audio system could be integrated into the ARS system. A number of (micro-) symbols could be generated on the basis of audio data. An example would be the symbol *fall* (see Chapter 6.3.2). But also additional symbols like a symbol *scream* or a symbol *wail* could be introduce in order to detect an accident happening in the room. Even more complex than the detection of screaming or wailing would be the recognition of spoken language. The question whether it is feasible to integrate language recognition into the ARS system should be the topic of further research work.

In principle it is also imaginable that the integration of the ARS system into a speech recognition system could improve the results of the ARS system in general and speech recognition in particular. This could be done in two ways. The ARS system can be used to identify the speaker and his or her position. This information can be used to adjust the parameters of the speech recognition system. The second way to improve speech recognition cannot be done with ARS-PC alone but needs also the semantic and episodic memory of the ARS-PA system. As described in [Die07], the generation of symbols in the human brain is not only based on the sensor input itself, but also on our knowledge and expectations of what we hear. So the idea of this particular item of improvement is that depending on the situation the speech recognition systems are adopted. This principle also applies to all other sensor types, but especially to the vision systems.

Installation of sensors: One of the problems of integrating huge amounts of sensors into a building is their commissioning. In the current implementation this problem is solved by using a sensor database. Beside the values of the sensor, the position and the active area of every single sensor are stored. This solution is suitable for our current (prototype) implementation. But when using a lot more sensors (e.g. up to ten thousand per room) this solution is no longer suitable, at least not in case the configuration data cannot be filled into the database automatically. For movable sensors e.g. mounted on movable objects, the database has to be updated permanently.

One possible solution could be that the system "learns" the configuration of its sensors. It is assumed that sensors, which are activated simultaneously, have a similar active area. To improve this method, the evaluation system can be used. Thus, not only sensors which are activated simultaneously are associated but also sensors which have similar individual e-status. For moveable sensors, tracking mechanisms have to be implemented in addition to object tracking sensors.

Another possibility would be to combine the position and active area information with the value data of the sensor. Therefore, at least each moveable sensor has to determine its position. For fixedly mounted sensors, the position could be configured. In order to reduce the bandwidth use, the position could be used as the address of the communication protocol. A communication protocol which uses the geographical position of the nodes as address would not only minimize the configuration effort of the network, but also the routing and services of the network. Typically, a router uses routing tables in order to decide how to route a packet through a network. Using geographical addresses, these tables can be omitted because the decision in which direction a packet should be routed can be easily calculated. A request of what the temperature at position X is or how many people are in room Y would then have only the position as address for the request. All nodes which have the requested information would answer the request. The network would automatically find the addressed sensors, even if the sensors are moveable. Such a scenario is illustrated in Figure 105.

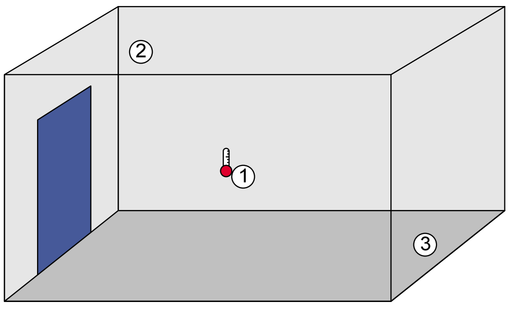


Figure 105: Active region of a temperature sensor

A temperature sensor is located in a room at position (1). Assuming that the temperature is equal in the room, the active area of the sensor is defined as the room. If an application would request the current temperature at position (2) or (3), the node of the sensor (1) would respond. Assuming that in the room there is not only one temperature sensor but e.g. 10 or even more, all sensors which have the active area at the point or area of request will respond. And that shows another advantage of the geographical addresses for communication systems. It offers at least a kind of redundant and fault tolerant system. Analyzing the responds of the sensors, the system can decide which sensors are out of order or respond with wrong values.

Introducing learning: One of our most exiting abilities as human beings, at least in a technical sense, is the ability to learn. In this context, learning should be understood as the forming of new symbols and the dependencies between the symbols. The presented model in this work has no ability to develop new symbol types. Every symbol and their dependencies are predefined. However, in [Lur73] it is described how the learning of symbols is achieved by people. It has been shown that the secondary fields (see Chapter 2.1.2) could not be formed properly without the integrity of the primary zones during the ontogeny. The same applies to the tertiary zone. In the ARS-PC system this means, that first the microsymbols have to be derived from the sensor values. Then the snapshot symbol layer has to be formed and so on. As mentioned above, hardwired evaluation systems could play a major role in that learning process. Sensors which have the same active area would be simultaneously active in case of changes in that area. This could be a way how microsymbols could be learned. A possible way to learn the associations is shown below.

Different view on symbols: In this work, the symbols have been described as data structures which are determined by their properties. The association between symbols can be done in various ways. As an example, let us consider the symbol *person upright* (see Chapter 6.3.6). To associate the symbol *upright* with the symbol *person* the spatial relationship is used. This means that symbols with similar properties are associated with each other. In other words, similar properties of symbols mean that the symbols have a relationship. This does not only apply to the position but to all properties. Symbols with the properties which have the same or similar values are associated with each other. Thus, in a system like ARS-PC the relationship between the systems could not only be established with the association of existing symbols with new ones as proposed in this work, but additionally, the system could itself could associate symbols by observing their properties. In Figure 106 shows some examples.

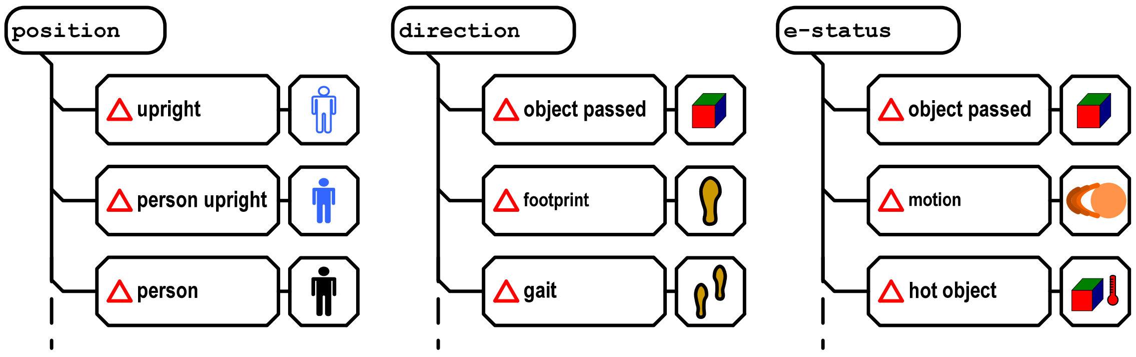


Figure 106: Properties and associated symbols

Beside position as one property of almost all symbols, the direction or velocity of movable objects are properties which can be used to associate symbols with each other. Another category of properties which might be used to associate properties might be the e-status of an e-system. In addition objects which cause a similar e-status are associated with each other. As proposed above, this could be used to associate sensors which each other.

Focus of Attention: In the current state of the project, a typical PC has enough computational resources for the manipulation of the defined symbols. However, assuming a much higher number of sensors connected to such a system and a larger number of scenarios that can be detected, the required computational power would rapidly increase unless measures are taken against it. One measure could be to introduce a focus of attention. Depending on the current situation (including inner and outer world) of the system, parts of the symbol trees could be activated or disabled. This might be done with simple rules, e.g. if a dangerous situation is detected, the comfort functions shall be reduced to the necessary minimum. Another way could be to use fuzzy logic or similar algorithms to take into account more factors and get graduated results for switching off parts of the symbol tree. However, the detection of dangerous or essential situations, events, or objects should never be disabled.

References

Add97	Addlesee, M. D. et al. “ORL Active Floor”, IEEE Personal Communications, Vol.4, No.5, October 1997, pp. 35-41
ALW03	Alwan M, Dalal S, Kell S, Felder R. Derivation of Basic Human Gait Characteristics from Floor Vibrations. Summer Bioengineering Conference, Sonesta Beach Resort in Key Biscayne, Florida. June 25-29 2003
Arb95	Arbib A. The Handbook of Brain Theory and Neural Networks. Massachusetts Institute of Technology, 1995. ISBN 0-262- 01148-4.
ASH135	ASHRAE 135-1995] ANSI/ASHRAE Standard 135-2004 – BACnet – A Data Communication Protocol for Building Automation and Control Networks. American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc., 2004.
Ayy06	B. M. Ayyub, G. J. Klir; Uncertainty Modeling and Analysis in Engineering and the Sciences; Chapman & Hall/CRC, Taylor & Francis Group; 2006
BAC04	Müller Ch. BACnet in Berlin Parliament Buildings’ Technical Network. BACnet Europe Journal Volume 01. Sep 2004. Pages 14 – 15
Bal05	A.O. Balan, L. Sigal, M.J. Black, “A quantitative evaluation of video-based 3D person tracking” in: Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. pp: 349- 356 ISBN: 0-7803-9424-0
Bia04	Bian X. Abowd G.D. Rehg J.M. Using Sound Source Localization to Monitor and Infer Activities in the Home. GVU Technical Report; GIT-GVU-04-20. 2004
Bon92	Bonfig, Karl Walter – Fuzzy Logik in der industriellen Automation. Expert Verlag 1992, ISBN 3-8169-0932-9
Bru07	Bruckner D. Probabilistic Models in Building Automation: Recognizing Scenarios with Statistical Methods. Dissertation an der Fakultät für Elektrotechnik der Technischen Universität Wien, 2007
Bur04	W. Burgstaller, T. Rausch. Gatewayless communication between LonWorks and BACnet. Proceedings of 2004 IEEE International Workshop on Factory Communication Systems. P. 361-364

Bur05 W. Burgstaller, S. Soucek, P. Palensky: Current challenges in abstracting datapoints. In: “Proceedings of 6th IFAC International Conference on Fieldbus Systems and their Applications”, (2005), S. 40 – 47.

Bur07 Burgstaller W., Lang R., Poerscht P., Velik R. Technical Model for Basic and Complex Emotions. In proceedings of 5th IEEE International Conference on Industrial Informatics 2007, Vienna pp 1033-1038

Cal03 Callan R. Neuronale Netze im Klartext. Pearson Studium, 2003. ISBN 3-8273-7071-X.

Cal03 I. Calvo, M. Marcos, D. Orive, A. Huidobro, “Object-Oriented Based Architecture for Accessing Remotely Electrical Protection Relays,” in Proc. IEEE Intern. Conf. on Emerging Technologies and Factory Automation, Lisbon, 2003, vol. 2, pp. 614-621.

Cel91 F.E. Cellier; Continuous System Modeling, Chapter 13 “Inductive Reasoning”; Springer-Verlag, New York; 1991

CHA04 H. Nait Charif and S. J. McKenna. Activity summarisation and fall detection in a supportive home environment. In Int. Conf. On Pattern Recognition (ICPR), 2004.

Dam99 Damasio, A.R. The feeling of what happens. Harvest Books. 2000

DER02 International Organization for Standardization. ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER) (ISO/IEC 8825-1:2002), Geneva, Switzerland, 2002

Deu06 T. Deutsch, R. Lang, G. Pratl, E. Brainin, S. Teicher. Applying Psychoanalytic and Neuro-Scientific Models to Automation. In: Proceedings of the 2nd IET International Conference on Intelligent Environments, Vol. 1, 2006, S. 111 – 118.

Deu07 Deutsch T., Zeilinger H., Lang R. Simulation Results for the ARS-PA Model. In proceedings of INDIN 2007 to be published

Die00 Dietrich D. Evolution potentials for fieldbus systems in Proceedings of the WFCS2000. Porto, Portugal; Sep. 2000. Pages 145 – 146.

Die01a Dietrich D., Fischer P., LonWorks Planerhandbuch. LNO Nutzer Organisation e. V. VDE Verlag 2001

Die01b Dietrich D., Russ G., Tamarit C., Koller G., Ponweiser W., Vincze M., Modellierung des technischen Wahrnehmungsbewusstseins für den Bereich Home Automation. Elektrotechnik und Informationstechnik (e&i), vol. 118, no. 11, 2001, pp. 545 – 555.

Die07 Dietrich D., Fodor G., Kastner W., Ulieru M. Technical transformation of the neuro-psychoanalytical models. In proceedings of ENF 2007, to be published

DiS00 Di Stefano, A.; Bello, L.L.; Bangemann, T. Harmonized and consistent data management in distributed automation systems: the NOAH approach. Proceedings of the 2000 IEEE International Symposium on Industrial Electronics, (ISIE 2000), p 766 – 771

Dou00 Doughty K. Fall Prevention & Management Strategies Based on Intelligent Detection, Monitoring & Assessment. New technologies in medicine for the elderly. Charing Cross Hospital. London, November 2000

EIA709 ANSI/EIA/CEA-709.1 Control Network Protocol Specification, Electronic Industry Alliance, Rev. B, 2002.

EIA852 Tunneling of Component Network Data Over IP Channels, EIA-852 Draft, Apr. 2000.

EN14908-1 Open Data Communication in Building Automation, Controls and Building Management – Control Network Protocol – Part 1: Protocol Stack. Brussels: CEN, EN 14908-1:2005

EN14908-2 Open Data Communication in Building Automation, Controls and Building Management – Control Network Protocol – Part 2: Twisted Pair Communication. Brussels: CEN, EN 14908-1:2005

EN50170 “General purpose field communication system,” CENELEC, Brussels, Belgium, EN 50170, 2002.

Fis99 Fischer P. Mapping of Fieldbus Protocols to Standardised Field Level Objects, in D. Dietrich, P. Neumann and H. Schweinzer (eds.): System Integration, Networking and Engineering, Springer, p. 54-52

Fre03 Freud, S. Vorlesungen zur Einführung in die Psychoanalyse. Frankfurt am Main S. Fischer Verlag 2003.

Goe06 Götzinger S. Scenario Recognition Based on a Bionic Model for Multi-Level Symbolization. Diplomarbeit an der Fakultät für Elektrotechnik der Technischen Universität Wien, 2006

Ham05 A. Hampapur, L. Brown, J. Connell, A. Ekin, N. Haas, M. Lu, H. Merkl, S. Pankanti, “Smart video surveillance: exploring the concept of multiscale spatiotemporal tracking” in: Signal Processing Magazine Volume: 22, Issue: 2, IEEE March 2005 pp: 38- 51 ISSN: 1053-5888

Har07 Hareter H. Worst Case Szenarien Simulator für die Gebäudeautomation. Dissertation an der Fakultät für Elektrotechnik der Technischen Universität Wien, to be published

Heg00 Hegewald, G., „Ganganalytische Bestimmung und Bewertung der Druckverteilung unterm Fuß und von Gelenkwinkelverläufen – eine Methode für Diagnose und Therapie im medizinischen Alltag und für die Qualitätssicherung in der rehabilitationstechnischen Versorgung“, Dissertationsschrift an der Humboldt-Universität zu Berlin (2000)

I2C Philips Semiconductors, The I2C-Bus Specification, Version 2.1, January 2000; document order number: 9398 393 40011, available at http://www.nxp.com/acrobat_download/literature/9398/39340011.pdf

IEC1131-7 International Electrotechnical Commission (IEC) Technical Committee No. 65: Industrial Process Measurement And Control Sub-Committee 65 B: Devices; IEC 1131 - Programmable Controllers Part 7 - Fuzzy Control Programming; Committee Draft CD 1.0 (Rel. 19 Jan 97)

IEEE754 ANSI/IEEE Standard 754-1985, "IEEE Standard for Binary Floating-Point Arithmetic."

ISO1000 ISO 1000. SI units and recommendations for the use of their multiples and of certain other units. ISO 1000:1992

ISO14543 Information technology – Home electronic system (HES) architecture – Part 3-4: System management – Management procedures for network based control of HES Class 1. ISO/IEC 14543-3-4:2007

ISO16484-2 Building Automation and Control Systems (BACS) – Part 2: Hardware. International Standards Organisation, ISO 16484-2, 2003.

ISO16484-5 Building Automation and Control Systems (BACS) – Part 5: Data communication. International Standards Organisation, ISO 16484-5, 2003.

ISO8802 Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks, International Standards Organisation, ISO 8802, 2000.

Jac99 Jackson, Peter – Introduction to Expert Systems; Addison Wesley Longman Limited; Edinburgh 1999; ISBN 0-201-87686-8

Kab02 K. Kabitzsch, D. Dietrich, G. Pratl (Editors). LonWorks – Gewerkeübergreifende Systeme. VDE-Verlag, Germany, 2002

Kas04 Kastner, W. and Neugschwandtner, G. Service interfaces for field-level home and building automation, Proceedings. 2004 IEEE International Workshop on Factory Communication Systems (WFCS2004), p.103-112

Kra00 Krause, R. Affekt, Emotion, Gefühl. In: Handbuch Psychoanalytischer Grundbegriffe. 2000. Hrsg.: Merten, W. und Waldvogel, B. Stuttgart (Kohlhammer)

Kru95 Kruse, Rudolf; Gebhart, Jörg; Klawonn, Frank – Fuzzy-Systeme; (2. Auflage), Teubner Verlag 1995; ISBN: 3519121301

LIGA05 LonMark Application-Layer Interoperability Guidelines. Echelon Corporation, Version 3.4, 2005.

LIGP05 LonMark Layer 1-6 Interoperability Guidelines, Echelon Corporation. Version 3.4, 2005.

Lob02 M. Lobashov, G. Pratl, T. Sauter, "Applicability of Internet Protocols for Fieldbus Access," in Proc. IEEE Intern. Workshop on Factory Communication Systems, Västerås, 2002, pp. 205-213.

Lob06 M. Lobashov and T. Sauter, Vertical Communication from the Enterprise Level to the Factory Floor – Integrating Fieldbus and IP-based Networks, 11th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Praha, 20.-22. Sep. 2006

Loy01	D. Loy, D. Dietrich, H. J. Schweinzer: Open Control Networks-LonWorks/EIA 709 Technology. Kluwer Academic Publishers, 2001.
LSML03	LonMark SNVT Master List, Echelon Corporation, Version 12, June 2003.
Lur73	Aleksandr Romanovich Luria. Working Brain: An Introduction to Neuropsychology Basic Books; June 1973
Mah06	A. Mahdavi, B. Spasojevic: Energy-Efficient Lighting Systems Control via Sensing and Simulation; in: 17th Air –Conditioning and Ventilation Conference 2006 – May 17-19, 2006, Prague, Czech Republic, J. Schwarzer, M. Lain (editor); Prague, Czech Republic, 2006, pages. 205 – 210.
Mel99	Mella A. and Russo F. NOAH Project, an Example of Application, in D. Dietrich, P. Neumann and H. Schweinzer (eds.): System Integration, Networking and Engineering, 1999, Springer, pp54-52
Mot97	LonWorks Technology Device Data, Motorola Inc, 1997
Orr00	Orr R. and Abowd G. The smart floor: A mechanism for natural user identification and tracking. In Conference on Human Factors in Computing Systems, The Hague, Netherlands, April 2000
Ort90	Ortony A., Turner T.J. What's basic about basic emotions? Psychological Review: 97 (3). 1990. 315–331.
OSI94	ISO/IEC 7498-1, Information technology – Open System Interconnection – Basic Reference Model: The Basic Model. Second Edition. 1994-11-15
Pal03	P. Palensky, P. Rössler, D. Dietrich: „Heim- und Gebäudeautomatisierung zur Effizienzsteigerung in Gebäuden"; Elektrotechnik und Informationstechnik (e&i), 4 (2003)
Pan98	Panksepp J. Affective Neurosciences: The Foundations of Human and Animal Emotions. Oxford University Press New York, 1998
Pra06	Pratl G. Processing and Symbolization of Ambient Sensor Data. Dissertation an der Fakultät für Elektrotechnik der Technischen Universität Wien, 2006
Pra07	Pratl, G.; Dietrich, D.; Hancke, G. P.; Penzhorn, W. T. A New Model for Autonomous, Networked Control Systems. IEEE Transactions on Industrial Informatics, Volume 3, Issue 1, Feb. 2007 pages 21 – 32
prEN15232	Energy performance of buildings – Impact of Building Automation, Controls and Building Management. CEN prEN 15232. Draft Version Feb. 2007
RFC791	Internet Protocol, J. Postel. September 1981. available at: http://www.ietf.org/rfc/rfc 791.txt
Ric07	Richtsfeld A. Szenarienerkennung durch symbolische Datenverarbeitung mit Fuzzy-Logic. Diplomarbeit an der Fakultät für Elektrotechnik der Technischen Universität Wien, 2007

Roe06	Ch. Rösener, B. Lorenz, G. Fodor, and K. Vock. Emotional behaviour arbitration for automation and robotic systems. Proceedings of 2006 IEEE International Conference of Industrial Informatics, 2006.
Roj96	Rojas, R. Theorie der neuronalen Netze – Eine systematische Einführung. Springer Lehrbuch, 1996. ISBN 3-540-56353-9.
Rus03	Russ G. Situation-dependent behaviour in building automation. Dissertation an der Fakultät für Elektrotechnik der Technischen Universität Wien, 2003
Sau04	T. Sauter, Linking Factory Floor and the Internet, in: R. Zurawski (Ed.), The Industrial Information Technology Handbook, CRC Press, 2004, pp. 52-1 – 52-15.
Sau05	T. Sauter, Integration Aspects in Automation – a Technology Survey, 10th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Catania, 19.-22. Sep. 2005, vol. 2, pp. 255-263.
Sau07	T. Sauter, The Continuing Evolution of Integration in Manufacturing Automation, IEEE Industrial Electronics Magazine, vol. 1, no. 1, Spring 2007
Sch97	Scherer A.. Neuronale Netze – Grundlagen und Anwendungen. Vieweg Verlag, 1997. ISBN 3-528-05465-4.
Schö04	Schötz S. Some acoustic cues to human and machine estimation of speaker age. Proceedings. FONETIK 2004. Dept. of Linguistics, Stockholm University 2004
Sol02	Solms M. and Turnbull O. The brain and the inner world. Other Press LLC New York, 2002.
SQL92	Database Language SQL, International Standard ISO/IEC 9075:1992
Sut03	Suter G., Mahdavi A., Ries R. Outline of a system for sensor-driven, high-resolution building models. Proceedings of the 20th CIB W78 Conference on Information Technology in Construction 2003, Waiheke Island, Auckland, New Zealand. Pp. 1 – 9.
Tai05	Taisuke H. Mineichi K. Person Tracking with Infrared Sensors. Lecture Notes in Computer Science. Volume 3684. Aug 2005. Pages 682 – 688
Tam03	Tamarit Fuertes C. Automation System Perception First Step towards Perceptive Awareness. Dissertation an der Fakultät für Elektrotechnik der Technischen Universität Wien, 2003
TAN02	Andrew S. Tanenbaum, Computer Networks, Prentice Hall PTR; 4 edition August 9, 2002, ISBN-10: 0130661023
Vel03	Velasco, M., Fuertes, J.M. and Marti, P. Process data abstraction/accessibility via internet, in Proceedings of the IEEE Conference on Emerging Technologies and Factory Automation (ETFA 2003), p. 357 – 363

Wol01 M. Wollschlaeger, “Framework for Web Integration of Factory Communication Systems”, IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Antibes Juan-les-Pins, 15.-18. Oct. 2001, pp. 261-265.

Wol02 M. Wollschlaeger, C. Diedrich, T. Mangemann, J. Mueller, U. Epple: “Integration of Fieldbus Systems into On-line Asset Management Solutions based on Fieldbus Profile Descriptions”, in Proceedings of the 4th IEEE International Workshop on Factory Communication Systems (WFCS), 2002, pp. 89-96.

X.680 Abstract Syntax Notation One (ASN.1) Specification of Basic Notation. ITU-T Rec. X.680 (2002) ISO/IEC 8824-1:2002

X.693 ITU-T Recommendation X.693, Information technology – ASN.1 encoding rules: XML encoding rules (XER), OSI networking and system aspects – Abstract Syntax Notation One (ASN.1), Series X: Data networks and open system communications, Prepublished version at http://www.itu.int/ITU-T/studygroups/com17/languages/X.693-0112.pdf, December 2001.

Internet References

ARS06	The ARS project website, http://ars.ict.tuwien.ac.at, accessed on July 20th, 2006
BBP06	Homepage of the Blue Brain project, http://bluebrainproject.epfl.ch/index.html, accessed on 30th June, 2006
Cco06	Homepage of the CCortex project, http://www.ad.com/, accessed on 30th June, 2006
ENF	1st international Engineering & Neuro-Psychoanalysis Form, July 23, 2007, Vienna, http://www.indin2007.org/enf/
FSFE	Free Software Foundation Europe. http://www.germany.fsfeurope.org/
Gray728	Henry Gray (1825–1861). Anatomy of the Human Body. 1918. FIG. 728 Principal fissures and lobes of the cerebrum viewed laterally. This is a plate from Gray's Anatomy. Unless stated otherwise, it is from the online edition of the 20th U.S. edition of Gray's Anatomy of the Human Body (http://www.bartleby.com/107/illus728.html), originally published in 1918. This image in the public domain because it's copyright has expired. This applies worldwide.
INDIN	5th IEEE Conference on Industrial Informatics 2007, http://www.indin2007.org/
JFL	JFuzzyLogic, Open Source Fuzzy-Logic, http://jfuzzylogic.sourceforge.net
LC3020	Data sheet of the LC3020, Loytec; Available http://www.loytec.com/assets/contents/pdf/DataSheet_LC3020.pdf
n-psa	Homepage of the International Neuro-Psychoanalysis Centre and Society, http://www.neuro-psa.org.uk/ accessed February 2007
NPSA07	8th International Congress of Neuro-Psychoanalysis: Neuro-Psychoanalytic Perspectives on Depression N-PSA 2007 http://www.neuro-psa.org.uk/npsa/index.php?module=pagemaster&PAGE_user_op=view_page&PAGE_id=48
oBIX05	oBIX, Open Building Information Exchange. Available: http://www.obix.org, accessed February 2007

OCV06	OpenCV Library Wiki. http://opencvlibrary.sourceforge.net, accessed on June 30th, 2006.
OPC05	OPC Foundation: http://www.opcfoundation.org, accessed February 2007
TC247	Comité Européen de Normalisation / Technical Committee 247; Building Automation, Controls and Building Management
TOP500	Homepage of the TOP 500 supercomputers, http://www.top500.org/, accessed on 4th November, 2006
VSF06	Vorwerk präsentiert RFID "smart floor" auf der CeBIT, http://www.vorwerk-teppich.de/vorwerk_cebit_2006_rfid.html, accessed on June 30th, 2006
W3C	Homepage of the World Wide Web Consortium W3C, http://www.w3.org/
XPath	W3C, XML Path Language (XPath) Version 1.0, http://www.w3.org/TR/xpath
XSL	The Extensible Stylesheet Language Family (XSL), http://www.w3.org/Style/XSL/
XSL-FO	W3C, Extensible Stylesheet Language (XSL) Version 1.1, http://www.w3.org/TR/xsl/
XSLT	W3C, XSL Transformations (XSLT) Version 1.0, http://www.w3.org/TR/xslt