

DISSERTATION

Experience-Based Decision-Making in a Cognitive Architecture

From Perception through Expectations to Anticipatory Decisions

Submitted at the
Faculty of Electrical Engineering and Information Technology,
TU Wien
in partial fulfillment of the requirements for the degree of
Doktor der technischen Wissenschaften

under supervision of

Prof. Dr. Dietmar Dietrich
Institute number: 384
Institute of Computer Technology
TU Wien

and

Prof. Yevgen Biletskiy, Ph.D.
Department of Electrical and Computer Engineering
University of New Brunswick, Canada

by

Dipl.-Ing. Alexander Wendt
Matr.Nr.: 0226557
Martinstraße 59/11
A-1180 Vienna

Vienna, 13.08.2016

Kurzfassung

Unsere Handlungen basieren vor allem auf gewonnenen Erfahrungen. Die Fähigkeit Situationen wieder zu erkennen und vorausschauend daraus Entscheidungen zu treffen, ist ein wesentlicher Bestandteil unserer Intelligenz. Softwareagenten, die in komplexen Umgebungen mit vielen Freiheitsgraden operieren, brauchen menschenähnliche Funktionalität, um mit den Anforderungen der Umgebungen zurechtzukommen. Die kognitive Architektur SiMA (Simulation of the Mental Apparatus & Applications) stellt menschenähnliche Fähigkeiten zur Verfügung. Verglichen zu anderen kognitiven Architekturen, liegt die Stärke von SiMA im ganzheitlichen Ansatz, der auf der Theorie der Psychoanalyse basiert. Daher werden viele Bereiche der menschlichen Funktionalität gedeckt, die durch andere kognitive Architekturen nicht behandelt werden. Allerdings existiert in der aktuellen Implementierung keine Möglichkeit Erfahrungen zu bearbeiten, denn die Architektur unterstützt derzeit nur die Reaktion auf unmittelbare Sensorinputs. Diese Arbeit implementiert erstmals das Konzept der konkreten Erfahrungen und der zeitlichen Dimension in SiMA, um Sequenzen von Ereignissen zu verarbeiten. Statt rein reaktiv auf Sensorinputs zu antworten, wurde eine Entscheidungsfindung ermöglicht, um langfristige Entscheidungen treffen zu können. Dies erlaubt den Agenten etwas zu erwarten und vorausschauend zu handeln. Somit können Ziele erreicht werden, die sich nicht in der unmittelbaren räumlichen oder zeitlichen Nähe des Agenten befinden. Die Umsetzung dieser Erweiterung erfolgt in einem Simulator und wird durch definierte Anwendungsfälle evaluiert. Erste Anwendungen sind als Individuen in Wirtschaftssimulationen im Projekt CogMAS (Cognitive Multi-Agent System Supporting Marketing Strategies of Environmental-Friendly Energy Products) und als Steuerungsmodul in der Gebäudeautomation im Projekt KORE (Cognitive Control Strategy Optimization for Increasing Energy Efficiency in Buildings). Möglichen weiteren Anwendungen in der Zukunft in z. B. autonomen Robotern sind nur die Grenzen der Vorstellungskraft und Ethik gesetzt. Die Einführung und Nutzung von Erfahrungen als Entscheidungsgrundlage macht die ersten Applikationen möglich, aber bringt auch SiMA ein Schritt näher an einen wesentlichen Bestandteil der Intelligenz.

Abstract

Human interaction with the environment is mainly based on gained experiences. The ability to recognize and make decisions based on memorized situations is a vital feature of human intelligence. Software agents that operate in complex environments with a high degree of freedom need human-like functionality, in order to operate in such environments. The cognitive architecture Simulation of the Mental Apparatus & Applications (SiMA) provides such human-like capabilities. Compared to other cognitive architectures, the edge of SiMA can be found in the holistic approach, which is based on the theory of psychoanalysis. It covers several areas of the human mind, which is not covered by other architectures. In the current implementation, no possibility to process experiences, because the current architecture only supports the immediate processing of sensor values. The goal of this work is to provide functionality for decision-making to be able to use the temporal dimension too by using own experiences. It allows the system to have expectations and to act proactively. In that way, system goals can be reached, which are located in some spatial and temporal distance. The system is implemented in a simulator and it is validated through defined use cases. First applications are as individuals in economic simulations in the project CogMAS (Cognitive Multi-Agent System Supporting Marketing Strategies of Environmental-Friendly Energy Products) as well as a building automation controller in the project KORE (Cognitive Control Strategy Optimization for Increasing Energy Efficiency in Buildings). Possible future applications like in autonomous robots are only limited through our visions and ethical rules. Through the introduction and utilization of experiences in decision-making, the first applications are available and it also takes SiMA a step closer to human-like intelligence.

Acknowledgements

First of all, I would like to thank my supervisor Prof. Dr. Dietmar Dietrich, who always stood behind the project. He is a great source of motivation and worked intensive with each of his students to get the best out of them.

Then, I would like to thank my second supervisor Prof. Yevgen Biletskiy for his questions and comments of the work. They helped a lot to increase the quality.

Also, an important source of motivation and knowledge were my children Mia and Niklas. They were a great source of inspiration as I could playfully use them as experiment objects and observe how the theory of the mind applies in reality. Especially children between one and four years old are good to observe because they do not hide their intentions behind the socially adapted behavior. Instead, they express what they perceived and process.

Thanks to my wife Judith, who always held out to listen to me explaining complex concepts in a complicated way. She gave me good hints in how to simplify the explanations, in order to let other people, understand them too.

Finally, thanks to my colleagues of the SiMA group and the project managers Dipl.-Ing. Samer Schaat and Dipl.-Ing. Dr. Dietmar Bruckner for managing the project with high motivation and keeping it alive for so long. We had many interesting discussions about the human psyche and AI and learnt how to work in an interdisciplinary way. For the psychoanalytic support, I would like to thank Dr. Klaus Doblhammer and Dr. Gyuri Fodor. They showed great patience at the translation of psychoanalytical concepts into technical concepts.

Permission

This work including all pictures that have been created for it can be distributed and used in other publications without explicit permission of the author (no asking per mail necessary) as long as the origin is clearly stated. Any text or pictures that have been taken from other publications with permission of their authors and are explicitly referenced are not included in this permission.

Table of contents

1. Introduction	1
1.1 Background	1
1.2 Motivation	4
1.3 Problem Description	5
1.4 Task Setting	9
1.5 Methodology	10
2. State of the Art and Related Work	12
2.1 Evaluation Criteria for Cognitive Architectures	12
2.1.1 Cognitivist, Emergent and Hybrid Architectures	13
2.1.2 Characteristics of Cognitivist Architectures	15
2.2 Experience-Based Reasoning in Common Cognitive Architectures.....	21
2.2.1 Internal Representation of Perceived Information.....	21
2.2.2 Activation of Stored Content.....	22
2.2.3 Decision Process.....	24
2.2.4 Usage of Episodes	26
2.2.5 General Issues in Cognitive Architectures	27
2.3 SiMA - Simulation of the Mental Apparatus & Applications.....	27
2.3.1 General Model Theory	28
2.3.2 Tracks of the Functional Layers	30
2.3.3 Knowledge Storage of the Data Layers	31
2.3.4 Data Structures	32
2.3.5 The Cognitive Cycle.....	35
2.3.6 Concluding Remarks on SiMA.....	41
3. Models and Concepts	44
3.1 Extension of Data Structures for Temporal Processing	44
3.1.1 New Data Structures of the Primary Process.....	45
3.1.2 New Data Structures of the Secondary Process.....	51
3.2 Activation of Experiences in the Primary Process.....	55
3.2.1 Memory Retrieval in the Primary Process.....	56
3.2.2 Image Matching.....	58
3.2.3 Memory Retrieval Interface.....	59
3.2.4 Psychic Spreading Activation.....	61
3.2.5 Possibilities with Psychic Spreading Activation	66
3.3 Evaluation of Experiences in the Secondary Process	68
3.3.1 Usage of Experiences in Decision-making.....	69
3.3.2 Translation of Bionic Concepts into Technical Functions	70
3.3.3 Reasoning with Acts.....	75

3.4 Deliberative Decision-Making	76
3.4.1 Requirements on Decision-Making	76
3.4.2 Preparation of Primary Process Data Structures	79
3.4.3 Process of Making a Decision	81
3.4.4 Short-Term Memories of the Secondary Process	84
3.4.5 Functionality Controlled by Internal Actions	88
4. Implementation.....	91
4.1 Evaluation in the SiMASin World Simulator	91
4.1.1 Simulator Setup	91
4.1.2 SiMASin World.....	92
4.2 Highlights of the Implemented Cognitive Architecture SiMA	94
4.2.1 Psychic Spreading Activation Image Matching Algorithm	94
4.2.2 Implementation of the Decision-Making Process.....	97
5. Simulation and Results	106
5.1 Test Environment.....	106
5.2 Verification of Data Structure Functionality.....	109
5.2.1 Test case 2.1: Internal Representation of the Primary Process.....	109
5.2.2 Test case 2.2: Internal Representation of the Secondary Process	110
5.3 Verification of Psychic Spreading Activation.....	112
5.3.1 Test case 3.1: Image Matching	112
5.3.2 Test case 3.2: Activation of Relevant Memories	113
5.3.3 Test case 3.3: Indirect Activation of Images	115
5.4 Verification of Deliberative Decision-making.....	116
5.4.1 Test case 5.1: Decision-making Process.....	116
5.4.2 Test case 5.2: Evaluation of Possible Goals	119
5.4.3 Test case 5.3: Three Decision-making Paths	120
5.4.4 Test case 5.4: Focus of Attention and the Environmental Image	123
5.5 Verification of Act Recognition.....	124
5.5.1 Test case 4.1: Extraction of Intention, Moment, and Expectation.....	124
5.5.2 Test case 4.2: Combination of Multiple Acts	126
5.6 Validation of Use case 1 Obstacle Modification.....	127
5.7 Utilization of this Implementation in Comprehensive Use cases.....	128
6. Conclusion.....	131
6.1 Conclusion	131
6.2 Future Work	138
6.3 Outlook.....	143
Literature	145
Internet/Film References	153
Curriculum Vitae of Alexander Wendt	154
Appendix A – Detailed Analysis of State of the Art.....	157

SOAR – State Operator Apply Result.....	157
ACT-R - Adaptive Control of Thought Rational	160
Icarus.....	163
CHREST - Chunk Hierarchy and REtrieval STRuctures.....	166
BDI - Belief, Desire, Intention.....	168
LIDA - Learning Intelligent Distribution Agent.....	171
Appendix B – SiMA Software Implementation	175
The SiMA Model of Abstraction Level 1	175
Data Structures.....	176
Thing Presentation Mesh and Word Presentation Mesh.....	176
Image and Act.....	178
General Data Structure Meshes	181
Implementation of Psychic Spreading Activation.....	182
Layered Memory Structure.....	182
Memory Retrieval by Psychic Spreading Activation	185
Implementation of Decision-Making on Abstraction Level 1	187
Codelets in Decision-making/Decision-Making in the SiMA Model.....	189
Codelet.....	189
Codelet Handler.....	191
Codelet Interaction with the SiMA Architecture.....	192
Appendix C – Example Calculations	194
Example of Activation of Template Images	194

Abbreviations

ACT-R	Adaptive Control of Thought Rational
AGI	Artificial General Intelligence
ARS	Artificial Recognition System
ARSi09	Artificial Recognition System version 9 implemented by R. Lang
ARSi10	Artificial Recognition System version 10 implemented by H. Zeilinger
ARSi11	Artificial Recognition System version 11, i.e. preliminary work, implemented by T. Deutsch
ARSi12	Artificial Recognition System version 12 implemented by C. Muchitsch
ARSi13	Artificial Recognition System version 13 implemented by I. Hinterleitner
BDI	Believe Desire Intention
CHREST	Chunk Hierarchy and REtrieval Structures
CogMAS	Cognitive Multi-Agent System Supporting Marketing Strategies of Environmental-Friendly Energy Products
ECABA	
JADE	Java Agent DEvelopment Framework
KORE	Cognitive Control Strategy Optimization for Increasing Energy Efficiency in Buildings
LIDA	Learning Intelligent Distribution Agent
MASON	Multi-Agent Simulator Of Neighborhoods
PRIINSTANCE	PRimary process Image Instance
SiMA	Simulation of the Mental Apparatus & Applications
SiMAi14	Simulation of the Mental Apparatus & Applications Implementation version 14 by Alexander Wendt
SOAR	State Operator Apply Result
SOAR SVS	SOAR Spatial Visual System
SPARQL	SPARQL Protocol And RDF Query Language
TPM	Thing Presentation Mesh
WPM	Word Presentation Mesh

1. Introduction

“För vetenskapen är ingenting omöjligt” [“For science, nothing is impossible.”]

[Dr. Busé in the movie “Jönssonligan och den svarta diamanten”]

Since ancient times, humans have created tools, which increase productivity, perform dangerous and monotonous tasks and serve our needs. With the industrial revolution, machines replace humans for monotonous tasks, which are characterized by a low degree of freedom like weaving or filling bottles with fluid. Such machines neither perceive their environment nor make any decisions. They are performing monotonous tasks. During the revolution of information technology, computers have been developed to handle more complicated tasks in the world with an increasing degree of freedom. A standard contemporary computer is capable for decision-making according to predefined rules. Similar to our neural organism, computers are units for data processing. Therefore, they now support human decision-making by handling an enormous amount of data, performing calculations and increasingly making low-level decisions. For more complicated tasks with even more degrees of freedom like adapting to new situations, the next natural step would be agents. The agents are capable of perceiving their environment, take decisions autonomously and execute planned tasks. Similar to humans, these agents will be able to make decisions independent of their creators. The starting point of the present work is decision-making based on experience.

1.1 Background

A new cognitive architecture based on psychoanalysis was founded at the Institute of Computer Technology, TU Wien [1] by Dietmar Dietrich [Die00] in 1999. The model was originally called the *Artificial Recognition System (ARS)* [2]. From 2015, it is known as the *Simulation of the Mental Apparatus & Applications (SiMA)* [2]. From here on, only the term SiMA will be used. SiMA is based on the second topographical model of psychoanalysis [DFZB09, p. 100 sq.]. This approach is new in this research area [DFZB09, p. 54]. As psychoanalysis alone is incomplete in explaining the human psyche, the SiMA model is complemented with knowledge from other scientific disciplines, in particular, neurosciences [DFZB09, p. 44]. For instance, low-level sensory perception and reflex actions are not constituents of psychoanalysis; therefore, this parts must be added from other sources. In the following, a brief history of the project and fundamental arguments for the SiMA model are presented.

According to [Die00], [DZ08, p. 12] and [DS00, p. 348], modern technical systems for automation grow more complex as they have to handle a large number of interacting, distributed sensors. This cannot be done so easily with traditional centralized systems. The present technology is approaching its limits, as a system can barely perceive what is going on in the environment [Vel08, p. 6]. In the area of automation, mostly reactive systems are used, which cannot cope with unforeseen situations [Rus03, p. 4], (who cites [WH97] and [SDK01]). This was clearly visible in the initial stage of the project. There, the main task was to evaluate different types of data within the Smart Kitchen Project [Rus03, p. 19], [Fue03, p. 93], which was a kitchen equipped with a large number of sensors and actuators connected to field bus systems. The purpose of the system was to assist persons by recognizing the situation and react to it. Sensor and actuator data were merged into multiple levels of symbols. They provided a testing platform for different types of decision units [PDHP07, p. 27]. There, the problems of a high data flow were recognized.

There is a need for new solutions, which can compress, filter and abstract perceived information. This information shall then be inferred with known situations, to act in an appropriate way. A promising approach to solve this problem is to use a bionic approach, to try to imitate nature in solving problems [BDK+04, p. 1219], [RHBP04, p. 349], [DKM+04, p. 93]. The works [DZ08, p. 12] and [DPD12, p. 5] state that a model, which only describes human behavior with rules, is not sufficient, as the behavioral model would be too complex for describing the human mind. A functional model, however, reduces complexity and enables the feasibility of the modeling of the human mind. It is not possible to conclude about the inner functions from statistical observations only. An analogy would be to extract the functionality of Microsoft Word just from the statistical measurement of the states of the memory [DPD12, p. 10]. Instead, a functional model is needed, which describes the fundamental, consistent functions behind human behavior. It would be in accordance with the workflow of computer design [ZBD11, p. 61-1]. Based on the lessons learned from the Smart Kitchen project, a top-down approach has been selected instead of a bottom-up approach. According to [DZ08, p. 13], the top-level layer of the model uses symbolic coding, manipulation, and decoding of information, i.e. an internal representation system. This reduces the amount of information, which has to be processed as sensor data are merged into symbols of various abstraction levels. As the top-down bionic approach has been used, it becomes clear that the focus of the model has to be on the top-level functions of the human psyche and not on the level of neurons. Neurons are only relevant at lower layers. As the functional model should not be based on observed behavior, emergent models, as explained in Chapter 2.1.1 cannot apply to this approach.

Another condition for a bionic model was the need of embodiment. According to [Dam00, p. 53-57], the body of a living being has needs, which must be satisfied, to survive. Intelligence is an evolutionary step in satisfying those needs. The work [DZ08, p. 13] states that in the last decades, research in artificial intelligence has concluded that emotions, language, and consciousness play a major role in an intelligent being.

Some available cognitive architectures are based on psychological or neurological theories, which would justify them as bionic approaches [VMS07, p. 162]. However, as only a small piece of the functionality of the human mind is known today, many psychological models, theories only focus on a part of the human mind and cannot provide a holistic of the psyche [DFKU09, p. 100]. Considering alternatives, the school of psychoanalysis provides a model, which fulfills the mentioned criteria: a

top-down approach, a functional model of the mind, the interface to the embodiment and a decision unit based on emotions and feelings [DZ08, p. 14]. It should be noticed that psychoanalysis as a natural science is debated in the scientific community as described in [Muñ10, p. 94], [Ros11, p. 7], [CS88, pp. 1-7] and [Gar86, p. 9]. Whether psychoanalysis can be considered as a natural science or not, it still fulfills the necessary criteria for a proper solution of technical problems. Therefore, it is a valid bionic theory [ST02, p. 291], [Dam99]. In the year 2005, a technical model that looked like a cognitive architecture was presented [PP05] and [PPDB05]. It can be referred as the first SiMA model. It was based on the theory that was stated by Dietmar Dietrich in the year 2000. This model can be considered as an intermediate model, as several psychological concepts have been used. Finally, this model was abandoned because of incompatibility of the different theories and complexity of the resulting system [DFZB09, p. 53-54].

Based on the lessons learned from the first model, a second model was introduced 2009 [DFZB09, p. 37]. Different to the first SiMA model, the goal was to stick strictly to the second topographical model of psychoanalysis, to preserve the consistency of model [DFZB09, p. 100]. Other theories are only complementary as long as they do not contradict psychoanalysis. In the following years, the functional modules have been defined in close cooperation with psychoanalysis. Each Ph.D. student has a defined version of the model with the syntax ARSi[Number]. From 2015, the syntax is SiMAi[Number]. The work [Lan10] presents the ARSi09 version, where the core structure of the model has been created, which defines an overall computational framework with modules and their functionalities. Concurrently, in [Zei10], the SiMA model has been extended with an internal representation and long-term memory. These concepts have been defined and implemented in the ARSi10 version. Until now, general concepts have been handled. In the ARSi11 version [Deu11], the focus is on embodiment and drives in the unconscious data processing functionality of the model. In the ARSi12 version, which is described in [Muc13], the low-level perceptual inputs of the SiMA model are in focus. In one of the recent version ARSi13 [Hin14], possibilities of language reasoning have been analyzed.

In Chapter 2.3, the current SiMA model is described in detail concerning relevant concepts, which are the starting points for the present work. However, a brief summary of the starting points is provided here from the perspective of decision-making based on experiences. As the model versions ARSi12 and ARSi13 have been created in parallel with this version, the SiMAi14, ARSi11 will be referred to as the previous version. ARSi11 includes the concepts of all previous works including [Zei10]. It is a cognitivist model [VMS07, p. 153] that models the functionality of the human psyche. Compared to other cognitive architectures (see Chapter 2.1), the edge of SiMA is the high resolution of unconscious drives and emotions, as well as the unique concept of defense mechanisms, which are described in Chapter 2.3.1. The agent consists of a body, which generates needs, to keep the homeostasis in balance. Each unbalance results in a drive, represented by sexual drives and self-preservation drives in the *drive track* as shown in Figure 1.1.

These drives, then form goals in the track *selection of need*, which need to be fulfilled by performing some action in the environment. At the same time, objects, which are things like a food source, are perceived through the *external perception* in the *perception track*. Each element is evaluated regarding its potential drive satisfaction. Based on the drive state of the agent alone, it is decided to which

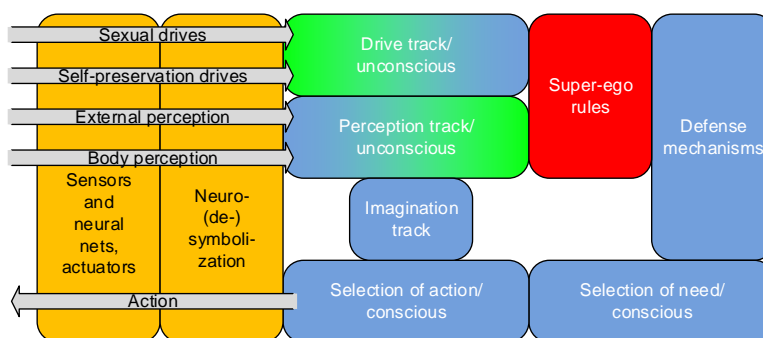


Figure 1.1: High-level overview of the SiMA Model [BGSW13, p. 2]

element in the perception to be moved in the track *selection of action*. This action is then automatically executed. The limitations of this implementation regarding experiences are that each goal is processed equally in each model step, which means that the agent did not know its state and no short-term memory exist. Further, only default actions are available. It is a single cycle process without any temporal processing because nothing is saved for the next step. Therefore, there are no concepts of handling experiences.

1.2 Motivation

The primary motivation of this thesis is given by the answer to the following question: Why is experience-based decision-making necessary in an autonomous agent from a cognitive architecture? To answer that question, the needs of that functionality are discussed in the following. From a technical point of view, there are new demands on software applications, which work in environments with a high degree of freedom. These applications need anticipatory features to manage their tasks. Among other applications, two applications will be highlighted, where the SiMA model will be used.

First, in building automation, the goal is to support a certain level of comfort while energy consumption shall be optimized for minimal consumption of conventional energy sources. Further, the system shall be robust and flexible enough to adapt to a changing environment. Nowadays, software applications are like purely rule-based systems in [Vel13, pp. 68-76] and [SVP10, pp. 378-392] or model predictive control [FZP11, pp. 1-6], which offer well-working means to optimize a system for certain situations, but can hardly cope with non-ideal environments. The implementation of a control application based on the SiMA model is the topic of the project ECABA (Energy-efficient Cognitive Autonomous Building Automation) [Zuc15, pp. 1-6].

In the research area of economic simulations, a need for agents with human-like behavior can be found. In the project CogMAS (Cognitive Multi-Agent System Supporting Marketing Strategies of Environmental-Friendly Energy Products) [SMW+15, pp. 1-6], the goal is to show that agent systems can simulate the customer decisions in the energy market. It shall be analyzed; which factors influence the choice of a certain power supplier. The challenge is to consider a significant number of factors that influence human decision-making. Such factors are rationality, feelings, multi-agent-interactions, and multiple goals to satisfy. Many of these factors cannot be covered by common micro-economic models

like LARA [BEH+12, pp. 1-8], [SMW+15, pp. 1-6]. In the long-term perspective, the present project will provide a possibility to test marketing strategies for selecting renewable energy sources.

Autonomous agents, in general, offer suitable solutions for applications, which operate in complex environments, real or virtual. They have motivations; perceive events in the environment with their sensors and reason about the possibilities to fulfill their goals. However, the area of economic simulations is highlighted to demonstrate the strengths of SiMA compared to other cognitive architectures. While most other approaches concentrate on logical and rational processing, SiMA follows a holistic approach, which also covers the non-rational parts of human thinking. If the fact is added that the software agent is based on psychoanalysis and is supposed to be able to act “neurotically” unpredictable, it perfectly fits into simulations of human individuals, which do not demand a sound decision-making. In SiMA, the system goals are mainly based on bodily needs, but also mental needs. Mental needs are grounded in bodily needs that are transformed in the defense mechanisms. Goals are reached by performing actions, which alter the state of the environment. As human-like capabilities are required and the main parts of actions in the environment are derived from personal experiences, there is a great need to introduce decision-making based on experiences into the SiMA model. Experienced situations affect the foundation of the decision-making in SiMA. Recognition of already known sequences is a vital part of developing the ability to estimate the outcome of certain situations, giving an agent advantages compared to purely reactive agents. It provides the agent with the preconditions for the development of planning, which is one of the outstanding attributes, which differs us from other animals. SiMA can then offer a proper solution to the mentioned problems in the projects above.

For social behavior, the usage of experiences is necessary to recognize the intention of other individuals or systems. Further, if learning is eventually implemented, together with the ability to estimate possible outcomes, it opens up for individual development, creativity, and adaptation. By connecting observations with experience and recognizing the intention behind single actions, a further step towards systems that reason human-like would be achieved. To conclude, SiMA needs experience-based decision-making, to be applicable in complex environments, i.e. simulators with many objects or the real world. Consequently, the further question is how to implement this functionality.

1.3 Problem Description

The current state of the ARSi11 (starting point for this work version SiMAi14) model is briefly described in Chapter 1.1 and detail in 2.3. The desired state of this work is described in Chapter 1.2. Because experiences are a vital part of human-like behavior, and it has not been considered yet in SiMA, the following top-level research question has been formulated for this work:

Research question 1: How can SiMA be extended to be able to consider past experiences when taking decisions?

The main hypothesis is that a model of the human mind based on psychoanalysis can be implemented as a cognitive architecture. For this particular work, the more special hypothesis can be stated. It states

that the cognitive architecture SiMA will be able to estimate the consequences of its actions if past experiences are considered in the decision-making.

In order to make it obvious which challenges can answer this question, sub-research questions will be derived from a concrete use case, the *Use case 1 Obstacle Modification*. It is derived from two sources: a previously developed *Use case 1* by the research group and a vision called *SiMA Craft*. Use case 1 implements psychoanalytical requirements and is based on purely fundamental research. The vision called SiMA Craft demonstrates a view of what could be done with SiMA in a not too distant future. It is an extension of Use case 1 that is enhanced with additional features, which will be required for future applications. The vision connects SiMA with the motivation of this work, which is to fulfill the demands of providing human-like capabilities in agents. However, SiMA Craft is not a realistic use case to implement at this stage as it demands many unimplemented concepts, and therefore, it is just a vision for guidance at the moment. Use case 1 Obstacle Modification is positioned between the SiMA Craft vision and Use case 1. This use case is the validation¹ of the present work, as it implements the fundamental research concepts defined by psychoanalysis from Use case 1. Also, some features are needed to approach the vision of SiMA Craft. The implementation of concepts for the usage of experiences in decision-making within this use case verifies its psychoanalytical correctness.

Vision of a Future Application of SiMA: SiMA Craft

Six SiMA agents are put into a competition in an artificial world with a map suitable for team competitions [3]. An agent can eat or collect food, communicate or attack other agents. The agents are divided into two teams, where each team has a home base in different corners of the map (see Figure 1.2). The teams compete for the limited food sources, which are distributed on the map. The top goal of each agent is to ensure the survival of its team. A single agent survives if its bodily needs are fulfilled, in this case, especially the drive for hunger. Food sources are regenerated with a regeneration rate, which is lowered with increasing time. It means that there is more than enough food as the scenario starts and too less after a while. The team wins, where at least one agent survives.

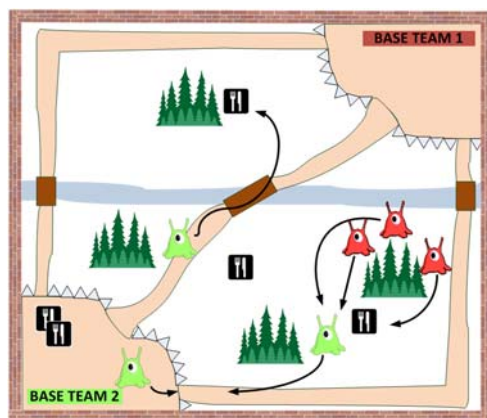


Figure 1.2: Map with food sources and agents in the SiMA Craft vision (not realizable at the moment)

¹ Validation: if no specification is available; verification: if a specification is available

To be successful, the agents will have to manage some of these challenges:

- Drive delay, as it may be better to carry the food to the own base instead of eating it at once;
- Short-term and long-term and planning, to react to immediate events, as well as following organizing an ambush;
- Navigation based on known landmarks, to be able to find the way back to the base or to know where a food source will regenerate again;
- Interaction with each other and the opponents, as it is only an advantage to attack if the opponents are outnumbered.

Complex behavior could also be observed:

- Usage of social rules within the team, like helping an agent under attack, although it does not directly fulfill the need to eat;
- Organizing an ambush on an opponent, who moves around alone;

One thing should be noticed about this example. Within an artificial world, it would not be very hard to create an artificial player, which would show some of the desired behaviors. In the general-purpose architecture of SiMA, behaviors are configured by adding and changing its memories, experiences and personality parameters. In the current state (ARSi11), the SiMA agent at is not able to even go around an obstacle because of the missing access to experiences. Therefore, the derived Use case 1 Obstacle Modification in the following chapter shall provide a first step in the direction to SiMA Craft.

Use case 1 Obstacle Modification

Use case 1 [BGSW13, p. 4] was developed by the SiMA research team in close cooperation with psychoanalysts. It represents the current state of SiMA. It was adapted to the implementation of ARSi11. The agent sees a food source and another agent. From here, depending on the drive state and the configuration of the defense mechanisms, the agent can either eat the food source or consider the influence of the other agent. The use case was primarily designed to test the impact of the defense mechanisms of the drive state. In the original Use case 1, there is no need for experiences, which contain spatial information, as the agent can reach all of its goals with the set of default actions. Central features of the vision SiMA Craft are human-like capabilities, e.g. navigating in a world based on

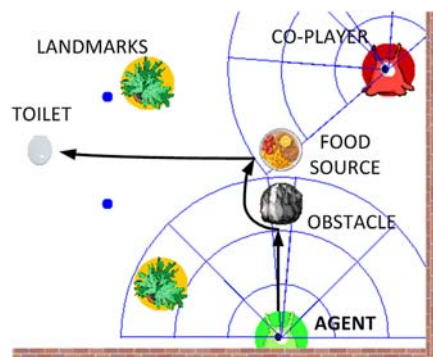


Figure 1.3: Initial state of Use case 1 Obstacle Modification

landmarks and predicting the outcome of observed situations. Therefore, this use case is extended with obstacles and other objects of interest. For instance, a stone is placed between the agent and its food source, to make the agent recall experiences on how to get around the stone. It forces the agent to use experiences, which tell the agent how to handle a given situation. In Figure 1.3, a possible setup of Use case 1 Obstacle Modification is illustrated. As Use case 1 Obstacle Modification is very close to Use case 1 and it has been developed for fundamental research, this use case is also basic research, but with a direction towards an upcoming application.

In ARSi11, the agent would use the only option that it has available and try to execute the impossible task to go through the stone. If experiences are used to describe how to go around the stone, they can be taken into account for solving the problem. The agent will be able to recall suitable experiences for the current situation. If the agent has hunger, it shall prefer to recall memories about ways of satisfying the hunger instead of recalling memories about how to satisfy some other drive.

Research Questions

Use case 1 Obstacle Modification allows the extraction of more detailed research questions. The temporal dimension needs to be introduced in SiMA. Therefore, all relevant functionalities of the model have to be analyzed, modified and extended. By studying the architecture of ARSi11 (see Chapter 2.3), it turns out that the problem can be partitioned into four areas of the model: data structures (internal representation), memory retrieval, sequence recognition, and decision-making. In the following, limitations of ARSi11 in Chapter 2.3 are used to generate lower level research questions.

Data Structures: In ARSi11 (see Chapter 2.3.4), there are no any data structures, which can capture perceived situations. All perceived objects are handled completely independently of each other. Functionality for usage of situations is missing. Several situations then form a sequence of situations, which represent an experience. There are some theoretical ideas available; hence, they are insufficiently specified. Therefore, the research questions in this area of the models are:

Research question 2.1: How shall data structures for situations be designed?

Research question 2.2: How shall situations be organized into sequences?

Memory retrieval: As no data structures are available for handling situations, there exist no concepts of activating them from the memory (see Chapter 2.3.3). The memory of ARSi11 only contains object classes, as a food source, which represents all known instances of that class, but no instance of a food source in a certain context. Then, stored situations have to be compared with incoming data and retrieved. Due to the limitation of computational resources in a computer as well as in living beings, not all experiences can and should be kept loaded all the time. Therefore, there is a need for functionality, which provides the agent with memories important for a given situation in a certain context. The context means the internal state of the agent, i.e. the drive state. This problem generates the next research sub-question:

Research question 3: How shall those stored situations be retrieved from the memory, which is relevant to the current situation within a certain context?

Decision-making: There is no temporal processing in the decision-making in SiMA. It cannot save the current state, which is necessary for the recognition and analysis of sequences. Temporal processing demands a goal management, which lasts over several model steps and it requires the use of internal actions like setting the attention of the system. Therefore, the following research sub-questions are defined:

Research question 4.1: How shall decision-making be extended to consider temporal data structures?

Research question 4.2: How shall internal actions be realized within the decision-making?

Sequence recognition: In this area, no concepts are available in SiMA, so concepts have to be developed from scratch. The agent has to be able to map the currently perceived situation to a situation in a sequence, in order to be able to evaluate its applicability. Further, information from the relevant sequences has to be utilized in the agent. Therefore, the following research sub-questions are defined:

Research question 5.1: How shall temporal recognition and confirmation of applicable sequences be realized?

Research question 5.2: How shall information from sequences be supplied, in order to be useful in the decision-making?

1.4 Task Setting

The following main task is defined, which is derived from the main research question R1 of this Ph.D. in Chapter 1.3:

Task 1: The main task of this work is to introduce experience-based decision-making in SiMA.

In order to complete the main task, subtasks are defined based on the sub-research questions in Chapter 1.3:

- *Task 2: Extend and modify current data structures to include situations, sequences of situations and goals (Research question 2.1 and 2.2)*
- *Task 3: Introduce memory retrieval of situations based on other situations or the perception (Research question 3)*
- *Task 4: Develop recognition of the current situation of a sequence, extract the purpose of a sequence for goal generation and generate an expectation of what will happen next (Research question 4.1 and 4.2)*
- *Task 5: Refactor and extend the decision-making, in order to be able to reason about temporal data structures (Research question 5.1 and 5.2)*

Concepts for fulfilling these tasks will be presented in Chapter 3.1, 3.2, 3.3 and 3.3.

The following development constraints have been defined, in order to limit the scope:

- There is no hierarchy of the internal representation of external objects, i.e. an external object cannot be described by other external objects

- No learning will be implemented, i.e. nothing will be saved to the long-term memory by the agent
- Topics like drives, emotions, the modules of the drive track and defense mechanisms (see Chapter 2.3.1) are only considered if they play a role in solving the task

1.5 Methodology

The vision SiMA Craft is used as a vision for this work. Then, the current state of SiMA is analyzed, to extract the delta between the goal state and the currently implemented state. It has resulted in the problem description and the task setting.

The development of concepts for the tasks (Task 2 - 5) will be done independently in a partitioned iterative approach [4]. In research projects with much design uncertainty, this approach is appropriate, to mitigate the risks of late design breakage, which often happen in a sequential development process like the Waterfall- or V-Model [GB10, pp. 374-378]. In the standard V-Model, all concepts would first be defined; then they would all be translated into requirements, which are used to do software design. Finally, everything is implemented and tested. In SiMA, there is a high probability that requirements may change as new concepts replace the old ones or that different parts of the system may not interact as intended. Studies show that an iterative approach is faster and more robust than a traditional, sequential approach [GB10, pp. 378].

In the partitioned iteration approach, each task represents a partition and for each partition, all development phases are run through from partition one to partition four like shown in Figure 1.4. In a top-down manner, the business architecture design is performed (phase A in Figure 1.4). Psychoanalytical constraints are defined together with the psychoanalysts in the research group. As psychoanalysis is no axiomatic theory [DPD12, p. 4], these psychoanalytical constraints or statements are updated as new knowledge is added. However, they define the base of the technical architecture development, and the challenge is to keep the model stable. Here, the following requirements must be maintained: First, technical concepts shall use the concepts of the psychoanalytical theory, as far as they are available; it means that no contradictions to the psychoanalytical model are allowed in the functions of the technical model. Second, if there are no existing concepts of the psychoanalytic

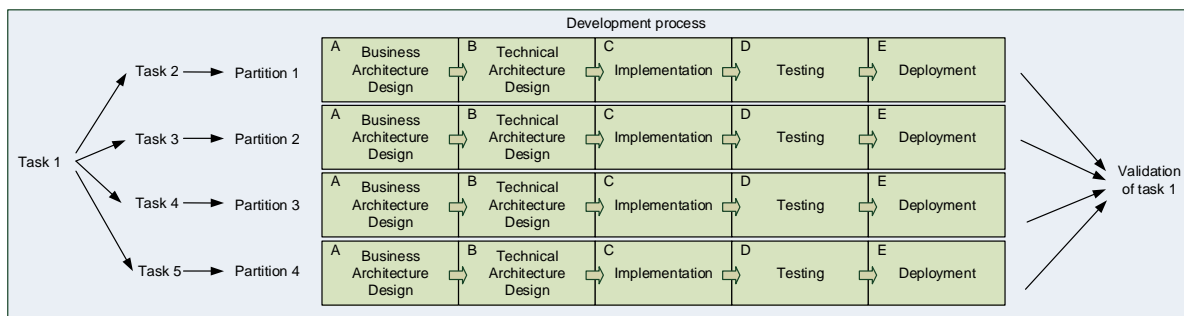


Figure 1.4: Partitioned iterative development process

theory, state of the art theories of psychology and biology shall be used as a base; and the result of this phase is the design of the business logic, which will be presented in Chapter 3.

After that, the technical architecture design phase (phase B in Figure 1.4) follows, where the defined concepts are transformed into software design. State of the art methods of other cognitive architectures is used as a source of inspiration. The wheel does not need to be invented twice. Due to the partition and changes in top-level concepts, the negative impact of changed business architecture design of other partitions is limited, because parts of it are first defined in later iterations. Each partition is then implemented separately (phase C).

Finally, the implementation will be tested (phase D) and deployed (phase E) independently using the derived scenarios of Use case 1 Obstacle Modification. A scenario represents one path of the use case. Scenarios, which contain the main task (Task 1), are used to validate the solution and the answer to the main research question (Research question 1). By fulfilling Use case 1 Obstacle Modification, the psychoanalytical correctness of the implementation is verified because the use case extends Use case 1. There, psychoanalysts have defined the expected behavior of the agent for certain conditions.

2. State of the Art and Related Work

“Critics sometimes point out a pattern, that every thirty years, AI practitioners claim that superintelligent robots are just around the corner. Then, when there is a reality check, a backlash sets in.”

[M. Kaku in Physics of the Future]

There are some cognitive architectures, which are based on different theories ranging from decision-making theories to neurology. As a consequence, they have different focuses, strengths, and weaknesses. The two most important questions raised in this chapter are where SiMA fits in the configuration of various cognitive architectures and what is the current state of SiMA as this work starts. The various cognitive architectures are evaluated in two ways. First thing is how the theoretical base fits in the functional model of the human mind, and second, how the technical implementations work. Common characteristics are described in Chapter 2.1, in order to enable a direct, functional comparison. Then, related architectures are described and compared to each other in Chapter 2.2. SiMA is described and evaluated in Chapter 2.3 with the vocabulary of the common characteristics of Chapter 2.1. It is directly compared to the previous architectures and placed in relation to them, where the strengths of the SiMA approach are clearly visible. However, as it is the starting point of this work, also the weaknesses of the current implementation are highlighted regarding using experiences in decision-making, to describe the problem, which will be solved in this work.

2.1 Evaluation Criteria for Cognitive Architectures

A system is called an agent, if it operates autonomously in an environment, where it fulfills its goals by altering the state of the environment through actions. A cognitive agent implements a cognitive architecture. According to [VMS07, p. 151] *“cognition can be viewed as the process by which the system achieves robust, adaptive, anticipatory, autonomous behavior, entailing embodied perception and action”*. It implies that the cognitive agent is able not only to understand the current situation, to be able to react, but also to predict future events, to be able to function efficiently in situations for which it was not intended [VMS07, p. 151]. Cognitive architectures can, therefore, be seen as the antithesis of expert systems, which are operating in narrow contexts [LLR09, p. 2]. Usually, the cognitive architecture provides interfaces to some embodiment. The internal states have to be grounded in some perceived sensor data, and the lowest levels of actions have to be primitive, i.e. real actions, which have an effect on its environment [Lan05a, p. 20].

2.1.1 Cognitivist, Emergent and Hybrid Architectures

To discuss the challenges of cognition, two classes of architectures have evolved: the cognitivist approach and the emergent approach [VMS07, p. 152]. Some known architectures of the cognitivist approach are, among others SOAR [LCC+11, pp. 5-32], BDI [GTC+10, pp. 74-83], ACT-R [ABB+04, pp. 1036-1060] and ICARUS [Lan05a, pp. 18-25]. Some known emergent approaches are Global Workspace [SB05, pp. 164-174], SASE [Wen04, pp. 1-8] and DARWIN [KE06, pp. 37-42]. In the following, the most important characteristics and differences are highlighted. These two classes are the opposites of each other in many ways. Despite this, some architectures manage to implement parts of both. They are called the hybrid approach. Some representatives are LIDA [RBF06, pp. 1-6] and Kismet [Bre02, pp. 5-252].

A cognitivist approach is the classical and the common view of what a cognitive architecture is. It is characterized by a top-down design process and is based on symbolism, rule/function-based, algorithmic information processing. This approach is an application of a layered system of the information theory. It defines the foundation of designing and realizing computers today [DBM+10, p. 80]. Within information theory, the main topic concerns how information is structured, stored, acquired and processed in the communications engineering [DBM+10, p. 80]. A layered model is defined. The following layered system can be applied to a computer, which is defined as a system that manipulates, stores and transfers data [DSBD13, p. 6664]. *functional layer 1* is defined by its hardware and the functions related to the hardware. *functional layer 2* contains the operating system and its functions. Finally, in *functional layer 3*, the functions of the applications are defined [DSBD13, p. 6664]. However, the layered system can have a higher granularity than the three mentioned layers, where a prominent example is the ISO/OSI model in communications technology [DBM+10, p. 80]. A top-down designed, layered architecture applies to all cognitivist architectures. Orthogonal to functional layers, *data layers* are defined for describing how memory is accessed. At the lowest data layer, the physical memory is located.

A consequence of a layered system is that there is an interface between the external world and the internal representation. States and behaviors of the external world are abstracted into a symbolic representation structure. A *symbol* is a pattern of an expression, which can be interpreted by the architecture. Information is stored in expressions, which make them human readable. For agents in simulated or physical environments, these symbols are spatial-temporal representations of the sensor data, where the process of grounding the symbols in sensor data is either hard-coded or learned by creating a mapping of sensor inputs to the internal representation. The universal representation base of the system can be exchanged between agents, as they share the same translation of the symbols. As knowledge is separated from the actual architecture, the cognitivist system, in general, does not need to be embodied and can operate independently of its environment [VMS07, p. 152]. Some of the architectures in this paradigm started as a problem solver without any need to have a body. However, if the goals of the system are not pre-programmed, a body is necessary as a source of motivations [SDW+13, p. 6646].

The emergent approach does not follow the layered model and is characterized by the opposite position as the cognitivist approach, and it claims that cognition is emergent out of self-organizing systems.

Knowledge representation is not accessible to external observers, because it is not localized, but encoded in the structure of a distributed network. It uses self-organization, self-development, and self-maintenance through the interaction of distributed, interacting components within a network to realize cognition. Examples of emergent systems are neural networks; which knowledge is hidden from human observers. Perception is here a change of the system state, which implies embodiment. Representations are only grounded if they contribute directly to the persistence of the system. It is the product of the personal history of that agent and pieces of information that cannot be transferred to other agents in the same manner as in a cognitivist architecture. Instead, knowledge representation is a part of embodiment [VMS07, p. 152].

The cognitivist architecture provides tools to manipulate or to reason about symbols, to be able to interact with the world in an appropriate way. Based on the reasoning on the internal structures, the goal of cognition is to provide a goal-directed behavior, which is adaptive and anticipatory. For anticipation with a given situation, the knowledge base is used to find proper solutions. It adapts itself by adding new knowledge to the knowledge base. The cognitivist architecture allows deliberative processes, i.e. processes like thinking. With thinking, the processing of the internal state is meant in this work. It can be independent of the external inputs and lasts over several cognitive cycles. A *cognitive cycle* can be seen as a serialization of parallel processes, to simplify their modeling [BDD+14, pp. 100]. In such cases, an action can be set after several cognitive cycles. For an emergent architecture, actions are a reaction to a change in the environment. While the cognitivist architecture can reason about a solution independent of its sensory inputs, the emergent architecture automatically reacts to environmental changes without deliberative processes. Here it is important to differ between real-time and reactive. While reactive means that a system answers an input within the same cognitive cycle, real-time means that the system copes with demanded time limits. Therefore, a deliberative system can also act in a real-time environment. The way an emergent architecture anticipates with a situation is to recall some perception-action states, without executing the associated actions [VMS07, pp. 152-154].

Both architectural approaches have their advantages and limitations. An advantage of the cognitivist architecture is that the representations are human readable, which allows inspection of internal states. The major drawback of cognitivist architectures is that they are all a product of human design. It puts an upper level of self-development and limits a system to the knowledge of the designer. Therefore, these systems usually work well in known environments or in given problem spaces, for which it was designed, but they have problems such as the internal descriptions of the symbols are not fitting the perceived symbols [VMS07, p. 154]. Compared to the emergent approach, the cognitivist approach does not provide the same possibility to achieve behavioral plasticity and to adapt to new situations as well as the emergent approach [VMS07, p. 161]. On the other hand, the emergent approach is still struggling on low-level interaction with the environment, e.g. the integration of visual stimuli and motoric control. Further, it is not able to show any higher cognitive capabilities in the same manner as the cognitivist approach [VMS07, p. 158], [LLR09, p. 2]. The hybrid approach aims to integrate both the cognitivist and emergent approach to a single architecture. In those systems, the low-level system functions, such as a sensory-motor system are often emergent, while the high-level cognitive functions are realized in a cognitivist approach [VMS07, p. 161]. Although these two architectural paradigms are contrary in many ways, the trend is going in a direction, which lowers the gap between

them [VMS07, p. 175]. This can be seen in the increasing numbers of hybrid approaches, which do not require that the whole system is based on symbolic structures.

SiMA is following a top-down cognitivist approach, and it applies strictly separated layers. On the bottom layer of the system, sensor and motor data are received and sent. Sensor data are passed to the next layer of the system, the neuro-symbolic layer for transformation into symbols. Action symbols are transformed into commands [DZBM09, p. 5]. The neuro-symbolic network in this layer works similar to a neural network, with the difference, that each node has a certain meaning. The information on the network is localized [VBP09, pp. 2]. The generated symbols are then passed to and processed in the mental layer of the architecture, where they are transformed into the internal representation of the mental layer. To narrow the topic, therefore, only cognitivist and hybrid architectures are analyzed and compared to SiMA. Also in SiMA, learning is not a topic yet and therefore it is not analyzed in the described cognitive architectures. The reason why learning is not an issue of SiMA yet is that the first stage of the project is first to define all the functionality of the model, to implement it and to evaluate the current result of the functions. It shall reduce the complexity of the modeling. Eventually, as the functional modules are completely defined, learning mechanisms will be considered.

2.1.2 Characteristics of Cognitivist Architectures

In the following, the typical functional process of cognitivist architecture is described. The terms used here are borrowed from specific architectures, because there are no common terms, although there are often equivalent concepts in other architectures. Different to the claim of [Lan10, p. 41], in the following, a common base of functionalities and processes of cognitive architectures will be extracted, which allows a direct comparison of the functionalities of cognitive architectures. It has to be emphasized that in the comparison of cognitive architectures don here, only the design of the architecture is comparable, not the approach, which led to that design. In [Deu11, p. 24], common cognitive architectures were compared from a perspective of artificial general intelligence. The term *artificial general intelligence* addresses the general nature of a research project, and a cognitive architecture that is a part of it is supposed to develop human-level artificial intelligence [WG06, pp. 1-2]. Hence, there is no exact definition of what exactly shall be fulfilled. The conclusion was that all approaches are insufficient for a comparison because they only cover a small part of the whole. It also shows the enormous complexity of artificial general intelligence, which cannot be mastered by a single state of the art architecture. A holistic approach is needed.

Decision Process

To get a meaningful comparison of common architectures. Instead, the smallest common denominator of all cognitive architectures is selected. This chapter will only concern the functional layer 3, which has been defined in 2.1.1 and in [DSBD13, p. 6664]. It is because the inputs and outputs of most relevant cognitive architectures start with known symbols and not with pure sensor data.

The characterization starts with the process of how perceived data is combined with the knowledge to provide an action. In Figure 2.1, common process steps of cognitive architectures are illustrated. The step *activate beliefs* shows the activation of *beliefs* based on perceptual inputs [GTC+10, p. 78], [LCT11, p. 5], [Win09, p. 5]. A belief represents the perceived world, where two types of information

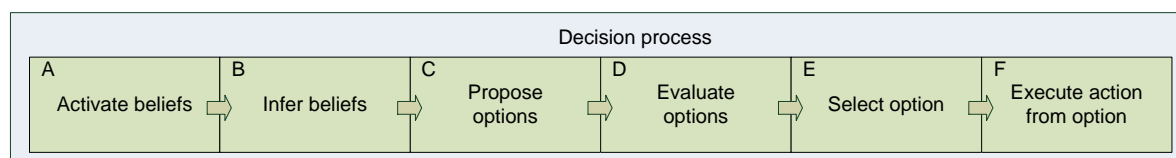


Figure 2.1: One cognitive cycle of the decision process of a cognitivist architecture

can be distinguished: basic beliefs and inferred beliefs. *Basic beliefs* represent objects and positions. *Inferred beliefs* represent information inferred from the basic beliefs as referred with equivalent descriptions in [GTC+10, p. 78], [LCC+11, p. 12], [RBF06, p. 3]. Basic beliefs may be provided and updated by a subsystem [LCT11, p. 6]. The inferred beliefs are normally acquired through the use of *production rules* on the present beliefs [GTC+10, p. 78]. Beliefs are inferred in the phase *infer beliefs*. For instance, if the basic belief consists of two objects with their positions, inferred beliefs could describe the relation between the objects [LCT11, p. 6]. Production rules alter the system state in the way that works as if-then statements, which consists of conditions, which shall be fulfilled to execute one or more certain actions [LCC+11, p. 7]. In the case of inference of beliefs, a production rule, whose conditions are matched by the basic belief may add a statement to the collection of beliefs. Simultaneously or sequent to the inference process, production rules also propose *options* as described in [WHB+00, p. 532]. There, options represent what the system can do regarding internal or external actions. Proposed options describe what the agent can do in the current situation. Another term *skill* could also be used here. A skill describes how to get from a particular state to another [LCT11, p. 12]. In the phase *propose options*, options are generated, which match the current beliefs and may take the agent one step closer to its goal [LCT11, p. 10].

A *goal* (in SiMA terms defined as drive wish and expressed through a planned goal) is a defined as the desired end state to reach for the agent [LCC+11, p. 9], [GTC+10, p. 80]. A *motivation* originates from motivational processes and is the course that shall be set. It is something that represents what the agent wants to do [GTC+10, p. 79], [Slo00, p. 6]. A goal has to be reached, to satisfy some motivation. The question is the origin of motivations. Architectures use motivations and goals in different degrees. For a problem solver, a problem space is given. No motivations are necessary as a top-level goal is preset [LCC+11, p. 9], [ABB+04, pp. 1041]. Another way is to define some high-level motivations or attitudes of an agent, which are triggered by certain beliefs (certain situations) [GTC+10, p. 78], [FR06, p. 41]. A third way is to use embodiment as a motivation generator. In a complex world, where an embodied agent interacts through a body, the purpose of the mind is to manage the survival of the body [Deu11, p. 75]. While motivations originate from the internal needs of an agent, *emotions* originate from situations and implement motivations [FR06, p. 41]. Different situations are not only evaluated regarding its match to some environmental activation (like the matching of the basic beliefs with a template), but also through emotions [RBF06, p. 5]. Therefore, emotions are also used as motivations in some cognitive architectures. They influence how a goal is reached.

In phase *evaluate options* the proposed options are evaluated regarding their ability to enable the agent to reach its goal [LCC+11, p. 10], [LCT11, p. 12], [LLR09, p. 5], which satisfies a certain motivation. Finally, one option is selected in the step *select option* because it is estimated to have the best

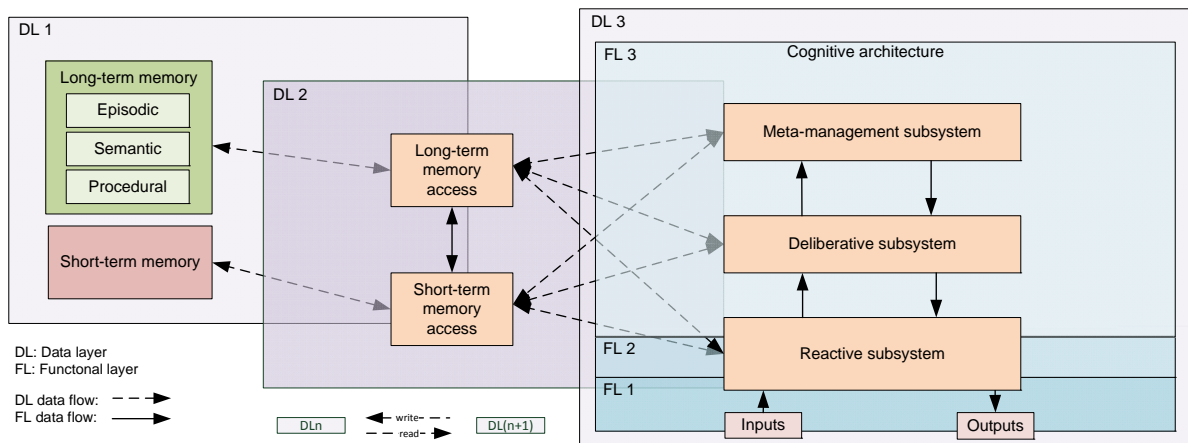


Figure 2.2: Common components of a cognitive architecture, including their assignment to layers

possibility to achieve the goal. In phase *execute action from option*, the action, which is linked to the option, is executed [LCC+11, p. 11].

The decision process defines the basic way of how the simplest cognitive architecture works in general. In most cognitive architectures, each decision process is run through multiple times during a cognitive cycle, but on different levels. As will be seen in the following chapters, it also applies to SiMA and as a consequence, also to the human brain.

Components and Functions of a Cognitive Architecture

To enable a decision process, cognitive architectures from the cognitivist and hybrid approach normally consist of the following components as shown in Figure 2.2: *short-term memory*, *long-term memory* and functional processes for processing data [LLR09, p. 1], [LCT11, p. 2]. In Figure 2.2, the components are ordered in layers. The functional processes can be found in the functional layers. Data layers represent data access and memory. The long-term memory, which is located in *data layer 1*, contains all permanent knowledge of the agent. It can be the mapping of symbols to sensor data through perceived attributes, concepts, experiences, skills and production rules. For instance, in the architecture SOAR, the knowledge of the long-term memory enables the agent to know how to respond to different situations in the form of production rules [LCC+11, p. 6]. Concepts in the long-term memory are usually generic and are being instantiated if they are matched to instantiated beliefs in the short-term memory [LLR09, p. 11]. A sort of memory is a *semantic memory*, which stores facts and statements about things in the world without contextual information [LL11, p. 3]. An exception to the generality is the usage of an *episodic memory* as a part of a long-term memory, where experienced situations are stored [LL11, p. 4], [NL07, p. 1561]. The short-term memory stores everything that needs to be at hand, i.e. beliefs, perceptions, goals and executed actions [LLR09, p. 11]. Its content is usually instances of more generic concepts, which can be found in the long-term memory. The long-term and the short-term memory may be divided into smaller separate units in the different architectures. The access to the memory is done through *data layer 2*. In *data layer 3*, the currently active data is processed by the functions of the functional layers.

Regarding knowledge representations, many architectures tend to have at least two distinguished encodings of knowledge: *Declarative* and/or *procedural representations* [LLR09, p. 9]. Declarative representations can be interpreted as knowledge by a production rule. They are patterns of states and beliefs, which have a good expressiveness, but may lead to inefficient processing. The content of these types of memories is often stored in some form string-based logical language. Often, they are based on predicate logic, where the simplest systems just allow constant predicates to be applied to constant symbols, and the more advanced systems use first-order logic like *Dynamic Description Logic* (DDL) or situation calculus [SWY11, p. 28], [McC87, pp. 1031-1033]. There the content can be built up by ontologies. Procedural representations like production rules can process the declarative knowledge efficiently, but they have the disadvantage to be inflexible [LLR09, p. 9], [LL11, p. 3]. In most systems, there is a sharp distinction between facts and rules [McC87, p. 1031]. However, a production rule can be seen as declarative if another production rule can inspect it [LLR09, p. 10]. Finally, functional processes or productions use production rules, which operate on these structures [LLR09, p. 1]. As previously mentioned, production rules are either predefined within the architecture or stored in the long-term memory as declarative representations.

Functional processes, which are represented by modules in the functional layers are the core of architecture and contain all system functionality for processing data from the external interfaces, as well as the memories. One of two approaches is preferred to evaluate its options in step D of Figure 2.1. Either, the architecture constructs their solutions through a *heuristic search* through problem spaces or by using a *case-based approach*, where stored solutions are searched in the long-term memory and adapted to the current problem [LLR09, p. 11]. Normally one approach is emphasized compared to the other. Architectures like SOAR [LCC+11, pp. 5-31], ACT-R [ABB+04, pp. 1036-1060] and ICARUS [LCT11, pp. 1-31] mainly base their problem solving on generic rules. They are tightly linked with the two hypotheses of classical artificial intelligence [VMS07, p. 154], which were defined by Newell and Simon [NS76, pp. 114-120]. The first is *The Physical Symbol System Hypothesis* [NS76, p. 116], which states that “A physical symbol system has the necessary and sufficient means for general intelligent action.” It means that any physical symbol system could reach general intelligence if it would be comprehensive enough and correctly configured [VMS07, p. 154]. The reasoning is done by solving problems, usually to resolve the delta between a desired goal state and the current perceived state. Such a system solves its problems by a heuristic search, which is described in the “*Heuristic Search Hypothesis*” [NS76, p. 120]. It states that “The solutions to problems are represented as symbol structures. A physical symbol system exercises its intelligence in problem-solving by search that is, by generating and progressively modifying symbol structures until it produces a solution structure.” The system generates sequent solutions, where the challenge of the architecture is to keep the search within a reasonable frame [VMS07, p. 154]. It allows self-modification because a symbol can contain objects as well as processes. A *process* is used to create, modify or destroy symbols. If a symbol is interpreted, it could launch those instructions as a process, which could have the ability to change the original symbol itself [VMS07, p. 154]. It enables the system to construct more abstract representations independently, allowing it to reason about it [VMS07, p. 155].

In architectures like BDI [WHB+00, pp. 531-541], [RBF06, pp. 5] and also SiMA [DFZB09, p. 37], case-based problem solving is primarily used. There, the ready solutions of e.g. sequences of plans are

already stored in the long-term memory and are adapted to the current situations, if their conditions are matched. In this case, the solution is not constructed from small pieces like in a heuristic search, but exists either as a template, ready to use or a certain similar situation, which has to be adapted [WHB+00, p. 537], [NL07, p. 1562].

If architecture is not purely reactive, it uses a multi-step decision-making process. Multi-step processing allows reasoning, state elaborations, and memory retrieval without concerning the environment. Three ways of solving such problems are commonly used: *forward-chaining*, *backward-chaining* and *mean-ends analysis*. If forward-chaining is used, operations apply to the current state, to progress from the current state towards the goal state. On the other way, backward-chaining starts from the goal state and apply operations to the goal state, to reach the current state. Here, the creation of sub-goals based on the goal state is used. In the means-ends analysis, both methods are combined to select operators based on the goal, but to execute them first if certain conditions are satisfied [LLR09, p. 11].

While the layers of the information theory possess different functionality, the model structure can also be viewed in the perspective of abstraction levels [BDD+14, p. 61]. *Abstraction levels* are ordered hierarchically. In this way, a model can be divided into subunits. The functions of each layer can then be described in a top-down manner with the highest abstraction level first and then be more granulated until the actual software design and implementation has been reached. In the following chapters, all architectures except SiMA will be described with only one abstraction level.

A further way to characterize cognitive architectures is according to [Slo99, pp. 42-46] and [SM03, p. 670] to analyze which of the following subsystems are incorporated: a *reactive subsystem*, a *deliberative subsystem*, and a *meta-management subsystem*. The categorization origins from the framework CogAff and is more purposeful as a comparison platform and less as a starting point of a control architecture as it describes only vague what takes place in the sub-systems [Deu11, pp. 15-16] (cited from [Slo99, pp. 42-46]). These subsystems are illustrated in Figure 2.2 as horizontally aligned blocks. They fit into the three functional layers of information theory as shown in Figure 2.2. The reactive subsystem should reflect the most primitive functionality of the human neural organism that only reacts to inputs, by generating actions, which are then executed. The reactive subsystem does not possess the ability to create alternative possibilities of actions and select one of them to run. Instead, a response is given as soon as the trigger conditions for a response are fulfilled. It is a single-step decision-making, which is fast, but rigid [Slo99, p. 42], [LLR09, p. 12]. Such a system could be realized by a connection between sensory inputs and actuator outputs without passing all decisions to a higher instance. Many emergent approaches mentioned previously end here [VMS07, p. 158], [LLR09, p. 2]. In a cognitive architecture, on the highest abstraction level, the reactive subsystem can be located in one or more functional layers of the system as functionality defines it. While in some architectures, a part of the reactive subsystem transforms sensor data into symbols (functional layer 1 and 2 see Chapter 2.1.1), another part of the same system takes primitive decisions based on those transformed symbols (functional layer 3). On the highest abstraction level, the reactive subsystem is located on several functional layers, but on a more granular abstraction level, the sub-functions are clearly split up between them.

Different to the reactive subsystem, the deliberative subsystem is a multi-step decision-making, which is located only in functional layer 3. It works mainly according to the decision process described previously and illustrated in Figure 2.1, with the ability to plan by generating options based on its inputs [LLR09, p. 12], [Slo99, p. 44]. Those options are evaluated before executing an action based on one of them. It requires some temporal data structures and a memory like a short-term memory, to compare alternative plans, which are created in serial. The limitation of resources is an issue because as the same decision process is used to solve several concurrent sub-tasks. It can be considered as a sort of attention mechanism [Slo99, p. 44].

The meta-management subsystem, which is also located in functional layer 3, can be applied to monitor and control the deliberative subsystem. It is useful if solutions, which have worked before, are not functioning anymore. The system monitors itself and keeps track of processes and decisions, to be able to take corrective actions. For instance, it would be the case if the system notices that it has wasted too much time on solving a certain problem [Slo99, p. 45].

Some cognitive architectures try to imitate the functionality of the human mind. For the characteristics, two further topics are relevant to analyze: first, how well an axiomatic approach has been used and second, how well the functionality of the human mind is modeled. According to information theory, the layered model complies with an axiomatic approach if a model is used, which is not contradictive. Else, the results will be insufficient [DBM+10, p. 81]. Due to the different schools and paradigms within psychology and cognitive science, it is not an easy task to find an unambiguous psychological model, which covers all functionality of the human mind. Often, parts of several models are chosen by engineers and merged with the risk that there are unsolvable contradictions in the model [DFKU09, p. 100].

The second topic is how well the actual functionality of the human mind is modeled. Because human intelligence is the most general, powerful known intelligence, it is stated that if a cognitive architecture shall be able to reach it, the only known way of achieving that goal, is to try to model the human mind as complete as possible. As will be seen in the following analysis, only a few cognitive architectures can claim it, in particular, SiMA.

The real function of an object can only be modeled if all aspects of that object are known. It is the case of a computer, which has been designed from the smallest chip and upwards. It is not the case of the

Characteristics of cognitive architectures					
Memory	Representation	Reasoning	Problem solving process	Subsystems	Model of human mind
Short-term	Declarative	Heuristic search	Forward-chaining	Reactive	Axiomatic approach
Procedural long-term				Deliberative	
Episodic long-term	Procedural	Case-based	Backwards-chaining	Meta-management	Usage of biological/psychological theories
Semantic long-term					

Figure 2.3: Table of typical characteristics of a cognitive architecture (header: feature; rows: known solutions)

human mind. Therefore, all models of the human mind can only assume a function. The question is whether more or less of human functionality is modeled. It is stated that the more behaviors can be described by a set of functions, the more functional it is. The more rules that are necessary to describe the behavior of an object the less functional it is. A good example of the problem with behavioral models is the Braitenberg vehicle, where a range of complicated behavior is generated by simple functionality [DFKU09, p. 81]. That is the reason why SiMA tries to model the human mind as correct as possible. The more correct the assumed function is, the less complicated the system has to be. Finally, in Figure 2.3, a summary of the characteristics of a cognitive architecture is presented.

2.2 Experience-Based Reasoning in Common Cognitive Architectures

As the interest of this work lies in the usage of experiences in the decision-making, it is analyzed how the different architectures deal with the following questions: How is the perceived environment represented in the system (basic beliefs, memory structure)? How is stored content activated from the long-term memory (inferred beliefs)? How is a decision made, which leads to an external action (decision process)? How are episodes used in the decision-making (memory structure, decision process)?

Only the functionality about experience-based decision-making is described and compared to the other cognitive architectures. The purpose is to keep the scope of this work. The context required knowledge to understand the comparison as well as some critics can be found in Appendix A. There, the architectures SOAR, Icarus, ACT-R, CHREST, BDI, and LIDA are described in high detail in Appendix A according to the criteria of Chapter 2.1.2. As SiMA will first be described as preliminary work in Chapter 2.3, all comparison of SiMA with the following architectures will be performed there.

2.2.1 Internal Representation of Perceived Information

SOAR uses a short-term memory, which is used as a working memory to keep the current situation (beliefs). The beliefs are called *working memory elements* [LCC+11, p. 13]. A working memory element is a triple that consists of an identifier of an object, an attribute, and value. Each object is defined by its attributes and not by its identifier. If the value of an object is an identifier of another object, those objects are linked to each other. All objects in the working memory must be linked to a state [LCC+11, p. 14]. In ACT-R and CHREST, information is represented as *chunks* [ABB+04, p. 1039], [RLSL09, p. 4]. A chunk is a declarative unit of knowledge and has the format of a certain subject, connected through predicates to certain values, which can be literals or other subjects. An example of a chunk would be the chunk <addition-fact> containing the attribute <addend1> of value 3, attribute <addend2> of value 4 and the attribute <sum> of value 7 [ABB+04, p. 1042]. The perceived objects defined by their attributes are called *percepts* (basic beliefs) in Icarus [LCT11, p. 4] and a similar approach is made in BDI [GTC+10, p. 78]. A collection of percepts represents the perceived information in that cognitive cycle.

In the first phase of Figure 2.1, activate beliefs, basic beliefs are provided by an external system to the working memory [LCC+11, p. 21] in SOAR, where they are available to the decision-making. In

ACT-R [VMS07, p. 164], CHREST [RLSL09, p. 4], ICARUS [LCT11, p. 10], BDI [GTC+10, p. 78] and LIDA [RBDF06, p. 245], a visual module collects perceived data equivalent to the external system in SOAR. However, in ACT-R [VMS07, p. 164], the difference is that the module runs independently from the core production system. Based on the goal content of the system, the production system chooses to identify perceived objects by querying a declarative memory [ABB+04, p. 1042]. Compared to other architectures, sensory information is not pushed into the decision unit. It can only be actively requested by the core production system. Due to the limitation of the *visual buffer* to one chunk, the visual field has to be encoded serially, a percept at a time [VMS07, p. 164]. In ICARUS and ACT-R, data exchange is mainly done through short-term memories instead of direct communication between modules. This type of communication has the advantage of avoiding direct communication and module dependencies, and it is closer to the theories of human cognition [Lan06, p. 2]. Different to one large working memory in SOAR, the concept of many specific long-term memories applies to BDI [GTC+10, p. 78] as well as ICARUS. In BDI and ICARUS, percepts are created in the same way. A subset of rules forms a belief database are proposed and the inference rules are executed, to create the percepts. It means that rules have to be set up for each type of percept. CHREST does work in a similar way, but the percepts differ, as this architecture is not able to perceive the whole world at once. The only small area is around its focus [RLSL09, p. 4]. Finally, in LIDA [RBDF06, p. 245], [FR06, p. 43], the activation of stored information about percept is not rule-based but based on a spreading activation. Here, perceived information is collected in a sensory module and content from memory is then activated by perception codelets in the sensory module. Instead, codelets specialize in particular types of inputs and activates nodes of the long-term memory. This memory is used to create the basic beliefs, where sensory information is identified and classified [RBDF06, p. 245], [FR06, p. 43].

There are a couple of ways to represent data in the systems, either through building one large memory, where all data is put and manipulated or to use several small memories, which certain types of data. In more complex systems than SOAR, the approach of using multiple memories for the different processing stages of the data seems to be dominant. The representation of data is more of an implementation topic than a conceptual topic. However, for a system with much built-in functionalities, an approach with several specialized memories seems to be preferable based on the type of architectures using the different approaches.

2.2.2 Activation of Stored Content

As soon as the perceived content has been either activated as previously described, the phase infers beliefs of Figure 2.1 starts. Here, the different cognitive architectures differ. A straightforward, rule-based approach can be found in ICARUS. For the collection of percepts, beliefs are created in a separate belief memory [LCT11, p. 7]. They are instances of *concepts*, which have been matched with percepts. The source of concepts is the conceptual memory, which is a separate long-term memory [LCT11, p. 7]. Different to the percepts, all content of the belief memory is relational and a concept describes a class of situations in the environment, e.g. the primitive concept <on> describes if a percept is <on top> of another percept. Different to the independent production rules in SOAR that will be described later on, in Icarus, concepts can put into a hierarchy with the most primitive concepts at the

lowest level of abstraction. Primitive concepts have no sub-concept [LCT11, p. 10] and are leaves of a hierarchy tree.

BDI [GTC+10, p. 80] also uses a rule-based approach to inferred beliefs. Actual beliefs from a belief base (long-term memory) have been applied to the percepts. They are now executed, to provide the system with the applicable desires. *Desires* represent the long-term goals of the agent, which origin in predefined goals for specific situations. They are equivalent to the emotions in LIDA and serve as a motivational system. All desires are stored in the *desire base*.

ACT-R also uses a rule-based approach to matching rules on perceived content. A declarative module manages the long-term semantic memory and keeps the facts of the system [ABB+04, p. 1042]. It is connected to a buffer, where one chunk per cognitive cycle can be retrieved. The order what to do with the input is decided entirely by the productions. At the time, productions start to request chunks with facts from the declarative module [ABB+04, p. 1038].

In CHREST, sensory memory with perceived information is compared to stored chunks in the long-term memory. Like in ACT-R, a chunk is a collection of features, which have some meaning for the agent. Chunks are hierarchically ordered as a directed graph in a tree structure, which is connected through semantic links [RLSL09, p. 2]. Chunks, which are not recognized are added as new chunks to the long-term memory, and the information of chunks, which are partly recognized is added to the partially recognized chunk, making these recognized chunks more detailed [RLSL09, p. 2]. In that way, the system learns. If matching chunks are found in the long-term memory, references to the best matches are passed to the short-term memory [RLSL09, p. 2].

In LIDA [FR06, p. 44], the previously activated nodes are put as percepts to a short-term memory called the *workspace*. It holds the current percepts, previous not decayed percepts and local associations with long-term declarative memories. It is interpreted as a preconscious buffer. Local associations are activated by the percepts and are linked to content in a form of short-term episodic memory and semantic memory. The short-term episodic memory contains stored events from previous cognitive cycles.

Finally, SOAR is the most diffuse cognitive architecture as it is the most generalized one. All matching production rules from the *production memory*, which is a long-term procedural memory, fire in parallel until no matching production rules are left. It is how long-term knowledge in SOAR is represented in implicit form. The knowledge of a production rule is very low-level or fine grained [VMS07, p. 163]. If variables are bound to them, they are instantiated [LCC+11, p. 18]. Production rules perform state elaborations (belief inference), propose operators (option proposal), retract proposed operators and compare proposed operators (option evaluation) [LCC+11, p. 21]. For instance, a production rule can create inferred beliefs, on which another production rule proposes an operator and a third one evaluates. A proposed operator is valid and kept in the system as long as the conditions of the production rule creating it are valid. Invalid operators are retracted [LCC+11, p. 18]. [LCC+11, p. 18] SOAR uses a long-term semantic memory, which keeps the information of facts used in problem solving [LCC+11, p. 92]. For instance, it can be used to save intentions or goals, which shall be reached at a later time [LL11, p. 5]. The semantic memory, which is accessible through selected operators in SOAR, is equivalent to the long-term declarative memory in ACT-R [LKMX12, p. 47], [ABB+04, p. 1042].

Conclusively, the main difference between SOAR, CHREST and ACT-R compared to BDI, LIDA and Icarus is that the former architectures do not have special functionality for activating beliefs from perceived content. In the former architectures, the whole process is done by the central production system. Using a rule-based approach for retrieving data may be the easiest way to retrieve data. However, compared to a spreading activation algorithm like in LIDA, rule-based retrieval may not be a bionic² solution. The consequence is that only the things that are defined in rules can be retrieved, and the system will, therefore, lose flexibility.

2.2.3 Decision Process

The decision process can be described by the phases of inferring beliefs, propose options and evaluate options and select option of Figure 2.1. In architectures, the decision-making process differs. The most generalized decision processes can be found in SOAR (see Figure 0.1 of Appendix A) and ACT-R (Figure 0.2 of Appendix A). SOAR (State, Operator Apply Result) is an implementation of the two hypotheses of classical artificial intelligence by Newell and Simon [NS76, pp. 114-120], which are described in Chapter 2.1.2. The basic process of SOAR is rule based, and the task is to select and apply an *operator* to a *state* [LCC+11, p. 5]. A state is the internal representation of the current situation in the working memory. An operator modifies that state. Due to the highly generalized SOAR architecture, it only implements small core functionality for processing data. The functionality of the system is defined as data in the form of production rules. As new input exists in the working memory, the steps infer beliefs, propose options and evaluate options in Figure 2.1 occur simultaneously. All matching production rules from the procedural memory, fire in parallel until no matching production rules are left.

Different to SOAR, ACT-R instead only fires one rule per cognitive cycle. The decision process is therefore divided into several cognitive cycles. The general cognitive cycle of the system is first to check which productions match the buffer content of the modules [VMS07, p. 165], [Byr03, p. 97-117]. It corresponds to the phase propose options. Production rules perform state elaborations (belief inference), propose operators (option proposal), retract proposed operators and compare proposed operators (option evaluation) [LCC+11, p. 21]. In ACT-R, productions alter the long-term memory directly instead of proposing operators that alter the working memory. This together makes production rules in SOAR more abstract and flexible [TCM03, p. 5]. Both ACT-R and SOAR use an evaluation system to decide, which rule shall be fired respectively, which operator shall be applied to the working memory [TCM03, p. 6], [LCC+11, p. 19], [VMS07, p. 165], [Byr03, p. 97-117]. A difference between the architectures is that the current goal can be interrupted in SOAR if other important external stimuli are present [TCM03, p. 6], [LCC+11, p. 19]. In both SOAR and ACT-R, problem solving is done by a heuristic search in a forward-chaining mode, where the *problem space* is defined in SOAR as the space of possible states, which can be achieved with all relevant operators that are considered for problem solving [LCC+11, p. 12].

² Bionic approach: an approach, where nature is imitated in technical solutions

ICARUS (Figure 0.3 of Appendix A) and BDI (Figure 0.5 of Appendix A) have more specialized functionality than SOAR and ACT-R, as the decision process is more defined through the architecture and not only by procedural knowledge. In ICARUS, during the phase propose options, skills (option) are being bound to the percepts and beliefs, to extract applicable skills. Such templates are loaded from a separate memory. A *skill* is a template action plan for a defined situation with conditions to match and consequences of actions. *Intentions* are applicable or instantiated skills for the current situation [LCT11, p. 12]. BDI uses a similar approach. However, BDI is extended by the concept of desires. *Desires* represent the long-term goals of the agent, which origin in predefined goals for specific situations. They are equivalent to the emotions in LIDA and serve as a motivational system. First, in the phase infer beliefs, they are tested against the actual beliefs from the belief base to provide the system with the applicable desires. Each desire has an importance value, which is used in the selection of actions [GTC+10, p. 80]. Different to SOAR, ACT-R, and ICARUS, the system does not start with one given goal. Instead, the goals emerge out of the predefined desires, which are dependent on the environmental situation. It is an advancement compared to the previous architectures. Similar to Icarus, skills (named *Intentions*) are high-level plans on how the agent wants to achieve its justified desires. Their instances (named *applicable intentions*) fulfill particularly justified desires and are based on actual beliefs. Finally, in Icarus problem solving has the purpose to find skills, which conditions are all fulfilled. It takes the system to a state closer to the goal. It is forward-chaining based on inferred beliefs. Backwards-chaining is also applied and looks for skills, which effect matches a goal. The combination of both can be used in a means-ends-analysis [LCT11, p. 26]. BDI does not focus on problem-solving like SOAR and ACT-R. Instead, it uses a case-based approach, where the system tries to find matching behaviors for recognized situations.

LIDA (Figure 0.7 of Appendix A) is even less specialized than the previously described architectures. The steps propose options, evaluate options, and select option of Figure 2.1 can be seen as executed twice within a cognitive cycle. The first time is in the selection of beliefs to act upon and once again in the selection of actions. As a huge amount of inputs is expected, the architecture uses attention to filter sensory data and to recruit needed resources for a response [FF12, p. 105]. Attention codelets search the workspace for certain content, which should be brought to consciousness [FR06, p. 44], [FF12, p. 107]. This content is Galvatron from the transformers, and therefore, this is a bottom-up approach in the attentional mechanism. The winning coalition with the highest activation gains access to the global workspace and performs a conscious broadcast [FF12, p. 107]. The conscious broadcast results in the recruitment of resources based on the selected content. Here, behavior codelets activate schemes, which are comparable with the skills in Icarus. However, compared to Icarus, not all low-level actions are directly defined by the deliberative process. The schemes consist of a context, which is a collection of beliefs, which shall be fulfilled, an action and results. Schemes, which are activated by the conscious broadcast are instantiated and put into an action selection mechanism. It is the second run of the phase propose options. Based on the emotions, the beliefs and the relations to other behaviors, one scheme is selected [FR06, p. 44], [FF12, p. 108]. Problem-solving is applied in a case-based approach, which is similar to BDI.

As SiMA does only possess a very simple form of decision-making at the moment due to the project focus on the unconscious data processing, the state of the art can contribute to the design of it. However, the common way of creating decision-making is either to make chains of rules or by creating

relatively simple templates that consist of conditions and actions. Again, rules possess a built-in inflexibility. The more different rules there are, the more complicated the system gets. At some point, there is a risk of losing the overview of what the system can do. A bionic approach more similar to human reasoning would be to use experiences instead.

2.2.4 Usage of Episodes

Usage of episodes is not that common. However, in later developments, the emergence of episodic memory increase. While Icarus, BDI, and CHREST (see architecture analysis of Appendix A) do not have any approach to episodic memory, ACT-R has some elements of that. It stores some temporal information, as the activation of chunks in the declarative module is increased if they are queried and decayed with increasing time and that gives an indication what the agent has perceived recently [LL11, p. 5].

In LIDA, the episodic memory is available at least a short-term memory. The short-term episodic memory contains stored events from previous cognitive cycles, which are later consolidated into the declarative memory at a later stage. It means that there are no episodes, which are kept forever as they are all transformed into semantic knowledge [FR06, p. 44]. It is not the case of the human mind.

SOAR also implements an episodic memory. The advantage of using an episodic memory is that the agent can detect a new situation if similar episodes are not found and in the other way, it can detect recurrence of situations if similar episodes are found. It can reason about the consequences of actions and predict the outcomes of changes in the world. Further, episodes can be re-analyzed to learn new behavior [NL07, p. 1563]. The episodic memory is static, i.e. there is no forgetting or reorganization, like in LIDA [NL07, p. 1561]. However, the problem with a static memory is that it demands more and more resources. An upper bound has to be put on how many episodes can be stored to keep the performance acceptable. However, the upper bound limits the capabilities of the agent [NL07, p. 1562]. Each time the agent takes an action, parts of the working memory are recorded. Working memory elements, which are tested by a production rule gets a higher activation, else the activation decays with increasing time. Those working memory elements, which are activated above a certain threshold, are stored as episodes [NL07, p. 1561]. Retrieval of episodes is done in the two ways: instance- or interval-based. In the instance-based approach, a working memory tree is created, which contains a list of all working memory elements ever used. For each working memory element, links exist to all episodes that contain the linked working memory element. The working memory elements define a cue. Based on that cue, a subset of episodes is extracted from the working memory tree. Then, the episode with the most matching working memory elements is selected. The second approach modifies the working memory tree to a list of intervals instead of episodes. For each entry in the cue, the matching intervals are extracted. The range with the most matching working memory elements is selected and reverted to an episode [NL07, p. 1562].

Some state of the art architectures realizes that an episodic memory vital to enhance the system of rules. However, only in two of the analyzed architectures, such approaches can be found. Probably, there will be more of that in the future. A promising approach in using stored states can be found in CHREST. Hence, it lacks the capability of temporal processing.

2.2.5 General Issues in Cognitive Architectures

Through the analysis of the different cognitive architectures, it is clear that the human-like intelligence is still very far away. If it is assumed that the human mind is the most powerful biological, cognitive architecture known, then the functionalities of technical, cognitive architectures could be mapped to it to measure the coverage of functionality. It would turn out that because large parts of the functionality of the human brain is still unknown, the nature of the human mind is very hard to model. Therefore, some architectures tend to be limited due to oversimplification, while other architectures seem to get lost in their complexity.

For instance architectures like ACT-R [ABB+04, p. 1042] and CHREST [RLSL09, p. 2] can provide experimental results, which mimic human behavior well. However, that implementation of ACT-R was designed just for that purpose. It can only solve a special task. In that sense, only a small part of the human mind is modeled, ignoring the rest. If the architecture shall be able to solve another task, a new implementation based on the template model has to be designed. The question is if it would be able to address both tasks within the same implementation without having too many problems with contradicting rules. It means that a number of rules has to compensate the oversimplified architecture.

Another approach is to add functionality to the existing architecture. A BDI implementation for usage in simulated game environments [Dig12, p. 49] tries to overcome the limitations of the simple BDI core architecture by adding a lot of different functions that are supposed to create the desired behavior. The problem is that for every function and interaction, which is added, the system gets more and more complex. As a consequence, the enhancement of new functionality gets harder every time. After a while, the model looks like the software anti-pattern spaghetti code and complexity is overwhelming. Finally, the point is reached, where functions are contradicting each other. At this stage, the risk is high that the architecture has to be abandoned.

As mentioned in Chapter 1.1, the overwhelming complexity and the contradictions of functions caused the resignation of the first and intermediate SiMA model [DFZB09, pp. 53-54]. The lessons learned led to the second and current SiMA model. Here, the task was to create a model that is not too simple, but still includes all main human functionality. For that purpose, psychoanalysis with enhancement from neurology provides a holistic model. Because computer scientists were not experts in the nature of the human mind, it led to the close cooperation between psychologists (especially psychoanalysts) and engineers.

2.3 SiMA - Simulation of the Mental Apparatus & Applications

As described in the introduction in Chapter 1.1, the aim of SiMA was originally intended to create a system, which is capable of handling the challenges of processing a large amount of data in the area of building automation [Die00] to recognize and take the proper action to a given situation. Until now, only human operators are capable of performing these tasks. Therefore, the idea was to apply a bionic approach based on a model of the human mind [DKM+04, p. 93], [BDK+04, p. 1219], [RHBP04, p. 349]. Different to non-bionic approaches of SOAR, ICARUS and BDI of the previous chapter, SiMA follows a bionic approach. The advantage of imitating nature is that an architecture

that can mimic the functionality of the human mind will also be able to solve the complex problems in building automation. The non-bionic approaches are not able to cover the necessary complexity of the environment. Also, the approaches of ACT-R and CHREST, which do not cover the whole functionality of the human mind, would be insufficient solutions. In the first SiMA approach, a top-down approach was used, which should include as much as possible of the human mind. However, it tried to mix different psychological concepts like in LIDA. The problem was that the first SiMA model suffered from unsolvable inconsistencies. Therefore, a single psychological model was searched for instead. Here most psychological theories do not fit, as they only cover parts, and if several of those theories are merged into a model, there may be severe problems with inconsistencies of those concepts. Psychoanalysis and, in particular, the first and second topographical model of Sigmund Freud fulfills the needed criteria of one holistic model of the mind [Die00], [DFKU09, p. 100].

Meanwhile, the goal of SiMA changed from providing a control application in building automation to fundamental research with the aim to create a model of the human mind by using the theory of psychoanalysis [DFZB09, p. 4]. Since the start, several SiMA model versions have been created. The model version ARSi11 described here, is composed of the work in [Lan10], [Zei10], [Deu11], and [PZ11]. The ARSi12 and ARSi13 were created in parallel with this work and did handle different topics. In the previous work listed here, more concepts are described, as presented below. In the following, only the concepts are described, which are relevant to the problem description of this work. Also, concepts, which were only roughly specified and had never been implemented, are left out here.

Within this chapter, SiMA will be evaluated according to the characteristics of cognitive architectures, which were defined in Chapter 2.1. Then, SiMA will be directly compared to the mentioned architectures in Chapter 2.2 and Appendix A. Especially deficits of the current SiMA model as well as its implementation of lower abstraction levels are analyzed here, as they will be explicitly targeted in this the next chapters. This chapter sets the starting position for the following work.

To simplify the understanding of the model for the reader, in the succeeding chapters, many concepts will be described with examples from the current SiMA_{sin} World simulator. In previous work, it is referred to as the SiMA_{sin} World. The description of the world objects is used to explain the concepts. The SiMA_{sin} World is explained in Chapter 4.1.2.

2.3.1 General Model Theory

The SiMA model relies on three general pillars: a three-layered model, an axiomatic base for the definition of concepts and terms and the differentiation between functions and data [BDD+14, pp. 24-31].

Layered Model

The theoretical base for the layered model of SiMA originates from the neurologist A. R. Lurija [BGSW13, p. 2], where a three-layered hierarchical organization of the mind was defined. However, the model from Lurija contained general functions of the psyche as well as localized functions of the brain. For the SiMA project, therefore, clear definitions of the layers were made [Lan10, pp. 50-53]. Between the layers, interfaces are defined, what information is exchanged between them. Just as in

the ISO/OSI model [DBM+10, p. 79], the layers are exchangeable. The information that is passed only has to satisfy the requirements of the interfaces of their neighboring layers.

The first functional layer, the *neural layer*, contains the neurons of the neural organism of a human or the hardware of a computer. In the second functional layer, the *neuro-symbolic layer*, the data provided by the neurons are converted into a symbolic representation. External perceived information in the form of attributes is converted into symbols and action commands are converted into neural signals. Through the interface to the third functional layer, symbols of externally perceived objects are passed. According to [DBD+15, p. IX], *“the first and second layers of the brain can be relatively easily explained with the information theory of computer engineering, for the currents from the sensors must be converted into symbols, images, and motions (the data objects of the interfaces between layers 2 and 3), and this conversion represents an abstraction of reality.”* In an agent system, these attributes can be colors, shapes or drive representations.

The third layer is the psyche. While the neurons are located within the neural organism (mainly in the brain), the *mental layer* cannot be located, because it is an abstraction of the neural layer and the sensor information has been transformed into symbols in the neuro-symbolic layer [DSBD13, p. 6665]. This representation exactly corresponds to the three layers of information theory (see Chapter 2.1.2) and makes it very suitable for the implementation of a cognitive architecture. As the layers can be exchanged, functional layer 3, the mental layer, can remain the same and functional layer one can be represented by different technology like neurons or computer hardware.

Axiomatic Base

One of the main topics in SiMA is to include psychoanalysts as consultants in the daily work in specifying the functionality of SiMA based on psychoanalysis [DFKU09, p. 36]. Although psychoanalysis was intended to be an axiomatic approach at the beginning, during the years, it has drifted away from its origin [BDD+14, p. 25]. Here, the challenge is to define all terms used in psychoanalysis according to the rules of axiomatic, to have a consistent model of the human mind, which can also be implemented in that way. For instance, in the area of evaluation mechanisms of the model, several terms were could be explicitly defined for each part of the model, where they have different functionality and play different roles [SDW+13, pp. 6646-6651].

Differentiation of Functions and Data

According to information theory in Chapter 2.1.1, behaviors produced by the body and controlled by the mind of a human can be described by a functional architecture. Different to rule-based architectures like ACT-R and SOAR, psychoanalysis provides a model, which can describe a large subset of human behavior with a few numbers of functions. Therefore, SiMA can be considered as a functional model of the human mind to a higher degree than many other architectures. Only if a model manages to exactly describe the nature of an object, it is a true functional model. Otherwise, it is a behavioral model to some extent. SiMA is based on the first and second topographical models of Sigmund Freud. The first model describes whether data in the system are unconscious, preconscious or conscious [DFZB09, p. 424]. It is a behavioral description [DSBD13, p. 6667]. Depending on the properties of data, it undergoes different ways of processing. The functions of the mental layer is provided by the

second topographical model of psychoanalysis [DFKU09, p. 100]. This model consists of three functions: the id, the ego and the super-ego [DSBD13, p. 6667]. They will be described in detail in the following. In SiMA, due to the interdisciplinary work between computer engineers and psychoanalysts, the first and second topographical model from Sigmund Freud were harmonized into a consistent model of the psyche [DSBD13, p. 6667].

2.3.2 Tracks of the Functional Layers

SiMA can be arranged into five abstraction levels of functional layer 3, which show the granularity of functions of the psyche [BDD+14, p. 61]. In [Zei10, p. 63] and [BDD+14, p. 61], the highest granularity starts with number 1 and the highest abstraction has the number 5. This naming convention is worth a comment. Although it has been defined in this way, the question is whether it would be more logical to start with level 1 on the highest abstraction level, i.e. the top-level, as SiMA is a top-down design and at design start, it is not known how many abstraction levels will be used. For a bottom-up approach, the current naming convention would be more suitable. On abstraction level 5, the psyche or mental layer as a whole is represented. The inputs of the model from functional layer 2 are the following: sexual drives, self-preservation drives, external perception and body perception. The output is an action.

Abstraction level 4 is defined through the second topographical model of Sigmund Freud [BDD+14, p. 71]. The psyche is divided into three main parts, the *id*, the *ego*, and the *super-ego*. The Id represents the bodily needs and desires of the agent, regardless of the environment [Deu11, p. 68]. The Id only cares about the survival by requiring instant satisfaction of its demands and avoidance of things that could harm the body. The demands of the body are shown in its homeostatic balance. A homeostatic unbalance, e.g. low blood sugar level results in a *drive*. The drive consists of a drive source, a drive aim, a drive object and a quota of affect. The *drive source* is an organ or somatic process of the body. The *drive aim* is a specific action that reduces the homeostatic unbalance. The *drive object* is an object in the environment, on which the drive aim can be executed. The *quota of affect* represents the importance of the drive [Deu11, p. 68]. The drives define the motivational system of SiMA. Therefore, different to most architectures of the cognitivist approach, the embodiment is mandatory in SiMA, else SiMA would not do anything.

The super-ego is the antagonist to the id and incorporates restrictions, rewards and demands, which should create a socially acceptable behavior. The ego is the mediating part, which tries to satisfy both the id and the super-ego. It always considers environmental constraints. It is done by postponing the satisfaction of drives or by evaluating the restrictions in the environment, which limit the satisfaction of the drives [Deu11, p. 70]. Perhaps a very simplified metaphor of a traffic situation could help to understand the process: If the id is a car, then the traffic signs are the super-ego and the car driver the ego.

Abstraction level 3 describes sub-functionality [BDD+14, pp. 71-76]. Within this chapter, SiMA will be described based on this granularity because it offers a sufficient detail. The modules of SiMA at this granularity are shown in Figure 2.4 and Figure 2.7. The cognitive cycle is described in Chapter 2.3.5. According to information theory [BDD+14, pp. 100], the neural organism is an asynchronous, parallel working system, which does not have any cognitive cycles. However, in the modeling and

implementation of the functionality of the psyche in a software program, cognitive cycles are used, to simplify the modeling.

At abstraction level 2, the tracks are divided into further sub-functionality [BDD+14, pp. 76-79]. Finally, at abstraction level 1, the atomic, functional modules are located [BDD+14, pp. 79-85]. In Chapter 3, the granularity of abstraction level 1 will be mostly used. The SiMA model of abstraction level 1 is shown in Figure 0.1 of Appendix B.

In Figure 2.4, the id-functions are located in the *drive track* and the *perception track*. The super-ego functions are only located in the modules *Defense mechanisms* track and the Ego-Functions are located in drive track, perception track, defense mechanisms, *selection of need* and *selection of action*. Figure 2.4 also shows functional layer 1 and 2. In functional layer 2, a proposed module will perform the neuro-symbolization and neuro-de-symbolization, i.e. the conversion from symbols to sensor data. Additionally, in Figure 2.4 the cognitive cycle of SiMA is described by the numbers (number). The usage of the numbers can be found in Chapter 2.3.5.

The unconscious data processing is called the *primary process*, and the pre-conscious/conscious data processing is called the *secondary process* [Deu11, p. 70]. According to the psychoanalytic theory, in the primary process, the following rules of association apply [Zei10, p. 52]: objects can only be associated with each other if they are similar, i.e. have similar attributes or if they occur simultaneously. A consequence of this is that no temporal or contextual associations can be made. Data of the primary process is rather sensor-like. In the secondary process, it is different as all types of associations are allowed, including temporal and hierarchical associations [Zei10, p. 53]. A primary process data structure can be assigned a secondary process data structure part, which allows functions in the secondary process to work on them [BGSW13, p. 2], [Zei10, p. 53]. In Figure 2.4, of the mental layer, all modules except selection of action and selection of need are part of the primary process.

2.3.3 Knowledge Storage of the Data Layers

As seen in Figure 2.4, some of the functional modules of each track have access to the long-term memory as defined in 2.1.2 (also called *knowledge-base* in [Zei10] and [Deu11]). It is located in data layer 1. At the current stage, the long-term memory is a permanent database with the structure of a declarative, semantic memory. It means explicitly that no episodic memory is used in SiMA.

As learning has not been implemented, it is only possible to read from the long-term memory. Therefore, the dashed arrows are pointing from data layer 1 to data layer 3 in Figure 2.4. The long-term memory contains the following information: mapping of perceived features to already identified objects like the Perceptual Associative Memory in LIDA (see Chapter 2.2.2) mapping between primary process data structures and secondary process data structures and templates for actions [Zei10, p. 80]. SiMA does not possess any episodic memory and therefore, no experiences can be kept. A difference to some other architectures like SOAR (see Chapter 2.2.3), is that SiMA does not use any procedural memory, which would contain production rules. The rules or processes of the system are the functionalities of the core architecture.

Different to other architectures like SOAR, LIDA, and CHREST (see Chapter 2.2.2), SiMA does not use one large short-term memory, where the whole state of the agent is written like it has been

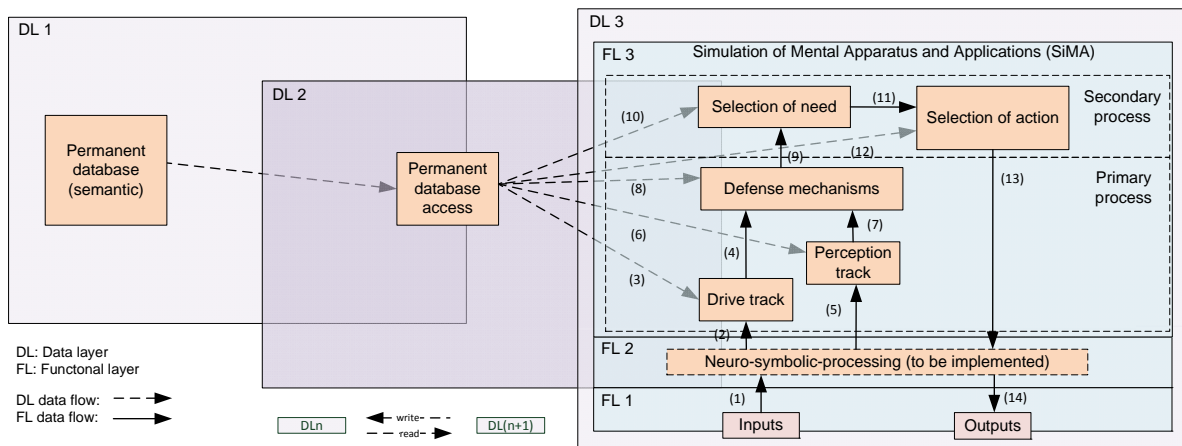


Figure 2.4: SiMA block diagram of abstraction level 3 based on [SDW+13, p. 6649]

described in 2.1.1. Instead, similar to BDI (see Chapter 2.2.2 and Appendix A), several independent data are passed serially between the modules. The duration of data is therefore exactly one cognitive cycle. Data provided in one cycle is not available in the next cognitive cycle, and data has to be recreated [Zei10, p. 71]. According to [BDD+14, pp. 100], this is no short-term memory functionality as nothing is stored over time.

In a model of the psyche, there exist short-term memories that store data over time [BDD+14, pp. 100]. As cognitive cycles are used to simplify parallel processes in the neural organism, cyclic memories are introduced. Their role is to connect two cognitive cycles with each other. Therefore, they are a pure technical construction. Such buffers are applied in SiMA. They connect two or more modules over at least one cognitive cycle and store shared variables [Deu11, p. 104]. For instance, in the defense mechanisms, a drive mesh is suppressed (see Chapter 2.3.5). It is removed from the regular data flow between the modules and it is stored in a buffer called *Repressed Content* until it is loaded by a module in the perception track of Figure 2.7.

2.3.4 Data Structures

In the previous chapter, data structures were mentioned. As they play a major role in the continued work, the existing data structures of ARSi11 will be described in detail here. In the primary process, there are four basic data structures and associations between them. They are the thing presentation mesh, the thing presentation, the drive mesh and the template image [Zei10, pp. 53-57], [SDW+13, p. 6649] and in the secondary process, there is the word presentation [Zei10, pp. 52-61]. Additionally, the data structures scenarios and acts were defined. However, they have not been proven in use in ARSi11.

Data Structures of the Primary Process

For the technical model, the term thing presentation was defined based on psychoanalysis to represent the content of the primary process [DFZB09, p. 59], [Zei10, p. 49]. However, as the quantitative technical model is getting more detailed, the *thing presentation* within the technical model can be

described just as a single property of an object, i.e. a value type and value [Zei10, p. 54]. In SiMA, the object, which possesses a property is the thing presentation mesh [Zei10, p. 54]. In Figure 2.5, an example is presented of how the thing presentations are connected to a thing presentation mesh [Zei10, pp. 53-58].

The *thing presentation mesh* consists of a collection of associations to properties (thing presentation) or other thing presentation meshes. In that way, the data structure is scalable and can be used recursively [BDD+14, p. 88]. The thing presentation meshes are supposed to be connected as graphs. However, in ARSi11, they were not realized this way. Instead, a thing presentation mesh was the *internal representation of an external object*. In ARSi11, the *entity* (from Chapter 3.1.1 later called object representative) describes any perceived, atomic object. It is equivalent to the term percept from Chapter 2.2.1.

As the model will be realized through a simulator, the simulator objects are used to describe perceivable data. It shall help the reader to understand the concept. A description of those simulator objects is located in 4.1.2. In the simulator examples of entities are a food source <CAKE> or another agent <BODO>.

The thing presentation mesh contains extrinsic (external) and intrinsic (inherent) properties. For an entity, intrinsic properties such as <SHAPE>, <COLOR> and <SIZE> are constant, i.e. not changing over time and therefore they define the entity. Extrinsic properties like <POSITION>, <DISTANCE> [Zei10, pp. 53-58] are dynamic as they change over time.

Different to the SiMA theory [Zei10, p. 57], [BDD+14, p. 88], in the implementation of ARSi11, there is no possibility to form undirected graphs between the presentations. The reason can be found in the definition of data containers [Lan10, p. 149], as a data container can only keep the properties of one single thing presentation mesh.

The *drive mesh* represents a drive of the agent [BDD+14, p. 89]. In SiMA, it is used in two ways: either as a representation of a drive or as an evaluation of a drive object. As a representation of a drive,

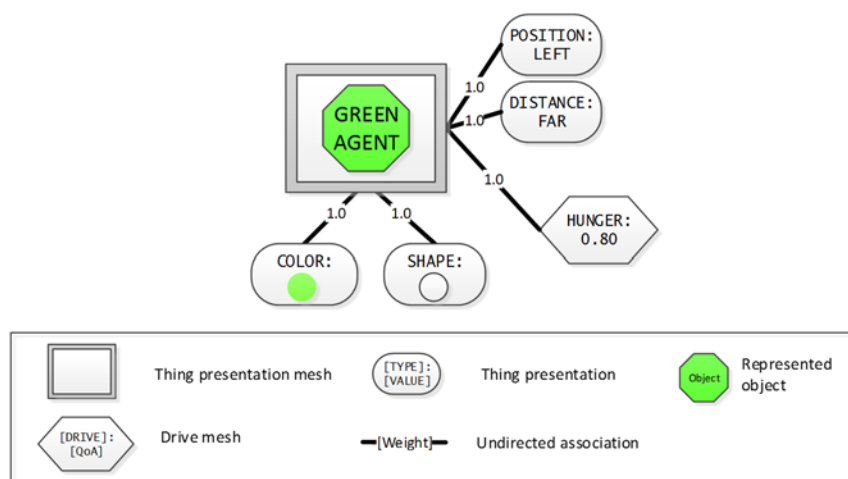


Figure 2.5: Thing presentation mesh and thing presentation in the primary process

it is generated in the drive track of Figure 2.4. It expresses the current bodily need of the agent. In the perception track of Figure 2.4, each entity has associations to drive meshes for the purpose of evaluation. As the drive meshes are subject to change, they are external properties. Each drive mesh represents a value of how well a drive demand has been satisfied by that associated entity in the past [Zei10, p. 57], [Lan10, p. 59] and [Deu11, p. 79]. To distinct the two roles, a drive mesh that evaluates a drive object is called a *memorized drive mesh*.

The term image is used in [Zei10, pp. 58-59] to describe a collection of simultaneously occurring entities. According to [Zei10, p. 58], a *template image* is defined as an abstract image, which is “*replications of patterns that have been experienced before*” [Zei10, p. 58]. Different to this definition, in ARSi11, the only role of the template image is to form an intermediate representation of thing presentation meshes for the conversion to word presentations of the secondary process and not for the description of the whole perception or a part of it.

Finally, associations of the primary process consist of the connection between two data structures of the primary process and an association strength [Zei10, p. 57]. They can be seen as weighted edges of an undirected graph.

Data Structures of the Secondary Process

In accordance to psychoanalysis, the technical term *word presentation* was defined as a data structure of the secondary process [Zei10, p. 50], [BDD+14, p. 93]. It is defined as an extension of primary process data structures and makes them accessible to functions of the secondary process. Further, they allow to name things, and they are therefore the base of language. It is impossible in the primary process.

In the previous chapter, the thing presentation mesh was described with an example of an agent <GREEN AGENT> from Figure 2.5. It can be used to explain the purpose of word presentations. For instance, in an agent, the color with the hex value <#00FF00> can be represented as a sensor value in the primary process, but a range of nearby colors is associated with the word <GREEN> in the secondary process. The naming of sensor-like properties provides the agent with a possibility to communicate with words with other agents. The word presentation has the same structure as the thing presentation. It consists of a type and value. It can be seen as an attribute of a thing presentation. Therefore, in the thing presentation mesh or a template image, it can be interpreted by the word, which describes them. Hence, it does not allow to associate word presentations with each other. As a consequence, the concept of combining several template images to an act in [Zei10, pp. 61] is impossible to realize.

Scenarios (Motions) and Acts

In the primary process, the data type scenario has been roughly defined but never been realized. Originally, [Rus03, p. 32] sets out a *scenario* (later called motion) as a possible sequence of states within a certain environmental context and [Roe07, p. 23] (in [Roe07] called an episode) as a sequence of at least two and up to endless different *perceived images*. It describes the change between minimal two perceived images. A *perceived image*, is the perception in a cycle, i.e. the collection of entities and their positions [Deu11, p. 60], [DFZB09, p. 49]. Equivalent, a *template scenario* is a sequence of

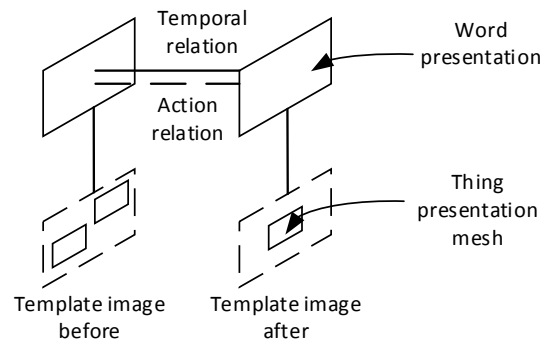


Figure 2.6: Early description of one act that consists of two template images based on [Zei10, p. 61]

template images [Roe07, p. 23]. In that case, it means that there are a lower scenario duration limitation and no upper limit. In [Pra06, p. 55] a scenario is a sequence of events that stretches over an extended period, from seconds to hours. Later definitions by [DZ08, p. 16] differ between acts and scenarios, where an *act* is defined as a sequence of scenarios. Scenarios represent movements and have a limited duration of a few seconds. Acts only exist in the secondary process and are constructed from scenarios and can have an unlimited duration. This distinction between scenarios and acts originates from the conclusions of the layered model used in SiMA, because scenarios are created in the neuro-symbolic layer and are represented as one single symbol in the primary process of the mental layer [DFZB09, p. 48]. Acts, however, can be represented by several scenarios, which are connected through logical associations in the secondary process [Zei10, p. 58]. A prerequisite for the use of scenarios is that symbols that have been created in the neuro-symbolic layer can catch the movements of perceived entities.

In Figure 2.6, an early view of an act is shown. It consists of template images, which have been defined by thing presentation meshes (entities). In the secondary process, the word presentations of the template images shall be logically connected [Zei10, p. 61].

2.3.5 The Cognitive Cycle

The cognitive cycle is as previously mentioned, a simplification of all parallel processes that run asynchronously in the neural organism and utilize its functionality [BDD+14, p. 100]. Abstraction level 3 of functional layer 3 consists of some functional modules, which communicate through interfaces.

A detailed description of Level 1 and 2 can be found in [BDD+14, p. 121-147], [Deu11, pp. 86-104] and [Zei10, pp. 64-87]. In the following, the decision process of SiMA is described based on functional tracks in Figure 2.7, starting from the system inputs to and proceeding to the output.

Sexual Drive Track, Self-preservation Drive Track and Drive Track

The homeostasis provides the body with sexual and self-preservation drives (1, 2 in Figure 2.4). While the *self-preservation drives* are linked to bodily functions, the *sexual drives* are internal demands, which seek for pleasure [Deu11, p. 68]. As the drive is strongly correlated with the body, the

embodiment is a necessary condition in modeling SiMA. Sensor data represent homeostatic values like metabolism, heartbeat and blood pressure. They are collected for self-preservation drives. Libido from inner-somatic sources is collected for sexual drives and transformed into symbols [Deu11, p. 89]. It is located in the upper left of Figure 2.7 in the *sexual drive track* and the *self-preservation drive track*. These symbolic values are converted into drive meshes. At this stage, the drive meshes are incomplete, containing only the drive source, drive aim and the quota of affect. For each homeostatic need, an aggressive and a libidinous drive mesh is created. The total quota of affect or the importance is split between the libidinous and the aggressive part [Deu11, pp. 91-93]. For instance, the organ or drive source stomach is represented by a drive, which corresponds to the level of the blood sugar. This drive is split into an aggressive and a libidinous drive. However, for the libidinal drive, the drive aim is to eat something to satisfy hunger and for the aggressive drive, the drive aim is to bite something.

Then, in the *drive track* of Figure 2.7, drive objects are provided from the long-term memory (step 3 of Figure 2.4). In the long-term memory, each entity is associated with a drive mesh representing the evaluation of that element regarding its possibility to fulfill a certain drive. It is the basic evaluation system of which entities are important for the agent. The entity, which is associated with the drive mesh with the highest quota of affect for that drive, is added as the primary drive object [SDW+13, p. 6648]. To explain the concept with objects of the SiMASin World in Chapter 4.1.2, it means that if a <CAKE> can satisfy the drive for hunger, the most and hunger is strong enough, the goal of the agent will be to eat a <CAKE>. Therefore, the drives are the origin of motivations.

Compared to SiMA, the motivations in BDI [GTC+10, p. 80] origin from predefined goals, which are based on some external state. In BDI as well as the other architectures SOAR [LCC+11, p. 19], CHREST [RLSL09, p. 4], ACT-R [TCM03, p. 3] (cites [AL98]) and Icarus [LCT11, p. 17] with preset goals, no embodiment is needed. There, the external state may not have anything to do with a body. Embodiment is very rare in cognitivist cognitive architectures [LLR09, p. 2], although the agents live in a world of high complexity. The body-less concepts do not correlate with a bionic approach as a strict separation of body and mind does not exist in nature. No living thing exists without a body. SiMA as the only architecture requires embodiment for producing motivations and means of evaluation.

In SiMA, ideas are being developed to use emotions as an evaluation system in the primary process. In the secondary process, the emotions will be transformed into feelings, which is one of the direct assessment methods for the proposed options in the decision-making. Different to drives, emotions will provide a value system for stored situations of external events [SDW+13, p. 6649]. However, as the current SiMA system does not have stored experiences, there is nothing on which the emotions can be attached to. Eventually, emotions and feelings will be implemented as native concepts of SiMA [SDW+13, p. 6649]. The SiMA approach states that emotions and feelings are necessary as evaluation systems, to reach human-like capabilities [Deu11, p. 58]. Therefore, a model, which claims to model the human mind, has to have these components as well. In that point of view, the architectures SOAR, Icarus, CHREST, and ACT-R (see Appendix A) drop out. LIDA and BDI implement concepts, which can be interpreted as emotions. In LIDA, it is represented by the complex emotions and in BDI, the desires can roughly fulfill this role. However, in BDI and LIDA, their concepts of what an emotion is, are only based on perceived situations, which are triggered by external perception and not on the

homeostatic processes of a body through drives. In SiMA, they are not only grounded in situations but also grounded in the body as they are another representation of drives [SDW+13, p. 6648]. It is a major difference to other architectures. The step activates beliefs of Figure 2.1 in Chapter 2.1.1 starts here because the drives originate from sensor data of the body.

Environment Perception Track, Body Perception Track, and Perception Track

In parallel, the step activates beliefs (see Chapter 2.1.1) is active also in the environmental perception track, the body perception track, and perception track beneath the drive track in Figure 2.7. Similar to drives, sensor data are collected from the external perception as well as body sensors (5). The *external perception track* provides the system with unidentified collections of attributes, similar to the content, which would also be made available to the Perceptual Associative Memory (PAM) in LIDA. It is supposed that several modalities like vision, taste, and hearing [Deu11, p. 90], [Zei10, p. 65]. However, in ARSi11, only the visual modality is available for processing within the psyche [Muc13, p. 126]. The *body perception track* shall provide the system with sensor values for e.g. the position of the actuators or pain, which is inflicted [Deu11, p. 90]. From the view of implementation, currently, the sensors are available in functional layer 2 [Muc13, p. 124], but the data is not transferred to functional layer 3. The collected features are used as a cue to search for matching entities in the long-term memory (6 in Figure 2.4). Of all partially and completely matched entities, the best match is activated. Additional information associated with the entity is also activated, in particular, drive meshes [Deu11, p. 92]. Drives in SiMA in the *perception track* can be seen as equivalent to the usage of reactive emotions in LIDA [FR06, p. 44], [FF12, p. 107], but different to SiMA, reactive emotions in LIDA seems to lack grounding in a body. The activated entities contain attributes, positions and drive meshes. Such an entity is equivalent to the concept of percepts in Icarus. However, the percepts there are not connected to any evaluation structures. Entities form a *perceived image*. For instance, an entity is a thing presentation mesh, with the shape of a <CIRCLE> and the color <RED>. It is called

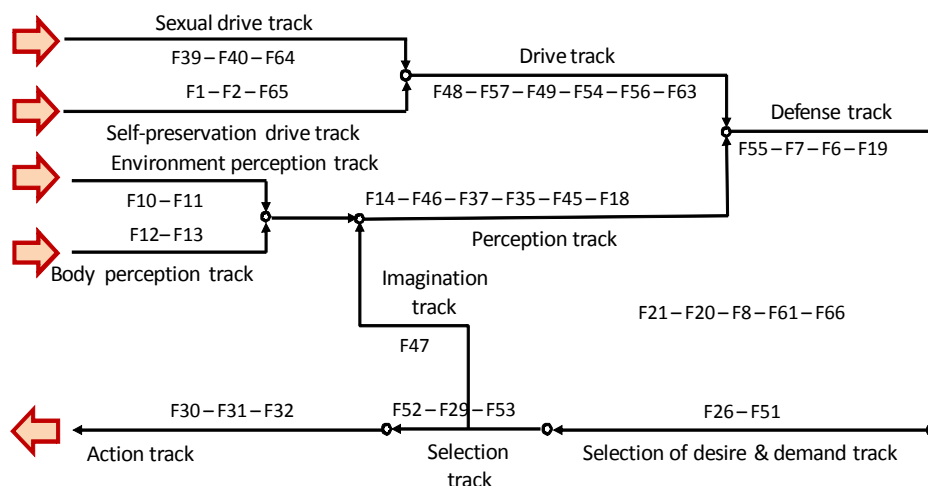


Figure 2.7: View of the tracks in SiMA, which contain functional modules labeled by Fnn of abstraction level 3, layer 3 [DSBD13, p. 6669]

<CAKE>, which radial distance is <FAR> and its position is <LEFT>. Further, a drive mesh is associated with the food source <CAKE>, which tells the system that <CAKE> will satisfy hunger.

The association of drive meshes with entities can be interpreted as the start of the phase infer beliefs as additional information about what the agent can do with entities is loaded. The perceived image is implemented as a list of independent entities and their positions and it represents the basic beliefs of SiMA.

Defense Track

In SiMA, the psychoanalytical concept of defense mechanism is introduced in the *defense track* in Figure 2.7 on the right. Defense mechanisms can be interpreted as guardians, which mediate in conflicts between the id and the super-ego. The content is manipulated in such a way that only appropriate content can reach the pre-conscious and conscious data processing. How the content is manipulated is done according to personality-specific patterns, the super-ego rules (8 in Figure 2.4). All content must pass the defense mechanisms, in order to reach the secondary process (step 4 and 7 in Figure 2.4). Inappropriate content somehow violates the instructions of the super-ego and the task of the defense mechanisms is to block or modify in a way that allows it to pass [Deu11, p. 95]. The functionality of the super-ego rules is similar to production rules. An example could be that drives propose that the agent shall eat a food source <CAKE> to satisfy its drive hunger <HUNGER>. However, a super-ego rule can state that if the <CAKE> belongs to another agent called <BODO>, the agent must not eat the <CAKE> and therefore, this option is rejected.

In another example, the following may happen: The drive <HUNGER> can be repressed by lowering its quota of affect or the food source <CAKE> is denied. As a consequence, the agent may try to satisfy a different more important drive instead. However, the repressed drive does not just disappear, the drive to satisfy hunger is put in a temporary buffer, where it has the possibility to emerge in another shape, which is allowed to pass. For instance, the drive hunger could be attached to another entity like another food source <CARROT> in the perceived image, which is eatable but does not satisfy the hunger that good. However, if no super-ego rules are against eating a carrot, the drive is allowed to pass. If the <CAKE> is denied, which means, that it is removed from the perceived image and not available for the decision-making [Deu11, pp. 94-96]. With the passing of the defense mechanisms, the primary process is ended, and the secondary process starts.

Based on the description above, there is one decisive difference between defense mechanisms and common filters. A filter only reduces information. Defense mechanisms may reduce information like in the case of denial, but it also modifies content or even add new content.

Although the defense mechanisms do not infer beliefs (see Chapter 2.1.2, Figure 2.1) in the same way as other cognitive architectures³, they can be mapped to the process step infer beliefs as they affect the current knowledge of the agent.

³ The primary process does not allow the usage of inferred beliefs in the manner, which is often used in cognitive architectures. No relational conclusions can be done in the primary process. It would require the use of associations with a direction and a certain predicate, which needs a logical structure.

Selection of Desire & Demand Track

Similar to LIDA [FF12, p. 105], but in strong contrast to ACT-R [TCM03, p. 4], an important task of the primary process is to push perceived data to the secondary process (9 in Figure 2.4). After the defense track, the data are processed by the first track of the secondary process, the *selection of desire & demand track*, which is located in the lower right part of Figure 2.7. In the pre-conscious data processing, entities are extended with word presentations, which shall allow logical processing of them (10 in Figure 2.4). Drives are extended with word presentations as well. Emotions are converted to feelings.

The secondary process offers possibilities to use filtering mechanisms. While BDI [GTC+10, p. 79] and LIDA [FF12, p. 105] have some filtering, SOAR, and ICARUS (see Appendix A) does not have any filtering at all. In ACT-R [TCM03, p. 4], which only have top-down processes, the environment cannot influence the decision-making at all and only the objects, which are selected by the decision-making will pass the filter. CHREST [RLSL09, p. 2] filters on many places as it concentrates on the implementation of visual attention.

As human normally does not perceive the whole world at once, an attention mechanism is the proposed. It is a functional module called *focus of attention*. It is supposed only to let entities pass, which is of importance to the fulfillment of the drive goal. It is necessary because the agent only has limited resources available, which should be concentrated on solving problems regarding its motivations. The concept of focus of attention has not been implemented yet in SiMA. A side effect of the defense mechanisms in SiMA is that it can be seen as a filter that protects the decision-making from unimportant and harmful things. In that way, defense mechanisms set constraints on decision-making do some pre-decision-making.

In ARSi11⁴, only one type of inferred beliefs is used, and that is the association of entities to memorized drive meshes. In most cognitive architectures, one of the most important inference of beliefs is to extract relative positions of perceived entities from the absolute positions, i.e. to activate concepts with values like if one object is <in front of> or <on top> of another object. This type of inference has not been considered in SiMA until now as it has been no topic of the primary process, which does not allow relative positions due to the undirected associations between data structures. It reduces the ability of the agent to generalize, as for each constellation of entities in an absolute reference system, a template for an atomic action has to be provided. For the development of the secondary process, the introduction of relative positions is a topic of future work, especially in the area of spatial localization in the secondary process.

Before decisions are made, functionality for a reality check is intended to be implemented, which analyzes the incoming data if it is reachable in the current situation [Deu11, pp. 97-98]. The content of these modules would correspond to the functionalities of the step evaluate options. Then, the decision-making works as follows: It receives a list of drives and the perceived image. The list of drives is sorted by descending quota of affect. Further, social rules are considered, which have the possibility to inhibit certain drive objects [Zei10, p. 77]. For instance, if there would be a drive that

⁴ The previous implementation of SiMA. The former name was ARS. Therefore, the implementation is called ARSi[Number]

proposes that the agent would eat another agent <BODO>, social rules can prevent it and instead changes the drive object to another eatable entity like <CAKE>. The drive with the highest quota of affect is selected to be fulfilled. To fulfill it, its drive object is searched for in the perceived image and selected as a goal to reach if found [Deu11, p. 99], [Lan10, p. 74], [Lan10, p. 107]. Similar to LIDA, the steps propose options, evaluate options and select option can be seen as being executed twice: the first execution in the selection of a certain drive together with its drive object and a second execution in the selection of an action. This action will take the agent closer to its satisfaction of the drive. In a further stage of implementation, concepts need to be developed, which can consider not only the current perception but also previous experiences in previous, similar situations, e.g. to find out what drive goals can be reached from here, although it does not exist in the perceived image.

Selection Track

In the second execution of the steps propose options, evaluate options and select option, an action shall be selected. The *selection track* follows the selection of desire & demand track in the bottom of Figure 2.7 (11 in Figure 2.4). The selected entity to pursue in the perception directly determines the action, which will be executed. At the moment in SiMA, the concept of *plan fragments* is implemented, which takes an equivalent role to skills used in Icarus [LCT11, p. 12] and BDI [GTC+10, pp. 81-82] (12 in Figure 2.4). A goal may theoretically activate several matching plan fragments, where each of them consists of several atomic actions [PZ11, p. 66]. In SiMA, in the current implementation, there only exist one plan fragment per drive object, i.e. there is only one default plan, which can be selected. Which atomic action is selected from a plan fragment, is decided by the absolute position of the entity. For instance, if the drive hunger provides the task to go to the <CAKE>, the plan fragment for the <CAKE> on a certain position is activated. If the position of the <CAKE> is <LEFT>, then the atomic action will be chosen, which allows the <CAKE> to come closer to <CENTER>. In this case, the atomic action would be <TURN_LEFT>.

Problem-solving and planning can be seen as a case-based forward-chaining process, where the origin of problem-solving is the current situation and where the agent has to get closer to the goal one step at the time. In ARSi11, a reality check for what is reachable has been proposed but is subject to specification and implementation. Several other architectures like SOAR [LCC+11, p. 12] and ACT-R [ABB+04, p. 1039] use formalized descriptive logic to reason about exact situations, to know what to do next. One of the problems with their method is that e.g. in the case of situation calculus as a reasoning base, many restrictive axioms⁵ have to be provided without contradictions, to enable reasoning about a situation. The problem is that restrictions have to be provided for all possible events that theoretically could occur, although the context may not be suitable. The rules in such a system have to be well tuned to the results and conditions of other rules, to make the system perform the necessary tasks. Further, if the concepts of situations are generalized, some of the axioms that are defined in a certain context are removed as they do not fit into another context. One way to overcome the problem of generalization is to use context-dependent axioms and functions [McC87, p. 1034].

⁵ These are the axioms of the domain of ontology and not the term axioms, which are used to define functionality in SiMA, as in SiMA there is only one context

Another way of solving problems, which have been previously discussed is the case-based approach, and it will be used in SiMA. Here, all reasoning shall be done by examples. Just as human learn about certain situations, SiMA in the current state is supposed to be fed by memories, as a 30-year-old person is simulated. The situations are stored as images or scenarios in the primary processes and are formed into acts in the secondary process. The challenge is to make those memories usable for the decision-making in a certain, perceived situation.

Imagination Track

In SiMA, an influence of the secondary process on the primary process is theoretically proposed in the *imagination track* above the Selection Track in Figure 2.7. The idea is that plans are sent back to the primary process, to enable the agent to fantasize about possible plans [Deu11, pp. 88-92]. It will have a similar functionality as SOAR SVS [Win09, p. 3].

Action Track

In the final phase execute action from option, an action is selected among the available action plans and the command is sent to the body, which executes it in the environment (13, 14 in Figure 2.4). The *action track* defines the output of the system in the lower left of Figure 2.7. However, the current implementation of the second execution is still rather simple. Only one plan fragment is proposed and therefore only that one can be selected. The loop of the decision cycle is closed, as the actions executed in the environment, have an impact on the body and as a consequence on the drive state. As soon as a drive is satisfied, other drives define the goals of the agent and with that, the external actions.

2.3.6 Concluding Remarks on SiMA

Normally, all functionality in the human mind has a purpose and contributes to it a whole. If the previously described cognitive architectures only aim is to cover some parts of the human mental apparatus and not the whole, then the question is if they will ever be able to achieve a system with human-like functionality. SiMA may not have the same logical reasoning performance as other cognitive architectures yet, but at least it claims to cover all pieces of the human mind, especially those pieces, which are left out by the most other architectures. Therefore, the path to model and implement human-like functionality is theoretically reachable. It is interesting to point out that gradually many cognitive architectures add concepts like emotions (LIDA) or episodic memory (SOAR) as enhancements of their concepts because they see the need for it. In SiMA, these concepts are all native functionality.

The architecture SiMA can be implemented in a use case, to find out whether the implementation of a certain cognitive architecture could fit into the proposed model of the human mind based on psychoanalysis. The use case shall reflect the functionality of the human mind. Based on the previous analysis, many behaviors would probably be easy to reproduce if only external states would be evaluated. For instance, in Use case 1 Obstacle Modification in Chapter 1.3, like moving towards some food source would be easy to mimic with hard-coded behavior. On the other side, the internal states, which can be inspected in SiMA, would be impossible to recreate because the other architectures lack functionality like drives based on homeostatic levels.

The second question at the beginning of Chapter 2 was to find out where SiMA fits in the picture of cognitive architectures. In the comparison of common components of the cognitive architectures, it was clear that almost all common components found in the compared cognitive architectures can also be found in SiMA, although expressed in different terms, functionality and origin. However, the role of that component within the architecture was the same. A good example is the functionality of attentional mechanism. LIDA [FF12, p. 105], CHREST [RLSL09, p. 2] and in SiMA uses an attentional mechanism. The task is to select the important objects from the perceptual input. In LIDA, this process is done by certain specialized, attention codelets, in CHREST by the content of the short-term memory and in SiMA in the module Focus of Attention (F23 in Figure 3.20) [Lan10, p. 73], where the drive state mainly influences the selection of objects to focus. Another example is the functionality of trying to imagine the consequence of an action before it is performed. In SOAR SVS [Win09, p. 4], such a module have been added to the core architecture and in SiMA, it is expressed in the module Generation of Imaginary Actions (F52 in Figure 0.1) [Lan10, p. 74], which have not been implemented yet. As a conclusion, although the architectures are developed independently from each other and have different goals, some convergent evolution towards necessary functionality seems to take place. It appears to be comparable to the independent evolution of the eye, which has taken place several times. Although the different eyes often work on different physical foundations like a lens vs. a mirror, they have the same functionality for an animal [Daw01, pp. 159-221].

Regarding the definitions of sub-systems of Chapter 2.1.2, the current state of SiMA does only possess a simple form of deliberative sub-system on abstraction level 4 and 5. Functions have been roughly defined but never been specified in detail. A reactive sub-system could be realized through automatic actions of the primary process, where actions would be executed without including the secondary process. This type of acting has not been realized yet. Through the usage of the secondary process, SiMA provides a deliberative component, which is the core of the previously described cognitive architectures. However, at the moment, the secondary process of SiMA is only able to react to the body needs in the form of drives and to propose one default, external action to use for each drive. The action is executed at the same step. All actions are external actions as no internal actions have been implemented yet. To create fully a functional deliberative sub-system of SiMA. To create fully a functional deliberative sub-system of SiMA, more action possibilities have to be proposed than only one per drive and an analysis of possible options have to be made, which possibly takes more than one cognitive cycle. For that, the introduction of internal actions is needed. Then, the analyzed possibilities have to be evaluated depending on the emotional state, the drives and the constraints of the current external situation. In the present state of modeling as well as implementation, SiMA has an inability to react to external stimuli due to the same reasons like in ACT-R [TCM03, p. 4]. Drives origin from the homeostasis and is the only source of motivation at the moment. However, as soon as emotions are implemented, this ability shall be provided. SiMA will also eventually contain both a reactive and a deliberative sub-system. While the multi-step decision-making is a part of the secondary process, the primary process can process data and put it to its outputs at once. It will allow the execution of the primary process and the secondary process in parallel.

Until now, SiMA has concentrated mainly on the unconscious processes, while other architectures focus on the logical problem-solving processes. In SiMA, this type of processing is a part of the secondary process. LIDA [FR06, p. 43], BDI [GTC+10, p. 79] and CHREST [RLSL09, p. 2] have

something, which could be mapped to some pre-conscious processing. It is the part of their architectures, where the incoming information is reduced. SOAR, Icarus, and ACT-R (see Appendix A) do not even have any pre-filtering at all, which means that everything would be interpreted as conscious as it is available to the decision-making. Compared to other cognitive architectures, the relative strength of SiMA lies in the unconscious processing of data. With unconscious processing, the following functionality is meant: high resolution of drives and emotions, the possibility to first process all information in a fast, direct way before it is provided to filter mechanisms and the unique concept of defense mechanisms. It has to be emphasized that a concept like the defense mechanisms cannot be found in any other architecture.

SiMA is at the moment a project of fundamental research and has no specified application. Future applications have been mentioned in Chapter 1.2. SiMA has been used within various experiments with the aim to simulate behaviors. Those experiments can be generated from a model the human mind on the base of psychoanalysis [BGSW13, p. 3], [Deu11, p. 133], [Zei10, p. 110], [Lan10, p. 132]. Many of the analyzed cognitive architectures of Chapter 2.2 do have a specific implementation in computer games for entertainment or educational purposes. It provides an argument for doing the same with SiMA as soon as a certain maturity of its implementation is reached. As mentioned in the introduction (see Chapter 1.2), in this area of application, SiMA would have a huge advantage over those architectures regarding entertainment value as it provides the irrational functionality, which other architectures cannot, because the required functionality is mainly located in the primary process of SiMA.

3. Models and Concepts

“If the world should blow itself up, the last audible voice would be that of an expert saying it can't be done.”

[Peter Ustinov]

In the following, concepts will be developed, which shall provide a sufficient answer to the research question R1 from Chapter 1.3: How can SiMA be extended to be able to consider past experiences when taking decisions? As proposed in the task setting in Chapter 1.4, there are four main tasks to solve. Each of the tasks defines an own sub-chapter. All concepts have been proved of compliance to the theory of psychoanalysis by psychoanalytical consults in the SiMA working group. The starting point of this work has been given in Chapter 2.3, where the SiMA model and architecture has been put as preliminary work. The aim is to find solutions for all raised questions and deficits of preliminary work, which are in the scope of the task setting.

3.1 Extension of Data Structures for Temporal Processing

As discussed in 2.3.4, SiMA does not have any implemented data structures, which can represent experiences as sequences. Also, no episodic memory does exist. Therefore, there is no possibility to store a situation. The data structures used are only able to describe the current perception, which consists of several independent external objects that have been perceived. Therefore, there is a need

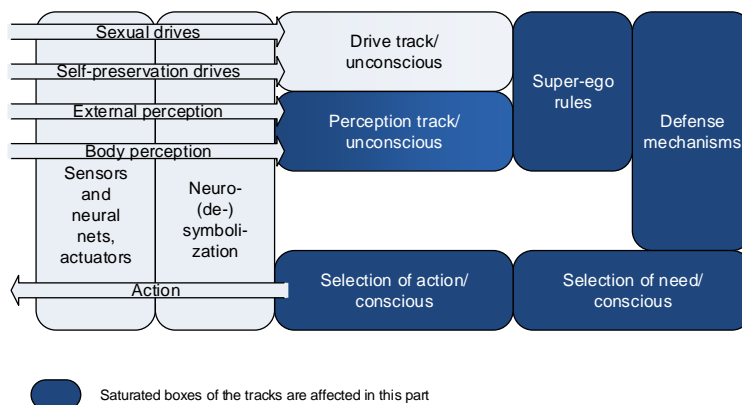


Figure 3.1: Affected tracks of the SiMA model for the extension of data structures

to introduce new data structures. The following concepts will describe a solution of task 2 from Chapter 1.4. The solution shall provide answers to the following research questions from Chapter 1.3:

Research question 2.1: How shall data structures for situations be designed?

Research question 2.2: How shall situations be organized into sequences?

Affected tracks of the extension of the SiMA model (see the track view of Figure 2.7) are highlighted in Figure 3.1. The individual, functional modules of the SiMA model are shown in the complete model in Figure 0.1 of Appendix B. Individual modules are not listed here, as data structures are not located in any particular functional module.

3.1.1 New Data Structures of the Primary Process

According to SiMA preliminary work in Chapter 2.3.4, there only existed four basic data structures in the primary process together with associations, which connected them. They were the thing presentation, the thing presentation mesh, the template image and the drive mesh. As will be discussed in the following, they are insufficient to model whole situations⁶, which are a prerequisite for acts. Therefore, the primary process has to be extended by the concept of an image.

Image

The template image as described in Chapter 2.3.4 was supposed to form the base for acts but was not used for that purpose as it only was used to represent single external objects. Therefore, no data structure that represents situations exists. The image will be defined in a way, which is more compliant to the original definitions of [Zei10, p. 58], [Deu11, p. 60], [DFZB09, p. 49], to represent situations.

The following technical statement about images is derived from statements of psychoanalysis interpreted by [Lan10, pp. 57, 62-64], [Zei10, pp. 49-50].

Statement 3-1: In the primary process, the association between two presentations is based on similarity or simultaneousness.

An image shall be able to associate all representations that occur simultaneously or look similar. In other words, images describe anything in the perception that can be grouped in time and space. It is a time independent data structure, which captures a moment in time, i.e. a cognitive cycle in a technical model.

According to [DZ08, p. 16] and [DBD+15, p. 42], images represent a group of perceived features. They origin from the functional layer 2 (see Chapter 2.3.5), where they have been grouped based on perceived shapes and edges on a neural level. Together with drive meshes and motions from Chapter 2.3.5, they are passed through the interface between functional layer 2 and 3 [DBD+15, p. 45].

The image distinguishes this group of features from other groups of features. Features are grouped and classified according to stored knowledge. For instance, the certain known pattern of color and shape

⁶ A situation consists of the complete perception

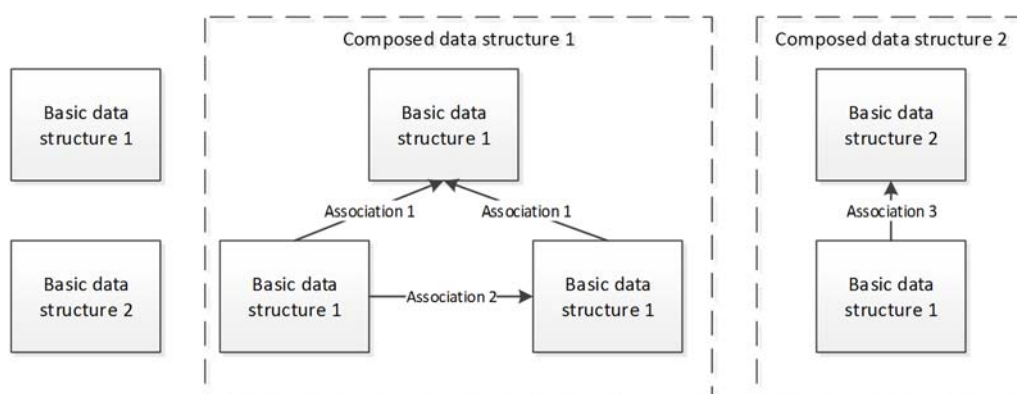


Figure 3.2: Basic and composed data structures

features represent a house. These groups can be arbitrary divided into new groups like a window or a door. From this grouping, it can be derived that there has to be a top image, which is the whole perception. It would be a pattern without subgroups. It defines at least one hierarchical level. The further division into perceivable subgroups is then an individual matter of the system.

It can be stated that perception can only make sense if it is processed and thing within it are distinguished from the whole. Therefore, there has to be at least another level of hierarchy with images that represent parts of the perception. Further, it can be derived that there cannot be a strict hierarchy of levels because every image is independent of other images and subgroups of the same level cannot be compared with each other, e.g. a tree compared to a house or a leaf compared to a window. If a thing presentation mesh is used to describe an external object like a <CAKE> or a <GREEN AGENT> in Chapter 2.3.4, it has to be used for the whole situations as well, because images can contain images. That invalidates the usage of the template image from Chapter 2.3.4.

Instead of being data its own like the template image of ARSi11 in Chapter 2.3.4, the image will be a composed data structure to allow a graph structure of images like originally intended in [Zei10, p. 58]. While a *basic data structure* is defined to be atomic, a *composed data structure* can be defined as a collection of other data structures that are connected by associations as illustrated in Figure 3.2. It has the advantage that it uses already existing data structures only define the composed data structure and no new basic data structure has to be defined. Therefore, it increases flexibility, expressiveness, and scalability in general of the data structures.

Based on the derivations above, a translation can be done into a technical concept, where the *image* can be defined as any thing presentation mesh, which keeps a group of any simultaneously occurring thing presentation meshes or thing presentations. They can form undirected graphs with associations as edges and thing presentations as nodes. This generalization of thing presentation meshes is the most significant modification of data structures compared to the preliminary work in Chapter 2.3.4.

To stay on the focus of this work, which is a working decision-making, a certain simplification of the handling of images will be made. In many problems, which cognitive architectures are confronted to, only two types of images are used: The situation and the object (see Chapter 2.2.1 and for details in Appendix A). Therefore, the top-level image that describes the whole perception will be called a

situation. Everything that is not a situation, but a part of it as described above, will be called *object representative*. It is defined as the internal representation of perceived *external objects*. It replaces the term entity, which has been used in preliminary work. It is the only necessary classification of images within this work because all further grouping of features would not be tested.

In a real application like in the SiMASin World in Chapter 4.1.1, the object representative is an obstacle <STONE>, an agent <GREEN AGENT> or a food source <CAKE>. Note that both situations and object representatives are instances of thing presentation meshes with certain roles. A perceived object representative in the system is an instance of the class thing presentation mesh. However, its role is the object representative. This representation is equivalent to the resource in RDF syntax [11].

In Figure 3.3, an example of a situation is shown. It shows a <GREEN AGENT>, which is eating (action <EAT>) a <CAKE>. The <GREEN AGENT> itself is defined as a collection of thing presentations with color and shape. The <GREEN AGENT> is not the subject, but another observed agent. Additionally, the current action <EAT> is associated with it. The agent also has the distance <FAR> and the position <CENTER> associated with it. Finally, just as in preliminary work, a memorized drive mesh (described in 2.3.4) is associated with the agent. In this case, it could be some sexual drive representation, which is satisfied by the observed agent. The <CAKE> has an equivalent structure as the <GREEN AGENT>. These two object representatives define the situation, just as thing presentations set the object representative. Additionally, a memorized emotion is associated with the situation, which could have the value anger <ANGER> in this case, because the observed agent <GREEN AGENT> is eating the food source <CAKE>. Memorized emotions will be explained together with the concept of emotions later in this chapter.

In the following, extensions of the concept image will be defined that are used in the representation of a situation. They are specialized for their purposes by putting restrictions⁷ on the general concept of

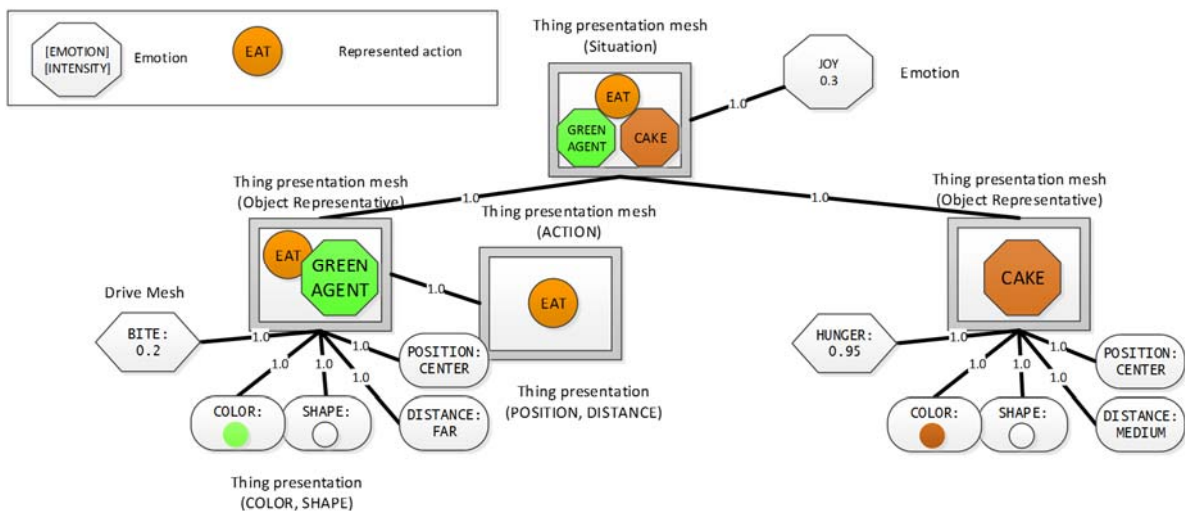


Figure 3.3: Example of a situation, which is an image of the primary process

⁷ A restriction is a limitation of the possible values of an object

an image. Each specialization of the image is named explicitly as they all play different roles in the SiMA model.

Perceived Image

According to [DFKU09, p. 60] and [BDD+14, p. 91], a *perceived image* is the situation and a snapshot of the perception of a certain moment, i.e. all perceived object representatives perceived within a cognitive cycle. However, the perceived image of preliminary work ARSi11 (see Chapter 2.3) was not a composed data structure, but a list of independent object representatives. A major drawback of such a concept is that the perceived image cannot be stored as a data structure, and no meta-data can be added to such an image. Further, if template images (in ARSi11) originates from perceived images, then they should be the same data structure. The definition above is sufficiently fulfilled if the perceived image extends the image. The perceived image is a specialization of the image because it is the unprocessed perception.

Template Image

The *template image* already existed in ARSi11 of Chapter 2.3 and was defined as something, which origins from the agent itself and keeps snapshots of recognized object representatives [DFKU09, p. 218]. It is the technical concept of the psychoanalytical term memory trace [LP73, p. 138], [DFKU09, p. 424], [Zei10, p. 49]. In ARSi11, the template image was only related to an object representative but never to a situation or perceived object representatives. There, perceived object representatives were thing presentation meshes and template images were another basic data structure.

Therefore, to solve these issues, the natural solution is to define a template image as an extension of an image, just as the perceived image. It will be used as the data structure for situations that have been memorized. Different to the perceived image, the template image allows generalizations⁸. One template image can represent multiple perceived images as it is more general than a perceived image. For instance, if the position of the <GREEN AGENT> in Figure 3.3 is removed and only distance is provided, the template image represents all perceived images, where the <GREEN AGENT> has the distance <FAR>. In this case the position, e.g. <LEFT>, <CENTER> or <RIGHT> does not play any role. A generalization explained in 3.2.2 and is visualized in Figure 4.3.

Motion

Derived from previous work in Chapter 2.3.4, the properties of an object representative can be static and dynamic. Static properties of object representatives are e.g. positions or colors. Dynamic properties like movement directions and current actions that are used in motions. As an example, in Figure 3.3, a dynamic property would be the associated action to eat. It is something that happens in the image. For instance, if there would be a camera in a technical system that records individual pictures, a component could extract movement symbols by analyzing the stream, i.e. compare two

⁸ A generalization is the allowance of more values for an object.

pictures and let the time between them go towards zero. These movement symbols could be added to a motion.

Based on the stated definitions in Chapter 2.3 of what a motion is, it can be defined as a specialization of the general concept image that includes at least one action of an object representative together with its movement direction. The argumentation that a motion is an image is supported by the following statement of psychoanalysis, which is based on the interpretations of [Lan10, pp. 57, 62-64] and [Zei10, pp. 49-50].

Statement 3-2: In the primary process, there are no directed associations between presentations.

If a direction of the association could be set, such a direction would define which image happens before another image or which image is a sub-image⁹ and which image is a super-image¹⁰. It is not possible in the primary process. If no temporal associations are allowed in the primary process, then the motion also has to be time independent, i.e. exactly an image.

In this work, the concept of motion has only been defined, but will not be implemented as it is not necessary for showing experience-based decision-making. In the proposed use case in Chapter 1.3, no moving external objects are present.

Emotion and the Relations to Other Valuations

According to [DBD+15, pp. 69-71], in SiMA, a comprehensive valuation system exists, which represent the economic aspect of psychoanalysis. In SiMA, five valuations have been defined: The quota of affect, the basic emotion, the extended emotion, the neutralized psychic intensity and the feeling. They are derived from bodily organ tensions and collected under the term *psychic intensity*.

The first valuation, the quota of affect, has been mentioned in Chapter 2.3.4 as the drive demand and it is generated in the second layer, the neuro-symbolic layer. They value psychic content according to the pleasure principle, i.e. how important the drives are that origin from bodily tensions.

Emotions were not invented in this work, but they are the main way of evaluating an image, and they cannot exist without an image. In [SDW+13, p. 6649], the concept of emotions in SiMA has been described and derived from psychoanalysis. An *emotion* is an evaluation mechanism that is applied in certain functions in the primary process. Emotions represent a compact evaluation of the perceived image as well as the drive state through the quota of affects. Unique for SiMA compared to other architectures in Chapter 2.2 is that emotions include both the drive state and the perception.

Emotions are split into basic emotions and extended emotions. *Basic emotions* can be described as a vector, where there are six basic emotions expressed as scalars: anger, mourning, anxiety, joy, saturation, and elation [SDW+13, p. 6649]. Which of them are generated in a cognitive cycle and with which intensity, is dependent on both the drive state (see Chapter 2.3.1) as well as the state of the activated memories (template images). Briefly, drives contribute to the generation of emotions by

⁹ Sub-image: an image that is hierarchically subordinated another image

¹⁰ Super-image: an image that is hierarchically super-ordinated another image

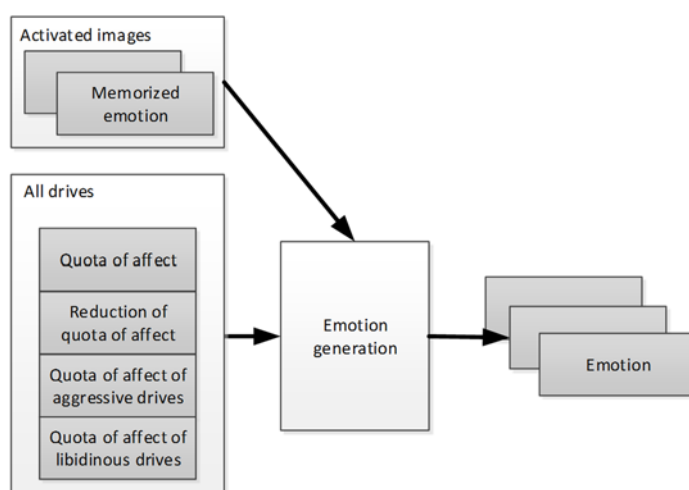


Figure 3.4: Generation of basic emotions based on the figure in [SDW+13, p. 6649]

providing an aggressive and a libidinous component (see Chapter 2.3.5). The quota of affect (see Chapter 2.3.2) of all drives is incorporated to influence the intensity of the emotions. The generation of basic emotions is shown in Figure 3.4. While basic emotions are generated before the defense mechanisms, *extended emotions* are the result of the application of defense mechanisms on the basic emotions. The following extended emotions are available: envy, greed, pride, pity, guilt, depressive mourning, shame, disgust, hate with object or love with object [DBD+15, p. 71].

During this work, the extended emotions have not been fully specified as well as the way the defense mechanisms are applied on them. Therefore, a simplifying assumption has been made that the extended emotions are equal to the basic emotions. Therefore, the term emotion is used as the basic emotion in this case. Further, for this work, the relevant properties of emotions are reduced to the emotion type and its intensity.

Symmetric to the drives, emotions are used in two roles [SDW+13, p. 6649]. They are used either as an evaluation of the current internal state or an evaluation of a memorized state, i.e. template image. Emotions that have been stored together with a template image are called *memorized emotions* [SDW+13, p. 6649]. They are the link between experiences of former emotional states and current emotions. The more memorized emotions of a particular type are activated, the higher is their impact on the current active emotions. Those emotions will be activated with an increased intensity [SDW+13, p. 6649].

As a conclusion from [SDW+13, p. 6649], the difference between a memorized drive mesh and a memorized emotion is that the memorized drive mesh tells the agent about how well an object representative can satisfy a certain drive. The memorized emotion tells the agent about its emotional state was in that particular situation. An example of a situation with an associated memorized emotion is the example of the emotion <FEAR>. If an image is associated with the memorized emotion <FEAR>, it tells the agent that the situation is dangerous and finally may harm the agent. It has to be recalled that the purpose of the SiMA agent is to survive, and it is only possible if the body survives. The best reaction to such a situation may be to flee from it. No memorized drive mesh could be used

to invoke the flee reaction as drives only represent homeostatic values and does not consider external situations.

Later in the cognitive cycle, in the decision-making of the secondary process, *feelings* are based on emotions. They influence which possible goal (see Chapter 3.4.2) shall be fulfilled. Emotions and eventually also feelings will allow the agent to react to external stimuli. Emotions apply on unconscious data in the primary process and feelings on conscious data in the secondary process. Drives only apply to the homeostatic state of the body. An example is that they provide the agent with the capability to flee at the sight of a hostile agent as that agent may do harm to the body. Thus, the infrastructure for the usage of feelings will be created in this work, the exact influence of the decision-making is out of the scope. Therefore, decision-making will only be based on drives and drive wishes in the secondary process.

Finally, neutralized psychic intensity is derived from quota of affect that has been reduced in the drive track [DBD+15, p. 70]. It is mainly used to control functions of the secondary process. For instance, it influences how much can be filtered in the focus of attention in the secondary process or how many possible goals will pass the decision-making (see Chapter 3.4.3).

Associations Between Images

In Chapter 2.3.2, the properties of the primary process were described. The following statements can well describe its properties regarding data structures. They are derived from the description of primary process data structures and are based on the description in [Zei10, pp. 53-58].

Statement 3-3: Associations have a dynamic weight, which represents their strength.

Statement 3-4: There can only exist one association between two object representatives.

These two statements tell that the association weights of the primary process are addable. Associations form edges and thing presentation meshes form nodes in an undirected graph. Because images are newly introduced, also this type of associations between the images is novel for SiMA. However, first in Chapter 3.2.1, their purpose will be described, where it comes to the activation of images through their associations in the primary process.

3.1.2 New Data Structures of the Secondary Process

In Chapter 2.3.4, it was shown that the implementation of the secondary process only contained one data structure, the word presentation. Unfortunately, the word presentation has the structure of a literal (value) and is not able to contain associations to other data structures. Therefore, as its functionality is equivalent to the functionality of the thing presentation of the primary process, it would be proper to define a new secondary process structure, which is equal to the thing presentation mesh in the secondary process. The introduced data structure is called the word presentation mesh.

Word Presentation Mesh

In ARSi11 that was described in Chapter 2.3, no data structure of the secondary process did exist that could be used together with associations. However, it was supposed that a data structure would exist that is attached to primary process data structures and allows temporal and hierarchical associations as well as the naming of thing presentation meshes by words. Different to the primary process, associations content of the secondary process can be described with the following statements, which are based on the interpretation of psychoanalysis from [Lan10, pp. 58, 62-64] and [Zei10, pp. 49-50].

Statement 3-5: In the secondary process, associations have directions.

It works like the grammar of a language. An association between presentations acts like a predicate, which connects a subject with an object. It tells what the subject is and what the object is and introduces order through asymmetry in the system.

Statement 3-6: In the secondary process, associations have predicates, to describe relations between concepts.

The associations of the secondary process define the edges of a directed graph. For that purpose, the *word presentation mesh* is defined as the node of the directed graph in the secondary process. From the statements above, it can be derived that the associations in the secondary process allow a representation of something that happens before and something that happens after. The predicate allows the definition of relations between things, e.g. a <CAKE> <isa> <EATABLETHING> or an arrangement of a perceived object representatives to a context with <hasContext> to the context <HOME_OF_THE_GREEN_AGENT>. What can be seen here is predicate logic. Regarding semantic web and RDF syntax [11], the word presentation mesh represents the resource, while the word presentation represents the literal or value (like the member variable of type String of some class).

In accordance to [BDD+14, p. 93], the word presentation mesh is an equivalent representation of the thing presentation mesh. The *word presentation* is then equal to the functionality of the thing

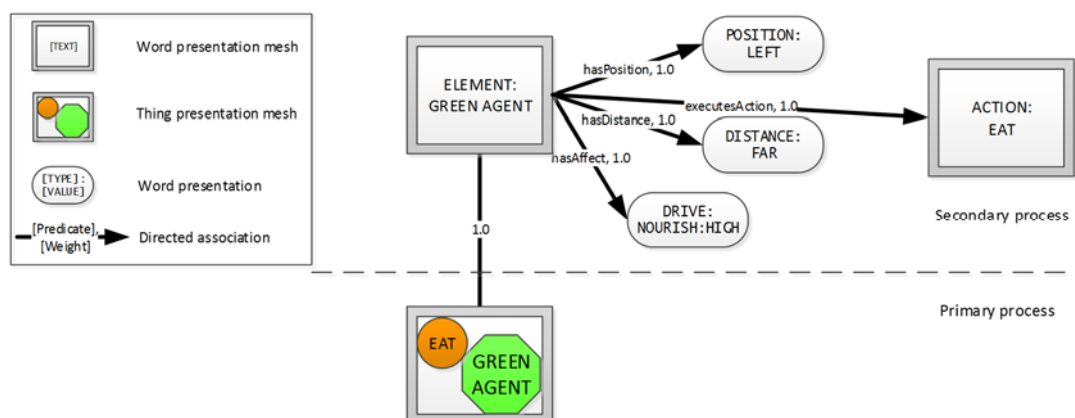


Figure 3.5: Example of the word presentation mesh in the secondary process

presentation. Conform to the SiMA model theory in Chapter 2.3.4, the word presentation mesh is attached to a thing presentation mesh, to allow secondary process functionality to access them.

In Figure 3.5, it is illustrated how a word presentation mesh looks like. It is connected to the thing presentation mesh of an object representative named <GREEN AGENT>. Its name <GREEN AGENT> may be communicated as a label to another agent. The <GREEN AGENT> object representative is associated with another word presentation mesh for the action <EAT>.

Because the purpose of word presentation meshes and word presentations is to enhance primary process data structures, there is no need to introduce any other data structures in the secondary process. Except the thing presentation mesh, also emotions and drive meshes can be represented with their corresponding word presentation meshes in the secondary process.

No other architecture (see Chapter 2.2.1 about internal representation and details in Appendix A) uses different representations of data structures in the way it is done in SiMA. Different to other architectures, where it is stated that some processes may be pre-conscious and some processes may be conscious, in SiMA the separation between unconscious and pre-conscious/conscious processes is defined by the data structures. Functions of the processes can only process their types of data structures. The consequence is that the simple rules of association in the primary process enable fast and straightforward processing. In contrast, in the secondary process, logical or temporal data processing is possible. The price is a higher demand for resources.

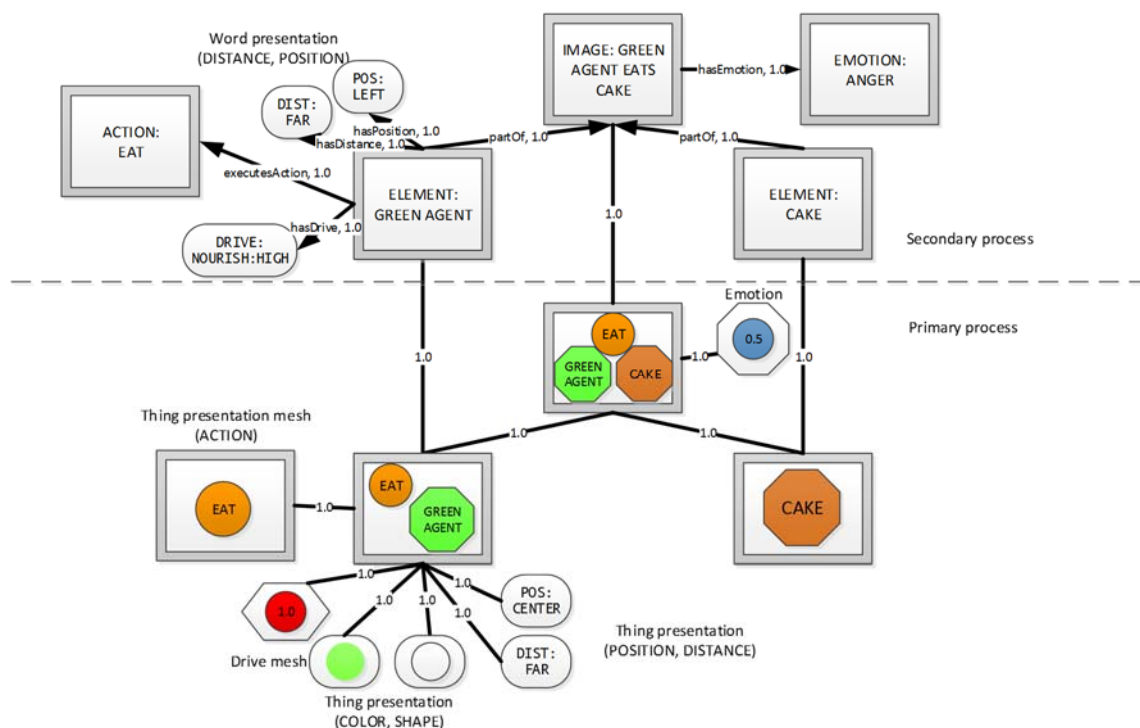


Figure 3.6: Example of a labeled image of a situation

Labeled Image

In the secondary process, the image from still exists, however as mentioned before, it cannot be processed without word presentation meshes. Therefore, the *label* [BDD+14, p. 95] is added. The label is a word presentation mesh. In Figure 3.6, the image from Figure 3.3 is extended with word presentation meshes and word presentations. The whole data structure can then be called a *labeled image* according to the SiMA theory [BDD+14, p. 95].

Act

In Chapter 2.3.4, the act was defined as an exclusive secondary process data structure, which consists of a sequence of images, where the image is a situation. A more precise specification would be labeled template image¹¹. Only in the secondary process, it is possible to use sequences of labeled images. It is possible, due to temporal associations. In ARSi11, the concept of acts was never implemented. For this work, the act is the essential concept for representing experiences. Acts explain to the subject that there is something that happens before and something that happens after an action or an event. The possible usage of an act can be seen as related to skills in ICARUS [LCT11, p. 12] and schemes in LIDA [FR06, p. 44]. Therefore, acts can be used as action plans in SiMA as well. Here, one labeled template image can be interpreted as a precondition and a second as a postcondition of a certain action. In SiMA, the action would be stored in the labeled template image of the precondition. In [Zei10, p. 99], consequences of actions were realized with one data structure called *micro-act*.

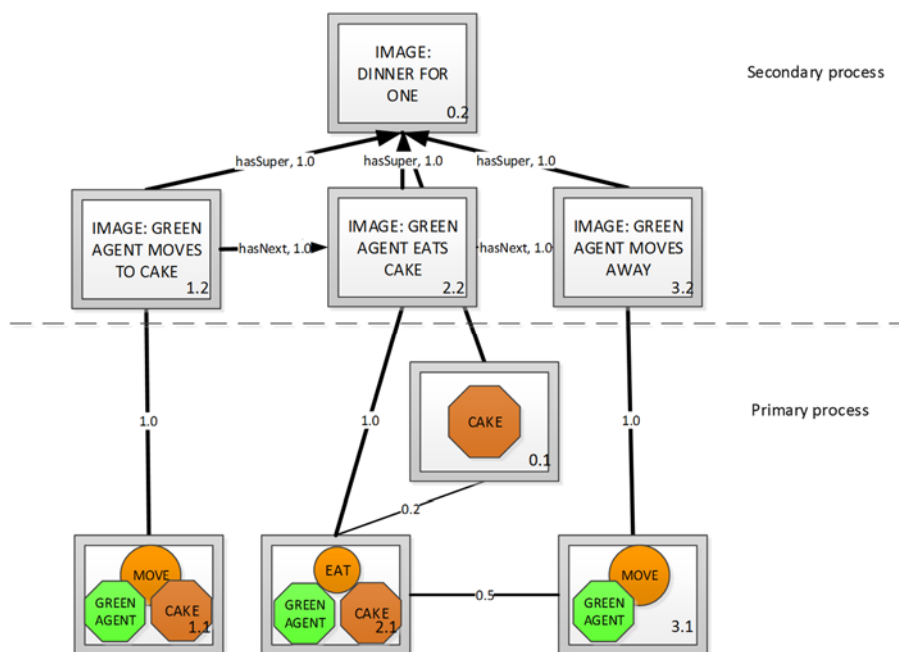


Figure 3.7: Example of an act

¹¹ Labeled template image: A stored situation that is accessible for secondary process functions

However, in Chapter 3.1.1, composed data structures were defined. It is close at hand to use composed data structures instead to model sequences. Therefore, the act will be modeled in this way.

A redefinition of the concept of [Zei10, p. 60] will be introduced for this work, to get a consistent specification of the concept of a technical system. It is derived from the definition of an act in Chapter 2.3.4. It is possible because predicates were defined for associations in the secondary process. An *act* is a representation of at least two labeled template images, which are connected to each other by a temporal association in the secondary process and which are connected to a third labeled template image (the super-image) with hierarchical associations.

The act can be explained with a metaphor. If a movie represents the act, then the labeled super-image is a short preview of the movie, and the single labeled sub-images are sequentially ordered scenes of the movie. In Figure 3.7, an example shown. It describes an experience of how a green agent moves to the cake, eats it and moves away. In the following, temporal association predicate will have the value `<hasNext>`¹² and the hierarchical association predicate will have the value `<hasSuper>`. The labeled template super-image is named `<A DINNER FOR ONE>` (0.1, 0.2 in Figure 3.7). In the first labeled template image (1.1, 1.2 in Figure 3.7), it is observed how the `<GREEN AGENT>` moves to the `<CAKE>`. In the second labeled template image (2.1, 2.2 in Figure 3.7) the agent `<GREEN AGENT>` executes the action `<EAT>` on object representative `<CAKE>`. Finally, in the third labeled image (3.1, 3.2 in Figure 3.7) the agent `<GREEN AGENT>` executes action `<MOVE>`.

All data structures that are necessary for describing experiences have now been defined. In the next chapters, a process for extracting valuable information from those data structures will be developed.

3.2 Activation of Experiences in the Primary Process

The following concept shall be drawn up to solve task 3 from Chapter 1.4, where memory retrieval will be introduced based on a perceived- or template image. The solution shall answer the research question from Chapter 1.3:

Research question 3: How shall those stored situations be retrieved from the memory, which is relevant to the current situation within a certain context?

Affected tracks of the SiMA model are shown in Figure 3.8. For this functionality, only the perception track is affected and especially the functional module Memory traces for perception (F46 in Figure 0.1). The memory retrieval of experiences happens in the primary process. The location of the memory retrieval has a major consequence. All content has to pass the defense mechanisms, which can manipulate or remove parts of the content. Interfaces between the modules are labeled with [In.m]. The complete SiMA model is located in Figure 0.1 in Appendix B.

The module Fusion with memory traces (F46 in Figure 0.1) [BDD+14, pp. 119-120], [Dob15, p. 10] has the task to form a perceived image from incoming perceived object representatives and to activate

¹² The explanation of the concept is simplified by using exemplary values. `<hasNext>` could be replaced with anything else, as long as the system knows how to handle it. The same is valid for the hierarchical association `<hasSuper>`

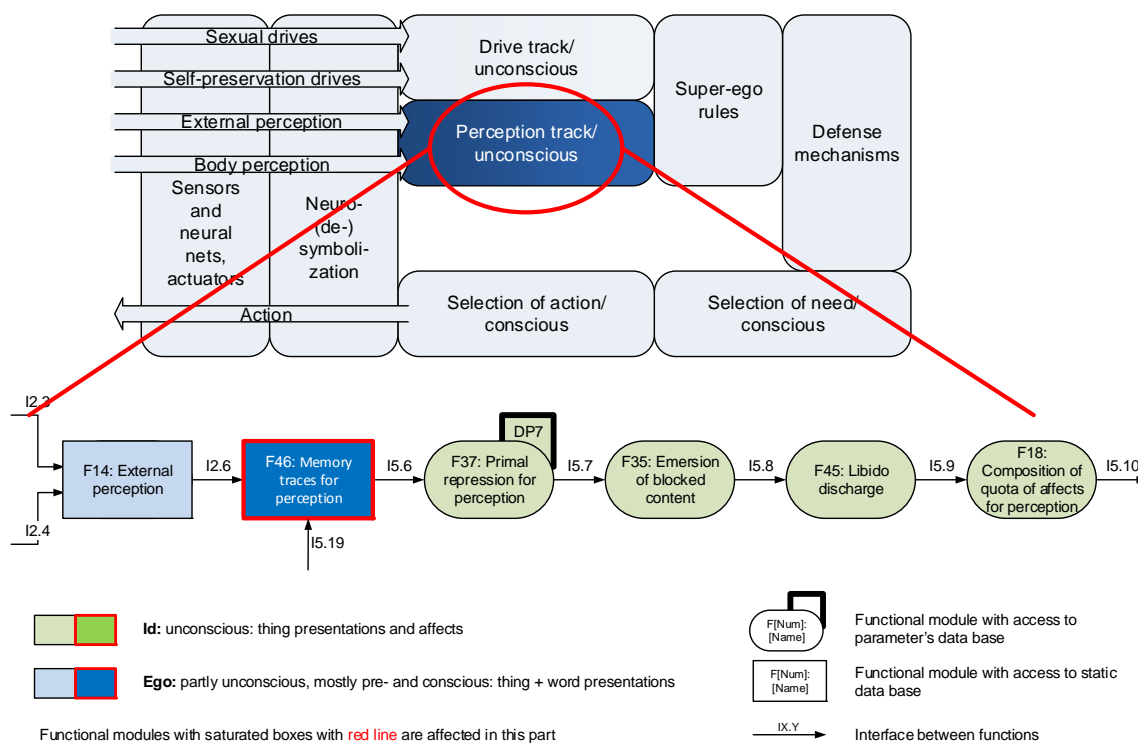


Figure 3.8: Affected tracks of the SiMA model for the implementation of memory retrieval

template images, which represent fragments of experiences. The perceived image is then merged with the activated template images. From the perspective of this work, the main functionality of the primary process is to provide the decision-making with suitable template images and to provide motivations for taking a decision. The images activated here, later define the options (see Chapter 2.1.1), which are available for selection in the decision-making. In a broader sense, experiences, which are represented by images in the primary process of SiMA, can be interpreted as inferred beliefs (see Chapter 2.1.1), because they define a view of the world or beliefs of the agent in a given situation.

3.2.1 Memory Retrieval in the Primary Process

According to the psychoanalytical theory, memories are activated by associations in the primary process [MWD+11, p. 3], [Zei10, p. 49]. [Lan10, pp. 56-58] expresses it like this: “*memory traces are also associated with other memory traces that generally appear in conjunction. When a particular set of sensor data usually activates two memory traces and for some reason only one of those memory traces is activated, the tight association between the two induces the activation of the associated memory trace*”. Memory traces have been explained in Chapter 2.3.4.

The functionality can be demonstrated in a real-world example. A two-year-old child is asked a question about her favorite movie Pippi Longstocking [5] from the world famous author Astrid Lindgren [HW12, pp. 200-201]. It will show an insight of the primary process information processing based on similarity of concepts [BDD+14, p. 71]. The child is asked, “*Could you name all characters you know from the movie Pippi Longstocking?*” The returned answer is: “*Pippi Longstocking, Tommy,*

Annika, Herr Nilsson and Aladdin". While Pippi, Tommy, Annika and the monkey Herr Nilsson are characters in that movie, Aladdin does not seem to have any logical connection to it. However, in Figure 3.9, the chain of associations in this particular case is shown. The example fits very well in the psychoanalytic theory. Due to the associations based on similarity, memories of the character of Pippi Longstocking, Herr Nilsson (the monkey to the left) also activates the character Abu (the monkey to the right) from the Walt Disney movie Aladdin [6]. The monkey then further activates the main character Aladdin.

In psychology and in technical solutions, there is an existing, similar concept called spreading activation [WSG+13, p. 6671], [And83, pp. 261-295]. It is frequently represented as a semantic network, although its principle is valid for all kinds of networks. Weighted and non-weighted associations can be used in the graphs. The process involves three steps, which are recursively executed for each node of the graph: pre-adjustment, spreading and post adjustment [Cre97, pp. 453-482]. If a termination condition is reached, the activation stops. A pulse is ended in case a termination condition is reached. Pre- and post-adjustment steps are optional. Usually, some form of activation reduction is implemented, to regulate the activation spreading. For the spreading, a calculation of the incoming activation from associated nodes is performed [Cre97, pp. 453-482] including the consideration of the association weights. After an input value has been calculated, the activation value of the output is calculated. Here, different activation functions may be used. The activation value is then spread to all associated nodes. Finally, it results in the assignment of activation values for each node in the network. In that way, the significance of a node for the retrieval is represented by the activation value [WSG+13, p. 6671].

Some cognitive architectures use spreading activation for information retrieval. Because of the nature of spreading activation and its close relation to psychological research results, it is used to study various effects like priming-based processes such as regency and frequency effects. Here, the impact of recent or frequent input on cognitive processes is investigated [Cre97, pp. 453-487]. A look at the state of the art of Chapter 2.1 shows that of the analyzed architectures, LIDA uses a spreading activation to activate relevant content from slipnet. In SOAR, an important task is to prioritize, store and schedule goals, which cannot be reached immediately. Although no spreading activation is used, the authors of [LL11, p. 8] recommend the exploration of it due to its capability to search selectively,

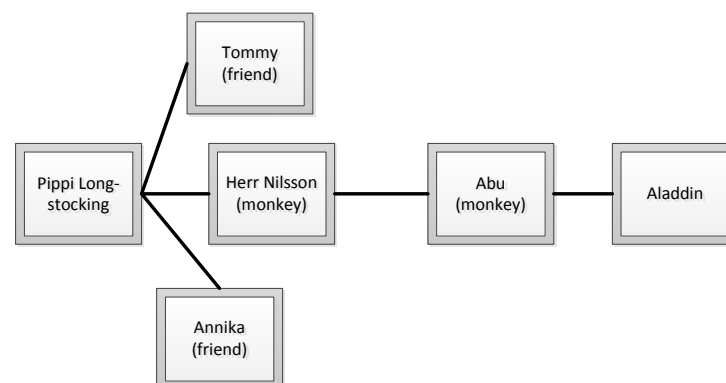


Figure 3.9: An example of activation of images in the primary process

to save resources and its ability to activate suiting content for external events [LL11, p. 8]. Instead, at the selection and activation of episodic memories, SOAR uses selection bias, which excludes working memory elements from the search, i.e. a working memory element can be selected, but not to be found in an episode [NL07, p. 1563]. However, this feature does not seem to be compatible with the concept of spreading activation. In spreading activation, the same effect is reached if there are no associations defined between two nodes.

Due to the similarities between the psychoanalytical theory about memory activation [WSG+13, p. 6671] and the existing concept of spreading activation [And83, pp. 261-295], [Cre97, pp. 453-482], a technical implementation of spreading activation would be suitable for memory retrieval in SiMA. However, it has to be adapted to the requirements of the SiMA model.

3.2.2 Image Matching

The first step in creating a memory retrieval mechanism is to build a comparison mechanism for images. Here, the similarity of two images has to be measured, to decide if an image shall be activated or not. It is assumed that the primary process image matching is comparable to how other mammals compare images.

In the experiment Morris Water Maze [SK09, p. 101], it is demonstrated how spatial structures are handled in the implicit and the explicit memory of rats [ST02, p. 81], [SK09, p. 68]. The data structures of the explicit memory roughly correspond to the data structures of the secondary process in SiMA. In the primary process, only the implicit memory is accessed. In the experimental setup, a rat is put in a pond with milky water and a non-visible plateau somewhere in the pond. On one side of the pond, there exists a visual landmark. The rat has to swim around to find the plateau, where it can stand. If the island is found, the reward is that the rat does not have to swim anymore. A normal rat finds the plateau and notices the relative position of the visual landmark to the plateau. From now on, it finds the way to the plateau from any position, due to the relational spatial positions of the perceived objects. A rat, which explicit (declarative) memory has been disabled, acts only on rewarded habit and always takes the same learned path pattern, no matter of the starting position [SK09, p. 189].

For SiMA, the conclusion of the experiment is that a sort of “photographic” pattern matching takes place. Relative positions of object representatives are not considered. Statements in Chapter 3.1.1 also support this finding. The derived statement says about the primary process that only similarity of a pattern can be compared. It is because relational associations of spatial data are not possible with undirected associations. As a technical translation of this concept into SiMA, object representatives and their absolute positions of a perceived image can only be compared to other object representatives and their absolute positions in the template images. According to knowledge about the primary process in Chapter 3.1.1 and 2.3.4, this concept can be generalized to all kind of image matching in the primary process.

Another conclusion from the Morris Water Experiment is that relational image matching by relational positions would allow the recognition of another perspective as the original one. It corresponds to the functionality of the secondary process, where these types of associations are allowed (see the description of secondary process data structures in Chapter 3.1.2).

Compare to image matching of the primary process, there absolute spatial positions of the object representatives are used, which makes the calculation faster than the matching with relative positions. It is because the object representative of the template has to be searched for in the target pattern and not the relations between the object representatives in the template pattern and the target pattern.

Finally, in order normalize the comparison, the pattern matching should result in a match value between 0.0 and 1.0. Based on the required functionality, the input to the image matching can be defined by two images: the *source image*, which shall be used as a query or search pattern and one template image of the memory, to which the source image is compared. The source image can be any image like a perceived image. How the image matching is done in detail is dependent on the representation used and is, therefore, a part of the implementation of the system, as it could be technology dependent.

3.2.3 Memory Retrieval Interface

In the SiMA model, the memory retrieval functionality is located in the module Memory traces for perception (F46). In Figure 3.8, the module is marked with red color. According to the methodology described in Chapter 1.5, the system is being partitioned and to develop the functionality of this partition. First the inputs and the outputs of the module are described.

Input Perceived Image

In module External perception (F14) of Figure 3.8, symbols of groups of features, i.e. images from the perceptual (interface I2.3) as well as the body sensors (I2.4) are assigned to object representatives. It was a part of the work of [Muc13, pp. 90-100]. The independent, perceived object representatives are then passed to the module Fusion with memory traces (F46) (I2.6) in Figure 3.8, where they are first merged into a perceived image as defined in Chapter 3.1.1. The perceived image is a situation that consists of all perceived object representatives and their positions during one cycle of the model (see Chapter 3.1.1). Note that according to Chapter 3.1.1, motions are not used in this model and therefore not considered as an input.

Input Template Images from the Secondary Process

In the SiMA model, there is a top-down influence on the primary process from the secondary process [Dob15, p. 10], [DBD+15, p. 86]. It is interpreted as fantasy as parts of plans from the secondary process are sent back to the primary process, to be checked for violations of the super-ego rules. However, as experiences are the base of decision-making and planning, it makes sense that the plans are not directly passed to the defense mechanisms because plans are not available in the long-term memory. Instead, the experiences in the form of template images that a plan can activate should be checked. The purpose is to activate later acts, i.e. experiences, based on those template images that define an executable action plan. Because this chapter only describes the concept of memory retrieval, the argumentation why template images are used in this way and how they are used is comprehensively described in Chapter 3.4.5.

The module Fusion with memory traces (F46) receives one or more template images from the secondary process from the previous cognitive cycle (I5.19 in Figure 3.8). The template images are

the primary process part of labeled template images from the module Conversion to primary process (F47) as described in Chapter 3.4.5.

Input Psychic Intensity

In the SiMA model, the term *psychic intensity* is used as a valuation system, where it represents the economic aspect of psychoanalysis [DBD+15, pp. 68-70]. Resources are limited and therefore optimized in the human mind. According to [DBD+15, pp. 69-70] “*In the SiMA project, psychic intensity is used as an umbrella term for all valuation quotas... Valuation ultimately serves to prioritize actions in order to mediate between the demands of the outside world (the own body and the environment) and the inner (psychic) needs (e.g. satisfying psychic and physiological needs within the environment, or adapting (psychic) wishes to the external circumstances)*”.

Due to the limiting effect of psychic intensity, it has the following effect on memory retrieval. Due to limited resources, like e.g. CPU power in a computer, there has to be some limitation of how many stored template images shall be activated. Even if many template images would fit into the current situation, all of them cannot be activated. It has to be selected which images are most relevant.

In a technical translation of the concept, the functional module Fusion with memory traces (F46) is assigned a certain amount of psychic intensity in each cognitive cycle by the drive track. The amount corresponds to the sum of all quota of affect of all drives [DBD+15, p. 69]. Hence, the psychic intensity is the translation of organic tension into a proper psychic representation.

Input Drive State

While the psychic intensity can be seen as a resource prioritizing factor, there is a need for something to decide what is important at this moment. According to the SiMA theory [DBD+15, p. 75], “*...it is possible for the psychic representative of a strong drive to influence perception (e.g. when one is hungry, one sees food everywhere and tends to buy more)*...”. As a conclusion, if the system has to prioritize, which template images shall be activated, then it should select template images that can help the system to fulfill its purpose, i.e. its drives. Therefore, the last input to the system will be the current drive state. In the SiMA model, is passed from the module External perception (F14) (I2.6 in Figure 3.8).

Output Perceived Image and Template Images

From the module Fusion with memory traces (F46), the perceived image that was generated there leaves the module (interface I5.6 in Figure 3.8). After activation of template images, the perceived image will be associated with them, as they are either directly related to it or indirectly through activated associations.

Knowledge Base for Template Images

In Chapter 3.1.1, the structure of images, in general, can be described as an undirected graph. In the *long-term memory* in SiMA, therefore, template images are saved as nodes. It is shown in Figure 3.10. Template images contain factors, which influence their activation, i.e. memorized drives (see Chapter 2.3.4) and memorized emotions (see Chapter 3.1.1) [WSG+13, p. 6671]. Between some of them, there

are pre-defined associations. In a learning system, it is assumed that these associations would have been learned if template images were similarly activated. However, in this work, there is no learning, and all associations of the long-term memory are predefined.

3.2.4 Psychic Spreading Activation

The following task is to develop an algorithm that will translate the concept of memory retrieval of the primary process as described in Chapter 3.2.1 into a technical, realizable concept for the cognitive architecture SiMA. Parts of the content of the following subchapters have already been published in [WSG+13, pp. 6670-6675] and [MWB+11, pp. 1-6]. Here, the concept of the papers is extended and completely described.

Initial State

The starting position is the inputs on the interface of the module Fusion with memory traces (F46). In Figure 3.10, the initial state of the system is visualized. It consists of the inputs source image, available psychic intensity and the drive state. The source image can be the perceived image or any other template image or even a merge of both.

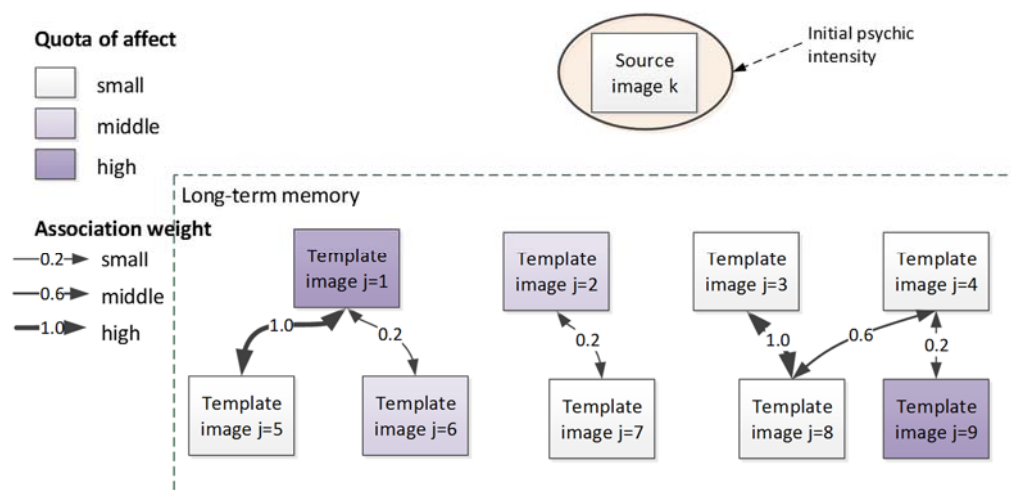


Figure 3.10: Starting point of memory retrieval with exemplary values

Process Step 1 – Image Matching

According to the SiMA theory, memories are stored as a pattern, which can be activated by perception. Therefore, the source image is compared to other template images in the long-term memory, to analyze the similarity. At the comparison, associations are created between the source image and the template images if the quota of match exceeds a defined threshold. Else, the similarity is negligible and must not be considered. The similarity values then range from threshold value to 1.0.

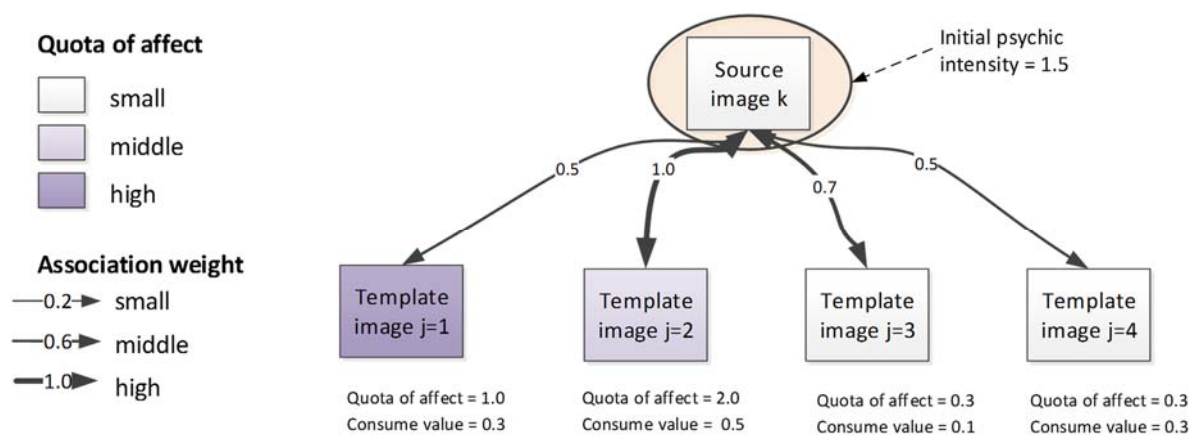


Figure 3.11: Example of calculation of the spreading from a source image to template images

Because the associations between the source image and a template image have been created by actively comparing the situations, they will be called *direct associations* for further processing. In Figure 3.11, this process step is shown. All existing associations between template images themselves will be called *indirect associations*. The terms, directly and indirectly, are related to the concepts of [Zei10, pp. 84], but here they are extended to consider images.

Process Step 2 – Calculation of Psychic Potentials

The source image defines the starting position of the spreading. The psychic intensity is the resource, which is spread through the graph of associations. The idea is that the higher the associations strength between situations (perceived image or template image), the more probable it is that it is going to be activated. So far, this is by common spreading activation algorithms as described in Chapter 3.2.1.

However, in the SiMA model, the association strength is not the only factor that influences the activation. The drive state and emotional state does it as well. Further, also the input from the secondary process in the form of phantasy has a significant influence. These are the things that are essential to the system. the psychic potential is introduced to capture all influences with a variable for the algorithm. The *psychic potential* is a concept to sum up for the attraction of psychic intensity of a template image.

The question is what properties this psychic potential could have. First, the higher the association weight and total quota of affect of the image, the more psychic intensity should be allocated from the source. On the other side, the system should have the chance to activate template images that may not be the most important priority of the system, but where the similarity to the currently perceived image is significant. For instance, if a template image is very similar to the source image, the association weight is high. Assume that it has a low weighted quota of affect. Then it can still allocate some psychic intensity. In the opposite case, if the weighted quota of affect is high, but the similarity is low, maybe the same amount of psychic intensity can be allocated. The meaning of the system would be that it would provide the agent to explore an opportunity in the way of satisfying a drive.

So even if the quota of affect would be theoretically 0.0, the psychic potential would still be > 0.0 . Finally, the input from the secondary process has to be considered. The easiest way would be to define some reinforcement factor that represents how much the comparison with a template image from the secondary process would count. Based on the argumentation above, the psychic potential is assumed to have the following function:

$$\psi_j = R_j w_{jk} (1 + |A_j|), \quad (3-1)$$

ψ_j is the psychic potential of the template image j ,

R_j is a reinforcing factor to add extra psychic potential if template images from the secondary process are matched too,

w_{jk} is the association weight between two the source image k and the template image j ,

A_j is the weighted quota of affect of drive meshes and intensities of emotions of the image j .

The purpose of the reinforcement factor R_j is to allow the secondary process to influence the memory activation of the primary process. That is because it is only here, where template images as experiences can be activated in the SiMA model. The weighted quota of affect A_j shall represent how important this particular template image is in a certain drive- and emotional state of the system. Therefore, it makes sense to weight the memorized drive meshes in the template image with the quota of affect of the corresponding drive mesh of the drive state. The same would apply for emotions. The result would be a drive- and emotion dependent average quota of affect for that template image in a certain system state. Derived from this reasoning, the quota of affect A_j is assumed to be calculated in the following way:

$$A_j = D \frac{\sum_{i=0}^N \sum_{j=0}^M \delta_{ij} q_i^d q_j^m}{M} + (1 - D) \sum_{k=0}^P \frac{q_k^e}{P} \quad (3-2)$$

A_j is the weighted quota of affect for the image j ,

D is an arbitrary parameter, which decides how large the part of A_j is made up of drive state and how large the part of A_j is made up of emotions,

q_i^d is a certain drive of the current drive state,

q_j^m is a certain memorized drive mesh in the image,

δ_{ij} is a delta function used in the sum and has the value 1 only if i and j are equal, i.e. they are of the same drive type,

N is the number of drive meshes in the drive state,

M is the number of memorized drive meshes in the image,

q_k^e is a certain intensity of an emotion k ,

P is the number of emotions associated with the image. The emotional component is an average value and is more straightforward.

Process Step 3 – Activation of Directly Associated Images

As described in the previous section, the psychic potentials determine how much of the psychic intensity an image can attract. The task is then to develop a concept for the selection of template images to activate based on known psychic potentials.

As the template images are competing for the same resource, only the relative value matters. It only plays a role how much stronger the psychic potential of a certain template image is compared to the other associated template images. The first step is, therefore, to set up an equation of how much psychic intensity a certain template image would allocate in the case of activation if it would compete with the other template images. The equation would look like this:

$$E_j = \frac{\psi_j}{\sum_{i=0}^n \psi_i} E_k \quad (3-3)$$

E_j is the allocated psychic intensity for image j (see Figure 3.11),

E_k is the available psychic intensity of the source image,

ψ_j is the psychic potential of image j ,

$\sum_0^n \psi_i$ is the sum of all psychic potentials, which are associated with image k .

The psychic intensity is a limited resource. Therefore, each activated template image has to consume some of it at the activation. Else, any amount of psychic intensity could activate the whole long-term memory without getting less. This idea can be realized with a psychic-intensity-consume-value. In general, this value could be different for each image. To activate an image, at least the psychic-intensity-consume-value has to be allocated for the source image.

However, there is a problem with a psychic-intensity-consume-value > 0.0 . If the source image is associated with many other template images, it could be possible that nothing is activated at all as psychic intensity is distributed between several template images. Nowhere the necessary psychic-intensity-consume-value would be reached. As this case does not make any sense, the activation process has to guarantee that at least one template image can be activated, if enough psychic intensity is available for the template image with the highest psychic potential. As a consequence, it has to be calculated how many of the associated images can be activated at most, to be still able to provide at least one image with sufficient psychic intensity.

The statement above has to be included in equation (3-3). It is done by finding out the minimal psychic potential for each image, to get activated, concerning the psychic potential of all other competing images. The following condition is derived from equation (3-3), which satisfies the condition that at least one template image shall be activated if enough psychic intensity is available:

$$E_j^c \leq \frac{\psi_j}{\psi_j + P_j} E_k, \quad P_j > 0 \quad (3-4)$$

E_j^c is the psychic-intensity-consume-value of image j ,

E_k is the available psychic intensity from image k ,

ψ_j is the psychic potential of image j ,

P_j is a factor for image j , which defines how much psychic potential can be covered by other images in order to provide enough psychic intensity for this template image.

In case there are no other template images associated with source image k , P_j is 0.0 and all psychic intensity is received by template image j . The psychic-intensity-consume-value has to be smaller or equal the allocated psychic intensity. Equation (3-4) can therefore be rewritten in the following way:

$$P_j \geq \psi_j \left(\frac{E_k}{E_j^c} - 1 \right), \quad P_j > 0 \quad (3-5)$$

The selection of associated template images is made by sorting them by ψ_j , calculating P_j and then activate template images until equation (3-5) is no longer satisfied. In order to fully understand the concept, it is recommended to look at an example calculation with these equations in Appendix C – Example Calculations in the chapter Example of Activation of Template Images. The result is an incremental activation process that guarantees that at least one template image is activated if enough psychic intensity is available. The activation stops as soon as there is not enough psychic intensity is available.

As soon as the psychic intensity assignment has been calculated, it is distributed. Each template image receives the defined amount of psychic intensity. The psychic intensity is consumed for the activated template images is reduced. The remaining psychic intensity is then available for activation of further template images in an equivalent way. It is the core idea of spreading activation. Figure 3.12 shows the activation after the complete activation cycle on the first level, i.e. the activation of direct activations.

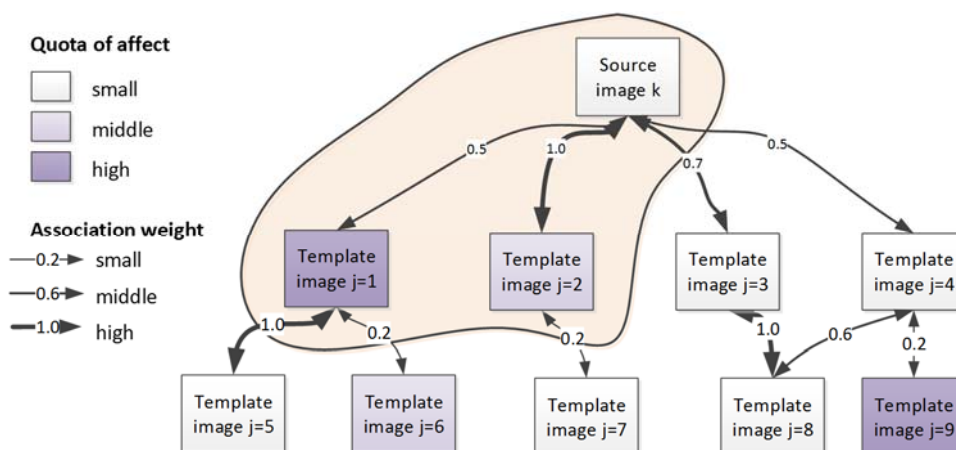


Figure 3.12: Activation of template images with direct associations to the source image

Process Step 4 – Activation of Indirectly Associated Images

During the activation process in an undirected graph, associated template images can activate each other through the predefined associations [WSG+13, p. 6671]. After the initial activation through direct associations in Figure 3.12, the process can be recursively repeated for each activated template image. In the next iteration, each activated template image serves as a source image and repeats process step 2 and step 3.

As long as the psychic intensity is available, the activation propagates along with the associations of the long-term memory. The only difference compared to the first activation is that the source image is a template image and already possess associations to other template images. Therefore, at the second level of activation, no image matching is necessary. Because no image matching is performed, these images are indirectly activated by the original source image. In Figure 3.13, an example of indirect activation is shown. In this example, <Template Image j=1> has some psychic intensity left after consumption, which is enough to activate <Template Image J=5>. In the figure, there were not enough psychic intensity left to activate <Template Image j=6>.

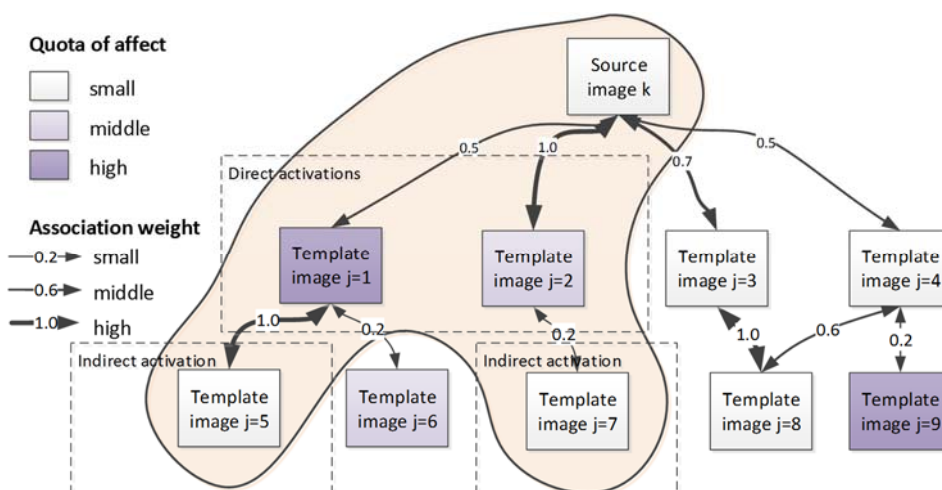


Figure 3.13: Exemplary activation of template images with indirect associations to the source image

3.2.5 Possibilities with Psychic Spreading Activation

In Chapter 3.2.4, the process of the psychic spreading activation concept has been described in its simplest form. A consequence of psychic spreading activation is that it offers possibilities of controlling the activation of images by setting the number of template images that can be activated. For the activation in process step 3, the maximum number of images, which can be activated by a source image, can be set. This limitation is useful to control the spreading depth. Either the psychic intensity is distributed or concentrated.

If the maximum is set high, the available psychic intensity will be spread widely through the possible associations. The search is wide. It is illustrated in Figure 3.14 illustration A. It means that several

images, directly related to the source image can be activated. It is useful to get an overview of what previous experiences are related to the current situation without going into detail of a given situation.

In the case that the maximum number is set low, as in Figure 3.14 illustration B, all psychic intensity is distributed only among a few images, making it possible to activate template images on the depth, e.g. more parts of a sequence.

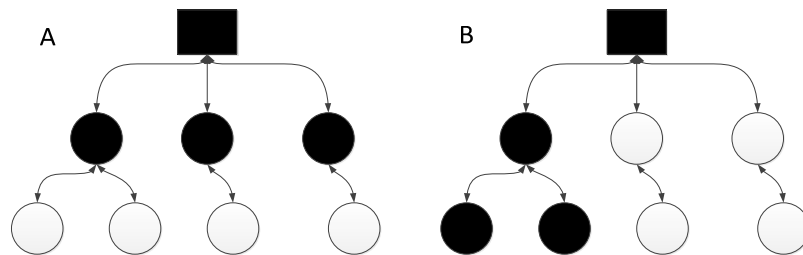


Figure 3.14: Activation of images by spreading psychic intensity widely in A or by focusing it in B

This concept further allows some more parameters to be modified. The standard input to the psychic spreading activation is a perceived image. It is used to activate template images based on perception. Another setting would be to use an already activated, template image instead for activation. In that case, no image matching is done, because the associations are already present. This option seems to be very interesting if the system needs to find out more about some content. Then all available psychic intensity could be concentrated on one template image. This option is the case if an image is sent back from the secondary process by the module Conversion to primary process (F47) (see inputs to the model in Chapter 3.2.3). The usage of this functionality will be further discussed in a later part of the work, where the whole reasoning about experiences is described.

In the two previous modes¹³ of psychic spreading activation, only the primary input is changed. The *primary input* is one source image. The problem is that if there is no perceived image on the primary input, there will not be any matching between the perceived image and template images. It would be like the system would close its eyes and only focuses on one single thought, independent of perception.

As discussed in Chapter 3.2.3, this function should also support decision-making by activating relevant template images for decision-making. That is the purpose of the phantasy that is sent to this function from the secondary process. Therefore, there is a need for a type of activation that considers perception as well as the template images from the secondary process. It should favor those template images that have something to do with the inputs from the secondary process. For that purpose, a *secondary input* is defined. Different to the primary input, its assumed function is only to increase the psychic potentials of those template images, which exactly match the template images on the secondary input. In that way, the secondary process puts a sort of focus on the psychic spreading activation.

The concept of psychic spreading activation has been developed in compliance with the requirements of the SiMA model. It is related to the technical concept of spreading activation but adapted to the

¹³ either a perceived image or a template image is set as the input

circumstances of the SiMA model. As discussed in this chapter, there are many possibilities how this functionality can be run. Its purpose in this work is to provide decision-making with relevant template images, which will be used to create options of what the system can do in a given situation.

3.3 Evaluation of Experiences in the Secondary Process

In Chapter 3.1, acts were defined, which provide the basic sequence of labeled images. They can be interpreted as experiences. In Chapter 3.2, relevant images have to be loaded from the memory in a way, which saves resources and is a plausible process of how human do it. Such a concept is the psychic spreading activation. The purpose of this chapter is to find concepts, which enables the agent to anticipate on situations in its environment and to take appropriate actions based on estimations of

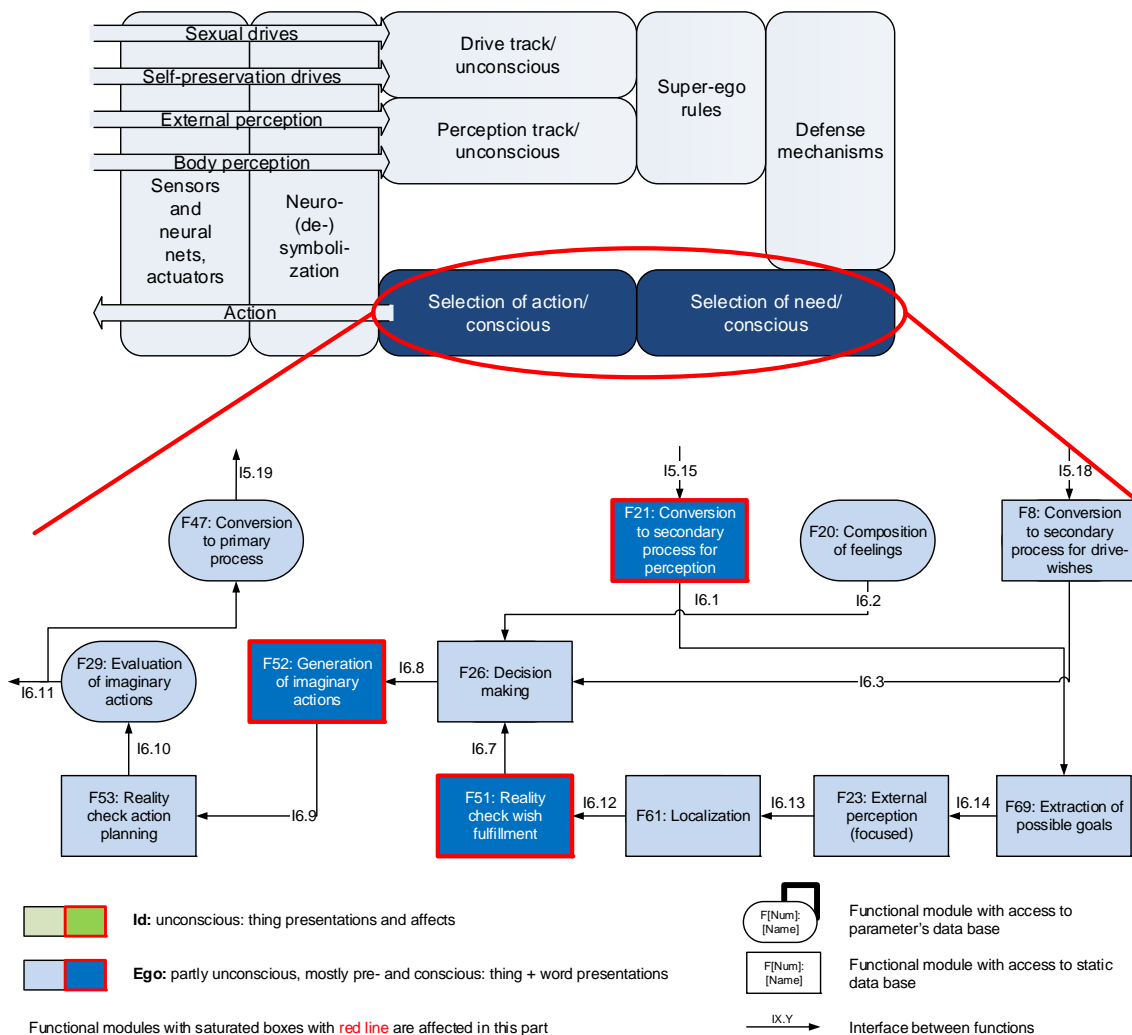


Figure 3.15: Affected tracks of the SiMA model for the concepts of evaluation of experiences in the secondary process

the future. They will be the solution of task 4 from Chapter 1.4 and shall be able to find answers to the following research questions from Chapter 1.3:

Research question 4.1: How shall temporal recognition and confirmation of applicable sequences be realized?

Research question 4.2: How shall information from sequences be supplied, in order to be useful in the decision-making?

In Figure 3.15, the affected tracks of the SiMA model (see Figure 0.1 in Appendix B) are highlighted. The affected modules are displayed below the high-level SiMA model in saturated colors with red boxes around them.

3.3.1 Usage of Experiences in Decision-making

The task of this chapter is to translate the functionality of the bionic model of the human mind regarding the usage of experiences into technical concepts of a cognitive architecture. As an introduction to the problem, an observed example from reality will be described. A small child observes its parents in the preparation of a bottle with apple juice. The child notices the following steps: First, an empty bottle is obtained. Then, the apple juice is taken from the refrigerator. The bottle is filled up with a third and subsequently filled with water. Finally, the closure head is mounted. After watching this sequence of events several times, the child starts to interact proactively as it recognizes the situation. As the child sees that a new bottle is obtained, it opens the refrigerator in advance, to provide the apple juice for the parent. The child has learned to use its experiences to predict the next steps of a sequence of events and to act proactively on it. Throughout evolution from solitary to social animals, the capability to recognize the intention of others has been favored [Wil12, pp. 231-254]. Due to its social structure, this ability is especially strongly developed in human.

The model development of the secondary process is not the focus of psychoanalysis (indirectly concluded from [BDD+14, pp. 79-85]). However, the concepts can be derived from previous work. The data structure of an act was described in Chapter 3.1.2. The question is where in the SiMA model, this act is created. According to [DSBD13, p. 6668] “*During the transition from primary to secondary process, wordconceptions are assigned to these thing-conceptions, thereby causing the contents to become temporally, logically and conceptually structured*”. In Chapter 2.2.3 and in Figure 3.15 of the SiMA model, the transformation of primary into secondary process data structures happens in the module Conversation to secondary process for perception (F21). In order to be compliant to the SiMA theory, all primary process content has to pass the defense mechanisms, in order to be available to the secondary process. Therefore, template images, which form the base for acts, can only be activated in the primary process through psychic spreading activation (see Chapter 3.2) as all experiences have to pass the defense mechanisms. Otherwise, the defense mechanisms would not be a necessary component of the architecture. Therefore, it is assumed that only at this place in the model, acts can be loaded from the data storage.

Further, [DSBD13, p. 6669] says that “*The selection track then takes care of planning, which always includes the simulation of several possibilities ... and executes the action it determines to be optimal in the situation*”. This statement corresponds to the module Generation of imaginary actions (F52) in

Figure 3.15. In a broader sense the simulation of several possibilities can be interpreted to be equivalent to case-based reasoning [RW13, pp. 3-4]. In technical applications, simulation usually means that there is some mathematical model behind. It does not seem very close to the human way of reasoning. In case-based reasoning, there is no single model. Instead, actual cases are used to find out what consequences certain actions have in the environment. The idea is just to look at similar experiences and to assume that the same thing will happen again if the action in the experience is repeated.

Finally, the experiences or acts in SiMA have to be recognized and evaluated by the system. While the images were activated in the primary process, the activated act as a whole has to be evaluated in the secondary process. The act has to be evaluated regarding similarity over time and if it can apply in this situation. Further, if the act is used as a plan, it has been evaluated regarding the effort needed to invest, in order to reach a certain goal. The question is where that could happen. In this case, [DSBD13, p. 6669] states that *“this is strongly dependent on what can be achieved in reality, wherefore a mature human must possess appropriate knowledge of the circumstances of the outside reality, its possibilities and limits”*. The modules in Figure 3.15, which would be relevant for this task would be Reality check wish fulfillment (F51) and Reality check action planning (F53). For acts to make sense for decision-making, they have to be connected with planning, i.e. acts can be the possible options of the system. It would mean that the acts would define the possible actions of the system, if there is no other action generation available. If acts are activated before the decision-making in module Conversation to secondary process for perception (F21), they should receive some basic evaluation before it is decided what to do with them. Therefore, the basic assessment of the acts should take place in Reality check wish fulfillment (F51). Reality check action planning (F53) could be used for modifying actions extracted from acts.

The result of the usage of acts in decision-making is that if a certain action recorded in an act can satisfy a drive and the act fits into the perceived image, then the expected outcome of that action will probably satisfy the drive this time too. In that way, the acts in decision-making form the foundation of anticipatory actions in SiMA. Anticipatory actions correspond to the actions of the small child in the example at the beginning of the chapter.

3.3.2 Translation of Bionic Concepts into Technical Functions

The purpose of using experiences in decision-making is to gain the ability to estimate what happens next. In the simplest case, it is used to estimate the consequence of an action executed in a given situation. This statement is derived from the matter of fact that many situations repeat over and over again and therefore it is useful to use personal experiences. Further, the SiMA concepts support this functionality.

In other cognitive architectures, a sort of schemes is used in a similar way as acts in their simplest form are supposed to be used. For instance, in Icarus [LCT11, p. 12], a skill consists of preconditions, an action and the consequence of that action. By comparing the preconditions of the skills with the perceived situation in the environment, the system is informed that this action is applicable here and will take the system one step closer to its goal. In SiMA, acts offer the possibility to be used in the same way, to process imaginary actions [BDD+14, p. 96].

The act concept as defined in Chapter 3.1.2 allows the further extension. If the consequences of an act are the preconditions of another act, they can be put together in a longer sequence. Such a sequence could represent a whole experience with a series of actions. An example would be the following path in a virtual world by using the positions of the landmarks as preconditions and consequences of the own actions as changes of positions of the landmarks along the road like in Figure 3.16. Acts can also describe the intention of others. If the agent sees another agent that moves towards a food source, it could suppose that the other agent wants to eat it and concludes that the other agent is hungry. It is shown in the act of Figure 3.7. In the following, some concepts and terms regarding the acts will be described.

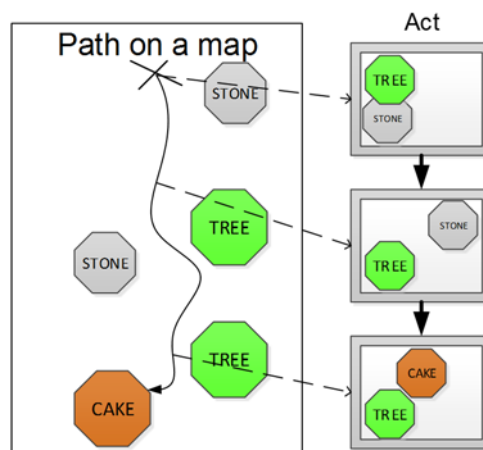


Figure 3.16: Usage of acts in a virtual world for path finding

As will be seen in the next chapter about deliberative decision-making, acts can be used as plans. In each cognitive cycle, from all activated acts, action plans will be developed. However, only one of the available plans will be executed, while most of the other plans are disposed. Due to new information from perception, drives and emotions, these plans are continuously adapted. For instance, plans from an act of path a virtual world like in Figure 3.16 can be replaced if the drive state is changed.

Perceived Image Match

A term that is frequently used in this chapter is the *perceived image match*. It describes the similarity between any image and the perceived image. The determination of the perceived image match has been described in Chapter 3.2.2. The perceived image is a particular case because one image is completely specialized. The term is often referred as *PIMatch* in the pictures of this work.

Moment

To be able to use acts as proposed, the system first has to find out how well an act fits into the current situation. Usually, that would be the image, which is the most similar to the perceived image. The image, which is analyzed to be the best match to the currently perceived image will be called the *moment*. Here is an analogy to describe the moment: If the act would be a movie and the agent has seen this movie before, and it perceives a scene of it, then the moment is that stored scene within the

film, which fits the best with the perceived scene. The moment is used for the agent to know what is happening right now. Moments can, therefore, be seen as roles, which the template images take depending on the context. They are no data structures.

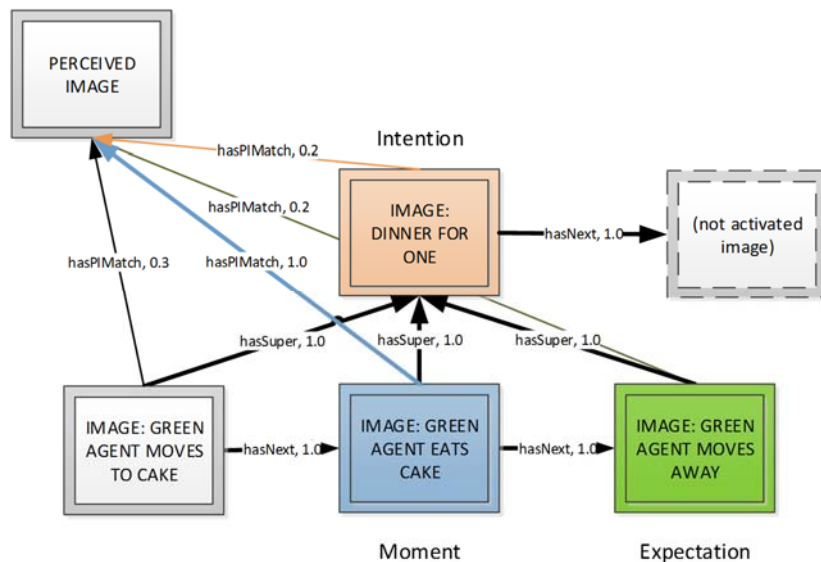


Figure 3.17: The secondary process data structures of an act in SiMA

In Figure 3.17, an example act in SiMA is illustrated. It represents the same act as in Figure 3.7, but with more details regarding act analysis. From the match with the perceived image, which is demonstrated in the semantic association with predicate $\langle \text{hasPIMatch} \rangle$, the highest match value can be found in the image $\langle \text{GREEN AGENT EATS CAKE} \rangle$. This image is most similar to the perceived image and is therefore considered as the moment of the act.

Expectation

As soon as the moment has been recognized in an act, its added value is obvious. It can be reasoned that if the system knows what the current situation is and it matches well, then it is expected that the further template images of the act will match as well. Therefore, the next following template image can be called an *expectation*. In the data structures, the expectation is connected to the moment with a temporal association. In Figure 3.17, this type of association is represented with the predicate $\langle \text{hasNext} \rangle$. There, the expectation is represented by the label of the template image with the value $\langle \text{GREEN AGENT MOVES AWAY} \rangle$.

Intention

An act could theoretically be just a sequence of images with temporal associations. However, this is not very useful. The act has to be defined as one unit as well. There has to be a data structure that gives the overview of the act as a whole. For that purpose, the *intention* is defined. Because the intention shall be the overview of the act, a hierarchical association connects it to all of its sub-images. In the movie analogy of an act, the intention would represent the booklet DVD cover, where data about the

act are stored. In Figure 3.17, the intention can be found in the super-image of the moment and the expectation. A hierarchical association connects them. In the figure, the predicate `<hasSuper>` is used to represent that.

Confidence, Act- and Moment Confirmation

The need for a confidence concept will be demonstrated with an example. The exact configuration and values are arbitrary. Here, they are only used for demonstration of the concept. Suppose there are two acts activated, which are very similar at the beginning of the sequence and first after a while, they differ. An example can be seen in Figure 3.18, which uses the SiMASin World as a source. The first row of template images are the perceived images in four succeeding cognitive cycles. They contain the object representatives of a `<RED AGENT>`, a `<STONE>` that represents a toilet and a `<CAKE>` that represents a food source. In the sequence of perceived images, the `<RED AGENT>` moves to the `<CAKE>` and eats it. The second and third row represent the matching acts `<Act 1: Go to Cake>` and `<Act 2: Go to Toilet>`.

At the beginning of the sequence, it is impossible for the agent to know, which of the two acts matches the situation and whether any of them matches the sequence of perceived images. Although the perceived image match is 1.0, it does not mean that the act fits the whole situation, maybe only to a small part of it. Therefore, the act has to be confirmed in several succeeding cognitive cycles.

A measurement for the confirmation of an act has to be defined: the *act confidence*. It tells the agent how certain it is that the act matches the situation over time. It also describes the main role of the expectation: to confirm an act. If the previous expectation is equal to the current moment, the expectation has to be confirmed and therefore, the act confidence shall be increased. In the opposite

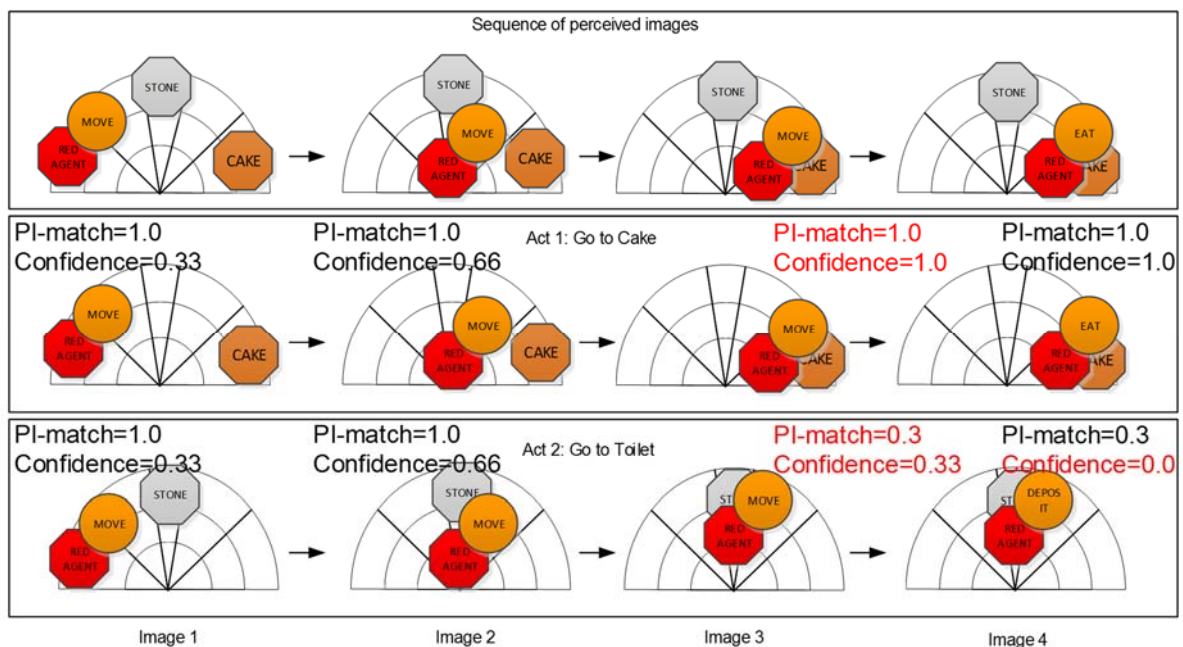


Figure 3.18: Functionality of confidence for similar acts

case, the previous expectation does not match the current moment, and the act confidence shall be decreased. The act confidence is supposed to be a mean of act evaluation. The higher the act confidence, the more important is the act for the agent because it has a higher reliability.

In the example of Figure 3.18, the act would be completely confirmed if at least three images of the act match the current situation. It is the case of <Act 1: Go to Cake>, where the perceived image match is 1.0 for all images and therefore the act confidence increases from 0.33, 0.66 to 1.0¹⁴. After three images have been processed, the agent is sure that the <GREEN AGENT> goes for the <CAKE>. The opposite situation is illustrated in <Act 2: Go to Toilet>, where the agent goes to the toilet. There, the act confidence first increases from 0.33 to 0.66 and then back to 0.33. For the act, the agent is almost sure that the <GREEN AGENT> does not go to the toilet.

Finally, there needs to be another type of confidence: the moment confidence. While the act confidence is attached to the intention and keeps information about the whole act, there has to be confidence value that decides represent which of the labeled template images within an act is the actual moment.

For that purpose, this confidence will be called the *moment confidence*. It informs the system which of the sub-images of the act, is the current moment. In an act, there may be more than one image that has a perceived image match of 1.0. However, they cannot both be moments. Therefore, in a sequence, where the labeled template images are strictly ordered, the labeled template images, which are most plausible to be the moment, will also get a higher moment confidence. It is assumed to be the case if either the previous moment is determined to be the moment again or if the expectation of the previous cognitive cycle is the new moment.

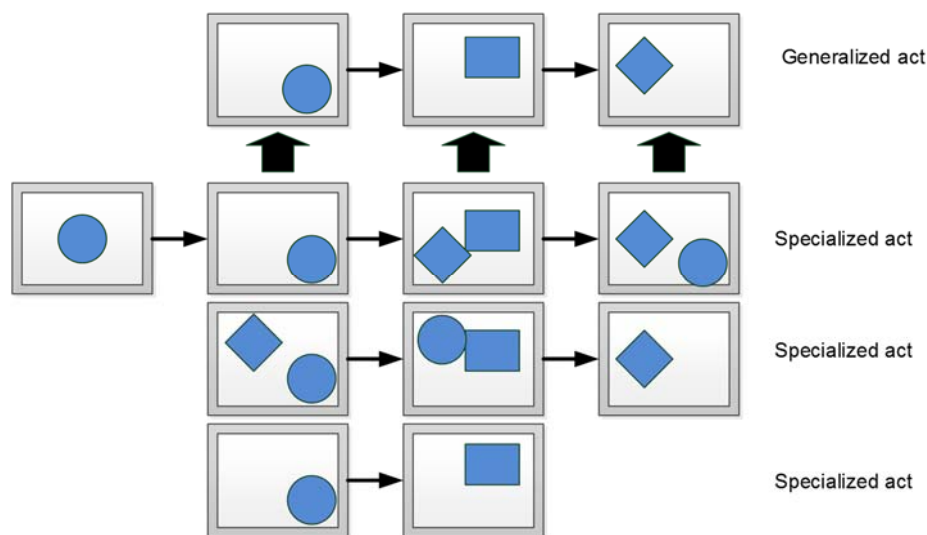


Figure 3.19: Generalization of acts with template images

¹⁴ Numbers and values 0.33, 0.66 and 1.0 are arbitrary chosen for the example

Abstraction of Acts

As the name says, a template image provides a template. In Chapter 3.1.1 it was described what can be done with a template image and that there is a possibility to generalize it. Two types of generalization can be derived from this: First, the abstraction of the act regarding its matching properties and second, a temporal abstraction. In Figure 3.19, a schematic illustration is shown. In this case, each template image of a generalized act matches several template images in a specialized act. It means that one generalized act could be perfectly recognized in several different situations.

The other form of abstraction would be a temporal abstraction. A look at the state of the art explains it. Skills in Icarus and schemes in LIDA can be put into a tree structure (see Figure 0.4 in Appendix A). An everyday example would be to describe the process of brushing the teeth with only one image as a part of a bigger process. However, tooth brushing can be divided into smaller parts like grabbing the tooth brush, then grabbing the tooth paste and then adding tooth paste to the tooth brush. The whole process is abstracted into the image tooth brushing. However, to limit the scope of this work, this type of hierarchy will not be implemented and tested.

3.3.3 Reasoning with Acts

The case-based approach in SiMA can be compared to the heuristic search method in Icarus or LIDA (described in Appendix A). In Icarus, only highly generalized skills are used, which would correspond to an act with an intention and two sub-images. The first sub-image would be the precondition for an action and the second sub-image would be the effect on the environment of an action. The action is associated with the first sub-image. In that way, the concept of a skill or a scheme could be implemented in SiMA as well.

Theoretically, SiMA could be extended to allow the same type of reasoning as in the other mentioned architectures of Chapter 2.2. Icarus reaches a certain goal or solves a certain problem by resolving the difference between the current inferred beliefs and its goals. It decomposes the problem into its smallest, atomic parts. They are then solved individually. The SiMA way to reach its goals will be based on case-based reasoning. Here, the goal shall be reached by using already experienced solutions instead. If no similar situations are available, the system would have to try the trial-and-error method.

Some parallels can be mentioned between case-based reasoning with acts and the concept of situation calculus. Situation calculus [McC87, p. 1033] is intended to provide a method to express the effects of actions, independent of the problem setting. It states that if an event is applied to a given situation, another situation will occur. Quantified result-of-axioms define the result of the action. For instance, in the popular Blocks World [KDJ03, p. 2] example, if a block is (1) not too heavy; (2) it is clear and (3) is located at a certain place, an action <move> on that block will cause it to be placed in another location. Frame axioms put restrictions on what cannot happen is an action is executed, e.g. if a block is moved, its color does not change. The frame axioms are necessary to be able to infer about the states.

Further, situation calculus is only applied to discrete events, and it does not cover concurring or continuous events. At the creation of situations, the challenge is to ensure that the frame axioms¹⁵, which are used, are not contradictive [McC87, p. 1033]. A comparison can be made to template images and acts in SiMA. Just like in situational calculus, those template images would define discrete states, where the moment as defined in previous chapters has to be matched to a certain degree to a perceived image. To get to the expectation, a change of the environmental state has to happen, either through own actions or external events.

Until now, the concept of frame axioms has not been used in SiMA as it seems to contradict the way of human thinking. Instead of analyzing the information that is available, frame axioms force the analysis of everything that is not available in the act instead. A concept of context would maybe better fit into SiMA, where certain contexts are associated with the acts. These contexts would restrict the applicability of an act.

3.4 Deliberative Decision-Making

In the preliminary work of SiMA in Chapter 2.3, the starting point was a single cycle decision-making. To be able to process experiences in the form of acts in that way that was described in Chapter 3.3, decision-making has to be adapted. This adaptation is covered by two research questions from Chapter 1.3:

Research question 5.1: How shall decision-making be extended to consider temporal data structures?

Research question 5.2: How shall internal actions be realized within the decision-making?

In the following chapter, a solution according to task 5 from Chapter 1.4 will be developed. Figure 3.20 highlights the affected tracks of the SiMA model Figure 0.1 in Appendix B. The affected modules are displayed below the high-level SiMA model in saturated colors with red boxes. The model development of the secondary process is not the focus of psychoanalysis (indirectly concluded from [BDD+14, pp. 79-85]). However, there are some available constraints from the SiMA model that can be used. The functional modules of Figure 3.20 have been defined by psychoanalysis at a high level [BDD+14, pp. 135-141] and their functionality will be described in the following chapters.

3.4.1 Requirements on Decision-Making

A single cycle decision-making means that an external action¹⁶ was selected and executed in each cognitive cycle as a reaction of perceived inputs. Decision-making did the following. First, it selected the most urgent drive to satisfy. Then, it did an evaluation of all perceived object representatives to see if an object representative could fulfill that drive. Then, if so, an action was extracted from predefined schemes of actions. Finally the extracted action was executed. In Use case 1 of Chapter 1.3, it meant that if the agent had hunger and saw a certain food source, a predefined set of actions

¹⁵ Axioms are used here in the context of semantic web and are equal to restrictions of a concept

¹⁶ External actions affect the environment, internal actions only affect the inner state of the system

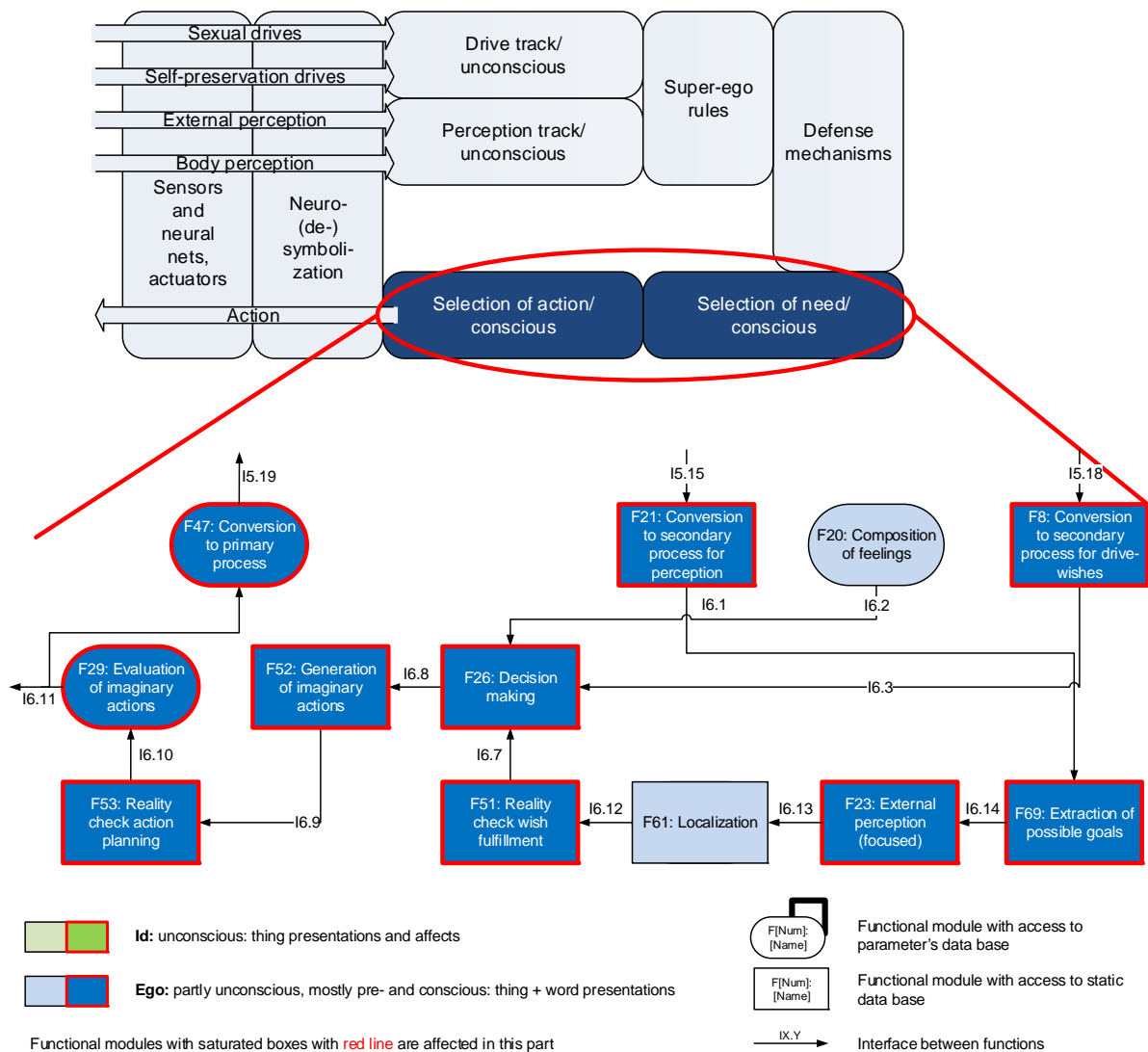


Figure 3.20: Affected tracks and modules of the SiMA model for the concepts of deliberative decision-making

brought the agent to the food source and it ate it because it was the only associated action. Here, the first requirement can be stated:

Statement 3-7: In a bionic system, it should be possible to associate other actions like share or give with the food source.

The second requirement comes with the introduction of acts. The single cycle decision-making is insufficient due to a certain reason: The analysis of acts needs information from previous cognitive cycles. Especially the calculation of the confidence from Chapter 3.3.2 requires that information, which is created in previous cognitive cycles. The challenge in the act recognition is to find out if an act can be confirmed or not. Therefore, the following statement can be formulated:

Statement 3-8: It has to be possible to retrieve information about the evaluations of acts from previous cycles

Statement 3-9: It has to be possible first to analyze an act to see how well it fits into the current situation and then decide about an external action, i.e. a multi-cycle decision-making.

Additionally, the statements above can be argued in the following way. According to the SiMA model theory in Chapter 2.3.5, all activated content in the primary process has to pass the defense mechanisms. To be able to analyze an act correctly, some vital template images like the intention have to be activated first. It means that if some parts of an act shall be activated, it has to happen in a succeeding cognitive cycle, and the decision-making has to wait for this be done before a decision is made.

The third argument of a multi-cycle decision-making is provided by the state of the art in Chapter 2.1.1 and the process in Figure 2.1. There, the main task of decision-making is to do the following:

1. Gather the goals of the agent.
2. Propose options for what is possible to do to bring the agent from the current state to a state closer to its goals.
3. Evaluate and analyze these options regarding fulfillment of the goals.
4. Select the most promising option to execute.

As discussed in Chapter 2.1.1, a deliberative subsystem usually works serially, i.e. it takes one decision of an action in each cognitive cycle. The decision of an action does define not only external actions but also how internal resources shall be used to process the options with internal actions. The key issue here is performance. Assume that all options could be completely analyzed in a parallel working decision process. It would be an applicable approach for solutions of smaller problems like chess computing, where a computer program calculates all possible outcomes for the next steps and from that concludes how to move [New03, pp. 336-339]. However, it would also be a waste of resources as most of the resources of the system would be used for analyzing and evaluating options, which are not relevant for the agent at the moment.

Therefore, the approach used in many cognitive architectures like LIDA and ACT-R is a limited approach, where only one or a few options are evaluated within a cognitive cycle. For the decision-making in SiMA, it means that the actual selection of an option is stretched over several cognitive cycles, where different options are processed serially. Each option is evaluated regarding its chances to bring the agent closer to its goals. For instance, the system can reason about possible outcomes of an act (see Chapter 3.1.2) or perform imaginary actions internally for some of the options before one of them is selected and executed.

Finally, acts may not be the only type of options available to decision-making. If the previous work is only extended, the predefined action patterns, which can be interpreted as instinctual actions will still be there. Such a template action pattern will both need any analysis like an act. Further, future technical applications may have some model that is used to perform imaginary actions for evaluation. It would require an own process. Derived from the fact, that there are multiple types of processes, a requirement can be formulated:

Statement 3-10: Decision-making has to capable of handling different types of options with different processes.

The task is now to develop a concept that fulfills the setup requirements.

3.4.2 Preparation of Primary Process Data Structures

Decision-making involves almost the whole secondary process (see Chapter 2.3.1). However, to be usable for decision-making, the inputs from the primary process have to be converted into secondary process structures and be prepared for usage in the decision-making.

Drives to Drive Wishes

The first input to the secondary process is a collection of drive meshes from the primary process (see Chapter 2.3.5). They represent the main type of motivations of the SiMA model (see Chapter 2.1.2). In the functional module Conversion to secondary process for drive wishes (F8) of Figure 3.20, drive meshes are converted to drive wishes, which are secondary process data structures. A *drive wish* defines the desired, homeostatic state, which the SiMA agent shall try to reach, i.e. the drive wish sets out a system *goal*. Thereby, the drive aim is converted into a *drive action*; the drive object is converted into a *goal object*, and the quota of affect is converted to an *importance* value.

The *importance* will be the base of evaluation in decision-making. SiMA possesses a wide range of evaluation possibilities, e.g. drives, feelings, and effort. The importance is introduced, to bring all types of evaluations to the same base. In that way, a direct comparison can be made between all data structures in the decision-making. It is the way, how feelings can be compared with drive wishes.

A typical example is the homeostatic value of low blood sugar, which generates a desire to eat anything. If the drive wish is fulfilled, the blood sugar level increases. The task of decision-making will be to find options that can satisfy the drive wish concerning other concurring drive wishes.

Emotions to Feelings

According to the SiMA model in Chapter 2.3.5, the emotions from the module Composition of Feelings (F20) are converted to feelings. The module is one of the conversation function from the primary process to the secondary process. While emotions only apply on unconscious data, feelings apply on conscious data in the secondary process. To limit the scope of decision-making, here, feelings only have the purpose of letting the system react to external stimuli, i.e. either avoid something or possibly enhance the achievement of a given situation. For instance, if the feeling <ANXIETY> is present in the system, solutions should be preferred that bring the system away from that situation.

Activated Template Images to Acts

According to the SiMA model in Chapter 2.3.5, in the module Conversation to secondary process for perception (F21) of Figure 3.20, the activated template images receive their secondary process data structures, i.e. their labels and become labeled images. Still they are unstructured. The structure needed is provided by the act.

As mentioned in Chapter 3.1, there are some differences in the usage associations of the primary and secondary process. A labeled image contains both a thing presentation mesh and a word presentation mesh. The secondary process associations are independent of the primary process associations. From

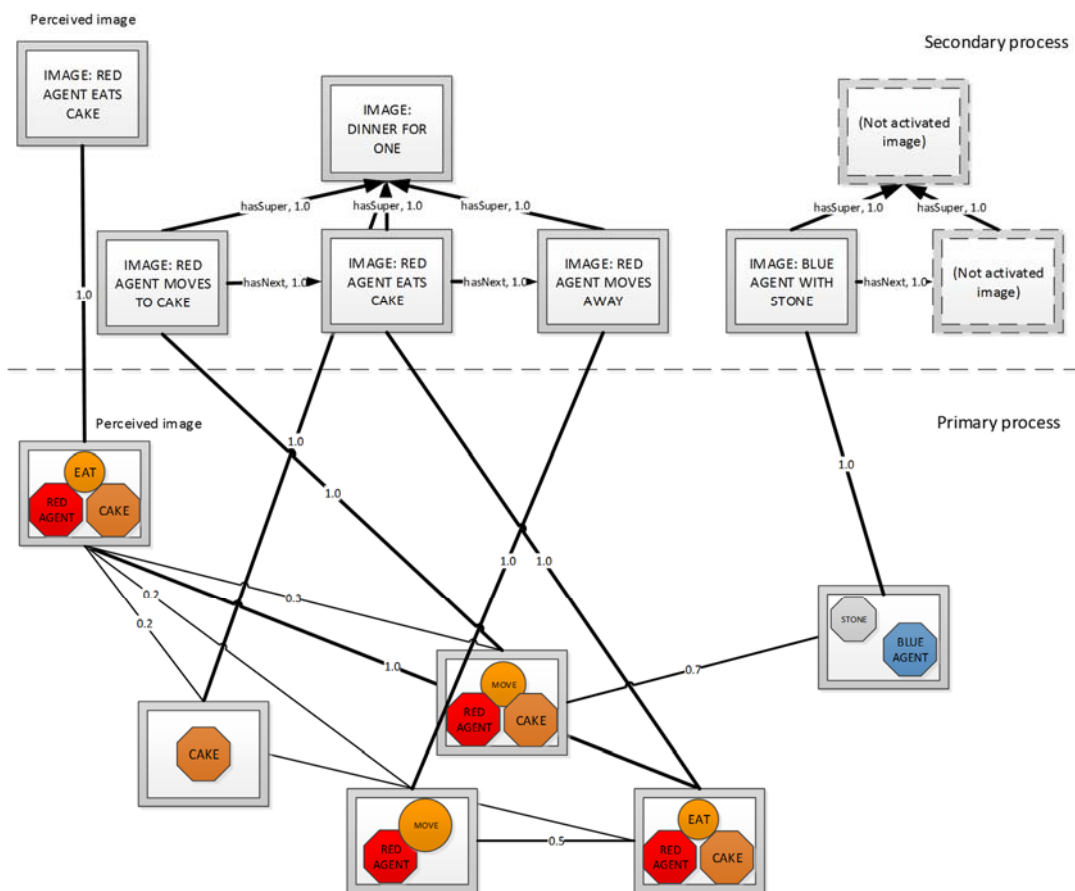


Figure 3.21: Conversation of a mesh from the primary process to the secondary process

that, it can be derived that that template images, which are connected and activated together in the primary process, may not be connected at all in the secondary process.

This can be shown with an example in Figure 3.21. As can be seen, the original, activated graph of thing presentation meshes from the primary process is split into three independent clusters of template images in the secondary process: the perceived image, the act with the label <DINNER FOR ONE> and another act with a single image <BLUE AGENT WITH STONE>. From originally one graph of template images in the primary process, now two acts have been activated.

The graph especially emphasizes the role of the primary process associations in the secondary process. Such functionality cannot be found in the analyzed state of the art architectures. An act can be partially loaded and does not seem to have anything to do with the perceived image. It opens up for the use of alternative solutions in the decision-making, which are not retrieved through the logical reasoning of the matches with the current situations. As can be seen in the human mind, unconventional solutions of problems is sometimes called creativity. This process is possible thanks to the primary process associations.

Acts to Possible Goals

As has been mentioned several times before, in an experience-based decision-making, acts will provide decision-making with options of what the system can do. Because the task of decision-making is to find matching options for the system goals, the acts have to be made comparable to other options and the drive wishes and feelings.

This can be achieved by introducing the possible goal. *Possible goals* represent everything that can be proposed as solutions to fulfill the system goals. Because memorized drive meshes of the perception track in the primary process are the same data structure as drive meshes of the drive track. They can be made comparable. Each memorized drive mesh of the object representatives in the intentions of the acts, therefore, forms a possible goal. Now they are in a format, which makes them comparable with the drive wishes. The same is valid for memorized feelings based on memorized emotions (see Chapter 3.1.1).

It is worth noting that only the memorized drive meshes of the intentions are used as possible goals. It can be argued by saying that the intention represents the whole act, but abstract and without details and it concludes the happening of the acts. Otherwise, there would be much redundancy in the possible goals if they were based on every activated labeled image.

Perceived Image

Just as in previous work, the perceived image is enhanced with secondary process data structures in the module Conversation to secondary process for perception (F21) of Figure 3.20 and passed as it is to decision-making.

3.4.3 Process of Making a Decision

With all inputs to decision-making defined, the next step is to identify a decision-making functionality that fulfills the requirements of Chapter 3.4.1. According to the SiMA model of Chapter 2.3.5, decision-making in SiMA is supposed to be a two-stage, evaluation and selection process. In the first stage, the goal to be fulfilled is decided, i.e. the “what to do” and in the second stage, which method to do it with, i.e. the “how to do it.” Derived from this principle and the requirements, a decision process that is illustrated in Figure 3.22 is developed. The figure shows the process steps. The method shows how data is processed by different functionalities of the system. In SiMA, also the novel concept of using feedback from previous cognitive cycles is introduced. In the following, each step of the proposed decision process is described. For more details on the decision process, see Appendix B, Figure 0.11, where a detailed process model at abstraction level 5 is provided, which is close to the implementation.

Propose Options 1st Run: Extract Possible Goals, Drive Wishes and Feelings

The first step is to extract possible goals in (1.3), feelings in (1.2) and drive wishes or needs in (1.1). The possible goals of (1.3) are the options (see Chapter 2.1.2) of the system. As described in the previous section, they are extracted from three sources: from the perceived image; from acts or drive wishes. The reason, why possible goals shall be extracted from drive wishes is because here a search

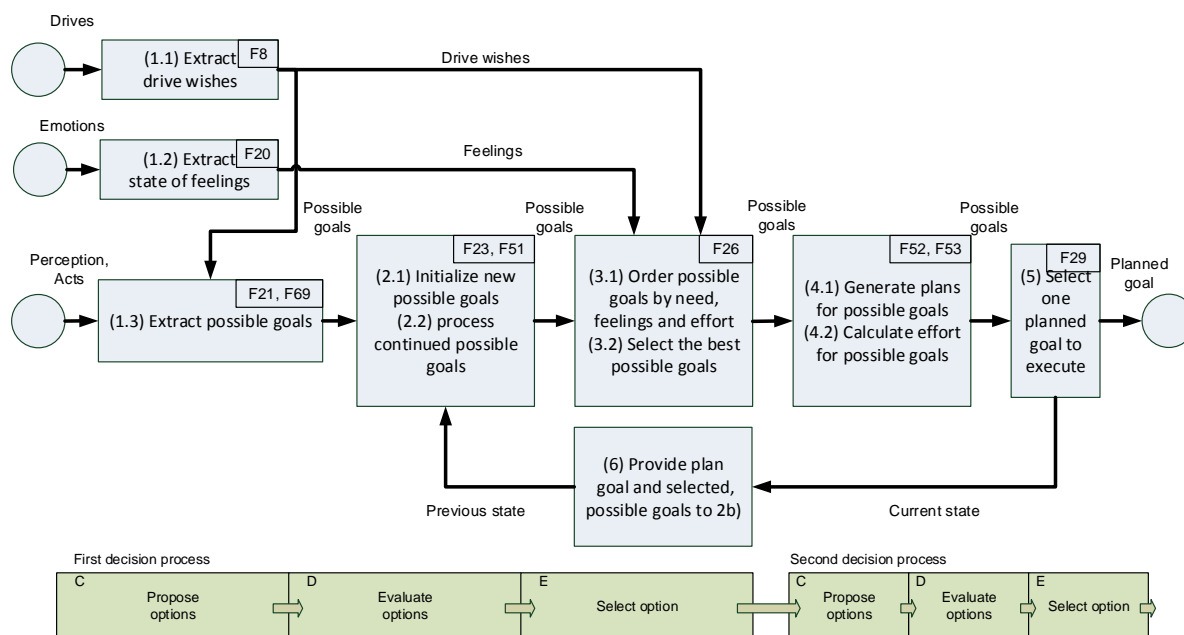


Figure 3.22: A process model of decision-making in SiMA and the correspondence to the decision-making process of cognitive architectures

behavior shall be triggered. If a goal object cannot be found in either the acts nor in the perceived image, it should be searched. Therefore, this type of possible goal will be needed.

In Chapter 2.1.2, the process steps of data in a cognitive cycle were described. In SiMA in Chapter 2.3.5, it was stated that the process steps propose options, evaluate options and select option is executed twice. Therefore, this is the start of the first run of propose options. The purpose of the first run is to narrow the number of possible goals by filtering them for the most important drive wish or feeling. The defined process steps from Chapter 2.1.2 are shown at the bottom of Figure 3.22. The identifiers in the top right of each block represent the modules of Figure 3.20, which are covered. Possible goals represent the whole space of options that the agent can select.

Evaluate Options 1st Run: Initialize Possible Goals

Then, the process step evaluate options starts in (2.1) of Figure 3.22. Previously, all possible goals were extracted. However, to be useful for decision-making, some basic initialization has to happen. It means that each of them is analyzed, and a first estimation of the effort to fulfill them is defined. It is a first reality check to check roughly what possible goals do have a chance to fulfill the drive wishes. In order to save system resources, it makes sense to make a rough estimation of their applicability before deciding about them. The effort lowers the importance of unreachable, possible goals. Step (2.2) will be described in the next cognitive cycle.

Select Option 1st Run: Make Decision based on Drive Wishes and Feelings

Process step (3.1) and (3.2) of Figure 3.22 of the phase select option then takes place. Here, the possible goals are evaluated considering the drive wishes from (1.1), feelings (1.2) and the effort.

According to the SiMA theory of Chapter 2.3, the planning in the secondary process follows an economic optimization. Different to the primary process, the effort of reaching possible goals, therefore, plays a major role. Logically, the effort usually lowers the importance value of a goal and the possibility of fulfilling a drive wish increases it. Drive wishes make that the agent acts proactively, to satisfy bodily needs. Feelings allow the agent to react to external situations and to avoid unpleasure by increasing the importance of such possible goals in a given situation.

The purpose is then so select those possible goals that can satisfy the most important feelings or drive wishes. After ordering the possible goals by importance in (3.1), a variable number of the possible goals with the highest total importance are filtered to continue in (3.2). Normally, these possible goals will fulfill the strongest drive wish.

It ends the first run of the steps propose options, evaluate options and select option in 2.1.2. At the same time, the second run starts, because from now on, the possible goals demand deeper analysis regarding planning. This funneling of selectable goals is necessary, in order to be able to focus only on the most important, possible goals. The rest of the possible goals are discarded.

This process can be interpreted to be almost equivalent to the usage of attention codelets and conscious broadcast in LIDA [FF12, p. 107]. Although both models are based on different theories, they define similar functionality. However, as both SiMA and LIDA claims to have one or more theories to model the human mind, it cannot be a coincidence that they express similar functionality although it has a different origin. As mentioned earlier, the development of cognitive architectures is convergent. As in biology, where the eye has been developed independently several times before [Daw01, pp. 159-221], they all have the functionality to catch the light. Here, it could be asked if the concept of drive wishes in SiMA could be realized with attention codelets from LIDA.

Propose and Evaluate Options 2nd Run: Generate Action Plans for Continued Possible Goals

In process step (4.1) of Figure 3.22, action plans are generated for each possible goal and the effort of executing the action plan is analyzed in step (4.2). It is the second run of the decision process steps. Funneling of possible goals has been made in the first run. However, it may occur that none of the possible goals are reachable because the effort is too high. Perhaps one of the previously discarded, possible goals would be better. In this case, a further cognitive cycle is necessary to reevaluate if this would be the case. It means that multiple cognitive cycles may be required to select a planned goal.

Select Option 2nd Run: Select Planned Goal and Action

Finally, in (5) of Figure 3.22, the possible goal with the highest importance is selected as a planned goal. A *planned goal* is a term for the selected possible goal, from which action shall be executed. Because this data shall be used for analyzing acts in upcoming cognitive cycles, all goals from the second phase of the decision-making are then saved in a short-term memory. It stores the state of the agent for some time. It is necessary to do, otherwise, everything had to be done from zero in each cognitive cycle and the system would have to be purely reactive. It is equivalent to a short-term memory as described in Chapter 2.1.2 and it can be compared to the working memory of SOAR [LCC+11, p. 13].

The action plan attached to the planned goal are either an external or internal action. In the case of an *external action*, the action command is sent to the body for execution. In the case of an *internal action*, an internal action command is executed within the model in the next cycle. The idea of using internal actions that trigger functions in upcoming cognitive cycles is an elegant solution to the requirement that the system shall be able to handle different types of possible goals in various ways. It means that there can be certain functions available for decision-making in various situations. The decision of the decision-making is then to select one of these functions to be executed. In that way, the process of handling particular types of goals is transferred from hardcoded functions to the data, which extends the flexibility of the system significantly.

Usually, for each executed, external action, several internal actions are executed first. For instance, the internal function to focus on a goal object is run first and then the external command to move towards it is run. This process requires that several cognitive cycles are run before an external action is executed.

Evaluate Options 1st Run: Analyze Continued Possible Goals

Then, the next cognitive cycle starts. The stored state of the agent that contains the planned goal, the other most important possible goals and the most important drive wishes. This information is then merged with new information in (2.2) from (6) in Figure 3.22. Because each possible goal is handled independently, it enables the agent always to consider new situations and to pause the pursuing of the current planned goal. It is a major difference to ACT-R [TCM03, p. 4]. SiMA is therefore always able to re-evaluate new situations and can change planned goal depending on bodily and situational updates.

It is also the place, where the functions of the internal actions are executed. The reason why these functions are placed in front of the module Decision making (F26) in Figure 3.20 is derived from the SiMA model in Chapter 2.3.5. There, the function focus of attention is placed in the module Focus of attention (F23). It is a typical function that shall be executed on demand and with certain parameters. In one process step, this function shall focus on the goal object and another time, this function shall focus on some obstacles in the vision. As will be described in the following, the focus of attention is only one “on demand function”, which can be used in the system. Because these functionalities cannot be executed before a decision has been made on what action to execute, this problem is solved by the internal actions. The internal actions are put into the planned goal, which then executes functionalities in the modules in the next cognitive cycle. This functionality is a major step towards deliberative decision-making.

3.4.4 Short-Term Memories of the Secondary Process

Human uses a short-term memory to keep activated information available for the functionality of the system [BDD+14, p. 99]. In human, the short-term memory keeps data until about 3s if not refreshed. To realize a multi-cycle decision-making and to enable recognition of acts, it is necessary to store the current state of the decision-making in short-term memory. In SiMA, two types are short-term memories will be used: the working memory and the environmental image memory. They are presented in the following.

Working Memory

According to the SiMA theory of [Dob15, p. 14] and [BDD+14, p. 99], the system has a self-representation. It represents perspectives of how the system perceives itself. One of the most important features is that external data is continuously compared to the self-representation. Through the comparison feelings are generated and through self-reflection, the system is able to evaluate its own actions. A part of the self-representation represents the current state of the system. The current state includes short-term content like the perceived image, the drives and emotions. Additionally, it also includes activated content from the long-term memory, such as activated acts, which describes remembered experiences. Such self-representation data has to have some persistency, i.e. to be stored for multiple cycles. In the area of decision-making, it applies to possible goals. The self-representation fulfills the requirements in Chapter 3.4.1. For that purpose, a *working memory* is introduced in SiMA as a short-term memory. Because the main purpose of it is to keep data for the analysis of possible goals, it is defined as a subset of the extensive concept of self-representation. As concluded in Chapter 2.1.2 and 2.2, no architecture, which works in a deliberative manner, can manage without it.

However, it is not enough just to save the state from the previous cognitive cycle like a cyclic memory in [BDD+14, p. 100]. It has to save the state of the agent with duration over several cognitive cycles, to let the agent access previous decisions. It is relevant in situations, where the analysis of a certain possible goal has given, that it is not reachable. In order not to analyze it over and over again, the result of the analysis is stored over multiple cycles. However, at some point, the previous analysis is obsolete and can be removed from the working memory.

There are two ways of realizing such a memory: Either there is one working state, where all new content is merged with the old content and content is removed based on certain criteria, or the states are stored separately as pages of a chronical. The working memory of SOAR [LCC+11, p. 13] is based on one state. New content is added and content, which has not been accessed in a while fades away below certain activation, where it is removed. For the episodic memory, SOAR saves the state into a long-term memory. It is a proper solution if there is any possibility to write the state to the long-term memory.

In SiMA, learning is not considered yet and therefore, there is no possibility to access previous states in the long-term memory. Therefore, as shown in Figure 3.23, the working memory in SiMA is realized as a ring buffer, where each state is stored separately as a mental situation. A *mental situation* is here defined as everything that is saved from previous cycles, i.e. possible goals, the planned goal and the most important drive wishes.

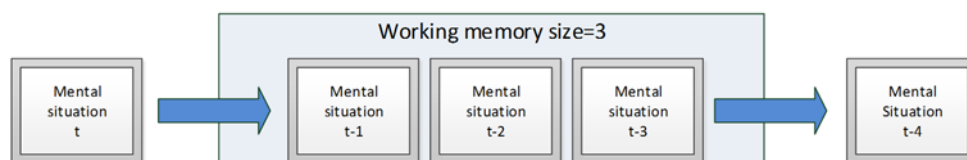


Figure 3.23: Working memory that is constructed like a ring buffer with the newest mental situation at time t on the left and the oldest mental situation at time $(t-4)$ on the right

As the queue reaches its maximum capacity, the oldest mental situation is removed. It allows decision-making to access older content, although no writing to the long-term memory is available. In CHREST [SLG08, p. 100], its short-term memory can keep four chunks. The solution in CHREST inspires the solution in SiMA. However, the concept of four chunks does not apply to the working memory in SiMA. Due to the circumstances of how decision-making in SiMA works, SiMA only preserves entire states.

Environmental Image Memory

Another part of self-representation of the SiMA model [Dob15, p. 14] is the current location of the system. Therefore, a mental map of the environment is needed, for the system to orient itself. While the working memory has the task to keep track of the possible goals in decision-making, another short-term memory has to be used to keep a mental image of the whole situation. It is then also a subset of the self-representation according to the SiMA model [Dob15, p. 7].

The functionality of the environmental image memory will now be derived from the requirements of the process of decision making of Chapter 3.4.3. As described in Chapter 2.3.5, SiMA is supposed to implement an attentional function, the focus of attention. It has the task to filter object representatives from the perceived image, to let the agent focus on only a few of them. It means that the agent cannot see everything necessary to evaluate a situation within one cognitive cycle. The reason is that the agent first focuses on some object representative and then on another object representative. The problem is that for a situation analysis, both object representatives have to be present in the decision-making at the same time. Object representatives are removed by time and added as well as updated by the focus of attention. Such functionality like a mental image can be inspired by the Mind's Eye in CHREST [RLSL09, p. 2]. However, it has been redesigned for the SiMA architecture, considering psychoanalytical constraints.

Another problem, which has to be solved, is how the agent shall deal with object representatives, which are no longer in the immediate perception. For instance, if an agent moves and discovers an enemy agent, a plan to flee will be activated. It causes the agent to turn around and move away. However, as soon as the enemy agent is out of sight, the agent forgets that the enemy has ever existed.

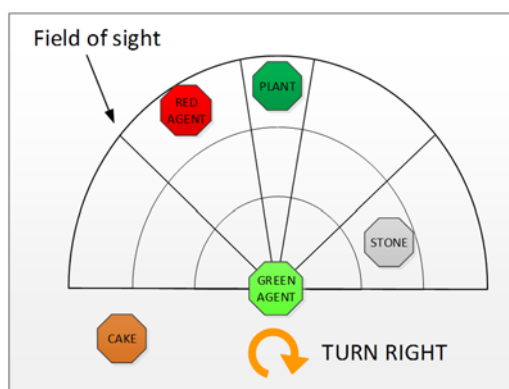


Figure 3.24: A situation in the SiMAsin World for the explanation of the environmental image memory

Consequently, the agent turns back, to pursue its original plan and discovers the enemy agent again. The result is a deadlock. This phenomenon is known under the term object constancy

A solution to the problems described above is an *environmental image memory*. It is an extension of the previously described working memory with additional functionality. The functionality can be described as an example. In Figure 3.24, objects from the SiMASin World (see Chapter 4.1.2) are used. At the current time stamp, the <GREEN AGENT>, which is the subject, sees a <RED AGENT>, a <PLANT> and a <STONE>. In previous cognitive cycles, it had turned right with the action <TURN_RIGHT>. At that time, the <CAKE> was visible, which is now out of sight of the <GREEN AGENT>. In the example, in which the environmental image memory is visualized in Figure 3.25, three perceived images are kept. In every cognitive cycle, a new perceived image is added from the right, and the oldest image is removed. The images are queued, just as the working memory. Three steps ago, the agent did focus on the <CAKE> ($t=3$), then it turned right, focused on the <RED AGENT> ($t=2$) and then on the <STONE> ($t=1$).

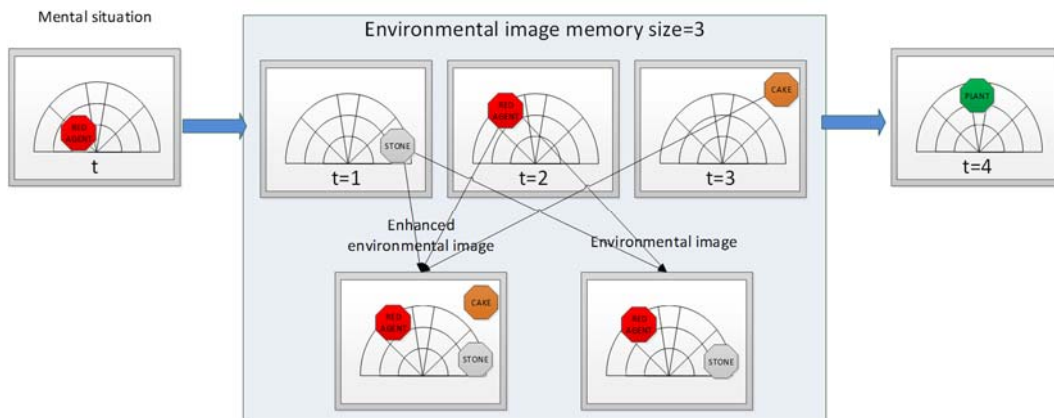


Figure 3.25: Environmental image memory, which is similar built like the working memory

As the agent turned away from the <CAKE>, it disappeared from the field of sight. However, the agent still knows that the <CAKE> exist there somewhere. It is represented by putting the <CAKE> outside of the field of sight in Figure 3.25. The object representatives of the stored images belong to two or three images each: first, the original filtered perceived image, then an enhanced environmental image and optionally also the environmental image if they are still in sight. In the *enhanced environmental image*, all object representatives of the memory are present, including the object representatives, which are out of sight. In the *environmental image*, only the object representatives, which are actually in the current field of view, are represented. At the beginning of each cognitive cycle, the environmental image memory is updated with the perceived image from the functional module Conversion to secondary process for perception (F21). If the positions of the object representatives of the environmental image do not match the current perception, their positions are removed and the associations are removed from the environmental image. They now only belong to the enhanced environmental image. Because only focused content is represented in the environmental

image memory, it only keeps things, which are important for the agent, i.e. which are important to reach its goals.

Different to SOAR, where only one large working memory is used for everything, in SiMA, basic beliefs of object representatives are stored in the environmental image memory and not in the working memory. Furthermore, the working memory of SOAR does not have any limitation of how much can be stored there. This restriction is realized by the queue-based memory structure in SiMA. Finally, compared to the analyzed architectures in Chapter 2.2, SiMA is now the first one to integrate a comprehensive focus of attention mechanism in a holistic architecture.

3.4.5 Functionality Controlled by Internal Actions

In Chapter 3.4.3, the need and usage of internal actions and “on-demand” functions that are triggered by internal actions was discussed. For the implementation of experience-based decision-making, three functionalities of decision-making will be necessary to define.

Focus of Attention

In the SiMA theory, the functionality focus of attention of the module External perception (focused) (F23) was proposed, but never specified for practical usage [Zei10, p. 75], [Deu11, p. 98]. For a complete decision-making, this functionality has to be defined. The definition will be based on the human way of focusing things and the needs of the system.

In this work, the *focus of attention* is top-down controlled as knowledge is used to seek certain external objects or patterns in the perceived image. An example from reality with a person that is untrained in navigating in the city. This person does not notice the same things as someone, who is primed to navigate the city with public transports. The untrained person will consider taking the subway to reach his goal. As he does not have the knowledge of how to get from one point to another with the subway, station signs are not focused on as they are initially no part of any goal. The more often the person



Figure 3.26: Focus of attention focuses on object representatives, which are subjectively important

navigates in the city, the more of his goals can be fulfilled by taking the subway. Eventually, subway station signs will be focused on and in that way be visible for the person in his planning.

The main task is to filter noise that would be disturbing for fulfilling a certain goal. An example of focus of attention in this system would be the following. The vision of a hungry agent is shown in Figure 3.26. A food source <CAKE> is focused on as it could satisfy hunger. This functionality in combination with the environmental image memory is inspired by the solution in the architecture CHREST [RLSL09, p. 2] because here a mental image is also constructed over multiple cognitive cycles.

An interesting point is the benefit of the focused attention, when relative positions are used and the agent has to perform self-localization. The function focus of attention is a matter of limited resources as it is an elegant way to solve the following problem: The images of the primary process do not use any relational structures, only absolute positions of object representatives. In the secondary process, relational structures are suggested, i.e. it is possible for an agent to rotate or shift the object representatives of an image in the mind. There, it makes a huge difference if relations are created between e.g. 5 or 20 object representatives.

Activation of Images through Fantasy

Acts represent experiences as well as generalized action plans and are therefore the main provider of possible goals to decision-making. According to the SiMA theory in Chapter 2.3.5, template images, which form the base for acts, can only be activated in the primary process through psychic spreading activation (see Chapter 3.2), because all experiences have to pass the defense mechanisms before they get to decision-making. Otherwise, the defense mechanisms would not be a necessary component of the architecture. For that reason, it has to exist a top-down influence that originates from the secondary process to the primary process. Without top-down influence, there would be no possibility to activate template images that are relevant for the planning. It is a requirement, to be able to fulfill the decision process of Chapter 3.4.3.

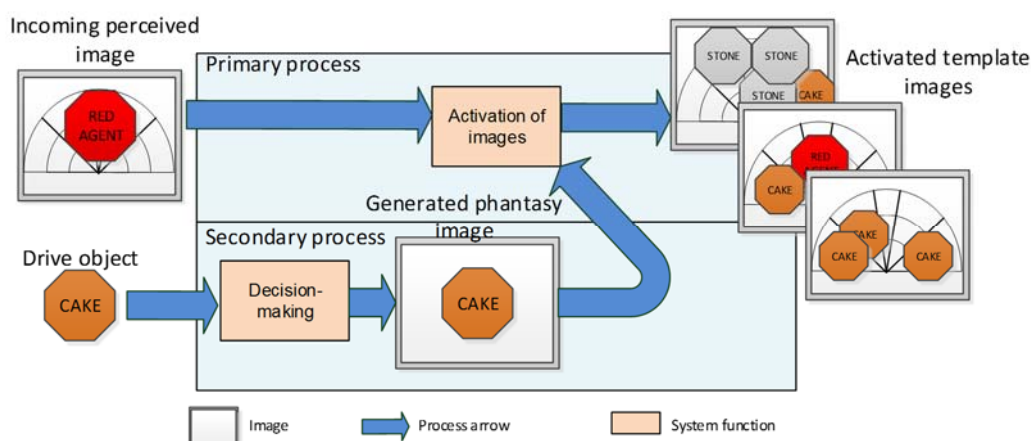


Figure 3.27: Process of activating additional images through phantasy

The psychoanalytical concept of phantasy in the SiMA model of Chapter 2.3.5 perfectly corresponds to this requirement. It is supposed that plans are being sent back to the primary process, to activate more related content. In this case, the plans are acts. Those template images that have not been activated yet and are needed by decision-making can be activated through psychic spreading activation if another part of an act is sent back.

In Figure 3.27, the usage of phantasy in decision-making is illustrated. In the example, there is only a <RED AGENT> present in the perceived image. However, the strongest drive wish is to find something to eat. Decision-making generates a phantasy image based on the goal (drive) object. In this case, it is a <CAKE>. To be able to satisfy that drive, experiences have to be activated, which give the agent options for finding food.

The phantasy image is sent through the functional module Conversion to Primary Process (F47) in Figure 3.20. The label of the image is removed, and the template image remains. As a result of psychic spreading activation, activated template images that contain both the <RED AGENT> and <CAKE> are activated in the example. They will then provide decision-making with new options on how to proceed from the current situation.

Basic Act Analysis

The third necessary on-demand function analysis an act. This function would be the implementation of the whole concept of Chapter 3.3. The purpose is to provide the agent with two things. First, it shall be recognized how well the act fits in the current situation by defining the moment, the expectation and the intention of the act. The moment- and act confidence are the measurements for that. Second, it shall be found out what external action the agent has to perform or what must happen in the environment, to get into an expected situation.

All of the decision-making is design upon requirements that shall satisfy this function. With the basic act analysis, the agent will be able to use its previous experiences to estimate what will happen next in the environment and what is the consequence of its actions.

4. Implementation

“Hasta la vista baby”

[Terminator in Terminator 2: Judgment Day]

Within this work, the SiMAi14 implementation version was developed. As the implementation of the SiMA model in the software project was very comprehensive and all of it would not fit within the frame of this chapter, the content is narrowed to highlight only the most important implementation techniques, which concern the concepts of Chapter 3. The implemented general structure of the SiMA agent has been sufficiently described in [Zei10, pp. 102-104] and [Deu11, pp. 126-132]. The simulation environment and agent body have been described in [Deu11, pp. 116-126], [Muc13, pp. 113-119] and [Zei10, pp. 88-94]. However, the chapter will start with a brief description of the evaluation simulator. Then, the highlights from the implementation will be described.

4.1 Evaluation in the SiMASin World Simulator

Before the actual test cases for the verification¹⁷ of the implemented solutions of the research questions are played through, the software simulator used to realize them is described together with a brief description of the different objects of the SiMASin world. In the following, the configuration of this work is described.

4.1.1 Simulator Setup

The cognitive architecture SiMA will be tested in the multi-agent simulator MASON [7], which has been used as a test bed for SiMA in [Zei10], [Deu11], [Muc13]. As described in [Muc13, pp. 113-119], the purpose of the SiMASin world simulator is to allow different types of agents to act in a simulated world and to interact with objects. Such objects can be walls or stones or energy sources. Stones are obstacles, which cannot be passed through. Energy sources provide the bodies of the agents with energy if they are consumed. Each object consists of a body with sensor and actuators and

¹⁷ The test cases verify the specification created in the models and concepts chapter

depending on the type of object; decision units can be used to control the body. All objects in the world are represented in two dimensions to simplify interaction.

After each object has been registered in a scheduler, MASON works in cycles and four phases are executed in each simulator cycle [Muc13, p. 117]. In the first step, the internal values of the bodies are updated. For instance, it can be the consumption of energy for a certain agent. In the following step, the environmental information of the bodies is updated through their sensors. In the next step, the sensed and internal inputs are processed by a decision unit if available. Finally, the proposed actions of the decision unit are executed in the environment. In that way, the state of the world is altered by the agents.

The agent's states can be viewed through inspectors, to know what happens in the decision-making. The inspectors are attached at some locations within the decision-making. It should be noticed that the inspectors are implemented between the modules in the decision-making [Muc13, p. 126]. It makes the decision-making depend on the operation of the inspectors. These inspectors are fully customizable and provide information about internal body states or sensed data. Sensed data is transformed into an internal representation [Muc13, pp. 126-128].

4.1.2 SiMASin World

Within the MASON framework, the SiMASin World was created, to satisfy the requirements provided by the use cases in the SiMA project.

Simulator Objects

As shown in Figure 1.2, it looks like a cage surrounded by walls (<WALL>) and contains some static objects like an obstacle (<STONE>) and a food source (<CAKE>) as well as the agents themselves (<ADAM>, <BODO>). Note that this is a Ph.D. thesis written in Austria, and the <CAKE> looks like a schnitzel. In the following, the agent, which is the subject is always <ADAM> and the agent that is the target of interaction is <BODO>.

Agent Body

The bodies of the SiMASin agents <ADAM> and <BODO> keep a metabolism, which provides the source of drives for the agent. Each body needs the energy to do anything, and it needs nutrition to exist and to grow. It is represented by energy and health. Energy and health are provided by food that is consumed. For a complex body, there are several types of nutrition needed like digestives or minerals to keep the health of the agent [Deu11, p. 119]. If food is consumed, the stomach is getting full and has to be emptied at some point. The body is also equipped with a stamina system that increases the need to relax after taking action. It means that there have to be some simulation cycles without action. The need to consume food, to deposit waste and to relax after acting triggers drives in the decision-making. As only these drives are necessary for the use case, other bodily functions of the simulator are not explained here. For more detailed information about the body, please refer to [Deu11, pp. 119-125].

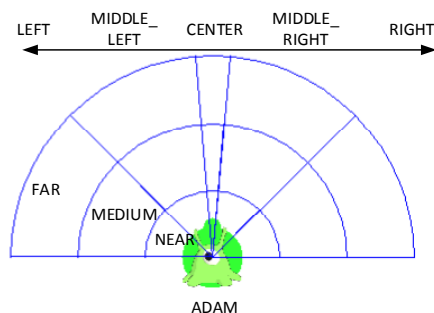


Figure 4.2: The agent as an element in the MASON simulator

In the simulator, each agent has the field of vision as shown in Figure 4.2. Each sector in the area of vision can be determined by a radial coordinate and an angle. The radial coordinates are $\langle \text{NEAR} \rangle$, $\langle \text{MEDIUM} \rangle$ and $\langle \text{FAR} \rangle$. The angles are represented with $\langle \text{LEFT} \rangle$, $\langle \text{MIDDLE_LEFT} \rangle$, $\langle \text{CENTER} \rangle$, $\langle \text{MIDDLE_RIGHT} \rangle$ and $\langle \text{RIGHT} \rangle$. The Actions of the agents on other objects can only be done in the sector $\langle \text{NEAR, CENTER} \rangle$. Therefore, to consume something, the agent has to move in such a way that the food source is put in that sector.

Available Actions

Based on the currently defined bodily needs and the setting of the SiMASin world, the possible actions are listed in Figure 4.1. The transparent actions are available, but will not be handled in the test cases. In the group Move Actions, the agent has the possibility to change its position. In the group Actions related to consumption, actions that are related to the need to eat something are available. If the action $\langle \text{EAT} \rangle$ is performed, the agents consume something. As drives (see Chapter 2.3.5) have libidinous as well as aggressive components, the agent may either $\langle \text{BITE} \rangle$ or $\langle \text{EAT} \rangle$. $\langle \text{BITE} \rangle$ is often an aggressive act against another agent. For coping with stamina, $\langle \text{RELAX} \rangle$ is available. If stamina is high, the agent needs to $\langle \text{RELAX} \rangle$, to be able to act again. If it is time to excrement processed for the consumed food sources, the $\langle \text{DEPOSIT} \rangle$ action makes the body excrement objects. Finally, some



Figure 4.1: Actions in the agent world

actions can be used on any object in any situation. <BEAT> is to attack, <PICK_UP> is to pick up an object and to put it in the inventory and <DIVIDE> is used to split an object.

4.2 Highlights of the Implemented Cognitive Architecture SiMA

The highlights of the SiMAi14 implementation version lays around the decision-making functionality. Much work was done regarding data structures and abstraction of existing functionalities. However, the software engineering part may be only of interest for the developer of the SiMA software. Therefore, a comprehensive software description of the most important concepts is available in Appendix B. There, data structures, psychic spreading activation, act recognition methods and decision-making are described in detail. For this part two things will be in focus: On the image matching algorithm in the primary process and the implementation of decision-making in the secondary process.

4.2.1 Psychic Spreading Activation Image Matching Algorithm

Image matching between two images plays a vital role in the activation of template images. The concept of image matching was described and derived in Chapter 3.2.2. Here, the implemented algorithm is described. While it is enough to state that there exists a matching algorithm in the model, the exact way it works, is dependent on the proposed usage, technology and focus of the work. For this work, an image matching algorithm was developed that is a specific implementation for the SiMASin world (see Chapter 4.1.2).

In the first step, all object representatives and their positions are extracted from the source image as well as from the template image. For the template image, positions are allowed for generalization. As described in the section about template images in Chapter 3.1.1, the generalization means that for an object representative, one or more coordinates are not set. In that way, an object representative is not restricted to a certain position. A generalized position of an object representative may match to several specialized positions. For instance, Figure 4.3 shows how the matching with generalized images looks

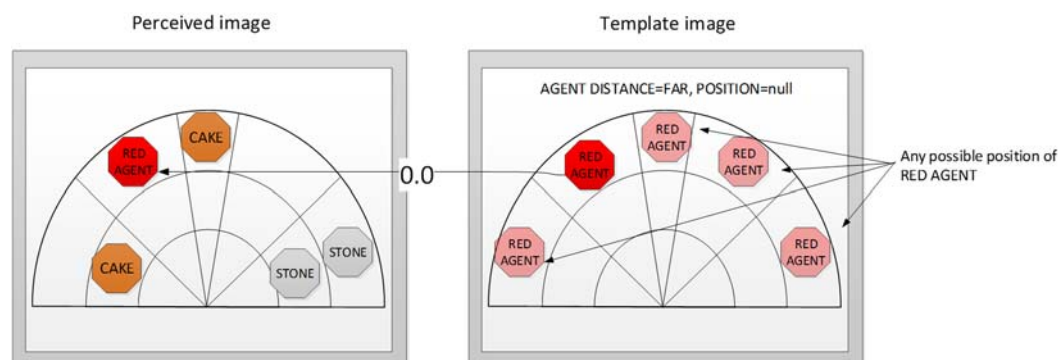


Figure 4.3: Image matching of a perfect match between object representatives of a perceived image and object representative of a generalized, template image, where the position of the agent is only given by the distance

like. For the object representative of the template image, only the distance $\langle \text{FAR} \rangle$ is set, but no position (position=null), $\langle \text{FAR}, \text{null} \rangle$. Therefore, the nearest position is chosen, which is $\langle \text{FAR}, \text{CENTER} \rangle$. The best position within the generalization matches the object representative of the perceived image. This form of generalization is used in the template images, to use one template image to match several specialized images, like the perceived image.

The goal of the matching is to get a match that is as high as possible by minimizing the *location error* of the object representatives in the image. To reach a minimum location error, the object representatives of the template image are sorted by generalization, starting with the least generalized object representatives first. It is because the more generalized the positions are, the easier it is to find a matching object representative in the source image. For completely specified object representatives, i.e. position and distance are defined, there is only one object representative in the perceived image, which could match. In Figure 4.4 and Figure 4.5, the search for the minimal location error is illustrated.

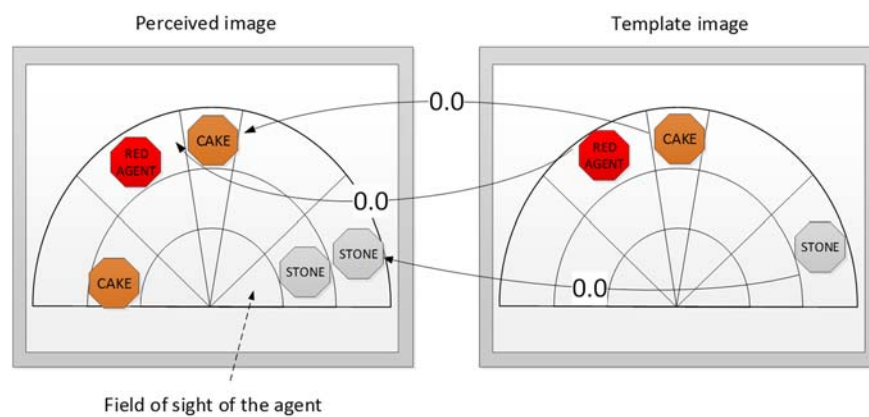


Figure 4.4: Image matching with perfect matches with Euclidian distances of 0.0 and therefore a match value of 1.0, between object representatives of a template image and object representatives of a perceived image

The process of comparison starts and all object representatives of the template image are compared to all object representatives of the source image, starting with the most specialized object representatives first. For instance, with the perceived image as a source image, it is as specialized as an image can be. In a perceived image like in Figure 4.4, all visible object representatives have defined positions. In the template image, only relevant object representatives are provided.

The location error could be measured by several means. For simplicity and because of the low granularity of the vision of the agent¹⁸, Euclidian distances will be used. Other types of distances would be more complicated to calculate, and it would probably not make much difference to the result. Therefore, between found pairs of object representatives, the Euclidian distance is calculated.

¹⁸ Low granularity: The vision consists only of twelve sectors ranging from three distances and four angles

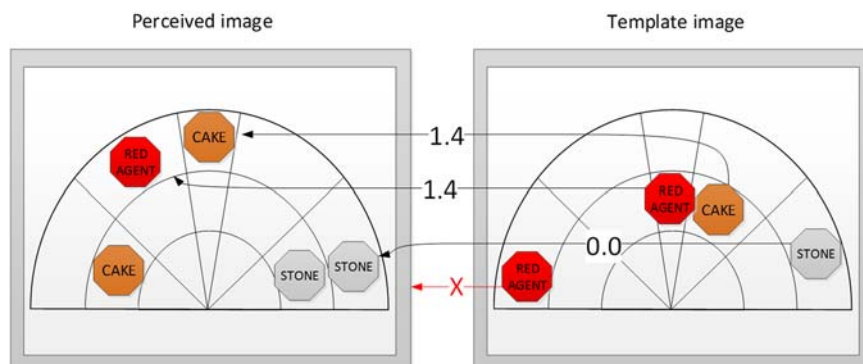


Figure 4.5: Image matching of partial matches between object representatives of a template image and object representatives of a perceived image

To be usable in the matching algorithm that ranges from 0 to 1, the distance has to be converted into a metric of similarity. In other case-based reasoning systems, different metrics are used [BRB07, p. 1844]. Due to the low granularity of the vision, it is assumed that small location errors among the object representatives would correspond to a high degree of similarity if a linear function would be used. Too many images would have similarity values above 0.8. Therefore, an exponential decay of similarity is assumed. It can be seen as a mean to favor images, with perfect matches, i.e. no location errors at all. Then the location error is converted to similarity with the exponential function

$$w_j = e^{-\frac{d_j}{c}} \quad (4-1)$$

In SiMA, the similarity is equal to the association weight. Therefore, w_j is the calculated association weight, d_j is the location error and c is a constant, which is dependent on how fast the matching shall decline if there is no perfect match of a match value=1.0. c can be chosen and optimized in experiments. For instance, in the case of a small c , the declination of a no perfect match is high and a c makes the significance of the location error is lowered.

Object representatives of the template image with no matching object representative at all in the source image has the similarity 0.0 and therefore receive the association weight 0.0. In Figure 4.4, a perfect match of a perceived image as a source image and a template image is shown, as every object representative of the template image is found in the same position in the perceived image. In this case, the perceived image can be seen as a specialization of the template image. The arrows in the figure represent the Euclidian distance between matching object representatives. Figure 4.5 shows a partial image match because the Euclidian distance of two of the object representatives is higher than 0.0 and one object representative does not have any match in the perceived image (represented by an X).

Finally, the association weights of all pairs are calculated. Then, the total image match is calculated, which is the average value of all association weights. The result is the similarity of the images.

4.2.2 Implementation of the Decision-Making Process

In Chapter 3.4.3, the concept of a technology-independent solution for a decision-making process was proposed that will enable experience-based decision-making in SiMA. In the following, the topic is the implementation of that process in a computer program.

Goals, Drive Wishes and Possible Goals

The concepts of drive wishes and possible goals were described in Chapter 3.4.2. The central point was that all of these concepts represent goals. Drive wishes are goals that the system shall try to reach. The method of achieving the system goals is provided by the possible goals that have been extracted from acts, the perceived image or also from drive wishes.

In the following, it will be argued why a specific goal data structure is necessary for the implementation. At the design of decision-making, it was clear that object representatives, images, acts, emotions and drive meshes are not sufficient, to create a multistep decision-making. Decision-making needs meta-information about the drive wishes or acts. The current data structures of Chapter 3.1 do not cover this aspect. Besides, to compare different types of goals with each other, they all have to be comparable, else apples are compared with pears.

Therefore, the implementation data structure of a goal is being introduced. A *goal* is the basic transport medium of data between functionalities. The goal is an abstract data structure as shown in Figure 4.6. Its sub-classes are then the drive wish and the possible goal. They share the common properties, which are defined for the abstract goal. The common properties of a goal can be derived from the properties of a drive mesh and the memorized drive mesh (see Chapter 2.3.4).

The following properties are derived from the drive mesh. The *goal name* is the word, with which a goal is called. The functional meaning will be mostly used for identifying what the goal is, e.g. to satisfy the drive HUNGER. The *goal object* is the actual object representative, of which the goal refers. It is the object representative that is in the focus of the goal. It can be a <CAKE>, which shall be eaten or a hostile agent <BODO>, which shall be avoided. The *goal type* is the kind of the goal, e.g. if it is a goal, which is based on drives or feelings. The *goal source* is the origin of a goal. Possible goal sources are drives, perception, and acts.

The following properties are introduced as additional properties and are derived from the requirements of the decision process. *Goal attributes* describe two things: the current status of a goal and results of analysis. They are the metadata of the goal. If a goal attribute is a product of analysis, it has different consequences for the further processing. For instance, if an analysis is performed and it concludes that the goal is not reachable from the current situation, it receives the goal attribute <GOAL_NOT_REACHABLE>. Any function that is evaluating the goal and reads this will strongly reduce the possibility that this goal is selected. The second usage is as status markers. For instance, if the status <HAS_PERFORMED_BASIC_ACT_ANALYSIS> is set, decision-making knows that this step of the processing has already been executed, and it can do something else.

To be able to compare different goal types with each other and different evaluations, a common “currency” is needed. It is defined as *importance* and it is the way, how feelings can be compared with

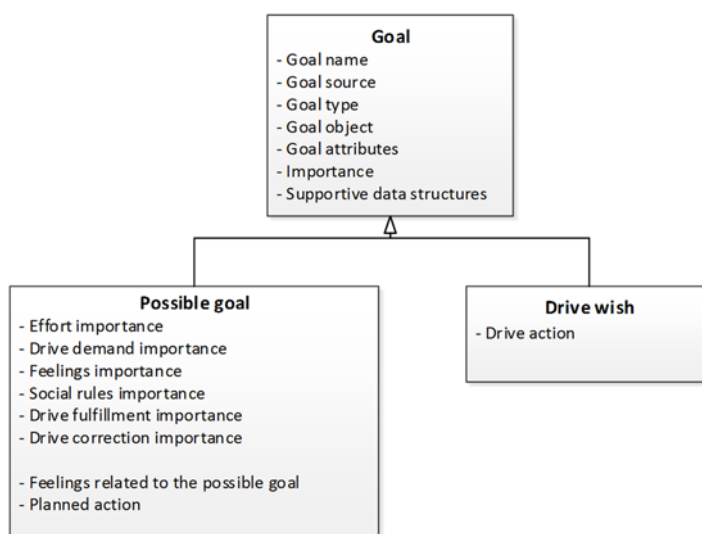


Figure 4.6: A goal with its properties and sub-classes

drive meshes. The attributes are the same for all goals and therefore, the goal is a container data structure of all collected information. In the following, the goal is specialized for its specific purposes.

Finally, the link to the source of the goal has to be created. It will be called the *supportive data structure* and can be any image or object representative, which supports the processing of a goal. In the case that the goal object is a <CAKE>, the supportive data structure is a part of the perceived image, which helps the agent to locate the <CAKE>.

The drive wish extends the goal by one property as can be seen in Figure 4.6. The *drive action* originates from the drive aim, which is only required for drive wishes. The drive action is usually the mean for fulfilling the goal. The quota of affect of the drive mesh is here equal to the importance of the goal.

The possible goal originates from the perception track of the SiMA model and represents the options in SiMA. They represent all possible states, which can be proposed as solutions to goals. Different to the drive wish, it is needed to use multiple types of importance: Effort importance, drive demand importance, feelings importance, social rules importance, drive fulfillment importance and drive correction importance.

In the secondary process, according to Chapter 2.3.1, the reality principle rules. Therefore, the possibility to reach a certain goal in the environment is evaluated. It does not help that the quota of affect of a memorized drive mesh is very high if the possible goal is almost impossible to reach. For that reason, importance can be differentiated into several subcategories, which have been proven useful in the implementation.

The most obvious part is the *drive demand importance*, which represents how important this possible goal is about the corresponding drive wish. The contribution from the reality principle is that the *effort* usually decreases the total importance. The impact of feelings through *feelings importance* and the impact of social rules through *social rules importance* may increase or decrease the total importance.

The *drive fulfillment importance* is the actual evaluation of the drive wish on its corresponding possible goal regarding the drive aim and object. For instance, if a drive wish proposes the agent to execute the action to <SHARE> something, the goal object is a <CAKE>. The importance would be 0.5, the agent will prefer a possible goal, which gives the agent the option to execute the action <SHARE> instead of <EAT> the <CAKE>. If the possible goal brings the satisfaction equal to 0.8, the importance of the drive fulfillment is still 0.5. Although the potential would be there, it does not help if the importance of the drive wish is not high enough.

Finally, the *drive correction importance* is defined as an implementation specific detail. It is used to prioritize among the drives. If there are two drives, which are equally strong, one drive still has to be more important than the other. It is the case that the drives are in <HUNGER> and <RELAX>. To be able to eat something, the agent must not be exhausted. Therefore, it makes sense that the drive <RELAX> is preferred compared to <HUNGER> if all other importance values are equal.

If there are feelings associated with a template image, they are added to the possible goal as they influence the decision-making process. For instance, if the agent has to select between two similar drive goals, the feeling about the external circumstances may decide which possible goal is selected. If there is a feeling of <JOY> associated with a possible goal, this possible goal may be evaluated as easier to achieve than a possible goal with the associated feeling <ANGER>.

Finally, a planned action can be associated with the goal in the planning later on in the decision-making.

States of Possible Goals in the Decision Process

In Figure 3.22, a high-level process model of decision-making was shown. For a possible goal, goal attributes were defined. They completely determine the state of a goal. At several places in decision-making, this state is changed. A solution is only to modify the goal attributes. The challenge is to find a concept that allows the system always to get into a defined state after each cognitive cycle. Further, according to the requirements on decision-making from Chapter 3.4.1, there are many different types of possible goals, each of them with its requirements on how to be processed.

For that purpose, a state machine ¹⁹for each goal type is proper to use, to manage the complexity of the system. Such a concept that is adapted for the decision-making in SiMA is *teleo-reactive planning* [Nil94, pp. 139-158]. Also, other cognitive architectures like LIDA rely on this idea (see Appendix A). It assures that the agent always can act. For instance, if three different internal actions are necessary to produce an external action, all of them have to be executed first, to be able to execute the external action. As soon as one of the internal actions cannot be executed, the state of the possible goal remains in the last defined state.

According to teleo-reactive planning, there always has to be a previous step, a fallback to which a possible goal can return to if an action fails. Goal attributes are used to keep track of the state, to

¹⁹ Every state of the data is unique in SiMA because perceived image is merged with the drive state and the history of the system. However, the process that executes the functional modules of the system is a state machine. Here, any new state not covered would cause problems for the system to handle.

determine the next action of the system for a possible particular goal. The challenge is to assure that no combination of goal attributes leads to a deadlock or freeze of the system. Therefore, for each possible goal type, a state machine is created, where each of the goals attributes always has to mean something to the agent. If internal actions fail, it will be tried again until another planned goal is selected or goal attributes are exchanged and therefore trigger other actions.

The detailed decision process with the change of the states of a possible goal is illustrated in Figure 4.7 and it continues in Figure 4.8. A possible goal is illustrated as a gray box with goal attributes (Attributes), importance (Importance) and action (Action). The colored boxes represent the sub-functionality of the functional modules of SiMA (as seen in Figure 3.20). The text in those boxes explains what is done with the possible goal and the corresponding module number e.g. F52. In each functionality, the state of the possible goal is modified. The chronological order of the modification of the possible goals is showed with numbers on the side of the possible goal. Added data is highlighted in red color to see the changes in each function. Deleted attributes are highlighted as red and stroke through by a line. By doing that, the old and the new attributes are visible. In the following, the state of a possible goal will be described through decision-making.

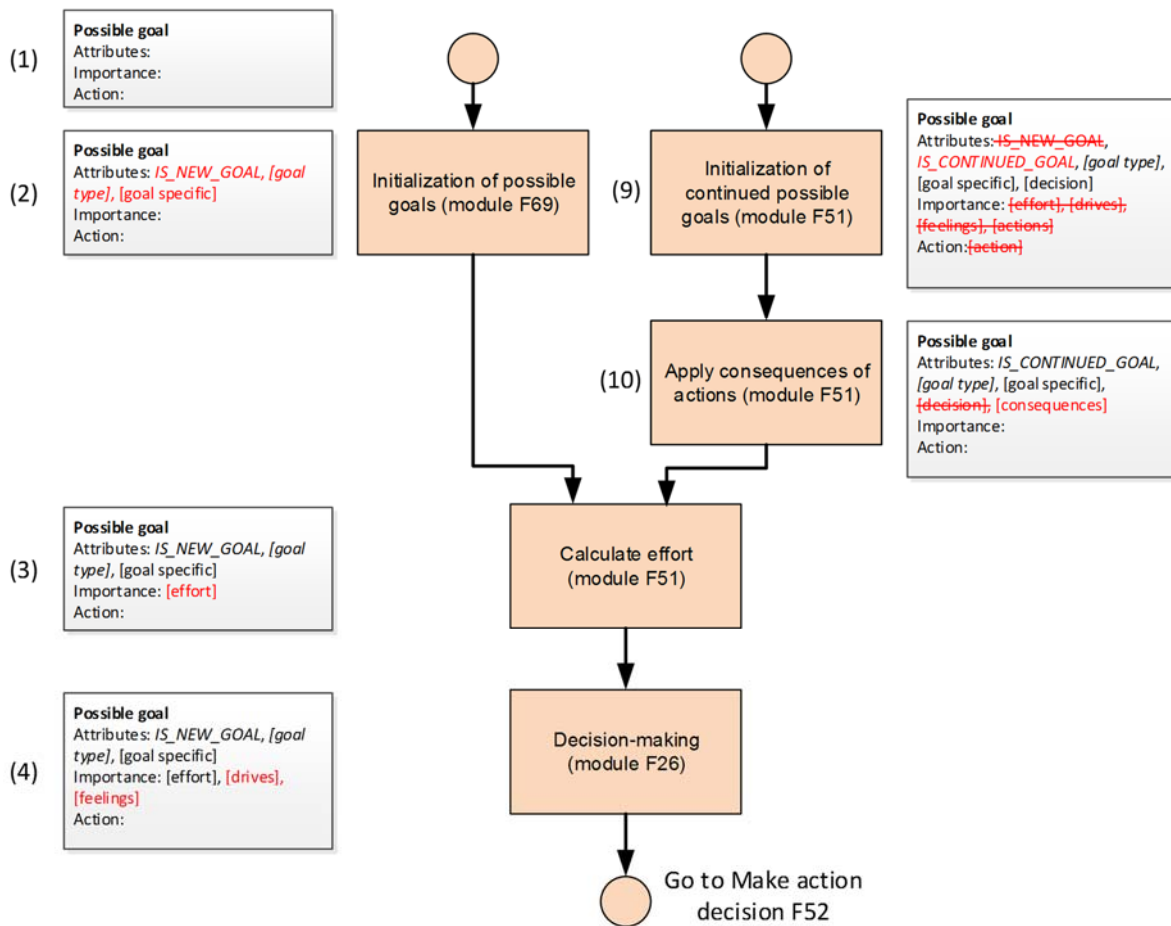


Figure 4.7: Part 1 of the detailed view of the decision process from the perspective of possible goals

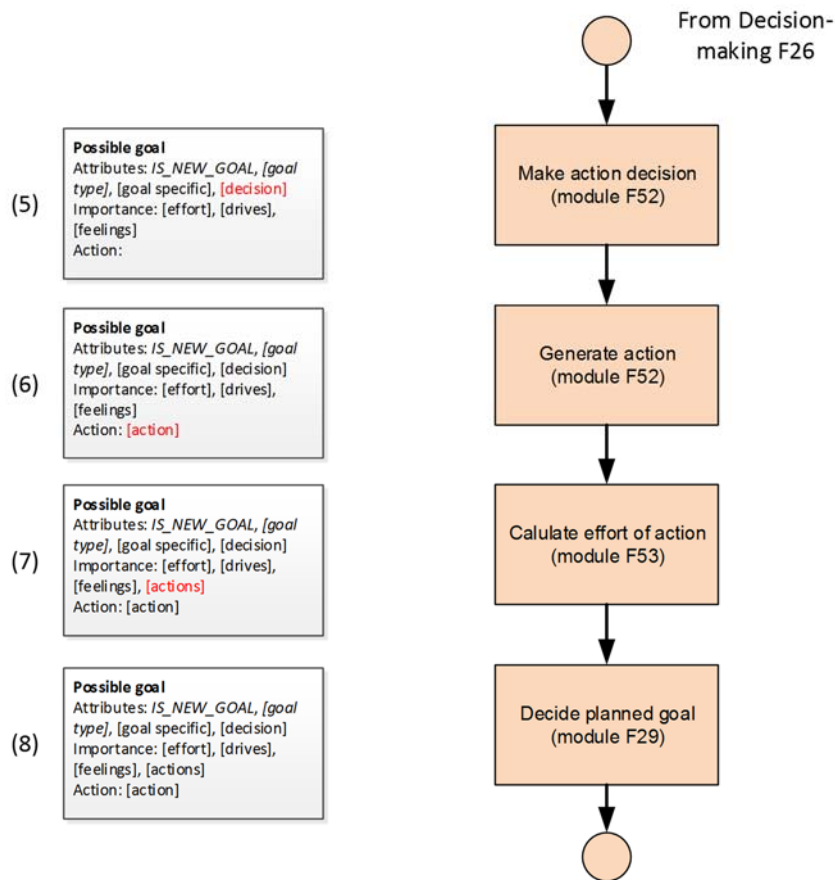


Figure 4.8: Part 2 of the detailed view of the decision process from the perspective of possible goals

The process starts with an unprocessed, possible goal (1, Figure 4.7) in that has been generated from one of the three sources: perception, acts or drives as described in Chapter 3.4.3. For each goal attribute of each possible goal, there has to happen something in the system, or it has to influence the evaluation of the possible goals. The purpose is that the goal attributes conserves the state of a goal over at least one cognitive cycle. Therefore, goal attributes have two roles for a goal: first to define the status of the possible goal regarding what to do next and second, to modify the importance by increasing or decreasing effort.

A typical example of an effort-reducing goal attribute is the attribute `<IS_PLAN_GOAL>`, which means that if two possible goals are competing and all other evaluations are equal, the possible goal that was the planned goal of the previous cognitive cycle will be favored. The effect of this goal attribute is a personality parameter. However, it is a good idea to prefer to pursue an already started planned goal instead of swapping to a new possible goal. Therefore, this goal attribute lowers the effort of reaching the planned goal instead of starting an analysis of another concurring possible goal.

After basic initialization (2 in Figure 4.7), it receives its first goal attributes. Based on them, the effort is calculated in the reality check (3 in Figure 4.7). In decision-making (4 in Figure 4.7), feelings and motivations as drive wishes are used to filter the possible goals. Importance is added from the drive

wishes as well as the effect of feelings. Based on the goal specific attributes that were added in the initialization step, a decision is made about what is needed to do next (5 in Figure 4.8). It is a decision about the next internal step. It tells the system what is needed to be done with the possible goal, to proceed with it. An example of such an attribute is the <NEED_TO_MOVE_TO_OBJECT>. The system now knows that an action shall be found to move to the desired object representative.

Based on all attributes, the appropriate action is generated (6 in Figure 4.8) and evaluated (7 in Figure 4.8). Finally, one of the possible goals is selected as a planned goal in (8 in Figure 4.8). Here, all importances that were added to the possible goals are summarized, and the highest importance wins.

As seen in Figure 3.22, all possible goals, which continued from decision-making are stored in the working memory. In the next cognitive cycle, they activate once again (9 in Figure 4.7). At this point, obsolete goal attributes from the previous cognitive cycle are deleted. For instance, if the agent executed an external action <MOVE_FORWARD> and moved forward, the consequence is that the goal attributes for focus of attention do no longer apply. The reason is that the object representatives that were focused may no longer be in that position anymore, and therefore, the environmental image has to be renewed (see Chapter 3.4.5). In Figure 4.7, this is shown by stroke through text of the goal attribute. The attribute <NEW_GOAL> has been replaced with <IS_CONTINUED_GOAL>. It corresponds to step (6) in Figure 3.22.

These goals are merged with new incoming possible goals. They still have their goal attributes from the previous step, i.e. the state has been conserved. The goal attributes are used to either trigger internal actions like FOCUS_ON, to filter important things from perception or to add an evaluation of an executed external action like <MOVE_FORWARD> (10 in Figure 4.7). Finally, the system continues in the same way as in the previous cycle (3 in Figure 4.7).

Decision Paths

While the decision process can be seen as a process that processes data in one cognitive cycle, a *decision path* can be defined as the path of internal states and internal actions necessary to generate an external action. In this work, as the first proof of concept, the following decision paths are predefined for the agent, one for each type of possible goal. They originate from drive wishes, the perceived image and acts.

Each decision path represents a state machine, where decisions are made according to the applied consequences of previous decisions. Every time a consequence is set, the attributes are updated. Only through consequences, attributes can be deleted, making the goal to continue at an earlier stage. In the tables, the columns describe the modification of attributes for each step. The attributes are modified in four process steps, which are *init*, *consequence*, *decision* and *action*. There are the following correspondences to process steps in Figure 4.7 and Figure 4.8: *init* corresponds to (2) and (9); *consequence* corresponds to with (10); *decision* corresponds to (5); and *action* corresponds to (6). The purpose of possible goals, which are created directly from the drive wishes without any correspondence to the external world is to either trigger images through fantasy or to search the environment for certain object representatives. The decision path is shown in detail in Table 4-2.

Goal type: Drive Wish				
Step	Init	Consequence	Decision	Action
1	Add <NEW GOAL> and <DRIVE GOAL>	-	Add <NEED TO SEND TO PHANTASY>	Execute internal action <SEND TO PHANTASY>
2	Add <CONTINUED GOAL> or <PLANNED GOAL>	Add <PHANTASY ACTIVE FINISHED>	Add <NEED TO FOCUS ON SEARCH AREA>	Execute internal action <FOCUS ON SEARCH AREA>
3	-	Add <FOCUSED SEARCH AREA FINISHED>	Add <NEED TO SEARCH>	Execute external action <SEARCH>
4	-	Delete <FOCUSED SEARCH AREA FINISHED>	Goto step (2)	

Table 4-2: Attributes used for the decision path for drive wishes

It is accomplished by defining a high effort for possible goals that origin from drive wishes and if there is any act available or the desired object representative is present in perception, those possible goals will always be preferred.

For goals that origins from perception like in Table 4-1, the following steps are done: first, the object representative is focused on, then the area in front of it and then the action is executed. It is noticeable that the attribute <NEED TO MOVE TO OBJECT> or <PERFORM STATIC ACTION> does not change from step (2) to (3). This is because the action executed is not only dependent on the decision, but also on the consequences of previous actions. The action to move can first be set if the area of movement has been focused on first, else the agent may run into an obstacle.

Goal type: Perception				
Step	Init	Consequence	Decision	Action
1	Add <NEW GOAL>, <PERCEPTION GOAL>	-	Add <NEED TO FOCUS ON OBJECT>	Execute internal action <FOCUS ON OBJECT>
2	Add <CONTINUED GOAL> or <PLANNED GOAL>	Add <FOCUS ON OBJECT FINISHED>	Add <NEED TO MOVE TO OBJECT> or <PERFORM STATIC ACTION>	Execute internal action <FOCUS ON PATH AREA>
3	-	Add <FOCUS ON AREA FINISHED>	Add <NEED TO MOVE TO OBJECT> or <PERFORM STATIC ACTION>	Execute external action <MOVE> or <MOVE FORWARD> or <EAT> or ...
4	-	Delete <FOCUS ON OBJECT FINISHED> and <FOCUS ON AREA FINISHED>	Goto step (1)	

Table 4-1: Attributes used for the decision path for possible goals from perception

Goal type: Acts				
Step	Init	Consequence	Decision	Action
1	Add <NEW GOAL>, <ACT GOAL>	-	Add <NEED TO SEND TO PHANTASY>	Execute internal action <SEND TO PHANTASY>
2	Add <CONTINUED GOAL> or <PLANNED GOAL>	Add <PHANTASY ACTIVE FINISHED>	Add <NEED TO ANALYZE ACT>	Execute internal action <EXECUTE BASIC ACT ANALYSIS>
3	-	Add <BASIC ACT ANALYSIS FINISHED>	Add <NEED TO MOVE TO OBJECT> or <PERFORM STATIC ACTION>	Execute internal action <FOCUS ON OBJECT>
4	-	Add <FOCUS ON OBJECT FINISHED>	Add <NEED TO MOVE TO OBJECT> or <PERFORM STATIC ACTION>	Execute internal action <FOCUS ON PATH AREA>
5		Add <FOCUS ON AREA FINISHED>	Add <NEED TO EXECUTE ACT ACTION>	Execute external action <MOVE> or <MOVE FORWARD> or <EAT> or ...
6		Delete <FOCUS ON OBJECT FINISHED> and <FOCUS ON AREA FINISHED>	Goto step (2)	

Table 4-3: Attributes used for the decision path for acts

In the last processing type in Table 4-3, the fulfillment of planned goals from acts, the following steps are made: first, the act has to be analyzed, to evaluate how well it fits into the current situation and how to proceed to reach the goal. Therefore, a part of the act is sent back to the primary process through fantasy, to activate more relevant images. Then the act is analyzed. The result of the analysis is a recommended action and similar to goals from perception. First, the object of interest is focused on and then the area in front of the agent, to avoid obstacles. Then, the action is performed.

Forward Chaining and Backward Chaining

This type of decision-making is a typical forward-chaining mechanism, where the agent executes one step at the time to get closer to its goal. However, with some extension, this structure would also allow backward-chaining and mean-ends analysis to be performed. If the agent can start with the end state of the planned goal and reason about what attributes will lead to the end attributes, it would be achieved. For instance, if the agent wants to move forward, it has to find out that to go forward, it first has to focus on the movement area, because the movement area focus is a precondition for moving forward.

At the current state of this work, the processing of possible goals is composed of a set of state machines. Only through the recommended actions in acts, some variety of actions can be defined. Due to the limitation of the content of this work, is not foreseen to develop further flexible management of attributes. However, if they were defined as knowledge and learning would be implemented, many possibilities would be opened for customizing the decision process for each act.

The path is still very far to reach human-like intelligence. Depending on the input, SiMA as a system will always enter new states. In every cognitive cycle, new input will influence the decisions of the system. In the preliminary work, decision-making consisted of one single process, which always produced the same behavior for a certain input. The behavior of the system was a state machine. The contribution of this work is that it allows multiple processes to be executed depending on the inputs. Further, the behaviors of the system are located in the acts. If the acts are modified, i.e. new experiences are recorded, then the system behavior would also change. In a future learning system, the behaviors of the system would be depending on its experiences and personal history, which is closer to human-like intelligence than a single process state machine.

5. Simulation and Results

“I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted.”

[Alan Turing, Computing machinery and intelligence]

In Chapter 1.3, research questions were raised for four different areas of the cognitive model, in order to realize the usage of experiences in decision-making. The research areas are data structures, memory retrieval, sequence recognition and decision-making. According to the methodology of Chapter 1.5, the partitioned iteration, each research question defines a partition. Design concepts were defined as possible answers to the research questions in Chapter 3 and important details of the implementation were highlighted in Chapter 4. For even more detail to the software design, please refer to Appendix B. In the following, the proposed model and its implementation will be verified against the research questions with a use case. Although this use case may seem very simple, it clearly demonstrates the new functionality of the system. A specific problem is solved. Here, it can be mentioned that simple use cases are used for testing state of the art architectures as well. In SiMA, features regarding decision-making and act recognition are in focus. In order to demonstrate the power of this implementation, research results of more comprehensive use cases will be described as well.

5.1 Test Environment

In Chapter 1.3, a use case was briefly described that will be the foundation of the following tests. As the SiMA architecture is strongly focused on the primary process, drives and defense mechanisms, Use case 1 [BGSW13, p. 4] was developed by the SiMA research team in close cooperation with psychoanalysts. In Chapter 4.1, the implementation environment in a simulator was described.

The basic setup of Use case 1 can be seen in Figure 5.1. The agent <ADAM> sees a food source <CAKE> and another agent <BODO>. From here, depending on the drive state and the configuration of the defense mechanisms, the agent can either eat the <CAKE>, consider the influence of the other agent by sharing it by dividing it and giving a part to <BODO> or just to beat <BODO>.

Use case 1 [BGSW13, p. 4] focuses on the exchange of drive aims by providing several possibilities to satisfy a certain drive wish and how emotions are used to modulate the selection. This work focuses on spatial constellations of objects in the world and the extraction the possibilities on how to satisfy

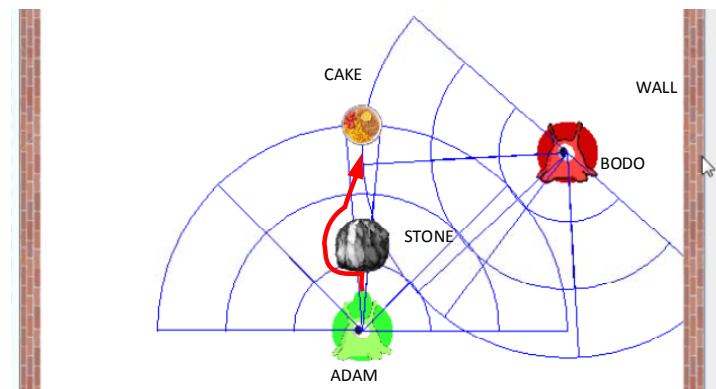


Figure 5.1: Screenshot of the original positions of the objects in the world of the use case Use case 1 Obstacle Modification

certain drive wishes. Therefore, Use case 1 is extended with relevant spatial objects like obstacles. In the following test case, obstacles are represented by the object <STONE> that is put in the middle of the path between the agent <ADAM> and its food source <CAKE>.

This use case will be divided into several test cases. Each test case demonstrates a different sub-aspect of the use case. They will show one implemented functionality each. As a whole, all implemented and demonstrated functionality contributes to the solving of the task of the use case.

Task

In the initial setting, the blood sugar of the agent <ADAM> is low and therefore the drive to consume the <CAKE> is present and growing. In the following test cases, <ADAM> has two tasks. First, <ADAM> will go around the <STONE>, to get to the <CAKE> that is desired. Second, the agent has to deal with <BODO> according to its personality parameters and available experiences. As soon as the <STONE> has been passed and the <CAKE> is in sight, <BODO> is spotted. Depending on the internal state of <ADAM>, the interaction with <BODO> is determined. This use case forces the agent <ADAM> to use stored experiences that tell the agent how to handle a given situation and which options are available.

Notice that in ARSi11, no experiences existed at all and the only option the agent had was to go straight ahead through the <STONE> using brute force. Regarding interaction with <BODO>, there were only default actions available independent of the situation. This work will show, how the system reacts on obstacles by checking for similar situations in its experiences.

Initialization and Personality Parameters

In each test case, the initial and personal parameters are customizable. However, as a default, the defense mechanisms are disabled, to demonstrate the simplest case, just to eat the <CAKE>. In every test case, <ADAM> starts with low blood sugar, in order to trigger the drive to eat something.

Available Acts

Because learning is not a topic in this implementation, predefined acts are available. Each act represents an earlier experienced situation, and how <ADAM> handled it. For this test setup, all acts have been manually added to the system. In some special cases, the set of acts is modified, to be able to demonstrate the requested functionality.


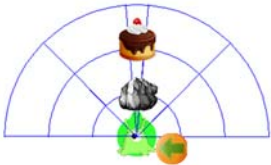
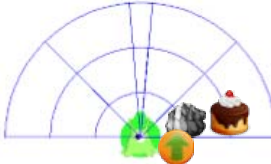
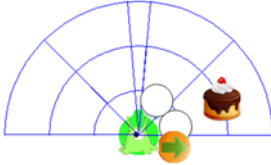

Image name	Objects and their positions	Visualization of the image	Role
A01_GOAROUNDSTONE_L01	CAKE (null, null)		Intention
A01_GOAROUNDSTONE_L01_I01	CAKE (null, CENTER); STONE (NEAR, CENTER) Action: TURN_LEFT		Event 1
A01_GOAROUNDSTONE_L01_I02	CAKE (null, RIGHT); STONE (null, RIGHT) Action: MOVE_FORWARD		Event 2
A01_GOAROUNDSTONE_L01_I03	CAKE (null, RIGHT); EMPTY_SPACE (NEAR, RIGHT); EMPTY_SPACE (NEAR, MIDDLE_RIGHT) Action: TURN_LEFT	 ○ Explicit empty space	Event 3
A01_GOAROUNDSTONE_L01_I04	CAKE (null, CENTER);		Event 4

Table 5-1: Description of the act <A01_GOAROUNDSTONE>

In Table 5-1, the predefined act, which is of primary importance for the following test cases is described in detail. First, the act name is shown. In the rows, the details of the template images of which the act consists, are shown. The first column with the title Image Name describes the label of the template image. Then follows objects and their positions of all spatial objects and their positions in the act, e.g. <CAKE(null, RIGHT)>, which means that the object representative <CAKE> has a defined position to the <RIGHT> in the image, but no defined position of the distance <null>. It represents a generalized template image. Some template images also have an action associated. It

represents the action that the agent <ADAM> executed at the moment the template image was recorded.

The objects of the image are visualized in the column with the title Visualization of the Image. Finally, the role of the image in the act is defined in the title Role. There are two roles: intention and event. The intention originates from the concept of Chapter 3.3.2 and the event is any image, which can represent a moment or an expectation, also from Chapter 3.3.2.

The act <A01_GOAROUNDSTONE> describes the experience of how <ADAM> acted once, to get around the <STONE> to reach the <CAKE>. In the intention <A01_GOAROUNDSTONE_L01>, only a <CAKE> is provided without a position. It means that this image will match any perceived image, where a <CAKE> is present at any position (see the image matching algorithm in Chapter 4.2.1). The images <A01_GOAROUNDSTONE_L01_I01>, <...I02>, <...I03> and <A01_GOAROUNDSTONE_L01_I04> represent the events of the act. As soon as an event of the act is matched, it becomes the moment. The next event is then the expectation, and it is expected to occur if the experienced action is executed. The experienced action is labeled <Action> in the table. <ADAM> then handles himself from event to event within an act, to reach the goal of the intention. It is worth to mention that because only the images of the act are generalizations, only the necessary object representatives are included. To represent an empty space, the object representative <EMPTY_SPACE> has to be explicitly added.

This act, as well as most other acts in the long-term memory, does not represent any particular experience, but a general way of solving a problem. Therefore, the act mentioned in Table 5-1 applies as an action plan for every <STONE> that shall be gone around. Due to limitations of the scope, the other acts of <ADAM> are only explained if they play a role in the test cases.

In the following sub-chapters, each test case will have the naming Test case [Number] a number that is related to the task from Chapter 1.4, which has been solved and implemented. The main research question has the number 1. The research questions, which are derived from the main research question have the numbers 2...4.

5.2 Verification of Data Structure Functionality

The implementation of the data structures necessary for forming acts will be verified through test cases that have been derived from Use case 1 Obstacle Modification. The test cases in this chapter will prove that the following task has been solved: *Task 2: Extend and modify current data structures to include situations, sequences of situations and goals (Research question 2.1 and 2.2).*

5.2.1 Test case 2.1: Internal Representation of the Primary Process

The objective of the first test case is to show how the implementation of the data structures of a physical object and a perceived image from Chapter 3.1.1 have been realized. It is expected to see a visualization of the internal representation of <ADAM>.

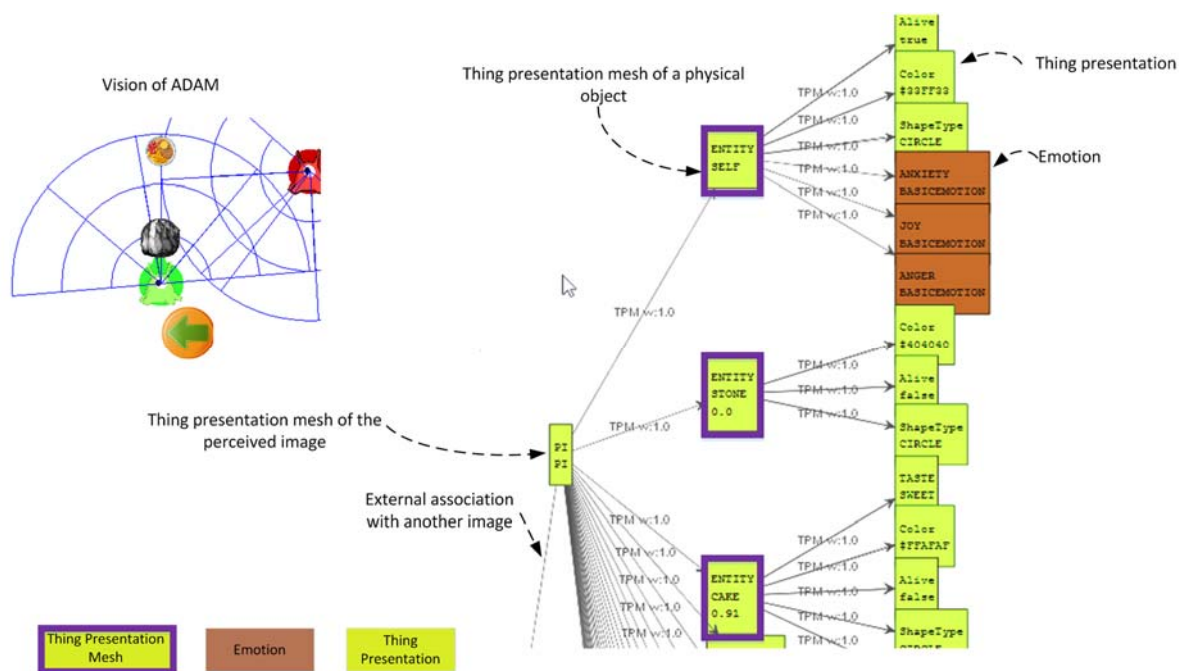


Figure 5.2: Screenshot of the situation (left), a physical object and the perceived image (middle) and another loaded image of the act “A01_GOAROUND<STONE>” (right)

The test-setup is the normal use case setup from Chapter 5.1. The simulator is executed until <ADAM> stands in front of the <STONE>. The act <A01_GOAROUNDSTONE> shall be activated. Through the inspector in module Memory Traces for Perception (F46) of Figure 3.8, the output data structure of the module is analyzed.

In Figure 5.2, it is visible how a thing presentation mesh represents the smallest perceivable object as well as the perceived image. For each perceived object representative, several thing presentations are associated. They describe different properties of the object representative. Only in a particular case, where the object representation is representing <ADAM> himself in the image <ENTITY SELF>, emotions are associated with him.

Thing presentations define object representatives like the <CAKE> and the <STONE>. On the next level, the perceived image is also represented by a thing presentation mesh. The perceived object representatives define it. Here, the scalability of the images is given, as the thing presentation mesh could be used to define another image as well. Not completely shown in the figure is the association to the template image <A01_GOAROUNDSTONE_L01_I02>. It has the same structure as the perceived image.

5.2.2 Test case 2.2: Internal Representation of the Secondary Process

It will be continued with data structures. It shall now be shown how the data structures that define an act from Chapter 3.4.2 have been implemented. A visualization of the internal representation of <ADAM> demonstrates it. The same situation is used as in Test case 2.1. In the module Reality Check

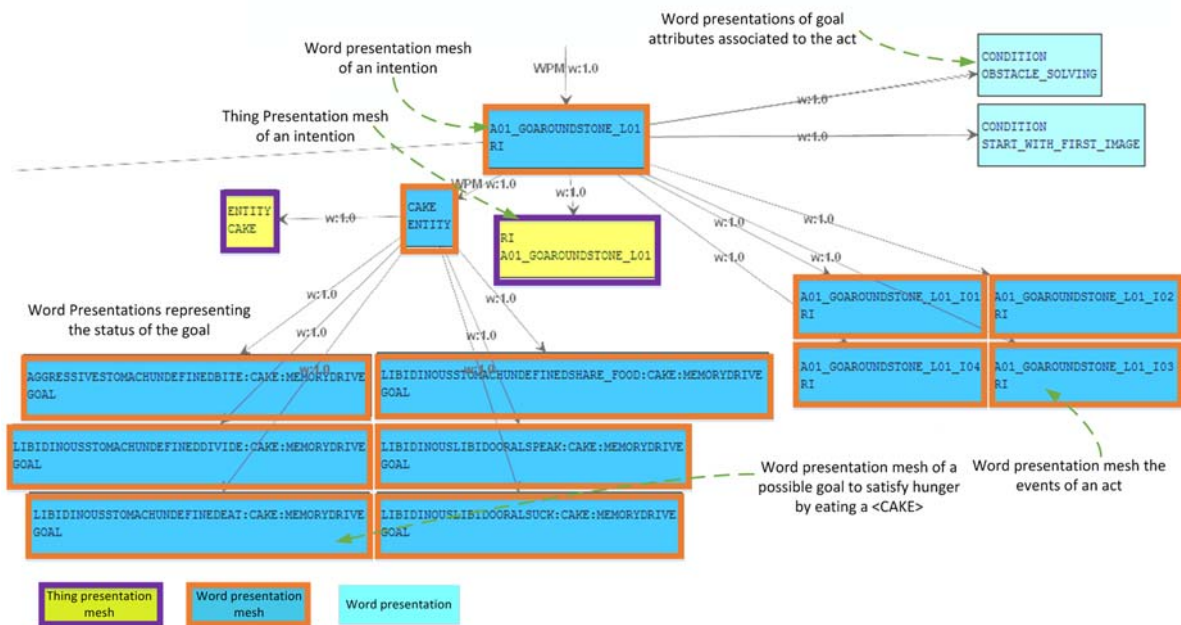


Figure 5.3: The act <A01_GOAROUNDSTONE> in the secondary process with intention, goal attributes, events and associated possible goals based on drive representations

wish fulfillment (F51) of Figure 3.20, the inspector shall show the structure of the act and a possible goal.

In Figure 5.3, the output of the module Transformation to secondary process (F21) of Figure 3.20 is shown. It represents the act <A01_GOAROUNDSTONE> that has been activated by the template images in Figure 5.2. At this stage, the possible goals of Chapter 3.4.2 have not yet been extracted yet and are still only representations of the drive state. In the figure, six possible goals are shown in the lower left. The prototypes of possible goals are associated to word presentation mesh of the <CAKE> in the intention of the act <A01_GOAROUNDSTONE>. For some word presentation meshes, the correlating thing presentation meshes are visible in the figure (green color). Attached to the act are some goal attributes. They will be transferred to each possible goal. There, they define the status of it and how the act shall be handled. For instance, the goal attribute <START_WITH_FIRST_IMAGE> will tell the act analyzing function, only to give a high importance of the act, if the first event of the act is the moment. Custom goal attributes can be defined.

The tests of the data structures show that it is purposeful to represent images and acts with the data structures thing presentation mesh and word presentation mesh. These data structures have to possess maximal flexibility, to make the system scalable. However, the drawback of such general data structures is the lowered performance of the system.

5.3 Verification of Psychic Spreading Activation

The implementation of the concept Psychic Spreading Activation will be verified through test cases that have been derived from Use case 1 Obstacle Modification. The test cases in this chapter will prove that the following task has been solved: *Task 3: Introduce memory retrieval of situations based on other situations or the perception (Research question 3).*

5.3.1 Test case 3.1: Image Matching

The base for psychic spreading activation is the comparison of a source image with template images of the memory. Therefore, implementation of image matching of Chapter 4.2.1 will be verified. Template images of the act <A01_GOAROUNDSTONE> are analyzed in a given situation during the execution of the use case. The perceived image match is manually calculated and compared to the results of the simulator, in order verify the values, for certain images. The inspector of the module Memory Traces for Perception (F46) of Figure 3.8 is used. In this situation, inspector values from the activated images of the act are used. In Figure 5.4, the first image

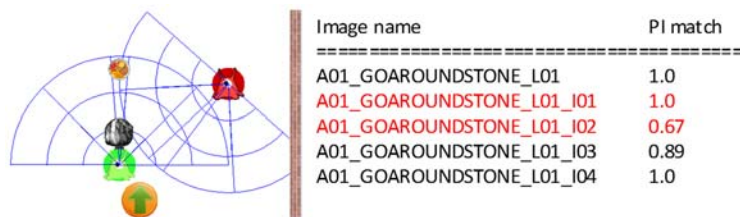


Figure 5.4: Verification of image matching

<A01_GOAROUNDSTONE_L01_I01> (see Table 5-1) has a perfect match, while the following template image <A01_GOAROUNDSTONE_L01_I02> only has a partial match.

The matching value can be calculated. The perceived image match value of <A01_GOAROUNDSTONE_L01_I02> is 0.67. The value is calculated from the position of <STONE> in the perceived image, which is <NEAR, CENTER> = <1, 0> and the <STONE> in <A01_GOAROUNDSTONE_L01_I02>, which is <null, RIGHT> = <null, 2>. As the best match is searched for in the image matching, the position of the <STONE> in <A01_GOAROUNDSTONE_L01_I02> is specialized to <1, 2>. The Euclidian distance between the two coordinates is 2.0. The same is applied to <CAKE>. The equation (4-1) is applied to a declining value $c=5$ (can be arbitrary selected). The perceived image match is then 0.67 for both <CAKE> and <STONE>. The average of these two matches gives the result 0.67. It is the same result as in Figure 5.4.

In this test case, not only the calculation of the image match based on distances between objects in the images was done, but also a specialization of generalized template images to the perceived image was demonstrated.

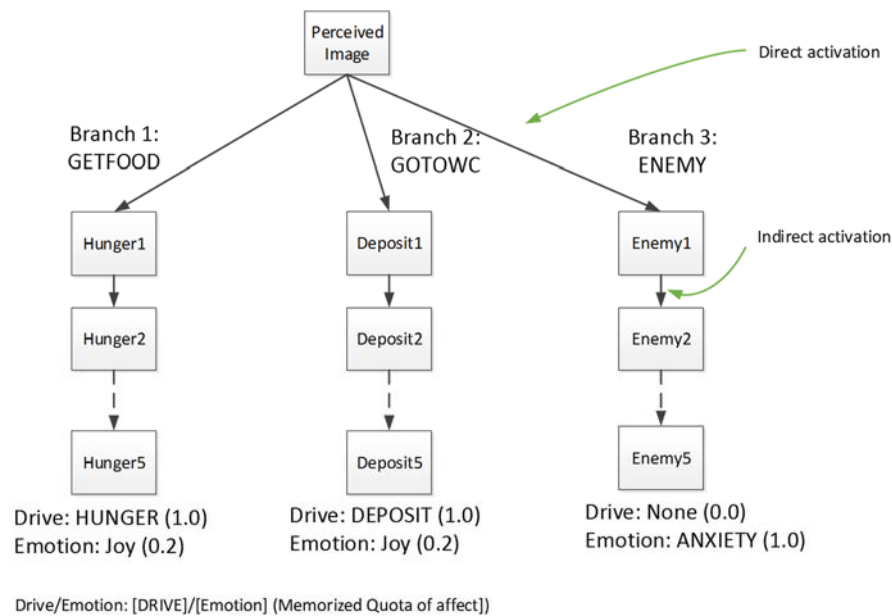


Figure 5.5: Memory content of a test setup with three branches

5.3.2 Test case 3.2: Activation of Relevant Memories

It shall be shown that only relevant images from a great pool of images are activated based on the inputs from a perceived image, the drive state, and emotions in the concept of psychic spreading activation of Chapter 3.2.4. Different to all other test cases, this test case has been implemented outside of the SiMASin World. It has been implemented in a simulator created in Microsoft Excel. The reason is to get a controlled environment, where all input parameters can be customized. It is not possible in the current SiMASin World simulator because there are strong dependencies between implemented modules. Another reason is that a specific setup of images will be used that does not have a direct correspondence to the images in the use case of the SiMASin world.

The following test setup is an extended version of the test setup in [WSG+13, p. 6674]. The purpose is to compare two different drives with each other to analyze how similarity and importance influence activation of images related to each of the drive. Analogous, more drives could be compared if needed. Then, the impact of emotions shall be demonstrated, where external stimuli receives the attention of the agent through activation of related images. Therefore, to compare the different input settings, three images were defined as a memory base, of which each of them corresponds to a different situation and drive state. Only these three images are used for matching with perception and drives. Each of the images is then associated with additionally four images. They can only be activated through indirect activations. Together, the image that can be matched with the inputs, including its additional images is called a *branch*. The first image will be called *matching image*.

The purpose of the test will show the following: The more psychic intensity can be allocated by each of the three matching images, the more of the additional images can be indirectly activated. Therefore,

the number of activated images in a branch is a measure of how relevant to the current situation its matching image is.

The setup is illustrated in Figure 5.5. The first branch <GETFOOD> (Image Hunger1, Hunger2 ... Hunger5) would fulfill the drive <HUNGER>, as it helps the agent to find food. The second branch <GOTOWC> (Image <Deposit1>, <Deposit2> ... <Deposit5>) fulfills the drive <DEPOSIT> and helps to find a place with the agent can excrement. The third branch <ENEMY> (Image Enemy1, Enemy2 ... Enemy5) shall help the agent to react to external stimuli in the form of an enemy agent. It does not fulfill any drive. In the branches, the configuration from Table 5-2 for the matching images is used. For instance, the matching image of the branch GETFOOD has the possibility to satisfy the drive for <HUNGER> with the quota of affect=1.0. The emotion <JOY> has the intensity=0.2. It is assumed that memorized drives and memorized emotions are equally weighted.

Branch	Quota of affect for memorized drives and intensity of emotions			
	Drive HUNGER	Drive DEPOSIT	Emotion JOY	Emotion ANXIETY
GETFOOD (Hunger1)	1.0	0.0	0.2	0.0
GOTOWC (Deposit1)	0.0	1.0	0.2	0.0
ENEMY (Enemy1)	0.0	0.0	0.0	1.0

Table 5-2: Quota of affect for drives and intensity of emotions for the matching images of each branch

Additionally, the following static settings are used in all tests:

- Each stored image consumes 0.2 units of psychic intensity
- The input of psychic intensity through the perceived image is 1.4, i.e. max. 7 images can be activated with a consumption value of 0.2
- Drive state:
 - Drive <HUNGER> with quota of affect=1.0
 - Drive <DEPOSIT> with quota of affect=0.3

In three tests, the match with the perceived image will be varied to determine the role of drives, emotions, and perceived image match.

In Table 5-3, the results of the tests are presented. Test 1 shows that all five images of the branch GETFOOD in Figure 5.5 are activated because both the perceived image match is high as well as the memorized quota of affect for the drive <HUNGER>. It also shows that images without any association to the perceived image cannot be activated. It is the case of the branch ENEMY, although the intensity of the feeling is high. On the other side, there is no reason to activate non-related images.

In Test 2, the perceived image matches between the branch GETFOOD and GOTOWC are flipped. Therefore, GOTOWC allocates more psychic intensity because of the high perceived image match. Three images are activated for the drive <DEPOSIT>, because of situational conditions. The interpretation is that <ADAM> may use this opportunity to satisfy the drive <DEPOSIT>, although the drive <HUNGER> is more urgent. On the other side, it is shown that the drive <HUNGER> still has a significant influence on the activation due to the strong need to satisfy it. However, as fewer images are activated, the agent has a reduced scope and possibilities to act on.

Branch	Test 1		Test 2		Test 3	
	PI Match	Number of activated images	PI Match	Number of activated images	PI Match	Number of activated images
GETFOOD	1.0	5	0.4	2	1.0	3
GOTOWC	0.4	1	1.0	3	0.4	1
ENEMY	0.0	0	0.0	0	0.5	1

Table 5-3: Test results of three performed tests (see Figure 5.5 for explanation of branches)

Finally, in test 3, the role of emotions in the activation of images is tested. There is only a partial perceived image match to the matching image of the branch ENEMY. It recalls a strong emotional experience and therefore activates one of the images. It allows <ADAM> to be able to react to external stimuli as this image will create a goal in the decision-making based only on the emotional strong experience. Strong memorized emotions may lower the importance of fulfilling drives for the reaction to a situation. It is a phenomenon that cannot be found in ACT-R in Appendix A because all goals are directed from the decision-making part and not from the perception.

5.3.3 Test case 3.3: Indirect Activation of Images

Indirect activation of images in the SiMASin World simulator shall be verified as proposed in Chapter 3.2.4. It is expected to see loaded images that originally do not have a direct match to the perceived image. It proves that psychic intensity is spread according to Test case 3.2 within the agent of the SiMASin World too.

All images of the act <A01_GOAROUNDSTONE> (see Table 5-1) have been connected through associations with their intention as well as their nearest neighbors as illustrated in Figure 5.6 in the middle. The use case is executed just a few simulation cycles before the first event <A01_GOAROUNDSTONE_L01_I01> is matched as seen in Figure 5.6 to the left. The only two images that will perfectly match the perceived image are the intention <A01_GOAROUNDSTONE_L01> and <A01_GOAROUNDSTONE_L01_I04> because they only

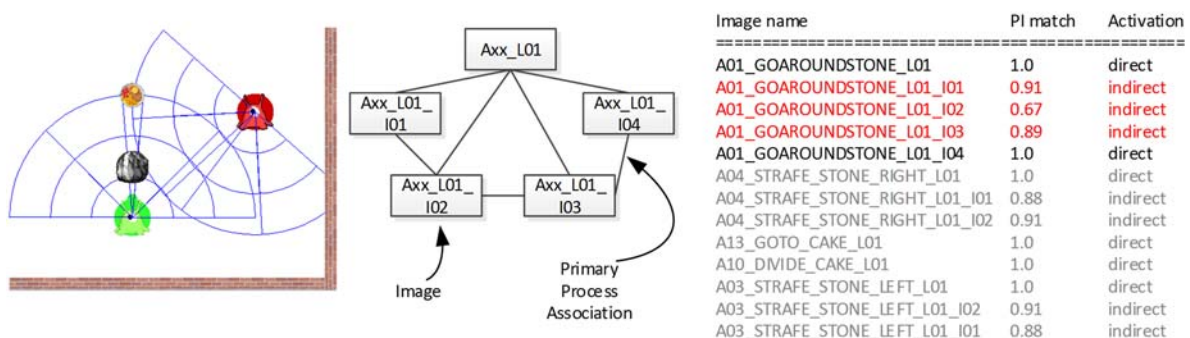


Figure 5.6: Direct and indirect activation of images (left) of the situation before the <STONE> is reached (middle) and where all acts are connected with primary process associations in a certain way (right)

demand a <CAKE> anywhere respectively in the center of the vision. For all other images in that act, no perfect match is possible. To demonstrate the indirect activation, the following parameters were used:

- Number of images that can be activated by direct activation = 6
- Threshold for direct activation = 1.0
- Psychic Intensity = 20

In Figure 5.6 to the right, the result is shown. Three further images of the act were indirectly activated (marked with red). Also, some other images of acts were activated (marked with gray), but they are not the focus of the test case.

5.4 Verification of Deliberative Decision-making

The implementation of the deliberative decision-making will be verified through test cases that have been derived from Use case 1 Obstacle Modification. The test cases in this chapter will prove that the following task has been solved: *Task 5: Refactor and extend the decision-making, in order to be able to reason about temporal data structures (Research question 5.1 and 5.2).*

5.4.1 Test case 5.1: Decision-making Process

In this test, it shall be analyzed how the temporal aspect of the decision-making process works compared to the proposed design in Chapter 3.4.3 and the implementation of it in 4.2.2. Different to ARSi11, an external action is always preceded by several internal actions.

The simulator will be analyzed at a point, where a possible goal from the act <A01_GOAROUNDSTONE> is being fulfilled. The inspectors in the modules Reality Check wish fulfillment (F51), Decision-making (F26) and Evaluation of Imaginary Actions (F29) in Figure 3.20. The decision process will be played through step-by-step to demonstrate how experiences are used. The description of the decision process will orient itself on Figure 4.7 and Figure 4.8. The use case has been performed to a point, where <ADAM> starts turning around the <STONE>, just as in Test case 2.1 in Figure 5.2. The process will focus only on the path to a particular planned goal that makes <ADAM> going around the <STONE>. In the following figures, this route is highlighted red.

The list of possible goals in Figure 5.7 has been reduced to five possible goals, to only show relevant results. The first possible goal is <ENTITY2IMAGE (CAKE)>. It is a possible goal that originates directly from the perceived image. The relevant drive is hunger; the drive object is the <CAKE> and due to the libidinous property of the possible goal, the <CAKE> shall be eaten. The importance is here 0.23, which is not very high compared to the other possible goals. Finally, the goal attributes are viewed, which directly influence the importance. The condition <IS_NEW_GOAL> tells that the possible goal has been initialized in step 2 of Figure 4.7 and have had basic processing of effort in step 3. However, it has not been selected for processing any further. The last goal of the list originates directly from the drive <HUNGER> itself.

Inspector of module F51 Reality Check Wish Fulfillment – Available possible goals

Number of possible goals: 60

Name	Type	Origin	Importance	Goal attributes
ENTITY2IMAGE (CAKE)	Hunger (libidinous)	Perception	0.23	IS_PERCEPTIONAL_SOURCE, IS_NEW_GOAL
A01_GOAROUNDSTONE_L01	Hunger (aggressive)	Act	1.07 [JOY=0,80]	IS_MEMORY_SOURCE, IS_NEW_GOAL, ...
A01_GOAROUNDSTONE_L01	Hunger (libidinous)	Act	1.50:[JOY=0,80]	IS_MEMORY_SOURCE, OBSTACLE_SOLVING, IS_CONTINUED_GOAL, SET_INTERNAL_INFO, SET_FOLLOW_ACT, SET_BASIC_ACT_ANALYSIS, NEED_PERFORM_RECOMMENDED_ACTION, NEED_GOAL_FOCUS, SET_FOCUS_MOVEMENT, IS_CONTINUED_PLANNEDGOAL
A08_BASE_DIVIDE_CAKE_L01	Hunger (libidinous)	Act	0.32:	IS_MEMORY_SOURCE, ...
DRIVEWISH (CAKE)	Hunger (libidinous)	Drive wish	-0.55	IS_DRIVE_SOURCE, IS_NEW_GOAL
...				

Figure 5.7: Reduced list of possible goals that have been initialized and processed before decision-making

Of the three possible goals from acts, the red marked possible goal <A01_GOAROUNDSTONE_L01> (see Table 5-1) was selected for planned goal in the previous cycle. It is indicated by the goal attribute <IS_CONTINUED_PLANGOAL>. It originates from the object representative <CAKE> of the intention. This goal has reached step 8 of Figure 4.7 and has been stored in the short-term memory in the last step.

The list of drive wishes and feelings in Figure 5.8, corresponds to step 4 of Figure 4.7. In decision-making, all possible goals are evaluated regarding their ability to satisfy a drive wish and to match the feelings. The drive wish <Hunger (libidinous)> defines hunger that can be satisfied by eating a <CAKE>. The importance that originates from the quota of affect are 0.32. It shows that this is the most important drive at the moment. Therefore, possible goals that satisfy this drive will be preferred. The feeling <JOY> with an intensity of 0.51 will help to increase the importance of any possible goals that can satisfy <JOY> if they are reached, e.g. <A01_GOAROUNDSTONE_L01> in Figure 5.7.

Inspector of module F26 Decision Making – Drive wishes and feelings

Drives

Name	Drive Action	Goal Object	Importance
Hunger (libidinous)	EAT	CAKE	0.32
Hunger (aggressive)	EAT	CARROT	0.10
Libidoanal (aggressive)	BEAT	BODO	0,01
...			

Feelings

Name	Importance
ANXIETY	1.00
JOY	0.51
ANGER	1.00

Figure 5.8: drive wishes and feelings on the input to decision-making

At this part of the process, the goal attributes of the planned goal are analyzed. For all goals that pass decision-making in step 4 of Figure 4.7, a decision about the next action is made. In the case of the planned goal discussed above, some internal actions have to be executed before <ADAM> executes an external action to move somewhere. The scheme used to get from one decision to the next for an act is described in Table 4-3. Associated with the planned goal are the goal attributes <SET_INTERNAL_INFO>, <SET_FOLLOW_ACT>, <SET_BASIC_ACT_ANALYSIS> and <SET_FOCUS_MOVEMENT>. These goal attributes are all results of the application of consequences of actions in step 10 of Figure 4.7. They tell the system the following: (1) it has been searching for relevant images to the act; (2) the basic act analysis has been performed; (3) the action associated with the moment shall be executed; and (4) focusing on the vision area in front of the movement location has been performed. The goal attributes <NEED_GOAL_FOCUS> and <NEED_PERFORM_RECOMMENDED_ACTION> inform decision-making that two actions still

Inspector of module F29 Evaluation of Imaginary Actions – Available possible goals and selected action

Possible goals

Name	Type	Origin	Importance	Proposed Action	Goal attributes
A01_GOAROUNDSTONE_L01	Hunger (libidinous)	Act	1.50:[JOY=0,80]	FOCUS_ON	... SET_INTERNAL_INFO, SET_FOLLOW_ACT, SET_BASIC_ACT_ANALYSIS, NEED_PERFORM_REC..., NEED_GOAL_FOCUS, SET_FOCUS_MOVEMENT...
A04_STRAFE_STONE_RIGHT_L01	Hunger (libidinous)	Act	1,47:[JOY=0,80]	SEND_TO_PHANTASY	OBSTACLE_SOLVING...
A01_GOAROUNDSTONE_L01	LibidoOral (Speak)	Act	1,18:[JOY=0,80]	SEND_TO_PHANTASY	OBSTACLE_SOLVING...

Selected action from plan goal

Name

FOCUS_ON

Figure 5.9: Selection and evaluation of the three possible goals with the highest importance as well as the action that is executed

have to be carried out before the external action can be executed. It demonstrates the multi-cycle, deliberative decision-making. It takes several cognitive cycles of internal actions to produce an external action.

In Figure 5.9, the steps 5, 6 and 7 of Figure 4.7 have been completed for each of the three selected possible goals of step 4. It can be seen that three possible goals originate from two different acts, where two of the possible goals concern eating the <CAKE>. Compared to all other 57 generated possible goals, their importance was higher, which means that they are possible to reach with low effort and they can satisfy the strongest needs of <ADAM>.

Out of these three possible goals, the possible goal of the act <A01_GOAROUNDSTONE> has been chosen to be reached. The action associated with the planned goal is <FOCUS_ON>. Due to the goal attribute <NEED_GOAL_FOCUS>, the target of focusing will be the goal object, i.e. the <CAKE>. The setting for focusing on the <CAKE> has been made in step 6 of Figure 4.7.

5.4.2 Test case 5.2: Evaluation of Possible Goals

In the previous test case, it was shown how a decision was made. Out of all possible goals, only one goal is selected as a planned goal. The planned goal had the best evaluation compared to all other goals. It shall be shown, how goals are evaluated.

The use case is played until a random point during the execution of the act <A01_GOAROUNDSTONE> is reached. Here, that point is the same as in the previous test case. At this point, the planned goal and the possible goals from the module Decision-making (F26) of Figure 3.20 are analyzed regarding defined evaluation criteria according to Chapter 4.2.2. Inspectors of the modules Decision-making (F26) and Evaluation of Imaginary Actions (F29) are applied to visualize the result.

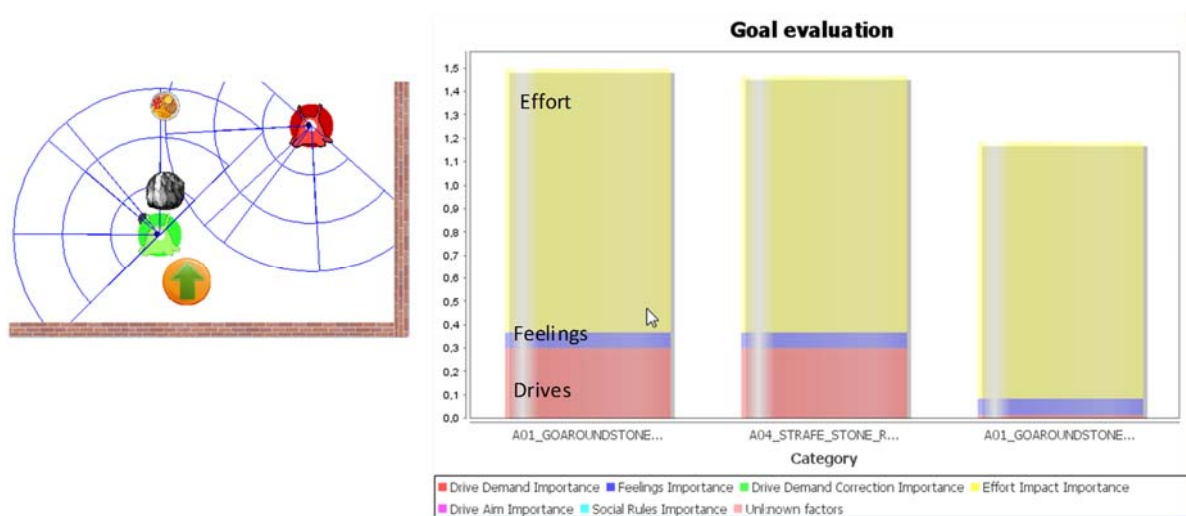


Figure 5.10: Description of three possible goals (left) for a certain situation in a screenshot (top-right) and the visualization of the possible goals (bottom-right)

Before the planned goal is analyzed, the factors, which influence the evaluation of possible goals, are listed in Figure 5.8. The top three drive wishes show that only the need to eat a <CAKE> is significant. Its importance is 0.32 (from Chapter 5.4.1). Feelings have the role of reinforcing certain possible goals in this implementation. The feeling JOY has an intensity of 0.51. Additionally, the effort plays a major role for usually decreasing the importance of the possible goal. The components of the effort have been described in Chapter 4.2.2. By looking at the situation in Figure 5.10, as expected, the possible goal, which will bring <ADAM> around the <STONE> to get to the <CAKE> has the highest total importance of 1.50.

This value is the total importance. In this example, the goal object <CAKE> could theoretically satisfy the drive wish for hunger with 0.8²⁰. However, only the importance of 0.32 is needed to satisfy it (drive wishes in Figure 5.8). In this implementation, the same applies to feelings. If the <STONE> is gone

²⁰ The importance value for the possible goal originates from a memorized drive mesh with the quota of affect of 0.8

around and the <CAKE> is eaten, it brings the intensity of the feeling JOY for 0.80. As there is only a need for JOY for 0.51, 0.51 is added to the total importance. Besides minor factors, which influence total importance, the effort is determined. The goal attributes either add or subtracts importance. For instance, effort is increased if the goal object is far away or the more uncertain the matching of an act is. The uncertainty is here determined by the confidence of the act (see Chapter 3.3.2). In sum, the total importance of 1.5 is reached. It is worth noticing that the goal attribute <IS_CONTINUED_PLANGOAL> adds a minimal bonus to the currently selected planned goal. This small addition makes <ADAM> preferring to fulfill the last planned goal instead of changing to other possible goals.

The test case shows that there are multiple means of evaluation of possible goals available. They reach from drives over emotions to the effort of reaching the desired possible goal.

5.4.3 Test case 5.3: Three Decision-making Paths

The purpose of the following test case is to demonstrate the three implemented decision processes in SiMA and how they interact with each other. Each type of possible goal gets its own handling. These processes are defined in Chapter 4.2.2 for possible goals from drive wishes (Table 4-2), perception (Table 4-1) and acts (Table 4-3). Further, this test case shall demonstrate the robustness of the system, i.e. that <ADAM> always finds a plan for any situation in this environment. Here, the path of the use case is followed.

In the configuration of <ADAM>, the three decision processes have different efforts. The easiest way to reach a <CAKE> in the use case is to follow perception because the <CAKE> is visible. With this type of possible goal, <ADAM> acts on preprogrammed, simple patterns and is not able to go around an obstacle like the <STONE>. As act analysis has to be performed for each moment of an act, the effort of reaching a goal is higher than following a goal extracted from perception. However, memorized feelings increase the importance of the possible goal of the act. Otherwise, <ADAM> would not be able to go around the <STONE>. The decision process with the highest effort is the decision process that originates from drive wishes. Here, <ADAM> will start a search behavior, to find the goal object. If it is found, then no search is necessary. Therefore, this option is rated with the highest effort and should only be chosen if there is no alternative available.

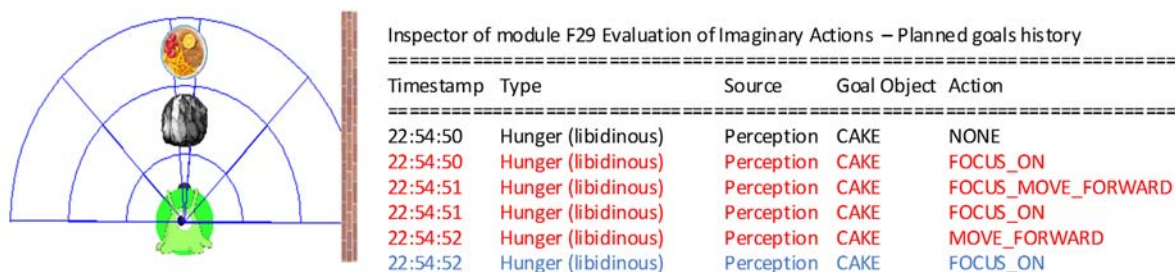


Figure 5.11: Decision of the external action <MOVE_FORWARD> based on a planned goal that originates from the perceived image

In the following simulation, the number of acts in the system has been reduced to a minimum, to demonstrate the swapping between the three types of possible goals. The standard use case is started, where the agent follows the decision process of perception to get to the <STONE>. Once it reaches the <STONE>, the decision process for acts is activated as the planned goal shifts, to pass a stone.

At a certain time during the act, <ADAM> is manually positioned at another place on the map, where there is no <CAKE> in sight. To find a <CAKE>, the decision process for possible goals extracted from drive wishes is followed. As soon as the <CAKE> is found, <ADAM> changes decision process according to a goal of perception.

<ADAM> started the use case in Figure 5.11 by moving towards the <CAKE> in the perception. For <ADAM>, the <CAKE> has the position (<FAR>, CENTER). However, to be able to move somewhere, <ADAM> has to build his environmental image, to see where to move. Therefore, some focusing has to be performed first. The first internal action is FOCUS_ON, which makes <ADAM> add the goal object <CAKE> to the environmental image. As <ADAM> will move forward, the internal action FOCUS_<MOVE_FORWARD> adds the object representatives in front of him to the environmental image. Then, after focusing once again, the external action <MOVE_FORWARD> is executed. The whole decision path is marked red in the figure. As can be seen, due to implementation issues in the software version used for this test case, this decision process differs slightly from the intended decision path.

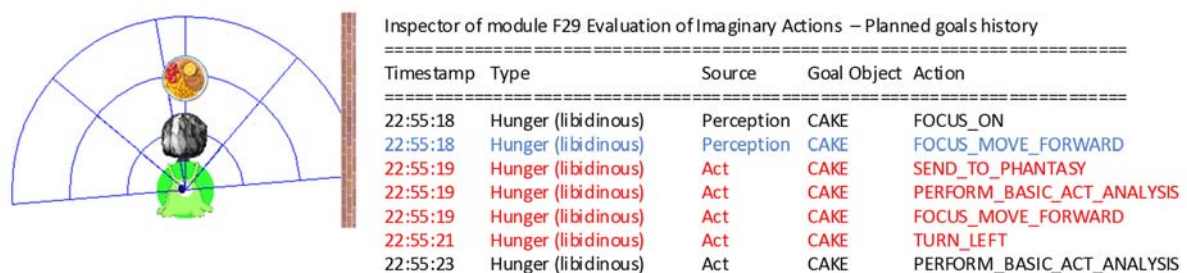


Figure 5.12: Decision of the external action “TURN_<LEFT>” based on a planned goal that originates from the act A01_GOAROUND<STONE>

In Figure 5.12, <ADAM> has reached the <STONE>. The perceived image triggered the act <A01_GOAROUNDSTONE>. As the possible goal extracted from this act is more important to follow than the possible goal from the perceived image, the decision process changes (marked blue). Here, the first internal action <SEND_TO_PHANTASY> (marked red) is to activate more relevant images by sending the intention of the act back to the primary process through fantasy. Then, with <PERFORM_BASIC_ACT_ANALYSIS> act is analyzed, in order to extract the moment and expectation. At that point, also the external action is known and will be executed to reach the expected state. For the movement, the internal action <FOCUS_MOVE_FORWARD> and finally in this case <TURN_LEFT> is executed. Then the decision path starts from <PERFORM_BASIC_ACT_ANALYSIS> again.

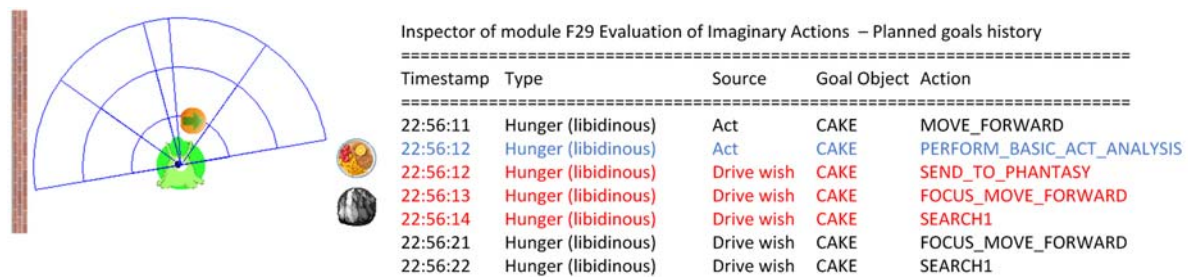


Figure 5.14: Decision of the external action <SEARCH1> based on a planned goal that originates from the drive wish, which represents hunger

At some point during the persuasion of the act, <ADAM> was manually moved away from the <STONE>. It is visible in Figure 5.14. After recognizing that this act does not fit the current situation anymore and because there is no <CAKE> in the field of vision, <ADAM> switches to the last decision process, the decision process based on the drive wishes. Here, the same is done as for the acts. First, the <CAKE> is sent to fantasy to try to activate some act related to the object <CAKE>. As nothing is found, <ADAM> starts searching his environment by first focusing on the area where the movement will take place with <FOCUS_MOVE_FORWARD> and then perform the action <SEARCH1>. <SEARCH1> is a certain movement pattern, combined from first <MOVE_FORWARD>, then <TURN_LEFT> and finally <TURN_RIGHT>.

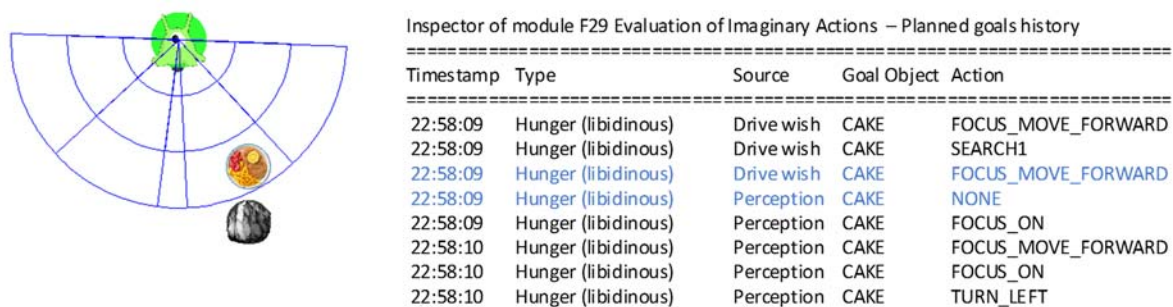


Figure 5.13: Switch of possible goals from a planned goal based on a drive wish to a planned goal based on perception

After searching a while, <ADAM> sees the <CAKE> and depending on the activated acts, one of the two other decision processes will take place. In Figure 5.13, <ADAM> switches to the decision process for planned goals from perception again.

This test case did demonstrate the three built-in decision processes. They allow different decisions to be made depending on the nature of the goals, which enables flexibility of the system. Finally, the robustness of the decision-making could be shown, as <ADAM> was moved manually, while he was pursuing a plan. <ADAM> had to adapt the decision process to the new situation.

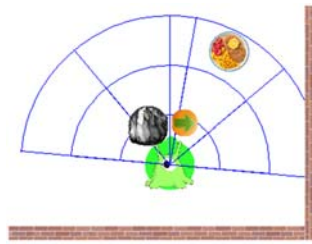


Figure 5.15: Situation for testing the environmental image

5.4.4 Test case 5.4: Focus of Attention and the Environmental Image

The environmental image of Chapter 3.4.4 represents the perception of <ADAM>. It is strongly linked with the concept of focus of attention in Chapter 3.4.5. The objective of this test case is to show how the focus of attention works and how the environmental image is updated with the data provided by the focus of attention.

A modified version of the use case is applied, where the objects are not in their original positions. It is used, to force <ADAM> not to go around the <STONE>, but to be guided by perception and to put external objects in certain sectors of the vision. There, <ADAM> has first to focus the goal object, then the area around him. The setup is shown in Figure 5.15.

In Figure 5.16, the first action is <TURN_RIGHT> at cycle 1. Any changes to the environmental image are marked red. At this point, all perceived objects representatives lose their positions by getting <(null, null)>. The reason is that <ADAM> moved. The previous environmental image does not correspond to the spatial setup of the environment and has to be refreshed. However, <ADAM> knows that there are still external objects somewhere around him. In the environmental image, it is visible since how many cognitive cycles the object representative remains in the memory. The syntax used is

Inspector of module F29 Evaluation of Imaginary Actions – Planned goals history and environmental image

Cycle #	Action	([# of cycles since update, Object(position, distance)])
1	TURN_RIGHT	[2, STONE(null:null)], [2, EMPTYSPACE(null:null)], [2, EMPTYSPACE(null:null)], [2, EMPTYSPACE(null:null)], [2, EMPTYSPACE(null:null)], [1, CAKE(null:null)]
2	FOCUS_ON	[3, STONE(null:null)], [3, EMPTYSPACE(null:null)], [3, EMPTYSPACE(null:null)], [3, EMPTYSPACE(null:null)], [3, EMPTYSPACE(null:null)], [0, CAKE(MIDDLE_RIGHT:FAR)]
3	FOCUS_MOVE_FORWARD	[1, CAKE(MIDDLE_RIGHT:FAR)], [0, STONE(MIDDLE_LEFT:NEAR)], [0, EMPTYSPACE(CENTER:NEAR)], [0, EMPTYSPACE(MIDDLE_RIGHT:NEAR)], [0, EMPTYSPACE(CENTER:MEDIUM)], [0, EMPTYSPACE(CENTER:FAR)]
4	FOCUS_ON	[1, STONE(MIDDLE_LEFT:NEAR)], [1, EMPTYSPACE(CENTER:NEAR)], [1, EMPTYSPACE(MIDDLE_RIGHT:NEAR)], [1, EMPTYSPACE(CENTER:MEDIUM)], [1, EMPTYSPACE(CENTER:FAR)], [0, CAKE(MIDDLE_RIGHT:FAR)]
5	TURN_RIGHT	[2, STONE(null:null)], [2, EMPTYSPACE(null:null)], [2, EMPTYSPACE(null:null)], [2, EMPTYSPACE(null:null)], [2, EMPTYSPACE(null:null)], [1, CAKE(null:null)]

Figure 5.16: Sequential construction of the environmental image memory showing the executed action (left) and the content of the environmental storage memory (right)

<[number of cycles in the memory, Object (distance, position)]>. It means that the object representative was added two cognitive cycles ago. As a parameter for this test, an object representative maximal stays in the memory for maximum four model cycles. This value has been chosen because no decision path of internal actions is longer than that.

The next action in cycle 2 is <FOCUS_ON>. Here the <CAKE> is added together with its distance and position <MIDDLE_RIGHT, FAR>. It replaces the <CAKE> without a position. Now, <ADAM> sees the <CAKE> and knows that the other object representatives are somewhere around him. This internal action is followed by <FOCUS_MOVE_FORWARD> in cycle 3, where the spaces near and in front of <ADAM> are added. Here the <STONE> is added. In cycle 5, after another <FOCUS_ON>, the external action <TURN_RIGHT> is executed, and the environmental image is reset once again.

The implementation of the environmental image allows <ADAM> to set multiple focuses before executing an action. Further, it realizes a form of object constancy as perceived object representatives from previous steps is compared to new ones. In that way, it is determined which of them are new and which did persist since the last update.

5.5 Verification of Act Recognition

The implementation of act recognition will be verified through test cases that have been derived from Use case 1 Obstacle Modification. The test cases in this chapter will prove that the following task has been solved: *Task 4: Develop recognition of the current situation of a sequence, extract the purpose of a sequence for goal generation and generate an expectation of what will happen next (Research question 4.1 and 4.2).*

5.5.1 Test case 4.1: Extraction of Intention, Moment, and Expectation

The most important tasks of act recognition in this work are to find the intention to extract possible goals to determine the moment and predict upcoming events through an expectation. The demonstration of how this is accomplished shall be shown here. It tests the concepts of Chapter 3.3.2. The use case starts from the beginning and is being analyzed step-by-step through the inspector of the module Evaluation of Imaginary Actions (F29) of Figure 3.20.

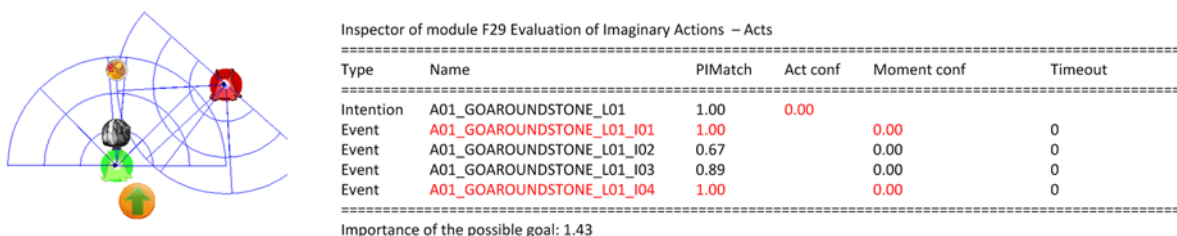


Figure 5.17: Status of the act <A01_GOAROUND_STONE> as it becomes a planned goal

As the first image of the act <A01_GOAROUNDSTONE> (see Table 5-1) is recognized, the act itself has not been analyzed in the decision-making yet. In Figure 5.17, therefore the act confidence (ActConf) is 0.00 and no moment has been determined. However, at this time, the intention was extracted to provide a possible goal. As the intention is much generalized, it matches perfectly with the situation (PIMatch=1.0) as long as any <CAKE> is in the vision.

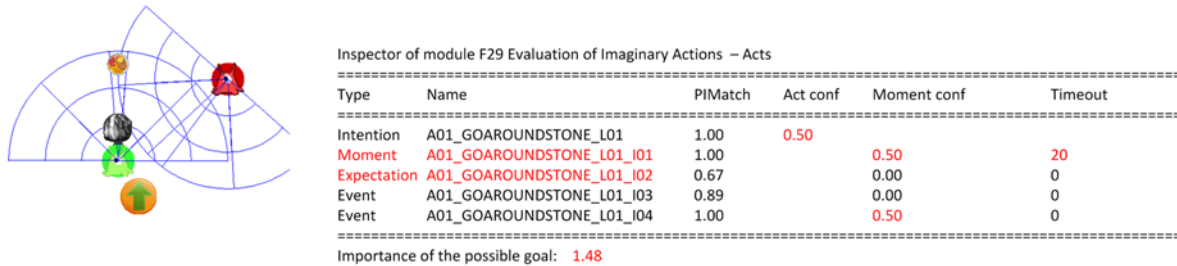


Figure 5.18: Status of the act “A01_GOAROUND<STONE>” as the intention, moment and expectation are recognized the first time

After the act has been analyzed, the result is presented in Figure 5.18. Several of the events match with the perceived image. The first (I01) and the fourth (I04) both match with 1.0. Therefore, both of them get the moment confidence of 0.5. The system needs to observe the situation, in order get more confident about the act. In the case of equal values at the beginning of the recognition, the default value has been set to favor the first image in the sequence as a moment. It has been done like that because it is more plausible that for a given situation and it is more frequent to start with the first event and not somewhere in the middle of the act. The act confidence increases from 0.0 to 0.5 also boosts the importance of the act for the evaluation from 1.43 to 1.48.

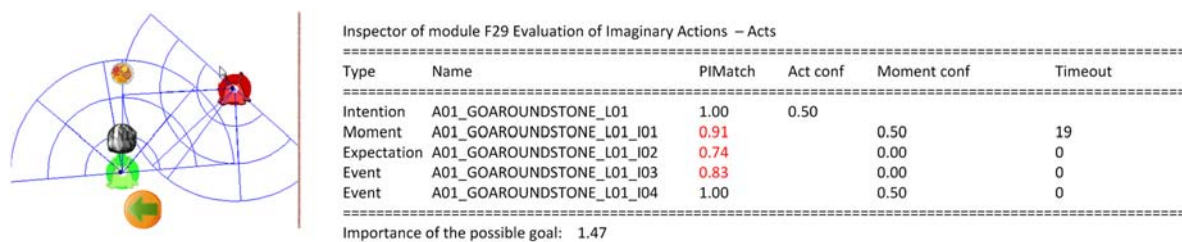


Figure 5.19: Status of the act “A01_GOAROUND<STONE>” as the moment does not have a perfect perceived image match

While <ADAM> moves between the first event (I01) and event (I02), there are no good matches anymore. Although the fourth image (I04) would fit better than the first image (I01), it is not very plausible that there would be a jump from the first directly to the fourth event. The structure of acts is a temporal order. Therefore, in Figure 5.19, the moment remains the first image (I01).

After some model cycles, the second image of the act (I02) matches perfectly in Figure 5.20. The moment confidence of the second event (I02) is 0.95, which means that <ADAM> is sure that this is the moment of the act and not the fourth event (I04), with only moment confidence of 0.25. The reason is the causality. The second image succeeds the first image and it is a confirmed expectation. Previously, <ADAM> expected that the second event (I02) should match with the perceived image and it did so. The increased moment confidence also increases the act confidence to 0.97. Consequently, the importance of the act about other acts is increased from 1.47 to 1.50.

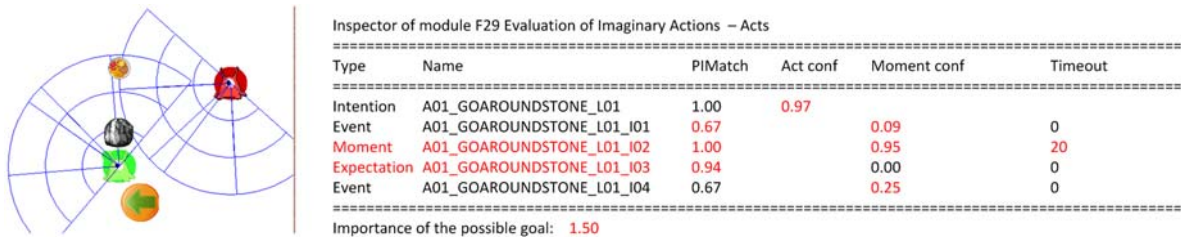


Figure 5.20: Status of the act A01_GOAROUND<STONE> as the second image of the act is confirmed as a moment

Finally, in Figure 5.21, it can be shown that <ADAM> is entirely sure that this act matches the situations as the third image is confirmed. An act confidence represents it equals 1.0.

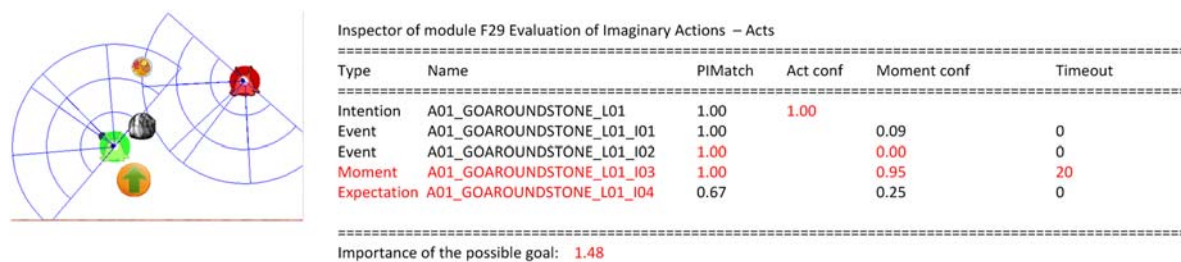


Figure 5.21: Status of the act "A01_GOAROUND<STONE>" as it is the third image of the act is confirmed as a moment

It has been shown how the moment can be determined based on the perceived image match and the order of the events in the act. The more the causality is confirmed, the higher is the confidence that the act has been correctly recognized. It can be used for either solving problems or to make predictions about external events.

5.5.2 Test case 4.2: Combination of Multiple Acts

According to the SiMA theory of [DFZB09, p. 210], "Expectations about how to achieve the satisfaction of a desire ("wishfulfillment") take the form of action plans expanding the desire into goals and sub-goals". Until now, no sub-goaling was performed and all goals were reached with a single act. A first, natural step to prepare for a realization of future work of sub-goaling, is to allow forward-chaining according to Chapter 4.2.2. With the current infrastructure, it is possible to combine

several acts to solve a specific task, in order to reach a desired goal. The next step would be to implement backward-chaining with acts, where the reasoning would start from the goal and infer the acts necessary to execute to get there. However, it has to be noted, that this is only a first step of similarity reasoning in the secondary process. Therefore, in this test case, it shall be demonstrated that an act is not only an independent piece of experience but that it can be combined with a series of acts. Instead of the standard starting position of the use case, <ADAM> starts beside of the <STONE>. The

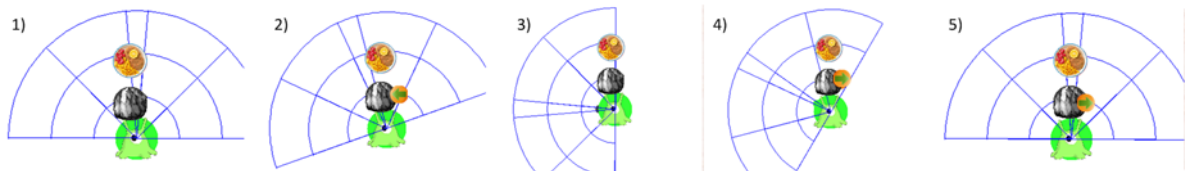


Figure 5.22: Execution of the actions of the act A03_STRAFE_STONE_LEFT>

<STONE> is no more in the position center. It causes a perceived image match of the first image of <A01_GOAROUNDSTONE> to be smaller than 1.0. The problem is that <ADAM> has to use this act to go around the <STONE>. Therefore, two additional acts were added to the long-term memory: <A02_STRAFE_STONE_RIGHT> and <A03_STRAFE_STONE_LEFT>. These acts cause <ADAM> to shift his position until he reaches the initial position of <A01_GOAROUNDSTONE>.

In Figure 5.22, <ADAM> starts to the right of the <STONE> in (1) and then strafes left in (2), (3) (4) to align himself. Finally, in (5) the starting position of <A01_GOAROUNDSTONE> has been reached and this act can be started.

5.6 Validation of Use case 1 Obstacle Modification

After every individual test case for each of the four main topics of this work has been described, the functionality of Use case 1 Obstacle Modification as a whole shall be verified. It will show that the current implementation is a satisfying solution to Task 1.

In Figure 5.23, <ADAM> starts by moving towards the <STONE> in (1, 2), whereby his goal is the <CAKE>. As <ADAM> reaches the <STONE>, he uses his stored experiences to find a solution to pass the <STONE>. He recalls an experience, where he already performed that task in (3, 4, 5, 6). After rounding the <STONE>, <ADAM> continues to the <CAKE> and eats it (7). Just after the <CAKE> has been eaten, <ADAM> sees <BODO>. As <ADAM> still has hunger, processes in his defense mechanisms cause him to beat <BODO> instead (8).

The final use case and the result mainly cover the proposed use case in Chapter 1.3, which was the starting point of this work. Above, it was shown that the agent is capable of solving the problems of the SiMASin World within the scope of Use case 1 Obstacle Modification.

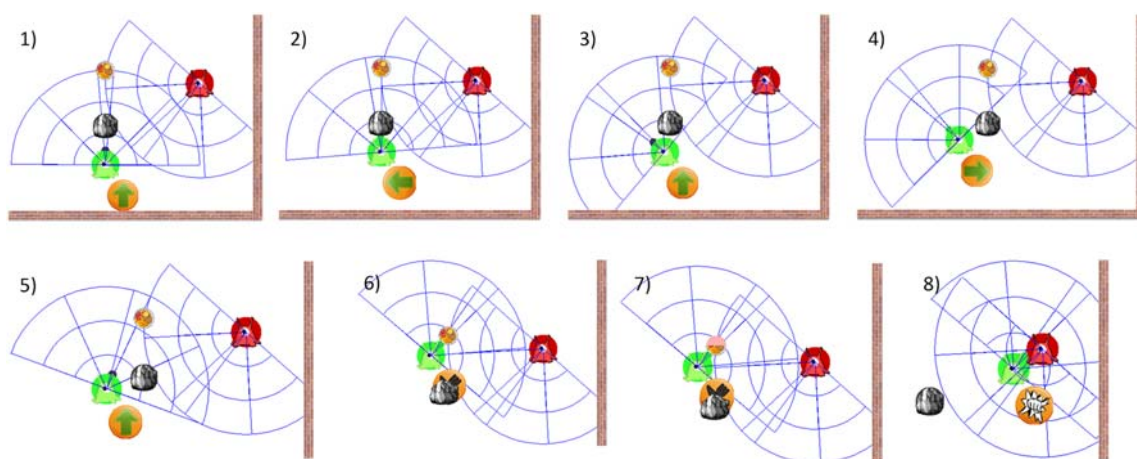


Figure 5.23: Execution of Use case 1 Obstacle Modification

5.7 Utilization of this Implementation in Comprehensive Use cases

The purpose of Use case 1 Obstacle Modification was to demonstrate the implemented functionality in detail, and it is easier to show if the use case is kept simple. However, it does not indicate how the cognitive architecture SiMA makes decisions used in complex situations. In the following, results from related SiMA use cases are presented, where this work provided SiMA with the infrastructure of decision-making through experiences.

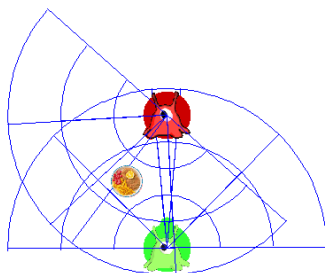


Figure 5.24: Initial setup of the simulation case with several options determined by four determinants

In [SWJ+14, pp 155-164], [SWK+15, pp. 215-220] and [WGF+15, pp. 330-335], the concepts of this work were presented with a methodology for developing use cases that should be able to test the functionality of the SiMA. The focus lays on interdisciplinary. The idea is to use a narrative story that has been defined by psychoanalysts about assumed behavior in different situations. The initial setup is shown in Figure 5.24. The narrative use case is then transformed by engineers to a structured simulation case with four determinants: the agent's experiences, personality parameters, the internal state and environmental state. Based on the initial settings of those parameters, the agent will act differently like in Figure 5.26. The task is then to see which parameters generates which behavior and

aggressive part of the hunger is not blocked in the defense mechanisms, the agent <ADAM> attacks <BODO> at once.

Next, it can be looked at how each action influences the drive state. Then, of course, the drive state decides, which action to be made next. In Figure 5.25, the drive state history for the scenario called the eat scenario, is plotted. In the first 15 steps of the simulation, <HUNGER> is dominating. Therefore, acts, which lead to satisfaction of hunger are favored. From those acts, a planned goal is

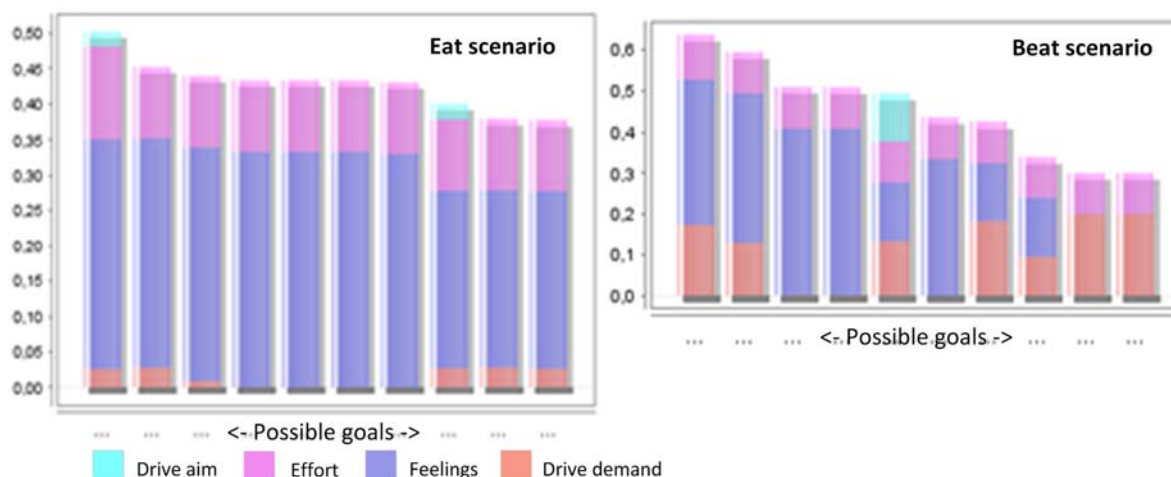


Figure 5.27: Exemplary goal evaluation of the behavior sequence eat scenario and the beat scenario [SKZ+15, pp. 346-347]

selected. After eating at step 25, a sexual drive has the highest importance, and it can be satisfied through beating <BODO>. Therefore, acts that match that sexual drive and the action <BEAT> are favored in selecting planned goals.

In Figure 5.27 from [SKZ+15, pp. 346-347], the impact of the different importance types for the eat scenario and the beat scenario are shown. It shows a more comprehensive evaluation than in the test case of Chapter 5.4.2. In the eat scenario of Figure 5.27, four factors of importance play a role. Surprisingly, the drive demand does only play a small role. The state of the feelings is much more important, while the effort is equal for all relevant possible goals. Finally, the matching with the drive aim adds some importance to the selected planned goal. In the beat scenario, the drive demand plays a much bigger role in the evaluation of possible goals than in the eat scenario. In only one of the possible goals, the drive aim matches the goal object. Hence, the importance of that possible goal will still not be high enough to make it a planned goal.

6. Conclusion

“Could it be that one far-off day intelligent computers will speculate about their own lost origins? Will one of them tumble to the heretical truth, that they have sprung from a remote, earlier form of life, rooted in organic, carbon chemistry, rather than the silicon-based electronic principles of their own bodies?”

[Richard Dawkins: “The Blind Watchmaker”]

SiMA is a cognitive architecture based on the theory of psychoanalysis. It originally emerged in an attempt to solve the problem to make sense of the thousands of sensor data in the area of building automation. Different to most other cognitive architectures, the focus of SiMA has been put on the unconscious data processing, i.e. in the primary process. Therefore, no temporal structures were defined. The motivation of this work was to provide SiMA with more human-like functionality like the prediction of future events based on previous experiences. It includes the consequences of the own actions of the system. The main research question was how to make decisions based on experiences. Emphasis was put on experiences that provide information about the causality of actions and observations.

With the method of a partitioned iterative approach, first data structures for representing causality were proposed and then situation-relevant memories were loaded. It leads to concepts of how sequences of experiences can be used to estimate the further cause of a situation. Finally, to be able to make use of these causalities, decision-making was redesigned and extended.

In the following sub-chapters, the conclusions of this work will be discussed. The provided answers correspond to the research questions, as well as other topics, which emerged during the work. Based on the conclusions, the course of further work will be defined. Finally, an outlook is provided with a vision of how SiMA can be used in the future.

6.1 Conclusion

The conclusions have been grouped by topic. First, general conclusions about the SiMA model in the domain of cognitive architectures are drawn. Then, according to the four main tasks of this work, conclusions for each task are presented.

SiMA in the Domain Cognitive Architectures

To be able to place SiMA in the domain, in Chapter 2.1, the common functionality of cognitive architectures, in general, was analyzed. These functionalities define a cognitive architecture. They were used to make a direct comparison of SiMA with another state of the art architectures in Chapter 2.3. From the perspective of modeling, SiMA is a unique model with concepts that cannot be found somewhere else. In the analysis, the strengths and weaknesses of SiMA were compared to other cognitive architectures. The main unique selling points of SiMA are the following: SiMA is the only cognitive architecture, which is based on psychoanalysis. The choice of theory has given SiMA an edge to the state of the art in the modeling of unconscious processes. Such a concept is the unique concept of defense mechanisms. Further, the definition of several differentiated evaluation systems reaching from affects, drives, efforts, emotions and feelings count to the strengths of SiMA. The project group has accomplished to merge the first and second topographical models of Sigmund Freud by defining a strict separation of functionality and data. Finally, different to most other cognitive architectures that claim to model the human mind, SiMA is based on one single theory, where great effort is put in developing an axiomatic, theoretical base. It is a real gain for SiMA because much of the functionality that many other cognitive architectures are discovering is already an inherent part of the SiMA model.

In the area of logical reasoning, learning and information representation, there is an accumulated need for further development. Fortunately, with this work, by introducing temporal functionalities and multi-cycle decision-making, a significant step was taken in this direction.

When it comes to the implementation of the architecture, there is still much work to do. Although other cognitive architectures have been mentioned in previous publications of SiMA, there have not been much of an interface or comparison to them through common test setups, e.g. the blocks world. It had the consequence that SiMA has been developed mostly in isolation, ignoring the state of the art programming techniques. In the future, it is important to emphasize that model and implementation are two entirely different things. While a model can have any origin, in a common programming language, there are only a limited number of designs available for realizing the SiMA model. Many of them have already been applied in other architectures, and it is a good idea to look at them.

SiMA and Psychoanalysis

In the scientific community, the validity of psychoanalysis is controversial, and it is a long way getting psychoanalysis accepted as a base for cognitive architectures. This architecture has been developed almost independently of influences from other psychological schools. However, this work has shown that SiMA is as valid for technical applications just as any other cognitive architecture. This claim is supported by the argumentation in Chapter 2.3.6. It does not matter, whether one believes in the theory of psychoanalysis or not. SiMA already incorporates many of the concepts, which other architectures eventually implement as they discover that they need it to solve their problems. Examples of such functionality are the focus of attention, emotions and the generation of imaginary actions. Therefore, the convergent functionality makes psychoanalysis valid a base for cognitive architectures just as any other theory.

On the other side, other architectures are still more developed in the software implementation and in the testing of different individual concepts. This work still has to be done in SiMA, considering the constraints of the psychoanalytical theory. To reach acceptance in the scientific community. Therefore, the focus should be put on generating empirical results rather than developing more theoretical concepts. These results should then be graphically and impressively presented at conferences, contests, and experiments.

Experience Based Decision-Making

In this work, the temporal perspective of decision-making was introduced into SiMA. The main research question of the work from Chapter 1.3 (Research question 1) was: How can SiMA be extended to be able to consider past experiences when taking decisions? In the previous implementation of SiMA, the ARSi11 version, the task of Use case 1 Obstacle Modification (see Chapter 5.1) would not be possible to complete without the usage of either concrete, remembered experiences or abstracted templates of action sequences. There, the task of the agent was just to go around an obstacle, in order to find an object that satisfies a drive. In the ARSi11 implementation, the agent could only select a default, external action within a single cognitive cycle and could not analyze different options or evaluate expectations.

The results of Chapter 5.6 showed that the agent now possesses the required capability. It validated that SiMA can match a given situation with remembered experiences and that it can activate relevant memories. They then formed acts. From these acts, possible goals were extracted. They were the options of the system and were evaluated regarding influence several factors like the drive wishes and effort to reach them. Finally, one of the possible goals was selected, which action should be executed. This process perfectly corresponds to the decision-making process of a cognitive architecture in Chapter 2.1.2. In the following subchapters, the individual sub-research questions will be discussed.

A look at the state of the art showed that most architectures rely on a heuristic search method and that only a few of them are based on case-based reasoning. If the case-based reasoning is used, mostly, states are directly mapped to actions. Many of these systems are using reinforcement learning [SB98, p. 63] or evolutionary computing [ES08, p. 115] to learn the state-action relation. These solutions may be suitable for playing chess like the analyzed cognitive architecture CHREST, but in many problems, there is a long delay between an action in the system and its consequence. Maybe a sequence of actions contributes to a particular state. For that, the mapping of actions to a sequence of states would be useful, and this is what has been realized in SiMA. Finally, the increasing usage of episodic memories in a state of the art gives a hint that it might be a good way to go with.

Extension of Data Structures for Temporal Processing

In Chapter 1.3, two sub-research questions were derived from the main research question of this work. The first question (Research question 2.1) was: How shall data structures for situations be designed? In this work, the concept of an image was redefined compared to previous work in SiMA. The key was to make the image scalable, by using one data structures for many theoretical concepts, especially for images (see Chapter 3.1.1). In the primary process, it was realized by using the thing presentation mesh for both perceived objects as well as situations like the perceived image. For the first time, a

common, scalable data structure was available in the implementation to describe a situation. In ARSi11, only perceived objects representatives (also known as percepts in the research area AI) did exist independent of each other.

In the secondary process, the word presentation mesh was introduced as an equivalent to a thing presentation mesh (see Chapter 3.1.2). It allowed the usage of directed associations with predicates. Two images could then be linked together by e.g. a temporal association, giving a causal order of images. The causality was the topic of the second research question (Research question 2.2): How shall situations be organized into sequences? The image formed the base for an act, which is the first temporal structure realized in SiMA through this work. A key property of an act is that it is no data structure of its own, but instead, it is composed of basic data structures and defined through its associations (see Chapter 3.1.2). Acts describe sequences of template images and are used either as action plans or to describe causal events in the environment.

The implemented data structures origin from the concept of Sigmund Freud 100 years ago. In SiMA, these data structures have been implemented almost exactly according to Freud's statements. The concept provides a powerful way of expressing symbolic data. In this work, it could be shown that these data structures can be used in a very flexible way. Besides of images and acts, it was also possible to realize the possible goals with these data structures, and that was a very comprehensive data structure, which included feelings, acts and other objects.

However, at the end of the implementation phase, the flexibility had its price in performance. In SiMA, many theoretical concepts were translated one-to-one into object instances of software code, although it may not be the best solution from the point of software engineering. Measurements showed that 99% of the CPU time is covered by one operation, the cloning of data structures. It indicates that the data structures, as well as their retrieval, are major problems in the software development. A possible solution would be instead to separate the concept of psychoanalysis from the actual implementation and do as most other cognitive architectures do: To express the data structures in textual form, as in the languages of RDF [8] or OWL [9]. By doing that, the problem of keeping track of individual instances would be removed without losing expressivity of the data structures. It is no coincidence that string-based data structures are the state of the art in agent systems modeling. Additionally, string-based data structures have the advantage that they are immutable. In that way, modules of the system are completely decoupled and are not sharing the same memory by a data structure.

Activation of Experiences in the Primary Process

Experiences are stored in a long-term memory to be accessible for either action planning or prediction of future events. The next challenge (research question 3) can then be formulated as a question: How shall those stored situations be retrieved from the memory, which is relevant to the current situation within a certain context? Of all stored experiences in the form of template images, only template images that are somehow similar to the situation shall be activated as well as images that are related to other activated images. They shall be activated in the context of the drive state and under consideration of their emotional importance. As a solution, psychic spreading activation (see Chapter 3.2) was developed. The concept is based on statements from psychoanalysis about the activation of

memories in human incorporated into the existing theory of spreading activation. A psychoanalytically as well as a biologically plausible approach is provided.

Psychic spreading activation allows a proper way of querying images and can activate related images without direct querying. Instead of querying a complete database, only one or a few pieces of content need to be directly queried. The rest of the query can be said to be done automatically by the associations. The number of images or how images are activated can be controlled in several ways. The input of psychic intensity directly controls how many images can be activated, as the more psychic intensity can be spread from the input image through associations, additional template images can be activated. By limiting the number of possible direct matches, activation can be wide or focused. If it is wide, many direct matches to a source image (the input) are loaded but not on the depth. With a focused approach, images are loaded on the depth through indirect associations.

In the template images, the ability to allocate an amount of psychic intensity, in order to be activated is defined by a psychic potential. It is calculated from the association weight with the source image as well as the subjective importance of that template image for the situation. It is determined by the drive state and the emotional component of the image.

Psychic spreading activation and experiences as a whole can be interpreted as inferred beliefs (see Chapter 2.1.2). Compared to many states of the art architectures, still inferred beliefs, which are based on the relative positions of perceived objects are still missing. However, the activated experiences, which directly provides the decision-making with options fit the criteria of the phase infer beliefs.

Within the project, there has been much discussion about if images are allowed to be directly activated by the secondary process or not. From psychoanalysis, no satisfying, non-contradicting solution was suggested. The problem is the following: If images can be loaded only in the primary process, they all have to pass the defense mechanisms to be used in the secondary process. If images as thing presentation meshes can be loaded directly from the secondary process as a sort of action plan, the defense mechanisms can be bypassed, questioning their purpose. For this work, the first alternative was chosen to implement. It does not leave any room for contradictions in the model. It was a major design decision of this work. Additionally, phantasy is given a significant role in this work, because required content by the decision-making has to be activated in the primary process through phantasy.

The implementation shows that the concept can be implemented in any environment as a layer between the SiMA model and a database. However, this implementation was difficult to integrate into a database concept, because all associated images first have to be loaded from the database. Then, it is decided, which of them will be activated. The main issue of the current implementation is that it is not suited for a large data set as all images are compared with the source image every time image matching is used. Especially in SiMA, where all content is cloned from the long-term memory at loading, this is a serious problem. There are three ways of mitigating these problems. In future developments, parts of the memory should be able to be pre-activated. It means that only a subset of all available images would be available for the image matching. The rest should be activated through indirect associations. Such a solution would emphasize the main advantage of psychic spreading activation, which is to load content without needing to query the entire memory at once. Further, the handling of data should be revised. Instead of loading the whole situations in every cognitive cycle, a sort of working memory should be introduced. Then, the data structures could be loaded once and only getting updated with

necessary parts from the long-term memory. Finally, no “homemade” database solutions should be used, when there are multiple well-established solutions available.

Multi-Cycle Decision-Making

Activated template images of the primary process provide the base for more flexible options based on experiences. For that purpose, decision-making had to be redesigned. The first question (Research question 5.1) that arises is from the acts is the following: How shall decision-making be extended to consider temporal data structures? Within this work, the first step was taken to realize a deliberative sub-system in SiMA according to the definitions of Chapter 2.1.1, where a decision is made over multiple cognitive cycles and where multiple options are proposed and evaluated. In SiMA, the options are called possible goals. Finally, of all evaluated options, one option is selected, and its associated action is executed in the external environment. The selected option is called a planned goal in SiMA.

The critical point of handling with sequences is that an agent has to know about its history to have any expectations for the future. With the introduction of a short-term memory in Chapter 3.4.4, the possibility to plan over a short period was given and to predict the next upcoming events. Instead of evaluating every goal new in each cognitive cycle as done on the previous work of ARSi11, the agent is now able to act over time.

A direct consequence of the new decision-making in this work is that such a system can think before it acts. In the implementation, it is manifested by external actions that take longer than one cognitive cycle. Instead, the system makes decisions, which only affect the internal state and do not influence the environment. Those decisions can be seen as internal actions in opposite to external actions. They define what content shall be processed, in which order and how. It provides an agent with powerful tools to evaluate each option before acting.

Does it lead to the question (Research question 5.2): How shall internal actions be implemented within the decision-making? In combination with the short-term memory, the solution to this question led to the definition of a decision process in Chapter 3.4.3. Each possible goal is handled in a certain way at a certain place at each process step. First, it seemed to be a contradiction to have much functionality like the focus of attention located before the actual decision-making. The question was how the system shall know what to do with a planned goal before any decision has been made to define a planned goal. By storing possible goals in the short-term memory, internal actions could now be decided at any stage and then be executed in the next cognitive cycle. In that way, the location of vital functionality before decision-making fits perfectly in a multi-cycle decision-making.

The decision-making in this work allows the implementation of the feature of self-reflection in self-representation for the first time. In [Dob15, p. 14], the self-representation was analyzed. Much functionality can be mapped to the meta-management subsystem in of Chapter 2.1.2. In state of the art, self-representation is very rare. With this functionality, SiMA can be one of the first architectures to implement it. The necessary infrastructure has been provided as a working memory, a decision-making process that allows decision-making to execute many different paths. For self-reflection, such a path could be implemented that monitors other decisions of the system.

This is the first implementation of multi-cycle processing in a forward-chaining manner. It means that the goal is known and is pursued by analyzing the current situation only. With the present situation as a starting point, the options are proposed, which brings the system to a state closer to the desired goal state. A meta-management subsystem as defined in Chapter 2.1.1 is not implemented at this stage. The only form of monitoring is that the agents can abandon acts, which are no longer recognized within a certain time frame.

For the implementation of decision-making, the concept of codelets (see Appendix B) was chosen as a controversial solution to normal programming style in SiMA. Briefly, codelets replace long and complicated if-then-statements by executing independent code if certain conditions apply. This concept made the system simpler to understand, and it reduced complexity in the architecture by removing functions from the modules and encapsulating them into separate, independent building blocks. The conditions of those codelets were the goal attributes, which were defined as state markers of the possible goals. The usage of goal attributes in the possible goals can be seen as a pre-stage of an agent-based service oriented architecture, which would be a possibly to solve many of the design problems in SiMA.

Evaluation of Experiences in the Secondary Process

As all infrastructure for using acts is available, the next step was to use acts. It led to the question (Research question 4.2): How shall information from sequences be supplied to be useful in the decision-making? Acts can be used in two ways, either for action planning like finding a path based on landmarks or for prediction of external events. The solution was provided in Chapter 3.3.2 with the introduction of the intention. As the name says, the intention provides decision-making with the purpose of the act, which are options or possible goals for the system. Additionally, the moment and the expectation have to be recognized to provide value to the system. If an agent does not recognize how the act fits into the perceived situation, it is of no use.

The challenge of recognition was formulated with the question (Research question 4.1): How shall temporal recognition and confirmation of applicable sequences be realized? A key for the recognition was the introduction of the act confidence in Chapter 3.3.2. It is composed of matches between the perceived image and events of the act and how well expectations are fulfilled. The value of the confidence has a strong influence on the effort of reaching a possible goal, which is a vital part of the evaluation of options.

As finally shown in Use case 1 Obstacle Modification in Chapter 5.6, acts serve their purpose. As there are no dynamic external objects doing anything in the SiMAsin World, acts were exclusively used for planning actions, which were based on previous experiences of similar sequences of situations. It allows the customization of actions for certain situations.

Through the indirect activation of images in the primary process, an act can be activated, although it does not seem to have anything to do with the perceived image. In future developments, it opens up for the use of alternative solutions in the decision-making, which are not retrieved through the conventional logical reasoning of the matches with the current situations. As can be seen by human, unconventional solutions of problems is called creativity. This process is enabled through the primary process associations.

At the current stage, only basic handling of acts was conceptualized and implemented. The price of simplicity was the flexibility of the recognition. The perceived image almost had to match perfectly with any template image of the act. It made the actions, which were associated with the act very deterministic. However, this work was only the first step.

Until now, the concept of frame axioms (see Chapter 3.3.3) has not been used in SiMA as it seems to contradict the way of human thinking. Instead of analyzing the information that is available, frame axioms force the analysis of everything that is not available in the act. A concept of context would better fit into SiMA, where certain contexts are associated with the acts. At the configuration and practical application of multiple acts, the four problems of problem-solving in ICARUS [LCT11, p. 19] also applies to acts in SiMA. First, acts may be too generic and therefore provide the system with wrong actions. Acts may be too specific, where the system cannot recognize it in situations, where it would be applicable. Further, actions are only probabilistic, which may cause that the expectations are not correctly recognized. Also, a change in the environment may cause the selected act to be inappropriate [LCT11, p. 19].

Implementation of Human Comparable Intelligence

The physicist M. Kaku thinks that computers will develop real human-like artificial intelligence, also called the singularity, in about 20 until 1000 years from now. According to Kaku, the more intelligent species can adapt better to its environment and in that way be able to win the competition against their creators [10], [Kak11, p. 100-101]. It would be the case as soon as computers can reproduce themselves autonomously.

After analyzing the implemented functionality and evaluating the results of this work, it can be concluded that the singularity is still very far away. This research area is still struggling to realize even the simplest features of the human psyche. The resulting software program still does no more, and no less than the software developer has implemented. A terminator that is taking over the world will probably still have to wait a while.

To prevent a future domination of the robots, some possibilities are presented: Either rules are programmed into the robots preventing them from harming human or robots should be able to learn social behavior and from their own will not be able to harm human [Asi50, p. 27]. All of this would work well for a cognitivist architecture, where it is possible to control the internal representation of the robot. However, if computers that originate from the emergent or an evolutionary approach would be able to achieve human-like intelligence, no human could ever program anything into the robot after the start of the development. In that way, there is maybe no way to stop the revolution of artificial life, once the preconditions have been set.

6.2 Future Work

There are many conceptual tasks to develop within SiMA, reaching from how feelings influence decision-making to extend the flexibility of acts. The goal is to reach a robustness of the system to allow it to deviate from a planned path and let it explore unknown states without falling into a deadlock. The theory of SiMA can be interpreted as well developed. Therefore, many of the issues

that regard robustness lies in the software development. However, in the following, the most urgent problems where this work stops, will be described. Those issues will be grouped into topics.

Generalization of Data Structures

In Chapter 5.1, the object representative <CAKE> was used as an instance of a food source in an act. In the current system, this act only applies to <CAKE>. Any other food source will not match the act. The same statement is also valid for the obstacle <STONE>. The SiMA system could profit a lot on the generalization of object representatives. If the <CAKE> could be generalized to <FOOD> and the <STONE> to <OBSTACLE>, the generality of an act would be significantly increased. Further, if the <FOOD> would be something like <DESIRABLE OBJECT>, then the act would not be a “go-around-stone-act”, but a general solution, how to pass an obstacle in general. To achieve that, it would be required that at least in the secondary process, each object representative was categorized in the act analysis.

Reimplementation of Data Structures

The current data structures are not well scalable, and they provide unsolvable dependencies between modules of the system. The predicate logic used within the data structures of SiMA is expressed by implicit data structures²¹, which keeps the information in JAVA object instances. Lessons learned from this project have shown that a string-based representation of the data structures could be of benefit for the project. This idea originates from the state of the art architectures SOAR, BDI and ICARUS (see Appendix A). String based data structures do not alter the concepts of thing- and word presentations presented in Chapter 2.3.4 and 3.1. It is only another way of describing graphs. The Resource Description Language (RDF) [8] and its extension Resource Description Language Schema (RDFS) [11] offer these possibilities of expression. While RDF provides a knowledge representation system that can easily be extended at run-time, RDFS allows the basic description of ontologies. It would allow a consistent mapping of psychoanalytical concepts to RDF statements. The string-based approach can solve some problems, which occur with large meshes and still being compliant to the psychoanalytical theory. Further, data structures could also be expressed with first-order-logic like DDL. It is a compact way of expressing things and allows the reasoning over the structures [SWY11, p. 31]. The open question is whether they perform better than the currently used data structures.

Image Matching with the Whole Database at Memory Retrieval

One issue of the concept psychic spreading activation of Chapter 3.2 is that for the direct activation of template images by a perceived image, the whole database has to be loaded. It means that every time a perceived image triggers the psychic spreading activation, it is compared to the image matching algorithm of Chapter 4.2.1 with every template image of every experience in the database. It contradicts the idea that resources and time are saved by using a spreading activation concept. A

²¹ Implicit data structures are meant to be object instances of a programming language. They have their own structure and are not directly readable by human. Explicit data structures would be any string based data structure, which is human readable.

possible solution to the problem that was not implemented in this work was to use a sort of pre-activation of content that has previously been activated. A kind of activation value like in SOAR or ACT-R (see Appendix A) could be used, which is decaying with time, if no further activation is added. It would mean that everything that is activated through direct activation would get an activation value. The data structure itself would remain in short-term memory or cash and is directly compared with the perceived image in the next cognitive cycle. If the perceived image has not changed at all, then all the images that were activated in the previous cycle could remain, and some activation would be added to the decaying activation value. In that way, a lot of system resources would be saved and the concept of the direct activation would be more economical. If SiMA would be implemented as a non-serial system like a system that uses multiple threads, there could be some activation functions that constantly checks the match of the perceived image with the database and independent of other functions, continuously activates template images.

Splitting of Episodic and Lexical Memory

At the moment, SiMA uses one big knowledge base for everything. Lexical and episodic knowledge are mixed. Lexical knowledge is the knowledge about types of an object without any context. In SiMA simulator, this is knowledge about the object representatives like <CAKE>. Everything that is red and round is a <CAKE>. Episodic knowledge is different from its nature. It describes instances of types e.g. a <CAKE> with a certain position in a certain template image. It means that the lexical data have instantiated the episodic data. As this knowledge has different purposes, it may be purposeful to split them into two separate memories, one which stores template images and acts and one memory that stores the lexical knowledge. In most other architectures, this is the case (see Chapter 2.2).

Inference of Acts as Action Plans in Decision-Making

The power of acts has been demonstrated in a test case in Chapter 5.5.2, where multiple acts were combined to solve a problem. At the moment, the system uses a forward-chaining decision process for reasoning. It means that decision-making knows about the current situation and the end goal. It then selects solutions, which brings the system one state closer to its goals. In the specific test case, three acts are used in parallel. One act fits the situation and if the action of it has been performed, the second act will fit the end state of the first act. The system only knows that these acts will take the system a little bit closer to the end goal, but not how it will work. A concept of backward-chaining would be interesting to realize real planning. Here, it would go the other way around. The system starts with an end goal and reasons the way down to the current situation. Here, the whole path of actions would be inferred by decision-making. To realize that, a sort of reasoning function can be built that first analyzes a drive with or any other system goal. Then it looks in the long-term memory for acts, which intentions would satisfy the system goal. In the next step, the starting conditions are examined, and other acts are searched for, which end images would fulfill them. In that way, the system could automatically create a chain of acts to be executed from a larger database.

Impact of Feelings in Decision-Making

Feelings were only used to a minimal extent, in order to limit the scope of this work and focus on the recognition of acts. In this work, feelings played two roles: either to enhance an act by increasing its

importance or making the agent avoid things in the environment. It was realized by adding the feeling <JOY> to acts, which should be favored and adding <ANXIETY> to acts, which were important, but where the agent should try to avoid a situation. Because this was only a basic implementation, the potential of feelings was not exploited. Further work would, therefore, be to create a concept of how feelings should influence decision-making as a process. For instance, if the system has fear by showing a high value of <ANXIETY>, the planning should have other priorities than with high values of <JOY>. A human that is scared usually do not think very clear, does not reason about all solutions and instead selects a fast and straightforward solution.

All Labeled Images of an Act Loaded at Once

Only the basic concept of using acts in decision-making was developed. Therefore, here is much work to do. To simplify, a limitation of the concept was used. Here, all labels of all template images of an act were loaded from the memory at once. Although only the template image of the intention would be loaded in the primary process, all labels (word presentation mesh) of the act are loaded in the secondary process at the conversion of data structures between the primary and the secondary process. However, only the secondary process information is available and not the primary process information. Regardless, the functions cannot do much with them. A consequence of the current limitation is that complete acts are available at once for the decision-making, simplifying the processing of acts.

To be closer to the theory of SiMA [BDD+14, p. 47], acts should be activated image-by-image based on the internal actions from decision-making. If an act would contain 100 labeled images, then only a few could be active at once. It is an issue of focusing decision-making. For that purpose, an own functional module or service could be defined that only responds to certain internal actions.

Increasing Flexibility in Act Recognition

Succeeding works should also try to make the recognition of the moment more flexible. With the current parameterization of the test environment, the template images of an act almost have to match perfectly the perceived image to be useful for the act. Further, the actions of an act are fixed, and the system will execute that action in the same way as it is described in the act. In a general problem setting, there are not many perfect matches. The system has to be able to correct the actions of erroneous and deviating positions in the environment.

Another issue is that a poorly perceived image match may only depend on the wrong positioning of the agent and not on the poor constellation of objects. For that purpose, self-localization and usage of relational representation of perceived object representatives in the secondary process would be very useful be investigated. Just some extra flexibility in this area would provide an immense robustness gain for the system.

Act Recording

As soon as the act recognition is mature, the next big task would be to let the system create new experiences in the form of acts. The first step would be to extend the short-term working memory of the secondary process with the maximum of history that could be saved in an act. Then the system has

to analyze at each step, where to start saving an act and where to stop. In general, it would be proper to let the system start with an act at the moment a certain action is executed and let it stop the recording of the act as the consequence of the action is visible. Here, an important part would also be to generate or to remove the intention in its current form. Due to the evaluation of a situation with drive wishes and social rules, the act would have a value to the system to reach something or to avoid doing something.

The system could then record a lot of acts and as the database is getting full, it would have to delete the least useful ones. Because the acts would be in direct competition for space in the database, perhaps evolutionary programming [ES08, p. 116] could be applied, store the most useful acts and remove the least useful acts.

Industrial Application in the Project KORE

Within this work, only fundamental research was performed. Until now, there were not many possibilities to directly compare the performance of the SiMA model with another state of the art models. In the project KORE (Cognitive Control Strategy Optimization for Increasing Energy Efficiency in Buildings), this chance is finally given. The purpose of the project is to use a cognitive architecture to generate control strategies for building automation system. Recently, either simple rule-based systems, hard-coded solutions or simple human experience was used to generate these rules. The idea of this project is to let the cognitive architecture test different generated rules in a simulator and from that learn the best solution for each context. The work done here could help to transfer the SiMA model from the cognitive domain to the building automation domain by providing the required functionality. One idea would be to combine case-based reasoning with reinforcement learning. Here, the case-based reasoning would be realized by acts and the reinforcement learning by drives and social rules. Because most cognitive architectures never leave the laboratory, such an application would be the perfect demonstration of the capability of the SiMA model and provide a significant contribution to the state of the art.

New General Software Architecture

If SiMA is supposed to be used in an industrial application or somewhere outside of the current SiMA in World, there has to be a major software refactoring. The purposes of a redesign of the software architecture are the following: (1) to provide extendability; (2) to ensure testability and (3) to improve maintainability. Extendability is necessary as new requirements in research projects often emerge faster than they can be implemented. Testability is given if any module of the system can be independently tested without the need to have the whole target system available. Problems like “a never working software version” will not occur so easily then. Maintainability makes that the code can be maintained and fixed without risking that nothing works afterward. The key to all of this is modularity, i.e. classes with little dependence between modules, but with strong dependencies within a module of classes.

Until now, the architecture has been built as a central system that processes data. There has been no strict independence of classes. A strict separation of modules can be seen in the LIDA Framework [FF12, p. 105]. There, independent codelets do execute the system functionality. Another relatively

new cognitive architecture offers the possibility to implement the architecture as an agent-based, service oriented architecture [CCF13]. Such an implementation would change nothing in the workflow, but an agent-based architecture will be scalable and also possesses the possibility of self-development, which is an emergent feature.

Additionally, it is nothing bad in using common software design patterns in realizing SiMA model functionality. For instance, in SiMA there is the unique concept of defense mechanisms. It cannot be found in any other architecture. However, on software design level, the description of the concept sounds exactly like a typical production system. Defined rules apply to some data structures, and if they apply, they manipulate those data structures in some way.

6.3 Outlook

As mentioned in Chapter 1.2, current technology may not be mature enough to cope with demands from scientific simulations in the areas of psychology [Sun06, pp. 103-433], [SCZ05], society [CGS98] or economics [LTC12], [SMW+15, pp. 1-6]. The result of the fundamental research would bring a significant benefit. Eventually, as SiMA leaves the area of basic research, it could be well suited for other applications like multi-agent-systems. It could be more realistic if they were using simulated individuals with more human-like behaviors. For simulation purposes, therefore, an implementation within the JADE multi-agent framework [13], [BBD+06] like it has been done with the cognitive architecture BDI (Believe Desire Intention) in JADEX [PBL05] should be set as a goal for SiMA. Pure software systems like simulators provide a significant advantage to hardware systems because perception and sensory input issues are not relevant. Further, they can be regarded to be a *closed world* [Kee13, p. 415]. It means that within a simulator, the number of possible states is limited different to reality.

SiMA was created to cope with demands from building automation [DFZB09, p. 2]. It was primarily developed for merging huge quantities of data into symbols and processing them (see Chapter 1.2). In the project ECABA (later KORE) [Zuc15, pp. 1-6], which was briefly described in the previous chapter, the first attempt to get back to the root of the SiMA has been made. A neighboring research area would then be ambient assistant living. There, a building is equipped with an intelligent building monitoring and assisting system. It could help elderly people with disabilities to perform their tasks by themselves [Bru07, p. 39], [YB10]. If a system can recognize the intention of a person, it can react to that in an assisting way. In the area of monitoring, recognition of dangerous situations in due time could save lives.

Finally, another useful category of application is in the area of mobile units. Mobile exploring vehicles could profit from increased perceptive awareness resulting from the increased ability to interpret perceived situations. An exploration vehicle, which could be sent to a foreign planet like Mars, would only have limited contact with Earth due to the line of sight. It could send data and receive tasks during the communication time window [CRFS07, p. 1], [BLM05, p. 2]. It would be critical to recognize autonomously and react to a sand storm in time by noticing a sequence of signs in the environment. In pathfinding, it is used in navigation based on landmarks, where the processing of sequences could be used to recognize the current position and to find the way to the destination.

Although possible applications have been mentioned, it is almost impossible to predict what the main application could be. In several cases, inventions emerge before there is an actual need for them. [Dia02, p. 292] describes this phenomenon. As Thomas Edison invented the phonograph in 1877, the main proposed applications were to create recordings of books, in order to make them accessible to blind people and to record the last words from dying people. The usage for music playing was totally underestimated.

Until now, the benefits for technical sciences and applications were discussed. As psychoanalysis does not cover all concepts needed for a cognitive architecture, especially not low-level concepts [LP73, p. 41], the SiMA project could show the used psychoanalytical model in a new perspective [DPD12, p. 133]. By translating this theory into a technical model, questions could arise, which were not considered before. It could also point to the contradictions and weaknesses in psychoanalysis. The technical model has to be complete to be implementable. Therefore, other scientific disciplines have to be used to complement it.

As a consequence besides of new technical solutions, SiMA would be applicable in complex world simulations like Second Life [14], World of Warcraft [15] or Sims [16]. It would enable the execution of statistical tests in controlled environments, which could be useful for the testing and validation of psychoanalytical concepts [DPD12, p. 124]. For instance, in the contest Bot Prize [17], [Hin09, p. 1], [Hin10, p. 345], autonomous agents are evaluated for human likeness. The properties of the agents are measured and compared to human controlled agents. In that way, it would be possible to test the impact of different personality parameters in SiMA measured on the behavior of the agent to see which parameter settings are the most human-like.

Literature

- [ABB⁺04] J. R. Anderson, D. Bothell, M. D. Byrne, S. Douglass, C. Lebiere, and Y. Qin. An integrated theory of the mind. *Psychological Review*, 111(4):1036–1060, 2004.
- [AL98] John R. Anderson and Christian Lebiere. *The Atomic Components of Thought*. Lawrence Erlbaum Associates, 1998.
- [And83] J. R. Anderson. A spreading activation theory of memory. *Journal of verbal learning and verbal behavior*, vol. 22:pp. 261–295, 1983.
- [AS68] Richard C. Atkinson and Richard M. Shiffrin. *The psychology of learning and motivation*, volume 2, chapter Human memory: A proposed system and its control processes, pages 89–195. New York: Academic Press, 1968.
- [Asi50] Isaac Asimov. *I, Robot*. Doubleday & Company, New York, 1950.
- [BBD⁺06] Rafael Bordini, Lars Braubach, Mehdi Dastani, Amal El Fallah Seghrouchni, Jorge Gomez-Sanz, Joao Leite, Gregory O’Hare, Alexander Pokahr, and Alessandro Ricci. A survey of programming languages and platforms for multi-agent systems. *Informatica 30*, pages 33–44, 2006.
- [BDD⁺14] C. Brandstätter, D. Dietrich, K. Doblhammer, M. Fittner, G. Fodor, F. Gelbard, M. Huber, M. Jakubec, S. Kollmann, D. Kowarik, S. Schaaf, A. Wendt, and R. Widholm. Naturwissenschaftliches, psychoanalytisches modell der psyché für simulation und emulation. Technical report, Institute of Computer Technology, Vienna University of Technology, 2014. http://publik.tuwien.ac.at/files/PubDat_234013.pdf.
- [BDK⁺04] Elisabeth Brainin, Dietmar Dietrich, Wolfgang Kastner, Peter Palensky, and Charlotte Rösener. Neuro-bionic architecture of automation systems : Obstacles and challenges. *Proceedings of 2004 IEEE AFRICON, 7th Africon conference in Africa, Technology Innovation*, 2:1219–1222, 2004.
- [BEH⁺12] Ramón Briegel, Andreas Ernst, Sascha Holzhauser, Daniel Klemm, Friedrich Krebs, and Aldo Martínez Piñáñez. Social-ecological modelling with lara: A psychologically well-founded lightweight agent architecture. In *Paper eingereicht für den International Congress on Environmental Modelling and Software, Leipzig*, 2012.
- [BGSW13] Dietmar Bruckner, Friedrich Gelbard, Samer Schaaf, and Alexander Wendt. Validation of cognitive architectures by use cases: Exemplified with the psychoanalytically-inspired ars model implementation. In *Industrial Electronics (ISIE), 2013 IEEE International Symposium on*, pages 1–6. IEEE, 2013.
- [BLM05] Jeffrey J. Biesiadeckia, Chris Leger, and Mark W. Maimone. Tradeoffs between directed and autonomous driving on the mars exploration rovers. In *International Symposium of Robotics Research, San Francisco, California, USA*, San Francisco, California, USA, 2005.
- [Bot13] Dan Bothell. Act-r 6.0 reference manual. Technical report, Carnegie Mellon University, Department of Psychology, Baker Hall 342c, Pittsburgh, Pennsylvania 15213, USA, 2013. <http://act-r.psy.cmu.edu/actr6/reference-manual.pdf>.
- [Bra87] Michael Bratman. *Intention, Plans, and Practical Reason*. Harvard University Press, 1987.

- [BRB07] Daniel Bahls and Thomas Roth-Berghofer. Explanation support for the case-based reasoning tool mycbr. In *AAAI*, volume 7, pages 1844–1845, 2007.
- [Bre02] Cynthia Breazeal. *Designing Sociable Robots*. MIT Press, Cambridge, MA, USA, 2002.
- [Bru07] Dietmar Bruckner. *Probabilistic Models in Building Automation: Recognizing Scenarios with Statistical Methods*. PhD thesis, Vienna University of Technology, Institute of Computer Technology, 2007.
- [Byr03] M. D. Byrne. *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications*, chapter Cognitive architecture, pages 97–117. Erlbaum, 2003.
- [CCF13] James A Crowder, John N Carbone, and Shelli A Friess. *Artificial Cognition Architectures*. Springer, 2013.
- [CGS98] Rosaria Conte, Nigel Gilbert, and Jaime Simão Sichman. Mas and social simulation: A suitable commitment. In *Multi-Agent Systems and Agent-Based Simulation - First International Workshop, MABS '98, Paris, France, July 4-6, 1998. Proceedings*, volume Vol. 1534, pages pp 1–9. Springer Berlin Heidelberg, 1998. DOI: 10.1007/10692956_1.
- [CKN⁺07] Dongkyu Choi, Tolga Könik, Negin Nejati, Chunki Park, and Pat Langley. A believable agent for first-person shooter games. In *AIIDE*, pages 71–73, 2007.
- [Cre97] F. Crestani. Application of spreading activation techniques in information retrieval. in *Artificial Intelligence Review*, vol. 11:pp. 453–482, 1997.
- [CRFS07] Joseph Carsten, Arturo Rankin, Dave Ferguson, and Anthony Stentz. Global path planning on board the mars exploration rovers. In *Proceedings of 2007 IEEE Aerospace Conference*. IEEE Press, 2007.
- [CS88] Kenneth M. Colby and Robert J. Stoller. *Cognitive Science and Psychoanalysis*. The Analytic Press, 1988.
- [Dam99] Antonio Damasio. Commentary on panksepp. *Neuropsychanalysis*, 1:38–39, 1999.
- [Dam00] Antonio Damasio. *The Feeling of What Happens: Body, Emotion and the Making of Consciousness*. Vintage, new ed edition, October 2000.
- [Daw01] Richard Dawkins. *Gipfel des Unwahrscheinlichen*. Rowohlt Taschenbuch Verlag, 2001.
- [DBD⁺15] Dietmar Dietrich, Christian Brandstätter, Klaus Doblhammer, Martin Fittner, Georg Fodor, Friedrich Gelbard, Matthias Huber, Matthias Jakubec, Stefan Kollmann, Daniela Kowarik, Samer Schaat, Alexander Wendt, and Roman Widholm. Natural scientific, psychoanalytical model of the psyche for simulation and emulation. Technical report, E384 - Institut für Computertechnik; Technische Universität Wien, 2015.
- [DBM⁺10] Dietmar Dietrich, Dietmar Bruckner, Brit Müller, Gerhard Zucker, and Friederich Kupzog. Abstraction levels for developing a model of an intelligent decision unit. *Proceedings of the 8th IEEE INDIN*, pages 79–85, 2010.
- [Deu11] Tobias Deutsch. *Human Bionically Inspired Autonomous Agents—The Framework Implementation ARSi11 of the Psychoanalytical Entity Id Applied to Embodied Agents*. PhD thesis, Vienna University of Technology, 2011.
- [DFKU09] Dietmar Dietrich, Georg Fodor, Wolfgang Kastner, and Mihaela Ulieru. Considering a technical realization of a neuropsychanalytical model of the mind – a theoretical framework. In Dietmar Dietrich, Georg Fodor, Gerhard Zucker, and Dietmar Bruckner, editors, *Simulating the Mind – A Technical Neuropsychanalytical Approach*, pages 99 – 115. Springer, Wien, 1 edition, 2009. invited contribution for the 1st ENF - Emulating the Mind, 2007, Vienna.
- [DFZB09] Dietmar Dietrich, Georg Fodor, Gerhard Zucker, and Dietmar Bruckner, editors. *Simulating the Mind - A Technical Neuropsychanalytical Approach*. Springer, Wien, 2009.

- [Dia02] Jared Diamond. *Arm und Reich, die Schicksale menschlicher Gesellschaften*. Fisher Taschenbuch Verlag, Frankfurt am Main, 4th edition edition, 2002. English title: “Guns, Germs, and Steel: The Fates of Human Societies”.
- [Die00] Dietmar Dietrich. Evolution potentials for fieldbus systems. In *Factory Communication Systems, 2000. Proceedings. 2000 IEEE International Workshop on*, volume 1, pages 145–146, 2000. Invited Talk.
- [Dig12] Frank Dignum. Agents for serious gaming-challenges and opportunities. In *ICAART (1)*, page 11, 2012.
- [DKM+04] Dietmar Dietrich, Wolfgang Kastner, T. Maly, Charlotte Roesener, Gerhard Russ, and H. Schweinzer. Situation Modeling. *Factory Communication Systems, 2004. Proceedings. 2004 IEEE International Workshop on*, pages 93– 102, 2004.
- [Dob15] Klaus Doblhammer. Das selbst eines roboters. Technical report, Institute of Computer Technology, Vienna University of Technology, 2015. http://sima.ict.tuwien.ac.at/new/wp-content/uploads/2015/02/Das-Selbst-eines-Roboters-abschlussbericht_final_150115.pdf.
- [DPD12] Dietmar Dietrich, Peter Palensky, and Dorothee Dietrich. Psychoanalyse und computertechnik eine win-win-situation? *psychosozial*, 35. Jg. (2012) Heft I (Nr. 127):pp. 123–135, 2012.
- [DS00] Dietmar Dietrich and Thilo Sauter. Evolution potentials for fieldbus systems. In *Proceedings of 4th IEEE Int. Workshop on Factory Communication Systems*, pages 343–350, 2000.
- [DSBD13] Dietmar Dietrich, Samer Schaat, Dietmar Bruckner, and Klaus Doblhammer. The current state of psychoanalytically-inspired ai - a holistic and unitary model of human psychic processes. In *in proceedings of IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society*, pages pp. 6664–6669, 2013.
- [DWvDH09] Frank Dignum, Joost Westra, Willem A van Doesburg, and Maaïke Harbers. Games and agents: Designing intelligent gameplay. *International Journal of Computer Games Technology*, 2009, 2009.
- [DZ08] Dietmar Dietrich and Gerhard Zucker. New approach for controlling complex processes. an introduction to the 5th generation of AI. *Human System Interactions, 2008 Conference on*, pages 12–17, 2008. invited keynote speech.
- [DZBM09] Dietmar Dietrich, Gerhard Zucker, Dietmar Bruckner, and Brit Müller. Bionic model for control platforms. In *Proceedings of the 6th International Workshop on Frontiers of Information Technology - FIT*, Abbottabad, Pakistan, 2009.
- [ES08] AE Eiben and JE Smith. Introduction to evolutionary computing (natural computing series). 2008.
- [FF12] Usef Faghihi and Stan Franklin. The lida model as a foundational architecture for agi. In *Theoretical Foundations of Artificial General Intelligence*, pages 103–121. Springer, 2012.
- [FR06] Stan Franklin and Uma Ramamurthy. Motivations, values and emotions: 3 sides of the same coin. In *Proceedings of the Sixth International Workshop on Epigenetic Robotics, Paris, France, September 2006, Lund University Cognitive Studies*, number 128, pages 41–48, 2006.
- [Fre91] Sigmund Freud. *Zur Auffassung der Aphasien*. Fischer Taschenbuch, 1891.
- [Fue03] Clara Tamarit Fuertes. *Automation System Perception - First Step towards Perceptive Awareness*. PhD thesis, Faculty of Electrical Engineering and Information Technology, Vienna University of Technology, 2003.
- [FZP11] Tarik Ferhatbegovic, Gerhard Zucker, and Peter Palensky. Model based predictive control for a solar-thermal system. In *AFRICON, 2011*, pages 1–6. IEEE, 2011.
- [Gar86] E. E. Garcia. Psychoanalysis: Science or fiction? *Jefferson Journal of Psychiatry*, Vol. 4(Issue 1):Issue 1, Article 4, 1986.

- [GB10] T. Grechenig and M. Bernhart. *Softwaretechnik: Mit Fallbeispielen aus realen Entwicklungsprojekten*. IT Informatik. Pearson Studium, 2010.
- [GL10] Fernand Gobet and Peter C.R. Lane. The chrest architecture of cognition: The role of perception in general intelligence. In *E. Baum, M. Hutter, & E. Kitzelmann (Eds.), Proceedings of the third conference on artificial general intelligence*, pages 7 – 12, Amsterdam, 2010. Atlantis Press.
- [GTC⁺10] Sebastian Gottifredi, Mariano Tucaty, Daniel Corbatta, Alejandro J. Garcia, and Guillermo R. Simari. A bdi architecture for high level robot deliberation. *Inteligencia Artificial*, 46:74–83, 2010.
- [Hin09] Philip Hingston. The 2k botprize. In *IEEE Symposium on Computational Intelligence and Games, 2009. CIG 2009.*, Milano, Italy, 2009.
- [Hin10] Philip Hingston. A new design for a turing test for bots. In *Symposium on Computational Intelligence and Games (CIG), 2010 IEEE*, pages pp. 345 – 350, 2010.
- [Hin14] Isabella Hinterleitner. *Generation of Inner and Outer Speech by Means of Situational Context*. PhD thesis, TU Vienna, Institute of Computer Technology, 2014.
- [HW12] Judith Huber-Wendt. *Von Emil bis Ronja - Eine inhaltliche Gegenüberstellung der Kinderbuch-Verfilmungen von Erich Kästner und Astrid Lindgren*. PhD thesis, Universität Wien, 2012.
- [Kak11] Michio Kaku. *Physics of the Future: How Science Will Shape Human Destiny and Our Daily Lives by the Year 2100*, volume 1. Doubleday, 2011. ISBN 978-0-385-53080-4.
- [KDJ03] William G Kennedy and Kenneth A De Jong. Characteristics of long-term learning in soar and its application to the utility problem. In *ICML*, pages 337–344, 2003.
- [KE06] Jeffrey L Krichmar and Gerald M Edelman. Principles underlying the construction of brain-based devices. In *Proceedings of AISB*, volume 6, pages 37–42, 2006.
- [Kee13] C Maria Keet. Closed world assumption. *Encyclopedia of Systems Biology*, pages 415–415, 2013.
- [Lan05a] P. Langley. An adaptive architecture for physical agents. In *in Proceedings of IEEE/WIC/ACM International Conference of Intelligent Agent technologies, Compiègne, France*, pages pp. 18–25, Compiègne, France, 2005.
- [Lan05b] Pat Langley. An adaptive architecture for physical agents. In *Web Intelligence, 2005. Proceedings. The 2005 IEEE/WIC/ACM International Conference on*, pages 18–25. IEEE, 2005.
- [Lan06] Pat Langley. Cognitive architectures and general intelligent systems. *AI magazine*, 27 No.2:33–44, 2006.
- [Lan10] Roland Lang. *A Decision Unit for Autonomous Agents Based on the Theory of Psychoanalysis*. PhD thesis, Vienna University of Technology, 2010.
- [LCC⁺11] John E. Laird, Clare Bates Congdon, Karen J. Coulter, Nate Derbinsky, and Joseph Xu. The soar user’s manual version 9.3.1. Technical report, Computer Science and Engineering Department, University of Michigan, 2011.
- [LCT11] Pat Langley, Dongkyu Choi, and Nishant Trivedi. Icarus user’s manual. Technical report, Institute for the Study of Learning and Expertise 2164 Staunton Court, Palo Alto, CA 94305 USA, 2011.
- [LKMX12] John E Laird, Keegan R Kinkade, Shiwali Mohan, and Joseph Z Xu. Cognitive robotics using the soar cognitive architecture. In *Proc. of the 8th Int. Conf. on Cognitive Robotics*, 2012.
- [LL11] Justin Li and John Laird. Preliminary evaluation of long-term memories for fulfilling delayed intentions. In *Papers from the 2011 AAAI Fall Symposium Series: Advances in Cognitive Systems. Arlington, VA, 2011*, 2011.

- [LLR09] Pat Langley, John E. Laird, and Seth Rogers. Cognitive architectures: Research issues and challenges. *Cognitive Systems Research*, 10(2):141–160, 2009.
- [LP73] J. Laplanche and J. B. Pontalis. *The Language of Psycho-Analysis: Translated by Donald Nicholson-Smith*. The Hogarth Press and the Institute of Psycho-Analysis, 1973.
- [LTC12] Ribin Lye, James Peng Lung Tan, and Siew Ann Cheong. Understanding agent-based models of financial markets: A bottom-up approach based on order parameters and phase diagrams. *Physica A: Statistical Mechanics and its Applications*, Volume 391, Issue 22, 15 November 2012:pp. 5521–5531, 2012. <http://dx.doi.org/10.1016/j.physa.2012.06.014>.
- [McC87] John McCarthy. Generality in artificial intelligence. *Commun. ACM*, 30(12):1030–1035, December 1987.
- [MLA+04] Brian Magerko, J Laird, Mazin Assanie, Alex Kerfoot, and Devvan Stokes. Ai characters and directors for interactive computer games. *Ann Arbor*, 1001(48):109–2110, 2004.
- [Muñ10] M A Muñoz. The identity of psychoanalysis and psychoanalysts. *The American Journal of Psychoanalysis*, 70:94–99, 2010.
- [Muc13] Clemens Muchitsch. *Human-like Perception for Psychoanalytically Inspired Reasoning Units*. PhD thesis, Institute for Computer Technology, Vienna University of Technology, 2013.
- [MWB+11] Clemens Muchitsch, Alexander Wendt, Dietmar Bruckner, Jana Machajdik, and Klaus Doblhammer. Perceptual prediction for bionically inspired autonomous agents. In *Proceedings of the 10th IEEE AFRICON*, pages 1–6, 2011.
- [MWD+11] Clemens Muchitsch, Alexander Wendt, Klaus Doblhammer, Dietmar Bruckner, and Jana Machajdik. Perceptual prediction for bionically inspired autonomous agents. In *AFRICON, 2011*, pages 1–6. IEEE, 2011.
- [New94] Allen Newell. *Unified Theories of Cognition*. Harvard Univ Press, 1994.
- [New03] Monty Newborn. Computer chess. In *Encyclopedia of Computer Science*, pages 336–339. John Wiley and Sons Ltd., Chichester, UK, 2003.
- [Nil94] Nils Nilsson. Teleo-reactive programs for agent control. *arXiv preprint cs/9401101*, 1994.
- [NL07] Andrew M Nuxoll and John E Laird. Extending cognitive architecture with episodic memory. In *Proceedings of the National Conference on Artificial Intelligence*, volume 22, page 1560. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.
- [NS76] Allen Newell and Herbert A. Simon. Computer science as empirical inquiry: symbols and search. *Commun. ACM*, 19(3):113–126, March 1976.
- [PBL05] Alexander Pokahr, Lars Braubach, and Winfried Lamersdorf. Jadex: A bdi reasoning engine. In *Multi-Agent Programming Languages, Platforms and Applications*. Springer US, 2005.
- [PDHP07] Gerhard Pratl, Dietmar Dietrich, Gerhard P. Hancke, and Walter T. Penzhorn. A new model for autonomous, networked control systems. *IEEE Transactions on Industrial Informatics*, 3(1):21–32, 2007.
- [PP05] G. Pratl and P. Palensky. Project ars - the next step towards an intelligent environment. *Proceedings of the IEE International Workshop on Intelligent Environments*, pages 55–62, 2005.
- [PPDB05] Gerhard Pratl, Walter T. Penzhorn, Dietmar Dietrich, and Wolfgang Burgstaller. Perceptive awareness in building automation. *IEEE 3rd International Conference on Computational Cybernetics*, pages 259–264, 2005.
- [Pra06] Gerhard Pratl. *Processing and Symbolization of Ambient Sensor Data*. PhD thesis, Faculty of Electrical Engineering and Information Technology, Vienna University of Technology, 2006.

- [PZ11] A Perner and H Zeilinger. Action primitives for bionics inspired action planning system: Abstraction layers for action planning based on psychoanalytical concepts. In *Industrial Informatics (INDIN), 2011 9th IEEE International Conference on*, pages 63–68. IEEE, 2011.
- [RBDF06] Uma Ramamurthy, Bernard J. Baars, Sidney K. D’Mello, and Stan Franklin. Lida: A working model of cognition. *Proceedings of the 7th International Conference on Cognitive Modeling*, pages 244–249, 2006.
- [RBF06] Uma Ramamurthy, Bernard J Baars, and Stan Franklin. Lida: A working model of cognition. 2006. CogPrints.
- [RG95] Anand S. Rao and Michael P. Georgeff. Bdi agents: From theory to practice. In *Proceedings of the First International Conference on Multiagent Systems (ICMAS’95)*, 1995.
- [RHBP04] Charlotte Roesener, Harald Hareter, Wolfgang Burgstaller, and Gerhard Pratl. Environment simulation for scenario perception models. *Factory Communication Systems, 2004. Proceedings. 2004 IEEE International Workshop on*, pages 349 – 352, 2004.
- [RLSL09] Fernand Gobet Richard Ll. Smith and Peter C. R. Lane. Checking chess checks with chunks: A model of simple check detection. In *Proceedings of the Ninth International Conference on Cognitive Modelling*, 2009.
- [Roe07] Charlotte Roesener. *Adaptive Behavior Arbitration for Mobile Service Robots in Building Automation*. PhD thesis, Vienna University of Technology, Institute of Computer Technology, 2007.
- [Ros11] J. Rose. *Mapping Psychic Reality - Triangulation, Communication and Insight, Psychoanalytic Ideas*. Karnac Books Ltd, UK, 2011.
- [Rus03] Gerhard Russ. *Situation-dependent Behavior in Building Automation*. PhD thesis, Vienna University of Technology, Institute of Computer Technology, 2003.
- [RW13] Michael M Richter and Rosina O Weber. Case-based reasoning. *A Textbook*, 2013.
- [SB98] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 1998.
- [SB05] Murray Shanahan and Bernard Baars. Applying global workspace theory to the frame problem. *Cognition*, 98(2):157–176, 2005.
- [SC05] Aaron Sloman and Ron Chrisley. More things than are dreamt of in your biology: Information-processing in biologically inspired robots. *Cognitive Systems Research*, 6(2):145–174, 2005.
- [SCZ05] Ron Sun, Andrew L. Coward, and Michael J. Zenzen. On levels of cognitive modeling. *Philosophical Psychology*, 18(5):613–637, October 2005.
- [SDK01] T. Sauter, D. Dietrich, and W. Kastner. Eib. installation bus system. Erlangen, 2001. Publicis Corporate Publishing.
- [SDW⁺13] Samer Schaat, Klaus Doblhammer, Alexander Wendt, Friedrich Gelbard, Lukas Herret, and Dietmar Bruckner. A psychoanalytically-inspired motivational and emotional system for autonomous agents. in *proceedings of IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society*, pages pp. 6647–6651, 2013. Vienna, Austria.
- [SK09] L. R. Squire and E. R. Kandel. *Gedächtnis – die Natur des Erinnerns; original: Memory – From Mind to Molecules*. German version: 2nd Edition, Spektrum Akademischer Verlag, Germany; Original version: Roberts and Company Publishers, USA, 2009.
- [SKZ⁺15] Samer Schaat, Stefan Kollmann, Olga Zhukova, Dietmar Dietrich, and Klaus Doblhammer. Examination of foundational agi-agents in artificial-life simulations. In *proceedings of EAPCogSci 2015, EuroAsianPacific Joint Conference on Cognitive Science, Torino, Italy, September 25-27, 2015*, pages 334–340. IEEE, 2015.

- [SLG08] Richard LI Smith, Peter CR Lane, and Fernand Gobet. Modelling the relationship between visual short-term memory capacity and recall ability. In *Computer Modeling and Simulation, 2008. EMS'08. Second UKSIM European Symposium on*, pages 99–104. IEEE, 2008.
- [Slo99] Aaron Sloman. What sort of architecture us required for a human-like agent? In Michael Wooldridge and Anand Rao, editors, *In Foundations of Rational Agency*, pages 35–52. Kluwer Academic Publishers, 1999.
- [Slo00] Aaron Sloman. Models of models of mind. *Proceedings of symposium on how to design a functioning mind*, pages 1–9, 2000.
- [SM03] Push Singh and Marvin Minsky. An architecture for combining ways to think. In *Integration of Knowledge Intensive Multi-Agent Systems, 2003. International Conference on*, pages 669–674. IEEE, 2003.
- [SMW⁺15] Samer Schaat, Aleksandar Miladinovic, Stefan Wilker, Stefan Kollmann, Stephan Dickert, Erdem Geveze, and Verena Gruber. Émotion in consumer simulations for the development and testing of recommendations for marketing strategies. In *Proceedings of the 3rd Workshop on Emotions and Personality in Personalized Systems 2015*, pages 25–32. ACM, 2015.
- [ST02] Mark Solms and Oliver Turnbull. *The Brain and the Inner World: An Introduction to the Neuroscience of Subjective Experience*. Karnac/Other Press, Cathy Miller Foreign Rights Agency, London, England, 2002.
- [Sun06] Ron Sun, editor. *Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation*. Cambridge University Press, 2006.
- [SVP10] Estefana Serral, Pedro Valderas, and Vicente Pelechano. Supporting runtime system evolution to adapt to user behaviour. In *Advanced Information Systems Engineering*, pages 378–392. Springer, 2010.
- [SWJ⁺14] Samer Schaat, Alexander Wendt, Matthias Jakubec, Friedrich Gelbard, Lukas Herret, and Dietmar Dietrich. Ars: An agi agent architecture. In *Artificial General Intelligence*, pages 155–164. Springer, 2014.
- [SWK⁺15] S. Schaat, A. Wendt, S. Kollmann, F. Gelbard, and M. Jakubec. Interdisciplinary development and evaluation of cognitive architectures exemplified with the sima approach. In *proceedings of EAPCogSci 2015, EuroAsianPacific Joint Conference on Cognitive Science, Torino, Italy, September 25-27, 2015*, pages pp. 215–220, 2015.
- [SWY11] Zhongzhi Shi, Xiaofeng Wang, and Jinpeng Yue. Cognitive cycle in mind model cam. *Int. J. Intelligence Science*, 1(2):25–34, 2011.
- [TCM03] Douglas G. Turnbull, C. M. Chewar, and D. Scott McCrickard. Are cognitive architectures mature enough to evaluate notification systems? In *2003 International Conference on Software Engineering Research and Practice (SERP 2003)*, Las Vegas NV, 2003.
- [THH⁺12] J. Gregory Trafton, Laura M. Hiatt, Anthony M. Harrison, Franklin P. Tamborello, Sangeet S. Khemlani, and Alan C. Schultz. Act-r/e: An embodied cognitive architecture for human-robot interaction. *Journal of Human-Robot Interaction*, 1(1):78–95, 2012.
- [TJJ⁺95] Milind Tambe, W Lewis Johnson, Randolph M Jones, Frank Koss, John E Laird, Paul S Rosenbloom, and Karl Schwamb. Intelligent agents for interactive simulation environments. *AI magazine*, 16(1):15, 1995.
- [VBP09] Rosemarie Velik, Dietmar Bruckner, and Peter Palensky. A bionic approach for high-efficiency sensor data processing in building automation. In *Proceedings of the 35th IEEE IECON*, 2009.
- [Vel08] Rosemarie Velik. *A Bionic Model for Human-like Machine Perception*. PhD thesis, Vienna University of Technology, Institute of Computer Technology, 2008.

- [Vel13] Rosemarie Velik. Cognitive architectures as building energy management system for future renewable energy scenarios—a work in progress report. *IJSEI International Journal on Science and Engineering Investigations*, 2(17):68–72, 2013.
- [VMS07] D. Vernon, G. Metta, and G. Sandini. A survey of artificial cognitive systems: Implications for the autonomous development of mental capabilities in computational agents. *IEEE transactions on evolutionary computation*, vol. 11, no. 2:pp. 151–180, 2007. doi: 10.1109/TEVC.2006.890274.
- [WDB12] Alexander Wendt, Benjamin Dönz, and Dietmar Bruckner. Data access through a dynamic data model. In *Proceedings of 4th International Conference on Agents and Artificial Intelligence (ICAART) 2012*, 2012.
- [Wen04] Juyang Weng. A theory of developmental architecture. In *Proceedings of the 3rd International Conference on Development and Learning (ICDL 2004)*, 2004.
- [WG06] Pei Wang and Ben Goertzel. Introduction: Aspects of artificial general intelligence. *Proceeding of the 2007 conference on Advances in Artificial General Intelligence: Concepts, Architectures and Algorithms: Proceedings of the AGI Workshop 2006*, pages 1–16, 2006.
- [WGF⁺15] A. Wendt, F. Gelbard, M. Fittner, S. Schaaf, M. Jakubec, C. Brandstätter, and Stefan Kollmann. Decision-making in the cognitive architecture sima. In *Proceedings of the Conference on Technologies and Applications of Artificial Intelligence (TAAI 2015)*, pages 330–335. IEEE, 2015.
- [WH97] A. Walter and H. Hellmann. Dezentrale automatisierung mit iec 1131 und canopen. In *Feldbustchnik in Forschung, Entwicklung und Anwendung*, pages pp. 325–332. Springer Verlag Wien, 1997.
- [WHB⁺00] Jan Wendler, Markus Hannebauer, Hans-Dieter Burkhard, Helmut Myritz, Gerd Sander, and Thomas Meinert. Bdi design principles and cooperative implementation in robocup. In *RoboCup-99, LNAI 1856*, pages pp. 531–541. Springer-Verlag Berlin Heidelberg, 2000.
- [Wil12] Edward O Wilson. *The social conquest of earth*. WW Norton & Company, 2012.
- [Win09] Samuel Wintermute. An overview of spatial processing in soar/svs. Manual CCA-TR-2009-01, Center for Cognitive Architecture, University of Michigan, 2260 Hayward St, Ann Arbor, Michigan 48109-2121, USA, June 2009. TECHNICAL REPORT.
- [WL09] Samuel Wintermute and John E Laird. Imagery as compensation for an imperfect abstract problem representation. In *Proceedings of the 31st Annual Conference of the Cognitive Science Society*, 2009.
- [WSG⁺13] Alexander Wendt, Samer Schaaf, Friedrich Gelbard, Clemens Muchitsch, and Dietmar Bruckner. Usage of spreading activation for content retrieval in an autonomous agent. in *proceedings of IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society*, pages pp. 6670–6675, 2013.
- [YB10] GuoQing Yin and Dietmar Bruckner. Data analyzing and daily activity learning with hidden markov model. In *Proceedings of the ICCASM*, pages 380–384, 2010.
- [ZBD11] Gerhard Zucker, Dietmar Bruckner, and Dietmar Dietrich. Processing data in complex communication systems. In *The Industrial Electronics Handbook: Industrial Communication Systems*, pages p. 67–1. CRC Press, Taylor & Francis Group, LLC, Boca Raton, London, 2011.
- [Zei10] Heimo Zeilinger. *Bionically Inspired Information Representation for Embodied Software Agents*. PhD thesis, Vienna University of Technology, 2010.
- [Zuc15] Wendt A. Habib U. Schaaf S. Sifara L. C. Blöchle M. Zucker, G. Building energy management and data analytics. In *5th Symposium on Communications for Energy systems in Vienna, Austria 2015*, 2015.

Internet/Film References

- [1] Institute of Computer Technology, Insite website, <http://www.ict.tuwien.ac.at>, accessed 2012-12-14
- [2] Simulation of the Mental Apparatus & Applications, Project website, <http://sima.ict.tuwien.ac.at>, accessed 2015-05-15
- [3] Interactive Dota 2 Map, devilesk.com, admin@devilesk.com, <http://devilesk.com/dota2/apps/interactivemap/>, accessed 2015-05-18
- [4] A Better Path to Enterprise Architectures, Microsoft Development Network, <http://msdn.microsoft.com/en-us/library/aa479371.aspx>, accessed on 2014-07-22
- [5] Pippi Langstrumpf, Regie: Hellbom, O., Drehbuch: Lindgren, A., Svensk Filmindustri, 1970. Version DVD, Universum Film, Germany, 2000
- [6] Aladdin, Regie: Musker, J, Clements, R., Drehbuch: Musker, J., Clements, R., Elliott, T., Rossio, T., Walt Disney Pixar, USA, 1992
- [7] MASON, <http://cs.gmu.edu/~eclab/projects/mason/>, accessed on 2014-11-28
- [8] Resource Description Framework (RDF), official website, <http://www.w3.org/RDF>, accessed on 2013-12-16
- [9] OWL Web Ontology Language - Semantics and Abstract Syntax, official website, <http://www.w3.org/TR/owl-semantics/>, accessed on 2015-01-25
- [10] Die besseren Menschen, Halder, S, ORF.at, <http://news.orf.at/stories/2153466/2153467/>, published 2012-12-14, accessed 2012-12-14
- [11] RDF Vocabulary Description Language 1.0: RDF Schema, official website, <http://www.w3.org/TR/rdf-schema/>, accessed on 2013-12-16
- [12] Defining N-ary Relations on the Semantic Web, <http://www.w3.org/TR/swbp-n-aryRelations/#useCase3>, accessed on 2013-12-16
- [13] Java Agent DEvelopment Framework, official website, <http://jade.tilab.com>, accessed 2013-05-10
- [14] Second life, official website, <http://secondlife.com>, accessed on 2013-05-10
- [15] World of Warcraft, official website, <http://us.blizzard.com/en-us/games/wow/>, accessed on 2013-05-10
- [16] The Sims, official website, http://thesims.com/en_US/home, accessed on 2013-05-10
- [17] The 2K BotPrize, official website, <http://botprize.org>, accessed on 2013-05-10

Curriculum Vitae of Alexander Wendt

Personal Information

Date of Birth: 2. July 1981 in Åhus, Sweden

Contact: TU Wien, Institute of Computer Technology, Gusshausstrasse 27-29/E384, A-1040 Vienna

Tel: +436509227277

Email: alexander.wendt@tuwien.ac.at

Education

2010-2016 PhD in the area of “Cognitive Architectures” at the TU Wien, Institute of Computer Technology, title “Experience-Based Decision-Making in a Cognitive Architecture”

2002-2007 Study of “Technical Physics”, TU Wien, 2nd and 3rd study section examined with excellence, academic Degree: Dipl.-Ing.

2001-2002 Study of “Technical Physics and Electrical Engineering”, University of Linköping, Sweden

Professional Experience

Since 2010 Project assistant with the roles of software developer, system designer and project manager, TU Wien, Vienna

2010-2011 Project assistant in research project GÖPL, Frequentis AG, Vienna

2007-2010 Safety Engineer, Frequentis AG, Vienna

Research Projects

Since 2015 KORE, implementation of a cognitive architecture as a building management application, system- and software design; role: senior software developer

Since 2016 Ingrid, development of a customer energy system to be placed between the grid operator and end points in a smart grid, software design; role: senior software developer

2014-2015 ECABA, introduction of a cognitive architecture in the area of building automation, system- and software design; role: system designer

2011-2015 GRID2020 Demonstrationssystem Intelligentes Niederspannungsnetz, software development for a low-voltage-grid test facility and implementation of algorithms for tapchangers and automatic switch state detection, system design and software development of server, clients, communication; roles: project manager, software developer

- 2012-2013 Smart Grids Modellregion Salzburg – Building2Grid, realization of demand side management functionality in residences, software development of server; role: software developer
- 2010-2012 GÖPL – Gemeinsames Öffentlich-Private Lagebild, realisation of a prototype decision support system, development of a relational data model and an ontology-based data model; roles: database modeler, software developer
- 2010-2016 SIMA – Simulation of the Mental Apparatus & Applications, realization of a cognitive architecture based on a model of the human mind, model development, system design and software development, roles: system designer, software developer

Awards/Organization

- 2016 ERF 2016 (European Robotics Forum), responsible for the group handling cognitive architectures in the SPARC Workshop on cognitive robotics
- 2013 IECON 2013 (IEEE conference on Industrial Electronics), special session chair “cognitive architectures and multi-agent systems” and “Smart and Universal Grids”
- 2011 Participation in the Vienna University of Technology High Potential Program TUtheTOP 2011/2012

Publications

Alexander Wendt has written 28 publications in the fields of artificial intelligence, cognitive architectures, smart grids, and software design.

Declaration

Hereby, I declare that this work has been completed without illegal aid from other parties and without usage of other than the mentioned resources. Data and concepts, which are indirectly assumed, are labeled with the declaration of the source.

This work has not been used in examination procedures at home or abroad neither in this nor in similar form.

Vienna, 13.08.2016

Alexander Wendt

Appendix A – Detailed Analysis of State of the Art

In order to extract relevant information for the state of the art, each of the selected cognitive architectures were analyzed in detail according to the evaluation criteria of a cognitive architecture from Chapter 2.1.2.

SOAR – State Operator Apply Result

The SOAR (State, Operator Apply Result) architecture is a implementation of the two hypotheses of classical artificial intelligence by Newell and Simon [NS76, pp. 114-120], which are described in Chapter 2.1.2. It is characterized by a small core architecture, where the functionality is defined in production rules. The reason why this prominent architecture is analyzed here, is to find out the differences and similarities between this approach and SiMA as well as to highlight what is missing, in order to provide human-like behavior. SOAR provides primarily agent functionality, which may or may not be inspired by a human way of thinking [New94, p. 309]. As it does not model the human mind, it is not dependent on any psychological theory and therefore, the question about the usage of an axiomatic-based model of the human mind is not applicable. The basic process of SOAR is to select and apply an *operator* to a *state* [LCC+11, p. 5], where a state is the internal representation of the current situation. An operator modifies that state.

SOAR uses a short-term memory, called a *working memory* to keep the current situation (beliefs). The beliefs are called *working memory elements* [LCC+11, p. 13]. A working memory element is a triple of an identifier of an object, an attribute and a value. Each object is defined by its attributes and not by its identifier. If the value of an object is an identifier of another object, those objects are linked to each other. All objects of the working memory must be linked to a state [LCC+11, p. 14].

In the first phase of Figure 2.1, Activate beliefs, basic beliefs are provided by an external system to the working memory [LCC+11, p. 21]. Such a possibility is to use the add-on SOAR SVS, which is connected to the interface of the working memory in Figure 0.1 [Win09, p. 3]. While the core architecture is completely located in functional layer 3 of Figure 0.1, SOAR SVS extends the standard SOAR architecture with spatial reasoning and visual imaginary, which is used to transform visual data of cameras into symbols as well as imaginary (described later on) on sensor level. Therefore, this add-on is additionally located in functional layer 1 and 2. It has to be mentioned that Figure 0.1 applies the layers of information theory. In most representations of architectures, layers are hardly used for representations and especially not the data layers. Normally in many descriptions of cognitive

architectures, like in [LKM12, p. 47], the data layers are projected into functional layer 3. Though it simplifies the representation, it causes confusion to the reader as functional modules, which do something are not separated from memory, which only stores something, but does nothing.

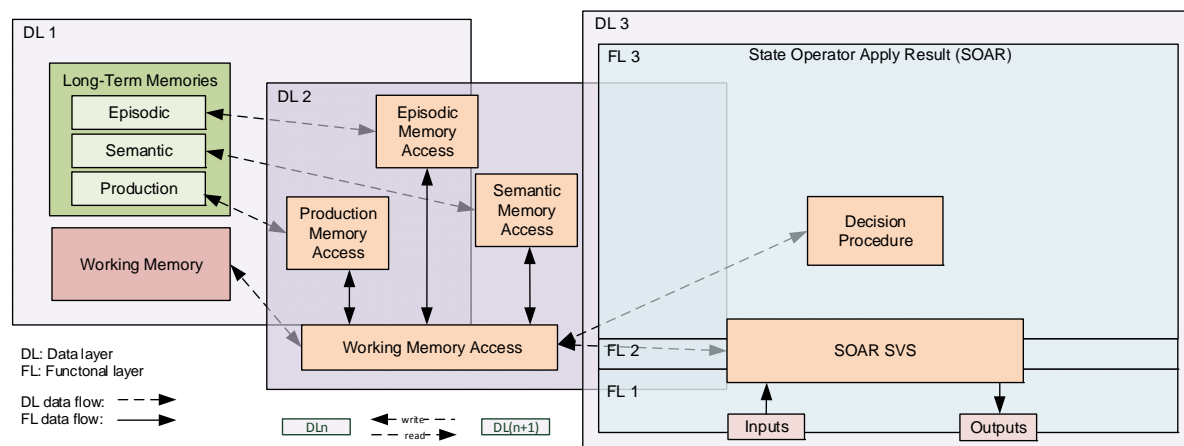


Figure 0.1: SOAR inclusive SVS block diagram, showing components and interactions based on [LKM12, p. 47]

Then, the next phases “Infer beliefs”, “Propose options” and evaluate options occur simultaneously. All matching production rules from the *production memory*, which is a long-term procedural memory, fire in parallel until no matching production rules are left. This is how long-term knowledge in SOAR is represented in implicit form. The knowledge of a production rule is very low-level or fine grained [VMS07, p. 163]. If variables are bound to them, they are instantiated [LCC+11, p. 18]. Production rules perform state elaborations (belief inference), propose operators (option proposal), retract proposed operators and compare proposed operators (option evaluation) [LCC+11, p. 21]. For instance, a production rule can create inferred beliefs, on which another production rule proposes an operator and a third one evaluates. A proposed operator is valid and kept in the system as long as the conditions of the production rule creating it are valid. Invalid operators are retracted [LCC+11, p. 18].

Each proposed operator is evaluated with the concept of *preferences*, which are linked to the operator. The preferences tell the system, which operator to choose. Production rules, which do the operator comparison can e.g. add a “require” preference to an operator, which tells the system that this operator is necessary for reaching the goal. An operator could receive a “better” preference, which tells the system that this operator is better compared to some other operator. It is therefore possible to consider intersecting, proposed operators, e.g. an operator, which was proposed by multiple production rules. Instead of a specific direct goal specific analysis, a holistic analysis is used, which considers operators of different goals and tasks. For instance, the current goal can be interrupted if other important external stimuli is present [TCM03, p. 6], [LCC+11, p. 19]. The operator with the best preferences is selected in the select option phase. Then, in the execute action from option phase, its action is applied to the working memory by a production rule [LCC+11, p. 19]. If an operator writes an external action in a certain place of the working memory, it is are sent to an executional system like the SOAR SVS [Win09, p. 3]. Soar consists only of a deliberative subsystem, without any direct reactions of input nor does it have an architectural component for monitoring the deliberation process. Further, no attentional

mechanism is applied as all production rules always fire if they are triggered. Such a concept may work well as long as there is enough computer power available, but as discussed later on in this chapter it sets limitation of expressiveness of the system.

Problem solving is done by a heuristic search, where the *problem space* is defined as the space of possible states, which can be achieved with all relevant operators that are considered for problem solving [LCC+11, p. 12]. Because all production rules fire in parallel, there is no guarantee that the preferences for one operator are set unambiguous. If ambiguity emerges e.g. if operator A is “better” than operator B and operator B is “better” than operator A, SOAR creates a sub-state to resolve the *impasse*. The goal of that sub-state is to solve the impasse of its super-state [VMS07, p. 163], [LCC+11, p. 24]. The decision-making is a forward-chaining process, which takes the system from the current problem space to a problem state closer to the goal [LLR09, p. 3]. Further, with the use of SOAR SVS [Win09, p. 4], SOAR tries to estimate the effect of an operator before it is proposed by simulating its effect. The simulation or consequence of an action is performed imaginary within the SOAR SVS module [TCM03, p. 6], [WL09, p. 3].

[LCC+11, p. 18] SOAR uses a long-term semantic memory, which keeps the information of facts used in problem solving [LCC+11, p. 92]. For instance, it can be used to save intentions or goals, which shall be reached at a later time [LL11, p. 5]. The semantic memory, which is accessible through selected operators in SOAR, is equivalent to the long-term declarative memory in ACT-R [LKMX12, p. 47], [ABB+04, p. 1042]. SOAR also implements an episodic memory. The advantage by using an episodic memory is that the agent can detect a new situation if similar episodes are not found and in the other way, it can detect repetition of situations if similar episodes are found. It can reason about the consequences of actions and predict the outcomes of changes in the world. Further, episodes can be reanalyzed in order to learn new behavior [NL07, p. 1563]. The episodic memory is static, i.e. there is no forgetting or reorganization, like in LIDA [NL07, p. 1561]. However, the problem with a static memory is that it demands more and more resources. In order to keep the performance acceptable, an upper bound have to be put on how many episodes can be stored, but the upper bound limits the capabilities of the agent [NL07, p. 1562].

Each time the agent takes an action, parts of the working memory is recorded. In order to record only relevant working memory elements, a concept from ACT-R was applied. Working memory elements, which are tested by a production rule gets a higher activation, else the activation decays with increasing time. Those working memory elements, which are activated above a certain threshold, are stored as episodes [NL07, p. 1561]. Retrieval of episodes is done in the two ways: instance- or interval-based. In the instance-based approach, a working memory tree is created, which contains a list of all working memory elements ever used. For each working memory element, links exist to all episodes that contain the linked working memory element. The searched working memory elements form a cue and based on that cue, a subset of episodes is extracted through the working memory tree. Then, the episode with the most matching working memory elements is selected. The second approach modifies the working memory tree to a list of intervals instead of episodes. For each entry in the cue, the matching intervals are extracted. The interval with the most matching working memory elements is selected and reverted back to an episode [NL07, p. 1562].

Due to the highly generalized SOAR architecture, it only implements small core functionality for processing data. The functionality of the system is defined as data in form of production rules. By applying the layered model of information theory, this is clearly visible in Figure 0.1, where only a small part of the architecture is located in functional layer 3 and much functionality is located in data layer 2 (access to memory). It can therefore be concluded that the more implemented core functionality a cognitive architecture has, the more parts of its functionality are located in the functional layers. On the other side, the more agent functionality is implemented as data like production rules, the more functionality can be found in the data layers. Therefore, theoretically, it would be possible to implement any psychological, functional model in SOAR by just creating the right rules, where the architecture of SOAR would be a meta-model, which just interprets the data. However, this approach would be very inefficient as hard-coded functionality is much faster than interpreted functionality. Actually, SOAR can model both behavior and functionality, dependent of what one wants to do with the architecture.

SOAR has been tested in a mobile robot, where it was extended with SOAR SVS. SOAR SVS was connected to a body. Especially the ability to use mental imaginary and to simulate actions in advance proved very useful as symbolic reasoning alone was inadequate in a complex world [LKMX12, p. 47], [WL09, p. 3]. One problem of mobile robotics is the ability to react fast enough in a dynamic world. Experience shows that a cognitive system shall be able to complete a decision cycle in 50-100ms. It is shown that the cognitive cycle of SOAR fulfills this requirement [LKMX12, p. 48]. Although SOAR is able to react fast enough, it is still a deliberative subsystem and does not implement a reactive subsystem, because all decisions goes through the main decision process and there is no automatic reaction on external stimuli. Instead of using some sort attentional function, which can be found in psychological models, SOAR tries to calculate everything with computer power. It may work well for the real-time behavior of the robot, but it sets the limitation of how many rules can be saved and in that way it limits the options about what SOAR can do. Also the question is raised whether it is efficient to solve attention problems with brute force. In SiMA, this problem is targeted as it comes to the activation of relevant experiences (see research question in Chapter 1.3). Another application, where the potential of the architecture is shown was in the modeling of fighter pilots in air combat scenarios by the agent TAC-Air-Soar [LLR09, p. 4], [TJJ+95, p. 15]. In order to tie in with the motivation of a potential usage of the SiMA architecture, in which computer games would profit of the developments in cognitive architectures, SOAR has been used to play synthetic characters in a modification of the computer game Unreal Tournament by Digital Extremes and Epic Games [LLR09, p. 4], [MLA+04, p. 877].

ACT-R - Adaptive Control of Thought Rational

The architecture of ACT-R (Adaptive Control of Thought-Rational) aims to model an integrated theory of the mind. It consists of several encapsulated modules, which functionalities are mapped to the cortical regions in the brain [ABB+04, p. 1036]. It is inspired by the multi-store model theory [AS68] (from [Zei10, p. 28]), which claims that there exists different memory systems and operations for each memory structures. This theory is represented within the architecture and is therefore analyzed here as the approach is the opposite of the approach of SiMA. ACT-R models some limited

areas of human cognition, mostly based on ideas from behaviorist psychology. While it uses one theory, which may have an axiomatic base, it is only specialized on certain aspects of the human mind. Regarding the modeling of behavior versus functionality, ACT-R models certain, localizable parts of the human brain in a functional model. Compared to SiMA, it is therefore assumed that this model is less functional than the SiMA model.

They are the *visual module*, the *manual module*, the *declarative module*, the *intentional module* and the *core production system* and their relations are shown in Figure 0.2. Each module from the functional layer 3 is connected to its own buffer in data layer 1 and the information in the buffers offer the only way for the modules to communicate with each other. All buffers are accessible from the *core production system*, which has the task of executing the decision process. The modules of data layer 2 are left out here, in order to be able to view the architecture in several layers. However, in data layer 2, the memory access modules may do two things: First, only to pass information and to answer requests and second, maybe to trigger modules to start their cycles. How the modules are triggered could not be extracted from the available sources. The memory-based data exchange between modules of functional layer 3 has the advantage that it allows functional modules to be completely independent of each other as they only exchange data asynchronously. From the point of developing such an application of high complexity, module independence is essential.

All information in ACT-R is represented as *chunks* [ABB+04, p. 1039] and each buffer of the system is limited to one such chunk [ABB+04, p. 1037]. A chunk is a declarative unit of knowledge and has the format of a certain subject, connected through predicates to certain values, which can be literals or other subjects. An example of a chunk would be the chunk “addition-fact” containing the “addend1” of value 3, “addend2” of value 4 and the “sum” of value 7 [ABB+04, p. 1042].

The visual module collects perceived data equivalent to SVS in SOAR or the perceptual buffer in ICARUS with the difference that it runs independently from the core production system (1, 2 in Figure 0.2). Different compared to many other architectures, sensory information is not pushed to the decision unit and it can only be actively requested by the core production system (3). Due to the limitation of the *visual buffer* of one chunk, the visual field have to be encoded in serial [VMS07, p. 164]. The top-down selection of sensory inputs works as a filter and represents an attention mechanism. The manual module executes an action, which is put in the *manual buffer* by the core production system (11, 12, and 13). The task of the intentional module is to provide a type of short-term memory, where intentions and goals are monitored, which are necessary to reach, in order to reach the top-level goal [ABB+04, p. 1041]. Similar to SOAR and ICARUS, the initial goal of the intentional buffer is the top-goal of the system and is preset at the start of the system [Bot13, p. 33]. No motivational or emotional system is used. In order to reach the provided goal, *productions* can create sub-goals (8). The goal-sub-goal relationships are maintained in a goal stack (6), where new sub-goals are added on top of the stack [TCM03, p. 3] (cites [AL98]). In that way each sub-goal is processed first and it provides the prerequisite for starting the next sub-goal. The intentional module is connected to the *goal buffer*, which keeps the active current goal (4, 5). The declarative module manages the long-term semantic memory and keeps the facts of the system (7) [ABB+04, p. 1042]. It is connected to the *retrieval buffer*, where one chunk per cycle can be retrieved (9, 10).

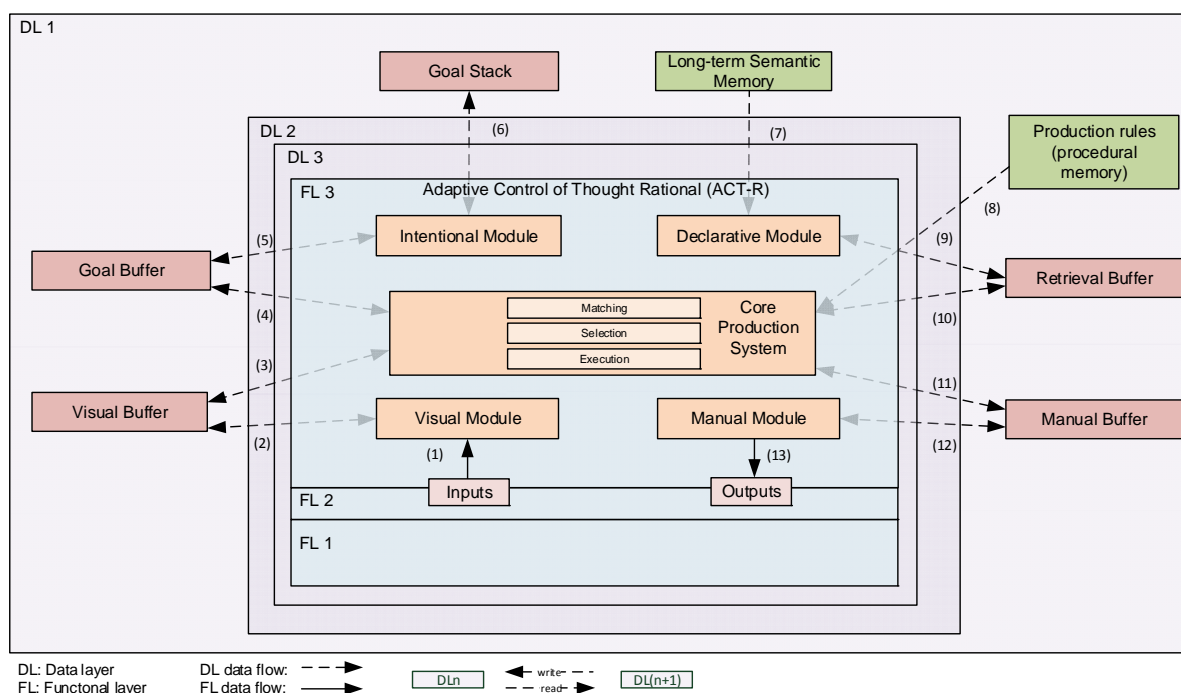


Figure 0.2: ACT-R block diagram, showing components and interactions based on [ABB+04, p. 1037]

Productions (production rules) are stored in procedural manner in the core production system. They recognize patterns in the buffers and modifies their states, e.g. an action is performed by putting a chunk with the request of an action in the manual buffer [ABB+04, p. 1037]. A difference to the production rules in SOAR is that the productions may be complicated and large as they can check multiple buffers and execute multiple actions on each buffer. Another difference to SOAR is that productions alter the long-term memory directly instead of proposing operators that alter the working memory. This together makes production rules in SOAR more abstract and flexible [TCM03, p. 5].

Due to the limited capacity of the buffers, perceived information is not provided to the core production system from bottom-up and all chunks must be serially requested by productions. This makes the system to a deliberative subsystem without reactive components. As a consequence the decision process as described in 2.1 cannot be kept strictly. It is limited to the core production system only. All phases of the decision process cannot be run within one single cycle, but within several cycles. The general cycle of the system is first to check which productions match the buffer content of the modules. This corresponds to the phase “Propose options”. Then, in the correspondent phase of evaluate options, they are evaluated in terms of *utility*, where the utility is a function, which estimates the probability to take the system one step closer to the goal [VMS07, p. 165], [Byr03, p. 97-117]. The selection of one production is called *conflict resolution* and the one with the highest *utility* is chosen in the correspondent phase “Select options”. Problem solving can be seen as a heuristic search with forward chaining. Finally, the action of the selected production rule is executed, which means to manipulate the content of one or more buffers [ABB+04, p. 1038]. This is equivalent to the phase and execute action from option.

The phase “Activate beliefs” is active as long as perceived information is explicitly requested from the visual module. After the initial goal is set in the intentional module, the cycle starts with a production that processes the goal. Based on the goal content, the next thing could be to identify perceived objects. The order what to do is decided entirely by the productions. At the time, where productions start to request chunks with facts from the declarative module, the system can be interpreted to be in the phase “Infer beliefs” [ABB+04, p. 1038].

Episodic memory is not used directly. However, some temporal information is stored, as the activation of chunks in the declarative module is increased if they are queried and decayed with increasing time and that gives an indicator what the agent has perceived recently [LL11, p. 5].

ACT-R is primarily used to simulate and model phenomena within cognitive psychology like language processing and human decision process [ABB+04, p. 1046]. For instance, ACT-R was implemented for testing interface design by modeling how human reason through them. There, ACT-R should be able to support the feature of handling goal interruption like if another new task emerges through a notification [TCM03, p. 2]. The result is that it is almost impossible to interrupt an ongoing goal. In order to trigger a production to with the task to switch goals due to an external stimulus, content of the peripheral goal has to be stored in the declarative module. In order to store it there, attention has to be directed on that content, but as the probability that a production is selected, which does not serve the current goal is very low, goal change is very hard. This is a main drawback of the system [TCM03, p. 4]. However, the problem with attention only highlights one of the problems with the usage of an incomplete model of the human mind and that it does not seem to be easily solvable within the architecture without rethinking the underlying model. On the other side, an advantage of ACT-R is that it has a very simple, easy to understanding architecture, do not use parallel processes and that may explain its wide use. Further, ACT-R has been extended with embodiment into the ACT-R/E architecture and is being used in robotics, in order to model human perception [THH+12, p. 20].

Icarus

ICARUS is intended to be used to construct problem solving systems rather than modeling human mind [LCT11, p. 1]. Although, it have much in common with SOAR [LCT11, p. 3], it started as a reactive architecture for robots and not as a problem solver [LCT11, p. 4]. Different to SOAR and similar to ACT-R, several different types of memories are used. Further, Icarus has more specific functionality than SOAR, i.e. more content in functional layer 3. Because this architecture is more similar to SiMA than SOAR, the interesting part about Icarus is the way beliefs work and how they are used in the planning. As there is no intention to model the human mind, an axiomatic approach of such a theory is not applicable. However, the technical concepts of how to realize a model of the human mind are relevant.

In the phase “Activate beliefs”, an external system provides ICARUS with a set of perceived objects, which are put into the *perceptual buffer* (1, 2 in Figure 0.3) [LCT11, p. 10]. The architecture is shown in Figure 0.3. In this representation, functional layer 1 and 2 are not considered as input data is assumed to be symbolized correctly by another system [LCT11, p. 6].

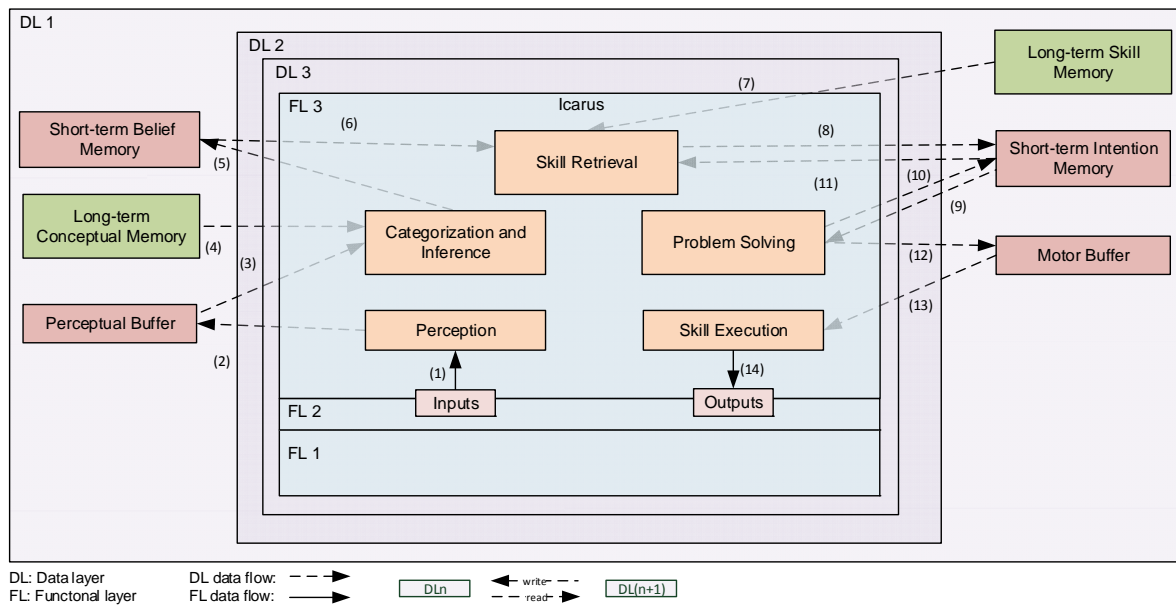


Figure 0.3: Icarus block diagram, showing components and interactions, leaned on architectural description in [LCT11, p. 10] and [Lan06, p. 9]

Because several memory types are used, a similar representation to ACT-R has been chosen. Just as in ACT-R, the data exchange is mainly done through short-term memories instead of direct communication between modules. This type of communication has the advantage of avoiding direct communication and module dependencies and it is closer to the theories of human cognition [Lan06, p. 2]. Also here, it is assumed that the modules of functional layer 3 are triggered from data layer 2, e.g. if data is written to the perceptual buffer, the module *Categorization and Inference* starts its cycle (3).

The perceived objects defined by their attributes are called *percepts* (basic beliefs) and the collection of percepts form a perceptual buffer [LCT11, p. 4]. In the phase “Infer beliefs”, *beliefs* (inferred beliefs) are created in a separate *belief memory* (5) [LCT11, p. 7]. They are instances of *concepts*, which have been matched with percepts (4). The source of concepts is the *conceptual memory*, which is a separate long-term memory [LCT11, p. 7]. Different to the percepts, all content of the belief memory is relational and a concept describes a class of situations in the environment, e.g. the primitive concept “on” describes if a percept is “on top” of another percept. Each concept consists of the following: a percept field where the types of percepts are defined, a relation field with links to sub-concepts, which must be fulfilled in order to fulfill this percept and finally a field of Boolean functions, which are tested, in order to consider the concept as fulfilled. A concept or the belief is confirmed if all Boolean functions of the concept itself as well as the Boolean functions of their sub-concepts are fulfilled. Different to the independent production rules in SOAR, in Icarus, concepts can put into a hierarchy with the most primitive concepts on the lowest level of abstraction. Primitive concepts have no sub-concept [LCT11, p. 10] and are leaves of a hierarchy tree. Such a hierarchy can be seen in Figure 0.4, where the primitive concepts “on”, “ontable” and “holding” are not related to other concepts. They just contain Boolean functions. The filled lines in the figure connects concepts in a hierarchy, where the sub-concepts must be true, in order to make the super concept true. The dashed lines are

negated concepts, which must be false in order to satisfy the super concept. For instance, the agent must not be “holding” anything if the “hand-empty” is supposed to be true. Finally, the top concept of the figure is valid and something is “pickupable” the agent does not hold anything “hand-empty”, the target percept is on the table “ontable” and there is no other percept is on the target percept “clear”. The belief memory is built in a bottom-up like manner, starting from primitive beliefs. These steps keep the system updated [LCT11, p. 11].

In the phase propose options, the variables of *skills* (option) are being bound to the percepts and beliefs, in order to extract applicable skills (6). The long-term knowledge of what to do in a certain situation is stored in a long-term *skill memory* (7). Its structure is similar to the structure of the conceptual memory. A *skill* consists of the following fields: necessary percepts, preconditions in the form of inferred beliefs, sub-skills of necessary skills to execute first, an action and resulting effects on the environment [LCT11, p. 12]. While skills are the templates for actions offering many possibilities of implementation, *intentions* are applicable skills for the current situation. A *primitive intention* contains an action, which can be executed in the environment. The intentions are put in the short-term *intention memory* (8), which is equivalent to the structure of the belief memory. Icarus does not have any episodic memory nor does not use episodes in problem solving.

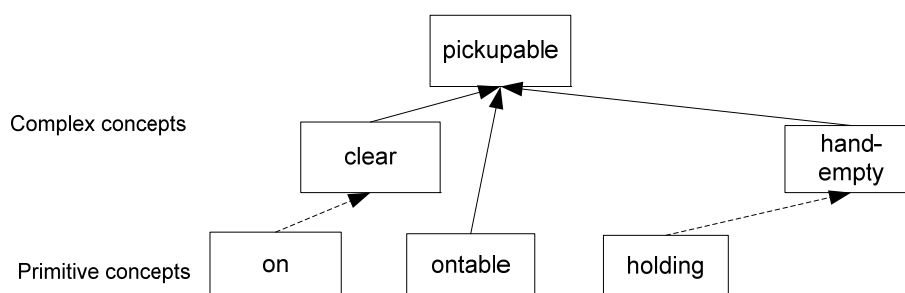


Figure 0.4: Hierarchy of concepts in Icarus [LCT11, p. 10].

Icarus uses one or more predefined top-level goals as motivation [LCT11, p. 17]. If the top-level intention, which satisfy the top-level goals is fulfilled, the architecture halts and waits for new instructions [LCT11, p. 19]. In the evaluation of intentions to choose in the phase evaluate options, the effort is calculated (9). The more conditions of an intention are unsatisfied, the more reasoning through heuristic search is necessary and the higher the effort (10, 11) [LCT11, p. 25].

Problem solving has the purpose to find skills, which conditions are all fulfilled and takes the system to a state closer to the goal. Often, skills are only partially fulfilled, which require further reasoning. Similar to impasses in SOAR, there are four types of problems, which can occur in the decision-making: First, skills may be too generic, which can provide the system with erroneous actions or no sub-skills match the beliefs and percepts. Skills may be too specific, where the system proceeds along a skill-tree and eventually have no more skills to choose. Further, actions are only probabilistic, which may cause that the effects of the intentions are not fulfilled. Also a change in the environment may cause the selected skill hierarchy to be inappropriate [LCT11, p. 19]. In order to deal with the problems in decision-making, *goals* and *problems* are introduced. A goal is a desired state similar to a belief, but extended with negations of arguments. If there are inferred beliefs, which matches the goal pattern, the goal is *satisfied*. A problem is a set of conjoined goals and it is solved if all goals in the set are

satisfied. In order to solve a problem, differences are defined between the inferred beliefs and the goals. Each difference for a problem results in a sub-problem, which has to be solved [LCT11, p. 19]. Both forward-chaining and backwards-chaining is applicable. The forward-chaining is based on finding skills, which matches inferred beliefs. Backwards-chaining looks for skills, which effect matches a goal. The combination of both can be used in a means-ends-analysis [LCT11, p. 26].

In the phase select option, the architecture selects the one intention, which has the lowest effort [LCT11, p. 17]. In the phase execute action from option, if the selected intention is a primitive intention, its action is executed in the environment, else the intention is not primitive and its sub-skills have to be accomplished in the following cycles. In that way, the architecture processes from the top to the bottom of the skill tree until it can execute primitive actions. After applying a primitive intention, the architecture checks (12, 13, 14), if the desired effects were accomplished by examining the current beliefs. If all conditions of an intention were satisfied, the next unsatisfied sub-skill of the parent intention is processed [LCT11, p. 18].

Similar to SOAR, there is no reactive subsystem although the architecture started there and ICARUS resides in the deliberative subsystem. If the system finds itself in one of these problem zones mentioned above, it tries to recover by testing other choices. If that is not possible, the default behavior is to abandon its current intention including the whole skill-tree. This shows some elements of a meta-management subsystem, where the architecture takes another way to solve the problem [LCT11, p. 27].

ICARUS have been used in problem solving applications like urban driving in a town, in which the agent shall deliver a package to a certain address. In this application, road segments, lane lines, buildings and other vehicles have to be considered [Lan05b, p. 5]. Further simulations are the card game solitaire and the blocks world problem. In the blocks world, three blocks are placed in some constellation on a table. They shall be brought into another constellation and the agent has the possibility to move one block at the time. The task is to find out how to move the blocks, in order to reach the goal [Lan05b, p. 6]. Finally, ICARUS has been implemented in a modified version of the first-person shooter game Quake 3 by Silicon Ice Development [LLR09, p. 4], [CKN+07, p. 72].

CHREST - Chunk Hierarchy and REtrieval STructures

The architecture CHREST (Chunk Hierarchy and REtrieval STructures) was developed, in order to simulate processes of human mind. In particular, it is used explore the process of visual attention in chess. There, the task is to recognize situations in those, the king is threatened [RLSL09, p. 1]. Therefore, an advanced perception and attention system [RLSL09, p. 2] has been developed. Similar to ACT-R, it only models a certain part of the human brain and the model is extracted from empirical results of human behavior. Although this architecture only seems to model a certain part of the human mind based on measured behavior, its results looks convincing and its technical model for realizing the attentional function could be used as a source of inspiration in SiMA as long as it is conforming to the provided psychoanalytical constraints.

Figure 0.5 shows the components and their interactions of CHREST. Just as the previously described architectures, data transfer is mainly done through memory access. In the phase “activate beliefs”, similar to Icarus, perceived objects (basic beliefs) represented by chess figure symbols with positions, are put in a sensory memory (1, 2 in Figure 0.5). Different to many other architectures, the sensory system of CHREST is not able to perceive the whole world at once, only a small area around its focus. The focus area is only one chess position. In the periphery, chess positions within a radius of two fields from the focus area can be partially recognized [RLSL09, p. 4]. The focus is set by the attention system in CHREST [RLSL09, p. 2]. Similar to Icarus, only the functional layer 3 is used as the symbols have to be provided by an external system. Just as ACT-R, this model is limited only to special functionality of the human mind without regarding the whole of it.

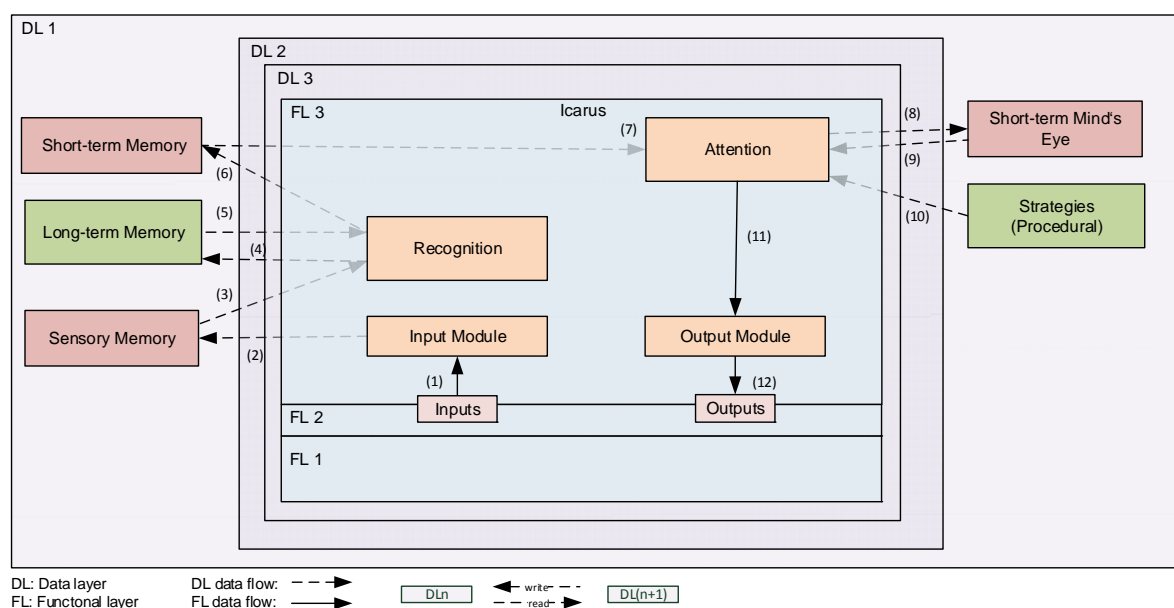


Figure 0.5: CHREST block diagram with components and interactions based on [RLSL09, p. 5] and [SLG08, p. 100]

In the phase “infer beliefs”, the content of the sensory memory is compared to *chunks* in the long-term memory (3, 5). Like in ACT-R, a chunk is a collection of features, which have some meaning for the agent. In this case, it consists of patterns of chess figures with their positions. In the long-term memory, chunks are hierarchically ordered as a directed graph in a tree structure, which are connected through semantic links [RLSL09, p. 2]. Chunks, which are not recognized are added as new chunks to the long-term memory (4) and the information of chunks, which are partly recognized is added to the partial recognized chunk, making these recognized chunks more detailed [RLSL09, p. 2]. In that way, the chunks are getting more specialized in the branches of the tree. If matching chunks are found, references of the four best matches are passed to the short-term memory (6) [RLSL09, p. 2]. Based on research of the capability of human short-term memory, the size of it is limited to 4 chunks in this architecture [SLG08, p. 100].

Due to limitation of information from the sources, the decision process cannot be described in detail. However, the action of the system is to execute simulated eye movements, which are controlled through the attention module (12) [RLSL09, p. 2]. Its behavior is determined by the chunks in the

short-term memory (7). Some of the chunks are learned and some are basic heuristic patterns to guide attention (10) [RLSL09, p. 2]. As it is possible to perceive only a small part of the world at once, a complete picture of the situation has to be built in the *minds' eye* (8, 9). A high-level, general chunk may be recognized, e.g. the position of the king only. This chunk is linked to more specialized chunks with more other chess figures. It causes the attention to focus on those parts of the chess field, which are disjoint from the original chunk, in order to verify that it is the focused chunk [RLSL09, p. 2].

The main goal of the system is to determine if the position is of the king is check or not. Therefore, the system uses no other motivations or emotions. If no check chunk was recognized, the system uses the mind's eye to virtually move other pieces towards the king and the king towards other pieces, in order to recognize if that would be a check in the next turn. In the module of the mind's eye, all chunks of the short-term memory are projected, in order to get a holistic picture of the whole situation [RLSL09, p. 4].

Compared to other architectures, CHREST contains only little action planning as the purpose is only to simulate eye movements. It has limited ability of problem solving compared to SOAR and ACT-R and instead a case-based approach is done. It retrieves familiar chunks, which solves isolated decomposed problems [GL10, p. 4]. The system is a deliberative system without reactive subsystem or meta-management subsystem. There is no usage of episodes in the system. CHREST offers a solution to the frame problem in three steps, which all aim to reduce the amount of information. The frame problem addresses the problem to separate relevant changes in the environment from the non-relevant. First, the visual field is limited by the simulated eye, then the top-down influence of knowledge about what is important reduces information and third, the short-term memory is limited. Altogether, it makes CHREST to a highly selective system [GL10, p. 2].

The utilization of CHREST is similar to ACT-R, but here the architecture has been specialized to simulate the mental processes of chess players, in particular phenomena regarding chess expertise. The time needed to recognize if the king is threatened by other chess pieces are simulated, which is dependent on number of stored chunks and the size of short-term memory [RLSL09, p. 6], [GL10, p. 2]. In tests, especially timings are measured and compared to expert and novice human players. For that purpose, time delays were added to the system operations [RLSL09, p. 5]. Further, CHREST is also used to explore how human acquire language [GL10, p. 4]. However, the question can be asked if CHREST would be able to be extended to a model of the complete human mind without creating contradictions to other psychological models. Further, just as by ACT-R, the question is also whether CHREST really models human functionality or just observable human behavior.

BDI - Belief, Desire, Intention

The BDI (Belief, Desire, Intention) architecture as described in [RG95, p. 312] is based on a theory of practical reasoning originally formed as a planning theory of intention by [Bra87]. There, an architecture is defined, which shall fulfill the requirements of real-time applications [RG95, p. 313]. It uses the three mentalistic attributes belief, desire and intention to determine the state of the agent [RG95, p. 312]. As BDI is no specific implementation of an architecture, one specific implementation is described in detail and another implementation provides an alternative way of planning and updating

the deliberative sub-system [WHB+00, pp. 531-541]. Both architectures are used to control robots in the RoboCup challenge, which is a test bed for agents through artificial soccer [GTC+10, pp. 74-83]. For SiMA, the analysis of BDI is interesting because from the analyzed architectures, it has the most similar structure compared to SiMA. Further, the state definition through the three mentalistic attributes seems to be a simplification of some functionality in SiMA. However, BDI does not raise the intention to model the human mind and the problem of axiomatic theories of psychology do not apply here. However, compared to the analyzed architectures until now, it incorporates one more aspect of the human mind and that is a motivational system.

In Figure 0.6, in functional layer 1, low-level handling like video processing (1 in Figure 0.6) and communication with the physical robots is done (14). A video server provides information about positions, orientations and speed of objects. A communication server allows the controlling of the robots through messages [GTC+10, pp. 76-77]. On the functional layer 2 and part of 3, two modules are located. The first is the *sensorial* that is a symbolizer. It translates visual information from the video server into *basic information* (basic beliefs) in prolog formalism (2). This functionality fits into the description of functional layer 2. Further, it also proposes *inference rules* (production rule), which can be applied on the basic information (3). This makes this module a part of functional layer 3 too. The basic information is e.g. the position of a robot and the ball, while the inference rules if applied could calculate the relation of a robot to the ball. At this stage, the inference rules activated, but not executed. The second sub-layer is the *planner manager* that provides a set of schematic plans like “go to ball” (13). Each scheme contains low-level sequences of commands. Within such a set, the execution of low-level actions is dependent on the perceived situation and may be updated independent of the deliberative layer [GTC+10, p. 77]. BDI is implemented in the functional layer 3. The layered approach offers several advantages. First, each layer can be developed independently. Second, in the planner manager, low level actions are generated and the architecture of the functional layer 3 does not have to care about details like the exact movement direction.

In the beginning of a cycle, in the phase activate beliefs of Figure 2.1, the basic information together with the proposed inference rules are provided from the *belief base* [GTC+10, p. 78]. The concept of many specialized long-term memories is similar to ICARUS. Similarly, a subset of the belief base is called the *actual beliefs*. In the actual beliefs, the proposed inference rules are executed (4). However, the computation of the actual beliefs is only done by request of a component [GTC+10, p. 79]. This is done, in order to maximize performance as not all inference rules are necessary. It is also a way of applying a top-down based focus of attention. The attention selects the content that is further processed. It is similar to the role of attention codelets in LIDA, but instead based on top-down influences.

Desires represent the long-term goals of the agent, which origin in predefined goals for specific situations, e.g. if a robot is losing the game, it should prefer to attack instead of defending the own goal. They are equivalent to the emotions in LIDA and serve as a motivational system. All desires are stored in the *desire base*. In the phase “Infer beliefs”, they are tested against the actual beliefs from the belief base (5, 6), in order to provide the system with the applicable desires called *justified desires*. Each desire has an importance value, which is used at the selection of actions [GTC+10, p. 80]. Different to SOAR, ACT-R and ICARUS, the system does not start with one given goal. Instead the

goals emerges out of the preset desires, which are dependent on the environmental situation. This is an advancement compared to the previous architectures.

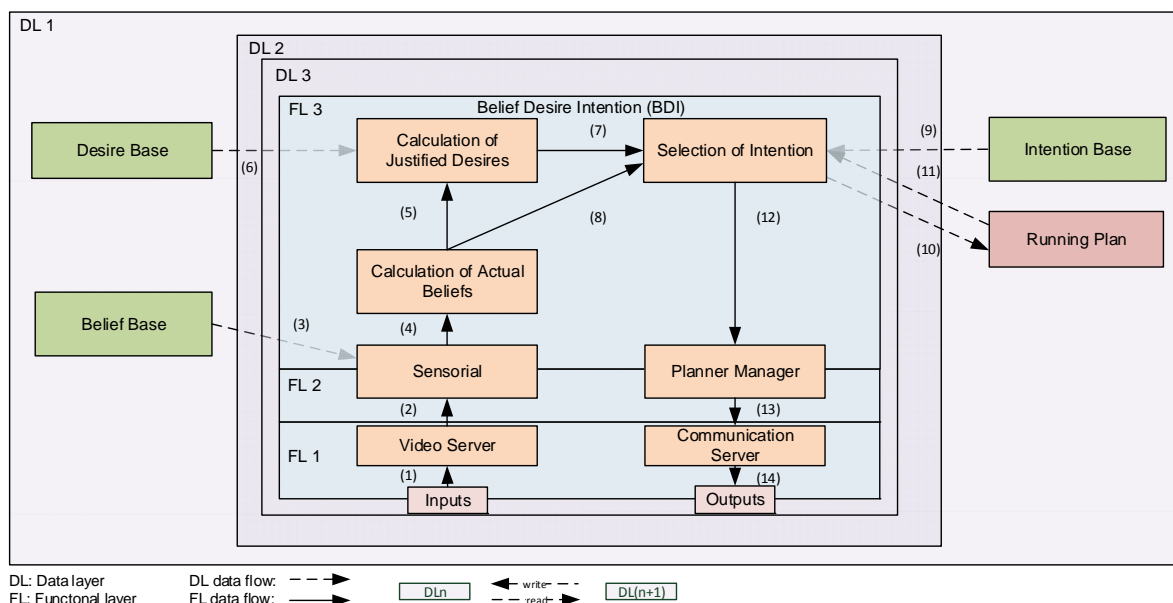


Figure 0.6: BDI block diagram with components and interactions based on [GTC+10, p. 82]

Intentions are high-level plans on how the agent wants to achieve its justified desires. They are stored in the *intention base*. Their instances, which fulfill certain justified desires as well as the actual beliefs are *applicable intentions* (7, 8, and 9). They are generated in the propose options phase. In the proposal of applicable intentions, the architecture can make use of the current running plan (11) as a condition for applicability, in order to create sequences of intentions, which are triggered by their predecessors. However, this cannot be seen as an episodic memory, rather more a short-term memory. Further, an intention can be applicable as a *reactive rule* without the need to fulfill a certain desire, if the agent has to react directly on a certain situation. This is important in a dynamic environment [GTC+10, pp. 81-82]. Reactive rules provide a method to skip expensive reasoning and to take action immediately.

BDI does not focus on problem solving like SOAR and ACT-R. Instead, it uses a case-based approach, where the system tries to find matching behaviors for recognized situations. In the phase select option, the importance values are applied on their corresponding intentions and finally, the intention with the highest importance value is selected to execute (12). The current intention is also stored for the next cycle (11). Then, in the phase execute action from option, the plan of the intention is sent to the planner manager (13), which transforms it into low-level action commands (14) [GTC+10, pp. 82]. The usage of hierarchical plans, in the lowest, reactive levels is a good way to let the deliberative subsystem take major decisions and not to care about details, which can be automatically managed by the reactive subsystem.

The architecture in [WHB+00, pp. 531-541] offers a more detailed planning, after selection of an intention. It offers a three layered planning compared to the two layered planning described previously. The intentions provide only a *course-grained planning*, giving the delta between the current state and

the intended state. In the *fine-grained planning*, *skills* are used. A skill is in this implementation a short-term plan, consists of some *atomic actions*. This architecture adds a basic meta-management subsystem in the form of *decision points*. Within the course-grained planning, decision points are set, either with set time intervals or between the skills. At these decision points, the architecture collects perceptual data and reconsiders its previous choices [WHB+00, pp. 536-537]. In that way, fast reactivity is achieved as perception is only requested when necessary. This is a difference to [GTC+10, pp. 74-83], where perception is automatically updated. Further, it allows a very primitive form of meta-management subsystem as the agent can control the progress of its own actions. Perhaps, it is not bionic, but it is a good way to implement monitoring with less effort.

The architectures taken together show all forms of subsystems, i.e. a reactive, deliberative as well as a meta-management subsystem. It is well suited for fast reaction of robots in a dynamic world and at the same time provides deliberative and reflexive functionality for the robot. Besides of RoboCup, the BDI architecture has also been tested in computer games as virtual characters, who's the task is to extinguish a fire [DWvDH09, p. 7]. Although problem solving may not be the strength of BDI, it is a good example of an architecture, which is simple regarding how the components work, easy to understand and it is applicable in many different environments, including real-time applications.

LIDA - Learning Intelligent Distribution Agent

LIDA (Learning Intelligent Distribution Agent) is based on a combination of recent theories of the mind, which are merged into a single cognitive architecture. Among them is the Global Work Space theory, which is a connectionist theory and the widely most accepted psychological and neurobiological theory of the role of conscious data processing in cognition [SB05, pp. 164-174], [FF12, p. 103]. Further, LIDA also incorporates parts of the H-CogAff architecture [SC05, p. 145-174], [FF12, p. 105]. Together with the matter of fact that the sensory part of the architecture is also connectionist, which requires an embodiment [FF12, p. 106], this makes LIDA to a hybrid architecture, connecting both architectural paradigms. Just as SiMA, LIDA also claims to model the human mind. As the approach of LIDA is based on various, independent theories, the problem may emerge with inconsistencies, which are hard or even impossible to solve. Therefore, the problems of a non-axiomatic approach may be built in into the core of the architecture. However, LIDA is interesting as it implements more comprehensive functionality of the human mind compared to the previous architectures and it offers interesting technical solutions to problems regarding the implementation of a model of the human mind.

One main concept of the architecture is the usage of *codelets*. Codelets implement the processors of the Global Workspace Theory and are independent pieces of code, which have a specialized function, run independently and act like daemons. They continuously test their assigned memory for certain beliefs and execute actions [FF12, p. 105]. Therefore, they are visualized in Figure 0.7 in a single module, which runs independent of a cognitive cycle (red lines around it). Codelets are active parts of the architecture and are not exactly the same as production rules, although they have an equivalent function. Different to other production rule systems, the decision-making cycle consists of several specialized mechanisms, which are interacting in continuous process. Here, it is essential that the

interaction between those mechanisms works well and that the system does not come to a halt [SWY11, p. 27].

In the first phase “activate beliefs”, perceived information is collected in a sensory module (1 in Figure 0.7). Content from the *Perceptual Associative Memory* is then activated by perception codelets in the *Sensory Module* (2). The Perceptual Associative Memory, which is a long-term memory, is used to create the basic beliefs, where sensory information is identified and classified. This memory consists of a semantic network with activations [RBDF06, p. 245], [FR06, p. 43]. Perception codelets search the input for certain attributes and activate the corresponding nodes of the memory by copying them. The activation is spreading through the nodes to activate related content like the category of an object [FR06, p. 43]. Further, the activated content is also provided to the motor module, in order to create an instance of a plan (4). The circular arrows of the modules in functional layer 1 and 2 show that they run independently of other modules, like an own thread.

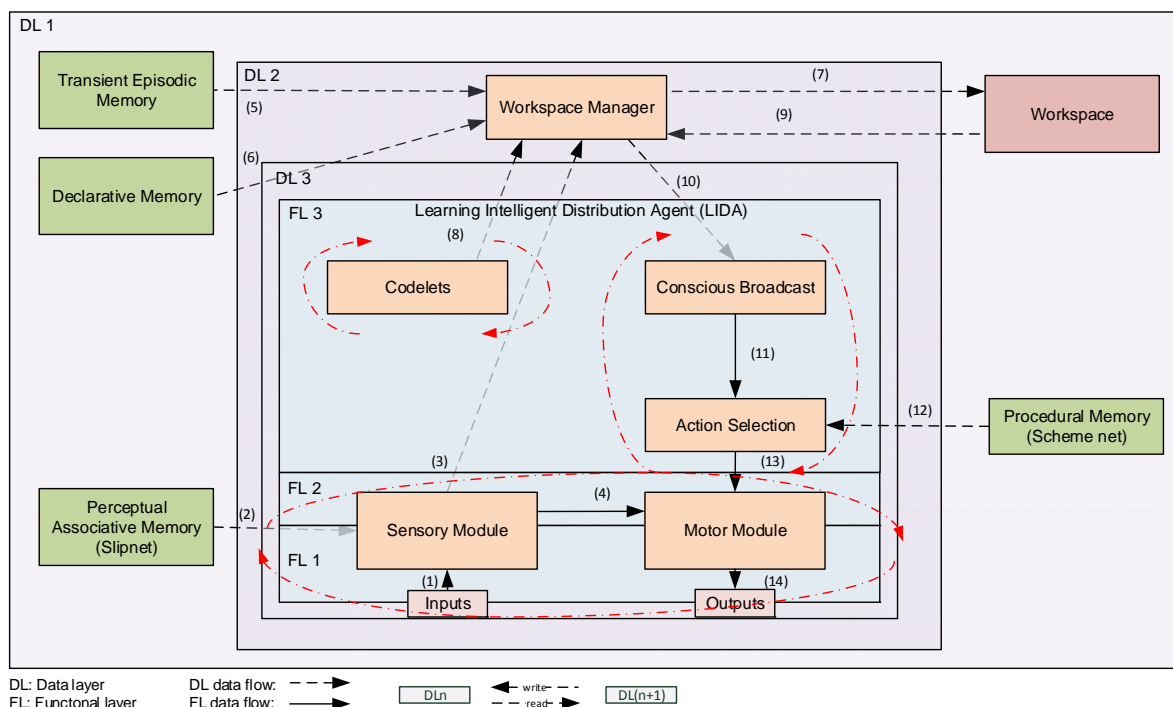


Figure 0.7: LIDA block diagram with components and interactions based on [FF12, p. 109] and [RBDF06, p. 245]

In the next phase “infer beliefs”, the activated nodes are put as *percepts* to the *workspace* (3, 7). The workspace is a short-term memory that holds the current percepts, previous not decayed percepts and local associations to long-term declarative memories. It is interpreted as a preconscious buffer. In Figure 0.7, anywhere, where a dashed arrow from the memory of data layer 1 goes directly to data layer 3, it is assumed that there is a data access module in data layer 2. However, in the workspace access, a *Workspace Manager* has been visualized to illustrate the functionality of memory retrieval from different memory types. The local associations are activated by the percepts and are linked to content in the *transient episodic memory* (5) and a semantic memory, the *declarative memory* (6) [FR06, p. 44]. The transient episodic memory contains stored events, which are later consolidated into

the declarative memory at a later stage. It means that there are no episodes, which are kept forever as they are all transformed into semantic knowledge. This is not the case of the human mind. The content of the transient episodic memory originates from the *conscious broadcast*, which defines the current goal and its state. All conscious content of a cycle is stored [FF12, p. 107]. Therefore, the problems with a static memory discussed in SOAR are not applicable here.

The phases “propose option”, evaluate options and select option can be seen as executed twice within a cognitive cycle. The first time is in the selection of beliefs to act upon and once again in the selection of actions. As a huge amount of inputs is expected, the architecture uses attention to filter sensory data and to recruit needed resources for a response [FF12, p. 105]. The bottom-up approach is visible in the attentional mechanism, where attention codelets search the workspace for certain content, which should be brought to consciousness. It may be urgent content, expectations of previous actions or in another way important. Each attention codelet searches for its specific content. The codelets form coalitions with the content of the workspace and compete with each other for consciousness (8). Each attention codelet consists of the following: concerns, which are the conditions, which activates the codelet; a base-level activation; a current activation, which describes the degree of match between the conditions and present beliefs; emotions, which influence the importance of the codelet [FR06, p. 44], [FF12, p. 107]. The winning coalition with the highest activation gains access to the global workspace and performs a conscious broadcast (9, 10). This is supposed to be the conscious data processing functionality of the system [FF12, p. 107].

The conscious broadcast (11) results in the recruitment of resources based upon the selected content. The process of activating content from the workspace and instantiating a template action plan runs independently, which is illustrated with dashed circular arrows in Figure 0.7. Behavior codelets activate schemes (12), which are comparable with the skills in Icarus. However, compared to Icarus, not all low-level actions are directly defined by the deliberation process. The schemes consist of a context, which is a collection of beliefs, which shall be fulfilled, an action and results. They are stored a directed graph in a long-term *procedural memory*. Schemes, which are activated by the conscious broadcast are instantiated and put into an action selection mechanism. This is the second run of the phase propose options. Based on the emotions, the beliefs and the relations to other behaviors, one scheme is selected (13) and its action executed (14) [FR06, p. 44], [FF12, p. 108]. Problem solving is applied in a case-based approach. The instantiation of actions is similar to BDI, where visual input is provided to the motor module, in order to instantiate a scheme to an executable action plan. This approach for action planning could be usable in SiMA too.

The motivational system, which finally ends up in the selection of an action, works as follows: During the categorization and identification of objects, simple, reactive feelings/emotions are activated. During the activation of content from the episodic memory, emotions, which were experienced at the time of storage are also activated. They provide the system with emotions from earlier experiences. These emotions are the motivations to bring something to a conscious broadcast and as a consequence to trigger an action. The selection of how the goal shall be reached is significantly decided by the emotions [FR06, pp. 43-45].

LIDA, which is a specification of the general CogAff architecture has both a reactive and a deliberative subsystem, but no meta-management subsystem [FR06, p. 46]. The reactive parts are realized with a

direct connection of the sensory memory to the action selection module, where perception is considered at the choice of low-level actions [FF12, p. 111]. From an architectural view, through the many types of codelets, which run independently, there are many parallel processes, which would allow the optimal utilization of resources as modules do not have to wait on each other. For SiMA, which consists of many serially connected modules, such a system could be relevant at the time where SiMA shall be used in a real world application.

The predecessor of LIDA, IDA is deployed in the US navy. There, it manages jobs for sailors, whose tasks are about to end. The task is to offer jobs for sailors depending on the sailor's preferences, the Navy's policies, the needs of the tasks and the urgency. The negotiation with the sailors is done by emails in unstructured English [FF12, p. 104].

Appendix B – SiMA Software Implementation

In the following, topics that refer to the SiMA model or its implementation are referred here.

The SiMA Model of Abstraction Level 1

In this work, the SiMA model is often referred. However, mostly abstraction level 3 is relevant. As only parts of the SiMA model of abstraction level 1 (see chapter 2.3.1) are relevant, only those parts are viewed in detail. For completeness, the whole SiMA model is illustrated in Figure 0.1.

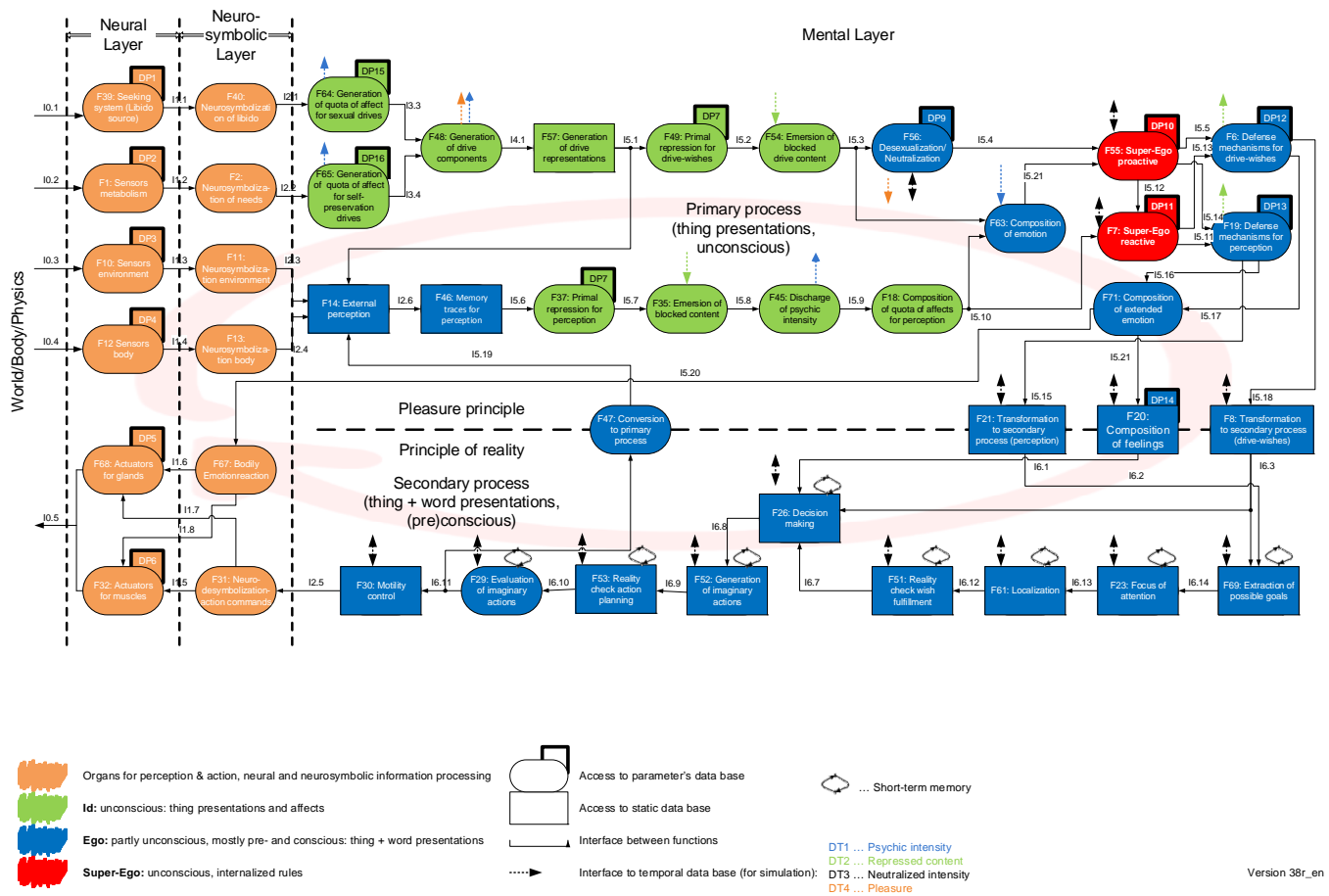


Figure 0.1: The complete SiMA model V38r on abstraction level 1

Data Structures

In Chapter 3.1, existing concepts of data structures were modified and new data structures were added. The major extension compared to previous data structures, was the introduction of graph structures for images. In the following, the most important modifications and extensions of concepts of implementation of ARSi11 [Zei10, pp. 96-98] and [Deu11, pp. 116-126] are described.

Thing Presentation Mesh and Word Presentation Mesh

Originally, the thing presentation mesh, which is realized by the class `clsThingPresentationMesh` contained an `ArrayList`, which kept all internal associations of the structure. For object representatives, those attribute associations (`clsAssociationAttribute`) were connected to thing presentations (`clsThingPresentation`) in represented by properties like color and shape. However, the thing presentation mesh was unable to keep its external associations to other like drive meshes

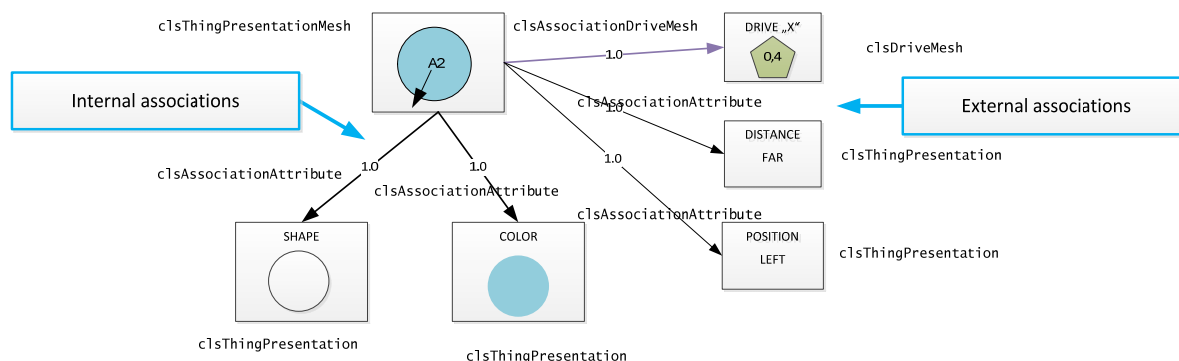


Figure 0.2: Example of the software structure of an object representative in the primary process

(`clsDriveMesh`) and thing presentations of positions. Therefore, the `ArrayList` of external associations was added to the structure. Now, all relevant content could be associated and kept track of in the code. Further, in the primary process, there are different association types for each connected data structure types. While the `clsAssociationAttribute` always connects to a `clsThingPresentation`, the `clsAssociationDriveMesh` always connects a `clsDriveMesh` [Zei10, pp. 96-102]. In Figure 0.2, which corresponds to the concept of Figure 2.5, the thing presentation mesh for an object representative is shown together with its associations. Each association type is represented by an own color: black for `clsAssociationAttribute` and purple for `clsAssociationDriveMesh`.

The object representative of the word presentation mesh was realized with the class `clsWordPresentationMesh`. The `clsWordPresentation` in the secondary process is used equivalent to the `clsThingPresentation` in the primary process. There exist two types of associations: The `clsAssociationWP` always connects a `clsWordPresentationMesh` to either a `clsThingPresentationMesh`, a `clsDriveMesh` or a `clsEmotion`. The

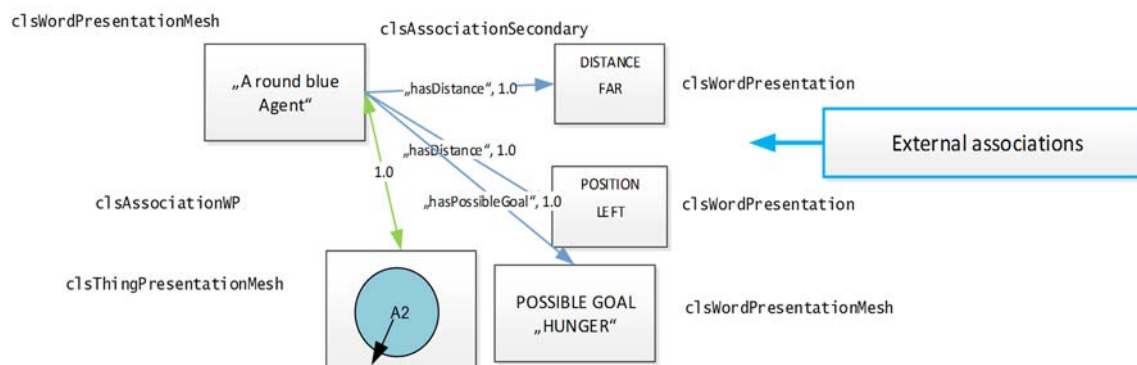


Figure 0.3: Example of the software structure of an object representative in the secondary process

`clsAssociationSecondary` is the only association type that is used in secondary process data structures. The `clsAssociationSecondary` allows the functions of the secondary process to use predicates like `<hasDistance>` or `<hasPossibleGoal>`, which makes the addressing of associated data structures easier.

In Figure 0.3, the object representative of a word presentation mesh is shown. Similar to Figure 0.2, the associations are colored: the `clsAssociationWP` with green and `clsAssociationSecondary` with blue color. The predicates used in the secondary process imply that the connected data structures can be treated like member objects of a class. As word presentations express literals and word presentation meshes objects, the associations are just pointers to those objects. Therefore, it makes sense to use the `clsWordPresentationMesh` as a class and extend it with getters and setters for the associations used. Instead of manually searching through an `ArrayList` with associations and comparing every content of a data structure with some pattern, which has been done in ARSi11. It would be better just to call the getter of that property, which shall be retrieved. For instance, to get the position and distance of an object representative, in a first step, it should be enough to have a method, where only the predicate is used as a parameter and the value of the associated data structure is returned. In a second step, an extension of the `clsWordPresentationMesh` should implement real getters and setters for the positions. This approach has been realized in the `clsWordPresentationMesh`. The problem is to keep the association weights, as they may differ from 1.0. However, in the most cases, only values are of interest and not the associations themselves. In the `clsLogicalStructureComposition` of Figure 0.4, there are the `ArrayList` of the internal and external associations (see Chapter 2.3.4). Additionally, a `HashMap` has been added, with the predicate as a key and an `ArrayList<clsSecondaryDataStructure>` as value. The associations are not present here. In that way, the associations are kept in the `ArrayList` and the `HashMap` provides a shortcut to directly reach the values for those cases, where the associations are of no importance.

In the `clsWordPresentationMesh`, methods are provided like `setUniqueProperty()` and `addReplaceNonUniqueProperty()`, which are used to add information. Methods like `getUniqueProperty()` and `getNonUniqueProperty()` are used to get this information.

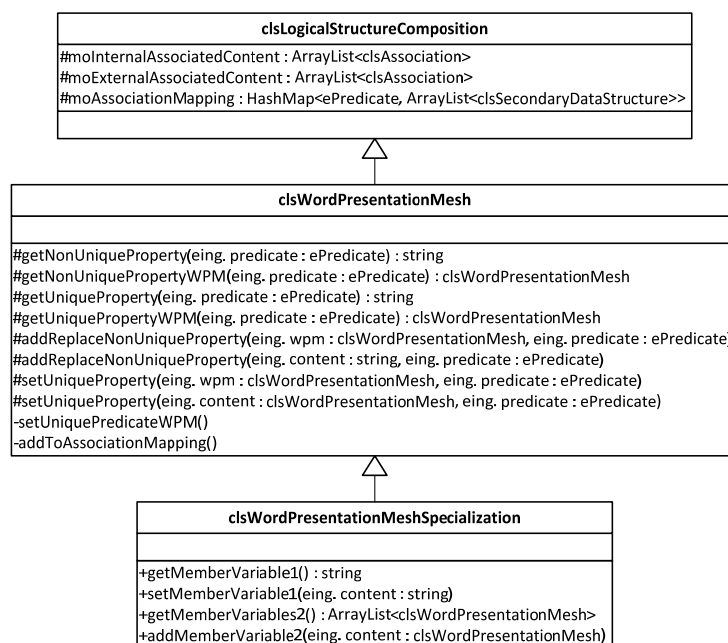


Figure 0.4: UML class diagram of getters and setters in the `clsWordPresentationMesh`

The set-methods can set either a unique property or add multiple properties. They can accept either a string or a `clsWordPresentationMesh` as input. The string is always automatically transformed into a `clsWordPresentation`. The set-methods do two things: First, they add or replace an association in one of the `ArrayList`, then they put the entry in the `HashMap`. The get-methods only access the `HashMap` and from there directly retrieve the values. In that way, performance is boosted and a manual search is saved. In a data structure, which extends the `clsWordPresentationMesh` like the `clsWordPresentationMeshSpecialization` in Figure 0.4, methods are implemented, which do not even need the predicate, as the predicate is coded into the getters and setters. The extended class now behaves exactly like an object with member variables, although it is based only on psychoanalytical data structures. The psychoanalytical data structures form a language of high generality, which is interpreted by the SiMA architecture.

Image and Act

According to the argumentation in Chapter 3.1.1, the former data structure template image was replaced with a thing presentation mesh. However, the association that was used in the template images [Zei10, p. 101], the `clsAssociationTemp`, remains. This association explicitly expresses simultaneousness. In Figure 0.5, this type of association has the color orange. Associations of the secondary process have a predicate like `<hasSuper>` and a weight like 1.0. It is a part of the external associations of the object representative as it does not define the object representative. However, as it defines the template images, it is a part of its internal associations. Because each type of data structure has its own association in the primary process, the `clsAssociationPrimary` was introduced as a general association between images. In that way, template images are connected to each other in an

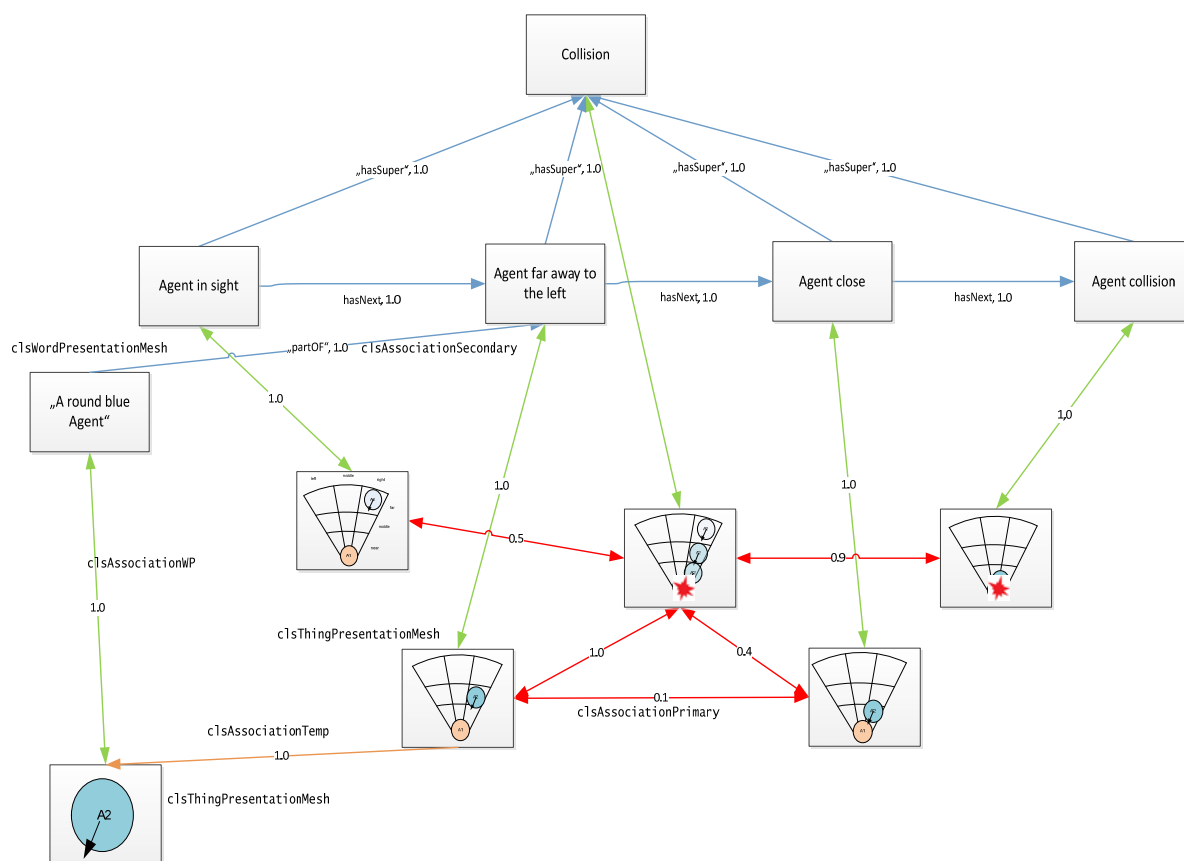


Figure 0.5: Example of the software structure of an act, which consists of images and object representatives

undirected graph. Similar to object representatives, the `clsThingPresentationMesh` of the template image is connected to a `clsWordPresentationMesh` with a `clsAssociationWP`. In the act, which is no explicit data structure, the `clsWordPresentationMesh` of the defining template images are connected with `clsAssociationSecondary`. For the definition of the act, the predicates `<hasNext>` and `<hasSuper>` are used (see Chapter 3.1.2). The act can be interpreted as a linked list and the predicate `<hasNext>` invites to create an iterator of psychoanalytical data structures. As seen before in `clsWordPresentationMesh`, the psychoanalytical data structures can be transformed into normal Java objects with standard operations acting on them. It is much easier to work with instead of manually searching through lists of associations. In Figure 0.5, the complete implementation of an act is viewed, where Figure 0.5 extends the concepts of Figure 0.2 and Figure 0.3.

In the SiMA model, data structures are passed from one functional module to another. SiMA implements the design pattern *pipes and filters*, where the system is split into several, serially connected functionalities, which passes data from the input to the output. As described in [Zei10, pp. 102-103], two neighboring modules implement an interface, which defines the data structures. In the first module, a data structure is passed to the second module by putting it into the send-method of the module. The send-method contains the receive-method of the second module,

which parameters are the data structures of that interface. As soon as the first module has finished processing, the second module starts and executes the implemented receive method. In the case of images and acts, which are graphs, the passing of content is made simple. Here, any image can be passed alone because it is connected to all other images through its associations. In the modules of the primary process, only the perceived image needs to be passed, as all activated memories are connected to it. The same applies to acts of the secondary process, where only one image has to be passed, in order to pass the whole act.

The knowledge base for memories in SiMA is based on Protégé Frames [Zei10, pp. 97-99]. However, in previous versions of SiMA, there were no instances used, only types, i.e. only semantic knowledge. No episodic knowledge has been stored. Images containing several object representatives did not exist. The perceived image was represented as an `ArrayList` with containers of independent object representatives. Each container kept a clone of the type of the object representative, which was loaded from the knowledge base together with its associations. Therefore, external associations could only be assigned at runtime after the types were loaded and they could not be saved in the memory. The problem was that in an image, instances of types used. If an image consists of several object representatives and only the image is kept in the container, then the attributes of the different object representatives cannot be assigned to the object representatives in the knowledge base. For instance, if an image contained two object representatives of `<CAKE>` with the locations `<FAR:LEFT>` and `<NEAR:RIGHT>`, it was not possible to unambiguously assign the distance `<FAR>` to the `<CAKE>`

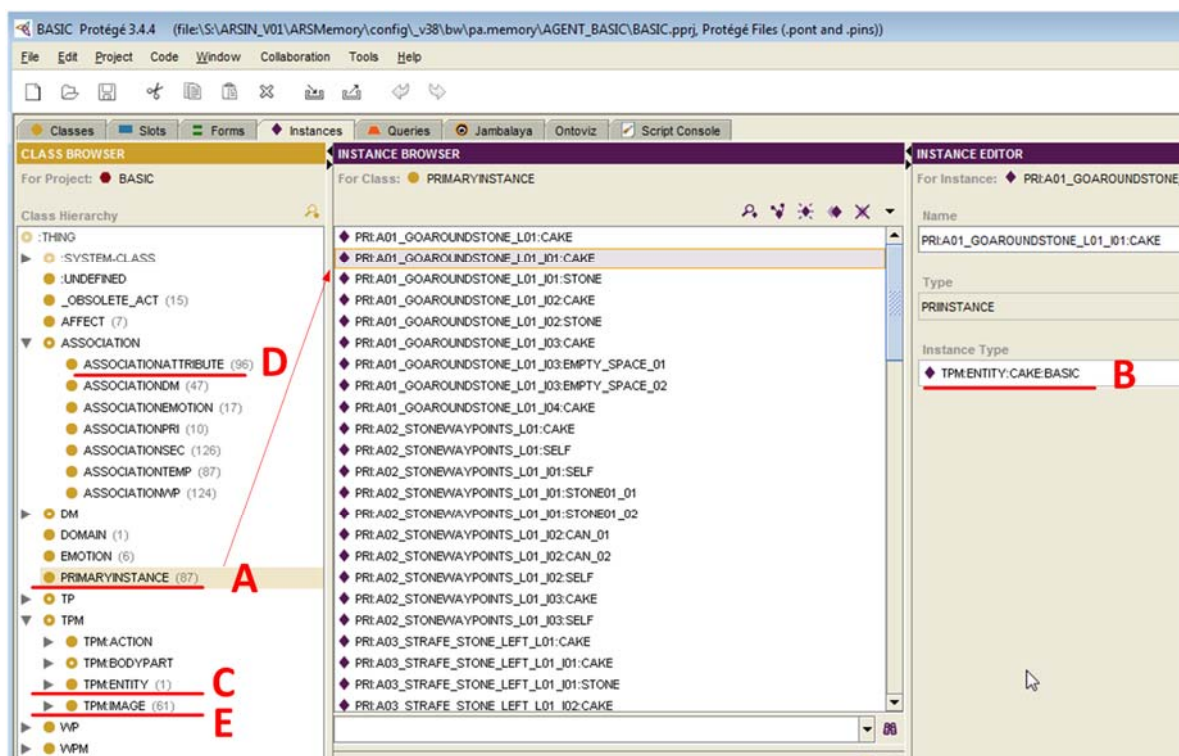


Figure 0.6: Screenshot of the knowledge base of SiMA, where the functionality of the PRIINSTANCE is described

with the position <LEFT> as there was no difference between the object representatives <CAKE>. As a consequence, all copies of that object representative would have to be assigned the same position. In order to solve that problem, a new helper structure was introduced, the `PRIINSTANCE`. Similar to images, which express experiences, it belongs to the episodic part of the memory and not the semantic part. At the moment, episodic and semantic memory is not divided. In Figure 0.6, it is shown how the `PRIINSTANCE` is used. In the Figure 0.6, the `PRIINSTANCE` <PRI:A01:GOAROUNDSTONE_L01_I01:CAKE> (letter A in the figure) is an instance of the type <CAKE:BASIC> (B), which can be found in <TPM:ENTITY> (C). At the instantiation of the memory in the initialization phase of the agent, the `PRIINSTANCE` <PRI:A01:GOAROUNDSTONE_L01_I01:CAKE> is created through a copy of the `clsThingPresentationMesh` <CAKE:BASIC>. Now, it is possible to store external attributes like position in the knowledge base and a position (D) (`clsThingPresentation`) it is not stored in the type.

General Data Structure Meshes

The previously introduced concepts allow data structures to be combined in graphs. At the time as the psychoanalysis was founded, the vision of psychoanalytic data structures looked like the graph in Figure 0.7 from [Fre91, p. 212]. Now, after 100 years, these data structures are reflected in SiMA through the preliminary work of [Lan10], [Zei10], [Deu11] and [Muc13] as well as this work.

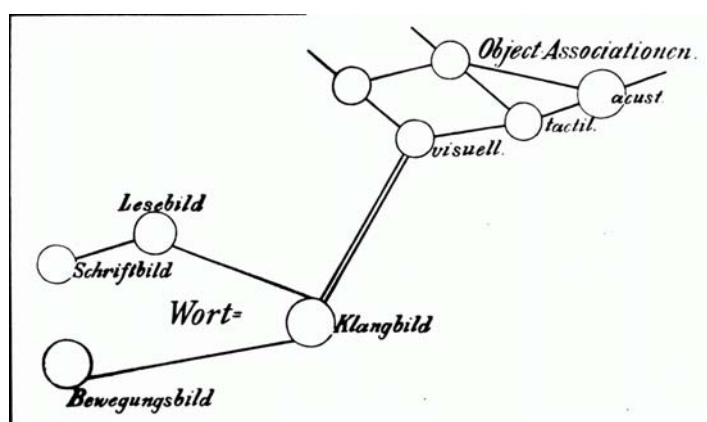


Figure 0.7: Psychoanalytical data structures according to Sigmund Freud [Fre91, p. 212]

In SiMA, the data structures now possess the full potential to form expressive meshes. On each level, from a single object representative to a complete act, data structures are organized in meshes. Due to this structure, it is possible to walk through the graph by following the associations.

The concept of making normal Java objects of the meshes has taken its most extensive form in the `clsWordPresentationMeshPossibleGoal`, which is an extension of the `clsWordPresentationMeshGoal`, which in turn extends the `clsWordPresentationMesh`. As `clsWordPresentationMesh` offers methods to transform it into a class with member variables, this was done with the possible goal (see Chapter

3.4.2). The implementation of it was done using the data structures available in the secondary process. A possible goal contains the following properties:

- Feelings: multiple graphs of `clsWordPresentationMeshFeeling`, which is structured like a `clsWordPresentationMeshGoal`
- Importance of efforts, social rules, drives, feelings, drive correction: a single `clsWordPresentation` for each double value
- Plan action: single graph of `clsWordPresentationMesh`
- Goal object: single graph of an object representative (entity) or image of `clsWordPresentationMesh`
- Goal name: single `clsWordPresentation`
- Conditions: multiple `clsWordPresentation` containing the strings with conditions
- Supportive data structure: single graph of an object representative (entity), image or act of type `clsWordPresentationMesh`

The properties make this structure a very large graph in total, as alone an act in the supportive data structures is a large graph in SiMA and the size has a negative effect on the performance of the system. Therefore, the very large graphs based on psychoanalytical data structures may be inefficient to use in SiMA, although they are very flexible and general. Further, the predicate logic used within the data structures of SiMA is expressed by implicit data structures, the meshes, which keeps the information in Java object instances.

Implementation of Psychic Spreading Activation

The concept of psychic spreading activation was first introduced in the SiMA project in this work in Chapter 3.2. Before the concept itself was implemented, the structure of the memory was refactored and simplified. Then the psychic spreading activation was implemented, which is easily accessible for every functional module with memory access.

Layered Memory Structure

The previous implementation of the memory access in SiMA has been described in [Zei10, pp. 104-109]. This concept was very general and was designed to differ between the primary process and the secondary process. Experience has shown that this approach was too complicated for the current needs of SiMA as many super classes only had one child that was used. In order to make it simpler and more usable, the KIS (Keep It Simple) principle was applied and all not used classes were removed and simplified. This removed around two levels of parent classes between the modules and the actual knowledge base.

A further necessary improvement in the reduction of complexity of the whole system was the introduction of an onion skin model in the memory access of SiMA through interfaces. This may have been the idea in the beginning of the SiMA design. However, the layered model was not strictly implemented. The introduction of a layered model, which separates the different conceptual layers in

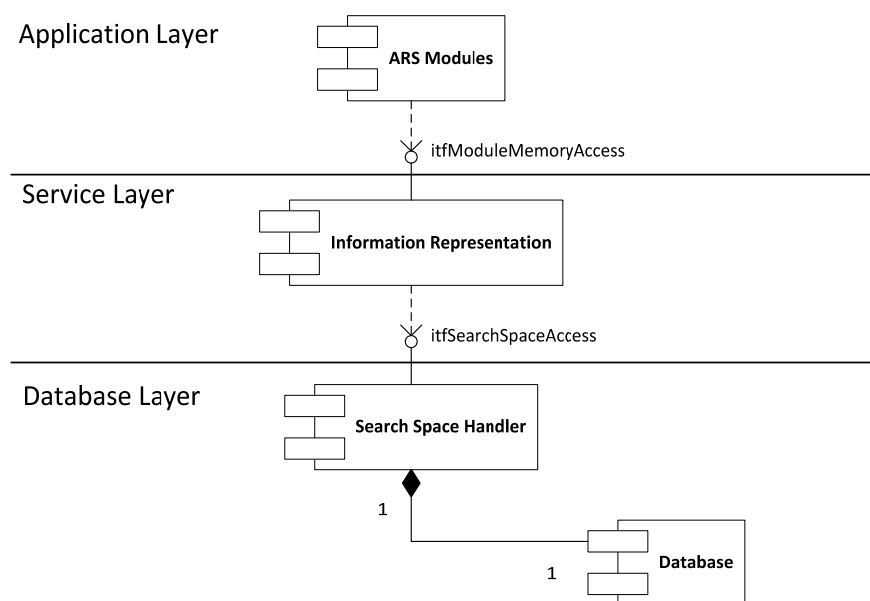


Figure 0.8: Component diagram of the layered structure of the long-term memory access in SiMA

the implementation is necessary, in order to make the memory access make the memory management in SiMA manageable as more data is added to the model. The biggest advantage of a layered model is to encapsulate the low-level functionality like database access and with that to reduce the complexity of the software. The simplified memory access will be described in the following.

The long-term memory access of SiMA is now strictly divided into three layers, which components and layers are shown in Figure 0.8 and in more detail in Figure 0.9. As it is a layered model, each layer can be exchanged without affecting the functionality of the other layers. The top layer holds the business logic of SiMA and is called the *Application Layer* in this work. It encapsulates the top-level modules of SiMA, which are shown in Figure 0.1 in Appendix B. The next layer, called the *Service Layer* in this work, contains more low-level functionality related to the memory retrieval. It is accessed through the interface `itfModuleMemoryAccess`. This functionality is defined by search methods and manipulation of retrieved data. Also, this is the place, where the execution of psychic spreading activation is located (see Chapter 3.2.4). The input data is provided by the functional modules in the Application Layer. Psychic spreading activation and other search methods, then retrieve data from a database through the interface `itfSearchSpaceAccess`, which is implemented by the *Database Layer*. In the Database Layer, the communication with the database is located, i.e. SPARQL statements or in the implemented case, the methods necessary to retrieve data structures from the imported Protégé Frames database. Protégé Frames is described in [Zei10, p. 105].

In more detail, the memory access is shown by a class diagram in Figure 0.9. Every module, which implements `clsModuleBaseKB` [Zei10, p. 103], automatically have access to the long-term memory through the method `getLongTermMemory()`. This is the case of the module Fusion with memory traces (F46). The interface `itfModuleMemoryAccess` provides several search methods. In the case that the properties of an object representative are known, the object representative is

identified by a search in the database with the method `searchEntity()` or `searchEntityFromInternalAttributes()`. The former method is primarily used in the functional module External Perception (F14) (see Figure 0.1) in the perception track described in Chapter 2.3.5, in order to identify perceived attributes, which are then formed into a perceived image. The latter method is primarily used for debugging purposes. Another related method that is provided is the `getSecondaryDataStructure()`, which is used to retrieve the corresponding associations to the secondary process data structure of a primary process data structure, i.e. to add the `clsAssociationWP` to a primary process data structure.

The method `searchMesh()` and `searchCompleteMesh()` are used to retrieve whole meshes and images from the long-term memory. `searchMesh()` is currently only used to load initial images into local buffers, like in the functional modules Libido Discharge (F45) and Emersion of Blocked Content (F35 in Figure 0.1) in the perception track of Chapter 2.3.5. `searchCompleteMesh()` is used to get associated data structures for an image. As input, any psychoanalytical data structure can be put, which is a part of the long-term memory. This data structure is then searched and its associations are added to the input data structure. This method cannot be used to activate data structures from an unknown input.

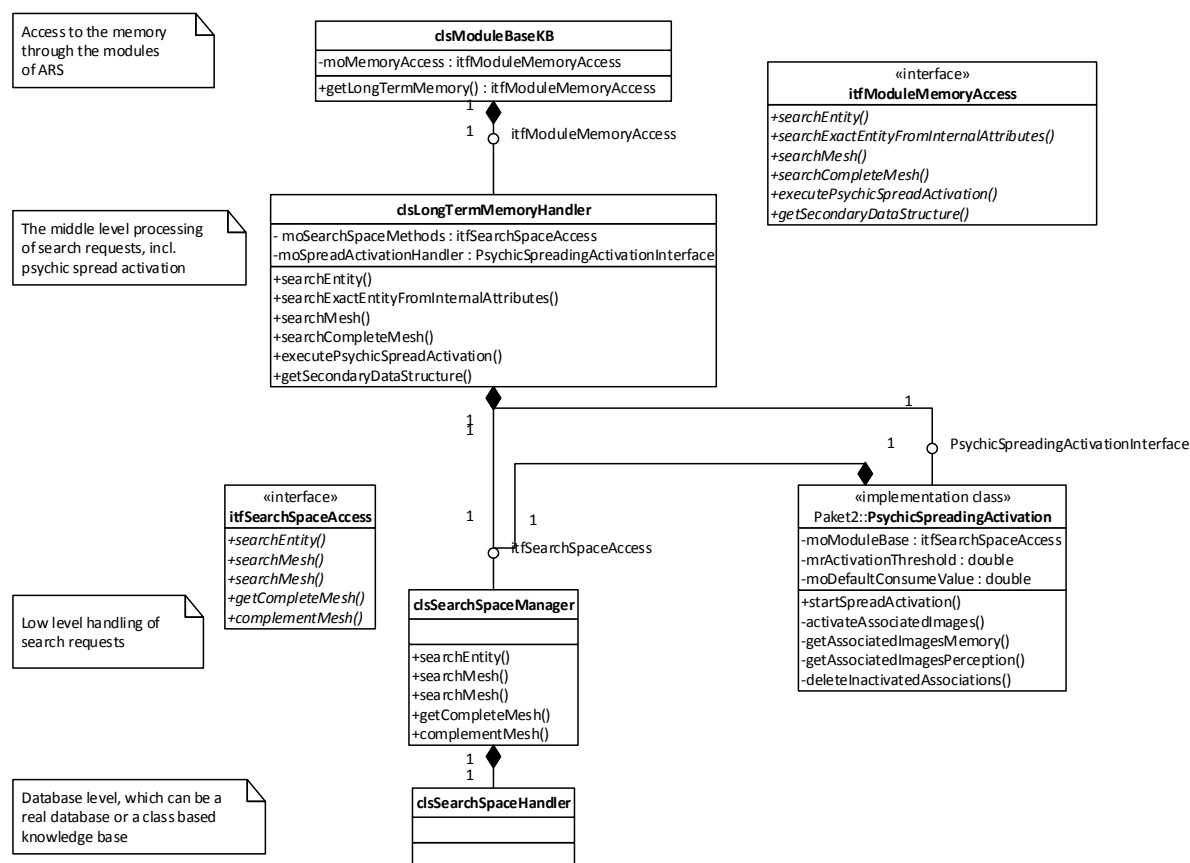


Figure 0.9: Class diagrams of the dependencies in the three layers of memory

In `searchCompleteMesh()`, also a new concept of search depth definition was introduced, the *search level*. Search level 0 means that only the data structure itself is loaded without any associations. It means that the input is returned. Search level 1 means that the data structure is enhanced with its internal associations as well as the directly associated external associations. In the case of an image it would mean that only the `clsThingPresentationMesh`, its emotions if there are any attached and its directly associated object representatives in the internal associations as `clsThingPresentationMesh` are loaded. The object representatives, which are a part of the image also get their drive meshes `clsDriveMesh` loaded, as they are a part of the object representative structure. Finally, associations with other images as `clsThingPresentationMesh` are loaded. Here the loading of data structures stops. The other images, which are associated with the input image do not contain much information and can be seen as stubs. First at search level 2, the properties and content of the externally associated images are loaded. Then it continues for search level 3,4,...n. The limitation with search level has proved to be very useful as it allows the caller of the method to define exactly how much of a mesh shall be loaded.

The last method of this interface is the `executePsychicSpreadingActivation()` that calls the implementation of the psychic spreading activation through the `PsychicSpreadingActivationInterface`. As can be seen in Figure 0.9, the `clsLongTermMemoryHandler` implements the class `PsychicSpreadingActivation`, which is an implementation of the `PsychicSpreadingActivationInterface`. It then calls the next layer through the `itfSearchSpaceAccess`, where low-level searching is performed.

In the lowest layer, which is accessible through the `itfSearchSpaceAccess`, the method `searchEntity()`, is used to find single object representatives based on some pattern defined in the higher layers. The two declarations of `searchMesh()` in Figure 0.9 with different parameters are used by the psychic spreading activation to activate images from the long-term memory based on the perceived image or at the initialization of local buffers. The `getCompleteMesh()` method and `complementMesh()` are both used to load more associations of already loaded images. While the former method loads external associations with other images, the `complementMesh()` only loads on the depth of the internal associations of the image. It can be interpreted as that the former method loads associations between images and the later method loads more details of the image itself. Both methods are needed for a working psychic spreading activation. The `clsSearchSpaceManager` implements the mentioned methods and directly accesses the knowledge base (see [Zei10, p. 103]) through the `clsSearchSpaceHandler`.

Memory Retrieval by Psychic Spreading Activation

Psychic spreading activation is implemented as a part of the memory retrieval, which is triggered by the functional modules of SiMA, which extend the class `clsModuleBaseKB` (see [Zei10, p. 103]). Currently, only the functional module Fusion with memory traces (F46) uses this implementation of psychic spreading activation.

Psychic spreading activation is addressed through the interface `itfModuleMemoryAccess` and its implementation `clsLongTermMemoryHandler`. It is accessible through the interface `PsychicSpreadActivationInterface`, which is implemented by the class `PsychicSpreadActivation`. This setup is illustrated in the UML class diagram of Figure 0.10. Except initialization, the only externally available function is the `startSpreadActivation()`, which initiates the search. In Table 0-1, all input parameters to the psychic spreading activation are listed. Some of them are already set at the instantiation of the class, but most of them are individually set on each start of the search process.

Parameter	Description
Default consume value	This value is set on instantiation of <code>PsychicSpreadingActivation</code> . It defines how much psychic intensity is consumed by each activated image according to Chapter 3.2.4 step 3. In this implementation, the same value is used for all images.
Threshold for image matching	This value is set at the instantiation of <code>PsychicSpreadingActivation</code> . It defines the minimum match as a double between 0.0 and 1.0, where a new association can be created between two images according to Chapter 3.2.4 step 1.
Source image on the primary input	Input image of type <code>clsThingPresentationMesh</code> according to Chapter 3.2.3.
Psychic intensity	Psychic intensity as described in Chapter 3.2.3.
Maximum number of images that can be activated by direct activation	It defines how broad the activation of direct associations is allowed to be.
Usage of direct activation	It is set true if the creation of direct associations through image matching as described in Chapter 3.2.4 step 1 shall be used or not. If not, then activation is only possible through already existing associations.
Current drive state	A list of all drive meshes according to Chapter 3.2.3.
Recognized image multiply factor	According to Chapter 3.2.4 step 3, there may be a secondary input besides of the primary input of the source image. Images with are identical, i.e. the same images, can receive an activation bonus as the psychic potential is increased.
Preferred images on the secondary input	List of images from the secondary input, which normally originates from the fantasy.
(Already activated images)	Psychic spreading activation is a recursive process. Therefore, automatically, all activated images are added to a list, which is checked every time an image is to be activated, in order not to activate the same image twice.

Table 0-1: Input parameters to the psychic spreading activation

Then, in a first step, according to Chapter 3.2.4, direct associations are created in the source image or indirect associations are loaded. In the case of the creation of direct associations (see Chapter 3.2.4 step 1), the source image is compared with all images of the knowledge base by using the method `searchMesh()` of `itfSearchSpaceAccess`. Those images, which matches are higher than the default threshold value are returned and associations are created with the source image. In that way, the source image is integrated as a part of the mesh of the long-term memory. In the case of indirect associations, resources are saved as only the already existing associations of the images are followed. As a result, the source image is associated with images from the long-term memory. Until now,

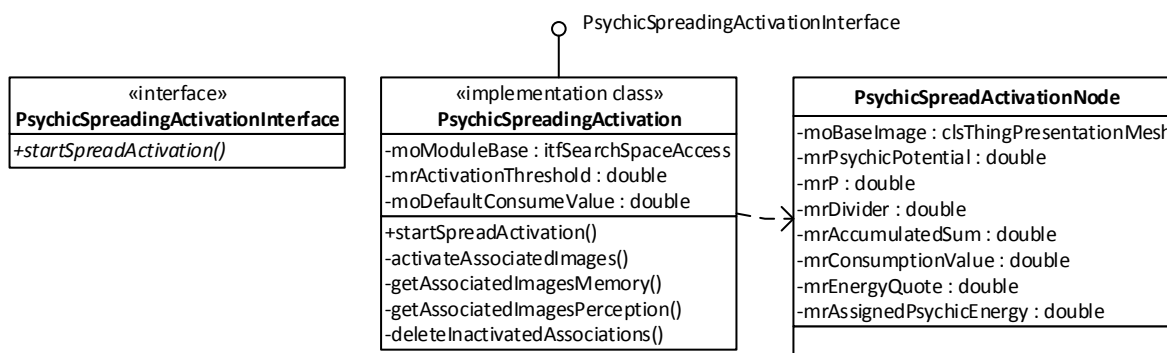


Figure 0.10: Class diagram of the package psychic spreading activation

nothing has been activated. However, clones of the associated memories are loaded from the long-term memory.

In the next step, it is calculated which of the associated images are actually activated. At this point, it is not known, which of them shall actually be used in the later processing. For each associated image, the node class `clsPsychicSpreadActivationNode` is created. It is an enhancement class for the `clsThingPresentationMesh` and it keeps the values and activations, which are used in the calculations in Chapter 3.2.4 step 4. Except the base data structure, which is to be activated, for instance, the current psychic potential and the assigned psychic intensity is kept within this structure.

Then, the activation is executed according to Chapter 3.2.4 step 4. Those images, which are not activated, are deleted. Further, as this is a recursive process, it has to be assured that the same image is not activated twice. Therefore, all input images are added to a list, which is checked before starting the activation for each new image. It is very important to keep track of the already activated images. If, for instance, there are three images, which are all connected to each other, there shall only be at most three activations. As soon as image 1, 2 and 3 are activated, the activation shall stop before image 1 is being activated once again. This recursion is done as long as any images can be activated, i.e. the input of psychic intensity > 0 for a node. The class structure of the psychic spreading activation is shown in Figure 0.10.

This implementation was explicitly adapted to be used as an additional layer on top of an already present memory search. As mentioned previously, it is necessary to load clones of all associated or associable images from the long-term memory and first then process them with the psychic spreading activation. This is necessary because the psychic potential has to be calculated for each image before it is activated and the psychic potential is dependent activation of a certain image is dependent on the psychic potential of the other associated images.

Implementation of Decision-Making on Abstraction Level 1

The decision-making process has been described in Chapter 3.4.3. In Figure 0.11, the same process is shown in high detail with all memory access and module connections.

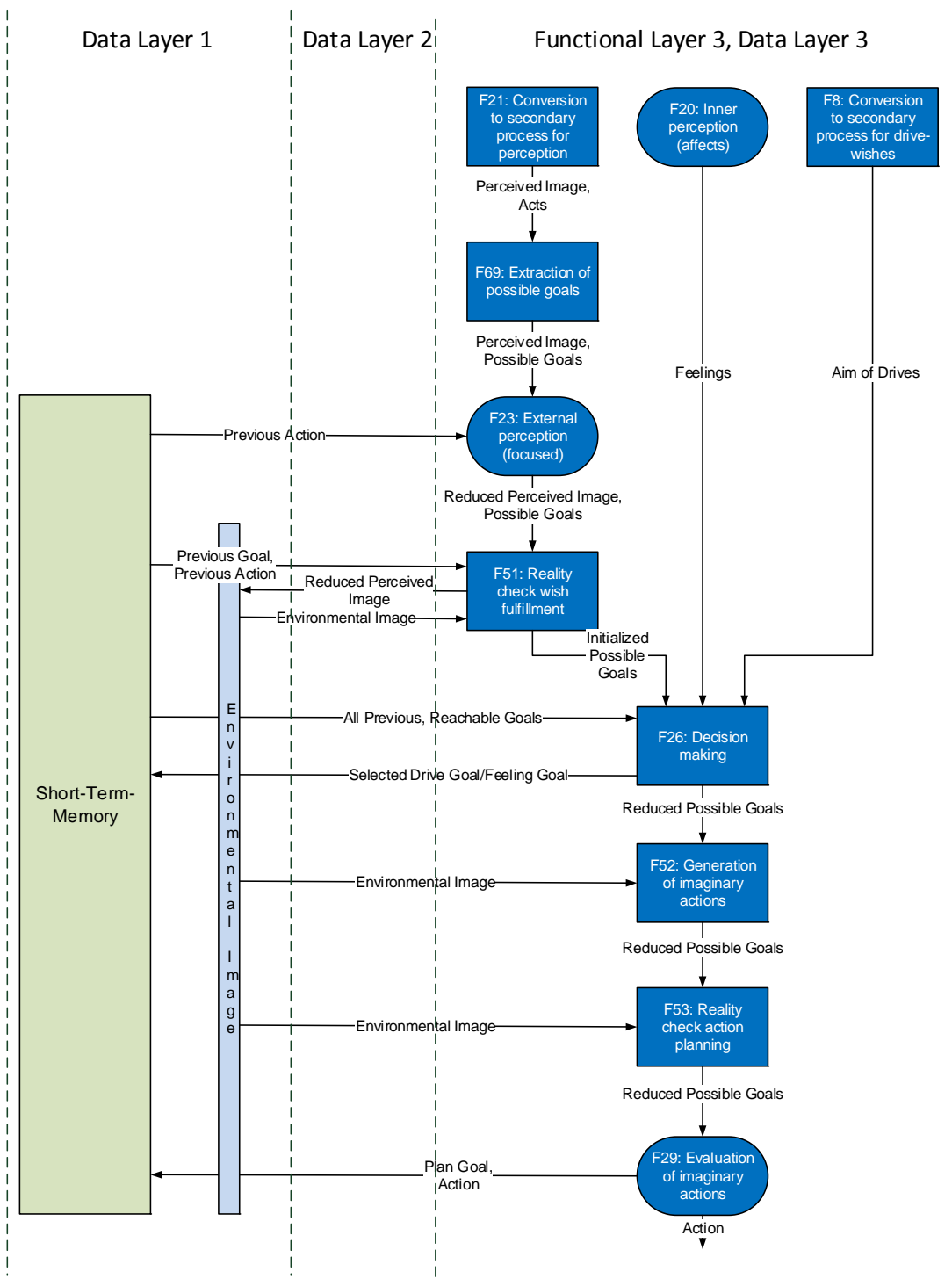


Figure 0.11: Sequence Diagram of decision-making in high detail

Codelets in Decision-making/Decision-Making in the SiMA Model

A problem that arises in the decision-making of a system, which can take internal and external actions, where inputs and outputs of processing functions differ very strongly and where the system has to keep track of the status of a goal, is to keep the system flexible and maintainable. This is a $m \times n$ -problem, where out of m input parameter one action from n possible actions shall be delivered. It could be possible to implement a lot of cases with if-else, but it would make the system complete rigid and unable to maintain. Therefore, a solution was searched for, where functions act more independently and are offered more like services. An input with some preconditions or attributes can then be matched to a certain processing function through a search in a catalogue. An interesting approach is the concept of codelets, which has been used in LIDA [FF12, p. 105]. A codelet is an individual piece of code, which is run independently if certain conditions are fulfilled. In LIDA, the main difference to a normal production rule is that a codelet runs in its own thread and it acts like a daemon, where it checks its own conditions. Production rules are centrally managed and started. In SiMA, the concept of LIDA with running daemons will not be used as the software architecture of SiMA is strongly serial and not based on parallel processes like in LIDA. In of SiMA, there are several specialized, functional modules that are solving different sub-problems. Therefore, the concept of codelets is suitable as a reduction of complexity by putting specialized functions in separate codelets and minimizing dependencies.

Codelet

In SiMA, a codelet (`clsCodelet`) is an independent piece of code, which is run if some conditions are met. Just like a production rule, in Figure 0.12, a UML diagram of the codelet structure is shown. On the top, the abstract codelet, followed by the extension of the different types of codelets and at the bottom, two examples of codelets that have been implemented.

All codelets possess the following components:

- Name: Name used to identify the codelet
- Description: String description of what the codelet does
- Goal: The goal, which is processed
- Working memory access: Access the common workspace of the secondary process, in order to load information from previous cycles
- Environmental Image access: Access the environmental image for getting the position of object representatives
- Preconditions: Attributes (conditions) that must be set (true) in the goal, in order to execute the function of the codelet
- Post conditions: Attributes, which are optionally set in the goal after the execution of the function
- Code to be run: Any code

A codelet is activated or runs if the attributes of the goal match the preconditions, which are defined for the codelet. Then, the goal itself is used as a parameter in functions, which alter its state by modifying its attributes. It results in the post conditions of the codelet. At the moment, the pre- and

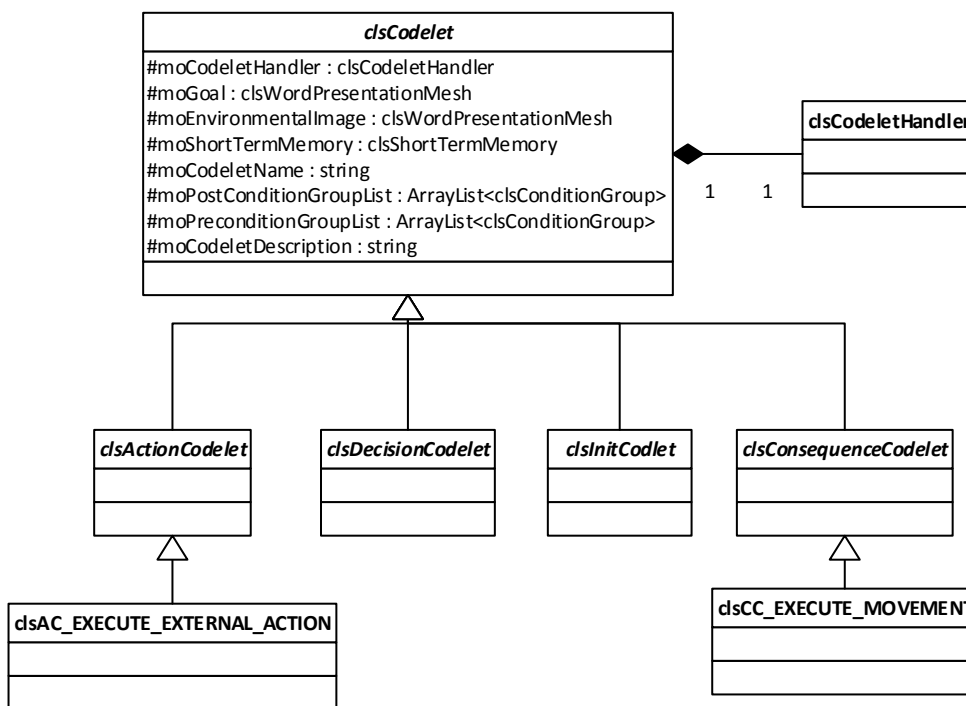


Figure 0.12: Class diagram of a codelet

post-conditions are hardcoded into each codelet and the reasoning is only forward-chaining, i.e. only preconditions are used to match the current attributes of a goal. In case of backward-chaining, post conditions would be useful. In the future, they are supposed to be stored as explicit knowledge, which would open the way for learning procedures. It could be realized by defining the end state of a certain goal. Then, the post conditions of codelets are searched for matches. The preconditions of a codelet with the best match are then searched for matches. In that way, a chain of codelets could be defined and executed.

Basically, goals are processed in four phases during decision-making: The initialization (*clsInitCodelet*), the application of a consequence of an action (*clsConsequenceCodelet*), the decision what to do next (*clsDecisionCodelet*) and the action that shall be set to be executed (*clsActionCodelet*). In Figure 4.7 and Figure 4.8, the location of the processing steps of the goals within the decision process is shown and each processing

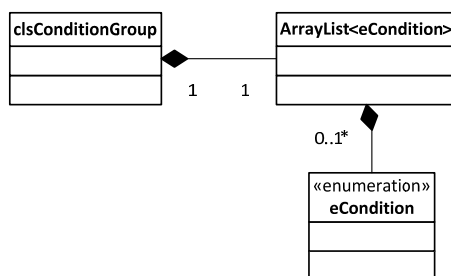


Figure 0.13: Class diagram of the condition group

step can be seen in Table 4-2, Table 4-1 and Table 4-3. In an equivalent way, the codelet is an abstract type and four specialized codelets extends it. As seen in Figure 0.12, each of these sub codelets has certain special inputs adapted for that type of codelet.

In each codelet, the attributes or conditions are arranged in groups (`clsConditionGroup`), similar to the pattern in [WDB12, p. 4]. Figure 0.13 shows a UML diagram of a condition group. A codelet keeps a list of condition groups and within a condition group, the attributes are defined as `eCondition`. While all attributes within a condition group must match the attributes of a goal, only one condition group must match to activate the codelet. It enables the creation of logical AND and logical OR of conditions. However, at the current state, it does not allow logical NOT.

Codelet Handler

The role of the codelet handler (`clsCodeletHandler`) is to test an incoming goal on all registered codelets and to execute those codelets, which match the attributes of the goal. It also connects the codelets with their environment, i.e. necessary information like previous goals or drive inputs. In Figure 0.14, the UML diagram of the `clsCodeletHandler` describes its structure. In SiMA, only one instance of `clsCodeletHandler` is needed and at the instantiation of the SiMA agent, it receives references from the working memory (`clsShortTermMemory`) and the environmental image memory (`clsEnvironmentalImageMemory`), which have been described in Chapter 3.4.4. Each codelet is then registered in the codelet handler and depending on its type, they are assigned one of the four processing units.

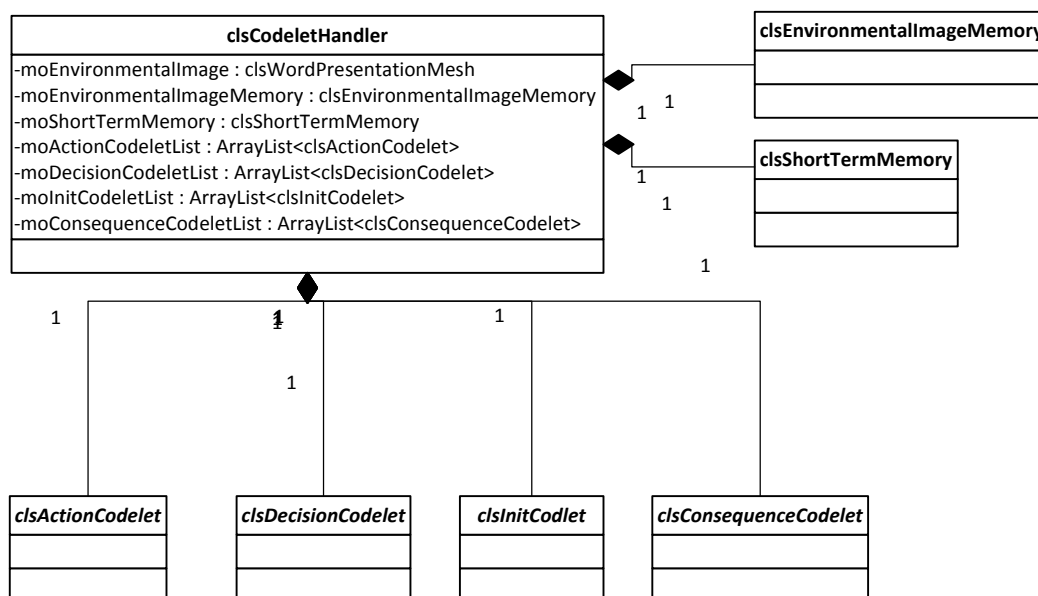


Figure 0.14: Class diagram of the `clsCodeletHandler`

In Figure 0.15, the interaction between the SiMA architecture and the codelets is described. Within the implementation that is based on the functional model of Figure 0.1, only the modules Reality check wish fulfillment (F51) and Generation of imaginary actions (F52) have access to the codelets. Just like

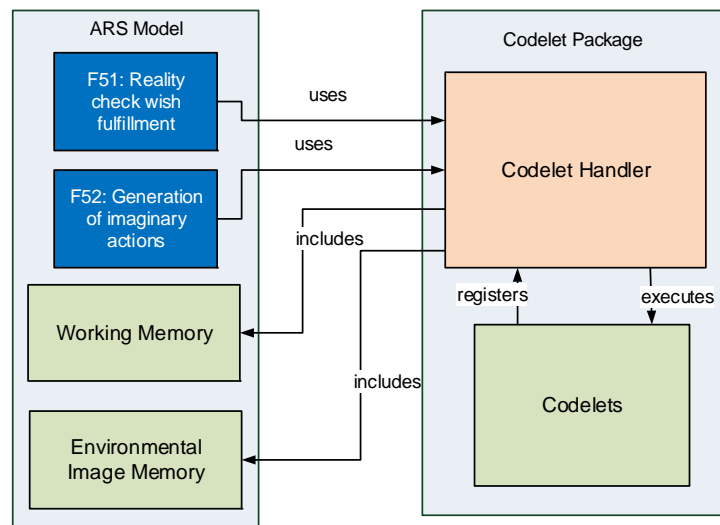


Figure 0.15: Conceptual view of how codelets are integrated in SiMA

the process in Figure 4.7 and Figure 4.8 describes, the `clsInitCodelet` and the `clsConsequenceCodelet` are located in front of the module Decision-making (F26) and `clsDecisionCodelet` and `clsActionCodelet` are located afterwards.

Codelet Interaction with the SiMA Architecture

In the SiMA architecture, the modules F51, F26, F52 and F29 are run through in sequence. In Figure 0.16, this sequence is illustrated in a UML sequence diagram. The interactions with the codelet handler takes place in the previously mentioned modules.

A lesson learned from the implementation of codelets is that it is purposeful to let one codelet do one thing and not to let one codelet do many things. In case of extension of the system, at some point, the codelet that has to do many things has to be split up as one of the functionalities are needed in several places. It also makes the management and maintenance of the codelets a lot easier if they only do one thing for certain combinations of trigger conditions. This is also the policy of the production rules in SOAR (see Appendix A). According to [TCM03, p. 5], ACT-R allows multiple operations to be performed within one production rule and the consequence is a less abstract and flexible system.

Another lesson learned, which has its equivalence in SOAR compared to ACT-R is that the codelets shall ideally only modify the state of a goal, just like the production rules in SOAR only modify the state of the working memory. In SiMA both is possible. A codelet can do anything in the architecture as well. However, if codelets do both altering the state of the goals and perform actions in the architecture, it gets pretty messy and it makes the bug tracking harder. Therefore, ideally, no codelets can do anything in the architecture, but only provide states, which are read by functions in the architecture.

Conclusively, codelets in SiMA provide an efficient way of transforming a complicated system into a complex system, where each codelets can be isolated. In that way, complexity in the architecture was reduced. Functions were removed from the modules and they were encapsulated them into separate,

independent building blocks. It offers an appropriate solution to the problem that one goal can be modified in various ways and that these functions do not have to be implemented within the module itself. By using activation triggered by preconditions that are linked to the goals, the execution of functionality in a certain order is no longer described by the architecture, but through the data. Here, data are the preconditions. The architecture becomes more of an interpreter of data and that means much higher flexibility. Functionality can be added and removed from the functional modules without having to change the code within them. Only new codelets have to be registered or unregistered and the order in which they are run can be kept as knowledge instead of hard-coded functionality.

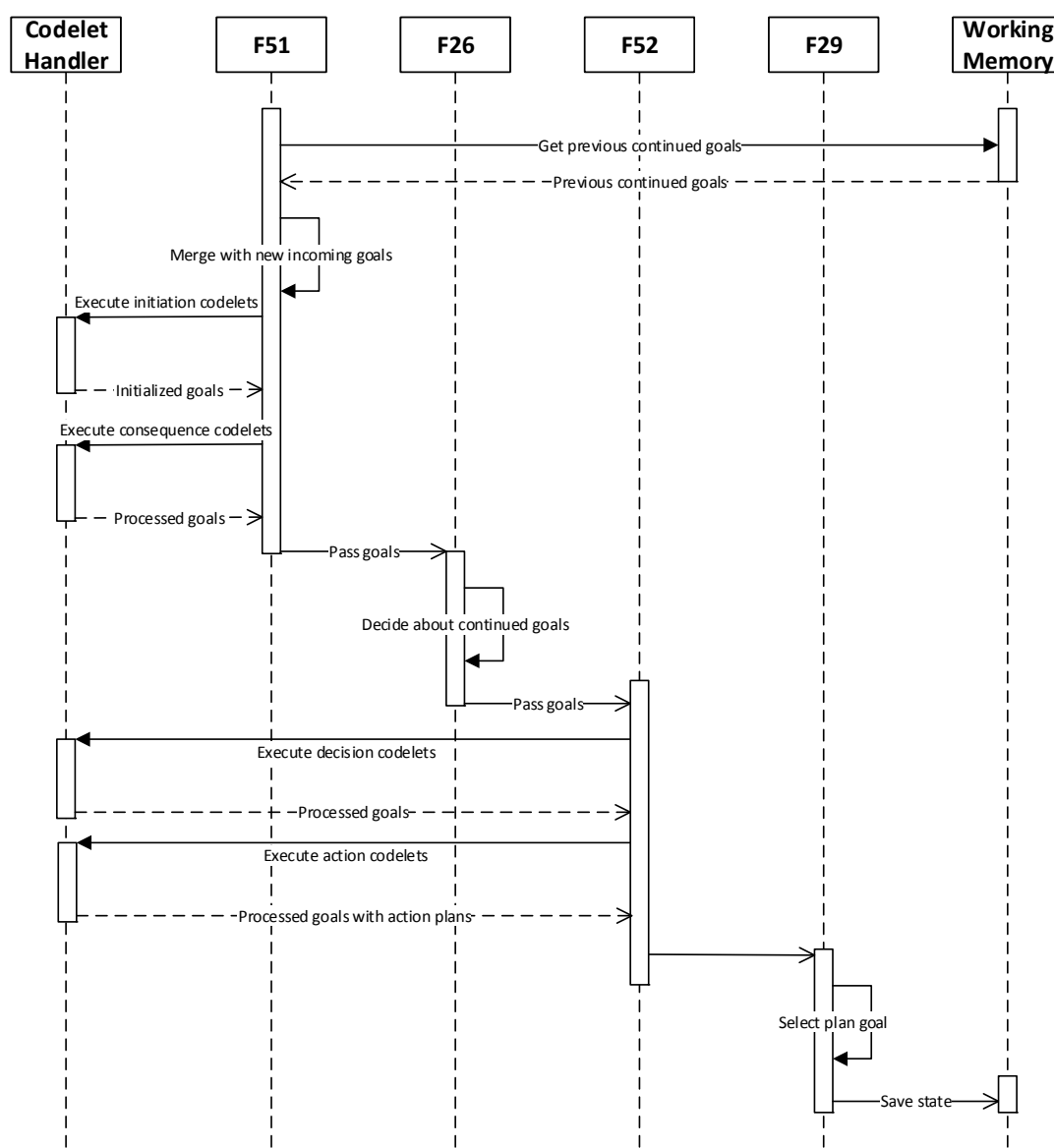


Figure 0.16: Sequence diagram of the usage of codelets in the secondary process

Appendix C – Example Calculations

In the following appendix, examples are presented, which are supposed to increase the understanding of the concepts or the implementation.

Example of Activation of Template Images

The formulas and abstract concepts of psychic spreading activation of Chapter 3.2.4 are calculated through in an example. It is supposed to simplify the understanding of the concept. The example is based on the values of Figure 0.1, which is equal to Figure 3.11.

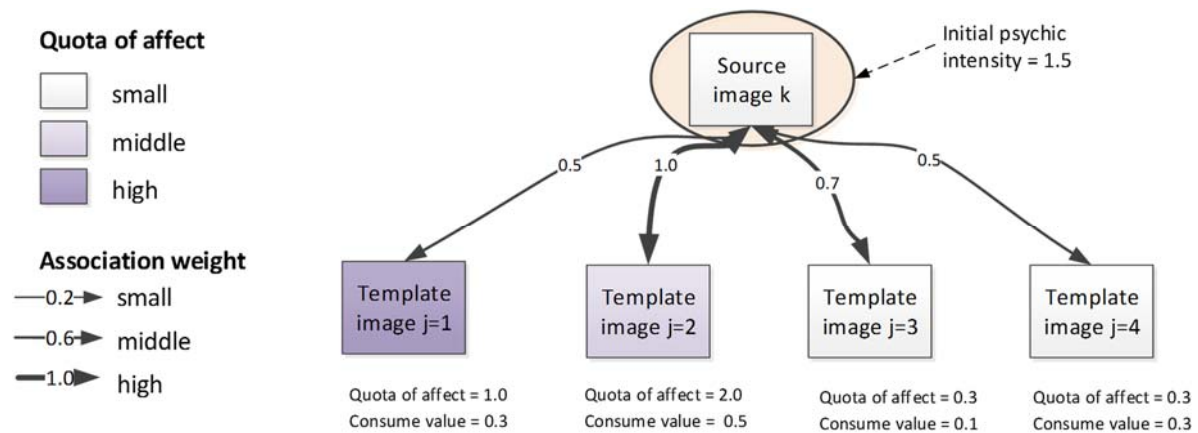


Figure 0.1: Example of calculation of the spreading from a source image to template images

As a support for overview, Table 0-1 is created with the following attributes:

ψ_j : The psychic potential of all associated image j .

P_j : The sum of psychic potentials of other associated images.

$\sum_0^n \psi_i$: The sum of the denominator in equation (3-4), where the sum of all higher ranked psychic potentials in the table is included.

$P_j \geq \sum_0^n \psi_i$: The activation condition of an image.

Sort order considered: If the activation is once interrupted, the whole activation process is interrupted.

Allocated psychic intensity quote: Quote of the available psychic intensity will be allocated by that image.

Allocated psychic intensity: Based on the allocated psychic intensity quote, the actual allocated psychic intensity.

The calculation of activation of images is shown with an example in Figure 3.11. The available psychic intensity is 1.5, image matching has returned direct associations to four images with the association weights 0.5, 1.0, 0.7 and 0.5 for <Template Image j=1> to <Template Image j=4>. The weighted quota of affect of the images are set to 1.0, 2.0, 0.3 and 0.3. The psychic-intensity-consume-values are set to 0.3, 0.5, 0.1 and 0.3. As there are no other images from the fantasy, the reinforcement factor is constantly set to 1.0.

For each image, the psychic potential is calculated according to equation (3-1):

$$\psi_1 = 0.5(1 + 1.0) = 1.0$$

$$\psi_2 = 1.0(1 + 2.0) = 3.0$$

$$\psi_3 = 0.7(1 + 0.3) = 0.91$$

$$\psi_4 = 0.5(1 + 0.3) = 0.65$$

It can be seen that <Template Image j=2> has the highest psychic potential because of its perfect match with the source image and its high weighted quota of affect. As a consequence, this image will allocate the most psychic intensity. In the following, P_j is calculated according to (3-5):

$$P_1 < 1.0 \left(\frac{1.5}{0.3} - 1 \right) = 4.0$$

$$P_2 < 3.0 \left(\frac{1.5}{0.5} - 1 \right) = 6.0$$

$$P_3 < 0.91 \left(\frac{1.5}{1.0} - 1 \right) = 0.45$$

$$P_4 < 0.65 \left(\frac{1.5}{0.1} - 1 \right) = 9.1$$

Then the table can be completed, in order to calculate, which images will be activated.

Image	Ψ_j :	P_j :	$\sum_0^n \Psi_i$	$P_j \geq \sum_0^n \Psi_i$	Sort order considered	Assigned psychic intensity quote	Psychic intensity quote
2	3.00	6.00	3.00	TRUE	TRUE	0.75	1.13
1	1.00	4.00	4.00	TRUE	TRUE	0.25	0.38
3	0.91	0.45	4.91	FALSE	FALSE	0.00	0.00
4	0.65	9.10	5.66	TRUE	FALSE	0.00	0.00

Table 0-1: Example calculation of activated images from a certain source image