CONCEPTUAL DESIGN OF THE MEMORY SYSTEM OF THE ROBOT COGNITIVE ARCHITECTURE ARMARX

Fabian Peller-Konrad, Rainer-Kartmann, Christian R. G. Dreher, Andre Meixner, Fabian Reister, Markus Grotz, Tanim Asfour
High Performance Humanoid Technologies (H²T)
Institute of Anthropomatics and Robotics (IAR)
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
{fabian.peller-konrad, rainer.kartmann, c.dreher, andre.meixner, fabian.reister, markus.grotz, asfour}@kit.edu

Abstract
We consider the memory system as a key component of any technical cognitive system that can play a central role in bridging the gap between high-level symbolic discrete representations used for reasoning, planning and semantic scene understanding and low-level sensorimotor continuous representations used for control. In this work we described conceptual and technical characteristics such a memory system has to fulfill, together with the underlying data representation. We identify these characteristics based on the experience we gained in developing our ARMAR humanoid robot systems and discuss practical examples that demonstrate what a memory system of a humanoid robot performing tasks in human-centered environments should support, such as multi-modality, introspectability, hetero-associativity, predictability or an inherently episodic structure. Based on these characteristics, we extended our robot software framework ArmarX into a unified cognitive architecture that is used in robots of the ARMAR humanoid robot family. Further, we describe, how the development of robot software led us to this novel memory-enabled cognitive architecture and we show how the memory is used by the robots to implement memory-driven behaviors.

Keywords: Humanoid Robotics, Memory-driven Cognitive Architecture, Working Memory, Episodic Memory, Long-term Memory, Knowledge Representation

1 Introduction
The goal of embodied intelligence is to create robot systems with cognitive and sensorimotor capabilities that are comparable to the human’s, especially regarding learning and development. But what are cognitive abilities and how do they emerge? How do they influence the way we learn? What components are needed to endow a robot with cognitive abilities? And how can we cast these components into software modules that are usable in robotics? To provide an answer to the first question, [1][2] describes cognition as “the process by which an autonomous system perceives its environment, learns from experience, anticipates the outcome of events, acts to pursue goals, and adapts to changing circumstances”. This definition identifies three core components that are necessary for a cognitive system:
1. A cognitive system needs *perception* components in order to perceive the environment.

2. A cognitive system needs *processing* components in order to learn from experience, to anticipate the outcome of actions and to adapt to changing circumstances.

3. Finally, a cognitive system needs *action execution* components in order to purposely act to achieve goals.

In addition to these core components, a robot cognitive architecture requires a place to hold information acquired from perception and action execution. The processing components then access and process this data and eventually store the results back in this data storage.

This motivates our first hypothesis: *The memory mediates between high-level abilities and low-level components of a cognitive system.* We locate abilities such as language and scene understanding, planning, reasoning or plan execution monitoring on the highest layer. Sensorimotor control and sensor data processing are part of the lowest layer. This means that the memory system must be able to process a huge amount of data—no matter if the data is symbolic (e.g. plans, words, relations, etc.) or sub-symbolic (e.g. images, joint configurations, forces, etc.). It bridges between the sub-symbolic representation of the lower level and the symbolic representation of the higher level, tackling the *signal-to-symbol-gap* known in embodied intelligence.

However, the memory system should not be seen as a simple, passive storage device. It is an active part of an agent’s cognitive architecture which is highly influenced by the current context. The context not only influences what we store and forget but it also influences how we store information. Highly emotional situations will create much stronger memories which decay much slower than memories taken from a standard situation. In particular, the memory must play a key role in deriving symbols from sub-symbolic multi-modal information by allowing to learn from experience, from interaction with the environment and by trial and error. This means that the information stored in the memory must be introspective, i.e. the memory must be able to analyze the information, adapt its behavior to the given data and possibly discard knowledge because that knowledge is redundant or not necessary for the current situation.

This observation leads to our second hypothesis: *Multi-modal representations are key.* The ability of storing multi-modal information, the efficiency of storage and retrieval or the ability to learn require a meaningful and an efficient representation of information. This representation must be specific enough to differentiate between symbolic and sub-symbolic data as both modalities. On the other hand, the representation must also support generalization. Further, it must support *associations of knowledge* because the system needs an understanding of how perception and action are coupled and which sensations usually occur together. With multi-modality we explicitly include meta-cognition, i.e. results of cognitive processes which are not taken from the environment through sensors.

As already motivated, time-series information is of particular importance for a cognitive system. In humans, the episodic memory is concerned with the recollection, organisation and retrieval of time-series information, i.e. personally experienced events occurring at a particular time, place and context. The knowledge about time-series information together with the ability to inspect the information gives the memory the ability to reason about conditions of past events, to explain why a subject acted in a particular way, or to predict how information might change in the future.

In this paper, we identified five conceptual characteristics of memory systems, shown in Figure 1. We promote the need of cognitive control architectures for complex robotic systems such as humanoid robots. We present a novel memory system implemented in our robot software framework ArmarX, see also [7]. ArmarX itself is built upon the idea that consistent disclosure of the system state strongly facilitates the

[https://armarx.humanoids.kit.edu](https://armarx.humanoids.kit.edu)
Figure 1: Identified conceptual characteristics of memory systems.

Figure 2: The memory-driven architecture of the cognitive robot software framework ArmarX, see [https://armarx.humanoids.kit.edu](https://armarx.humanoids.kit.edu) and [7]
development process of distributed robot applications. As shown in Figure 2, ArmarX implements a classical three-layer architecture for cognitive robotics. As motivated before, the memory is located between the high-level abilities such as scene and language understanding, task planning or action execution and the low-level components like real-time control, sensor data processing. To support the efficiency of a memory system we argue that a memory should not be seen as a centralized component of the architecture. All components must be able to run on different machines in parallel, making the process of transferring data from a producer to the memory as easy as possible and ensuring the availability of the memory.

Through a distributed design, the memory system manages the information flow in the robot software framework efficiently and transparently. Our data representation supports efficient encoding, introspection and therefore facilitates automated predictions as well as multi-modal associations. Data, both symbolic and sub-symbolic, are stored in a highly structured manner — logical segments, which hold e.g. information about perceived object instances, object classes and agent, including the robot itself. In particular, data can be associated to other instances of the memory, e.g. an extracted pose of the human can be linked to the image it was calculated from.

Inspired by memory taxonomies in cognitive psychology [8], our memory system can be divided into four parts:

1. **Sensory Memory (SM)** where data is held for a very short duration until it may be passed to the Working Memory.
2. **Working Memory (WM)** that holds the current state of the world and the robot’s internal state. It can be updated by the SM through perceptual processes, by cognitive processes (e.g. recalling an episode from the Long-Term Memory) or prior knowledge. Information in the WM is held for only a limited time. If it is not consolidated into Long-Term Memory, it is forgotten.
3. **Long-Term Memory (LTM)**. Compared to the WM, the LTM provides long term storage capabilities and encodes the information into a more graduated representation. This representation focuses on the generalization of the data, as it allows multiple data points to be summarized. Nevertheless, the representation should still support introspection and thus predictive ability.
4. **Prior Knowledge (PK)** contains information which is already known to the robot. During startup, the WM gets initialized with this known data such as 3D models or features for object detection which usually refer to user generated data.

We argue, that keeping track of temporal changes is of special importance for a memory system. This is also true for semantic (e.g. the knowledge that the object “apple” is a “fruit” or which force to apply to grasp a fragile wine glass) and procedural (e.g. the knowledge how to open a door) knowledge. Therefore, in our novel memory architecture, all information is stored episodically.

The remainder of this paper is structured as follows: Section 2 gives an overview over the memory classes known in cognitive psychology, based on which memory systems of cognitive architectures for technical systems are often modeled. In addition, we will compare several cognitive architectures with a special focus on their consideration and technical implementation of the memory. In Section 3 we further explain how the here motivated concepts are embedded in our memory system.

## 2 Related Work

In this section, we give the reader a rough overview of the distinction between memory structures, without going into detail about the neuro-scientific aspects of memory, such as which areas of the brain are relevant. Further, we discuss what we consider to be the most relevant artificial cognitive architectures in the context
of robotics which are still actively developed. We analyze the related work with respect to the identified core characteristics of memory system, presented in Figure 1. For a general review on cognitive architectures for artificial intelligence, we refer to [9, 10, 11].

2.1 Memory

By studying amnesic patients and animals in the second half of the 20th century, researchers found that different areas of the brain are responsible for different memory tasks. This motivated the assumption that the memory consists of several subsystems [8]. Findings from the medical history of famous patients such as Henry Gustav Molaison or Clive Wearing greatly influenced the development of cognitive psychology and theories that attempt to explain the connection between brain function and memory.

Based on these findings, [12] proposed the so called Multi-Store Model. This model introduced three types of memory: A sensory memory which processes perceptual information. A long-term memory which holds information for a long duration and a short-term memory (also known as working memory) which holds information through repetitive rehearsal and which receives information from the sensory memory through attention and from long-term memory through retrieval. While such a clear distinction between different memory types is questionable for biological systems [13], it provides a basis to structure and classify artificial systems. Figure 3 shows an extended version this taxonomy of memory classes. Data is passed from sensory memory over working memory to long-term memory. The long-term memory itself can be subdivided into declarative memory and procedural memory (as a part of non-declarative memory). While in biological systems non-declarative memory also consists of priming, non-associative learning and procedural memory, robotic applications usually only explicitly implement the latter.

2.1.1 Sensory Memory

The sensory memory (SM) holds perceptual information for a very brief moment. It acts as a repository for incoming sensory information. In biological systems, sensory receptors take e.g. visual, auditory or touch information and forward it directly to the nervous system for further processing. Iconic memory refers to the visual information stored in this this short-term cache and it is thought to hold information for less
than a second. Aural information is stored in a so called echoic memory. The echoic memory is assumed to have a longer decay rate than the iconic memory \[14\]. Information related to touch interactions but also proprioceptive information (e.g. joint angles and relative position of the body) are stored in a so called haptic memory in which information decays after around two seconds.

The sensory memory is assumed to be outside of cognitive control. It is a highly volatile storage containing raw, unanalyzed data that is derived from the senses. The data is just stored long enough to be passed to the working memory (WM). To limit the amount of data transferred to the WM, data is only transferred when it is attended to, and is lost otherwise.

### 2.1.2 Working Memory

Similar to the SM, the working memory (WM) holds information for a limited amount of time. However, the duration of information residing in the working memory is much longer (in the order of seconds). In addition, the WM can consciously be controlled by putting attention and is therefore important for reasoning, learning, problem solving, and other mental processes. The ability to have certain details ready, even if they are not yet stored in long-term memory, supports a variety of everyday mental processes at a fundamental level. Examples include remembering the first part of a sentence to understand the second part, keeping a number in mind while solving a mathematical problem or remembering where you just saw an object.

While early works on models of the WM were motivated by delayed memory experiments where only one item had to be retained in memory, newer tests showed that people are able to keep track of several items at a same time \[15\]. This motivated the assumption that the WM has a relatively small capacity of \(7 \pm 2\) chunks \[16\], when not using exploits like repeating information out loud, regardless of whether the elements are digits, letters, words, or other units. Newer research differ between the modality (e.g seven chunks for digits, six for letters, and five for words) \[17\]. \[18\] proposed that the WM only has a capacity of about four chunks in young adults. The complexity of information stored in each chunk differs from person to person. While most adults can only repeat about seven digits correctly, some people have shown impressive improvements of being able to repeat up to 80 digits. This improvement can be achieved through massive training, e.g. learning to improve the way data is grouped together and how these groups are encoded into individual units. According to \[19\], the total amount of chunks can not be increased - we can only increase the complexity of the referred information. The WM is not only exclusive to humans as animals have also shown similar abilities such storing and maintaining several items simultaneously in memory, remembering their order and manipulating them \[15\].

### 2.1.3 Long-Term Memory

The long-term memory (LTM) is intended for storage of information over a long period of time. Through the process of repetitive rehearsal and association, memories of the WM consolidate to the LTM or an existing memory in the LTM gets reinforced \[12\]. The WM can then retrieve the data from the LTM when it is needed for processing. During the process of consolidation, data is encoded into a special representation to identify groups and to generalize the knowledge. However, memories stored in LTM are not saved in a static state. Studies showed that memories in LTM are transformed every single time they are accessed \[20\].

According to \[8\], the LTM can be divided into two types. The declarative memory (also known as explicit memory) contains information such as facts and events and is managed through conscious control. The non-declarative memory contains implicit knowledge, such as the ability to perform various actions or behavioral control parameters.

**Declarative Memory**
The declarative memory can be further subdivided into two types. The semantic memory consists of general knowledge about the internal state and the environment such as facts, ideas and concepts. In the context of an artificial systems, this part of the memory is usually assumed to be symbolic [9]. In comparison, the episodic memory [5] contains episodes or autobiographical experiences occurring in an explicit spatial and temporal context (i.e. what, where, and when something happened). The knowledge of how information has changed in the past in the episodic memory as well as the knowledge about facts in the semantic memory form the basis for prediction and explanation. There is a strong coupling between the two, as we derive new concepts from the experiences we have stored in the episodic memory [21].

Non-Declarative Memory

The non-declarative memory covers all information that is not consciously accessible. In the literature of cognitive psychology several subsystems are identified [8]. Priming describes the ability to strongly accelerate the retrieval of information from long-term memory by a related stimulus. It requires that knowledge can be associated heterogeneously (no matter how the knowledge is represented). Priming can be further subdivided into positive and negative priming, semantic priming, perceptual priming, conceptual priming. In classical conditioning, different stimuli are linked together. A well-known example of classical conditioning is Pavlov’s dog [22], which showed increased saliva production just by ringing the dinner bell. The neutral stimulus “ringing of the bell” was thus linked with the positive stimulus “there is food”, which triggers a physical reaction. Non-associative learning is the simplest form of learning as it does not require stimuli association [23]. Habituation and sensitization are the two forms of non-associative learning. Habituation describes the process of inhibiting a response after repeated exposures of a stimulus. The degree to which a response is inhibited depends on the repetition rate of the stimulus, its intensity, the duration of the stimulus, and how often the person is exposed to the stimulus. On the other hand a sensitized stimulus has increased intensity and sensitization does not require repetitive stimuli. Even a single stimulus may cause a reinforced response. For example, relapse of addiction can be seen as sensitization. Even a few stimuli, e.g. from drugs or gambling, can trigger a strong physical desire. The procedural memory is required for skilled behavior and habits. There is less known on how we store skills and abilities except that skills are learned and refined through practical training and that in learning a skill a variety of areas of the brain are involved.

Artificial cognitive systems usually only explicitly model the declarative memory and the procedural memory. Priming and non-associative learning are usually not modelled or at least there is usually no dedicated module to store such knowledge in the long-term. However there exist approaches that focus on how an artificial systems can be conditioned and how this mechanism can be used in social robots [24].

2.2 Memory Systems and Artificial Cognitive Architectures

The development of artificial cognitive architectures is a longstanding and still unsolved problem, e.g. the development of the cognitive architecture Act-R [25] started in the early 1980s and is still ongoing. There is not the one correct implementation which is able to solve all tasks with similar performance to humans. Many architectures focus only on different aspects of cognition.

[9] estimate the number of artificial cognitive architectures to be around three hundred, of which about one hundred are being actively developed. As in the related works [9] [26] [1] [27] [28], we group cognitive architectures in cognitivistic, emergent (also known as connectionistic) and hybrid approaches, according to the way they represent and process information.

2.2.1 Cognitivistic approaches

Cognitivistic approaches only perceive and process symbolic information. Because of its simplicity, this way of representing information is natural and intuitive. For a programmer, processing symbolic data can
often be done through simple mental models, e.g. if-then rules. Especially high-level abilities such as planning, reasoning and language understanding usually require symbolic information. In the context of robotic agents which perceive from and interact in a highly sub-symbolic world, these approaches represent a strong simplification of the reality as the signal-to-symbol gap needs to be solved or bridged in advance.

One of the earliest cognitive architecture that is still maintained and developed is SOAR \[29, 30\]. The goal of the SOAR (State, Operator Apply Result) project is to develop an artificial system that has same cognitive capabilities as humans (i.e. knowledge-intensive reasoning, reactive execution, hierarchical reasoning, planning, and learning from experience) and to find out what computational structures are required to support human-level agents. The integration of the SOAR architecture on a real robotic systems include early projects such as \textit{Robo-Soar} \[31\] and newer works in the field of autonomous driving \[32, 33\]. The system builds upon a \textit{Spatio Visual System} which transforms the sub-symbolic information directly into a scene-graph based representation. The transformed data is then forwarded into the symbolic working memory. In addition to the systems current understanding of the situation, the WM holds information about the targeted goals. Beyond the WM, SOAR manages three different long-term memories: A procedural memory that contains skills as simple if-then-rules, a semantic memory that contains facts and declarative information about tasks and an episodic memory that manages experiences consolidated from the WM in form of snapshots. The scene-graph-based representation together with the collection of past experiences allows the system to forward simulate the outcome of actions or to generalize over past episodes. Generalized knowledge is again represented symbolically. The fact that the memory is able to generalize shows that it is able to inspect the data to search for groups of similar information.

Another cognitivist architecture that has been applied to robotics is EPIC \[34\]. The goal of EPIC (Executive-Process/Interactive Control) is to represent executive processes that control other processes with a focus on a better understanding of human-computer interaction, i.e. which response delay for a process feels natural for the user. EPIC requires \textit{Sensory-} and \textit{Perceptual Processors} to derive symbolically-coded changes in sensory properties. These processors accept visual, aural and tactile inputs. The WM is a collection of modality-specific items, but also contains perceptually unrelated information such as goals and actions, and is updated periodically. EPIC also has a long-term memory module, however, there is only a one-way connection from LTM to WM, hence the LTM can only initialize the behavior of the full model. The system is not able to derive new behavior rules and put them into the LTM.

We define one core competency of memory to mediate between the high-level symbolic abilities and low-level sub-symbolic motor control and perception. Therefore, cognitivist approaches are better suited for modeling higher cognitive abilities than a complete cognitive architecture due to the absence of sub-symbolic representations.

### 2.2.2 Emergent approaches

Emergent approaches focus on sub-symbolic representations of highly parallel models, such as neural networks. Learning is usually achieved through back propagation \[35\] of the training error based on the network’s prediction compared to desired or expected outcomes. However, real implementations often lack transparency and it is hard for programmers to implement inference rules or to provide prior knowledge.

Based on the enactive approach of cognition \[26\], the goal of the iCub cognitive architecture \[36\] is to integrate phylogenetic (innate) capabilities (e.g. auto-associative memory, action selection, motivation) as well as ontogenetic development in a way that is meaningful for both neurophysiology and developmental psychology. The architecture focuses on the connection of visual impressions, action selection and action execution. Thus, the system requires visual, tactile and proprioceptive sensors. Self-development is achieved through modulation of the innate abilities, which is highly inspired by the functionality of the hippocampus, basal
ganglia, and amygdala. The architecture explicitly models attention and motivation to achieve exploratory but also goal directed behavior. Prospective abilities (e.g. the internal simulation of motor action) are used to influence the action selection. Further, the iCub cognitive architecture explicitly models an auto-associative memory, divided into motor-sensory- and sensory-motor-memory. Episodic memory and procedural memory cover the aforementioned capabilities for prediction, model construction, internal simulation and action. Variants of the iCub cognitive architecture analyze the connection of episodic and procedural memory, combining both to a proof-of-principle joint episodic-procedural memory [37] with shared representation of perception and action [38]. The episodic memory accepts associations but does not generalize over multiple instances. The iCub uses a special representation of saliency and events, namely a *Ego-Sphere* [39]. Events are projected on a virtual dome surrounding the robot. This representation also allows to group perceptions spatially. Due to its strong relation to the design and functionality of the human memory, which, as far as we know, is connectionistic, we assume the iCub cognitive architecture to be emergent. Nonetheless, there are also arguments to classify it to be hybrid, e.g. it uses symbolic tagging of percepts and actions [37]. Although not a full cognitive architecture, [40] provide a promising way to encode images into a so called *deep episodic memory*. The authors use a variational auto-encoder and two decoders to find a representation of the sub-symbolic percepts that allows efficient recall and prediction. [41] extended the model to further include proprioceptive information, recognized objects, speech, task and action information. This episodic memory is a series of latent vectors from the auto-encoder. Associations are learned through back propagating the reconstruction and prediction error through the network. These associations however are not accessible for programmers. A special verbalization component uses the encoded information from the episodic memory to react to user queries, generating natural language of what the robot did, has seen or recognized. The verbalization component as well as the knowledge representation are learned end-to-end which is why we classify these modules to be emergent.

### 2.2.3 Hybrid approaches

Hybrid systems try to combine the advantages of both previous approaches. Implementations often use symbolic representations for high-level cognitive abilities and sub-symbolic representations for learning and development. Ideally, the memory keeps track of both, the sub-symbolic information of the world and the internal state of the agent as well as the symbolic derivations.

**ISAC** (Intelligent Soft Arm Control) [42] is a hybrid cognitive architecture for an upper torso humanoid robot. It is constructed from an integrated collection of software agents and associated memories. The software agents encapsulate all aspects perception, cognition and action and operate asynchronously. Comparable to the iCub cognitive architecture, perceptual events are encoded through a *Sensory-Ego-Sphere (SES)* [43], a discrete representation of what is happening around the robot. An attentional network determines which events are relevant for the current situation. The LTM stores procedural (i.e. robot motions), semantic (i.e. facts) and episodic knowledge (i.e. snapshots of SES, enriched with targeted goals, performed actions, outcomes and valuations). Associations are stored as state transitions in episodic memory. The working memory temporarily stores information that is related to the current task and encapsulates expectations for the future simulated by a *Central Executive Agent*.

**LIDA** (Learning Intelligent Decision Agent) was developed as a biologically inspired cognitive architecture that attempts to model all aspects of cognition [44]. It includes a large amount of cognitive modules, some of which have long-term storage capabilities. LIDA processes information in a cycle which is conceptually divided into three phases: a perception and understanding phase, an attention phase, and an action and learning phase. During the perception and understanding phase, sub-symbolic data from the robots environment gets analyzed and may be translated into symbols corresponding to objects, entities, events or feelings in the *Perceptual Associative Memory* module. A *Current Situational Model* holds information about an
agents present situation enriched through recall of experiences from the long-term memory modules from similar situations. Further, it creates causality links. During the attention phase, the stored information is strengthened or weakened based on programmed specifications (e.g., loudness, brightness or painfulness). The strongest information is broadcasted to all modules. During the action and learning phase, LIDA tries to find correlations between a situational context, an action and the expected outcome of that action.

The CRAM (Cognitive Robot Abstract Machine) cognitive architecture \[45\] fuses perceptual information, semantic knowledge taken from a suite of knowledge bases, and execution results of simulated candidate actions in order to carry out vaguely defined goal-directed tasks. To abstract execution plans in vaguely defined tasks, CRAM uses designators (i.e., placeholders) which require runtime resolution, taken from the memory system KnowRob [\[46\]]. This memory system contains a large scale ontology of symbolic information used for reasoning and generalization. Knowledge is represented as \[NEEMS^{2}\] a first-order time interval logic expression enriched with episodic sub-symbolic information, i.e., the robots’ configuration. This data structure allows to inspect information in the memory in order to use it for reasoning or learning and to simulate the outcome of candidate actions and to evaluate their feasibility. This approach still has a very strong focus on symbolic reasoning and inference. Even sub-symbolic information is lifted into a symbolic representation during querying. Further, the knowledge base only uses robot configurations in sub-symbolic representation.

### 2.3 Contributions

[4] criticized cognitive architectures for having the wrong focus on memory, since in most artificial systems, the memory is assumed to be passive storage, as this facilitates the implementation and management of knowledge. This applies e.g. to the working memory of SOAR or the deep episodic memory. Instead, a memory must be active and change during operation. In addition, many cognitive architectures focus only on the implementation of different types of memory without considering the interconnection between them. For example, the iCub and ISAC architecture implement a procedural and episodic memory but associations are only learned auto-associatively. Therefore, the cognitive system cannot learn that a specific cognitive ability, such as generating grasp candidates, is associated with an action, such as grasping. \[4\] also criticized the way how information is represented in artificial memory systems. Physiologically, all areas of memory are the same; The information stored in memory must have a unified and multi-modal representation. EPIC manages modality-specific knowledge which indicates that the authors employ special containers for special modalities, making the system inflexible for new data types. Often, cognitive architectures only focus on the perception-action coupling, ignoring the fact that meta-cognition also should be connected to memory. Finally, a memory must be able to access distributed data sources. This allows the system to be flexible and to reduce network traffic on distributed systems as the data can be stored and processed where it is produced. In addition to the identified requirements by \[4\], we emphasize that introspection is another key component of memory and data representation as it allows the system to adapt its behavior based on the stored information, to learn and to use the information during internal simulation and prediction.

As shown in Table \[1\] no other system implements all requirements for both symbolic and sub-symbolic information. Hence we introduce the concepts of the ArmarX Memory System.

---

\[2\]https://ease-crc.github.io/soma/owl/current/NEEM-Handbook.pdf
This work

Table 1: Comparison of the most relevant, actively developed cognitive architectures with respect to the identified requirements for a memory system in robotics. Similarly to [9], we use \( V \) for vision, \( T \) for tactile and \( P \) for proprioception. Additionally, we introduce \( A \) for audition and \( M \) for meta-cognition.

Figure 4: Technical characteristics of memory systems in robotic applications. This figure should be seen as an extension to Figure 1.

3 The ArmarX Memory Architecture

In this section we introduce our novel ArmarX hybrid memory architecture that combines experience-based learning and generative knowledge extension. We start by outlining the technical requirements of a complex robotic system in order to define the cognitive architecture in addition to the conceptual requirements of an active, multi-modal, associative, episodic and introspective memory motivated in Section 1. These requirements include technical aspects such as the need of \textit{long-term} capabilities for reducing the dist space and a \textit{distributed design}. Afterwards, we describe how this cognitive architecture was implemented in the robot development environment ArmarX in Section 3.2 and we conclude this section by showing as a case study, how the ArmarX memory architecture is used with the ARMAR humanoid robot family.

3.1 Technical Requirements

Complex real-world humanoid robotic systems pose several challenges to a cognitive memory architecture. In this section, we highlight these challenges, identify the resulting requirements, and derive necessary paradigms.
3.1.1 Considering Data Streams and Events as Episodes

In general, we can distinguish two paradigms in which data sources behave. Either the data is produced periodically in streams, or as a consequence of certain conditions, thus being event-driven. Both ways have in common, that they are episodic. We think that this is an important insight, because even (seemingly) factual knowledge is dependent on the temporal context or the situation, and it is only considered a fact because it changes at a very low frequency. An intuitive example of this is how Pluto was considered a planet for the longest time, until being reclassified as dwarf planet in 2006. Thus, every bit of information must be tied to the exact point in time it was produced in.

3.1.2 Assessing What to Store

Humanoid robots are equipped with a variety of sensors that generate large amounts of data, potentially at high frequencies. A memory architecture thus needs to be able to handle these volumes, providing long-term capabilities due to limited storage. Since the storage system of a mobile robot cannot be large and fast enough to simply store everything, the data must be assessed and reduced according to their importance. This can be done by primarily storing data which was produced at or near significant points in time (e.g. keypoints, identified by higher-level cognitive processes), and/or using dimensionality reduction approaches to construct meaningful latent spaces, for example by employing autoencoders, which would further also allow faster access and comparisons between data points. Depending on the use case, specific models can also be used to aggregate data into a dedicated representation. Finally, the cognitive system must also be able to assess and delete data that has already been stored if it is outdated, irrelevant, or proves to be incorrect. Overall, in terms of assessment, we want to store data that enables better execution of actions on the robot, analysis and reasoning on recorded episodes, and the prediction of future states of the robot itself or its environment.

3.1.3 Scaling through a Distributed System Design

Many robotic systems consist of several individual special purpose computers, which are connected to each other via some common interface (usually Ethernet). Again, its hard to transfer the masses of data produced by the robotic system through such interfaces. To reduce the throughput and to reduce response times, we believe that a cognitive memory architecture must be set up in a distributed manner, with special purpose memories located directly on the machine the data is produced. This trait also helps extending the system at runtime, as additional special purpose memories can be enabled when needed, and deactivated to save resources.

With this, we conclude that, in addition to the motivated conceptual requirements, our memory system needs to be distributed, event-driven, and long-term.

3.2 System Overview

In this section, we provide a system overview of the ArmarX memory system on conceptual level as seen by a user of the system. First, we will describe the episodic working memory, which is the part most clients are directly interacting with. Afterwards, we present the long-term memory and its learning capabilities which adopts the working memory’s principle structure but provides a more permanent storage than the working memory.

From an simplified perspective, the memory system can viewed on three levels of detail:

1. The distributed memory system, which is a collection of memory servers running in their own processes.
2. A single memory server, which stores different data modalities in episodically structured segments.
3. A single data instance, which holds data in a general, interpretable format.

3.2.1 Distributed Memory System

As described above, the ArmarX memory system is a distributed system implemented through several memory servers. Memory servers can communicate with other processes of the system. For reading and writing, all memory servers offer a common interface. A concrete memory server may provide specialized interfaces for its respective modalities (e.g. objects). All memory servers register themselves in the Memory Name System (MNS) on startup which allows memory clients, i.e. processes that use the memory, to establish connections to the memory servers.

Memory servers can be distributed to several computers, distributing the load and reducing the network traffic and response times by running memory servers close to related hardware and memory clients such as a camera and our visual perception pipeline, greatly supporting the system’s scalability.

As shown in Figure 5, the distributed memory system can be visualized as a cloud of memory servers, running in separate processes on different machines, each one extending the system with new modalities. The robots’ full memory is the union over all distributed memory servers. Each memory server manages its own working memory and a corresponding long-term memory. The working memory represents a volatile in-memory data storage that can be accessed by clients to read and write data in a hierarchical structure. By querying the memory servers using temporal cues, users can get access to the stored information. The memory servers
Conceptual Design of the Memory System of the Robot Cognitive Architecture ArmarX

accept precise queries to return data for a precise point in time as well as broad queries to return approximate information. Further, working memory servers can notify clients as soon as new they receive new information. Memory servers are the first level of structuring the data — each memory server holds information related to a specific modality, sensor or functionality. E.g. the Object Memory holds all information related to objects such as object class information (name, parent classes, meshes) or concrete object instances (instance name, pose).

Each working memory is limited. If data is too old or if the amount of data in the working memory reaches a limit, information is moved to the long-term memory. Both, the working memory and the long-term memory hold the stored information episodically, i.e. they hold a dictionary mapping from timestamps to data instances. Additional meta-information such as the name of the provider or the time it took to transfer it to the memory can be used to analyze the data.

Memory servers implement the aforementioned concepts of a distributed memory system and the inherently required episodic and associative structure. By the time of writing, our working memory servers hold data in plain text. There is no encoding as most components require precise information of how the robot is moving or where objects are located.

3.2.2 Interpretable Data Format

We introduce the ArmarX Interpretable Data Format (IDF), a special format which allows introspection and that is used in all memory servers. Basically, this data format implements a recursive variant-based datatype with special extensions useful for robotic programs. All data in the memory is a composition of basic data objects (int, long, float, double, bool, string and multi-dimensional byte array) and recursive data objects (list and dict). Additionally, the segments in our memory can be annotated with static type information, that is again a recursive variant based formulation. Those types consists of basic type objects (i.e. int, long, float, double, bool, string, time), special type objects which get encoded into multi-dimensional byte arrays (i.e. matrix, orientation, image, pointcloud) or recursive container type objects (i.e. list, tuple, dict, object, pair). The type objects do not contain data and only provide additive information for the data objects. E.g. the data object of a list contains a sequence of all elements of that list. These elements are variant data objects. The type object for that list only has one member representing the accepted type of that list.

This extended type information does not have to be present in the memory servers. Even if it has no knowledge about the detailed type, it knows which members are present, their name and which data they contain. This is used to implement type agnostic procedures which can be applied to all data objects in the memory, no matter what they represent. E.g. for data objects with only numerical values it is possible for the memory to provide predictions using linear regression over the last \( n \) entries in the episodic storage of the memory server. However, if type information is present, the memory server can use that specialized knowledge (e.g. knowledge about special regression models, min or max constraints of single members, ...) to provide better predictions.

Users can specify the static type information through XML. Our system uses code generation as a abstraction mechanism to separate the type description from the implementation. Given the type information in XML, IDF automatically generates business objects for a particular implementation language. These business objects are used by the clients to interact with the memory, but they can also be used as a data representation for peer-to-peer communication. Thus, the memory servers do not all have to be implemented in the same language and clients can also access the memory from a different language. By the time of writing, IDF supports C++ and Python as target implementation languages. For being able to transfer objects via the network, IDF has a mirrored representation in our data transfer middleware to which IDF is automatically converted when required. Further, the generated business objects automatically cover the conversion from
Conceptual Design of the Memory System of the Robot Cognitive Architecture ArmarX

and to IDF, human readable formats and compressed formats. During conversion, these auto-generated methods also keep track of asserting that all specified constraints are fulfilled. Through the code generation, users usually do not have to know about this specialized introspectable format as they can use the business objects and all conversion is done behind the scenes.

3.2.3 Long-Term Memory

Each memory server manages its own long-term memory. The LTM part of a memory server receives data from the corresponding working memory if data is removed from the WM or if the user explicitly commands the WM to consolidate its information.

First, information is filtered based on their frequency or their similarity compared to other entities of the same type in the LTM. After that, special encoders convert the data objects into a format that is better suited for further processing in terms of compression or usability. In addition to this specialized encoding, all data is converted into a binary format using standard lossless methods such as binary or entropy based compression. Finally, the binary streams are stored on the hard drive. The LTM only compresses and converts the data objects — it keeps the structure of the memory servers, segments, episodes and the associations between memory servers. This first part of the long-term memory processing pipeline is done online and only uses fast and standard compression techniques.

But the long-term memory is not only a tool to record and persistently store information. It must also be used to generalize over existing knowledge. During an offline phase, we further compress the data using machine learning techniques. For each entity, the memory instantiates an auto-encoder which uses the latest \( n \) entries to learn a latent representation which is able to reconstruct and predict the given information. To reduce the needed amount of disk-space, the memory only keeps the weights together with the latent representation of the entity. The learned representation allows the memory to identify similar instances that have not been filtered yet. This knowledge can be used to remove redundant information or to get an estimated distance between two instances. Further, the learned representation supports prediction by mapping from the latent representation to its successor.

Through the introspectability of data, the LTM can even learn such a representation, if type information about an entity is not present. The whole process of learning latent representations can be automated. Given the lossless compressed information from the online phase of the LTM one can automatically extract a feature vector and pass it to the auto-encoder. This method differs from our previous work [41] by learning a model for every entity instead of concatenating all entities and learn one single model. This makes the overall system more flexible as introducing new modalities does not require that the whole model needs to be retrained.

This means that the long-term memory manages data of two compression levels - lossless compressed information from the current online phase and lossy compressed information from previous phases. Whenever a client requests information which is not present in the WM, the WM forwards this request to the LTM and if a matching instance is found, the LTM decodes the corresponding data and returns it to the WM. The WM stores the returned information in its volatile storage and forwards a copy to the client.

Both, the WM and the LTM are active parts of our memory system. The WM can adapt its behavior to the current computer load and the LTM actively searches for suitable data representations in order to compress, reproduce, generalize and predict them.

3.3 Use of the Memory System in Robotic Systems

In this section, we will showcase different use cases of varying complexity, showing how different modalities as well as software and hardware components can be integrated into the memory system and how processing
pipelines can structure their data flow to make best use of the memory system. The following examples should be seen as different views on one and the same robot’s cognitive architecture, e.g. there is only one robot state memory in the system which can be used by any other component.

3.3.1 How Basic Components Make Use of the Memory System

**Robot State** One of the most fundamental memories is the robot state memory, which stores high-frequency streams of robot joint angles, odometry, and other low-level sensor information. The robot state memory is the memory closest to the actual hardware, and uses a direct channel to the robot’s bus system so that the robot state information can directly be streamed into it. This is depicted in Figure 6. It is essential, that high-level components can reconstruct the state of the robot for any given time.

![Figure 6: Integrating the robot state into the memory system.](image1)

One use case could be the assessment of self-occlusion of object, as for example one which is currently being grasped. Here, several components are involved in a pipeline: First, images are being taken with a camera system. Second, the object pose in the images is estimated. Third, the self occlusion can be assessed using the robot model, the object pose, as well as robot configuration at the time of taking the picture. It is evident, that passing this pipeline can consume non-negligible amounts of time. Hence, the component assessing the self-occlusion of the object must be able to reconstruct the robot’s state at the images were taken. With our proposed memory architecture, this exact use case is supported directly and conveniently.

**Speech to Text & Text to Speech** Natural language provides one of the most natural interfaces for humans to interact with a robot. The most basic speech components of a robotic systems are Speech to Text (STT) and Text to Speech (TTS). An STT system takes audio an audio stream or sequence as input and outputs what the user said as text. Conversely, a TTS system takes text as input and converts it into a playable audio sequence.

![Figure 7: Integrating speech to text (STT) and text to speech (TTS) components into the memory system.](image2)

An example integration of STT and TTS components into the memory is shown in Figure 7. While speech is usually much more sparse and event-driven than the robot state, it uses a similar architecture. The STT component reads audio from the hardware (e.g. a microphone), converts it to text and commits the text to a Speech to Text segment in the Speech memory. Other components can subscribe the memory segment to react to incoming commands. In contrast, a TTS component can subscribe a Text to Speech segment in the same memory. When another component commits text to this segment, the TTS component receives an update notification from the memory. It then queries the segment for the new text, converts it to audio, and plays it back using the hardware speakers.
Object Detection and Localization   Another important ability of a robot is to detect and localize known or unknown objects in its environment. In this example, we consider the task of 6 DoF localization of known, textured objects based on RGB or RGB-D images as in, e.g., [47, 48, 49]. Consider the processing pipeline in Figure 8. A camera driver component reads data from a hardware camera. In each frame, it commits the RGB and depth images to an RGBD segment in the Vision memory. An object localization component can subscribe the Vision/RGBD segment to be notified about new images. When notified, it queries the new images from the vision segment. In addition, it uses object information such as 3D models or pre-extracted visual features from the Object/Class segment, which contains static information about known object classes. The localization component then applies its internal method to detect objects and estimate their poses. Finally, it commits the new observations to the Object/Instance segment, which represents the current state of objects in the scene.

![Figure 8: Integrating an object localization component into the memory system.](image)

3.3.2 How Complex Robot Abilities Make Use of the Memory System

Navigation   Being able to navigate safely and reliably in a human-accepted manner to reach arbitrary places is a fundamental skill of mobile robots. The navigation stack in ArmarX spans 4 layers in total, namely (i) global path planning, (ii) local/reactive trajectory optimization, and (iii) trajectory following and (iv) safety mechanisms. All of the required algorithms and control loops reside in the navigator component which is shown in Figure 9. The navigator creates its own internal environment model based on the state of the working memory. In particular, it consists of the robot’s state including an estimate of the robot’s global pose, laser-scanner-based clusters and navigation cost maps. As can be seen, the navigation cost maps

---

4https://gitlab.com/ArmarX/skills/navigation
Conceptual Design of the Memory System of the Robot Cognitive Architecture ArmarX

combine information from different domains of the WM such as pre-processed laser scanner data and object poses. As the optimal platform movement differs between tasks, each client sending a request to the navigator can fully parameterize the behavior of the navigator. This includes the selection of different algorithms and its parameters. There also exist pre-defined sets of parameters which a client can choose from as part of prior knowledge. All of this information is stored and logged in the memory which makes the robot’s behavior both introspectable and reproducible.

The communication between a client and the navigator is event-driven. After the navigation request is sent to the navigator, the client can focus on secondary tasks and gets notified about success and failure events e.g. the robot reaching the desired target but also if a failure of a planning step occurs. The navigator stores these events in the memory, including a timestamp and a detailed description. By using the memory’s built-in notification and event subscription mechanisms, the client gets notified and can react.

**Grasping and Manipulation** Grasping is one of the most important abilities of a robotic system. Here, we consider the example of an affordance-based grasping pipeline as in [50]. In such a pipeline, an affordance extraction component first visually detects grasping affordances of an object or parts of the scene. Then, an action planning component attempts to plan executable actions representing the extracted affordances, solving sub-problems like inverse kinematics, motion planning and grasp quality estimation. Finally, an action execution component selects a planned action and executes the action on the robot.

This pipeline can be implemented with the memory system as shown in Figure 10. The Object and Vision memories provide the necessary input data, i.e. localized object instances and depth images of the scene. In addition, there is a Grasping memory with Affordance, Action and Outcome segments, each of which representing the result of a pipeline step. Consequently, the affordance extraction component commits extracted affordances to the Grasping/Affordance segment, which is subscribed by the action planning component, etc. During execution, the action execution component gathers result information such as success or failure and commits it to the Grasping/Outcome segment, which may be used by the robot to improve its internal success-or-failure prediction.

![Figure 10: A pipeline for affordance-based grasping.](image)

This example also nicely demonstrates how associative memory links can be used to enhance explainability: An extracted grasping affordance has a link to the object instance and image it was extracted from. The grasping action, in turn, has a link to the grasping affordance, and the outcome to the executed action. In this setup, the whole trace of a specific grasping outcome can be reconstructed by following the memory links backwards. The trace can be visualized as a tree like in Figure 11.

![Figure 11: Trace of a grasping outcome reconstructed from memory links.](image)
In addition, one could imagine many more advanced usages of the memory system with respect to grasping. For example, consider a method for affordance extraction which depends only on the object model and not on the rest of the scene (i.e., collision checking is done by the action planner). In this case, the affordance extraction could query the grasping affordance segment for previously computed affordances. Only if there are none, or if the object model changed, it would need to compute new affordances. Another use case is a high-level execution monitoring component, which starts the grasping pipeline and waits for an outcome to be added to the segment. When the outcome is received, and the monitoring component finds the outcome of a failure, it can restart the pipeline, potentially reconfiguring the components based on the information in and linked by the outcome.

**Human-Robot Handover** In human-robot collaboration, handing over objects is a fundamental skill. It involves the estimation of the human pose, the localization of objects, the calculation of grasps, and a reactive control strategy for grasping since the object might be moving. Also in this case, a memory system proves to be beneficial. When handing an object to a human, especially the poses segment of the Human memory is involved. Here, the 3D poses of humans identified in RGB-D images in the Vision memory are stored. Most importantly the hand positions of the human are of interest to determine where the handover is to take place. As for the handover of an object from a human to a robot, the Object and Grasping memory are also involved. For one, to determine which object to grasp, and second, to plan grasps on it and execute them.

### 4 Conclusion and Future Work

We have introduced our novel memory system used in our robot software framework ArmarX and motivated 8 requirements for a memory system whereby 5 of them are conceptual and 3 are technical requirements. Our new memory system fulfills these requirements in the following way:

- It is an **active** memory, because the WM adapts its behavior based on the current computational load and the LTM adapts its ability to predict and generalize to the given data.
- It is **multi-modal**. The memory has no constraints against the input data. Everything can be stored and everything can be encoded, no matter if it is symbolic or sub-symbolic information.
- It is inherently **episodic**. All information is stored episodically — even semantic information.
- It supports an **associative** structure. Entities can be linked to other entities of the same or even of different modalities. We assume, that the knowledge how information is connected is as important as the information itself.
- The chosen data representation is **introspective**. This allows the memory investigate the data, to check constraints and it finally enables the memory to support explainability, predictability and reasonability.
- Due to the fact that our robots have several special purpose computers, our memory follows a **distributed** design. All subsystems of the memory (memory servers) may run on different machines which reduces the network traffic and increases response times. A special component, the Memory Name Service, manages the connections to all memory servers.
- As robots act in the real world, they need an understanding of consequences. Often, events are triggered through the occurrence of certain conditions which is why our memory supports an **event-driven** approach.
- Finally, as robot systems are not active all the time and they have limited storage, the memory must be able efficiently store information in the **long-term**.
Further, we explained how the implementation of these requirements influence the way we write and evaluate our software. Nonetheless, the system is by far not complete as the development of a novel cognitive architecture (from which the memory is the key element) usually requires years. So far the memory accepts only temporal requests, however, episodic memory should also store data spatially. Thus, we plan to support arbitrary keys (encoded in IDF) for the management of data. This would enable our memory to use spatio-temporal keys but also more complex ones like socio-temporal key, i.e. a mapping from social interactions and timestamps to multi-modal experiences. The ability to use arbitrary keys greatly accelerates the retrieval of knowledge as the memory must not look into every single data instance. The decision, what to use as a key in the memory can, due to the introspection ability and that the memory knows about the general structure of data, eventually be automated and learned. The memory must correlate the content of the data with its confidence and can determine which information is relevant to the latest experiences. It can also refer back to past queries and optimize access time for them by inserting new keys. In the end, ideally we have a memory system which is able to automatically learn how to represent data and how to manage data.

As mentioned before, our memory currently only provides rudimentary tools to filter incoming data. In the WM, data is not filtered or encoded at all. During our experiments we did not reach the limits of storage capacity — probably because we are using a distributed system and not a centralized memory — but we believe that we need early filter and encoder steps to manage the data more efficiently. Moreover, it is currently not possible to incrementally refine the models found for learning representations in LTM. Either we generate a new model only for the experiences of the current work cycle or we remove the models we have, concatenate the decoded experiences and train a new model with this dataset extended by the current work cycle. The latter approach is less storage space consuming but also carries the decoding error into the next model. Incremental approaches are desirable.

Last but not least, there is a lack of methods to evaluate and learn from the stored data. In particular, it would be interesting to compare the memories of several robots and make them available to each other (comparable to what is done in [51]). Shared memories, with other robots or with humans, also open up questions about security and privacy which should already be addressed at memory level. For this reason, we have also recently started to make our memory system aware of the confidentiality of data [52].

The planned work mentioned above represents only a small part of the further development of our cognitive architecture. Nevertheless, every single point is a complex research topic on its own. We believe that the architecture we have chosen, based on the identified requirements of a memory system, is general and at the same time specific enough for robotic applications to help us to address these issues.
References

[1] David Vernon. *Artificial Cognitive Systems: A Primer*. The MIT Press, 2014.

[2] David Vernon. Cognitive system. In *Computer Vision: A Reference Guide*, pages 100–106, 2014.

[3] Norbert Krüger, Christopher Geib, Justus Piater, Ronald Petrick, Mark Steedman, Florentin Wörgötter, Ales Ud, Tamim Asfour, Dirk Kraft, Damir Omčen, Aleandro Agostini, and Rüdiger Dillmann. Object–action complexes: Grounded abstractions of sensory–motor processes. *Robotics and Autonomous Systems*, 59(10):740–757, 2011.

[4] Rachel Wood, Paul Baxter, and Tony Belpaeme. A review of long-term memory in natural and synthetic systems. *Adaptive Behavior - ADAPT BEHAV*, 20, 01 2012.

[5] Endel Tulving. Episodic and semantic memory. In Endel Tulving and W. Donaldson, editors, *Organization of Memory*, pages 381–403. Academic Press, New York, 1972.

[6] Endel Tulving. Précis of elements of episodic memory. *Behavioral and Brain Sciences*, 7(2):223–238, 1984.

[7] Nikolaus Vahrenkamp, Mirko Wächter, Manfred Kröhntert, Kai Welke, and Tamim Asfour. The robot software framework armarx. *it - Information Technology*, 57, 01 2015.

[8] Larry R. Squire. Memory systems of the brain: A brief history and current perspective. *Neurobiology of Learning and Memory*, 82(3):171–177, 2004. Multiple Memory Systems.

[9] Iulia Kotseruba and John K. Tsotsos. 40 years of cognitive architectures: core cognitive abilities and practical applications. *Artificial Intelligence Review*, Jul 2018.

[10] Susan EF Chipman. *The Oxford handbook of cognitive science*. Oxford University Press, 2016.

[11] Alexei V Samsonovich. Toward a unified catalog of implemented cognitive architectures. *BICA*, 221(2010):195–244, 2010.

[12] R. C. Atkinson and R. M. Shiffrin. Human memory: A proposed system and its control processes. In K. W. Spence and J. T. Spence (Eds.), *The Psychology of learning and motivation: Advances in research and theory (vol. 2).* , pages 89 – 105, 1968.

[13] Bart Aben, Sven Stapert, and Arjan Blokland. About the distinction between working memory and short-term memory. *Frontiers in Psychology*, 3, 2012.

[14] David Vernon. Cognitive architectures. In *Cognitive Robotics*, 2022.

[15] Omri Barak and Misha Tsodyks. Working models of working memory. *Current Opinion in Neurobiology*, 25:20–24, 2014. Theoretical and computational neuroscience.

[16] George A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *The Psychological Review*, 63(2):81–97, March 1956.

[17] Elisabet Service. The effect of word length on immediate serial recall depends on phonological complexity, not articulatory duration. *The Quarterly Journal of Experimental Psychology Section A*, 51(2):283–304, 1998.

[18] Nelson Cowan. The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences*, 24(1):87–114, 2001.

[19] Fernand R. Gobet. Some shortcomings of long-term working memory. *British journal of psychology*, 91 (Pt 4):551–70, 2000.

[20] James C. Taylor. A dynamic model of memory for research on human information processing. *Instructional Science*, 12(4):367–374, 1983.
[21] Kim S Graham, Jon S Simons, Katherine H Pratt, Karalyn Patterson, and John R Hodges. Insights from semantic dementia on the relationship between episodic and semantic memory. *Neuropsychologia*, 38(3):313–324, 2000.

[22] D. Coon, J.O. Mitterer, and T.S. Martini. *Introduction to Psychology: Gateways to Mind and Behavior*. Cengage Learning, 2018.

[23] Androulla Ioannou and Xenia Anastassiou-Hadjicharalambous. *Non-associative Learning*, pages 1–13. Springer International Publishing, Cham, 2018.

[24] Rony Novianto, Mary-Anne Williams, Peter Gärdenfors, and Glenn Wightwick. Classical conditioning in social robots. In *International Conference on Social Robotics*, pages 279–289. Springer, 2014.

[25] John Anderson, Daniel Bothell, Michael Byrne, Scott Douglass, Christian Lebiere, and Yulin Qin. An integrated theory of the mind. *Psychological review*, 111:1036–60, 11 2004.

[26] David Vernon, Claes von Hofsten, and Luciano Fadiga. A roadmap for cognitive development in humanoid robots. In *Cognitive Systems Monographs*, 2011.

[27] Ron Sun. Desiderata for cognitive architectures. *Philosophical Psychology*, 17, 09 2004.

[28] Wlodzislaw Duch, Richard Jayadi Oentaryo, and Michel Pasquier. Cognitive architectures: Where do we go from here? In *Agi*, volume 171, pages 122–136, 2008.

[29] John E. Laird, Allen Newell, and Paul S. Rosenbloom. SOAR: an architecture for general intelligence. *Artif. Intell.*, 33(1):1–64, 1987.

[30] John E. Laird. *The Soar Cognitive Architecture*. The MIT Press, 2012.

[31] John E. Laird, Eric S. Yager, Michael Hucka, and Christopher M. Tuck. Robo-soar: An integration of external interaction, planning, and learning using soar. *Robotics and Autonomous Systems*, 8(1):113–129, 1991. Special Issue Toward Learning Robots.

[32] John Laird. Toward cognitive robotics. *Proceedings of SPIE - The International Society for Optical Engineering*, 7332, 05 2009.

[33] Scott D. Hanford and Lyle N. Long. Development of a mobile robot system based on the soar cognitive architecture. *Journal of Aerospace Information Systems*, 11(10):714–725, 2014.

[34] Davis E Kieras and Davis E Meyer. An overview of the epic architecture for cognition and performance with application to human-computer interaction. *Human–Computer Interaction*, 12(4):391–438, 1997.

[35] Robert Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural networks for perception*, pages 65–93. Elsevier, 1992.

[36] David Vernon, Giorgio Metta, and Giulio Sandini. The icub cognitive architecture: Interactive development in a humanoid robot. In *2007 ieee 6th international conference on development and learning*, pages 122–127. Ieee, 2007.

[37] David Vernon, Michael Beetz, and Giulio Sandini. Prospection in cognition: the case for joint episodic-procedural memory in cognitive robotics. *Frontiers in Robotics and AI*, 2:19, 2015.

[38] Armin Stock and Claudia Stock. A short history of ideo-motor action. *Psychological research*, 68(2):176–188, 2004.

[39] Jonas Ruesch, Manuel Lopes, Alexandre Bernardino, Jonas Hornstein, José Santos-Victor, and Rolf Pfeifer. Multimodal saliency-based bottom-up attention a framework for the humanoid robot icub. In *2008 IEEE International Conference on Robotics and Automation*, pages 962–967. IEEE, 2008.

[40] Jonas Rothfuss, Fabio Ferreira, Eren Erdal Aksoy, You Zhou, and Tamim Asfour. Deep episodic memory: Encoding, recalling, and predicting episodic experiences for robot action execution. *IEEE Robotics and Automation Letters*, 3(4):4007–4014, 2018.
Conceptual Design of the Memory System of the Robot Cognitive Architecture ArmarX

[41] Leonard Bärmann, Fabian Peller-Konrad, Stefan Constantin, Tamim Asfour, and Alex Waibel. Deep episodic memory for verbalization of robot experience. *IEEE Robotics and Automation Letters*, 6(3):5808–5815, 2021.

[42] Kazuhiko Kawamura, Stephen M Gordon, Palis Ratanaswasd, Erdem Erdemir, and Joseph F Hall. Implementation of cognitive control for a humanoid robot. *International Journal of Humanoid Robotics*, 5(04):547–586, 2008.

[43] Richard Alan Peters, Kimberly E Hambuchen, Kazuhiko Kawamura, and D Mitchell Wilkes. The sensory ego-sphere as a short-term memory for humanoids. In *Proceedings of the IEEE-RAS international conference on humanoid robots*, pages 451–459. Citeseer, 2001.

[44] Stan Franklin, Tamas Madl, Sidney D’mello, and Javier Snaider. Lida: A systems-level architecture for cognition, emotion, and learning. *IEEE Transactions on Autonomous Mental Development*, 6(1):19–41, 2013.

[45] Michael Beetz, Lorenz Mösenlechner, and Moritz Tenorth. Cram—a cognitive robot abstract machine for everyday manipulation in human environments. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1012–1017. IEEE, 2010.

[46] Michael Beetz, Daniel Beßler, Andrei Haidu, Mihai Pomarlan, Asil Kaan Bozcuoğlu, and Georg Bartels. Know rob 2.0—a 2nd generation knowledge processing framework for cognition-enabled robotic agents. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 512–519. IEEE, 2018.

[47] Pedram Azad, Tamim Asfour, and Rüdiger Dillmann. Combining Harris Interest Points and the SIFT Descriptor for Fast Scale-Invariant Object Recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4275–4280, October 2009.

[48] Karl Pauwels and Danica Kragic. SimTrack: A Simulation-Based Framework for Scalable Real-Time Object Pose Detection and Tracking. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1300–1307, September 2015.

[49] Yisheng He, Haibin Huang, Haoqiang Fan, Qifeng Chen, and Jian Sun. Ffb6d: A full flow bidirectional fusion network for 6d pose estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3002–3012, 2021.

[50] Christoph Pohl, Kevin Hitzler, Raphael Grimm, Antonio Zea, Uwe D. Hanebeck, and Tamim Asfour. Affordance-based grasping and manipulation in real world applications. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9569–9576, 2020.

[51] Markus Waibel, Michael Beetz, Javier Civera, Raffaello d’Andrea, Jos Elfring, Dorian Galvez-Lopez, Kai Häussermann, Rob Janssen, JMM Montiel, Alexander Perzylo, et al. Roboearth. *IEEE Robotics & Automation Magazine*, 18(2):69–82, 2011.

[52] Saskia Bayreuther, Florian Jacob, Markus Grotz, Rainer Kartmann, Fabian Peller-Konrad, Fabian Paus, Hannes Hartenstein, and Tamim Asfour. Combining Task Planning and Activity-Centric Access Control for Assistive Humanoid Robots. In *Symposium on Access Control Models and Technologies (SACMAT)*, 2022.