Computational Framework for and the Realization of Cognitive Agents Providing Intelligent Assistance Capabilities

Marcin Skowron, Joerg Irran, Brigitte Krenn[1]

Abstract. The scope of the presented research covers virtual agents providing intelligent assistance capabilities for accessing and processing information from the Internet, domain specific databases and knowledge repositories. They receive natural language inputs and communicate findings to their users via a set of task oriented interfaces. Cognitive agents are conceived to evolve in a response to the changes of interests, needs and preferences of the users and the alterations in their environment. We present a virtual embodied cognitive agents architecture, and a computational framework that allows their modular and flexible creation, based on a set of components. The framework supports the creation of an environment for multiple agents and provides communication mechanisms, used to share knowledge between the agents. The exemplary assembly of these building blocks to realize smart assistance applications further demonstrates the platform's capacity to support development, instantiation and evaluation of collaborative cognitive agents.

1 MOTIVATION AND OBJECTIVES

Our motivation is to create virtual agents that provide personalized assistance to their users in finding and retrieving information from the Internet and other resources such as domain-specific databases and knowledge repositories. The developed virtual agents represent a growing class of cooperative agents that do not have a physical presence, but nevertheless are equipped with major ingredients of cognition including situated correlates of physical embodiment to become adaptive, cooperative and self improving in a virtual environment, given certain tasks. The design of the agents is partially inspired by embodied cognition originating from interaction-based robotics, transferred to a virtual context[5, 4]. This is persuade in the projects RASCALLI (Responsive Artificial Situated Cognitive Agents that Live and Learn on the Internet) and SELP (Advanced Knowledge Technologies: Grounding, Fusion, Applications). The presented work covers the following topics: development of a computational framework for realization of cognitive agents providing intelligent assistance capabilities, cognitive architecture and modeling, perception and action, reasoning, learning, communication, agent-to-agent and agent-to-user interfaces.

In the symbolic AI tradition, there has been a substantial work on Internet agents that perform pre-defined tasks on the Internet, including information retrieval and extraction tasks[2], checking for website updates, doing price comparisons etc. However, such Internet agents relied rather on extensive statistical analysis and on existing search engines than on the usage of cognitive architectures in an attempt to develop adaptive and flexible virtual agents. The Rascalli project aims at extending the state of the art by developing situated cognitive agents which inhabit the Internet environment. The investigation of the added value through the usage of cognitive architectures for improving the agent's capabilities and the user experience in interaction with that agent is a further objective of the presented work. This is feasible due to the modular approach used in the Rascalli platform, which supports multiple agent architectures and agent definitions, and therefore serves as a testbed for the evaluation and comparison of the agents.

The major standards organization in the area of Agent Oriented Software Engineering (AOSE) is FIPA[3], which is concerned with the standardization and interoperability of multi-agent systems. JADE[4] and similar FIPA compliant agent platforms offer a strong middleware layer for distributed multi-agent systems, including agent lifecycle management, agent communication, as well as rich graphical tools for agent development but they do not meet major requirements of the RASCALLI platform. While the RASCALLI platform supports the execution of multiple agents, it is not a multi-agent system in the traditional sense, where agents are independent components of a larger application. Instead, Rascalli are complete individual entities that simply happen to exist in the same environment and are intended to communicate with each other. Furthermore, none of the aforementioned agent platforms supports the development style targeted by the RASCALLI platform, where multiple agent architectures and agent definitions, as well as multiple versions of agent components co-exist in a single platform instance. This development style is specifically geared towards research projects experimenting with alternative cognitive architectures, and combining them with a variety of action and perception tools as it is the case with the exemplary applications (see section 6) created and integrated in the Rascalli platform. The RASCALLI approach differs also from existing software systems for cognitive modeling such as AKIRA VI[5] or AmonI[6]. While the latter two provide specific means for modeling cognitive processes, the RASCALLI platform is a more general framework for implementing a variety of different models and architectures.

The rest of the paper is organized as follows: section 2 describes concepts and design principles of the cognitive agents providing intelligent assistance capabilities. Section 3 introduces the computational

[1] Austrian Research Institute for Artificial Intelligence,Austria, email: marcin.skowron,joerg.irran,brigitte.krenn@ofai.at

[2] http://www.cs.washington.edu/research/softbots

[3] http://www.fipa.org

[4] http://jade.tilab.com

[5] https://sourceforge.net/projects/a-k-i-r-a/

[6] htpp://www.cs.bath.ac.uk/ai/AmonI-sw.html

framework for realization of the cognitive agents as implemented in the Rascalli platform. Section 4 provides an overview of a bottom-up and top-down cognitive components used in the system. Section 5 introduces the concepts and components implemented in the platform, that are used to create different incarnations of Rascalli agents, presented in section 6. The last section provides the conclusions and presents the future work.

2 CONSTITUTION OF THE AGENTS

In the RASCALLI project the creation of virtual agents that provide personalized assistance to users in finding and retrieving information from the Internet and other resources such as knowledge repositories and domain-specific databases is pursued. The set of actions the agents are able to perform includes web and database search mechanisms, techniques for extracting information from social networks and news-feeds, techniques for extracting information from texts, logging of user interactions and recommendation of related content. The agents evolve in response to the changes of interests, needs and preferences of the users and to the alterations in the agents environment. Therefore the agents need a certain degree of autonomy and flexibility in their behavior. To achieve this, the agents have a biologically inspired virtual embodiment consisting of three layers (see Fig. 1), a perception layer with which they perceive (analyse) their environment, a cognitive layer with learning and reasoning capabilities, and an actuator layer which allows them to act on the environment employing a selection of processing tools. Everything outside the agent, including the user, Internet resources, local databases and other Rascalli agents is considered as a part of the environment. Due to the modular design of the platform (refer to section 3), a cognitive component can be realized using various approaches (refer to section 4) and different agents can be assembled using a selected set of action-perception and user-agent interface components (refer to section 5).

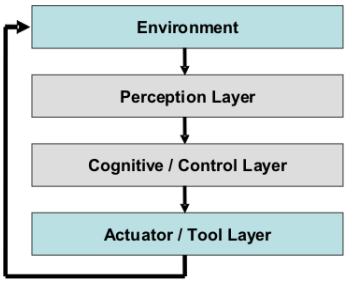

Figure 1. Layers of the agents virtual embodiment and the interaction loop with the environment

3 RASCALLI PLATFORM - A COMPUTATIONAL FRAMEWORK FOR COGNITIVE AGENTS

In the scope of the Rascalli project, various sets of action-perception tools as well as knowledge acquisition, reasoning and decision making mechanisms are being conceptualized, developed and tested. The Rascalli development platform[10] supports a flexible creation of individual virtual agents via assembly from a set of existing software components. The platform allows also multiple instances of agents to be run on a single platform instance, thus different incarnations of virtual agents can be exposed to the same environmental conditions, as well as developed and evaluated in parallel. The Rascalli project provides also several interface modules for the realization of agent-human interaction, including an Embodied Conversational Agent interface and other interfaces described in section 5.7. The interfaces are treated in the Rascalli platform in the same way as the other components. Since the components are isolated building blocks, they can be separately evaluated, according to what information they have gathered, what outputs they have produced to certain tasks or according to the the user satisfaction. Consequently, different system incarnations (including various implementations of a cognitive component) can be assembled and evaluated to find an optimal set of system components for a given objective (see section 6).

The design of the software architecture of the RASCALLI platform is determined by a number of requirements. They stem from the need for distributed development of individual components, the wish to run multiple Rascalli on a single platform, the need for a platform that supports a plug and play approach, and that supports the realization of and experimentation with different cognitive architectures/models/components. We additionally aim at a research setting, where the integration of existing components is preferable to re-implementation, and where system integration is likely to be a technically non-trivial task.

The RASCALLI platform supports the implementation of agents of different constitutions, structured in several layers (see Fig.2 for an overview):

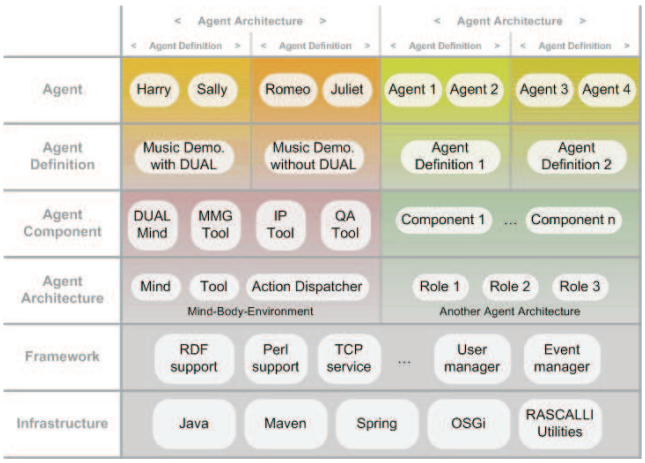

Figure 2. Rascalli platform and its layers

Infrastructure Layer: The infrastructure layer contains basic technologies and components the RASCALLI platform is build upon.
Framework Layer: The framework layer contains general platform services and components available to all RASCALLI agents.
Agent Architecture Layer: This layer defines the constitution of the general agents architecture (e.g. Mind-Body-Environment).
Agent Component Layer: The Agent Components layer contains implementations of the components/interfaces defined by the Agent Architecture Layer.
Agent Definition Layer: An agent configuration is an assembly of specific components of the Agent Component Layer.
Agent Layer: The agent layer contains the actual individual agents.

Each agent is an instantiation of a specific agent configuration, based on a set of agent components defined within a specific agent architecture. An individual agent might have additional configuration beyond what is provided on the agent configuration layer such as an individual agent's name, name of its user, etc. A more detailed description of the agents cognitive and other components is provided in the following sections.

4 TOP-DOWN AND BOTTOM-UP COGNITIVE CAPABILITIES

Considering the agents' cognitive component, we follow two strands of knowledge acquisition and action selection for the agents: a low level bottom-up approach where an agent acquires knowledge based on logs from its interaction with the user and the other entities from its environment, and a theory driven (top-down) approach based on the DUAL/AMBR cognitive model[8]. To realize an interaction based knowledge acquisition architecture for virtual environments, a virtual embodiment for our agents is created. They are equipped with a collection of sensor channels geared towards the particular environment, and a set of specialized software tools (actions) through which they interact with the environment. These are described in more detail in section 5. Both approaches have a specific characteristic and predispositions in the scope of their applications. While the data-driven approach, is more flexible and robust, the theory-driven approach provides more fine-tune capabilities and accuracy for restricted domains. Our aim is to take advantage of the best properties from both approaches and employ the above presented features of the Rascalli platform, reconciling both strands in a hybrid model, by matching the structures emerging in the bottom-up approach with the ones implemented in a top-down fashion by the cognitive model[15]. The research on the cognitive component includes the following topics: knowledge acquisition process, impact of the design of the environment, the design and constitution of the agent, and the training situation, flexibility/robustness in task solving strategies, dealing with changing situations (dynamic environment, novel input, sensor channels, tools), scalability (with increasing complexity of sensor channels, tools and tasks).

5 BUILDING BLOCKS OF THE SYSTEM

In this section the concepts and implemented components are presented in more detail, including the principles of the agents cognitive components, their relations to the agents action-perception capabilities, strategies used for knowledge acquisition, user profiling and user-agent interfaces. In section 6 it is demonstrated how the selected building blocks can be assembled to achieve a particular goal.

5.1 Agent environment - virtual entities

The agent environment includes the Internet, databases, knowledge resources, the user and other Rascalli agents. The Rascalli agent perceives its environment (via a set of perception sensors, implemented as software tools - see 5.2) as a set of unique virtual entities with their own characteristics and properties. These include strings of written language originating from the user or extracted from HTML documents, markup tags, meta-data information about binary files, information about the accessibility of various Internet and local tools and resources, user feedback, etc. Therefore, the agent has to deal with a dynamic environment i.e. evolving content of the websites, permanent or temporary inaccessibility of Internet services, appearance of a new content or services, changes in the user preferences and interests, as well as the unrestricted natural language input from the user and the web-pages. Similarly, all the agents actions in its environment are performed on the above introduced set of entities (refer 5.4 for an extensive overview of the agents" actuators).
Since interaction-based learning is a bottom-up driven learning approach of increasing complexity, it necessitates a simplification of the initial environment of the Rascalli. This is achieved by providing a reduced set of resources, e.g. closed domain databases, selected RSS feeds and websites, as well as a reduced and simplified instruction-driven user input.

5.2 Perception layer - sensor channels of the agent

The aim of the perception layer is to provide the agents with the capabilities to perceive an input situation and to distinguish a set of features necessary for selecting an appropriate action (tool application), based on the similarities between input situations and the episodes, which the agent encountered before.

5.2.1 Concept of the classification driven perception

The perception layer contains all components that are used by agents to sense their environment. Those components are designed to allow classification driven perception. The perception layer implementation in the Rascalli platform includes the Input-Processing Tool, which is based on the classification of input data and recognition of the source of an input (e.g. user utterance, pdf document, web page, user feedback). Various classes and categories are for example assigned to the data originating from the user utterance such as named entities (locations, organizations, person names), part-of-speech tags, coarse-grained utterance classes (greeting, question, acceptance, rejection, etc.), question categories, syntactic parse trees. Similarly, various classes are provided for the other types of virtual entities such as file extension and size, the content of an HTML file (title, headers, links, etc.), the meta-data related to binary files (title, artist, album, width, frame rate, etc.), input class (user utterance; praise/scolding button value, data input from the Internet/databases), IO error message status from the action application, user feedback. The classes are assigned based on:

- information perceived directly from the environment (e.g. file extension, name of an artist originating from a music file header, input class, feedback)
- machine learning based classifiers developed or adopted for the agents requirements(Maximum Entropy[6] based utterance classification, Supported Vector Machines[19] based question classification[17]
- available NLP tools (e.g. parsers, pos-taggers, named entity classifiers)

5.2.2 Natural Language Input Processing

Natural Language Input Processing (T-IP) involves a set of sensor channels each of which allows the agent to perceive certain aspects of a natural language input situation. T-IP performs actions on entities from the agent's environment such as part-of-speech tagging the user utterance, finding the question focus word or segmenting text. At the current stage of development, the tool provides the following information to the agent:

- utterance class (greeting, question, agreement, rejection, find-similar, other),
- question class (6 coarse grained categories: abbreviation, description, entity, human, location,number; and 50 fine-grained categories [12]),
- POS tags [1],
- question focus word [17],
- instances, concepts, relations from the Rascalli ontology,
- NP Chunks,
- Minipar parse [13],
- Wordnet entries (antonyms, synonyms, hypernyms,coordinate terms, polysemy count, etc.)[2],
- DUAL interest [9],
- DUAL question [9],
- DUAL free-question flag [9].

5.3 Cognitive layer - from knowledge acquisition to action selection

To achieve intelligent assistance capabilities, an agent must have abilities to select appropriate actions in a given situation. It has to be able to find cues in an input situation that can be related to one or more possible action applications. In the current Rascalli system, these cues can be related to action selection via a rule based system, a top-down driven cognitive architecture, a bottom-up based knowledge acquisition and classification driven action selection.

5.3.1 Rule based action selection

The rule based action selection component 'Simple Mind' is a trivial implementation of a mind component, based on hard-coded action selection rules. These rules match to specific cues in the input data arriving from sensor channels. 'Simple Mind' extracts relevant information and passes this information on to the appropriate effector tool. Even though seemingly non-trivial behavior can be accomplished through a series of interactions of the Simple Mind and the available tools, the Simple Mind does not contain any cognitive aspects such as memory or learning.

5.3.2 Top-down driven action selection

The top-down action selection mechanism is implemented in the current version of the platform as the DUAL/AMBR cognitive architecture[8]. It includes a long term memory (LTM) where general and episodic knowledge is stored, and a working memory (WM) constituted by the active part of LTM, perceptual input and goals. The DUAL mind operates only on represented knowledge and has only a mediated connection to the body and the environment. Thus it contains a partial, selected representation of the environment at an abstract conceptual level and experiential memories related to specific episodes like organization of the interaction of a Rascalli agent with its environment, see [15] for more details.

5.3.3 Bottom-up driven knowledge acquisition

In the bottom-up approach, the Rascalli agents acquire knowledge about their environment by relating the input to their own action-perception capabilities. In this process the agents build their own "understanding" about the properties of the virtual entities that constitute the environment. The agents ground the knowledge about the entities by relating them to a set of actions applicable to those entities in a given input situation, as well as to the consequences of a given action application.
With the bottom-up knowledge acquisition, we aim at a robust mechanism which to a possible extent autonomously endows the agent with the knowledge necessary to perform its tasks. Since the agent deals with a dynamic environment as described in section 5.1, it is infeasible to provide ex-ante (e.g. through human expert knowledge) a full spectrum of knowledge required by the agent to perform its wide range of tasks and to adapt to the ever-occurring changes. Therefore we propose a learning mechanism that is sensitive to its environment, including the user activities within the system.
The major objective is the development of a cognitive component that enables the agents to gain knowledge, clearly different from human knowledge but grounded in interaction based self experience. This kind of grounded knowledge forms the basis for action selection within a Rascallo and allows modeling information exchange between Rascalli (see 5.5), as well as knowledge exchange between the Rascalli and their users (see 5.7).
Depending on the developmental stage of an individual Rascallo, tools are selected and executed arbitrarily, motivated by drives, or deliberately chosen based on the given input and previously made experience. The outcomes of the tools application on the environment (see 5.4) are also treated as an input to the sensor channels (see 5.2). In a way this is similar to a robot's perception of the consequences of an action application. For each action tool t_i an application space app_i is created over time that contains tool related interaction episodes. These episodes include the input situation short before the tool was applied and the outcome situation as perceived by the agent via its sensor channels. Feedback provided by the user (if available) is part of the outcome. Steps of the learning approach as 'partitioning the episodes', 'extraction of relevant sensor channels', 'feature extraction' and 'characterizing involved entities' [4] are applied to the application spaces to derive generalized input (c^i_n) and outcome characterizations (c^o_m) from the sets of stored episodes. The generalized input characterizations are - related to the affordances theory - cues for possible action applications. If a given input situation matches to one ore more of those generalized input characterizations (c^i_n), then the assumption can be made that this situation affords the application of the related tool t_i resulting in the expected outcome (c^o_m). The set of input characterizations (I), the set of tools (T), the set of outcome characterizations (O) and their interrelation form the knowledge repository I-T-O. As this repository is growing during further exploration of the environment including user feedback, it allows the Rascallo to act more purposefully on future input.

5.3.4 Classification driven action selection

The classification driven approach is a foundation of the agent action selection mechanism. Based on the available knowledge, including the perception of an input situation (task description, set of available actions and resources, user feedback received previously, etc.) the agent finds a set of actions that can be applied to this input situation. The selection of a particular action is based on the similarities with other actions the agent had successfully performed in the past, i.e. received positive feedback from the user. The action selection classifiers are implemented as Maximum Entropy[6] models. The data used for training the classifiers represents an input situation in terms of entities perceived by the agent and an action associated with it. The examples of the classification driven action selection application are e.g. a choice of a particular website which is returned as an answer to the user, based on the similarities between the current input and

previously experienced inputs that led the agent to decide to access a given website (e.g. wikipedia.org, news.google.com, youtube.com); a selection of a particular communication channel with the user (e.g. email, ECA, web-browser, music player) based on the type of data to be presented, or the user's preferences and his/her status (online, offline).

5.3.5 Bottom-up knowledge acquisition scenarios

The Rascalli agent acquires the knowledge about its environment, the agents own capabilities, task solving strategies and the user preferences via the following mechanisms:

- learning via self-experience about the entities that constitute the agent environment in the relation to the agent capabilities,
- learning via exploration of the agents environment and the interaction with the user,
- learning in the training mode from examples provided by an expert (human or other Rascalli agent),
- learning via the communication with other Rascalli agents using grounded and agreed upon symbols.

5.4 Action layers

The Rascalli agents actuators are geared towards performing actions necessary to assist the user in accessing information from the Internet, knowledge databases and communicating the findings to the user. The agent actions are realized as software tools the agent is equipped with, including:

- actions related to accessing information from the Internet and the databases - Question Answering System (see 5.4.2), Natural Language Database Query Interface (see 5.4.1), various Internet site specific accessing tools (wikipedia.org, dictionary.com, youtube.com, news.google.com, etc.),
- actions related to communication with the user - Multi-Modal Generation Tool (see 5.4.5),
- actions related to the agents perception[7] (equivalent to active perception in robotic and human)- the extensive set of components integrated in the Perception Layer, (see 5.2).

5.4.1 Natural Language Database Query Tool

The Natural Language Database Query Tool (T-Nalqi) is used in the Rascalli platform for querying the databases accessible to the Rascalli agents, in a search for instances and concepts that can provide answers to the user questions. The component analyses a user's natural language questions and retrieves answers from the system's domain-specific databases. The tool comprises the following three sub-components:
1. A relation extraction component, which finds patterns of pos-tags that represent relational structures and their arguments.
2. A relation-to-DB mapping component which identifies relations or concepts that are contained in a relational DB.
3. A query generation component which generates SQL queries and retrieves results from the database, and post-processes the results.
If the mapping is successful a query is formulated and executed, returning the information about question relevant instances and concepts.

[7] The concept of active perception, includes tool usage to sense the agent environment.

5.4.2 Question Answering System

The open-domain question answering system (T-QA) (which is based on the work described in [18], [16]) is used in the Rascalli platform to provide answers to the user factoid-type questions expressed in natural language. A typical use-case scenario involves a situation where an answer cannot be found or does not exist in the databases accessible to the Rascalli agents. The processing stages of the system include: question analysis, accessing the Internet resources, and analyzing the accessed documents. The system incorporates a number of natural language processing tools and resources (named entity recognition, part-of-speech tagger, text segmentation, chunker, stemmer, gazetteers, etc.), information retrieval tools (document indexing and querying engine) and machine learning based classifiers and clustering solutions. A set of tools usable to access a variety of Internet websites such as wikipedia.com, dictionary.com, howstuffworks.com, wordnet.org., and the Internet search engines such as Google, Yahoo, Altavista posses the capabilities to interpret the results of Internet resources, e.g. extract distinct definitions for a given term, report on ambiguity of a used term or possible misspellings, provide information on the number of available documents for a given query, distinct senses for a given term, the lack of term related documents or of a searched definition.

5.4.3 ChatBot

In the Rascalli project a wrapper to an existing ChatBot system was implemented. This component is based on the work undertaken in the A.L.I.C.E Artificial Intelligence Foundation[8], which aims at the creation of a natural language ChatBot capable to engage in the conversation with a human. In the Rascalli platform the current JAVA based implementation of the ChatBot can be used to handle unspecific user utterances of the type 'Are you a fish?' or 'Is it boring to be a computer?'.

5.4.4 RSS Feed Tool

The RSS Feed tool provides a mechanism for Rascalli agents to retrieve current information that might be of interest for the user. While technically RSS feeds have to be polled and retrieved (and thus obtaining information from an RSS feed is an active behavior), they can be easily modelled within the Rascalli system to be part of a dynamic and changing environment, with new feed items arriving in a manner that is temporally unpredictable for the agent. Thus a RSS feed tool which continuously (i.e. in very short intervals) polls and retrieves feeds that have been registered by a Rascalli agent without requiring any further intervention was implemented. As soon as new data is retrieved, the tool triggers a signal which is perceived by the Rascalli agent. The RSS feed tool includes a mechanism to filter news feeds for sets of keywords, allowing Rascalli to retrieve only relevant information of user interest based on user profiling (see 5.6).

5.4.5 Multi-Modal Generation Component

The Multi-Modal Generation Component provides a middle-ware functionality between generated agent output and the user interfaces. The generation component implements a template-based approach by encoding vocabulary, phrases, gestures etc. - which can be combined with the output of the Rascalli tools and context data - in the

[8] http://www.alicebot.org

form of Velocity templates. The use of Velocity[9], a template generation engine, allows to design and refine templates separately from the application code. For example the multi-modal generation component generates the speech, gestures and facial expressions for the Embodied Conversation Agent interface (see 5.7.1) used to communicate with the user. The output is encoded in an XML format that includes SSML (speech synthesis markup language) and BML[20] (behavior markup language) markup, making it interpretable by the MARY speech synthesis and the Rascalli User Interface (Nebula Client[10]).

5.5 Sharing knowledge - communication of grounded symbols between the agents

To enrich the task solving capabilities of the agents, which are equipped with symbol acquiring mechanisms, agent-to-agent communication was conceptualized in the Rascalli platform. This concept and its implementation allow agents to share their experiences (exchange knowledge) obtained through interaction with the environment[5]. Each agent has its own experience base comprising generalizations over the outcomes (O) of action applications, and generalizations of the input types (I) tools (T) that can be applied to. Due to the differences in their experience bases a negotiation process between Rascalli takes place establishing common labels for their individual knowledge of inputs and outcomes. Such an agreement process includes several cycles of exchanging prototypes and/or single episodes. For instance an agent A provides a prototype description *d* to another agent B, B tries to match *d* to its own experience base. Given a successful match, the agents choose a common label depending on the existence of labels from previous negotiations. In case the match at prototype level is not successful, the agents resort to instance level. This may lead to the creation of new prototypes in the agents which form a new basis for establishing a common label. If the negotiation at instance level fails, the agents' current experience bases are too distant. This however may change with further acquisition of knowledge. Using a set of labels, agents can exchange information more efficiently than by exchanging prototype representations or instance data. Since the shared labels are grounded in each agent's experience due to virtual embodiment and the affordance approach, knowledge exchange is possible even though the agents do not share the same internal representations. With the agreed labels, agents can exchange task solving strategies: which tools or tool chains to use in a given situation, how to react on a given input, or reach a desired outcome. For example, an agent may present a user input to another agent and ask for a recommendation what tool to use. This reduces the search space of individual agents for finding applicable tool or tool chain usage and increases the probability of satisfying the user.

5.6 User profiling

User profiling is based on logs from the user interactions with the system via selected user interfaces. The data are aggregated in a profiling component that allows the Rascalli agent to learn from the user actions about the interests and preferences of the respective user. These mechanisms are part of the agent assembly we term Smart Music Companion (6.3). The Smart Music Companion is an incarnation of a Rascalli agent that supports the user in finding and retrieving information on the popular music domain. Thus, the Music Companion is equipped with two special purpose tools/components which allow the user to browse a large music archive (currently up to 150 000 songs, see 5.7.2) and also access background information on the artists from a database comprising entries for approx. 15 000 individual artists and groups (5.7.3). Apart from these domain-specific interfaces, user action data are also collected from the ECA interface (5.7.1) which is the major user interface to the RASCALLI agent. Through this interface, the user may pose questions to her agent and evaluate the agent's responses, as well as introduce specific URLs and RSS feeds the agent should monitor for the user. The logs of the user actions contain the following information: user X does Y on item Z at time T. They are part of the (implicit) user profile which is used by different generator components of the profiling system in order to provide the agent with up-to-date information of the kind USER X LIKES/IS INTERESTED IN Z. This is crucial information for the Rascalli to adapt to their users and provide their users with (new) information that might be relevant given the user profile. For instance, when retrieving information from the domain databases (5.4.1), the Rascalli agent first of all provides information that is high ranked according to the user's current interest profile. Similarly, when applying the question answering system (5.4.2), first of all those web pages are analyzed which best fit the current user interests and/or have been explicitly introduced by the user to her agent. Combining the user interest profiles with an item-based recommender system that operates on the data from the music domain enables the Rascalli to propose new music/artists to the user. Moreover, the logs from the user actions may also be used as an input to the episode learning mechanism of the cognitive model DUAL (6.2).

5.7 Interfaces for agent-user interactions

In the following, we give an overview of the user interfaces which have been implemented up to date for the Rascalli agents.

5.7.1 Embodied Conversational Agent interface

While user input to the system via the ECA interface is constrained to free text input into a window of restricted size and pressing a praise and a scolding button (thumbs up/thumbs down), the system presents its output to the user via multi-modal, verbal and nonverbal behavior. See Figure 3, for a screenshot of the current design of the ECA interface, where the character sits in his fantasy room and waits for the user to type in her question. The output of the generation component is encoded in a BML-compliant format[20], which is then interpreted by a 3D animation component built on top of the Nebula platform[11] For speech synthesis the MARY TTS system [12] is employed. Figure 3 shows the ECA interface together with the Rascalli web page through which the user may introduce URLs and RSS feeds to the agent, and from which the music exploration interface and the music browser interface may be accessed. Through these interfaces the following user actions are logged: 1. the user utterance (text string), 2. clicks on the praise and scolding buttons, 3. switches to the music exploration and browsing interfaces, 4. user specified URLs and RSS feeds.

5.7.2 Music exploration interface

Figure 4 shows the exploration interface to the music archive the Rascalli have access to. The user may enter her favorite artist, song

[9] http://velocity.apache.org
[10] http://nebuladevice.cubik.org
[11] http://www.radonlabs.de/technologynebula2.html
[12] http://mary.dfki.de

Figure 3. Embodied Conversational Agent interface

or genre. The underlying application provides then a list of songs which belong to the selected artist or genre and computes for each song the user is interested in a list of songs that have similar acoustic properties. The best matching eight songs are displayed in a music map visualized in a 3x3 grid of album covers. Every suggested song can be listened to as a 30 second sample. The music player is invoked by either clicking on a cover or on an element in the play list. For each song played the user can state 'I like it!' (thumbs up) or 'No thanks!' (stop hand). From each song, there is a link to background information about the artist, which can be browsed via the music browser interface (5.7.3). All user clicks within the music exploration interface are assigned with a specific action label, logged and used as input to the profiling component (5.6).

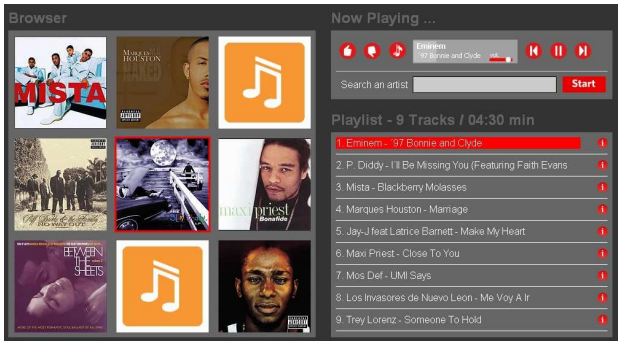

Figure 4. Music explorer interface

5.7.3 *Music browser interface*

Figure 5 shows the music browser interface which allows the user to explore the background information on a certain artist or group available to the Rascalli. This information has been harvested from the Internet, employing a seed-based information extraction methods which uses the already known in order to extract new, related information for text documents[21]. The browser provides not only facts about artists but also a network view of people related to them. Again, all user clicks on the links in the interface are logged. Each link is assigned a specific action label. The logs are in put to the profiling component (5.6).

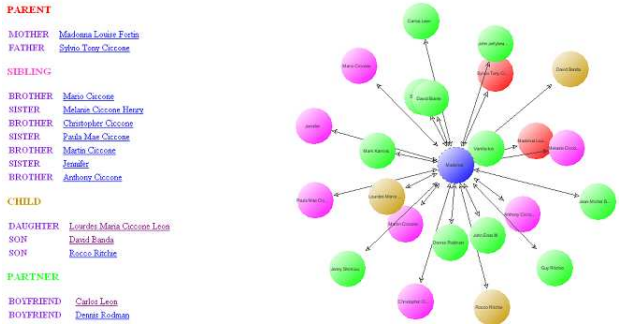

Figure 5. Music browser interface - 2

5.7.4 *Jabber interface*

A Jabber client interface (6) was developed as a simple user interface mechanism for the Rascalli system. It allows Rascalli agents to connect to the Jabber (an XML-based open source instant messaging protocol) network[13]. Various Jabber clients provide an easy-to use text-based interface by which the user can contact her Rascallo (the Rascallo can be added to the user's contact list just like other - human - contacts), and can be notified or contacted by her Rascallo in an unobstrusive manner.

Figure 6. Jabber client interface

5.7.5 *Command line client interface*

A plain version of the agent-user communication interface is realized as a command line client. This type of interface is especially useful for system development and testing.

[13] http://www.jabber.org

6 ASSEMBLING THE BUILDING BLOCKS - REALIZATION OF SPECIAL PURPOSE RASCALLI

Below we present how selected building block and interfaces presented in chapter 5 and section 5.7 were used and integrated in the Rascalli platform to create sample agents fulfilling particular goals.

Based on the Rascalli computational framework three different Rascalli configurations were conceptualized, developed and implemented. These are the initial implementation of a Rascalli agent with a basic set of tools (sandbox for testing the system components and their integration), a Rascalli agent utilizing an implementation of the DUAL cognitive architecture as central control unit, and a Smart Music Companion.

6.1 Sandbox for testing the system components and their integration

The goal of the sandbox application is to provide a method that allows to evaluate single components with respect to their feasibility for integration and their interoperability in the platform. For this purpose the building block 'simple mind' - a rule based action selection mechanism (see 5.3.1) has been developed. It allows the definition and application of a set of rules that describe how to react on a certain input/cue. Interaction with the user is handled via the ECA interface which is connected with the natural language input processing tool to analyse the user input and the multi-modal generation tool to produce the multi-modal output specification for the ECA. The command line interface (see 5.7.5) provides a fast and sufficient capability for developing and testing the system.
As a case-study example a particular system incarnation was assembled with the following building blocks: T-IP (see 5.2.2), 'simple mind', T-QA (see 5.4.2), T-ChatBot (see 5.4.3), T-Nalqi (see 5.4.1), T-MMG (see 5.4.5), ECA interface (see 5.7.1).
In addition to testing system integration, it was possible to assemble a system that was capable to realize a wide range of actions, providing a user with relevant information to her input. Due to the smart characteristics of the perceptor and actuator components (T-Nalqi, T-IP, T-QA) this was achievable even with a fairly simple rule-based action selection mechanism.

6.2 DUAL cognitive architecture as a control unit for a Music Information Assistant

As a cognitive enhancement to the initial Rascalli configuration, (see 6.1) a system incarnation was created that incorporates a proof-of-concept implementation of the DUAL cognitive architecture[8] as its central control unit. This implementation was accomplished for a very restricted set of show cases in the music domain. The implemented system incarnation is capable of providing answers to music and gossip related questions of the following kind: "Tell me something about Britney Spears", "Who are the children of Madonna?" or "Do you happen to know who is Madonna married to?".
A case-study example for the particular system incarnation was assembled with the following building blocks: T-IP (see 5.2.2), T-QA (see 5.4.2), T-Nalqi (see 5.4.1), T-MMG (see 5.4.5), ECA interface (see 5.7.1).
The goal of this Rascalli incarnation was to explore and further develop the cognitive model in order to prepare for reimplementation of selected cognitive functionality in a way that the system is able to scale up.

6.3 Smart Music Companions

The Smart Music Companion incarnation of the Rascalli agent provides information from the music domain, learns about user preferences, and uses this knowledge to present the user with new information that is related to her preferences[11]. The particular environment a Rascalli agent has to deal with in this application consists of: external knowledge sources, in particular the Internet and some domain-specific knowledge bases and the user. The application domain is popular music. Apart from the Internet, the Rascalli have access to the following two domain-specific knowledge resources: (1) a database with up to 150 000 tracks annotated with meta-information such as track name, album name, genre, artist or group name; (2) a database comprising background information of artists and groups such as band members, personal relations of artists, etc.
As case-study example for the particular system incarnation was assembled with the following building blocks: T-IP (see 5.2.2), 'Simple Mind' (see 5.3.1), User Profiling Component (see 5.6), T-QA (see 5.4.2), T-Nalqi (see 5.4.1), T-MMG (see 5.4.5), Music Exploration Interface (see 5.7.2), Music Browser Interface (see 5.7.3), ECA interface (see 5.7.1).
The user interacts with the companion via two kinds of interfaces: 1) The ECA interface where face-to-face dialog interaction between the embodied conversational companion and the user takes place. 2) The special purpose interfaces Music Explorer and Music Browser which support the user in exploring the Rascalli music databases. The special purpose interfaces are an addition to ECA for the following reasons 1) easiness to obtain information about the user preferences; 2) easiness and naturalness in providing unambiguous information to the system in an explicit way, which would be hard to achieve via the ECA interface due to erroneous natural language processing.

7 CONCLUSIONS

The RASCALLI computational framework introduced in this paper allows to flexibly add, remove and modify components according to the needs of a particular assembly of a system application. Due to the platform modularity and its well defined interfaces the integration of existing components is possible, and even preferred to the creation and re-implementation of new ones. In this way the platform can be continuously updated and enhanced, matching the state-of-the-art developments in intelligent artifacts. As demonstrated above, the platform provides already a sandbox for testing new and existing components and their interoperability. This development style is specifically geared towards research projects experimenting with alternative cognitive architectures, and combining them with a variety of processing and generation tools. The presented work demonstrates that the platform and its components support a prompt and purpose-driven creation of various incarnations of agents, where different cognitive components, action-perception tools and interfaces can be realized. The cognitive component can incorporate a wide range of approaches, including bottom-up driven, top-down driven or combined cognitive models. The Multi-Modal Generation Component as a middle-ware layer allows the usage of various user-agent interfaces without the need of modifying the agents output. This allows to operate in different contexts and to realize various agent appearances, different interaction behavior capabilities, etc.

The presented computational framework also provides an environment for a simultaneous existence of multi-agents, ranging from multiple instances of one agent configuration to multiple instances of agents assembled with different components. The agents have individual users and due to their unique interaction history their experience bases deviate from each other. As a result the collective experience base is diverse and covers a broader range of knowledge. To enable an agent to take advantage of this distributed knowledge, a method for communication between the agents in the platform was conceptualized. In addition the multi-agent platform provides the basis for the evaluation of the fitness of particular incarnations of agents, and thus helps to identify those agents that solve a given problem more effectively and purposefully than others. It also helps to explain why some agent types are more successful in their environment. The presented action-perception components are used by the agents to perceive their environment and perform actions on their environment such as accessing and processing information from the Internet and knowledge repositories. The three presented application scenarios exemplarily demonstrate the modularity, flexibility and integration capabilities of the processing tools and the platform.

Future work includes further enhancement of the platform, extended integration between the top-down and bottom-up cognitive approaches at a conceptual level and at execution level. The characteristic of the presented computational framework also enables further research on the goal orientated collaboration of multiple-agents and the usage of various communication strategies that facilitate distributed task solving.

ACKNOWLEDGEMENTS

This research is supported by the EC Cognitive Systems Project IST-027596-2004 RASCALLI, by the national FFG programme 'Advanced Knowledge Technologies: Grounding, Fusion, Applications' (project SELP), and by the Federal Ministry of Economics and Labour of the Republic of Austria. The work presented here builds on joint work within the Rascalli project. In particular the authors would like to thank our collegues from the following institutions: SAT, Radon, NBU, Ontotext and DFKI.

REFERENCES

[1] E. Brill, Transformation-based Error Driven Learning and Natural Language Processing: A Case Study in Part of Speech Tagging, Computational Linguistics 21(4), 543-566, (1995).

[2] Fellbaum, Wordnet: An Electronic Lexical Database, Bradford Books, (1998).

[3] G. T. Heineman and W. T. Councill, Component-Based Software Engineering: Putting the Pieces Together. Addison-Wesley Professional, Reading, (2001).

[4] J. Irran, F. Kintzler and P. Plz, Grounding Affordances. In proceedings: Trappl R. (ed.): Cybernetics and Systems 2006, Vienna: Austrian Society for Cybernetic Studies, (2006).

[5] J. Irran, G. Sieber, M. Skowron, B. Krenn, Acquisition and Exchange of Knowledge From Real to Virtual Embodiment, In Proceedings of the Symposium on Language and Robots, (2007).

[6] E.T. Jaynes, Information Theory and Statistical Mechanics, in Physical Review May 1957 Volume 106, No. 4 (pp. 620-630), (1957).

[7] F. Kintzler, et al., Affordance-related Robotics Research A Survey . Journal for research on adaptive behavior in animals and autonomous, artificial systems, (2007).

[8] B. Kokinov, A hybrid model of reasoning by analogy. In K. Holyoak and J. Barnden (Eds.), Advances in connectionist and neural computation theory: Vol.2. Analogical connections (Chapter 5, pp. 247- 318). Norwood, NJ: Ablex, (1994).

[9] S. Kostadinov, Petkov, M. Grinberg, Embodied conversational agent based on the DUAL cognitive architecture, International Conference on Web Information Systems and Technologies, WEBIST 2008, (2008).

[10] B. Krenn and C. Schollum, The RASCALLI Platform for a Flexible and Distributed Development of Virtual Systems Augmented with Cognition. In Proceedings of the International Conference on Cognitive Systems (CogSys 2008), (2008).

[11] B. Krenn, and G. Sieber, Functional Markup for Behaviour Planning: Theory and Prac-tice. In Proceedings of the AAMAS 2008 Workshop FML: Functional Markup Language. Why Conversational Agents do what they do. 12. -16. May 2008

[12] X. Li and D. Roth, Learning Question Classifiers, In Proceedings of the 19th International Conference on Computational Linguistics, pp. 556–562, (2002).

[13] Lin, Dependency-based evaluation of MINIPAR, In Proceedings of the Workshop on the Evaluation of Parsing Systems, First International Conference on Language Resources and Evaluation, (1998).

[14] E. Pampalk, S. Dixon and G. Widmer, On the Evaluation of Perceptual Similarity Measures for Music. In Proceedings of the 6th International Conference on Digital Audio Effects, (2003).

[15] G. Petkov, K. Kiryazov, M. Grinberg, and B. Kokinov, Modeling Top-Down Perception and Analogical Transfer with Single Anticipatory Mechanism. In Proceedings of EuroCogSci07: The second European Cognitive Science Conference, (2007).

[16] M. Skowron, A Web Based Approach to Factoid and Commonsense Knowledge Retrieval, Doctoral Dissertation, Hokkaido University, (2005).

[17] M. Skowron and K. Araki, Effectiveness of Combined Features for Machine Learning Based Question Classification. Special Issue of the Journal of the Natural Language Processing Society Japan on Question Answering and Automatic Summarization, Vol.12, No. 6, (2005).

[18] M. Skowron and K. Araki, What Can Be Learned from Previously Answered Questions? A Corpus-based Approach to Question Answering, Advances in Soft Computing. New Trends in Intelligent Information, Proceedings of the International IIS: IIPWM04 Conference, 379–387, (2004).

[19] V. Vapnik, The Nature of Statistical Learning Theory. Springer-Verlag, (1995).

[20] H. Vilhjalmsson et al., The Behavior Markup Language: Recent Developments and Challenges. In Proceedings of the 7th International Conference on Intelligent Virtual Agents, (2007).

[21] F. Xu, H. Uszkoreit and H. Li., A Seed-driven Bottom-up Machine Learning Framework for Extracting Relations of Various Complexity. In Proceedings of ACL 2007, (2007).