




A Web-Based Tool for Collaborative Modelling and Analysis in Human-Computer Interaction and Cognitive Science

Antonio Cerone^(✉) , Anel Mengdigali, Nuray Nabiyeva, and Temirlan Nurbay

Department of Computer Science, School of Engineering and Digital Sciences,
Nazarbayev University, Astana, Kazakhstan

{antonio.cerone,anel.mengdigali,nuray.nabiyeva,temirlan.nurbay}@nu.edu.kz

Abstract. Human-computer interaction and cognitive science are interdisciplinary areas in which computer scientists and mathematicians often work together with social scientists, such as psychologists and sociologists, as well as with more focussed practitioners such as usability experts and system analysts. In order to work effectively, interdisciplinary teams need to agree on a common communication language as a compromise between the computer scientists and mathematicians' formal modelling approach and the conceptual models normally used by social scientists for describing their domain-related theories and frameworks. Moreover, even when proper communication is established within a specific research team, the next challenge is the presentation of the result to a heterogeneous international community, to allow for cross-fertilisation, exchanges of ideas, sharing of models, results and data, work replication and review. This tool paper presents ColMASC, a web-based tool and portal for the collaborative modelling and analysis of human cognition and behaviour as well as interactive systems. ColMASC aim is for researchers in human-computer interaction and cognitive science to freely use the tool provided by the web portal in order to collaborate in the development of models of computer/physical systems as well as human behaviour, run in silico experiments, compare the results of in silico experiments and experiments with human beings, perform simulations, analyse systems consisting of computer/physical components and human components, as well as download and upload datasets and models. Domain oriented modelling and visualisation interfaces ease the modelling and analysis processes by hiding the simulation and formal analysis engines.

Keywords: Tool development · Formal methods · Collaborative research · Human-computer interaction · Cognitive science

Work partly funded by Project SEDS2020004 “Analysis of cognitive properties of interactive systems using model checking”, Nazarbayev University, Kazakhstan (Award number: 240919FD3916).

© Springer Nature Switzerland AG 2022

J. Bowles et al. (Eds.): DataMod 2021, LNCS 13268, pp. 175–192, 2022.

https://doi.org/10.1007/978-3-031-16011-0_12

1 Introduction

Research in cognitive science has resulted in the development of a large number of *cognitive architectures* over the last decades [6, 11]. Cognitive architectures are based on three different modelling approaches, *symbolic* (or *cognitivist*), such as Soar [7], which are based on a set of predefined general rules to manipulate symbols, *connectionist* (or *emergent*), such as DAC [13], which count on emergent properties of connected processing components (e.g. nodes of a neural network), and *hybrid*, such as CLARION [12], which combine the two previous approaches.

However, the complexity of these cognitive architectures makes it difficult to fully understand their semantics and requires high expertise in programming them. Moreover, Kotseruba and Tsotsos [6] note that most cognitive architectures have been developed for research purposes rather than for real-life usage. They are actually very specialised tools, each of them only usable within focussed research communities and normally capable to address only one of the following categories of application [6]: psychological experiments, robotics, human performance modelling, human-robot interaction, human-computer interaction, natural language processing, categorisation and clustering, computer vision, games and puzzles, and virtual agents. Finally, although cognitive architectures can mimic many aspects of human behaviour and learning, they never really managed to be easily incorporated in the system and software verification process.

In our previous work, we proposed a notation, the *Behaviour and Reasoning Description Language (BRDL)* [1], for describing human behaviour and reasoning. The semantics of the language is based on a basic model of human memory and memory processes and is adaptable to different cognitive theories. This allows us, on the one hand, to keep the syntax of the language to a minimum, thus making it easy to learn and understand and, on the other hand, to use alternative semantic variations to compare alternative theories of memory and cognition. In our previous work we have implemented parts of BRDL [2–4] using the Maude rewrite language and system [8, 9]. The automatic translation from BRDL to Maude facilitates modelling, but the results are still the formal textual output from Maude, which is difficult to interpret.

This paper describes ColMASC (Collaborative Modelling and Analysis of Systems and Cognition), a web-based tool and portal¹ that we are developing to address the widespread and heterogenous scientific community that is interested in the modelling and analysis of cognitive systems and interactive systems. The current version of the tool allows researchers to collaborate in modelling systems consisting of cognitive components (cognitive models), which describe human thinking and behaviour, and physical components (system models), which describe computer systems, electronic/mechanical devices, as well as any components without cognition or whose cognition is not relevant to the modelling context (e.g. animals or other humans, whose behaviour is observed or expected to occur in a specific way). Users can perform experiments on an overall model consisting of the composition of a cognitive component and a system component.

¹ colmasc.herokuapp.com/.

The tool is structured in terms of projects and enables model reuse within and between projects, thus fostering collaboration. Modelling is based on a linguistic approach [5, 10]. The *basic entities* of the model are combined to define syntactic structures that formally describe linguistic phrases. Phrase construction is driven by one of the tool interfaces by allowing the user to appropriately choose and correctly and unambiguously combine the syntactic elements of the phrase. Other interfaces support the guided creation of rules by filling their structural templates with the appropriate entities. Rules are then used to define cognitive and system models. The simulation of an overall model, given by combining a cognitive and a system model, can be carried out in steps and the output may be presented as a global state, as specific component states and even in natural language.

This approach make the tool usable by researchers working in different areas, from social scientists to computer scientists. In fact, the implicit linguistic knowledge of an average user is sufficient for creating meaningful basic entities and use them to create the rules defining the models, while the correctness of the syntax is guaranteed by the tool interfaces. Moreover, the alternative views for the presentation of the simulation results address different categories of users. For example, computer scientists would be interested in seeing the entire global state or just the system state, cognitive scientists would be interested in seeing the cognitive components, and psychologists and other social scientists would be interested in a presentation in natural language.

The paper is structured as follows. Section 2 presents our BRDL-based modelling approach and illustrates it informally using phrases in natural language as elements of the rule templates. Section 3 describes the formal construction of such elements using a linguistic approach. Section 4 describes the tool, the database, the web portal and the project management, using an example to illustrate how simulation is carried out. Finally, Sect. 5 summarises the current implementation of ColMASC and plan and discusses current and future development.

2 The Modelling Approach

In this section, we present our modelling approach in a top-down way, starting from cognitive and system models down to basic and structured entities.

2.1 Cognitive Model

A cognitive model consists of human memory components, in particular *long-term memory* (LTM), which has a virtually unlimited capacity and permanently stores the acquired knowledge, and *short-term memory* (STM), which has a small capacity and temporarily stores the information needed for cognitive processing. STM plays an important role in cognitive processing, since information must be normally transferred to STM in order to be processed. The STM temporary store is often called *Working Memory* (WM) when it is considered together with all its information processing functionalities.

Long-Term Memory (LTM) Model. Any piece of knowledge stored in LTM may be either a *fact*, which we can retrieve and transfer to STM in order to answer *questions* that we are processing in STM, or a rule, which is enabled by information in STM and/or by a perception in the environment and drives human thinking and behaviour. In this section we consider rules, whose definition and usage are already implemented in ColMASC. In Sect. 3.3 we will discuss facts and questions, which can currently be defined by the tool, but whose usage is still under development. ColMASC models LTM using six kinds of BRDL rules, which we call *LTM rules*. Each LTM rule has a general structure

$$g : info_1 \uparrow perc \implies act \downarrow info_2$$

where

- g is a goal
- $perc$ is a perception;
- act is an action;
- $info_1$ information to be removed from STM;
- $info_2$ information to be stored in STM.

Symbol \uparrow suggests removal from STM whereas symbol \downarrow suggests storage in STM. We call *enabling* the part of the rule on the left of \implies and *performing* the part of the rule on the right of \implies . Depending on which components are present an LTM rule models distinct cognitive activities as explained below and summarised in Table 1.

Table 1. Structure of LTM rules

Rule	Goal	STM removal	Perception	Action	STM storage
Automatic behaviour	Absent	Only one mandatory		Mandatory	Optional
Deliberate behaviour	Present	Only one mandatory		Mandatory	Optional
Implicit attention	Absent	Optional	Mandatory	Absent	Mandatory
Explicit attention	Present	Optional	Mandatory	Absent	Mandatory
Inference	absent	Mandatory	Absent	Absent	Mandatory
Planning	Present	Mandatory	Absent	Absent	Mandatory

Automatic Behaviour Rule: $info_1 \uparrow perc \implies act \downarrow info_2$

This rule models a basic activity of human automatic behaviour, that is, the performance of an action *act* as a response to the presence of some information *info₁* (which is not a goal) in STM and/or a perception *perc* from the environment, and may result in further information *info₂*, which may be a goal, stored in STM. Only *act* and one between *info₁* and *perc* are necessary, the other rule elements are optional. For example, an adult who is a dog lover, is perceiving a dog close enough to be touched and knows that the dog is friendly, will be automatically driven to perform the action of patting the dog and will know that

the dog is happy about it. Using natural language phrases as the BRDL rule elements, we can semi-formally describe this human basic activity as follows:

$$\begin{aligned} &\text{the dog is friendly} \uparrow \text{the dog is close enough} \\ &\implies \text{pat the dog} \downarrow \text{the dog is happy} \end{aligned} \quad (1)$$

Deliberate Behaviour rule: $g : \text{info}_1 \uparrow \text{perc} \implies \text{act} \downarrow \text{info}_2$

This rule models a basic activity of human deliberate behaviour, that is, the performance of an action *act* that is also driven by a goal *g* in STM. For example, a child who is not familiar with dogs and may also be scared by them will not be automatically driven to pat the dog, but may be willing to do so by setting the goal of patting in STM. Using natural language phrases as the BRDL rule elements, we can semi-formally describe this human basic activity as follows:

$$\begin{aligned} &\text{I want to pat the dog: the dog is friendly} \uparrow \text{the dog is close enough} \\ &\implies \text{pat the dog} \downarrow \text{the dog is happy} \end{aligned} \quad (2)$$

Implicit Attention rule: $\text{info}_1 \uparrow \text{perc} \implies \downarrow \text{info}_2$

This rule models a basic activity of human implicit attention, that is, the implicit selection of focusing on a specific perception *perc* from the environment and transfer such a perception to STM by representing it as information *info*₂. Such information may be a direct representation of the perception or include some form of processing. Only *perc* and *info*₂ are necessary, whereas *info*₁ is optional. For example, an adult who is a dog lover may be focussing on a wagging dog without any explicit will and without any goal. Using natural language phrases as the BRDL rule elements, we can semi-formally describe this human basic activity as follows:

$$\uparrow \text{the dog is wagging} \implies \downarrow \text{the dog is wagging} \quad (3)$$

Some form of implicit processing of the wagging perception could be carried out to directly acquire the knowledge that the dog is friendly:

$$\uparrow \text{the dog is wagging} \implies \downarrow \text{the dog is friendly} \quad (4)$$

It is also possible that the attention is driven by some information *info*₁ already present in STM. For example, the presence of the dog may have been noticed earlier and the information concerning it be already in STM:

$$\text{there is a dog} \uparrow \text{the dog is wagging} \implies \downarrow \text{the dog is wagging} \quad (5)$$

Explicit Attention rule: $g : \text{info}_1 \uparrow \text{perc} \implies \downarrow \text{info}_2$

This rule models a basic activity of human explicit attention, that is, the explicit selection, driven by goal *g*, of focusing on a specific perception *perc* from the environment. For example, a child who is not familiar with dogs may focus on the

dog's behaviour only after setting the intention to pat it as a goal. Using natural language phrases as the BRDL rule elements, we can semi-formally describe this human basic activity as follows:

$$\text{I want to pat the dog: } \uparrow \text{ the dog is wagging} \implies \downarrow \text{ the dog is wagging} \quad (6)$$

Inference rule $info_1 \uparrow \implies \downarrow info_2$

This rule models the inference of information $info_2$ from information $info_1$. It is based on logic, which may be deductive logic but also other forms of logic, such as inductive logic and abductive logic. It is pure reasoning independent of possible established goals, so that it can be used in any context. Moreover, neither $info_1$ nor $info_2$ may be a goal. For example, although an adult who sees a dog wagging may automatically internalise this perception by knowing that the dog is friendly, as we have seen in rule 4, a child who knows about dog behaviour but seldom interacts with dogs will need to explicitly apply the learned inference rule that a wagging dog is friendly:

$$\text{the dog is wagging } \uparrow \implies \downarrow \text{ the dog is friendly} \quad (7)$$

Planning rule $g : info_1 \uparrow \implies \downarrow info_2$

This rule models the mental planning defined by information $info_2$, which may be a goal, whereas, obviously, $info_1$ cannot be a goal. The planning is driven by goal g and depends on $info_1$. For example, a child who wants to pat the dog may establish the goal of following the dog if this leaves:

$$\text{I want to pat the dog: the dog leaves } \uparrow \implies \downarrow \text{ I want to follow the dog} \quad (8)$$

Short-Term Memory (STM) Model and Cognitive Processing. STM is modelled as a set of pieces of information. A piece of information may be

- a goal, which may be established initially or be produced by the application of an LTM rule, normally a planning rule;
- an element produced by the application of a LTM rule;
- a fact retrieved from LTM or observed in the environment;
- a question observed in the environment.

The content of STM provides the state of the cognitive model in terms of the information and goals that contribute to enable rules stored in LTM. Normally the initial state is either the empty STM or one or more goals, though some additional information may be added to set the context provided by a previous, non-modelled behaviour. Cognitive processing is carried out in two possible ways:

- by applying an LTM rule whose enabling part matches the content of STM and storing the information or goal from the performing part of the rule (which may be a representation of a perception from the environment) in STM;
- by retrieving, driven by a question in STM, a fact from LTM and storing a copy of it in STM.

The retrieval part is currently under implementation.

2.2 System Model and Interaction

A system that provides an environment for a given human behaviour is modelled as a transition system.

Transition rule $state_1 \xrightarrow{act} state_2$

This rule models the transition from state $state_1$ to state $state_2$ triggered by action act , which is the action performed by the user on the environment (interface, device, human/animal, etc.). The interaction between the human and the system is given through the synchronisation between an LTM rule and a transition rule, which share the same action, and by identifying $state_2$ of the transition rule with a new perception available for the user. For example, a dog who enjoys being patted will not go away

$$\text{the dog is close enough} \xrightarrow{\text{pat the dog}} \text{the dog is close enough} \quad (9)$$

whereas a dog who does not enjoy being patted will go away

$$\text{the dog is close enough} \xrightarrow{\text{pat the dog}} \text{the dog is far away} \quad (10)$$

Either transition rules may synchronise with one of the LTM rules 1 (automatic behaviour) or 2 (deliberate behaviour). Autonomous system behaviour with no interaction is also possible. In such a case, there is no action in the transition. For example, the dog may autonomously decide to go away:

$$\text{the dog is close enough} \longrightarrow \text{the dog is far away} \quad (11)$$

Autonomous behaviour can be observed by the user in terms of the target state ('far away' in LTM rule 11).

3 The Linguistic Approach

In Sect. 2 we have focussed on our modelling approach and used informal phrases as elements of LTM rules and transition rules. This allowed us to illustrate the modelling approach in a simple, understandable way. However, in order to allow cognitive and system models to be executable and be able to synchronise, it is necessary to structure the LTM rule and transition rule elements and associate them with semantics.

To this purpose, we exploit BRDL flexibility and extensibility and we use a linguistic approach to define the building blocks of our models, starting from *basic entities*, which are basically names associated with (linguistic) syntactic categories. Based on Chomsky's concept of *universal grammar* [5], we then combine basic entities using *deep structure* to produce *phrases* [10]. The resultant *structured entities* are then equipped with semantics that becomes operational when such *semantic entities* are used as components of *dynamic entities*, which are the rules, facts and questions we introduced in Sect. 2. Models are basically

sets of dynamic entities. The semantics embedded in the dynamic entities is reflected in the Java functions that define the simulation engine and, in future versions of the tool will be also reflected in the Maude rewrite rules that will define the analysis engine.

3.1 Basic Entities

A *basic entity* consists of three components:

name chosen by the user to provide a concise but meaningful characterisation of the entity linguistic and semantic role in modelling;

kind which characterises the entity syntactic role as a word;

definition a string of characters that is not as general as a dictionary definition but is specific to the abstraction level of the system under analysis and to the world of entities considered.

We can denote basic entities by their names. The *kind* of a basic entity may be one of the following:

Noun such as *child*, *lover* and *dog*;

Verb such as *leave*, *pat*;

Participle such as *wagging* (present participle) and *inserted* (past participle);

Adjective such as *friendly*, *happy* and *close*;

Adverb such as *enough*, *fast* and *slow*;

Auxiliary such as *can*, *has* and *is*;

Preposition such as *to*, *from*, *in*, *at* and *on*.

3.2 Structured Entities

A *structured entity* consists of the same type of components as a basic entity. However,

name may be structured and its purpose is to be used as an element in rule, fact and question definitions.

kind characterises the entity syntactic role as a phrase;

Structured entities are recursively defined on basic entities as follows:

- Any basic entity whose kind k is not Auxiliary or Preposition is a structured entity with kind k Phrase.
- Given a structured entity whose name is h and whose kind is k Phrase and a structured entity whose name is a and whose kind is compatible with k according to Table 2, we define a new structured entity with name $h(a)$, kind k Phrase and a new description that characterises $h(a)$

Table 2. Possible argument kinds for each kind of head in structured entities

Head Kind	Possible Phrase Kinds as Argument
Noun	Noun, Participle, Adjective, Preposition
Verb	Noun, Participle, Adverb, Preposition
Participle	Noun, Adverb, Preposition
Adjective	Noun, Adjective, Adverb, Preposition
Adverbs	Adverb
Auxiliary	Noun, Verb, Adjective, Preposition
Preposition	Noun

Therefore the name $h(s)$ of a structured entity consists of a part h called *head* and a possible part s called *argument*. It follows from the definition that the argument is mandatory when the head is the name of a basic entity of kind Auxiliary or Preposition, it is optional otherwise.

We can denote structured entities by their names. A structured entity may be of one of the following *kinds*:

Noun Phrase such as *child*, *lover*, *lover(dog)*, *dog(friendly)*, *child (running)*, *child(at(school))*;

Verb Phrase such as *leave*, *pat(dog)*, *go(shopping)*, *drive(fast)*, *go(to(school))*;

Participle Phrase such
as *wagging*, *inserted(bankcard)*, *inserted(completely)*, *coming(from(school))*;

Adjective Phrase such as *happy*, *slow(car)*, *high(fat)*, *close(enough (dog))*, *happy(at(school))*;

Adverb Phrase such as *enough*, *slowly(enough)*, *enough(slowly)*;

Auxiliary Phrase such as *has(legs)*, *can(move)*, *is(black)*, *is(on(table))*;

Preposition Phrase such as *in(box)*, *on(table)*.

3.3 Semantic and Dynamic Entities

A *positive declarative semantic entity* is a structured entity with the additional component:

semantics which characterises the set of its possible semantic roles.

Table 3 maps the kind of a semantic entity to its possible semantics.

Table 3. Possible semantics for each kind of semantic entity

Kind	Possible Semantics
Noun Phrase	Identifier, State, Information, Goal
Verb Phrase	Action, Goal
Participle Phrase	State, Information, Goal
Adjective Phrase	State, Information, Goal
Adverbs Phrase	State, Information, Goal
Auxiliary Phrase	Fact, Question, Goal
Preposition Phrase	State, Information, Goal

Any positive declarative semantic entity e may be turned into a *negative declarative semantic entity* by changing its name to $not(e)$. Any positive or negative declarative semantic entity e may be turned respectively into a *positive* or *negative interrogative semantic entity* by changing its name to $?(e)$. Any positive or negative declarative semantic entity e may be turned respectively into a *goal* by changing its name to $goal(e)$. Note that the application of *goal*, *not* or $?$ does not change the kind of the semantic entity. Therefore, we can now formally define STM as a set of semantic entities.

As an example of semantic entity consider the phrase “the dog is close enough” from LTM rule 1 in Sect. 2.1. The most important part in the context of the rule is “close enough”. In fact, regardless of whether the animal is a cat or a dog, the human can only pat it if the animal is close enough. Therefore, $close(enough)$ must be the head of the semantic entity and the phrase is formalised as $close(enough)(dog)$. This is a correct formalisation because, according to the compatibility in Table 2, Adjective *close* as head can have Adjective *enough* as an argument and Adjective Phrase $close(enough)$ as head can have Noun *dog* as an argument.

A *dynamic entity* is of one of the following three kinds:

- fact** which is a (positive or negative) declarative semantic entity;
- question** which is a (positive or negative) interrogative semantic entity;
- activity** which is the instantiation of a rule with names of semantic entities.

The instantiation of the rules should match the following instantiation for the semantics of the rule elements:

LTM rules Information \uparrow State \implies Action \downarrow Information
transition rules State $\xrightarrow{\text{Action}}$ State

As an example, State is one possible semantics for an Adjective Phrase, as shown in Table 3. Thus Adjective Phrase $close(enough)(dog)$ can instantiate the State element of a cognitive rule, as in the formalisation below of rule 1 from Sect. 2.1. In fact, rules 1–8 from Sect. 2.1 can be rewritten as dynamic entities (activities) as follows:

1. $\text{friendly}(\text{dog}) \uparrow \text{close}(\text{enough})(\text{dog}) \implies \text{pat}(\text{dog}) \downarrow \text{happy}(\text{dog})$
2. $\text{goal}(\text{pat}(\text{dog})): \text{friendly}(\text{dog}) \uparrow \text{close}(\text{enough})(\text{dog}) \implies \text{pat}(\text{dog}) \downarrow \text{happy}(\text{dog})$
3. $\uparrow \text{wagging}(\text{dog}) \implies \downarrow \text{wagging}(\text{dog})$
4. $\uparrow \text{wagging}(\text{dog}) \implies \downarrow \text{friendly}(\text{dog})$
5. $\text{present}(\text{dog}) \uparrow \text{wagging}(\text{dog}) \implies \downarrow \text{wagging}(\text{dog})$
6. $\text{goal}(\text{pat}(\text{dog})): \uparrow \text{wagging}(\text{dog}) \implies \downarrow \text{wagging}(\text{dog})$
7. $\text{wagging}(\text{dog}) \uparrow \implies \downarrow \text{friendly}(\text{dog})$
8. $\text{goal}(\text{pat}(\text{dog})): \text{dog}(\text{left}) \uparrow \implies \downarrow \text{goal}(\text{follow}(\text{dog}))$

Our notation is obviously not as expressive as natural languages. In fact, as shown in Table 3, its aim is to express, in a meaningful, unambiguous way: names of components (Identifier), system states (State), information in STM (Information) including goals (Goal), actions performed by the human (Action), facts that are part of the human knowledge or are perceived and possibly learned by the human (Fact) and questions that are perceived from the environment or internally elaborated through self-reflection (Question).

The fact “A dog is an animal” is formally modelled as

– $\text{is}(\text{dog})(\text{animal})$

Its negation is

– $\text{not}(\text{is}(\text{dog})(\text{animal}))$

and the corresponding questions are

– $?(\text{is}(\text{dog})(\text{animal}))$

– $?(\text{not}(\text{is}(\text{dog})(\text{animal})))$

4 Tool and Web Portal

A high-level visual description of the ColMASC tool and web portal is provided in Fig. 1. The thick boxes with names in bold represent components already developed and included in the current release (Version 1.0). The other components are still under development.

The purpose of ColMASC is to allow an interdisciplinary community of scientists and practitioners to model cognitive and interactive systems, share their models and collaborate with each other, both within the modelling process itself and by providing feedback and reviews. The ColMASC modelling approach aims at addressing the knowledge domain of a variety of modellers, including computer scientists, interaction designers, usability analysts, cognitive scientists, psychologists and linguists. The current version features a Java-based engine, which is described in Sect. 4.1. A Maude-based Analysis Engine is currently under development.

4.1 Java-based Simulation Engine

Purpose of the Java-based engine is to provide system simulation and presentation of the results. This simulation engine has been integrated into the server-side, which is a Spring Boot project using Java 11 and the Maven framework.

Since an overall system produces, in general, a non-deterministic behaviour, it could be possible to run the simulation in several modes, depending on how non-determinism is resolved. In the current implementation, choices are deterministically made by the simulation engine, consistently with the way simulation is carried out in the Maude system. The user can choose, instead,

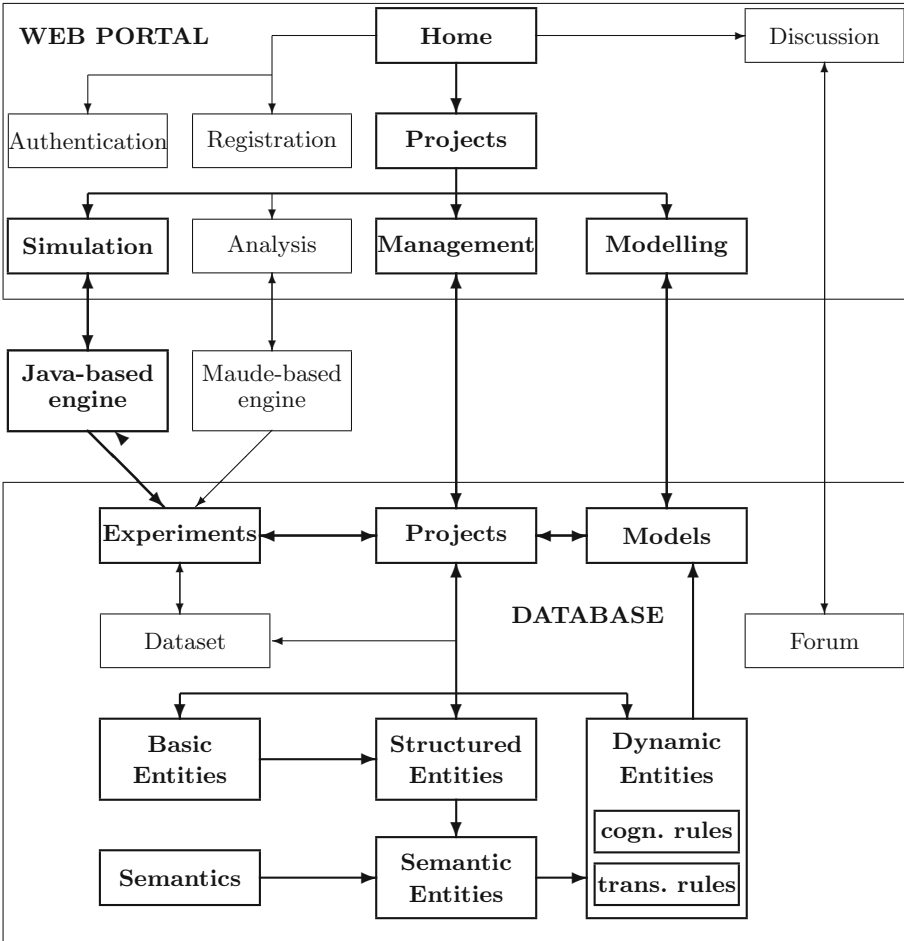


Fig. 1. High-level description of the tool architecture

- either a *stepwise* simulation, in which the system state is presented after each rule application and the user may move forward and backward through the steps,
- or an *all steps* simulation, which shows all steps in one screen.

Furthermore, the user may select which of the following information to include in the presentations:

- System State** which only presents applied transition rules and resultant system states;
- STM Content** which only presents applied LTM rules and resultant STM contents;
- Natural Language** which use natural language to informally describe the application of the rules and their effects.

Two further simulation modes are currently under development:

- making the choices *randomly*
- making the choices *interactively* (stepwise simulation only), whereby the user resolves non-determinism.

Finally, the storage of the results of the simulation in the ColMASC database is also under development.

As an example of simulation, consider the following cognitive model *Child* of a child who is willing to pat a dog, but has not developed any automatic behaviour for this task. This may be modelled by the following LTM rules

1. $goal(pat(dog)): \uparrow wagging(dog) \Rightarrow \downarrow wagging(dog)$
2. $goal(pat(dog)): wagging(dog) \uparrow \Rightarrow \downarrow friendly(dog)$
3. $goal(pat(dog)): friendly(dog) \uparrow far(dog) \Rightarrow approach(dog) \downarrow friendly(dog)$
4. $goal(pat(dog)): friendly(dog) \uparrow close(enough)(dog) \Rightarrow pat(dog) \downarrow happy(dog)$

The dog is actually the ‘system’ and may be modelled by the cognitive model *Dog* consisting of the following transition rules:

1. $far(dog) \xrightarrow{approach(dog)} close(enough)(dog)$
2. $close(enough)(dog) \longrightarrow far(dog)$
3. $sleepy(dog) \longrightarrow wagging(dog)$
4. $wagging(dog) \longrightarrow sleepy(dog)$
5. $sleepy(dog) \xrightarrow{pat(dog)} wagging(dog)$
6. $wagging(dog) \xrightarrow{pat(dog)} wagging(dog)$

Note that the dog is modelled from the human perspective. Transition rule 1 of *Dog* models that if the dog is far from the child, the child can approach it and the dog will be close enough to the child (in order to be patted). Transition rule 2 of *Dog* models that if the dog is close enough to the child (in order to be patted), the dog may autonomously decide to get far away from the child.

If the goal $goal(pat(dog))$ is the initial STM content and $far(dog)$ and $sleepy(dog)$ make up the initial system state, none of the LTM rules 1–4 of

Child is enabled. Only transition rule 3 of *Dog* is enabled and its occurrence changes the system state to *far(dog)* and *wagging(dog)*. Note that, although the ‘system dog’ may be approached when the system state contains *far(dog)* (transition rule 1 of *Dog*), the cognitive model *Child* enables approaching only after the dog has been assessed as friendly (LTM rule 3 of *Child*).

Once the system state has been changed to *far(dog)* and *wagging(dog)* there is a non-deterministic choice between the autonomous transition rule 4, which sets the dog back in a sleepy state, and the human explicit attention modelled by LTM rule 1, which may only be followed by the human inference modelled by LTM rule 2 (assuming that the dog is not getting sleepy). If the non-determinism is resolved by applying LTM rules 1 and 2 in sequence, *friendly(dog)* is added to the goal in STM (*wagging(dog)* is added by LTM rule 1 and then removed by LTM rule 2) while the system state is unchanged.

Now LTM rule 2 can synchronize with transition rule 1 and the system state changes to *close(enough)(dog)* and *wagging(dog)* while STM is unchanged. Figure 2 shows a screenshot of the presentation of this step in ColMASC. This presentation includes the applied LTM rule and/or transition rule, the natural language description and the formal description of STM content and system state. Note that in the step in Fig. 2 the dog is wagging, but LTM rule 3 is not dependent on this and can be applied also if the dog is no longer wagging and is sleepy instead.

Interaction human-dog
Simulation of Overall Model
Child and Dog (abstract)

simulation succeeded

☒ Natural Language

☒ System State

☒ STM Content

STM Capacity: 7

Run: stepwise

Step 5

Natural Language: Since the dog is friendly and dog is far Child does approach dog to pat dog (goal) and realizes that dog is friendly.

Applied Transition: *far(dog)* - *approach(dog)* → *close enough(dog)*

System State: *wagging(dog)*, *close enough(dog)*.

Performed Action: *approach(dog)*

Applied rule: *goal(pat(dog))*: *friendly(dog)* ↑ *far(dog)* ⇒ *approach(dog)* ↓ *friendly(dog)*

STM Content: *goal(pat(dog))*, *friendly(dog)*.

Go Back

Previous

Next

Download Results

Fig. 2. Screenshot of stepwise simulation

Finally, LTM rule 4 can be applied, again independently on whether the dog is wagging or sleepy, establishing the knowledge that the dog is currently happy (*friendly(dog)* is replaced by *happy(dog)* in STM). However, the synchronisation of LTM rule 4 is with transition rule 5, if the dog is wagging, or with transition

rule 6, if the dog is sleepy. But in both cases the dog will be wagging after being patted.

4.2 Database, Web Portal and Project Management

All defined entities and models are collected in a shared relational database. We have chosen PostgreSQL as the database management system for its power, high flexibility, and support for a wide range of data types and functions. The database is accessed with REST-controllers. The back-end, developed with Java 11 on Spring Boot, acts as middleware between the REST client and the database in order to validate the client input and format the output for presentation.

The portal allows researchers to access the database, create new entities and models as well as reuse existing entities and reuse and expand existing models, and use them to run experiments. We are currently developing a discussion forum which will support not just messaging and providing feedback, but also proposing features and reviewing models.

The current version of the *Projects* page is illustrated in Fig. 3. Each project is represented by a card showing title, development stage, an illustrative picture and a brief textual description. The card links to the *Specific Project Page*.

Projects are categorised according to their development stages, inspired by the development stages normally used for open source software products:

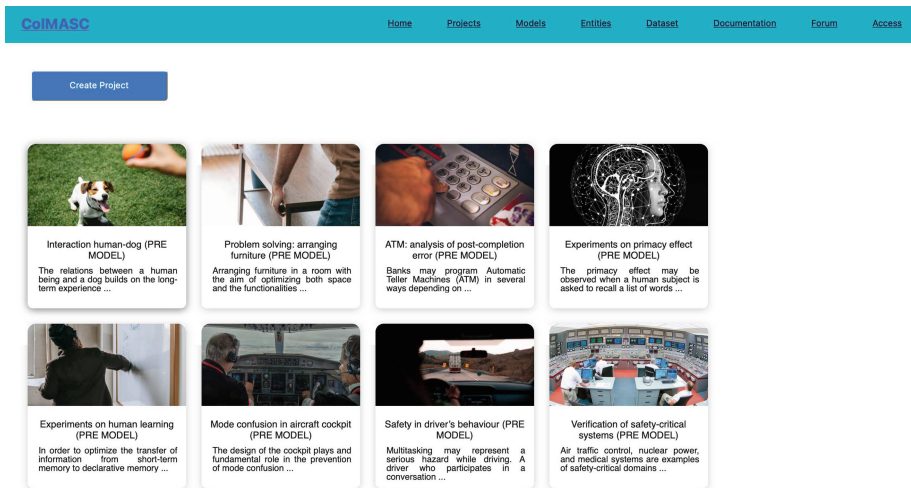


Fig. 3. Project main page

- Pre-model** It is a planning stage and comprises a description but no models, although models may be under construction at this stage.
- Pre-alpha** This stage is entered when at least one model is created. During this phase some tentative overall models may also be created and tested. Models are mostly unstable and subject to frequent changes. In general, models are tested using simple experiments and toy case studies.
- Alpha** Some models have reached a significant maturity level and may be used in real-world case studies with real datasets. The system architecture may still be changed and details and functionalities are added incrementally.
- Beta** The system architecture is now stable and includes all major functionalities. Most models are extensively used in real-world case studies with real datasets. The results of in silico experiments and simulations are compared with the results of real-world experiments and the outcome is used to calibrate component models and/or the overall model.
- Stable** All models are stable and can be now used as research tools to analyse cognitive science theories and to carry out system simulation.
- Re-beta** Stable overall models are reworked by changing the component models or starting the development of new models. The system architecture is not modified. Testing and usage are the same as for the Beta stage.
- Re-alpha** It builds up on a stable version but modifies the system architecture. Testing and usage are the same as for the Alpha stage.

The project may go back to the Pre-alpha stage if both system architecture and component models are heavily changed. Development stages are more dynamic and fluid than in software development. It is expected that for a number of projects their stages are continuously ‘oscillating’ between Stable and Re-alpha/Re-beta. This would be a typical situation when research outcomes are the main goals of the project.

5 Conclusion and Future Work

We have presented ColMASC 1.0, a tool and web portal that allows various categories of scientists and practitioners, including computer scientists, interaction designers, usability analysts, cognitive scientists, psychologists and linguists, to carry out collaborative research in human-computer interaction and cognitive science. Collaboration involves both the modelling process itself and testing and review activities. With the current version of the tool, cognitive scientists and psychologists may carry out in silico simulations to mimic and accelerate experiments with human subjects, aiming at testing theories in cognitive psychology, and computer scientists and usability experts may automatically generate formal models of computer interfaces and simulate their interaction with a human

component. In fact, ColMASC 1.0 supports the composition of a cognitive model and a system model, and the untimed simulation of the resultant overall model. Additional features are currently under development:

- storage of the results of experiments as well as external datasets in the database and analytical features to compare results;
- extension of the simulation engine to fact retrieval from LTM and their processing aiming to both self-reflection and answering questions;
- development of an analysis engine by defining a translation to Maude and exploiting the Maude model-checking capabilities;
- development of a time extensions of the simulation and analysis engines.

These additional features will support cognitive scientists and psychologists in the modelling of long-term learning processes, linguists in performing in silico experiments to emulate human processing of texts and language learning processes and computer scientists in the formal verification of interactive systems. The tool is currently undergoing empirical evaluation with potential users from both the computer science and the cognitive science sides.

The current version of the tool features the automatic translation of the experiment outcomes into natural language. As part of our future work we aim at having various customised natural language translation that address the different domain expertises of users. We also will explore the possibility of considering task descriptions in natural language from which to extract structured entity definitions and infer their semantics, and then recombine these two kinds of information to build semantic and dynamic entities.

The Maude-based analysis engine will make use of Real-Time Maude [8,9], the real-time extension of Maude. The analysis engine will build on our previous work [2–4]. The Maude code will run in parallel to the backend code on a Linux virtual machine. In this way the web application will have access to the Maude system through bash scripts to ensure consistency of simulation results.

Although our web-based tool addresses research collaboration in human-computer interaction and cognitive science, the same approach may be used in other application domains. Other possible application domains are coordination model, socio-technical system, systems biology and ecology.

References

1. Cerone, A.: Behaviour and Reasoning Description Language (BRDL). In: Camara, J., Steffen, M. (eds.) SEFM 2019. LNCS, vol. 12226, pp. 137–153. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-57506-9_11
2. Cerone, A., Murzagaliyeva, D.: Information retrieval from semantic memory: BRDL-based knowledge representation and Maude-based computer emulation. In: Cleophas, L., Massink, M. (eds.) SEFM 2020. LNCS, vol. 12524, pp. 159–175. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-67220-1_13
3. Cerone, A., Ölveczky, P.C.: Modelling human reasoning in practical behavioural contexts using Real-Time Maude. In: Sekerinski, E., et al. (eds.) FM 2019. LNCS, vol. 12232, pp. 424–442. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-54994-7_32

4. Cerone, A., Pluck, G.: A formal model for emulating the generation of human knowledge in semantic memory. In: Bowles, J., Broccia, G., Nanni, M. (eds.) *Data-Mod 2020*. LNCS, vol. 12611, pp. 104–122. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-70650-0_7
5. Chomsky, N.: *Language and Mind*. Cambridge University Press, Cambridge (2006)
6. Kotseruba, I., Tsotsos, J.K.: 40 years of cognitive architectures: core cognitive abilities and practical applications. *Artif. Intell. Rev.* **53**(1), 17–94 (2018). <https://doi.org/10.1007/s10462-018-9646-y>
7. Laird, J.A.: *The Soar Cognitive Architecture*. MIT Press, Cambridge (2012)
8. Ölveczky, P.C.: Real-Time Maude and its applications. In: Escobar, S. (ed.) *WRLA 2014*. LNCS, vol. 8663, pp. 42–79. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-12904-4_3
9. Ölveczky, P.C.: *Designing Reliable Distributed Systems*. UTCS, Springer, London (2017). <https://doi.org/10.1007/978-1-4471-6687-0>
10. Pinker, S.: *The Language Instinct*. William Morrow, New York (1994)
11. Samsonovich, A.V.: Towards a unified catalog of implemented cognitive architectures. In: *Biologically Inspired Cognitive Architectures (BICA 2010)*, pp. 195–244. IOS Press (2010)
12. Sun, R., Slusarz, P., Terry, C.: The interaction of the explicit and implicit in skill learning: a dual-process approach. *Psychol. Rev.* **112**, 159–192 (2005)
13. Verschure, P.: Distributed adaptive control: a theory of the mind, brain, body nexus. *Biol. Inspir. Cogn. Architect.* **1**, 55–72 (2012)