Cognitive Aspects in the Formal Modelling of Multi-party Human-computer Interaction*

Antonio Cerone[0000-0003-2691-5279] and Olzhas Zhalgendinov

Department of Computer Science, School of Engineering and Digital Sciences, Nazarbayev University, Astana, Kazakhstan
antonio.cerone@nu.edu.kz olzhas.zhalgendinov@nu.edu.kz

Abstract. The unpredictability of human behaviour makes formal analysis of human-computer interaction (HCI) already difficult when one single user is interacting with one computer system. However, nowadays interaction normally involves several users, i.e., human components, interacting simultaneously with separate interfaces which communicate with and/or act upon the same computer resources. Independently of whether human components aim at using computer resources to communicate with each other or to concurrently act upon and modify stored information, interfaces should be designed in order to protect users from the potential negative effects caused by the behaviour of other users.
In this research paper, we define a framework that combines the formal modelling of cognitive aspects of human components and the modelling of the actual computer system into an overall model that can be formally analysed using model-checking. We use the Behaviour and Reasoning Description Language (BRDL) to model human cognition and implement our framework using Real-time Maude, whose model-checker is then used to carry out formal verification.

Keywords: Human-computer Interaction · Human Cognition · Behaviour and Reasoning Description Language · LTSs · Formal Analysis.

1 Introduction

Although attempts to conceptualise human-computer interaction (HCI) and multi-party interaction date back to the late 1980s and early 1990s [6, 10, 15, 16], most of HCI research still focuses nowadays on interactions between a single user and a single system [7] and we are still far from a unified theory of HCI. This in spite of the fact that current interactive systems involve a large variety of interaction modalities for dialog integration, from web forms to spoken dialogs up to sophisticated multimodal systems, to support access to shared data and their manipulation and modification, to carry out computer-supported collaborative work (CSCW), to share own resources and even to just interact socially. Thus multiple users interact with the same system and, through a single system

* Work partly supported by CIDMA (Research Center of Mathematics and Applications) and the University of Aveiro through the SEFM local organization.

or a number of interconnected systems, with each other. The resultant forms of interaction involve simultaneous actions and reactions, which have to be consistent and should not negatively affect each other [17]. This makes it difficult to conceptualise multi-party HCI and even more difficult to tackle it using formal approaches. As a result, also formal approaches to HCI tend to consider a single user and a single system [18] and neglect the verification of those properties that emerge from the possibly implicit user-to-user interaction mediated by an interconnected computer system.

In this paper, we propose an approach to the formal modelling of interactive systems in which multiple users interact with an interconnected system of servers and clients. We model computer systems using an extension of labelled transition systems (LTS) that supports communication via asynchronous messaging between system components and interaction with users. We model human tasks using the Behaviour and Reasoning Description Language (BRDL) [4, 5], which characterises human memory and memory processes and the way they determine human behaviour in terms of cognitive rules that are enabled by the user's goal, mental state and perception. Rule execution determines actions on the system and evolution of the user's mental state. The labels of LTS transitions play the double role of receiving a message from another system component and reacting to a user's action. A further extension of both LTSs and BRDL is the use of time in the form of a timing interval that determines the minimum and maximum delay for the rule execution.

The rest of the paper is organised as follows. Sect. 2 introduces the LTS extension and Sect. 2.1 illustrates it on a course registration system example. Sect. 3 introduces the BRDL notation and Sect. 3.1 illustrates it on the example. Sect. 4 describes the communication between the extended LTSs and the interaction between systems and human components, while Sect. 4.1 illustrates them on the example. Sect. 4.2 shows the use of model-checking to formally verify two properties of the course registration system example. Sect. 5 concludes the paper and discusses possible future work.

2 Computer System Model

In order to model interconnected computer systems with asynchronous communication between each other and time constraints, we extend labelled transition systems by introducing messages that can be received and consumed through the transition labels. Message content production is modelled by adding it together with the recipient identifier to a component of the current state that we assume to instantaneously deliver the pair recipient-content to the network while keeping track of the set of sent messages and their recipients. Moreover, in order to support interaction with users, the system state is partitioned into an invisible part, which is used internally, and a visible part, which is used as output in the interaction, while transition labels can also assume the role of input in the interaction.

This approach allow us to uniformly model interaction, whether between two system components or between a human component and a system components. In fact, the transition label models the system component's reception of either another system component's message or a human action.

Definition 1 (CLTS). *A Communicating Labelled Transition Systems (CLTS) is an tuple* $\mathcal{S}_n = \langle \mathbb{I}, n, V_n, I_n, \mathbb{T}, A_n, M_n, T_n, \mathcal{V}_n, \mathcal{I}_n \rangle$ *where*

- $\mathbb{I}$ *is an identifier domain;*
- $n \in \mathbb{I}$ *is the CLTS identifier;*
- V_n *is a set of visible atomic states;*
- I_n *is a set of invisible atomic states;*
- $\mathbb{T}$ *is a time domain;*
- A_n *is a set of actions;*
- M_n *is a set of messages;*
- $T_n \subseteq 2^{V_n} \times 2^{I_n} \times (A_n \cup M_n) \times \mathbb{T}^2 \times 2^{V_n} \times 2^{I_n} \times 2^S$, *with*

$$S = \{sent(i, m) \mid i \in \mathbb{I} \backslash \{n\} \wedge m \in M_n\}$$

being a set of sent messages together with their recipients, is a set of transition rules whose elements are represented with the following syntax

$$visible_1 \; [invisible_1] \xrightarrow[{[a,b]}]{lab} visible_2 \; [invisible_2] \; msgSent$$

where $a, b, \in \mathbb{T}$, *sets* $visible_1, visible_2 \in 2^{V_n}$ *and* $invisible_1, invisible_2 \in 2^{I_n}$ *are denoted by elements separated by commas,* $lab \in A_n \cup M_n$, *and* $msgSent \in 2^S$*;*
- $\mathcal{V}_n \subseteq V_n$ *is the current set of visible atomic states;*
- $\mathcal{I}_n \subseteq I_n$ *is the current set of invisible atomic states;*

such that $A_n \cap M_n = \emptyset$ *and* V_n, I_n, A_n *are pairwise disjoint.*

Note that messages can also be elements of V_n or I_n.

A CLTS evolves starting from the initial state consisting of a set of initial visible atomic states and a set of initial invisible atomic states and, as explained in Sect. 4. also depending on the set of sent messages and respective recipients available in the environment.

The execution of the transition rule is triggered by the label *lab* and the inclusion of $visible_1$ in the set of initial visible atomic states and $invisible_1$ in the set of initial invisible atomic states. The label *lab* may be either a message received from another CLTS or an action performed by a user.

A delay between a minimum a and a maximum b must occur before the transition is executed. When $a = b$, the time interval $[a, a]$ is shortened as a.

2.1 Example: A Course Registration System

We model a course registration system consisting of a central server that stores the information about user registrations and instances of client websites running locally for various students. Students interact with the system through the interface provided by the website. We model our system using separate CLTS, $\mathcal{S}_0$ for the central server and $\mathcal{S}_i$, with $i \in \mathbb{N}\backslash\{0\}$, for the i-th website instance. For sake of simplicity, we only consider two students using two separate client websites, modelled by $\mathcal{S}_1$ and $\mathcal{S}_2$, and just one course with a cap of one student, that is, only one student can register for the course. In this way, when one of the two students successfully registers for the course, the system should prevent the other student from registering for that same course. For each of our CLTSs we use a distinct natural number $n \in \mathbb{N}$ as an identifier and a time domain $\mathbb{T} = \mathbb{N}$, with a time unit of 1 millisecond (ms).

Central Server When students intend to register for a course, they first check the availability of the course. The server waits for such a request in the form of a message and returns a new message depending on the availability of the course. Then the server waits for a student's message that requests to register for the course. When the registration message is received, the server makes the course unavailable, so that other students cannot register.

The initial configuration $\mathcal{S}_0 = \langle \mathbb{N}, 0, V_0, I_0, \mathbb{N}, A_0, M_0, T_0, \mathcal{V}_0, \mathcal{I}_0 \rangle$ of the server CLTS is defined as follows:

- $V_0 = \emptyset$,
- $I_0 = \{available, unavailable, registered_i, labRegistered_i\}$,
- $A_0 = \emptyset$,
- $M_0 = \{check_i, register_i, checkLabs_i, labChosen_i, labInfo_i, labFailed_i\} \cup I_0$,
- $\mathcal{V}_0 = \emptyset$;
- $\mathcal{I}_0 = \{available\}$;
- T_0 consists of the following transition rules:

$$[available] \xrightarrow[{[10,1000]}]{check_i} [available]\ sent(i, available) \tag{1}$$

$$[unavailable] \xrightarrow[{[10,1000]}]{check_i} [unavailable]\ sent(i, unavailable) \tag{2}$$

$$[available] \xrightarrow[{[10,1000]}]{register_i} [unavailable, registered_i] \tag{3}$$

$$[unavailable] \xrightarrow[{[10,1000]}]{register_i} [unavailable] \tag{4}$$

$$[] \xrightarrow[{[10,1000]}]{checkLabs_i} []\ sent(i, labInfo) \tag{5}$$

$$[] \xrightarrow[{[10,1000]}]{checkLabs_i} []\ sent(i, labFailed) \tag{6}$$

$$[] \xrightarrow[{[10,1000]}]{labChosen_i} [labRegistered_i] \tag{7}$$

Each of these transitions may take between 10 and 1000 ms, as modelled by the $[10, 1000]$ time interval. This delay reflects both the server processing time and the network transmission time. Moreover, $\mathcal{S}_0$ does not have visible states, since user's visibility only occurs at the level of the client websites. Instead, all transitions are labelled with messages received from the clients. In fact, the user cannot interact directly with the server and, as a result, the set of actions of the CLTS that models the server is empty.

Transition rules 1 and 2 model the response of the server to the student's check on course availability, which was sent through the website whose identifier is i. The received $check_i$ message triggers transition rule 1 or 2, depending on whether the invisible state of $\mathcal{S}_0$ contains *available* or *unavailable*, respectively. When executed, the transition rules send the course availability invisible atomic state to the website i that has sent the check request.

Transition rules 3 and 4 model the response of the server to the student's request for course registration. The received *register* message triggers transition rule 3 or 4, depending on whether the invisible state of $\mathcal{S}_0$ contains *available* or *unavailable*, respectively. When executed, transition rules 3 replaces the *available* invisible atomic state with *unavailable* and adds the invisible atomic state $registered_i$, whereas transition rules 4 consume the messages without modifying the state.

Transition rules 5 and 6 model the response of the server to the student's request for lab information by sending either the information or a failure message to the website i that has sent the check request.

Transition rule 7 models the response of the server to the student's choice to enrol in the lab by adding $labRegistered_i$ to the invisible state.

Note that only transition rules 1 and 3 may be enabled in the initial configuration ($\mathcal{I}_0 = \{available\}$) provided that the appropriate message is received ($check_i$ and $register_i$, respectively).

Client Website The client website waits for the page to be refreshed in order to get relevant course information. When the student clicks on the *refresh button*, the client sends a request to the server for checking the course availability. Then the client waits for the response and visualises the course availability according to the server response. If the course is available, the student can click the *enrol button* to send the registration message to the server and feedback on the successful enrolment is visualised. At this point, the student needs to click the *proceed button* and wait for the next page to load. This may take some time and it can fail due to external reasons such as too much internet traffic or temporary unavailability of the server. In case of failure, the student needs to click the *proceed button* again. Finally, the student must choose whether to also register for a lab. Clicking the *register lab button* allows the student to register for a lab whereas clicking the *proceed button* once again allows the student to skips this step. This choice ends the registration process.

The initial configuration $\mathcal{S}_i = \langle \mathbb{N}, i, V_i, I_i, \mathbb{N}, A_i, M_i, T_i, \mathcal{V}_i, \mathcal{I}_i \rangle$, with $i \in \mathbb{N} \backslash \{0\}$, of the i-th client website CLTS is defined as follows where

– $V_i = \{emptyPage, waiting, available, unavailable, enrolled, chooseLab, noLab, labRegistered\}$,
– $I_i = \{labs, loadingLabs\}$,
– $A_i = \{refresh, enrol, proceed, registerLab\}$,

- $M_i = \{check_i, available, unavailable, register_i, checkLabs_i, labInfo_i, labFailed_i, labChosen_i\}$,
- $\mathcal{V}_i = \{emptyPage\}$;
- $\mathcal{I}_i = \emptyset$;
- T_i consists of the following transition rules:

$$emptyPage\ []\ \overset{refresh}{\underset{0}{\longrightarrow}}\ waiting\ []\ sent(0, check_i) \tag{8}$$

$$waiting\ []\ \overset{available}{\underset{0}{\longrightarrow}}\ available\ [] \tag{9}$$

$$waiting\ []\ \overset{unavailable}{\underset{0}{\longrightarrow}}\ unavailable\ [] \tag{10}$$

$$available\ []\ \overset{enrol}{\underset{0}{\longrightarrow}}\ enrolled\ [labs]\ sent(0, register_i) \tag{11}$$

$$enrolled\ [labs]\ \overset{proceed}{\underset{0}{\longrightarrow}}\ enrolled\ [loadingLabs]\ sent(0, checkLabs_i) \tag{12}$$

$$enrolled\ [loadingLabs]\ \overset{labInfo_i}{\underset{0}{\longrightarrow}}\ chooseLab\ [] \tag{13}$$

$$enrolled\ [loadingLabs]\ \overset{labFailed_i}{\underset{0}{\longrightarrow}}\ enrolled\ [labs] \tag{14}$$

$$chooseLab\ []\ \overset{proceed}{\underset{0}{\longrightarrow}}\ noLab\ [] \tag{15}$$

$$chooseLab\ []\ \overset{registerLab}{\underset{0}{\longrightarrow}}\ labRegistered\ []\ sent(0, labChosen_i) \tag{16}$$

All these transitions are instantaneous, as modelled by the $[0, 0]$ time interval represented by the shortening 0 in the notation. In fact, the delay experienced by the user is actually the delay in the response from the server, which we have considered in the model $\mathcal{S}_0$ of the server.

Transition rule 8 models the website waiting for the *refresh* user's action. Then the visible state is instantaneously changed by removing the *emptyPage* atomic state and adding the *waiting* atomic state, which provides feedback to the user that the client website is waiting for an answer from the server. In fact, message $check_i$ is sent to the $\mathcal{S}_0$ server to request information about course availability.

Transition rules 9 and 10 model the arrival of the response from the server as the message that labels the transition (*available* or *unavailable*) and instantaneously change the visible state by removing the *waiting* atomic state and adding the received message.

Transition rule 11 models the *enrol* user's action, which is enabled when the visible state contains *available*. It instantaneously removes such a visible atomic state, adds the *enrolled* visible atomic state, as a feedback to the user, and sends a registration message to the $\mathcal{S}_0$ server. Moreover, it adds the *labs* invisible atomic state to enable transition rule 12, which starts the optional lab registration process.

Transition rule 12 models the *proceed* user's action enabled after the course registration has succeeded. A message $checkLabs_i$ is sent to the $\mathcal{S}_0$ server to request information about labs and the *labs* invisible atomic state is instantaneously replaced by *loadingLabs*, to enable the reception of the response from the server.

Transition rules 13 and 14 model the successful reception of a response from the server and a failure in receiving the response, respectively. If the information is received successfully (label $labInfo_i$ in rule 13) the *enrolled* visible atomic state and the *loadingLabs* invisible atomic state are instantaneously replaced by the *chooseLab* visible atomic state to enable the lab choice. If there is a failure due to external reasons (label $labFailed_i$ in rule 14) the *loadingLabs* invisible atomic state is instantaneously replaced by the *chooseLab* invisible atomic state to enable a new request through rule 12.

Transition rule 15 models the *proceed* user's action that is enabled when the visible state is *chooseLab*. The effect of the action is to skip the lab choice and instantaneously replace the *chooseLab* visible atomic by *noLab*.

Transition rule 16 models the *registerLab* user's action, which is enabled when the visible state is *chooseLab*. The effect of the action is to register for a lab by instantaneously sending the $labChosen_i$ message to the $\mathcal{S}_0$ server and replacing the *chooseLab* visible atomic state by *labRegistered*.

Note that only transition rule 8 may be enabled in the initial configuration ($\mathcal{V}_i = \{refresh\}$) provided that the appropriate action (*refresh*) is performed by the user.

3 User Cognitive Model

The user's behaviour is modelled in terms of the way cognition affects the actions performed in response to visible states (*interface*) of the computer system. In this section, we recall the *Behaviour and Reasoning Description Language (BRDL)* [4, 5], a notation for modelling user's knowledge. BRDL is based on the *information processing theory*, which was developed in cognitive psychology in the 1950s, and describes human thinking as a computational process with at least two levels of information storage: *long-term memory (LTM)* that stores knowledge and *short-term memory (STM)* that temporarily stores information used for processing [1]. This approach has originated a number of conceptual models of human memory in cognitive psychology, from the basic distinction between LTM and STM [1], to the most complex versions of the Multistore Working Memory Model [2].

The semantics of BRDL is based on the basic model of human memory [1]. In this paper, we are interested in the user's knowledge of the interface. This is modelled in BRDL in terms of *cognitive rules* that are stored in LTM and drive the user's response to visible states of the interface, which we call *perceptions* from the user's perspective. That is, the user perceives the visible state of the interface and responds to it by performing an action, either automatically or according to the current mental states and a goal. BRDL provides a standard structure of cognitive rules, with full flexibility concerning the complexity of its components, which may vary from just mnemonic identifiers or phrases in natural language to complex data structures. 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 for practitioners, and, on the other hand, to use semantic variations that correspond to alternative theories of memory and cognition and to combine BRDL models of the user with any formal notation that models the computer components.

User's current mental states and goals are stored in STM. Mental states reflect thinking or, more specifically, reasoning and decision making. A goal is determined by what is intended to be achieved in terms of mental state or action performed. We represent a mental state as a set of pieces of information that may contain perceptions. A number of processing activities are carried out on the information stored in STM. Thus STM together with such processing activities makes up what is called human *working memory (WM)* [2]. Typical processing activities of WM are: *attention*, which stores perceptions in STM, *retrieval*, which copies information from LTM to STM, and *inference*, which applies rules stored in LTM to transform information in STM.

Definition 2 (Goal). *Let A be a set of actions, P a set of perceptions and H a set of pieces of information, with $P \subseteq H$ and A and H disjoint. The notation $goal(G)$, with $G \in 2^{A \cup H}$, denotes the goal that is achieved when*

- *either one of the actions in $G \cap A$ is performed;*
- *or one of the pieces of information in $G \cap H$ is in STM;*

We call G set of achievements.

The set of goals on A and H is denoted by $\mathcal{G}_{A,H}$.

Definition 3 (STM Models). *Let $\mathbb{T}$ be a time domain, A a set of actions, P a set of perceptions and H a set of pieces of information, with $P \subseteq H$ and A and H disjoint. An* STM *model of capacity $n \in \mathbb{N}$ and decay time $d \in \mathbb{T}$ on $\mathbb{T}$, A and H, is a set $\mathcal{H} \subseteq (A \cup H \cup \mathcal{G}_{A,H}) \times \mathbb{T}$ of cardinality lower than or equal to n and such that for each $p \in A \cup H \cup \mathcal{G}_{A,H}$ and $t_1, t_2 \in \mathbb{T}$, if $(p, t_1), (p, t_2) \in \mathcal{H}$, then $t_1 = t_2$.*

The set of STM models *of capacity $n \in \mathbb{N}$ and decay time $d \in \mathbb{T}$ on $\mathbb{T}$, A and H is denoted by $STM^{n,d}_{\mathbb{T},A,P,H}$.*

The decay time d denotes how long the information persists in STM. The time associated with the piece of information is called *lifetime*. The lifetime equals d when the piece of information is stored in STM, then decrease with the passing of time, and the piece of information disappears from STM when its lifetime becomes 0. The capacity of STM that is widely accepted in psychology is 7 ± 2 pieces of information, as determined by Miller's experiments in 1956 [9]. Such capacity limits are based on the numbers of chunks of information that can be held in STM and are also supported by modern experimental cognitive psychology [11] In our models, in order to be safe, we normally set STM capacity to 5.

As an example, we may consider $STM^{5,2700}_{\mathbb{N},A,P,H}$, where

- $A = \{refresh, enrol, proceed, registerLab\}$;
- $P = \{emptyPage, available, enrolled, chooseLab, labRegistered\}$;
- $H = P$.

The elements of A are the actions of the client websites $\mathcal{S}_i$, with $i \in \mathbb{N}\backslash\{0\}$, introduced in Sect. 2.1, which are actually instances of the same website. The elements of P are visible states of those $\mathcal{S}_i$, that is, perceptions from the user's viewpoint. The *enrolled* piece of information models the user's feeling of having enrolled in the course. We may consider two goals $goal(enrolled), goal(registerLab) \in \mathcal{G}_{A,H}$. The user feels to have achieved goal $goal(enrolled)$ when the feedback provided by the system determines an *enrolled* mental state ($enrolled \in P = H$) and to have achieved goal $goal(registerLab)$ when performing the action of registering for the lab ($registerLab \in A$).

Definition 4 (LTM Model). *An* LTM model *is a tuple* $\mathcal{L} = \langle H, P, \mathbb{T}, A, C\rangle$ *where*

- H *is a set of pieces of information;*
- $P \subseteq H$ *is a set of perceptions;*
- $\mathbb{T}$ *is a time domain;*
- A *is a set of actions;*
- $C \subseteq \mathcal{G}_{A,H} \times 2^H \times 2^P \times \mathbb{T}^2 \times A \times 2^{(H \cup \mathcal{G}_{A,H})}$ *is a set of cognitive rules whose elements are represented with the following syntax*

$$goal: \; info_1 \uparrow perc \underset{[a,b]}{\Longrightarrow} act \downarrow info_2$$

where $a, b, \in \mathcal{T}$, $goal \in \mathcal{G}_{A,H}$, $info_1 \in 2^H$, $perc \in P$, $act \in A$ *and* $info_2 \in 2^{H \cup \mathcal{G}_{A,H}}$,

with H *and* A *disjoint.*

In cognitive rules, the $\uparrow$ symbol suggests removal from STM whereas the $\downarrow$ symbol suggests storage in STM. We call *enabling* the part of the rule on the left of $\Longrightarrow$ and *performing* the part of the rule on the right of $\Longrightarrow$. The execution of a cognitive rule is enabled by the presence of goal *goal* and information $info_1$ in STM, and by the perception *perc* from the environment, and results in the removal of $info_1$ from STM, the performance of action *act* on the environment and the storage of new information $info_2$ in STM. Note that $info_1$ only contains pieces of information whereas $info_2$ may also contain goals. In fact, when the goal *goal* has a nonempty set of achievement, it is the only goal enabling the rule, while goals that may be in $info_2$ are actually produced in STM by performing the rule. A goal without achievements (empty set of achievements) denotes the absence of actual goal. In this case the syntax of a cognitive rule is shortened as

$$info_1 \uparrow perc \underset{[a,b]}{\Longrightarrow} act \downarrow info_2$$

Definition 5 (UCM). *Let* $\mathbb{T}$ *be a time domain,* A *a set of actions,* P *a set of perceptions and* H *a set of pieces of information, with* $P \subseteq H$ *and* A *and* H *disjoint. A* User Cognitive Model (UCM) *with STM capacity* $n \in \mathbb{N}$ *and STM decay time* $d \in \mathbb{T}$ *on* $\mathbb{T}$, A *and* H *is a tuple* $\mathcal{U} = \langle \mathbb{J}, n, \mathcal{L}_n, \mathcal{H}_n\rangle$ *such that*

- $\mathbb{J}$ *is an identifier domain;*
- $n \in \mathbb{J}$ *is the UCM identifier;*
- $\mathcal{L} = \langle H, P, \mathbb{T}, A, C\rangle$ *is an LTM model;*
- $\mathcal{H} \in STM^{n,d}_{\mathbb{T},A,P,H}$ *is the* initial STM model.

3.1 Example: A Registration Task

In the context of the course registration system, the user has a goal to register for the course and may also have an additional goal to register for a lab. In Sect. 3, we modelled these goals as $goal(enrolled), goal(registerLab) \in \mathcal{G}_{A,H}$, respectively.

The registration task of a student who intends to also register for a lab is modelled in BRDL as the UCM

$$\mathcal{U}_{withLab} = \langle \mathbb{N}, withLab, \langle H, P, \mathbb{N}, A, C\rangle, \mathcal{H}_{withLab}\rangle$$

where

- $P = \{emptyPage, available, enrolled, chooseLab, labRegistered\}$,
- $H = P$,
- $A = \{refresh, enrol, proceed, registerLab\}$,
- C consists of the following cognitive rules:

$$\uparrow emptyPage \underset{[100{,}300]}{\Longrightarrow} refresh\downarrow \tag{17}$$

$$goal(enrolled) :\uparrow available \underset{[100{,}300]}{\Longrightarrow} enrol\downarrow \tag{18}$$

$$goal(enrolled) :\uparrow enrolled \underset{[100{,}300]}{\Longrightarrow} \downarrow enrolled \tag{19}$$

$$goal(enrolled) :\; enrolled\uparrow \underset{[100{,}300]}{\Longrightarrow} proceed\downarrow enrolled \tag{20}$$

$$goal(noLab) :\uparrow chooseLab \underset{[100{,}300]}{\Longrightarrow} proceed\downarrow \tag{21}$$

$$goal(noLab) :\uparrow noLab \underset{[100{,}300]}{\Longrightarrow} \downarrow noLab \tag{22}$$

$$goal(registerLab) :\uparrow chooseLab \underset{[100{,}300]}{\Longrightarrow} registerLab\downarrow \tag{23}$$

and the initial STM model is

$$\mathcal{H}_{withLab} = \{(goal(enrolled), 2700), (goal(registerLab, 2700))\}$$

In terms of timing we assume that the access to LTM requires between 100 ms and 300 ms and that the decay time of the information in STM is 2700 ms. This timing is suggested by the most recent experimental evidence in cognitive psychology [3].

Cognitive rule 17 does not have a goal part. It models that the user automatically reacts to the perception of an empty page ($emptyPage \in P$) by refreshing ($refresh \in A$).

Cognitive rules 18–20 are driven by the $goal(enrolled)$ goal. Rule 18 models the action of enrolling in the course ($enrol \in A$) triggered by the perception of course availability ($available \in P$). Rule 19 models the attention to the feedback of the system that shows successful enrolment ($enrolled \in P$): the perception is transferred to STM ($enrolled \in H$). Rule 20 models clicking the proceed button ($proceed \in A$) once becoming aware that the enrolment was successful ($enrolled \in H$) and preserves the content of STM ($enrolled$ appears as information removed from STM in the enabling part of the rule as well as information stored in STM in the performing part of the rule).

Cognitive rules 21 and 22 are driven by the $goal(noLab)$ goal. Rule 21 models the action of skipping the lab registration by clicking the proceed button ($proceed \in A$) again and is triggered by the perception of the alternative choice (which is not taken) of choosing lab registration ($chooseLab \in P$). Rule 22 models the attention to the feedback of the system that shows that registration to lab has been successfully skipped ($noLab \in P$): the perception is transferred to STM ($noLab \in H$). Note that $\mathcal{H}_{withLab}$ cannot evolve to an STM model that enables these two cognitive rules, since $goal(noLab)$ is not in $\mathcal{H}_{withLab}$ and is not generated by any cognitive rule in C.

Cognitive rule 23 is driven by the $goal(registerLab)$ goal. It models clicking the register lab button ($chooseLab \in A$) once becoming aware that the course enrolment was successful ($enrolled \in H$).

A student

$$\mathcal{U}_{withoutLab} = \langle \mathbb{N}, withoutLab, \langle H, P, \mathbb{N}, A, C \rangle, \mathcal{H}_{withoutLab} \rangle$$

who does not intend to choose a course has instead an initial STM model

$$\mathcal{H}_{withoutLab} = \{(goal(enrolled), 2700, (goal(noLab, 2700))\}$$

which cannot evolve to an STM model that enables cognitive rule 23. Instead, $\mathcal{H}_{withoutLab}$ can evolve to an STM model that enables cognitive rules 21 and 22.

4 Overall System Model and Dynamics

An overall system model consists of computer components modelled as CLTSs and human components modelled as UCMs. Each UCM has to interact with one and only one CLTS, which models a system interface. Therefore, interaction can be modelled as an injective function between UCMs and CLTSs. To facilitate modelling we index CLTSs and UCMs on their identifiers as we did for the CLTSs of the course registration system example introduced in Sect. 2.1. Then we can define an injective function between the sets of identifiers to characterise the interaction.

Definition 6 (OSM). *Given two sets of identifiers $\mathbb{I}$ and $\mathbb{J}$ and a time domain $\mathbb{T}$, let*

– $\{\mathcal{S}_i\}_{i \in \mathbb{I}}$ *be a family of CLTSs on time domain $\mathbb{T}$,*

– $\{\mathcal{U}j\}_{j\in\mathbb{J}}$ *be a family of UCMs on time domain* $\mathbb{T}$,
– $\varphi : \mathbb{J} \longrightarrow \mathbb{I}$ *be an injective function, which we call* interaction function,
– μ *be a set of sent messages together with their recipients*

Then $\mathcal{M} = \langle\{\mathcal{S}_i\}_{i\in\mathbb{I}}, \{\mathcal{U}j\}_{j\in\mathbb{J}}, \varphi, \mu\rangle$ *is an* Overall System Model (OSM) *on time domain* $\mathbb{T}$.

Definition 7 (Interaction). *Let* $\mathcal{M} = \langle\{\mathcal{S}_i\}_{i\in\mathbb{I}}, \{\mathcal{U}j\}_{j\in\mathbb{J}}, \varphi, \mu\rangle$ *be an* Overall System Model (OSM) *on time domain* $\mathbb{T}$ *with*

– $\mathcal{S}_i = \langle\mathbb{I}, i, V_i, I_i, \mathbb{T}, A_i, M_i, T_i, \mathcal{V}_i, \mathcal{I}_i\rangle$, *for each* $i \in \mathbb{I}$
– $\mathcal{U}_j = \langle\mathbb{J}, j, \langle H_j, P_j, \mathbb{T}, A_j, C_j\rangle, \mathcal{H}_j\rangle$, *for each* $j \in \mathbb{J}$

If, there exists $j \in \mathbb{J}$ *such that*

$$(goal(info) :\ info_1 \uparrow perc \underset{[a,b]}{\Longrightarrow} act \downarrow info_2) \in C_j$$

and

$$(visible_1\ [invisible_1] \xrightarrow[{[c,d]}]{act} visible_2\ [invisible_2]\ msgSent) \in T_{\varphi(j)}$$

satisfy the following conditions

C.1 $(goal(info), t_0), (h_1, t_1), \ldots (h_n, t_n) \in \mathcal{H}_j$
for all $h_i \in info_1$, $i = 1, \ldots, n$, *and for some* $t_0, t_1, \ldots t_n \in \mathbb{T}$*;*
C.2 $perc \in visible_1$
C.3 $visible_1 \subseteq \mathcal{V}_{\varphi(j)}$*;*
C.4 $invisible_1 \subseteq \mathcal{I}_{\varphi(j)}$*;*

then the cognitive rule and the transition rule may synchronise on action act and, if the synchronisation occurs, $\mathcal{M}$ *evolves to* $\mathcal{M}'$ *where*

E.1 $\mathcal{V}_{\varphi(j)}$ *is replaced with* $\mathcal{V}'_{\varphi(j)} = \mathcal{V}_{\varphi(j)} \backslash visible_1 \cup visible_2$.
E.2 $\mathcal{I}_{\varphi(j)}$ *is replaced with* $\mathcal{I}'_{\varphi(j)} = \mathcal{I}_{\varphi(j)} \backslash invisible_1 \cup invisible_2$.
E.3 *Let be* $\mathcal{H}'_j = \mathcal{H}_j \backslash \{(h_i, t_i) \mid h_i \in info_1\} \cup \{(h, \Delta) \mid h \in info\}$, *with* Δ *denoting the STM decay time,* $\mathcal{H}_j$ *is replaced with*
  E.3.1 $\mathcal{H}'_j$, *if* $act \notin info$ *and there is no* $(h_i, t_i) \in \mathcal{H}_j$ *such that* $h_i \in info$.
  E.3.2 $\bar{\mathcal{H}}_j = \mathcal{H}'_j \backslash \{(goal(info), t_0)\}$, *otherwise.*
E.4 μ *is replaced with* $\mu' = \mu \cup msgSent$.

Synchronisation may occur if the cognitive rule and the transition rule share the same action *act*, which is performed by the user on the interface, *goal*(*info*) and all pieces of information h_i in $info_1$ are in STM, associated with their current lifetime t_i (condition **C.1**), the perception *perc* of the cognitive rule is in the visible source state $visible_1$ of the transition rule (condition **C.2**), the visible source state $visible_1$ of the transition rule is included in the visible component $\mathcal{V}_{\varphi(j)}$ of the current state of the CLTS (condition **C.3**) and the invisible source state $invisible_1$ of the transition rule is included in the invisible component $\mathcal{I}_{\varphi(j)}$ of the current state of the CLTS (condition **C.4**).

When the synchronisation occurs, the current state of the CLTS, the current content of the STM and the current set of sent messages and respective recipients evolves. The visible part $\mathit{visible}_2$ (evolution **E.1**) and the invisible part $\mathit{invisible}_2$ (evolution **E.2**) of the target state replace the visible part $\mathit{visible}_1$ and the invisible part $\mathit{invisible}_1$ of the source state in the $\mathcal{V}_{\varphi(j)}$ visible component and $\mathcal{I}_{\varphi(j)}$ invisible component, respectively, of the current state of the CLTS. The current content $\mathcal{H}_j$ of STM evolves by removing the timed version of info_1 information in the enabling part of the cognitive rule and by adding the timed version of info_2 information in the performing part of the cognitive rule (evolution **E.3.1**) and, if the shared action is in the goal achievements ($\mathit{act} \in \mathit{info}$) or one of the pieces of information in the *info* goal achievements has a timed version in STM, by also removing the timed version of goal $\mathit{goal}(\mathit{info})$ from STM, because it has been achieved and is no longer needed (evolution **E.3.1**). The current set of messages μ evolves by adding the messages $\mathit{msgSent}$ generated by the transition rule (evolution **E.4**). When more synchronisations are possible the choice is nondeterministic.

Definition 8 (Message reception). *Let* $\mathcal{M} = \langle \{\mathcal{S}_i\}_{i\in\mathbb{I}}, \{\mathcal{U}j\}_{j\in\mathbb{J}}, \varphi, \mu\rangle$ *be an* Overall System Model (OSM) *on time domain* $\mathbb{T}$ *and*

$$\mathcal{S}_e = \langle \mathbb{I}, e, V_e, I_e, \mathbb{T}, A_e, M_e, T_e, \mathcal{V}_e, \mathcal{I}_e\rangle$$

with $e \in \mathbb{I}$*, be an CLTS model in* $\mathcal{M}$*. If*

$$(\mathit{visible}_1\ [\mathit{invisible}_1] \xrightarrow[{[c,d]}]{rec} \mathit{visible}_2\ [\mathit{invisible}_2]\ \mathit{msgSent}) \in T_{e)}$$

satisfies the following conditions

C.3 $\mathit{visible}_1 \subseteq \mathcal{V}_e$*;*
C.4 $\mathit{invisible}_1 \subseteq \mathcal{I}_e$*;*
C.R $\mathit{sent}(e, \mathit{rec}) \in \mu$*;*

then the transition rule is enabled and, if it is executed, $\mathcal{M}$ *evolves to* $\mathcal{M}'$ *where*

E.1 $\mathcal{V}_{\varphi(j)}$ *is replaced with* $\mathcal{V}'_{\varphi(j)} = \mathcal{V}_{\varphi(j)} \backslash \mathit{visible}_1 \cup \mathit{visible}_2$.
E.2 $\mathcal{I}_{\varphi(j)}$ *is replaced with* $\mathcal{I}'_{\varphi(j)} = \mathcal{I}_{\varphi(j)} \backslash \mathit{invisible}_1 \cup \mathit{invisible}_2$.
E.R μ *is replaced with* $\mu' = \mu \backslash \{\mathit{sent}(e, \mathit{rec})\}$.

4.1 Example: User's Interaction with its Registration Website

The overall registration systems initial configuration is modelled by the OSM

$$\mathcal{M} = \langle \{\mathcal{S}_0, \mathcal{S}_1, \mathcal{S}_2\}, \{\mathcal{U}_1, \mathcal{U}_2\}, \varphi, \emptyset\rangle$$

where $\mathcal{S}_0$, $\mathcal{S}_1$ and $\mathcal{S}_2$ are defined as in Sect 2.1, $\mathcal{U}_1 = \mathcal{U}_2$ as $\mathcal{U}_{withLab}$ introduced in Sect. 3.1 and $\varphi : \{1, 2\} \longrightarrow \{0, 1, 2\}$ is defined by $\varphi(1) = 1$ and $\varphi(2) = 2$.

As an example of interaction, both user $\mathcal{U}_1$ and user $\mathcal{U}_2$ are initially enabled to interact with the corresponding interface, $\mathcal{S}_1$ or $\mathcal{S}_2$, respectively, as characterised by the interaction function φ. Thus we can say generically that $\mathcal{U}_i$ performs cognitive rule 17 and interacts with $\mathcal{S}_i$, which concurrently performs transition rule 8. Cognitive rule 17 has neither a goal nor an enabling information to be removed from STM, thus the initial visible state $\mathcal{V}_i = \{emptyPage\}$ of $\mathcal{S}_1$ provides the *emptyPage* perception that is sufficient to enable it. The performance of rule 17 determines the synchronisation on action *refresh* with transition rule 8, which is also enabled by the *emptyPage* visible atomic state. As a result of the synchronisation, there is no evolution of the user STM $\mathcal{H}_i$, since the performing part of the cognitive rule has no information to store in STM. Instead, the *waiting* visible atomic state replaces *emptyPage* in $\mathcal{V}_i$, resulting in $\mathcal{V}_i' = \{waiting\}$, and message $sent(0, check_i)$ is produced, changing $\mu = \emptyset$ to $\mu' = \{sent(0, check_i)\}$. This interaction requires a time between 100 and 300 ms, since $[100, 300]$ is the time interval associated with rule 17 while rule 8 is instantaneous.

As an example of message reception, transition rule 1 is enabled on the invisible state $\mathcal{I}_0 = \{available\}$ of $\mathcal{S}_0$ and on the evolved set $\mu' = \{sent(0, check_i)\}$ of sent messages and respective recipients. The performance of the rule results in no change of the invisible state but in the consumption of message $sent(0, check_i)$ and the production of message $sent(i, available)$, that is, in the evolution of μ' to $\mu'' = \{sent(i, available)\}$. This message reception requires a time between 10 and 1000 ms, since $[10, 1000]$ is the time interval associated with rule 1 that models the network delay.

4.2 Formal Analysis

Our approach is implemented using the Maude rewrite system [8, 13] and its real-time extension Real-Time Maude [14, 12]. Real-Time Maude model checker features a timed search command that traverses the reachable states using breadth first search and checks for each state whether it satisfies the search pattern and condition. We use the search command to formally verify two properties:

1. Only one student believes to have registered for the one place course;
2. A student cannot unintentionally skip the lab registration.

A reasonable time bound for the timed search command is $[0, 10000]$. It takes at most 1800 ms for the $UCM_{withLab}$ task to be completed by performing six of seven cognitive rules. In absence of network failures, it takes at most 5000 ms for the task to be served by the $\mathcal{S}_0$ central server by performing five of seven transition rules.

According to cognitive rule 19 the STM model stores the *enrolled* piece of information when the student achieves the goal of registering for the course. Therefore, for property 1 we can use

search pattern: $\langle S, \{\langle \mathbb{N}, 1, \mathcal{L}_1, \mathcal{H}' \rangle, \langle \mathbb{N}, 2, \mathcal{L}_2, \mathcal{H}'' \rangle\}, \varphi, M \rangle$
seach condition: $(enrolled, T') \in \mathcal{H}' \wedge (enrolled, T'') \in \mathcal{H}''$

where S is a placeholder for any set of CLTS, M is a placeholder for any set of snt messages, $T', T'' \in \mathbb{N}$ are placeholders for a time value, and $\mathcal{H}', \mathcal{H}''$ are placeholders for any STM. The search pattern looks for an OSM that contains two UCM models. The search condition specifies that both UCM models must contain the *enrolled* STM item. The search provides as a result an OSM evolution that satisfies the search condition thus showing that property 1 is not satisfied. The OSM evolution shows that the error occurs because in transition rule 4 the server does not provide the website client with the information about the registration failure. The erroneous belief could be prevented if transition rule 4 sends a failure message to the client website and this, upon reception, changes the visible state to provide the user with appropriate feedback.

In order to analyse property 2, we need to check whether the user may click the *proceed* button in rule 15 with intention of clicking it in rule 12. Therefore, for property 2 we can use

search pattern: $\langle \{\langle \mathbb{I}, N, V_N, I_N, \mathbb{N}, A_N, M_N, T_N, \mathcal{V}_N, \mathcal{I}_N \rangle\} \cup S, U, \varphi, M \rangle$
seach condition: $noLab \in \mathcal{V}_N$

where $N \in \{1, 2\}$ is a placeholder for the CLTS identifier, S is a placeholder for any set of CLTSs, U is a placeholder for any set of UCMs, and M is a placeholder for any set of sent messages and respective recipients. The search should result in an OSM with one website containing the *noLab* information

In a correct system behavior, this search would result in an OSM with one website containing the *noLab* visible atomic state only if the user's STM contains $goal(noLab)$ in the initial configuration. Instead, this also occurs when the user's STM contains $goal(registerLab)$ in the initial configuration, that is, also when the user intends to register for a lab. The OSM evolution shows that the error is due to the fact that the *proceed* action occurs both in transition rule 12 and transition rule 15. As a consequence, the user may keep clicking the *proceed* button to perform the interaction modelled by transition rule 12, whereas the performed *proceed* results in the execution of rule 15. The error could be prevented by using two distinct actions, that is, two distinct buttons, to enable the two rules.

5 Conclusion and Future Work

In this paper, we have defined a framework for the formal modelling and analysis of interactive systems in which multiple users interact with an interconnected system of servers and clients. To model such an interconnected system we have defined CLTSs, an extension of LTSs with asynchronous communication and time constraints. Our approach combines BRDL-based cognitive models of human components and the CLTS-based models of the computer system components into an overall system model (OSM). We have implemented our framework with Real-Time Maude and exploited its model-checking capabilities to formally verify two interaction properties of a course registration system example. The Maude code of the example presented in this paper can be downloaded from GitHub[1].

[1] https://antoniocerone.github.io/Publications/2024/CIFMA/

In our framework users are statically assigned to interfaces, as modelled by the interaction function φ. However, in the real world, users frequently need to switch attention between different interfaces. This is actually common in safety-critical systems, in which operators are requested to multitask by monitoring several interfaces at the same time, and is becoming more and more common in daily life, with multitasking emerging as a trendy life style. Therefore, in our future work, we are planning to extend our framework to dynamically assign users to interface.

References

1. Atkinson, R.C., Shiffrin, R.M.: Human memory: A proposed system and its control processes. In: Spense, K.W. (ed.) The psychology of learning and motivation: Advances in research and theory II, pp. 89–195. Academic Press (1968)
2. Baddeley, A.: The episodic buffer: a new component of working memory? Trends Cogn Sci **4**(11), 417–423 (2000). https://doi.org/10.1016/s1364-6613(00)01538-2, https://www.ncbi.nlm.nih.gov/pubmed/11058819
3. Campoy, G.: Evidence for decay in verbal short-term memory: A commentary on berman, jonides, and lewis (2009). Journal of Experimental Psychology: Learning, Memory, and Cognition **38**(4), 1129–1136 (2012)
4. Cerone, A.: Behaviour and reasoning description language (BRDL). In: SEFM 2019 Collocated Workshops (CIFMA), Lecture Notes in Computer Science, vol. 12226, pp. 137–153. Springer (2020)
5. Cerone, A.: Modelling and analysing cognition and interaction. In: Formal Methods for an Informal World, Lecture Notes in Computer Science, vol. 13490, pp. 30–72. Springer (2023)
6. Dowell, J., Long, J.: Towards a conception for an engineering discipline of human factors. Ergonomics **32**, 1513–1535 (1989)
7. Kirchhoff, K., Ostendorf, M.: Directions for multi-party human-computer interaction research. In: Proc. of HLT-NAACL 2003, pp. 7–9. Association for Computational Linguistics (2003)
8. Martí-Oliet, N., Meseguer, J.: Rewriting logic: roadmap and bibliography. Theoretical Computer Science **285**(2), 121–154 (2002)
9. Miller, G.A.: The magical number seven, plus or minus two: Some limits on our capacity to process information. Psychological Review **63**(2), 81–97 (1956)
10. Norman, D.A.: Cognitive engineering. In: User-Centred System Design: New Perspectives on Human-Computer lnteraction, pp. 31–65. Lawrence Erlbaum Associates (1986)
11. Oberauer, K., Jarrold, C., Farrell, S., Lewandowsky, S.: What limits working memory capacity? Psychological Bulletin **142**(7), 758–799 (2016)
12. Ölveczky, P.C.: Real-time maude and its applications. In: Proc. of WRLA 2014, Lecture Notes in Computer Science, vol. 8663, pp. 42–79. Springer (2001)
13. Ölveczky, P.C.: Designing Reliable Distributed Systems. Undergraduate Topics in Computer Science, Springer (2017)
14. Ölveczky, P.C., Meseguer, J.: Semantics and pragmatics of Real-Time-Maude. Higher-Order and Symbolic Computation **20**(1-2), 161–196 (2007)
15. Storrs, G.: A conceptual model of human-computer interaction? Behav. Inf. Technol. **8**(5), 323–334 (1989)

16. Storrs, G.: A conceptualization of rnultiparty interaction. Interacting with Computers **6**(2), 173–189 (1994)
17. Tung, T., Gomez, R., Kawahara, T., Matsuyama, T.: Multi-party human-machine interaction using a smart multimodal digital signage. In: Proc. of HCII 2013, Lecture Notes in Computer Science, vol. 8007,, pp. 408–415. Springer (2013)
18. Weyers, B., Bowen, J., Dix, A., Palanque, P. (eds.): The Handbook of Formal Methods in Human-Computer Interaction. Human–Computer Interaction Series, Springer (2017)