A Cognitive Framework Based on Rewriting Logic for the Analysis of Interactive Systems

Antonio Cerone(✉)

IMT School for Advanced Studies, Lucca, Italy
antonio.cerone@imtlucca.it
http://sysma.imtlucca.it/people/antonio-cerone/

Abstract. Interactive systems may appear to work correctly and safely when analysed in isolation from the human environment in which they are supposed to work. In fact, the same cognitive skills that enable humans to perform complex tasks may also become the source of critical errors in the interaction with systems and devices designed as supports for such tasks. It is thus essential to verify the desired properties of an interactive system using a model that not only includes a user-centered description of the task, but also incorporates a representation of human cognitive processes within the task execution.

In this paper we consider automatic and deliberate cognitive processes in combination with the use of the Short Term Memory (STM), and provide a formal notation to model the set of basic tasks that a human component (user or operator) has to carry out to accomplish a goal by interacting with an interface. The semantics of the notation is given in terms of a cognitive framework that makes use of rules driven by the basic tasks to rewrite both the system state and the STM until all necessary tasks have been completed. Potential human errors are then detected using model checking. Our notation, which is implemented using the MAUDE rewrite system, and our formal verification methodology are finally illustrated by two case studies: a user of an Automatic Teller Machine (ATM) and an operator of an Air Traffic Control (ATC) system.

Keywords: Formal modelling and verification · Rewriting logic · Interactive systems · Model checking · MAUDE

1 Introduction

Interactive systems are characterised by a cooperative work between a *human component* and the *interface* of a system, which can be a computer system, a device, a control system, a transportation system, etc. The purpose of the cooperation is the accomplishment of a goal, which may be a specific objective to achieve, such as purchasing a product from a vending machine, or a correct state of the system to be preserved. In the former situation the human component is the *user* of the system underlying the interface, in the latter the *operator*, through the interface, of a plant control (e.g. a nuclear plant) or a control service (e.g. traffic control).

The systematic analysis of human errors in interactive systems has its roots in Human Reliability Assessment (HRA) techniques [12], which mostly emerged in the 1980's. However, these first attempts in the safety assessment of interactive systems were typically based on *ad hoc* techniques [13], with no efforts to incorporate a representation of human cognitive processes within the model of the interaction. Although Mach already stated at the beginning of last century that "knowledge and error flow from the same mental sources, only success can tell the one from the other" [15], we had to wait until the 1990's to clearly understand that "correct performance and systematic errors are two sides of the same coin" [21]. At that time the increasing use of formal methods yielded more objective analysis techniques [9] that resulted, on the one hand, in the notion of *cognitively plausible user behaviour*, based on formal assumptions to bound the way users act driven by cognitive processes [1] and, on the other hand, in the formal description of expected effective operator behaviour [20] and the formal analysis of errors performed by the operator as reported by accident analysis [11]. Thus, research in the formal analysis of interactive systems branched into two separate directions: the analysis of cognitive errors of users involved in everyday-life [2] and work-related [18,22] interactive tasks, and the analysis of skilled operators behaviour in traditionally critical domains, such as transportation, chemical and nuclear plants, health and defence [3,4,7,17,23]. The different interaction contexts of a user, who applies attention very selectively and acts mainly under automatic control [2,19], and an operator, who deals with high cognitive load and whose attentional mechanisms risk to be overloaded due to coping with Stimulus Rich Reactive Interfaces (SRRIs) [23], have led to the development of distinct approaches, keeping separate these two research directions. However, users have sometimes to deal with decision points or unexpected situations, which require a "reactivation" of their attentional mechanisms, and operators must sometime resort to automatisms to reduce attentional and cognitive loads.

In this paper, we try to unify these two research directions by providing a general framework to reconcile automatic control with attentional and cognitive loads. Section 2 adopts the information processing approach in explaining human behaviour and defines a framework, together with a formal notation, to describe the cognitive processes underlying human behaviour and the way they exploit human memory. Section 3 provides the semantics of our notation in terms of a rewriting system model (Sect. 3.1) and briefly presents its implementation and use (Sect. 3.2). Section 4 illustrates the generality of our cognitive framework on two case studies: a user of an Automatic Teller Machine (ATM), and an operator of an Air Traffic Control (ATC) system. Both case studies had been previously modelled [2–4] using the CSP (Communicating Sequential Processes) process algebra [10]. While in such previous work two distinct *ad hoc* frameworks had been developed to model a user [2] and an operator [3,4], in this paper we unify the two contexts within the same formal framework, which is based on rewriting logic [16] and is implemented within the MAUDE rewrite system [5].

2 Modelling Cognitive Processes

Following the *information processing* approach normally used in cognitive psychology, we model human cognitive processes as processing activities that make use of input-output channels, to interact with the external environment, and three main kinds of memory, to store information: *sensory memory*, where information perceived through the senses persists for a very short time; *short-term memory (STM)*, which has a limited capacity and where the information that is needed for processing activities is temporary stored with rapid access and rapid decay; *long-term memory (LTM)*, which has a virtually unlimited capacity and where information is organised in structured ways, with slow access but little or no decay [8].

2.1 Input as Perceptions and Output as Actions

Input and output occur in humans through senses. In our work we give a general representation of input channels in term of *perceptions*, with little or no details about the specific senses involved in the perception, but with a strong emphasis on the semantics of the perception in terms of its potential cognitive effects. For instance, if the user of a vending machine perceives that the requested product has been delivered, the emphasis is on the fact that the user will be induced to collect the product and not on whether the user has seen or rather heard the product coming out of the machine.

We represent output channels in term of *actions*. Actions are performed in response to perceptions. We are interested in the urgency to react created by the perception: for example, if we are withdrawing cash, we need to collect the delivered cash before the machine takes it back. Analogously, if an operator perceives an anomalous system behaviour, in general we are not interested in whether such perception occurs through sight, hearing, or even by touching a hot component or through a burning smell; instead, we are interested in the action that the operator has to carry out to solve the problem and in the urgency of such an action.

2.2 Attention and Processing Control

Perceptions are briefly stored in the sensory memory and only relevant perceptions are transfered to the STM using *attention*, a selective processing activity that aims to focus on one aspect of the environment while ignoring others. Inspired by Norman and Shallice [19], we consider two levels of cognitive control:

automatic control fast processing activity that does not require attention to occur and is carried out outside awareness with no conscious effort;

deliberate control processing activity triggered and focussed by attention and carried out under the intentional control of the individual, who is aware and conscious of the required effort.

For example, automatic control is essential in driving a car and, in such a context, it develops throughout a learning process based on deliberate control: during the learning process the driver has to make a conscious effort to use gear, indicators, etc. in the right way (deliberate control) and would not be able to do this while talking or listening to the radio. Once automaticity in driving is acquired, the driver is aware of the high-level tasks that are carried out, such as driving to office, turning to the right and waiting at a traffic light, but is not aware of low-level details such as changing gear, using the indicator and the colour of the light, amber or red, while stopping at a traffic light (automatic control).

2.3 Tasks and Short-Term Memory (STM)

The purpose of an interaction between a human and an interface is to allow the human to accomplish a goal. In Sect. 2.2 we have referred to high-level and low-level tasks. The goal is associated with the top-level task. For both users and operators the top-level task can be decomposed in a hierarchy of tasks until reaching basic tasks, which cannot be further decomposed. A difference between the user and operator cases is that the user's goal is normally associated with the basic task that accomplishes it, whereas there is no such basic task in the operator case.

We model a basic task as a quadruple

$$\mathit{info}_i \uparrow \mathit{perc}_h \Longrightarrow \mathit{act}_h \downarrow \mathit{info}_j$$

where perception perc_h triggers the retrieval of information info_i from the STM, the execution of action act_h and the storage of information info_j in the STM.

Information is kept promptly available, while it is needed to perform the current top-level task, by storing it in the STM. Several kinds of information may be stored in the STM: the goal of the interaction (which we identify with the top-level task), a partial result of a calculation, a piece of information retrieved from the LTM, a perception transferred from the sensory memory through the attention mechanism, a reference to a future action to be performed, the current state of the ongoing reasoning process or plan currently carried out. For the purpose of our work we consider only three kinds of information that can be stored in the STM:

task goal represented as the action that leads to the direct achievement of the goal (user), or as the action that contributes to preserve the correct system state or a placeholder if such an action cannot be identified (operator);
action reference which refers to a future action to be performed;
cognitive state that is the state of the plan developed by the user/operator.

A task goal is formally modelled as

$$goal(act, type)$$

where *act* is the action that either leads to the direct achievement of the goal, if $type = achieve$, or contributes to preserve the correct system state, if $type = preserve$ (in this case the action may be left unspecified).

We formally denote by *none* when an entity (information, perception, action) of a task or the action of a task goal is absent or left unspecified (*none* is a placeholder in the latter case). When *none* is used as information it denotes the absence of action reference.

We model the two levels of control considered in Sect. 2.2 as three categories of basic tasks:

automatic task triggered by a perception, or an information in the STM;
cognitive task triggered by a cognitive state;
decision task triggered by a task goal in the STM.

An automatic tasks must include an action, but may not include a perception or may not use the STM (thus it may have one or both information fields empty). A cognitive task must always have the two information fields to contain the current cognitive state to retrieve from STM and the next cognitive state to store in the STM, but it has neither perception nor action. A decision task must include a perception and store in the STM a reference to an action that is related to the task goal contained in the retrieval information field, with the perception triggering the retrieval of the task goal; for instance, if the first part of the driving route to our workplace is in common with the driving route to our favourite supermarket and our goal is driving to work, the perception of approaching the branching point will trigger the storage of a reference to the action related to the goal (i.e. taking the road to drive to work) in the STM.

Automatic tasks are performed under automatic control, whereas cognitive and decision tasks are performed under deliberate control. Normally, a user works mainly under automatic control [19], with most of the performed tasks being automatic tasks, whereas an operator works mainly under deliberate control [21], with most of the performed tasks being cognitive tasks.

2.4 Interface

In our context a user perception refers to a stimulus produced by an action of the interface with which the human is interacting. Hence we identify an interface state created by an interface action with the perception such an action produces in humans. For example, the interface state created by the action of giving change, performed by the interface of a vending machine, is identified with the perception (sound of falling coins or sight of the coins) produced. Thus, in our notation, interface state and corresponding human perception are denoted by the same formal entity (which, assuming the user's perspective, we call "perception").

In Sect. 2.1 we anticipated that perceptions may induce different degrees of urgency in reacting. Since we identify a perception with the interface state caused by the interface action that produced that perception, the urgency of a perception can be modelled by associating a timeout with such an interface state (hence with the perception itself). For example the urgency of the user of a cash machine in collecting the delivered cash is associated with the machine timeout for taking back the cash. In order to use perceptions as interface states, possibly with timeouts, to define interface transitions, we decorate a perception *perc* as follows.

$perc!0$ state that produces a perception inducing no urgency in reacting and is not associated with a timeout;

$perc!1$ state that produces a perception inducing urgency and is associated with a timeout that is not expired;

$perc!2$ state that produces a perception inducing urgency and is associated with a timeout that has already expired.

By interpreting perceptions in terms of the interface states that caused them, we model an interface transition as a triple

$$perc_h!m \xrightarrow{act_h} perc_k!n$$

where interface state $perc_h$, with possible timeout characterised by m, triggers the execution of action act_h with a transition of the interface to state $perc_k$, whose possible timeout is characterised by n. An action $act \neq none$ is, therefore, performed through a cooperation between human and interface and thus belongs to both a task and an interface transition and represents the basic form of interaction. An action $act = none$ is denoted by an unlabelled arrow. The initial state of the interface is normally an idling state (the interface is available for an interaction), thus it is not associated with a timeout ($perc!0$). In our formal representation we keep track of the action *act* that produced the state *perc* by defining an interface state as a pair $act \gg perc!m$. The initial state becomes then $none \gg perc!0$. We will exploit this redundant notation in Sect. 3.1.

2.5 Closure and Post-completion Error

An important phenomenon that occurs in automatic behaviour is *closure* [8]. When the goal of a task has been achieved there is a tendency to flush out the STM to be ready to start a new task. This may cause the removal from the STM of some important subtasks that are still not completed and result in some form of failure of the main task, called *post-completion error*. Undesired closure most commonly occurs when the main goal of the task is achieved before completing some subsidiary tasks, due to the task sequentialisation forced by the interface. A classical example is provided by an ATM that delivers cash before returning the card. Since the user's main goal is to get cash, once the cash is collected, the STM is flushed out and the user may terminate the task, thus forgetting the card in the ATM. That is why modern ATMs return the card before delivering cash. Closure has been formally modelled in previous works using Higher Order Logic (HOL) [6] and the CSP process algebras [2].

2.6 Long-Term Memory (LTM) and Supevisory Attentional System

LTM is used for long-term storage of "factual information, experiential knowledge, procedural rules of behaviour — in fact, everything that we know" [8]. Information may be transferred from the STM to the LTM through *rehearsal*, the well-known recycling mechanism functionally equivalent to the idea of repeating things to yourself.

In our cognitive framework, we do not consider transfer of information from STM to LTM. In fact, we assume that the LTM already contains procedural rules of behaviour, such as the basic tasks (automatic, cognitive and decision tasks) introduced in Sect. 2.3. Moreover, during automatic control, experiential knowledge already stored in the LTM may be used to solve situations in which automatic tasks result inappropriate. Norman and Shallice [19] propose the existence of a *Supervisory Attentional System* (SAS), sometimes also called *Supervisory Activating System*, which becomes active whenever none of the automatic tasks are appropriate. The activation of the SAS is triggered by perceptions that are assessed as danger, novelty, requiring decision or the source of strong feelings such as temptation and anger.

We formalise such an assessment as a function $assess(act, perc)$, where $perc$ is the perception that triggered the SAS activation and act is the last interaction before that perception. The function returns one of the following values: $danger$, $decision$, $novelty$, $anger$ and $auto$. For example, if we start overtaking a car (act) and we hear honking from behind ($perc$) the assessment will be $assess(act, perc) = danger$. Normally the automatic response to a danger is to abandon the ongoing task without accomplishing the goal, in our example the overtaking task/goal. Responses to novelties ($novelty$) and feelings (e.g. $anger$) vary from individual to individual and cannot be captured by our framework. Response to requiring decision ($decision$) are driven by a specific basic task of the model. Value $auto$ denotes that the SAS is not activated.

Therefore, the assessment function is a way of formalising experiential knowledge that has been stored in the LTM, in our example the experience that honking is a warning of danger.

3 Rewriting System Model and Analysis

3.1 Rewrite Rules

Let Π be a set of perceptions, Σ be a set of actions, Γ be a set of action references and Δ a set of cognitive states, with $\Gamma \cap \Delta = \emptyset$. We model our cognitive framework on Π, Σ, Γ and Δ as a rewrite system consisting of four sets of objects

$\mathcal{T}$ a set of basic tasks;
$\mathcal{I}$ a set of interface transitions;
$\mathcal{C}$ a singleton containing the current interface state and its causal action;
$\mathcal{M}$ the set of entities in the STM;

and a set $\mathcal{R}$ of rewriting rules

$$\mathcal{T}\ \mathcal{I}\ \mathcal{C}\ \mathcal{M} \overset{\text{rewrite}}{\longrightarrow} \mathcal{T}\ \mathcal{I}\ \mathcal{C}'\mathcal{M}'$$

that are defined as follows:

interacting: if $info_i \uparrow perc_h \Longrightarrow act_h \downarrow info_j \in \mathcal{T}$, with $act_h \neq none$
and $\mathcal{C} = \{act \gg perc_h!m\}$ and $perc_h!m \xrightarrow{act_h} perc_k!n \in \mathcal{I}$, with $m < 2$,
and $info_i \in \mathcal{M}$ and there exists a goal in $\mathcal{M}$
then $\mathcal{C}' = \{act_h \gg perc_k!n\}$
and $\mathcal{M}' = \mathcal{M} -\ \{info_i\} \cup \{info_j\}$

closure: if $info_i \uparrow perc_h \Longrightarrow act_h \downarrow info_j \in \mathcal{T}$, with $act_h \neq none$
and $\mathcal{C} = \{act \gg perc_h!m\}$ and $perc_h!m \xrightarrow{act_h} perc_k!n \in \mathcal{I}$ and $m < 2$
and $goal(act_h, achieve), info_i \in \mathcal{M}$
then $\mathcal{C}' = \{act_h \gg perc_k!n\}$
and $\mathcal{M}' = \{info_j\}$

danger: if $info_i \uparrow perc_h \Longrightarrow act_h \downarrow info_j \in \mathcal{T}$, with $act_h \neq none$
and $\mathcal{C} = \{act \gg perc_h!m\}$ and $perc_h!m \xrightarrow{act_h} perc_k!n \in \mathcal{I}$ and $m < 2$
and $info_i \in \mathcal{M}$
and $assess(act, perc_h) = danger$
then $\mathcal{C}' = \{act_h \gg perc_k!expired(n)\}$ where

$$expired(n) = \begin{cases} 2 \text{ if } n = 1 \\ n \text{ otherwise} \end{cases}$$

and $\mathcal{M}' = \{info_j\}$

timeout: if $\mathcal{C} = \{act \gg perc_h!m\}$ and $perc_h!m \longrightarrow perc_k!n \in \mathcal{I}$ and $m > 1$
then $\mathcal{C}' = \{none \gg perc_k!n\}$
and $\mathcal{M}' = \mathcal{M}$

cognitive: if $info_i \uparrow perc_h \Longrightarrow none \downarrow info_j \in \mathcal{T}$
and $info_i \in \mathcal{M} \cap \Delta$ and $info_j \in \Delta$
then $\mathcal{C}' = \mathcal{C}$
and $\mathcal{M}' = \mathcal{M} - \{info_{\rangle}\} \cup \{info_{|}\}$

decision: if $info_i \uparrow perc_h \Longrightarrow none \downarrow info_j \in \mathcal{T}$
and $info_i \in \mathcal{M}$ is a goal
and $assess(none, perc_h) = decision$
then $\mathcal{C}' = \mathcal{C}$
and $\mathcal{M}' = \mathcal{M} \cup \{info_{|}\}$

Automatic tasks enable the application of rules **interacting**, **closure** and **danger**, which involve an interaction between user and interface ($act_h \neq none$). Cognitive and decision tasks enable the application of rules **cognitive** and **decision**, respectively, which operate on the STM only, without involving any interaction with the interface ($act_h = none$) and with no change to the interface state ($\mathcal{C}' = \mathcal{C}$). Rule **timeout** refers to an autonomous action of the interface, with no involvement of the human component (there is no basic task involved).

The **interacting** rule is applied if there is a perception $perc_h$ in the current state $\mathcal{C}$ and/or information $info_i$ in the STM $\mathcal{M}$ that are associated in a task of $\mathcal{T}$ with the execution of action act_h, there is a goal in the STM $\mathcal{M}$ and there is no expired timeout ($m < 2$) associated with the interface state $perc_h!m$ that has generated perception $perc_h$. The next state $\mathcal{C}'$ of the interface is $perc_k!n$, which results by executing action act_h, and the next STM $\mathcal{M}'$ is obtained by removing information $info_i$ and storing information $info_j$.

The **closure** rule is very similar to the interacting rule, but now the goal in the STM $\mathcal{M}$ must be of type *achievement* ($goal(act_h, achieve)$) and the execution of action act_h results in emptying the STM before storing information $info_j$.

The **danger** rule is applied if the current perception $perc_h$ that follows the execution of action act is assessed as a danger ($assess(act, perc_h) = danger$). The user performs action act_h. Moreover, since, as we have seen in Sect. 2.6, the user's normal response to a danger is to abandon the task, if there is a timeout associated with the current state ($perc_k!1$), then the next state is $perc_k!2$, which is the current state now associated with an expired timeout (since $expired(1) = 2$), otherwise it is $perc_k!n$ (since $expired(n) = n$ for $n \neq 1$). The next STM $\mathcal{M}'$ is obtained by removing all information and storing information $info_j$, as it happens for the closure. The need for this rule to assess a perception with respect to the action that has caused it explains why, in Sect. 2.4, we have kept track, in the formal notation of an interface state, of the action that produced that state.

The **timeout** rule is triggered by the expiration of the timeout ($m > 1$) and leads through the autonomous action act_h to the new interface state $perc_k!n$.

The **cognitive** rule refers to a cognitive process of the human, with cognitive state $info_i$ retrieved from and cognitive state stored in the STM.

Finally, the **decision** rule differs from the cognitive rule because the retrieved information is a goal, which is then stored again in the STM, and because of the presence of the assessment as a precondition. It models the SAS-induced switch from automatic control to deliberate control due to a required decision.

3.2 Implementation and Analysis with MAUDE

The MAUDE implementation, which can be downloaded at

http://sysma.imtlucca.it/cognitive-framework-sefm-2016/,

consists of the following generic modules

entities which defines the basic sorts that model perceptions, actions, and information that can be stored in the STM;

cognitive architecture which defines the structures of tasks, STM, LTM, and interfaces (including the current interface state) and the MAUDE rewrite rules that work on such structures;

and the following modules that are specific to the case study under analysis

tasks which defines the basic tasks and the goals of the case study;
interfaces which includes the different interfaces to be analysed;
LTM information such as the assessment function introduced in Sect. 2.6.

Simulation is performed by running the rewrite commands available in MAUDE.

Model-checking analysis requires the use of the **model-checker** predefined MAUDE module and the definition of two further modules that are specific to the case study:

preds which defines predicates on perceptions, actions and STM information;
check which includes properties to be verified and runs the model checker.

We define the truth value of predicates on an entity e as follows:

$$\mathcal{P}_x(e) = \begin{cases} true & \text{if } x = cogn \text{ and } e \in \mathcal{M} \\ & \text{or } x = act \text{ and there exist } p, m \text{ such that } e \gg p!m \in \mathcal{C} \\ & \text{or } x = perc \text{ and there exist } a, m \text{ such that } a \gg e!m \in \mathcal{C} \\ false & \text{otherwise} \end{cases}$$

The **preds** module implements predicates $\mathcal{P}_{cogn}(e)$, $\mathcal{P}_{act}(e)$ and $\mathcal{P}_{perc}(e)$.

4 Case Studies

In this section our cognitive framework is illustrated through two case studies by effectively using it in two distinct ways: to formally verify properties of interfaces in the context of human usage and compare different interface designs (Sect. 4.1); and to analyse the operator's behaviour by formally checking whether a given decomposition of the operator's task failure is sound and complete (Sect. 4.2). These two case studies were presented in our previous work [2,3] using two independent *ad hoc* approaches, both based on the CSP process algebra.

4.1 Automatic Teller Machine (ATM) User

Let be $\Pi = \{cardR, pinR, cashO, cardO\}$, $\Sigma = \{cardI, pinI, cashC, cardC\}$, $\Gamma = \{cardB\}$ and $\Delta = \emptyset$. A simple ATM task, in which the user has only the goal to withdraw cash, is modelled by the following four basic tasks:

$none \uparrow cardR \Longrightarrow cardI \downarrow cardB$
When the interface is perceived ready ($cardR$), the user inserts the card ($cardI$) and remembers (in the STM) that the card has to be taken back ($cardB$) at a later stage;

$none \uparrow pinR \Longrightarrow pinI \downarrow none$
When the interface is perceived to request a pin ($pinR$), the user inputs the pin ($pinI$);

$none \uparrow cashO \Longrightarrow cashC \downarrow none$
When perceiving that the cash has been delivered ($cashO$), the user collects the cash ($cashC$);

$cardB \uparrow cardO \Longrightarrow cardC \downarrow none$
When perceiving that the card has been returned ($cardO$), the user collects the card ($cardC$) and no longer needs to remember to collect it ($cardB$);

The goal ("to withdraw cash") is identified with the act of collecting cash (action $cashC$) and is formally modelled as $goal(cashC, achieve)$.

We model an old interface that sequentially requests a card, requests a pin, delivers the cash and returns the card, and a new interface that returns the card before delivering the cash. The two interface models are as follows.

Old ATM: transitions	**New ATM: transitions**
1. $cardR!0 \xrightarrow{cardI} pinR!1$	1. $cardR!0 \xrightarrow{cardI} pinR!1$
2. $pinR!1 \xrightarrow{pinI} cashO!1$	2. $pinR!1 \xrightarrow{pinI} cardO!1$
3. $cashO!1 \xrightarrow{cashC} cardO!1$	3. $cardO!1 \xrightarrow{cardC} cashO!1$
4. $cardO!1 \xrightarrow{cardC} cardR!0$	4. $cashO!1 \xrightarrow{cashC} cardR!0$
5. $pinR!2 \longrightarrow cardO!1$	5. $pinR!2 \longrightarrow cardO!1$
6. $cashO!2 \longrightarrow cardO!1$	6. $cashO!2 \longrightarrow cardR!0$
7. $cardO!2 \longrightarrow cardR!0$	7. $cardO!2 \longrightarrow cardR!0$

For both interfaces the initial state is $none \gg cardR!0$.

In both interfaces, transitions 1–4 model the normal sequences of interactions for the specific design (old or new). The last three transitions model interface autonomous actions. In both interfaces, if the timeout expires after requesting a pin, then the card is returned (transitions 5). If the timeout expires after delivering the cash (transitions 6), then in the old ATM the card is returned, whereas in the new ATM the control goes back to the initial state, so inhibiting a cash collection action and, as a result, implicitly modelling that the cash is taken back by the ATM. Finally, in both interfaces, if the timeout expires after returning the card, then the control goes back to the initial state, so inhibiting a card collection action and, as a result, implicitly modelling that the card is confiscated (transitions 7).

We model the user experience for the two ATM designs as follows.

Old ATM: user experience	**New ATM: user experience**
1. $assess(cardI, pinR) = auto$	1. $assess(cardI, pinR) = auto$
2. $assess(pinI, cashO) = auto$	2. $assess(pinI, cardO) = auto$
3. $assess(cashC, cardO) = auto$	3. $assess(cardC, cashO) = auto$
4. $assess(cardC, cardR) = auto$	4. $assess(cashC, cardR) = auto$
5. $assess(pinI, cardO) = danger$	5. $assess(cardC, cardR) = anger$
6. $assess(pinI, cardR) = anger$	6. $assess(act, perc) = novelty$, otherwise
7. $assess(act, perc) = novelty$, otherwise	

In both experiences the value of the assessment is *auto* when the sequence of action and perception is the same as in the experienced interface (assessments 1–4, corresponding to the tasks 1–4 above). A user who has experience with the old ATM design could interpret: the perception that the card is returned after having input the pin as if the pin were incorrect and there were a danger for the card to be confiscated at one of the next attempts (5. $assess(pinI, cardO) = danger$); the perception that the ATM goes back to the initial card request without returning the card after having input the pin as if the card were confiscated (6. $assess(pinI, cardR) = anger$); and any other sequence of action and perception as a novelty. A user who has experience with the new ATM design could interpret: the perception that the ATM goes back to the initial card request after returning the card without delivering cash as a sign that cash cannot be withdrawn, e.g. because the ATM is out of cash (5. $assess(cardC, cardR) = anger$); and any other sequence of action and perception as a novelty.

We want to verify, for each interface design, whether there are cognitive errors that may prevent the user from collecting the card and from collecting the cash. The properties that the user is always able to collect a returned card (property AlwaysCardBack) and is always able to collect the delivered cash (property AlwaysCashGot) are specified as follows:

$$\text{AlwaysCardBack} = \Box(\mathcal{P}_{perc}(cardO) \rightarrow (\neg\mathcal{P}_{perc}(cardR)\ \mathcal{U}\ \mathcal{P}_{act}(cardC)))$$
$$\text{AlwaysCashGot} = \Box(\mathcal{P}_{perc}(cashO) \rightarrow (\neg\mathcal{P}_{perc}(cardR)\ \mathcal{U}\ \mathcal{P}_{act}(cashC)))$$

The model checking analysis shows that AlwaysCardBack is true with the new ATM and not with the old ATM, independently of the user experience, while AlwaysCashGot is false only with the new ATM and a user experienced with the old ATM. Property AlwaysCardBack detects possible post-completion errors in using the old design of the ATM and shows that such errors cannot occur in the new design of the ATM. Property AlwaysCashGot detects the possibility of missing the collection of delivered cash. Although the new design of the ATM works in an ideal world where all ATMs are designed according to the new criterion, there are countries, in the developing world, where ATMs are still designed according to the old criterion. Thus we can imagine that a user from one of such countries, while visiting a country where all ATMs are designed according to the new criterion, is likely to assess the early return of the card as a danger and is prone to abandon the interaction forgetting to collect the cash (falsifying AlwaysCashGot).

4.2 Air Traffic Control (ATC) Operator

The goal of an ATC task is to avoid that the distance between aircraft goes below a minimum prescribed distance. If this happens, we say that the aircraft violate separation. The ATC operator has to monitor the local air traffic situation and execute communication actions to urge aircraft to change speed, altitude and/or direction in order to avoid separation violation. Aircraft whose trajectories are leading to separation violation are called "in conflict".

We consider a purely cognitive task, which models the cognitive processes of the operator after having perceived the state of the system, independently of whether such a perception is correct or erroneous. Thus basic tasks will have no perceptions. Let be $\Pi = \emptyset$ $\Sigma = \{act\}$, $\Gamma = \emptyset$ and $\Delta =$

$\{scan, part, con, non\ decide, reclassify, intend\}$. Following the Operator Choice Model (OCM), defined by Lindsay and Connelly [14], we decompose the ATC task into a number of basic tasks that the operator has to perform:

$scan \uparrow none \Longrightarrow none \downarrow part$
The operator scans the interface (*scan*) until finding a part where there are aircraft that may violate separation (*part*).

$part \uparrow none \Longrightarrow none \downarrow con$
In the part of the interface under analysis (*part*) the operator identifies aircraft that are in conflict (*con*).

$part \uparrow none \Longrightarrow none \downarrow non$
In the part of the interface under analysis (*part*) the operator does not identify aircraft that are in conflict (*non*).

$con \uparrow none \Longrightarrow none \downarrow scan$
If the conflict (*con*) does not require urgent action, the operator goes back to scan the interface (*scan*), looking for more urgent conflicts.

$non \uparrow none \Longrightarrow none \downarrow scan$
If no conflict has been identified (*non*) in the part under analysis, the operator goes back to scan the interface (*scan*).

$con \uparrow none \Longrightarrow none \downarrow decide$
The operator develops a plan (*decide*) to solve the conflict under investigation (*con*).

$con \uparrow none \Longrightarrow none \downarrow reclassify$
The operator reclassifies (*reclassify*) a conflict (*con*) under investigation as a non conflict.

$reclassify \uparrow none \Longrightarrow none \downarrow scan$
After reclassifying (*reclassify*), the operator goes back to scan the interface (*scan*).

$decide \uparrow none \Longrightarrow none \downarrow scan$
After developing a plan to solve a conflict (*decide*), the operator goes back to scan the interface (*scan*), looking for other conflicts.

$decide \uparrow none \Longrightarrow none \downarrow intend$
After developing a plan to solve a conflict (*decide*), the operator intends to carry out a specific action to solve the conflict (*intend*).

$intend \uparrow none \Longrightarrow act \downarrow scan$
The operator implements the intention (*intend*) by performing an action (*act*) and then goes back to scan the interface (*scan*).

The goal ("to prevent separation violation") is expressed simply as the preservation of the state by performing actions (*act* models a generic action) and is formally modelled as $goal(act, preserve)$. Note that only the last basic task is an automatic task; all other tasks are cognitive tasks. Once the intention is established, the cognitive process terminates and the execution of the action is a purely automatic activity triggered by the intention.

In this case study we focus on the cognitive aspects of the ATC operator rather than on the specific aspects of the interface that may induce the operator's errors. Many cognitive errors may occur in the execution of the tasks above independently of the characteristics of the interface that presents the air traffic situation to the operator. Our aim is the analysis of a set of task failures that have been identified by psychologists through the observation of operators while using an ATC simulator [3,4,14], in order to find out if such a set is a sound and complete decomposition of the top-level ATC task failure, that is the occurrence of separation violation. As in our previous work [3,4], we use temporal logic to formalise the task failures and model checking to verify the soundness and completeness of the decomposition. However, in that previous work, the OCM and a "toy environment" consisting of three aircrafts were formalised using the CSP process algebra, and the results could not be generalised to any environment. Here, instead, we use our cognitive framework to formalise the OCM as a list of basic tasks, as shown above, and we model the environment as the following trivial interface consisting of just one transition, which is independent of the number of aircrafts involved and the number of conflicts between them.

1. $\mathit{none}!0 \xrightarrow{act} \mathit{none}!0$

The initial state is obviously $\mathit{none} \gg \mathit{none}!0$. In this case study there is no information on previous experience. Hence, there is no need to introduce an assessment function.

We can characterise a separation violation as an operator who persistently misses the intention to carry out a specific action to solve the conflict [3,4]. Hence the top-level task failure is formalised as $\Box\neg\mathcal{P}_{cogn}(\mathit{intend})$. We distinguish between intention (intend) and action (act) to be able to model an unintended action that does not match the intention [21]. Although this is not part of our analysis, such a mismatch would be relevant in the analysis of errors induced by a specific interface design, which could be carried out on this case study by introducing alternative interface designs and using our formal cognitive framework as in the ATM case study.

The formalisation of the ATC task failure decomposition suggested by Lindsay and Connelly [14] is

$$\mathcal{D} = \{\text{FailureOfScanning}, \text{PerMisClass}, \text{PerMisPrior}, \text{DeferAction}\}$$

where

$$\begin{aligned}
&\text{FailureOfScanning} = \Box\neg\mathcal{P}_{cogn}(\mathit{part})\\
&\text{PerMisClass} = \Diamond\mathcal{P}_{cogn}(\mathit{part}) \wedge \Box(\mathcal{P}_{cogn}(\mathit{part}) \vee \mathcal{P}_{cogn}(\mathit{con}) \rightarrow \bigcirc\mathcal{P}_{cogn}(\mathit{non}))\\
&\text{PerMisPrior} = \Diamond\mathcal{P}_{cogn}(\mathit{con}) \wedge \Box(\mathcal{P}_{cogn}(\mathit{con}) \rightarrow \bigcirc\mathcal{P}_{cogn}(\mathit{scan}))\\
&\text{DeferAction} = \Diamond\mathcal{P}_{cogn}(\mathit{decide})) \wedge \Box(\mathcal{P}_{cogn}(\mathit{decide}) \rightarrow \bigcirc\mathcal{P}_{cogn}(\mathit{scan}))
\end{aligned}$$

Failure of scanning (FailureOfScanning) occurs when the operator fails to monitor a specific part of the interface, thus missing possible conflicts. *Persistent mis-classification* (PerMisClass) occurs when the operator persistently classifies as a non conflict what is actually a conflict. *Persistent mis-prioritisation* (PerMisPrior) occurs when the operator persistently gives a low priority to a conflict, thus missing to solve it. *Defer action for too long* (DeferAction) occurs when the operator persistently delays to implement an already developed plan to solve a conflict. Note that the "eventually" part in the last three formulae guarantees that the task failures are not overlapping.

The soundness of the decomposition is expressed by model-checking formula

$$\bigwedge_{F \in \mathcal{D}} (F \rightarrow \Box \neg \mathcal{P}_{cogn}(intend))$$

The completeness of the decomposition is expressed by model-checking formula

$$(\Box \neg \mathcal{P}_{cogn}(intend)) \rightarrow \bigvee_{F \in \mathcal{D}} F$$

Model checking analysis using MAUDE shows that decomposition $\mathcal{D}$ is sound but not complete. However, if we redefine PersMisClass as

$$\text{PersMisClass'} = \Diamond \mathcal{P}_{cogn}(part)) \wedge \Box(\mathcal{P}_{cogn}(part) \rightarrow \bigcirc \mathcal{P}_{cogn}(non))$$

and we define

$$\begin{aligned}\text{ConDecPro} = {} & \Diamond \mathcal{P}_{cogn}(reclassify) \wedge \\ & \Box(\mathcal{P}_{cogn}(con) \rightarrow \bigcirc(\mathcal{P}_{cogn}(scan) \vee \mathcal{P}_{cogn}(reclassify))\end{aligned}$$

then model checking analysis using MAUDE shows that decomposition

$$\mathcal{D}' = \{\text{FailureOfScanning}, \text{PerMisClass'}, \text{PerMisPrior}, \text{ConDecPro}, \text{DeferAction}\}$$

is sound and complete.

Contrary decision process (ConDecPro) new task failure occurs when a conflict is persistently reclassified as a non conflict. Details on the psychological interpretation of all task failures can be found in our previous work [3].

5 Conclusion

We have presented a cognitive framework for the formal analysis of the interaction between humans and interfaces both in the case of a user, who acts mainly under automatic control using selective attention, and in the case of an operator, who deals with high cognitive and attentional load. The ATC case study presented in Sect. 4.2 illustrates how cognitive processes carried out under deliberate control result in automatic activities performed under automatic control.

This is a major generalisation with respect to our two previous CSP works. In fact, in one work [2] the user model captured automatic control with attentional mechanisms limited to automatic responses to unexpected events but not sensitive to decisional clues that trigger responses carried out under deliberate control (decisions); in the other work [3,4] the operator model expressed deliberate control with no capability to formalise perceptions, which are instead fundamental in the SRRIs used in plant and traffic control, and was *ad hoc* for an ATC task in a fixed "toy environment".

References

1. Butterworth, R., Blandford, A.E., Duke, D.: Demonstrating the cognitive plausability of interactive systems. Formal Aspects Comput. **12**, 237–259 (2000)
2. Cerone, A.: Closure and attention activation in human automatic behaviour: a framework for the formal analysis of interactive systems. In Proceedings of FMIS 2011. Electronic Communications of the EASST, vol. 45 (2011)
3. Cerone, A., Connelly, S., Lindsay, P.: Formal analysis of human operator behavioural patterns in interactive surveillance systems. Softw. Syst. Model. **7**(3), 273–286 (2008)
4. Cerone, A., Lindsay, P., Connelly, S.: Formal analysis of human-computer interaction using model-checking. In: Proceedings of SEFM 2005, pp. 352–361. IEEE (2005)
5. Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Talcott, C.: The maude 2.0 system. In: Nieuwenhuis, R. (ed.) RTA 2003. LNCS, vol. 2706, pp. 76–87. Springer, Heidelberg (2003)
6. Curzon, P., Blandford, A.: Formally justifying user-centred design rules: a case study on post-completion errors. In: Boiten, E.A., Derrick, J., Smith, G.P. (eds.) IFM 2004. LNCS, vol. 2999, pp. 461–480. Springer, Heidelberg (2004)
7. De Oliveira, R.A.: Formal specification and verification of interactive systems with plasticity : applications to nuclear-plant supervision. Ph.D. thesis, University of Grenoble (2015)
8. Dix, A., Finlay, J., Abowd, G., Beale, R.: Human-Computer Interaction. Pearson Education, Englewood Cliffs (1998)
9. Dix, A.J.: Formal Methods for Interactive Systems. Academic Press, Cambridge (1991)
10. Hoare, C.: Communicating Sequential Processes. International Series in Computer Science. Prentice Hall, Upper Saddle River (1985)
11. Johnson, C.: Reasoning about human error and system failure for accident analysis. In: Howard, S., Hammond, J., Lindgaard, G. (eds.) INTERACT 1997. IFIP, pp. 331–338. Chapman and Hall, London (1997)
12. Kirwan, B.: Human reliability assessment (chap. 28). In: Evaluation of Human Work. Taylor and Francis, London (1990)
13. Leveson, N.G.: Safeware: System Safety and Computers. Addison-Wesley, Boston (1995)
14. Lindsay, P., Connelly, S.: Modelling erroneous operator behaviours for an air-traffic control task. In: Proceedings of AUIC 2002. Conferences in Research and Practice in Information Technology, vol. 7, pp. 43–54. Australian Computer Society (2002)
15. Mach, C.: Knowledge and Error. Reidel (1905). English Translation (1976)
16. Martí-Oliet, N., Meseguer, J.: Rewriting logic: roadmap and bibliography. Theoret. Comput. Sci. **285**(2), 121–154 (2002)
17. Martinie, C., Palanque, P., Fahssi, R., Blanquart, J.P., Fayollas, C., Seguin, C.: Task model-based systematic analysis of both system failures and human errors. IEEE Trans. Human-Mach. Syst. **46**(2), 243–254 (2016)
18. Masci, P., Rukšėnas, R., Oladimeji, P., Cauchi, A., Gimblett, A., Li, Y., Curzon, P., Thimbleby, H.: The benefits of formalising design guidelines: a case study on the predictability of drug infusion pumps. Innovations Syst. Softw. Eng. **11**(2), 73–93 (2015)
19. Norman, D.A., Shallice, T.: Attention to action: willed and automatic control of behaviour. In: Consciousness and Self-Regulation. Advances in Research and Theory, vol. 4. Plenum Press (1986)

20. Palanque, P., Bastide, R., Paterno, F.: Formal specification as a tool for objective assessment of safety-critical interactive systems. In: Howard, S., Hammond, J., Lindgaard, G. (eds.) INTERACT 1997. IFIP, pp. 323–330. Chapman and Hall, London (1997)
21. Reason, J.: Human Error. Cambridge University Press, Cambridge (1990)
22. Rukšėnas, R., Curzon, P., Blandford, A.E., Back, J.: Combining human error verification and timing analysis: a case study on an infusion pump. Formal Aspects Comput. **26**, 1033–1076 (2014)
23. Su, L., Bowman, H., Barnard, P., Wyble, B.: Process algebraic model of attentional capture and human electrophysiology in interactive systems. Formal Aspects Comput. **21**(6), 513–539 (2009)