Modeling and Analysis of Human Memory Load in Multitasking Scenarios

Late-Breaking Results

Giovanna Broccia
University of Pisa
Italy
giovanna.broccia@di.unipi.it

Paolo Masci
INESC TEC & University of Minho
Portugal
paolo.masci@inesctec.pt

Paolo Milazzo
University of Pisa
Italy
milazzo@di.unipi.it

ABSTRACT

This paper presents on-going work developing a formal framework for the model-based analysis of human-machine interaction in multiple critical systems. The framework builds on classical results from applied psychology on selective attention and working memory. The framework is intended for developers of interactive critical systems to identify plausible human multitasking strategies that are likely to be adopted by operators when using multiple interactive systems at the same time, and to estimate the memory load necessary to complete concurrent tasks. This type of analysis is especially useful at the early stages of system design, to better understand the effort necessary to operate the system when an implementation or a prototype of the system is unavailable. The analysis can also be used retrospectively, to analyse already implemented systems and complement results from user studies. An example based on infusion pumps, used in chemotherapy to infuse doses over a period, is employed to demonstrate the utility of the framework. The framework makes it possible to model the interactive tasks necessary to configure the pumps and start the infusion. The results of the analysis indicate situations where the operator is unable to carry out the task because of omission errors. These results are in line with experimental results reported in the literature, and may provide more detailed hypotheses that can be validated experimentally.

EICS'18, June 19–22, 2018, Paris, France

ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.1145/3220134.3220140

CCS CONCEPTS

• **Human-centered computing** → **User models**; **User interface design**; • **Software and its engineering** → **Formal methods**;

KEYWORDS

Cognitive model; Multi-tasking; Formal Analysis

ACM Reference Format:
Giovanna Broccia, Paolo Masci, and Paolo Milazzo. 2018. Modeling and Analysis of Human Memory Load in Multitasking Scenarios: Late-Breaking Results. In *EICS '18: ACM SIGCHI Symposium on Engineering Interactive Computing Systems, June 19–22, 2018, Paris, France.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3220134.3220140

1 INTRODUCTION

Working memory (WM) has been recognised by psychologists to explain aspects of the human reasoning and decision-making processes. WM has limited capacity and plays a key role in immediate storage and manipulation of information [4]. Several theories relating to the characteristics of WM have been proposed. One of the most recent and popular theories is the *Time-Based Resource Sharing Model* [6]. This theory builds on the following cognitive hypotheses:

- Items stored in WM are subject to processing and maintenance activities, where the term maintenance refers to the process necessary for preserving memory items, and typically involves rehearsal activities [4];
- Processing and maintenance activities both use the same cognitive resource, namely "attention";
- If attention is drawn away from maintenance activities, the activation of items in WM suffers from time-related decay;
- Processing activities that require retrieval of items in WM have a strong detrimental effect on maintenance activities;
- Simultaneous retrieval of multiple items in WM cannot be done.

Building on these hypotheses, cognitive psychologists have defined an indicator, *Cognitive Load* (CL), for measuring the temporal density of attentional demands in user tasks.

Specifically, CL provides a measure of the total amount of time during which maintenance of items in WM is impeded. When activities on items in WM are performed at a constant pace, CL is equal to $\sum(a_i \times n_i)/T$, where n_i is the number of retrievals of type i, a_i is the difficulty of retrievals of type i, and T is the total duration of the task.

Contribution. This paper presents on-going work developing a novel framework for the formal analysis of human-machine interaction in multitasking scenarios. The aim here is to demonstrate the benefits of using the framework as a complement to experimental user studies. Model-based analysis is quicker and more systematic, and can therefore help designers to explore cognitive hypotheses that could explain the outcome of user tasks carried out using multiple devices at the same time. This is important to help developers identify in advance possible design flaws in interactive systems. The specific example considered in this work is a scenario in which a clinical operator interacts with multiple infusion pumps at the same time. The analysis identifies situations of memory overload that could induce omission errors, and provides a plausible explanation of the cognitive causes of the omission errors that are not captured by other model-based analysis frameworks.

2 OUR COGNITIVE FRAMEWORK

Our cognitive framework attempts to integrate the Time-Based Resource Sharing Model with experimental results on human performance in multitasking scenarios. These studies have shown that attention has a key role, not only in the execution of a single task, but also in multitasking. In these studies (e.g., see [13]), it has been shown that human performance with a "main" task decreases when the CL of a "distractor" task increases. This result can be explained intuitively as a redirection of attentional resources from the main task to the distractor task.

To enable fine-grained analysis of multitasking strategies within the framework, a task is decomposed into a sequence of subtasks, and each subtask into a sequence of basic (atomic) tasks. Hence, integration of the experimental results in the cognitive model is carried out by redefining the formula for estimating CL of a task as follows:

$$CL = \frac{\sum(d_i \times t_i)}{\sum(t_i + \delta_i)} \quad (1)$$

where for each basic task of the current subtask:

- d_i is the difficulty of basic task i;
- t_i is the duration of basic task i;
- δ_i is an additional delay representing the time lapse the user may need to wait before performing the basic task (e.g., because they are waiting feedback from the device).

The CL associated with a task is recomputed every time a new subtask becomes active. To capture additional cognitive hypotheses about plausible multitasking strategies adopted by a user, a *rank* is assigned to each task. The rank measures the likelihood that the task will attract the user's attention and will therefore be executed by the user.

3 FORMAL MODEL

The formal specification of the proposed cognitive framework is described in Real Time Maude. There is only space here for a high-level description of the model. A more in-depth presentation of the specification is described in [8]. The full specification can be downloaded from GitHub[1].

The cognitive model is specified as a set of objects, including: a *Working Memory* object representing the user's WM; one or multiple *Interface* objects representing the interfaces of the devices with which the user would interact; and a *Task* object for each Interface object representing the task the user would perform with that device. The behaviour of the system is specified through a set of rewriting rules over objects. In the following, the main elements of the model are described.

Working Memory. The WM is modelled as a finite list of items. Three types of items are considered: *basic information* (e.g., a cognitive item acquired through a procedural step in the current task), *cognition* (i.e., a mental plan resulting from the process of acquiring knowledge and understanding), and *goals*. Goals are expressed in terms of actions, i.e., `g(`*act*`)` indicates that goal `g` is achieved when action *act* is performed.

Interfaces. The behaviour of the user interface of a device is modelled as a transition system. An interface transition has the form `p1 -- `*act*` --> p2`, indicating that the interface state changes from `p1` to `p2` when the user performs an action *act*. The state of the user interface describes what the user perceives. For instance, if the user is interacting with an infusion pump, and information on the device front panel indicates that infusion rate needs to be entered, then the interface state capturing this is represented as a symbolic constant `rateToSet`. Some interface states may be subject to a timeout, capturing the fact that the effect may be temporary. For example, an infusion pump may allow entry of infusion rate only for a timeframe t. This is modelled using the construct `rateToSet for time `t, indicating that `rateToSet` has property `expired` after time `t`.

Tasks. A task is modelled as a sequence of subtasks, where each subtask is a sequence of basic (atomic) tasks. Task decomposition is carried out using standard task modelling techniques based on hierarchical decomposition: starting from the high-level goal, a sequence of actions is identified that explain how to achieve the goal. Each action is decomposed into a sequence of simpler actions until possible.

[1] http://github.com/brocciagi/cognitive_framework

Basic tasks represent basic actions which can be no longer decomposed. In our framework, they are modelled as function with two parameters, `inf1` and `perc`, representing a piece of information in WM and the perceived interface state necessary that activates the execution of the basic task. Two kinds of basic tasks can be defined in our framework: a *user action* to be performed on the device; a *cognition* carried out by the user without involving the device. In the first case, the basic task specifies the action to be performed on the corresponding interface and the information item used to update the WM. In the second, the basic task specifies only an information item that will be used to update the WM.

Each basic task is characterised by a delay, a cognitive load and a criticality level. When the delay is greater than zero, the basic task is not enabled. The model keeps track of these indicators for each basic task. It also keeps track of the elapsed time since the task was last executed. These factors are used when choosing the next task to be performed (explained below).

Choice of Tasks & Ranking Function. The next basic task to be selected is chosen from the set of first basic tasks of each enabled task. The rank associated to each task is described as the product of three factors: the cognitive load of the current subtask; the criticality level of the task; and the total amount of time the task has not been executed. The rank reflects the observation that the nature of the task usually has an influence on the user's attention — a person is inclined to focus more frequently on critical tasks, and tends to give attention to a task if the task has not been performed for some time. In our framework, the task with highest rank is the one selected for execution. A similar approach has been used by Houser et al. in [18] to estimate workload of air traffic controllers in air traffic conflict resolution tasks (further details are discussed in the related work section).

The specification of the ranking function is shown in Listing 1. The function's behaviour is as follows: the task can be executed when the task is not delayed (line 7 in Listing 1), and there is a goal in memory for the task (line 6). In these cases the rank of the task is computed as the product of its criticality level PR2, cognitive load CL and amount of time T since the task was last executed (line 7). Otherwise the rank is defined as 0, and the task is not executed (lines 8–9).

```
1 eq rank(< I : Interface | task :
2  < TASK : Task | subtasks : ((INF1 | P1 ==> DACT | INF2
3                  duration NZT difficulty PR delay T2)
                                BTL) :: OTHER-SUB-TASKS,
4                  waitTime : T, cognitiveLoad : CL,
5                  criticalityLevel : PR2 > >,
6     (I |-> goal(ACT) INF-SET) ; MEMORY) =
7   if T2 == 0 then PR2 * CL * (T + 1)
8   else 0 fi .
9 eq rank(< I : Interface | >, MEMORY) = 0 [owise] .
```

Listing 1: Ranking function

A function *bestRank* is used to calculate the basic task with highest rank (see Listing 2). The function uses the *max* function defined in Maude (line 2 in Listing 2) .

```
1 eq bestRank(NEC1 NEC2, MEMORY) =
2       max(rank(NEC1, MEMORY), rank(NEC2, MEMORY)) .
```

Listing 2: bestRank function

This is of course a simplified interpretation of human cognitive strategies. Similar approaches have however proved useful in the avionics domain for estimating the actual memory load of operators in air traffic conflict resolution tasks [18].

Rewrite Rules. The rewrite rules specified in our framework are generic in the sense that they need to be instantiated using actual information about the task being modelled. They determine how attention is addressed in different types of actions and how this affects the WM. Six rewrite rules are defined:

- *Cognitive Rule:* models either a change of the user's cognition (i.e., mental state) that has not been caused by interactions with a device, or the acquisition of new knowledge.
- *Interacting Rule:* models an interactive task with a device. It is triggered by a perceived user interface state, and may add information items to WM.
- *Forgetting Rule:* this rule deletes items in WM when the WM is full and new information items need to be added. Items that are not associated with the current task are deleted first. This rule captures the observation that a user tends to forget information that is not rehearsed [4].
- *Timeout Rule:* models situations where the timeout associated to a device state expires.
- *Closure Rule:* models the achievement of a goal.
- *All Idling Rule:* this rule rule is applied when all tasks are scheduled to begin in the future — time is fast forwarded till a task can be executed.

4 CASE STUDY

This section illustrates the application of the framework to a case study based on an experiment described in [3], where users were asked to interact with two medical devices. The aim of the experiment was to study multitasking strategies adopted by clinicians, to assess whether particular strategies could induce omission errors, for example forgetting to perform a procedural step required to complete the task.

The original experiment involved the use of two simulated infusion pumps (see Figure 1). Infusion pumps are medical devices routinely used in hospitals to inject fluids (e.g., drugs or nutrients) in the bloodstream of a patient in precise amount and at controlled rates. The devices under consideration provide a front panel with a display and a number of buttons used by clinicians to configure, operate, and monitor the

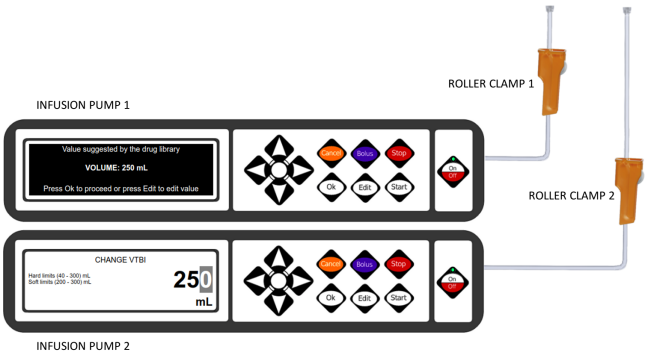

Figure 1: Example scenario with two infusion pumps.

Time	*Prescription Chart*	*Pump 1*	*Pump 2*
1	read vtbi1		
2	read vtbi2		
3		enter vtbi1	
4			enter vtbi2
5	read time1		
6		enter time1	
7		open clamp1	
8	read time2		
9			enter time2
10			open clamp2
11		start infusion1	
12			start infusion2

Table 1: A possible multitasking strategy for setting up two infusion pumps.

pump. To set up an infusion pump, clinicians are usually required to perform five main steps:

- *Step 1.* Read infusion parameters, typically volume to be infused (vtbi) and infusion duration or infusion rate, from a prescription chart;
- *Step 2.* Enter the infusion parameters using the data entry system provided by the pump;
- *Step 3.* Connect the pump to the patient using a "giving set" (a transparent plastic tube with a needle at one end, and a bag with fluid at the other end);
- *Step 4.* Open the roller clamp to allow the fluid to circulate;
- *Step 5.* Start the infusion.

Intensive care patients may be connected to more than one infusion pump at the same time. When multiple infusion pumps need to be configured, clinicians may choose to interleave the steps necessary for setting up the two pumps. This is usually done to optimise cognitive resources (e.g., memory load), or time (e.g., to perform operations on one pump while waiting that the other pump complete an operation) [3].

Different multitasking strategies may produce different memory loads. An example multitasking strategy for setting up the two pumps is shown in Table 1. The question we consider is *"What is the memory load necessary to complete a given task successfully using a particular multitasking strategy?"* An answer to this question could help manufacturers design devices that are simpler to use. It could also be used by hospitals, to develop better training material for clinicians. Academic researchers would also gain benefits, e.g., to test cognitive hypotheses before running an experimental study. In the rest of this paper, the framework is used to address this last question.

5 MODELLING OF MULTITASKING STRATEGIES

The approach offered by the framework is first to model the concurrent interaction with two infusion pumps. This model is used to estimate expected memory load (in Section *Analysis*) and to explore possible system redesigns that could reduce the expected memory load (in Section *Redesign*).

The model includes interfaces representing each pump. The concurrent tasks relate to the procedure for setting vtbi and time values for the two pumps. To set the values, clinicians must read and memorise the values provided by the prescription chart, and then use the pumps' data entry system to enter the values.

The specification of the task for setting up *Pump 1* is shown in Listing 3 (the task for setting up *Pump 2* is specified the same way). Each line in the specification represents a basic task, and is expressed using the following syntax:

`inf1 | perc ==> act | inf2 duration` τ `difficulty` d `delay` δ

where: `act` indicates the user action necessary to complete the basic task; `inf2` is a new piece of information that replaces `inf1` in WM after the execution of `act`; τ is the duration of the action; d is the difficulty level of the action; and δ is a delay.

```
1 ((noInfo | prescriptionFormVtbiP1 ==> noAction | vtbi300
              duration 1 difficulty 2/10 delay 0)
2  (vtbi300 | setVTBIP1 ==> type300 | noInfo
       duration 1 difficulty 2/10 delay 0)) ::
3 ((noInfo | prescriptionFormTimeP1 ==> noAction | time3
              duration 1 difficulty 2/10 delay 0)
4  (time3 | setTimeP1 ==> type3 | noInfo
              duration 1 difficulty 2/10 delay 0)) ::
5 ((clampOpeningP1 | clampP1 ==> openClampP1 | noInfo
              duration 1 difficulty 2/10 delay 0)) ::
6 ((noInfo | infusionReadyP1 ==> startInfusionP1 | noInfo
              duration 1 difficulty 2/10 delay 0))
```

Listing 3: Specification of the task for *Pump 1*

The task specified in Listing 3 includes six basic tasks:

(1) Read and memorise the vtbi value for pump 1 from the prescription chart (line 1 in Listing 3);
(2) Enter vtbi in pump 1 (line 2);
(3) Read and memorise the infusion duration for pump 1 from the prescription chart (line 3);
(4) Enter infusion duration in pump 1 (line 4);
(5) Open clamp 1 (line 5);
(6) Start infusion (line 6).

6 ANALYSIS

The developed model provides the basis for the analysis of memory load. To perform the analysis, two initial states are required which contain hypotheses about the memory load necessary to complete the task. An example initial state is in Listing 4, where x is the capacity of the WM (line 4).

```
1 < wm : WorkingMemory | memory :
2      (pump1 |-> goal(startInfusionP1) clampOpeningP1);
3      (pump2 |-> goal(startInfusionP2) clampOpeningP2),
4   capacity : x >
```

Listing 4: Example initial state of WM

Lines 2 and 3 represent the hypotheses about the initial content of the WM. Two goals are specified (i.e., starting the infusion), and two memory items to remember to open the roller clamps before starting the infusion (`clampOpeningP1` and `clampOpeningP2`).

Real Time Maude is then used to check whether a given WM capacity is sufficient to achieve the goal. This helps to obtain a quantitative evaluation of the complexity of the task (in terms of memory load) and to identify situations where the multitasking strategy could exceed the WM capacity of the operator. The following search command in Real Time Maude checks whether there is any such situation where the user forgets a piece of information that is relevant to the tasks and is therefore unable to complete the tasks successfully:

```
Maude> (utsearch [1] {initState1} =>!
       {< I:InterfaceId : Interface |  task :
           < T:Oid : Task | status : TS:TaskStatus,
                  A:AttributeSet > > REST:Configuration}
       such that TS:TaskStatus =/= completed .)
```

For the considered case study, the model checker finds interleaving strategies where the user is not able to complete the tasks when the capacity of the WM is set to 5. One such example is in Table 1: with WM capacity set to 5, the user can perform correctly the concurrent tasks up to *enter time 2* (i.e., an omission error occurs for action *open clamp 2*). If the WM capacity is set to 6, on the other hand, the analysis indicates that the user is always able to reach the goal successfully, using any multitasking strategy.

The results of our analysis are in line with the experimental results obtained in [3], and provide an explanation to the omission error in terms of CL.

7 REDESIGN

To check whether a design solution could be adopted to reduce memory load, the pump design was modified using the Next-Action Cueing technique [11]. A set of cues is presented in the user interface of the system at appropriate moments, to remind the operator what action should be performed next. For example, when the clamp needs to be opened, the operator does not need to retrieve this information from WM if there is a visual cue on the pump screen that indicates what needs to be done (e.g., a simple message "OPEN CLAMP" on the display of the pump).

This design solution was added to the model, by introducing a new *Cognitive Rule* in the subtask for setting up an infusion pump — perceiving the cue will trigger the activation and execution of a certain action (see line 5 in Listing 5).

```
1 ((noInfo | prescriptionFormVtbiP1 ==> noAction|vtbi300
                duration 1 difficulty 2/10 delay 0)
2  (vtbi300 | setVTBIP1 ==> type300 | noInfo
                duration 1 difficulty 2/10 delay 0)) ::
3 ((noInfo | prescriptionFormTimeP1 ==> noAction|time3
                duration 1 difficulty 2/10 delay 0)
4  (time3 | setTimeP1 ==> type3|noInfo
                duration 1 difficulty 2/10 delay 0)) ::
5 ((noInfo | clampP1 ==> noAction|clampOpeningP1
                duration 1 difficulty 2/10 delay 0)
6  (clampOpeningP1 | clampP1 ==> openClampP1|noInfo
                duration 1 difficulty 2/10 delay 0)) ::
7 ((noInfo | infusionReadyP1 ==> startInfusionP1|noInfo
                duration 1 difficulty 2/10 delay 0))
```

Listing 5: Modified task for setting up *Pump 1*

Analysis of this new version of the task indicates that the user is always able to complete the tasks with a WM capacity of 5 for all possible interleaving strategies.

Note that our framework facilitates testing of possible design solutions — developers still need to use their own experience in interactive system design to come up with a redesign solution.

8 RELATED WORK

The framework described here is an extension of one proposed by Cerone in [10] for the analysis of interactive systems. Cerone's model includes the description of human memory and a set of cognitive processes involved in human-computer interaction. However, that model focuses on a single device. Further, our framework enables the description of hypotheses about the capacity of WM, as well as timing features that enable a more fine-grained analysis of multitasking.

In [15], Harrison et al. model and analyse the same case study as the one described here. They use the IVY workbench and a cognitive framework based on the salience of information resources. Their cognitive model does not explicitly model WM, and is guided mainly by assumptions about the salience of user cues (see also [22] for a detailed description of the framework).

The work of Mori, Paternó and Santoro [20] has been used as baseline by several researchers for developing general methods and tools for model-based analysis of user tasks. In their work, the main focus is to assess the compatibility of task specifications with a user interface design. User workload is estimated using simple indicators such as the total number of cognitive tasks, see for example [5].

Combéfis et al. [12] present a formal framework for estimating effort, in terms of workload and complexity of operations, necessary to carry out a task with a given interactive system. The framework generates mental models from the specification of the user interface automatically. The mental model includes information about the sequence of actions and the knowledge a human operator needs to know to be able to interact successfully with the system. The main target of this work is the development of user manuals rather than evaluation of human multi-tasking strategies.

Houser et al. [18] developed a formal model for reasoning about human task load in avionics systems. Two action queues are used for reasoning about task scheduling strategies and workload. One action queue keeps track of which actions are currently being executed. The other queue keeps track of which actions the operator will eventually need to execute to accomplish the task. Actions can move from one queue to the other according to scheduling rules defined by the authors, based on parameters such as task priority level and task execution time.

Bolton et al. [16] have also developed a formal model for the analysis of human-machine interaction with multiple medical devices. The focus of their work is not on multitasking strategies but on how humans perceive multiple auditory alarms – the aim is to detect in advance situations where a sounding alarm could mask other concurrently sounding alarms, therefore preventing clinicians from perceiving one or more sounding alarms.

Anderson et al. [2] presented the ACT-R architecture, a rule based framework for modeling cognitive processes. This model has been applied in many different studies, e.g. to analyse the effect of phone distractions while driving [23], or to study the occurrence of possible aviation errors [9]. The ACT-R model only supports simulation, while our framework supports both simulation and model checking.

Several other user models have been developed in recent years for estimating human performance in applications such as automotive and avionics. For example, in [21], a formal model is introduced for the analysis of human performance in a typical driving task — driving behind a vehicle. Their model is specified as an hybrid I/O automaton, and incorporates specific notions such as distance estimation errors affecting a driver's perception, driver's reaction delays, and ability of a driver to anticipate the movement of other vehicles based on current temporal dynamics. In the avionics domain, various models have been developed for estimating human cognitive workload for monitoring and remote control of multiple Unmanned Air Vehicles (UAVs) (as in [17] and [19]). Because of the domain-specific concepts included in these models, the applicability to other application domains may be limited.

9 DISCUSSION AND CONCLUSION

An executable framework has been described in this paper. Its use for modelling and analysing different multitasking strategies in relation to multiple medical infusion pumps has been demonstrated. Results provided by the analysis of the case study are similar to those presented in [15], however the analysis in [15] explains omission errors in terms of salience of information, resulting in redesign solutions related to the visibility of specific user interface elements. Our framework, on the other hand, provides a different view on the problem, showing that certain omission errors could be interpreted in terms of CL – in these cases, redesign solutions that simply enhance visibility of certain user interface elements might not be sufficient to prevent the problem.

Our framework builds on accepted theories in cognitive psychology supported by experimental studies in multitasking. In the described framework, each task is characterised by a measure (rank) that quantifies the likelihood that the task will attract the user's attention. Using the rank value, it is possible to model multitasking strategies adopted by the user. The current version of the specification is developed in Real Time Maude, and uses a deterministic algorithm to choose the task to be executed. As part of future work, we aim to introduce probabilistic choice in relation to task selection. As a first step towards this goal the framework has been translated into Java (see [7]). This will allow us to quickly test various hypotheses about where probabilities should be introduced and how. After consolidating these understandings, a formal model will be developed using a probabilistic framework such as PVesta [1].

Currently, the adopted model checking technology is able to analyse efficiently the considered examples because device specifications are relatively simple. We expect to face potential scalability problems when analysing detailed models like the ones described in [14]. A possible solution in that case would be to explore the complementary use of different verification technologies, as suggested in [14].

Validation of the cognitive hypotheses embedded in our framework is also part of ongoing work. A first experimental study is being finalised in which users are asked to interact with a screen presenting two tabs, one presenting the main task, the other presenting a distractor task. Users will need to interact with both tabs concurrently. The cognitive load of the distractor task will be changed systematically during the experiment to test selected hypotheses about cognitive attention. The experiment will be modelled in our framework, to check the framework's predictive power and accuracy in anticipating the experimental results. The experimental results will also be used to fine-tune various model parameters related to hypotheses about task criticality level, cognitive load and the task ranking algorithm.

Acknowledgement. Michael Harrison (Newcastle University) provided useful feedback. Paolo Masci's work is supported by project NORTE-01-0145-FEDER-000016, financed by the North Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, and through the European Regional Development Fund (ERDF).

REFERENCES

[1] Musab AlTurki and José Meseguer. 2011. PVeStA: A Parallel Statistical Model Checking and Quantitative Analysis Tool. In *Algebra and Coalgebra in Computer Science*, Andrea Corradini, Bartek Klin, and Corina Cîrstea (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 386–392. https://doi.org/10.1007/978-3-642-22944-2_28

[2] John R Anderson, Dan Bothell, Christian Lebiere, and Michael Matessa. 1998. An integrated theory of list memory. *Journal of Memory and Language* 38, 4 (1998). https://doi.org/10.1006/jmla.1997.2553

[3] Jonathan Back, Anna Cox, and Duncan Brumby. 2012. Choosing to Interleave: Human Error and Information Access Cost. In *SIGCHI Conference on Human Factors in Computing Systems (CHI'12)*. ACM, 1651–1654. https://doi.org/10.1145/2207676.2208289

[4] Alan Baddeley. 1992. Working memory. *Science* 255, 5044 (1992).

[5] Eric Barboni, Jean-François Ladry, David Navarre, Philippe Palanque, and Marco Winckler. 2010. Beyond Modelling: An Integrated Environment Supporting Co-execution of Tasks and Systems Models. In *2nd ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS '10)*. ACM, 165–174. https://doi.org/10.1145/1822018.1822043

[6] Pierre Barrouillet, Sophie Bernardin, and Valérie Camos. 2004. Time constraints and resource sharing in adults' working memory spans. *Journal of Experimental Psychology: General* 133, 1 (2004). https://doi.org/10.1037/0096-3445.133.1.83

[7] Giovanna Broccia, Paolo Milazzo, and Peter Csaba Ölveczky. 2018. An Algorithm for Simulating Human Selective Attention. In *Software Engineering and Formal Methods*, Antonio Cerone and Marco Roveri (Eds.). Springer International Publishing, Cham, 48–55. https://doi.org/10.1007/978-3-319-74781-1_4

[8] Giovanna Broccia, Paolo Milazzo, and Peter Csaba Ölveczky. 2018. An Executable Formal Framework for Safety-Critical Human Multitasking. In *NASA Formal Methods*, Aaron Dutle, César Muñoz, and Anthony Narkawicz (Eds.). Springer International Publishing, Cham, 54–69. https://doi.org/10.1007/978-3-319-77935-5_4

[9] Michael D. Byrne and Alex Kirlik. 2005. Using Computational Cognitive Modeling to Diagnose Possible Sources of Aviation Error. *The International Journal of Aviation Psychology* 15, 2 (2005), 135–155. https://doi.org/10.1207/s15327108ijap1502_2

[10] Antonio Cerone. 2016. A Cognitive Framework Based on Rewriting Logic for the Analysis of Interactive Systems. In *Software Engineering and Formal Methods*, Rocco De Nicola and Eva Kühn (Eds.). Springer International Publishing, Cham, 287–303. https://doi.org/10.1007/978-3-319-41591-8_20

[11] Phillip H. Chung and Michael D. Byrne. 2008. Cue Effectiveness in Mitigating Postcompletion Errors in a Routine Procedural Task. *Int. J. Hum.-Comput. Stud.* 66, 4 (April 2008), 217–232. https://doi.org/10.1016/j.ijhcs.2007.09.001

[12] Sébastien Combéfis, Dimitra Giannakopoulou, Charles Pecheur, and Michael Feary. 2011. A formal framework for design and analysis of human-machine interaction. In *2011 IEEE International Conference on Systems, Man, and Cybernetics*. 1801–1808. https://doi.org/10.1109/ICSMC.2011.6083933

[13] Jan W de Fockert, Geraint Rees, Christopher D Frith, and Nilli Lavie. 2001. The role of working memory in visual selective attention. *Science* 291, 5509 (2001). https://doi.org/10.1126/science.1056496

[14] M.D. Harrison, P. Masci, J. C. Campos, and P. Curzon. 2017. Verification of User Interface Software: The Example of Use-Related Safety Requirements and Programmable Medical Devices. *IEEE Transactions on Human-Machine Systems* 47, 6 (2017), 834–846. https://doi.org/10.1109/THMS.2017.2717910

[15] Michael D. Harrison, José C. Campos, Rimvydas Rukšėnas, and Paul Curzon. 2016. Modelling Information Resources and Their Salience in Medical Device Design. In *8th ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS '16)*. ACM, 194–203. https://doi.org/10.1145/2933242.2933250

[16] B. Hasanain, A. D. Boyd, and M. L. Bolton. 2016. Using Model Checking to Detect Simultaneous Masking in Medical Alarms. *IEEE Transactions on Human-Machine Systems* 46, 2 (2016), 174–185. https://doi.org/10.1109/THMS.2014.2379661

[17] Constance L. Heitmeyer, Marc Pickett, Elizabeth I. Leonard, Myla M. Archer, Indrakshi Ray, David W. Aha, and J. Gregory Trafton. 2015. Building high assurance human-centric decision systems. *Automated Software Engineering* 22, 2 (2015), 159–197. https://doi.org/10.1007/s10515-014-0157-z

[18] Adam Houser, Lanssie Mingyue Ma, Karen M. Feigh, and Matthew L. Bolton. 2018. Using formal methods to reason about taskload and resource conflicts in simulated air traffic scenarios. *Innovations in Systems and Software Engineering* 14, 1 (01 Mar 2018), 1–14. https://doi.org/10.1007/s11334-017-0305-2

[19] J. Moore, R. Ivie, T. Gledhill, E. Mercer, and M. Goodrich. 2014. Modeling human workload in unmanned aerial systems. In *AAAI Spring Symposium Series: Formal Verification and Modeling in Human-Machine Systems*. https://www.aaai.org/ocs/index.php/SSS/SSS14/paper/view/7753

[20] Giulio Mori, Fabio Paternò, and Carmen Santoro. 2002. CTTE: support for developing and analyzing task models for interactive system design. *IEEE Transactions on software engineering* 28, 8 (2002). https://doi.org/10.1109/TSE.2002.1027801

[21] Jin Woo Ro, Partha S Roop, Avinash Malik, and Prakash Ranjitkar. 2018. A Formal Approach for Modeling and Simulation of Human Car-Following Behavior. *IEEE Transactions on Intelligent Transportation Systems* 19, 2 (2018). https://doi.org/10.1109/TITS.2017.2759273

[22] Rimvydas Rukšėnas, Paul Curzon, Ann Blandford, and Jonathan Back. 2014. Combining human error verification and timing analysis: a case study on an infusion pump. *Formal Aspects of Computing* 26, 5 (2014). https://doi.org/10.1007/s00165-013-0288-1

[23] Dario D Salvucci. 2001. Predicting the effects of in-car interface use on driver performance: An integrated model approach. *International Journal of Human-Computer Studies* 55, 1 (2001). https://doi.org/10.1006/ijhc.2001.0472