Validation of a Simulation Algorithm for Safety-Critical Human Multitasking

Giovanna Broccia[1](✉), Paolo Milazzo[1], Cristina Belviso[2], and Carmen Berrocal Montiel[2]

[1] Department of Computer Science, University of Pisa, Pisa, Italy
{giovanna.broccia,milazzo}@di.unipi.it

[2] Dipartimento di Patologia Chirurgica, Medica, Molecolare e dell'Area Critica, University of Pisa, Pisa, Italy
belvcristina@gmail.com, carmen.berrocalmontiel@med.unipi.it

Abstract. Multitasking has become surprisingly present in our life. This is mostly due to the fact that nowadays most of our activities involve the interaction with one or more devices. In such a context the brain mechanism of *selective attention* plays a key role in determining the success of a human's interaction with a device. Indeed, it is a resource to be shared among the concurrent tasks to be performed, and the sharing of attention turns out to be a process similar to process scheduling in operating systems. In order to study human multitasking situations in which a user interacts with more than one device at the same time, we proposed in a previous work an algorithm for simulating human selective attention. Our algorithm focuses, in particular, on safety-critical human multitasking, namely situations in which some of the tasks the user is involved in may lead to dangerous consequences if not executed properly. In this paper, we present the validation of such an algorithm against data gathered from an experimental study performed with real users involved concurrently in a "main" task perceived as safety-critical and in a series of "distractor" tasks having different levels of cognitive load.

Keywords: Validation · Simulation algorithm · Human-computer interaction · Safety-critical · Multitasking · Experimental study

1 Introduction

Nowadays we often interact with multiple devices or with a single device performing multiple tasks concurrently: keeping up several instant message conversations at once, answering an e-mail while listening to a talk at a conference, hanging out with social network while watching television, just to give some examples. However, despite what most of us could believe, the multitasking performance takes a toll on productivity and psychologists who study the mental processes involved in multitasking have found that the human mind and brain are not designed for doing more than one task at a time [23, 25]. We cannot focus on more than one thing at a time, what we can do is to switch from one task to another with incredible speed, since all these tasks use the same part of the brain [20].

One of the main resources to be shared in such multitasking contexts is the *human selective attention* – a selective activity whose purpose is to focus on one element of the environment while ignoring the others: several studies show how attentional limitations could cause troubles while performing multitasking [22,24,28]. According to [15], the *cognitive load* of each task (i.e., the amount of cognitive resources required by each task) influences the activity of the attentional mechanisms. In particular, focusing attention on a "main" task may be impeded by a secondary "distractor" task with a high cognitive load.

Moreover, in a multitasking context, another factor influencing the attentional mechanism is the fact that some tasks might be more critical than others. If the user is performing concurrent tasks, one of which is safety-critical and the others non-critical but characterised by a high cognitive load, such a cognitive load could cause users to draw away their attention from the safety-critical task.

To study such kinds of problems we proposed a model of safety-critical human multitasking (SCHM model), which describes the cognitive processes involved in a multitasking interaction with safety-critical systems [5].

Although the proposed model is designed according with psychological literature and results from experimental psychological studies, we conducted an experimental study with real users involved in a multitasking interaction on a web application with a "main" critical task and a secondary "distractor" task. Essentially, the main question we wanted to answer is:

Does the SCHM model "mirror" the task prioritisation that real users perform in a safety-critical multitasking context?

The experiment and the analysis of the experimental data, together with the development of a simulator in Java, allowed us to fine-tune the proposed model and to validate it.

We will present the model and its Java implementation in Sect. 2, the experimental study in Sect. 3, the design of the simulation experiments in Sect. 4 and the results we obtained in Sect. 5.

2 Safety-Critical Human Multitasking Model

The SCHM model is a mathematical model of human selective attention used to study situations where users concurrently interact with multiple devices and they have to voluntary choose which task to execute next.

The model is an extension and modification of the cognitive framework proposed by Cerone for the analysis of interactive systems [9]. Other related models of human multitasking are ACT-R [1], SEEV [30], STOM [29] and the models proposed in [19,21] (see [6] for a deeper discussion about related work).

As in that work, we describe the cognitive processes involved in HCI and the human working memory. However, we focus on multitasking and not on the analysis of the interaction with single device as in [9]. Moreover, our model also describes the limitations of the working memory, enabling us to reason about memory overload, and includes timing features, enabling us to reason about hazards caused by distractions.

We implemented a simplified version of the model as a Java simulator based on the algorithm proposed in [3] in order to have quick feedback about whether users can successfully and safely complete a given group of tasks at the same time. The simulator models human multitasking where users are allowed to perform a single task on each interface, we then decide not to model the interfaces.

Moreover, we implemented the multitasking model in Real-Time Maude, which is a rewriting logic language and tool which supports the formal specification and analysis of real-time systems [4,6]. The Real-Time Maude framework enables us to analyse safety-critical human multitasking through simulation and reachability analysis.

Within the model, each task is defined as a sequence of subtasks, which in turn are defined as a sequence of basic tasks (i.e. atomic action composing the task which cannot further decomposed). Between two basic tasks, it is possible to have some time, which could correspond to the time necessary to switch from one basic task to the next, but also to the time required by the device to process the received input and to enable the execution of the next basic task; we call such a time *delay*. Moreover, each basic task is characterised by a duration and a measure of how much it is difficult. By using such information we can compute the cognitive load of each task, starting from the definition of the cognitive load presented in [2]. Moreover, each task is characterised by a measure of how much the user perceives it to be safety-critical.

We define basic tasks as follows:

$$j \mid p \implies a \mid k \textbf{ duration } t \textbf{ difficulty } d \textbf{ delay } \delta$$

where j, k are information items, p is a perception (from the user viewpoint) about the device (interface) state, and a is an action to be performed on the device interface.

The basic task indicates that when the interface is on state p and the user has inside his/her working memory the information j, he/she can perform the action a and replace the information j with the information k in his/her working memory; such a basic task has duration t and difficulty d and it is enabled – and thus it can be executed – if and only if the delay δ is elapsed.

For each task we compute an α-factor representing the likelihood the task will attract the user's attention. At each step of the interaction the user chooses the task to be executed with a probability proportional to its α-factor. The α-factor of a task T is described as the product of three parameters: the cognitive load of the task (computed over the current subtask), the criticality level of the task, and the time elapsed since the last time the task has been executed:

$$\alpha_T = CogLoad_T \times c \times (waitTime_T + 1) \tag{1}$$

where:

- $CogLoad_T$ denotes the cognitive load of the task T;
- c denotes the criticality of the task T;
- $waitTime_T$ denotes the time the task T has not been executed.

As already mentioned, we implemented a Java simulator whose full specification is available at http://www.di.unipi.it/msvbio/software/AttentionSim.html.

We implemented a Java class for each element of the SCHM model:

- a `BasicTask` class;
- a `Subtask` class, containing a list of `BasicTasks`;
- a `Task` class, containing a list of `Subtaskss` and the criticality parameter c;
- a `WorkingMemory` class, modeling a (limited) working memory of a user;
- a `Configuration` class, which models the state of the simulation. An object of this class contains
  - a vector of `Task` objects, representing the concurrent tasks
  - a `WorkingMemory` object
  - variable `globalClock` of type `Integer` measuring the time elapsed from the beginning of the execution of the tasks
- a `Simulator` class, where the algorithm for simulating selective attention is specified.

The algorithm performs a main loop that essentially executes one basic task in each iteration. The basic task to be executed is the first basic task of one of the enabled tasks. For each of such enabled tasks, the α-factor is computed. These α-factors are then normalized in order to obtain a probability distribution used for the choice of the task. The first basic task of the chosen task is then executed as follows:

- the global clock is updated with the duration of the executed basic task;
- the chosen task is marked with a timestamp tracking the last time the task has been executed;
- the working memory is updated as specified by the executed basic task;
- the executed basic task is removed from the configuration.

If the algorithm reaches a configuration where no task is enabled, the main loop performs an iteration where only the global clock is updated with the minimum value needed to reach a configuration where at least one task is enabled.

The simulation terminates when all of the tasks in its configuration are completed.

3 Experimental Study

The development of the web application for the experimental study is part of a collaboration between computer scientists and psychologists from the University of Pisa (authors of this paper), which led to the definition of a set of appropriate tasks for the validation of the proposed algorithm. We defined two separate tests: one for evaluating the working memory (WM) performances of the participants, and one called *shared attention test* where users were asked to interact with two tasks concurrently.

3.1 Working Memory Span Tasks

Before the shared attention test, we administrate to the participants two different *working memory span tasks* (WMST) – standard tasks used to measure the performance of the working memory – in order to identify different inclinations to multitasking.

WMST are widely used in cognitive psychology [13] since WM plays an important role in a wide range of complex cognitive behaviours, such as comprehension, reasoning, and problem solving [18], and it is an important individual variable in general intellectual ability [11,12,17].

WMST were created to require not only information maintenance, but also the concurrent processing of additional information [8,14,27]. Such tasks involve performing two sequential activities: one mnemonic activity which imposes the memorisation and recall of a set of elements (such as digits or words); and one secondary activity which imposes a processing operation (e.g. comprehending sentences, verifying equations, or enumerating an array of shapes). Participants are asked to see or hear a sequence of elements spaced by a processing operation. At the end of each trial they have to recall the sequence correctly (which means recall the correct elements and in the correct order), with increasingly longer sequences being tested in each trial (from two to five elements per trial).

We administrate two different WM span tasks: the reading span task (RST) [14], and the operation span task (OST) [27][1]. In both, a sequence of numbers of variable length (from 2 to 5 numbers) is presented on the screen; each number is spaced by a sentence (RST) or an equation (OST) to evaluate. When all numbers are presented to the users, they have to recall the numbers in the exact order they were presented. We administrate 3 test repetitions for each sequence length: in total 12 repetitions for both RST and OST.

As regards the procedure for measuring the WM capacity, different scoring procedures are available in the literature [13]; we use the partial-credit unit scoring (PCU), namely the mean proportion of elements within a test that were recalled correctly. The PCU for each user is computed as follows:

$$PCU = \frac{\sum_{i=1}^{N} \frac{b_i}{a_i}}{N}$$

where N is the number of items, b_i the number of elements correctly recalled, and a_i the number of elements to recall.

[1] Available at http://pages.di.unipi.it/milazzo/AppSpans/.

3.2 Shared Attention Test

As regards the shared attention test[2], we defined two tasks, (i) a main and critical one, and (ii) a secondary distracting task (with different levels of cognitive load):

i As shown in Fig. 1, in the main task users visualise on the screen a chain of 9 rings and a black pellet which randomly moves left and right along the chain. Every time the task starts, the black pellet is on the central green ring and moves randomly every second. Users are asked to avoid that the black pellet reaches one of the red rings at the two ends of the chain, by pushing two buttons on the screen which move the pellet in the two directions: if they do not succeed, the task fails.

ii In the secondary distracting task (shown in Fig. 2), users visualise on the screen a sequence of boxes and a keyboard. At cyclic intervals, a letter appears inside a box; letters appear one by one until all boxes are full.
Users have to find and push on the keyboard the letter corresponding to the one inside the box indicated by the arrow, until all the letters are inserted in the same order they were presented. Every time they have to insert a new letter (i.e. the previous letter has been successfully inserted and the next one has appeared) the keyboard changes.
Such activity has a total duration expressed through a timeout, visualised by a decreasing number and a black progress bar. Once the timeout expires the task is concluded: if the user did not succeed in inserting all the letters, the task is considered failed, otherwise the task succeeds.

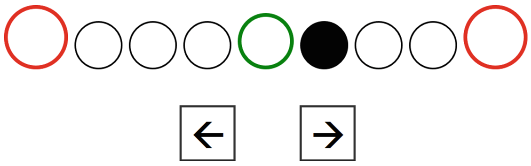

Fig. 1. Main critical task. (Color figure online)

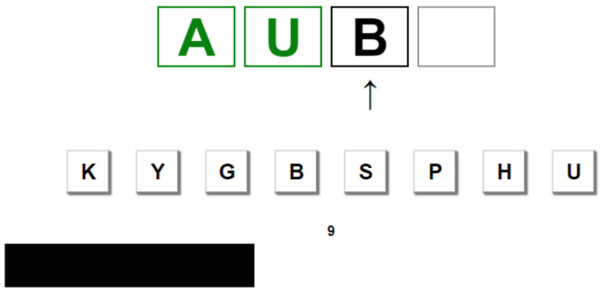

Fig. 2. Secondary distracting task.

[2] Available at http://pages.di.unipi.it/milazzo/AppSpans2/.

The tasks are presented on two separate tabs of the same window: users can see only one of the two tabs at a time, and they can switch from one to another by pushing the space bar. So, the user has to perform the two tasks concurrently by interleaving them. Both tasks have to be completed successfully.

The secondary task is instantiated with different levels of cognitive load. In order to do this, three different parameters of the secondary task are varied: the number of letters to insert (*letters*), the number of keys composing the keyboard (*keys*), and the total duration of the task (*duration*). As regards the letters, we define three kinds of task where users have to insert 2 letters, 4 letters, and 8 letters, respectively; as regards the keys, we define a task where users should search for the right letter in a *single* keyboard (i.e. the number of keys composing the keyboard is equal to the number of letters to be inserted), and a kind of task where they have to search for the letter in a *double* keyboard (i.e. the number of keys is twice as the number of letters to be inserted); as regards the duration, it can be either 18, 22, or 26 s.

Summing up, we have 18 different levels of CL for the secondary task, and the web application administrates 3 test repetitions for each level (presented randomly). In total, users have to perform 54 test repetitions. Accordingly to the definition of the α-factor (see Eq. 1) each task has a different α value depending on its cognitive load, its criticality and the time the task has not been chosen by the user.

3.3 Participants

The definition of the experimental study has been submitted to the ethical committee of the University of Pisa, which authorised the administration of the test. To take part in the experimental study, participants were asked to sign an informed consent form and a consent for the processing of personal data.

We performed two test sessions taking care that the environment, the provided equipment, and the test timetable were the same in both sessions. Exclusion criteria for the participants were cognitive functions disorders and drug consumption with an effect on such functions. Participation has been voluntary and without any incentive; participants were free to abandon the test at any time.

In total, 26 participants took part in the experimental study: mother-tongue Italian, of both sexes (60% men, 40% women), aged between 18 and 40 years, and with a normal visual acuity (or corrected by lenses).

3.4 Data Collection

WM Span Tasks. The web application is able to collect users' answers for both WM span tasks for each item, and it is thus able to compute the PCU for the OST task and for the RST task. From such data, we can compute the total PCU score for each participant, calculated as the mean of the scores of both tasks. Total PCU values go from a minimum of 0.35 to a maximum of 0.97.

Shared Attention Test. For the shared attention test, the web application is able to track every user action, and it is thus able to compute the number of errors for the main task and for the secondary task, as well as the time users pass on the main task and on the secondary task.

In order to explore correlations between the users' PCU and their multitasking performance – which would be consistent with the relevant psychological literature – we divide PCU values into 3 intervals and we divide participants into 3 different groups:

1. *lowPCU*: user total PCU ≤ 0.80;
2. *mediumPCU*: $0.80 <$ user total PCU ≥ 0.90;
3. *highPCU*: user total PCU > 0.90;

In Table 1 we show, for all users whose PCU is in a given interval, the average time to find and push the right letter on the keyboard for each level of cognitive load (number of letters to find and keybord size). We notice that the higher the PCU, the faster participants find and push the correct letter. On the other hand, from the collected data the total duration of the secondary task seems to have no influence on the time required by the user to find and push the letter on the keyboard. Hence, we didn't group users on this parameter while computing the average times shown in Table 1.

Table 1. Average time to find and push the correct letter for each PCU group and each combination of number of letters (nL) and number of keys (nK) in the keyboard.

	2L 2K	***2L 4K***	***4L 4K***	***4L 8K***	***8L 8K***	***8L 16K***
lowPCU	1.257	1.4	1.528	1.843	1.538	1.931
mediumPCU	1.197	1.262	1.466	1.711	1.421	1.665
highPCU	1.042	1.175	1.2	1.471	1.291	1.572

As regards the time spent on the main task and on the secondary task, from data we can deduce how much each participant perceives as critical the main task with respect to the secondary task. We call *criticality* the percentage of time a user stays on the main task with respect to the secondary task. In order to check if the less a user perceives the main task as critical (i.e. the criticality is lower), the more he/she fails in such task, we divide the criticality values (which vary from 48% to 66%) into 2 groups: the first groups values up to 57%, the second groups values higher than 58%. We then compute the average number of errors for each of these groups, and we find that the more the main task is perceived as critical, the less the users fail in it: 3,36 errors on average for the low criticality group and 1,8 errors on average for high criticality group. Since we observed this correlation, we decided to keep these two groups of users separated in the analysis, by identifying 2 additional subgroups that we call *lowCriticality* and *highCriticality*.

Therefore, we consider, overall, 6 different groups of users, by considering the 3 PCU groups and the 2 criticality subgroups, that are:

1. *lowPCU – lowCriticality*
2. *lowPCU – highCriticality*
3. *mediumPCU – lowCriticality*
4. *mediumPCU – highCriticality*
5. *highPCU – lowCriticality*
6. *highPCU – highCriticality*

4 Simulation Experiments

For each group devised above, and for each level of cognitive load of the secondary task, we implement a different simulation experiment. Parameters of these simulations have been estimated through data fitting.

Main Task. As regards the main critical task (i.e. the one where users are asked to avoid that the black pellet reaches one of the two red rings), we implement it as a sequence of basic tasks, whose duration is set to 1 and difficulty is set to 0.1. We implement the same task for each PCU group, and two variants of the task for each criticality subgroup: for *highCriticality* we set the criticality of the task to 40, for *lowCriticality* we set it to 4.

Secondary Task. As explained in Sect. 3.2, in the secondary task, a letter appears inside the white boxes at a specific time: the total duration of the task is divided by the number of letters to insert, and such measure gives us the interval of time between the appearance of a letter and the next one. Therefore, the secondary task could be defined as follows:

$$
\begin{aligned}
&noinfo \mid letter_1 \;\Rightarrow\; findL_1 \mid noInfo \;\textbf{duration}\; t \;\textbf{difficulty}\; d \;\textbf{delay}\; \delta_1\\
&\quad\vdots\\
&noinfo \mid letter_n \;\Rightarrow\; findL_n \mid noInfo \;\textbf{duration}\; t \;\textbf{difficulty}\; d \;\textbf{delay}\; \delta_n
\end{aligned}
$$

where:

– n is the number of letters to insert, and thus the number of basic tasks composing the secondary task;
– $findL_i$ represent the action of finding and pushing in the keybord the $i-th$ letter appeared;
– t_i is the duration of the action $findL_i$, set as the average duration for a given combination of number of letters and keys, according to the duration presented in Table 1;
– d_i is the difficulty of the action $findL_i$, which we set to 6;
– δ_i denotes the time which has to elapse so that the letter appears, namely the interval of time between the appearance of two letters minus the duration t_i.

Actually, the appearance of a letter in the secondary task is independent of the previous letter, which means that each letter in a sequence appears as soon as the given time interval has passed, whether the previous letter has been correctly inserted or not. Instead, the task presented above, implies that the delay δ_i of each basic task (namely of each letter) starts elapsing as soon as the basic task becomes the first one of the current subtask, which means that by modelling the secondary task in that way, the appearance of a letter would wait for the correct insertion of the previous letter.

We thus decided to model a different task for each letter to be inserted in the secondary task, namely to divide the *unique* task presented above into n different tasks:

$$info_i \mid letter_i \;\Rightarrow\; findL_i \mid info_{i+1} \;\textbf{duration}\; t \;\textbf{difficulty}\; d' \;\textbf{delay}\; \delta'_i$$

In this way, each delay of each task represents the time which has to elapse from the beginning of the simulation of the interaction with the secondary task in order that the letter appears.

Each task composing the secondary task shares a memory. In this way it is possible to ensure that all tasks are executed in the right order: each task has to put inside the memory the information to be retrieved by the next task to be executed so that a task cannot be carried out until the previous task has not been accomplished (i.e. letters have to be inserted in the correct order).

Moreover, the difficulties of each task are computed in order to ensure that the cognitive load of each task is equal to the one of the unique task presented above.

Simulation Settings. In total, we implemented 108 different tests for each combination of PCU levels (3), levels of cognitive load (18), and level of criticality (2). For each of these combinations, we performed 1000 simulations and we computed the average value for the time to complete the simulated secondary task and the maximum time the simulated main task is ignored.

It is worth to note that the simulated tests are approximation of the real users performance. For instance, by modelling the secondary task as a single basic task where the user finds and push the right letter in the keyboard, we cannot simulate the case where a user goes backward and forward from the main task to the secondary task, just to check if the next letter appeared. Hence, in the real data we subtracted such time from the time passed on the secondary task, and we added it to the time passed on the main task.

5 Results

We performed 1000 simulations for each of the 6 groups presented above. Namely during a simulation are executed 18 different tests (one for each level of cognitive load of the secondary task), where the main task has a given criticality according to which of the 2 criticality subgroups we are simulating, and the secondary task has precise durations and difficulties according to which of the 3 PCU groups we are simulating. We performed each simulation in order to check if the "simulated users" behave as the real users. In particular, we observed:

1. If the time passed on the main task with respect to the time passed on the secondary task is equal to that observed in the data;
2. If the number of errors in the main task follows the same distribution of the one observed in the data;
3. If the number of errors in the secondary task follows the same distribution of the one observed in the data.

The time passed on the main task respect to the time passed on the secondary task is what we called criticality in Sect. 3.4.

As regard the main task, we know that it fails as soon as the black pellet reaches one of the red rings, and we know that the minimum number of steps for the pellet to reach a red ring (starting from the green one) is 4 steps; the longer such a task is ignored by the user, the higher is the probability to fail it. We thus consider the maximum wait time of the main task – namely the longest time it has been ignored – as a measure of the probability to fail it: the higher is the maximum wait time, the higher is the probability.

On the other hand, for the secondary task we consider the time its last basic task has been executed and we compare such time with the total duration of the task: the higher is the difference between such two values (i.e. the former is greater than the latter), the higher is the probability that the secondary task has failed.

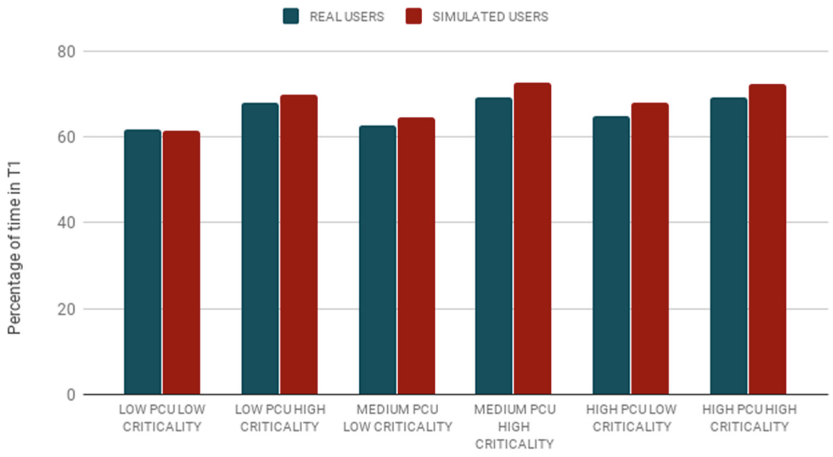

Fig. 3. Time on task T1 for simulated and real users.

Regarding the number of errors observed from data, it is worth to note that it is particularly low and it can be subject to statistical noise. Therefore, we concentrate more on the criticality and as regards the errors (for both the main task and the secondary task) we analyse if the simulated trend is similar to the real trend.

Criticality. As regards the criticality, we compute the percentage of time the simulated test has passed on the main task. We compute the average of such measures for the entire simulation, for each of the six groups.

As shown in Fig. 3, the time the simulated users pass on the main task is very close to the time real users pass on the main task: the level of approximation varies from −0.2% to +3.3%.

Tasks Fails. As regards the errors on the main task, we compute the average of the maximum wait time for the main task of the entire simulation, for each of the six groups, and we compare such measures with the average number of errors for each of the six group.

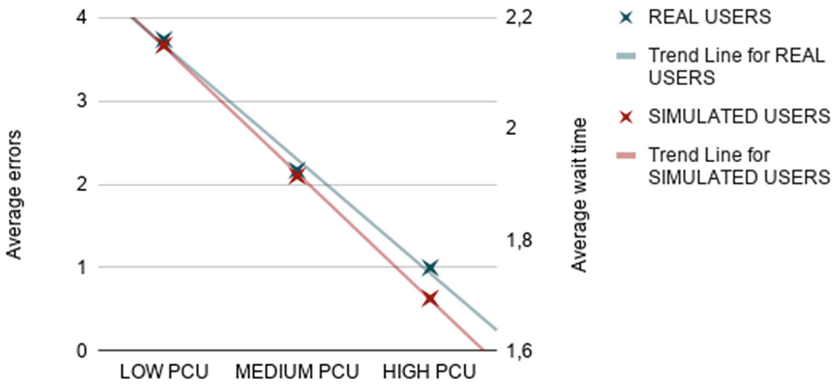

Fig. 4. Average number of errors for the main task (T1) and average wait time when varying PCU.

As shown in Fig. 4 the probability to fail the main task decreases as the level of PCU increases, as well as the number of errors which decreases as the users' PCU increases. The probability to fail the main task decreases as well as the criticality increases, and such trend is observed also in the data. Finally, as shown in Fig. 5, the probability of fail and the average number of errors decrease for each of the six groups as the PCU decreases and the criticality passes from low to high.

Regarding the errors on the secondary task, we subtract the final duration of the secondary simulated task to the total duration of the task and we compute the average of such measures for the entire simulation, for each of the six groups.

Also in this case, we notice a decrease of the probability of errors in the secondary task when the PCU level increases, and a growth in the probability of errors when the criticality increases. We observe the same trend in the data.

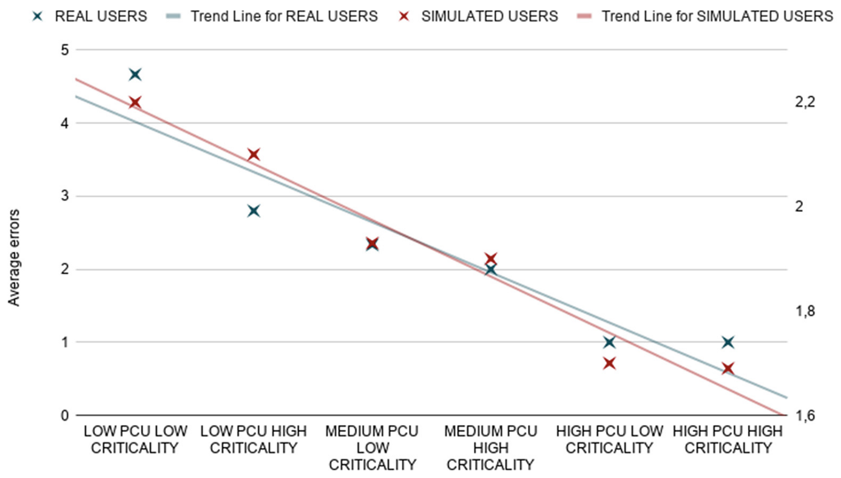

Fig. 5. Average number of errors for the main task (T1) for each group and average wait time for simulated tasks for each group.

Concluding, it is worth to note that the small sample size and the approximation of the simulation are factors to be taken into account when analysing the results obtained. However, such results agree with the data gathered from the experimental study, for both the probability of fails in both tasks and the time passed on the main task. We, thus, consider our algorithm and, in particular, SCHM model capable to produce relevant results, according to the behaviour of real users in a safety-critical human multitasking context.

References

1. Anderson, J.R., Matessa, M., Lebiere, C.: ACT-R: a theory of higher level cognition and its relation to visual attention. Hum. Comput. Interact. **12**(4), 439–462 (1997)
2. Barrouillet, P., Camos, V.: The time-based resource-sharing model of working memory. Cogn. Neurosci. Work. Mem. **455**, 59–80 (2007)
3. Broccia, G., Milazzo, P., Ölveczky, P.C.: An algorithm for simulating human selective attention. In: Cerone, A., Roveri, M. (eds.) SEFM 2017. LNCS, vol. 10729, pp. 48–55. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-74781-1_4
4. Broccia, G., Milazzo, P., Ölveczky, P.C.: An executable formal framework for safety-critical human multitasking. In: Dutle, A., Muñoz, C., Narkawicz, A. (eds.) NFM 2018. LNCS, vol. 10811, pp. 54–69. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-77935-5_4
5. Broccia, G.: A formal framework for modelling and analysing safety-critical human multitasking. Ph.D. Thesis. University of Pisa, Department of Computer Science (2019)
6. Broccia, G., Milazzo, P., Ölveczky, P. C. Formal modeling and analysis of safety-critical human multitasking. Innov. Syst. Softw. Eng., 1–22 (2019)
7. Dangerous distraction: Safety Investigation Report B2004/0324. Australian Transport Safety Bureau (2005). https://www.atsb.gov.au/media/36244/distraction_report.pdf

8. Case, R., Kurland, D.M., Goldberg, J.: Operational efficiency and the growth of short-term memory span. J. Exp. Child Psychol. **33**(3), 386–404 (1982)
9. Cerone, A.: A cognitive framework based on rewriting logic for the analysis of interactive systems. In: De Nicola, R., Kühn, E. (eds.) SEFM 2016. LNCS, vol. 9763, pp. 287–303. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-41591-8_20
10. Clark, T., David, Y.: Impact of clinical alarms on patient safety. American College of Clinical Engineering Healthcare Technology Foundation Web site (2009)
11. Conway, A.R.A., Cowan, N., Bunting, M.F., Therriault, D.J., Minkoff, S.R.B.: A latent variable analysis of working memory capacity, short-term memory capacity, processing speed, and general fluid intelligence. Intelligence **30**(2), 163–183 (2002)
12. Conway, A.R.A., Kane, M.J., Engle, R.W.: Working memory capacity and its relation to general intelligence. Trends Cogn. Sci. **7**(12), 547–552 (2003)
13. Conway, A.R.A., Kane, M.J., Bunting, M.F., Hambrick, D.Z., Wilhelm, O., Engle, R.W.: Working memory span tasks: a methodological review and user's guide. Psychon. Bull. Rev. **12**(5), 769–786 (2005). https://doi.org/10.3758/BF03196772
14. Daneman, M., Carpenter, P.A.: Individual differences in working memory and reading. J. Verbal Learn. Verbal Behav. **19**(4), 450–466 (1980)
15. de Fockert, J.W., Rees, G., Frith, C.D., Lavie, N.: The role of working memory in visual selective attention. Science **291**(5509), 1803–1806 (2001)
16. Dingus, T.A., Guo, F., Lee, S., Antin, J.F., Perez, M., Buchanan-King, M., Hankey, J.: Driver crash risk factors and prevalence evaluation using naturalistic driving data. Proc. Natl. Acad. Sci. **113**(10), 2636–2641 (2016)
17. Engle, R.W., Tuholski, S.W., Laughlin, J.E., Conway, A.R.A.: Working memory, short-term memory, and general fluid intelligence: a latent-variable approach. J. Exp. Psychol. Gen. **128**(3), 309 (1999)
18. Engle, R.W.: Working memory capacity as executive attention. Curr. Dir. Psychol. Sci. **11**(1), 19–23 (2002)
19. Gelman, G., Feigh, K.M., Rushby, J.M.: Example of a complementary use of model checking and human performance simulation. IEEE Trans. Hum. Mach. Syst. **44**(5), 576–590 (2014)
20. Hamilton, J.: Think you're multitasking? Think again. Morning Edition (2008). https://www.npr.org/templates/story/story.php?storyId=95256794&t=1566830957675
21. Houser, A., Ma, L.M., Feigh, K., Bolton, M.L.: A formal approach to modeling and analyzing human taskload in simulated air traffic scenarios. In: Complex Systems Engineering (ICCSE), pp. 1–6. IEEE (2015)
22. Lavie, N., Hirst, A., De Fockert, J.W., Viding, E.: Load theory of selective attention and cognitive control. J. Exp. Psychol. Gen. **133**(3), 339 (2004)
23. Mittelstädt, V., Miller, J.: Separating limits on preparation versus online processing in multitasking paradigms: evidence for resource models. J. Exp. Psychol. Hum. Percept. Perform. **43**(1), 89 (2017)
24. Pashler, H.: Dual-task interference in simple tasks: data and theory. Psychol. Bull. **116**(2), 220 (1994)
25. Redick, T.S., et al.: Cognitive predictors of a common multitasking ability: contributions from working memory, attention control, and fluid intelligence. J. Exp. Psychol. Gen. **145**(11), 1473 (2016)
26. Shorrock, S.T.: Errors of memory in air traffic control. Saf. Sci. **43**(8), 571–588 (2005)
27. Turner, M.L., Engle, R.W.: Is working memory capacity task dependent? J. Mem. Lang. **28**(2), 127–154 (1989)

28. Wickens, C.D.: Processing resources and attention. Mult. Task Perform., 3–34 (1991)
29. Wickens, C.D., Gutzwiller, R.S., Santamaria, A.: Discrete task switching in overload: a meta-analyses and a model. Int. J. Hum. Comput. Stud. **79**, 79–84 (2015)
30. Wickens, C.D., Sebok, A., Li, H., Sarter, N., Gacy, A.M.: Using modeling and simulation to predict operator performance and automation-induced complacency with robotic automation: a case study and empirical validation. Hum. Factors **57**(6), 959–975 (2015)