Enhancing the Validation of Human Factors in User Interface Software Testing with AI

Marie Goetz(✉)

German Aerospace Center (DLR), Institute of Flight Systems, Lilienthalplatz 7, 38108 Braunschweig, Germany
marie.goetz@dlr.de

Abstract. This paper introduces a novel approach to human factor focused software testing in the loop that combines Optical Character Recognition (OCR), and large language models (LLMs) for scenario-driven, non-intrusive testing of graphical user interfaces (GUIs). By defining a testing scenario and analyzing a captured image of the UI, the system identifies visual elements using OCR and detection algorithms, storing them in a standardized format. This output serves as input for an LLM, which interprets scenario requirements and generates user actions, such as mouse clicks or keyboard inputs, to achieve the defined objectives.

The system operates in a closed loop, iterating until the target outcome, e.g., the scenario is achieved, failures are detected, or usability metrics are generated. This non-intrusive method avoids software modification, making it ideal for cases where direct instrumentation is infeasible, like closed-source software. Fine-tuning the LLM for domain-specific metrics, such as cognitive load and task complexity, enhances usability testing by simulating users with varying expertise. The concept also supports environmental simulations, enabling realistic testing of safety-critical systems like automotive or aviation software.

This idea offers a scalable and flexible addition to traditional GUI testing, enabling early in the software development loop human factor insights, and a first level of functionality validation for safety-relevant applications.

Keywords: Large Language Model · Testing in the Loop · User Studies · Human Factors · Evaluation · Error Detection · Workload · Usability

1 Introduction

Evaluating human factors such as usability, error management, workload, and interaction intuitiveness is crucial for developing reliable and user-centered software, particularly in safety-critical systems like airborne applications. These sys-

M. Kurosu and A. Hashizume (Eds.): HCII 2025, LNCS 15769, pp. 59–76, 2025.
https://doi.org/10.1007/978-3-031-93861-0_4

tems demand a high level of precision, resilience, and user-friendliness to minimize risks and ensure operational safety. While human participants have traditionally been the cornerstone of usability studies, this approach has limitations and binds resources. The proposed concept, which simulates human interaction using a Large Language Model (LLM) in a looped testing framework, addresses these limitations.

Human participants, though extremely helpful for understanding real-world user behavior, are constrained by several factors. First, safety-critical systems such as airborne cockpit software often require extensive testing under a wide range of scenarios, including rare edge cases or extreme conditions. Recreating such scenarios for human participants can be both costly and logistically challenging.

Another challenge lies in the resource-intensive nature of human-based studies. Recruitment, training, and scheduling for participants require significant time and financial investment, often leading to limited sample sizes. This is particularly problematic in research contexts, where access to a diverse and representative user group is difficult to achieve. The voluntary nature of such studies also introduces biases, as participants may be more motivated or resilient to errors and inconveniences compared to typical end-users. These factors can skew results and reduce the quality of insights. As these issues are prevalent, the question arises, how can users be simulated. And how can the effects on the usability be measured? Although this question is not new, research progress and obtained knowledge enables new possibilities to be investigated.

Simulating human decision through an LLM could be a practical and effective option. Thereby we use a chain of multiple tools, such as the LLM itself, screen recording of a User Interface (UI), Optical Character Recognition (OCR), simulated user inputs and decision-making by loops and conditional statements. This approach allows developers to emulate user behavior programmatically, overcoming the logistical constraints of human-based studies for at least an early feedback. Thereby, no application under test-specific handling is implemented. Either way, this approach cannot avoid these studies, but generating simple feedback early on can simplify the development process and guide the design decision-making [15]. Furthermore, an LLM-driven system can simulate diverse user profiles, from novice to expert, providing a wide range of perspectives on usability and error handling, which cannot be, in general, guaranteed in small user studies [11].

The implementation of this concept extends beyond usability testing. By integrating metrics for cognitive load, task performance, and error handling, the approach provides a broad analysis of human factors, offering insights that are critical for optimizing user interfaces. It is particularly beneficial for assessing safety-critical applications, where intuitive and error-tolerant design is essential to ensure operational safety and efficiency.

In this paper, we propose a novel testing concept that leverages LLMs in combination with iterative simulation techniques to evaluate human factors in software systems. By addressing the limitations of traditional human-based stud-

ies, this approach provides a scalable, adaptable, and non-intrusive method for assessing usability, workload and functionality, paving the way for more efficient and user-centered software development processes. We tested the approach for three scenarios and compared the results with 11 real users.

This paper introduction is followed by a background chapter on how to simulate human users and their decision-making process. In addition, it presents the related work in Sect. 2.1. Afterwards the Sect. 3, presents the concept itself. For a small insight, we show in Sect. 4 a brief overview of the implementation, which is followed by an initial comparison to real user study results. Finally, in Sect. 5, we summarize the results with a conclusion and investigate the conducted work.

2 Simulation of Human Users

Simulating human interaction is in general not something new, and a lot of research has been conducted. In this context, it consists of three basic tasks which work in conjunction:

1. Perception & Interpretation
2. Strategy-Planning & Decision-Making
   (a) Identify Choices: Reasoning and Action
   (b) Evaluate the Solutions: Question Generation
   (c) Decide on one Option: Consideration
3. Interaction

In general, an interaction starts with a perception of a given UI, where all visible elements are interpreted and categorized. These elements are used for the second step, where possible choices are identified. The decision-making process can be simplified to identifying possible options, evaluating the possible solutions and deciding upon one option. For the evaluation, a context specific and individual rating is being made, where hedonic assessment, possible rewards, anticipation, time, probability, and expectancy come in effect. Thereby, the user-based knowledge has a significant influence. How and why a decision is made, is based on a complex and situative procedure [6]. This individual decision is followed by an interaction, where the user performs its decision. This process can be iterated multiple times and stopped at any time. [1,5,32] By simplifying the environment while maintaining the dynamic complexity and reducing the output, to a defined list, a cognitive process can be part of a dynamic task. In particular, these systems are mainly accurate under hard constrains with low stake. [4,7,8] To ensure such a limited user experience, an LLM needs to be designed accordingly. Depending on its statistical language model type and their used datasets, the result can gravely distinguish [8]. A simulated user should be able to replicate all relevant characteristics of a human user. As stated by Miller et al. in 2003, there are at least 12 elements that are relevant to identify a human user. These elements can be grouped into external and internal human characteristics. Foremost is the environment which, as an external factor, has a significant impact. This includes the sensory and cognitive demands of the task, such as

noise, skill requirements and overall environment. In addition to them, there are social requirements and mental demands posed by the task, which includes primary stress and workload. On the other hand, there are six internal factors. This includes primarily anthropometric features, such as height or weight, sensory and physiological attributes, namely vision, touch or conditioning, cognitive and psychological characteristics, like personality and intelligence quotient and social characteristics, for instance the empathy quotient. Furthermore, the user state has a significant influence on the situation. All these elements are crucial for a human factor evaluation. Depending on the scenario of a user-study, the importance can differ. But for an automated UI testing process they can be simplified to a reduced scope, since some of them are undesirable either way and some are not investigated in the current state of the art. [13,19]. For this specific use case, some factors are not particularly helpful and therefore are not needed in a simulated user, since in an early user feedback loop they would mainly impede the results. Furthermore, for some it is unclear how a LLM can be prompted/-trained to respect these constraints, or a simulation should be conducted, since there is nearly no research regarding their mouldability [26,32]. Figure 1 shows the elements that are subject to the presented validation. It mainly consists of environmental impacts, cognitive and mental demands by the task, and cognitive and psychological characteristics of the user. In addition, lately, significant research regarding interaction behavior and movement simulations has been conducted. Even if these models are not impeccable, they can indicate additional flaws and issues regarding user experience [12,18].

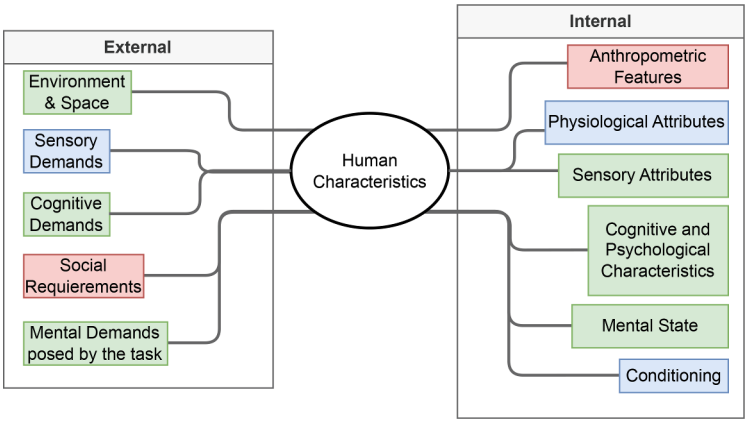

Fig. 1. Human Characteristics ordered by their estimated simulatability (green: focused on, blue: not considered, red: unclear) in the context of Simulated Large Language Model User. Adapted from Miller et al. (Color figure online) [19].

The last major element in this chain is the widely used OCR approach regarding characters [20,21,25]. Unfortunately, for other GUI elements, such as buttons or input fields, there is no such well-established way. There are multiple

approaches that enhance these OCR techniques, but they are often not generalized for other use cases or have a higher failure rate [10,30].

2.1 Related Work

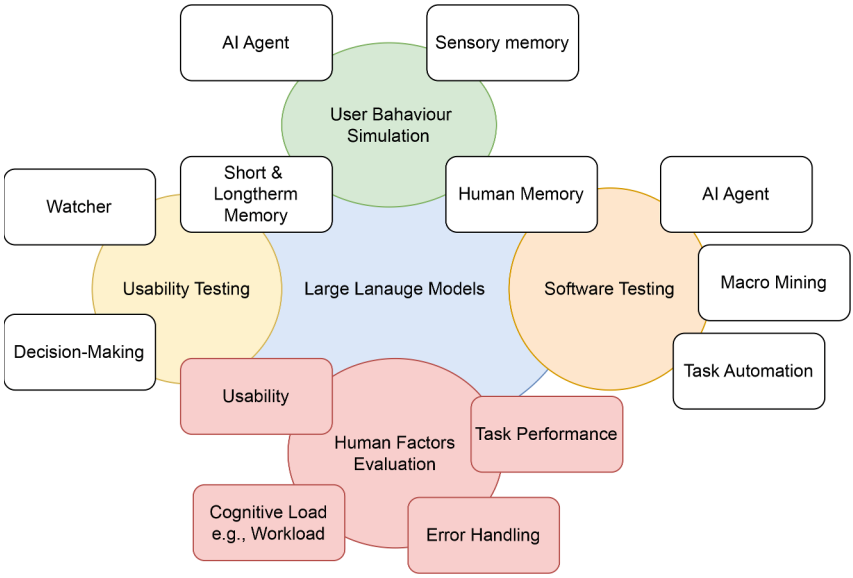

Fig. 2. Related Research defining this Works Research Gap.

Large Language Models (LLM) are used in a wide range of recent research. In the context of this work, there are three closely related topics that influenced this work, as shown in Fig. 2. Foremost, there is a wide research area regarding software testing, where Artificial Intelligence (AI) agents are trained to emulate users' interactions with a GUI. Their main issue thus far is the task completion itself, since the usage of LLMs comes with various random behavior and hallucinations. In addition, they are mostly focused on finding software bugs instead of validating and evaluating some sort of usability. In addition, there is no concept of user knowledge being followed to simulate different user groups. So even with AI agents, macro mining and task automation, there is no evaluation of usability effects [3,17,29].

Moreover, there is research focusing on usability testing. Their current focus is based on LLM memory management and decision-making processes by reducing the workload by simplifying the central task into subtasks. Thereby, they focused on evaluating the integration of online LLM services, such as GPT-3.5 or GPT-4 by OpenAI[1], into an on-device LLM, to reduce costs add latency for the usage [14].

[1] https://openai.com.

Finally, there has been conducted some research regarding the users' behavior simulation, where most of the related characteristics were investigated. Especially the work conducted by Wang et al. is one of the closest ones. They propose an LLM-based-framework with a designed environment to simulate human behavior. They were able to demonstrate a well performing human-like behavior with their framework. In addition, the work by Zhang et al. can be named as a comparable approach. Basically, they investigated a loop that iterates over a user interface to perform a specific task. They were able to perform some simple scenario tasks for their given mobile application WikiHow[2]. They mentioned the possibility of using this concept for further benchmarks, but didn't investigate the options. They compared different prompting techniques and analyzed possible rewards of different LMMs, while having the application design in mind. In addition, they didn't perform automated sanity checks and didn't handle misguided decisions [27,31].

Using these results, the research for human factors evaluation seems promising and, in particular, could simplify the development process. The following research questions arise:

1. Is a simulated user capable of simulating human usage behavior well enough to analyze cognitive load and usability?
2. Can task performance and error handling be acceptably predicted to assist the development process?

3 Concept

This approach integrates LLMs with OCR and interaction emulation to simulate user behavior, evaluate usability metrics, and detect functional anomalies. The methodology focuses on non-intrusive techniques, allowing it to be applied even to closed-source or legacy systems without access to the underlying codebase (Fig. 3).

The process begins with capturing GUI visuals through a screenshot or a physical camera that captures the user interface. This image is then used by the OCR, to extract textual elements and graphical components such as buttons, sliders, and menus. Since LLM tends to assume there are non-existing elements, a deterministic way was necessary. Therefore, we first detected elements, and only used LLM to validate or modify their existence. These elements are structured into a machine-readable format, which standardizes the representation of the interface, encapsulating its elements, attributes, and possible interactions.

Once the user interface data is captured in this format, it checks every second for changes. If no changes are recognized, the existing data interchange format will be used to trigger the LLM-Loop. Thereby, a vague scenario will be given along with a couple of metrics to guide it. These metrics are, for instance, only to recommend using existing user interface elements in a data exchangeable parsable format. If no new step is possible, the process either goes backwards

[2] https://de.wikihow.com.

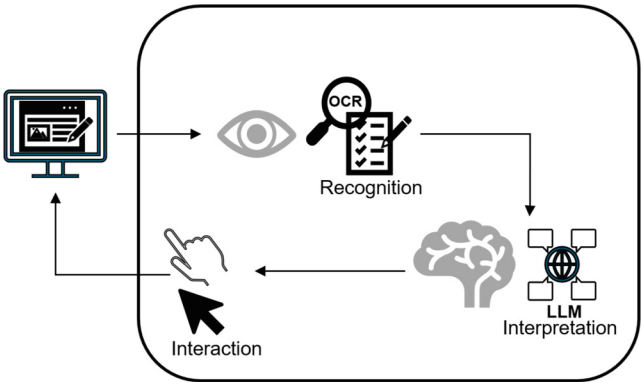

Fig. 3. Testing Loop: Replacing Human Recognition, Interaction, and Interpretation with Optical Character Recognition (OCR), Large Language Model (LLM) and simulated interaction.

or restarts the interaction process from the initial entry point with a modified starting conditions. In addition, we asked the simulated user to try out wrong inputs, such as special characters, leaving fields empty or simply some other values. Furthermore, some human characterizations can be specified, depending on the simulated user's knowledge base and the testing scenario. In parallel, the control loop will track time, error, and usability, based on some of the Nielsen's heuristics, NASA-TLX and System Usability Scale by Brooke [2,9].

The LLM interprets the scenario requirements and suggests one action needed to achieve the next step of the specific objectives, such as completing a form, clicking a button or navigating to a different view. This action will then be performed by an input. Since this step is not possible, when the LLM hallucinates, there needs to be some preventive measures [16].

Simulated user interactions are executed programmatically using controlled input devices like mouse and keyboard emulators or, for instance, physical robots that interact directly with the hardware device. Either way, these devices replicate user actions such as clicking, typing, and scrolling, incorporating human-like variability, such as random delays and slight inaccuracies in pointer movements. This realistic behavior helps evaluate interface responsiveness and identify usability issues. In addition, a moving UI can be simulated, like we would expect in a moving environment.

Usability and performance metrics are evaluated based on task completion time, error rates, and interaction complexity. By simulating different user profiles, such as novices and experts, the system identifies potential bottlenecks, cognitive load challenges, and areas for improvement. For instance, novice simulations focus on learnability, while expert scenarios assess efficiency under high-stress conditions. These insights are valuable for improving interface design and ensuring software adaptability to a wide range of user needs.

3.1 Implementation

The implementation of this concept leverages various tools and libraries, primarily utilizing the Python programming language[3] for its variety of extensions regarding LLMs and AI-APIs. The project is managed with Python Poetry[4], a dependency and environment management tool that ensures reproducibility and simplifies dependency resolution. Furthermore, snapshots of the graphical user interface are taken utilizing libraries suited to the operating system. For instance, `Pillow`[5] is used for image processing, while `pyautogui`[6] facilitates screenshot capturing and interaction automation. To manipulate the position of application windows to simulate user interface movements, `pygetwindow`[7] is employed.

For text recognition within graphical user interfaces, OCR is implemented using the `pytesseract`[8] library, which serves as a Python wrapper for Tesseract OCR, and `easyocr`[9], a deep learning-based OCR solution. These libraries enable the identification of textual elements in captured screenshots. Simulated cursor movements are achieved using `Human Cursor`[10], a Python library that generates realistic mouse movements to emulate user interactions. To interface with LLMs, the implementation utilizes the `Ollama Python Library`[11] and the `OpenAI Python API library`[12], providing seamless interaction with LLMs such as in this case used GPT-4.0[13] and Llama 3.3[14].

Finally, some basic tools from the `Python Standard Library`[15] to track time, convert formats and processing data were used. With these tools, a first version of the concept was implemented. A simplified activity sketch can be found in the following Fig. 4.

4 Results Using Our Demonstrator

To get an idea of how accurately the concept performs, we tested a first implementation with our Aircraft Departure Optimizer (ADO; Example view in Fig. 5). It is an Electronic Flight Bag (EFB) application, which is made to be used in the cockpit before takeoff. It focuses on departure phase optimization, particularly for noise-sensitive areas around airports. It aids pilots in determining the optimal climb rate, thrust settings, and departure trajectory to achieve regulatory noise thresholds while maintaining operational efficiency.

[3] https://www.python.org.
[4] https://python-poetry.org.
[5] https://python-pillow.org.
[6] https://pyautogui.readthedocs.io.
[7] https://pygetwindow.readthedocs.io.
[8] https://github.com/madmaze/pytesseract.
[9] https://github.com/JaidedAI/EasyOCR.
[10] https://github.com/riflosnake/HumanCursor.
[11] https://github.com/ollama/ollama-python.
[12] https://github.com/openai/openai-python.
[13] https://openai.com/research/gpt-4.
[14] https://ollama.com/library/llama3.3.
[15] https://docs.python.org/3/library/index.html.

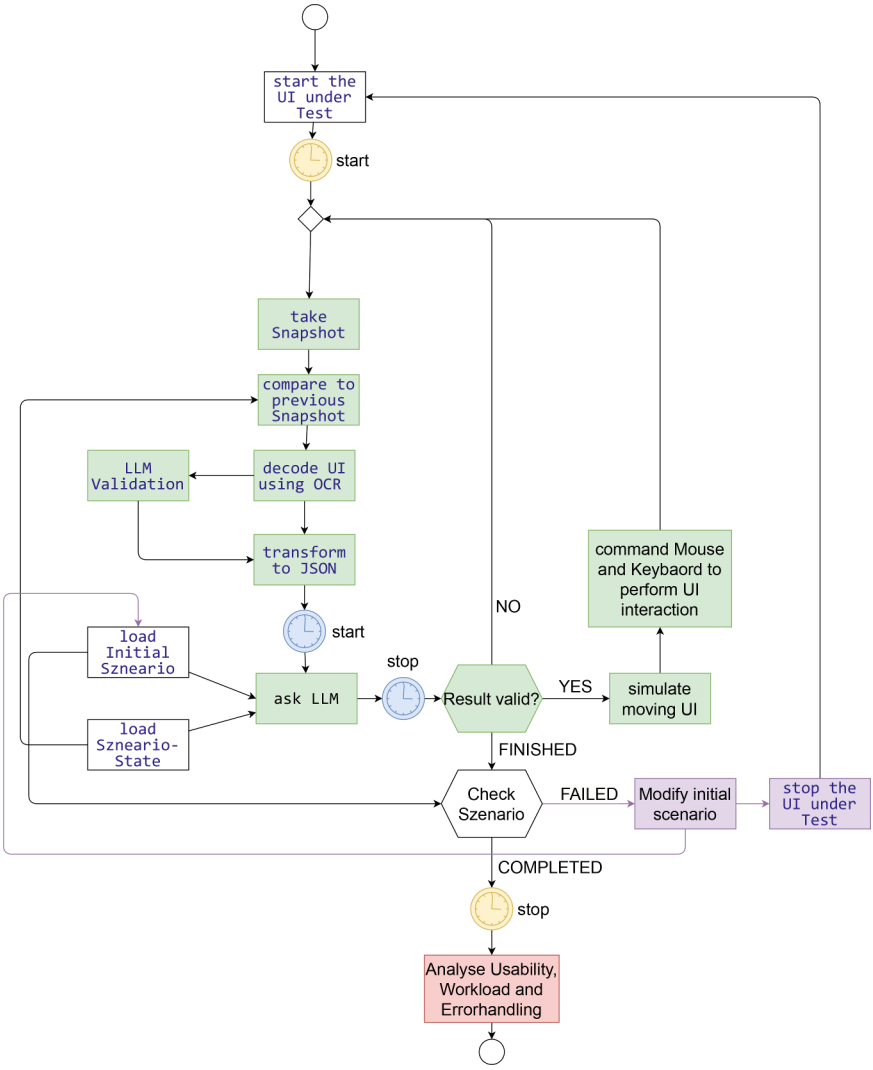

Fig. 4. Implementation Procedure of the Simulated User.

Our Scenario was to get an optimized result where seven different inputs during the application's runtime were needed, in a moving environment, such as a taxing aircraft. Due to the moving environment, we simulated various abrupt changes in hand acceleration and curvature [28]. Furthermore, a heuristic for identifying errors and usability should be pursued. The LLM, were prompted with a simple starting prompt along to the existing elements, in form of a JSON[16] object as this one:

[16] https://www.json.org.

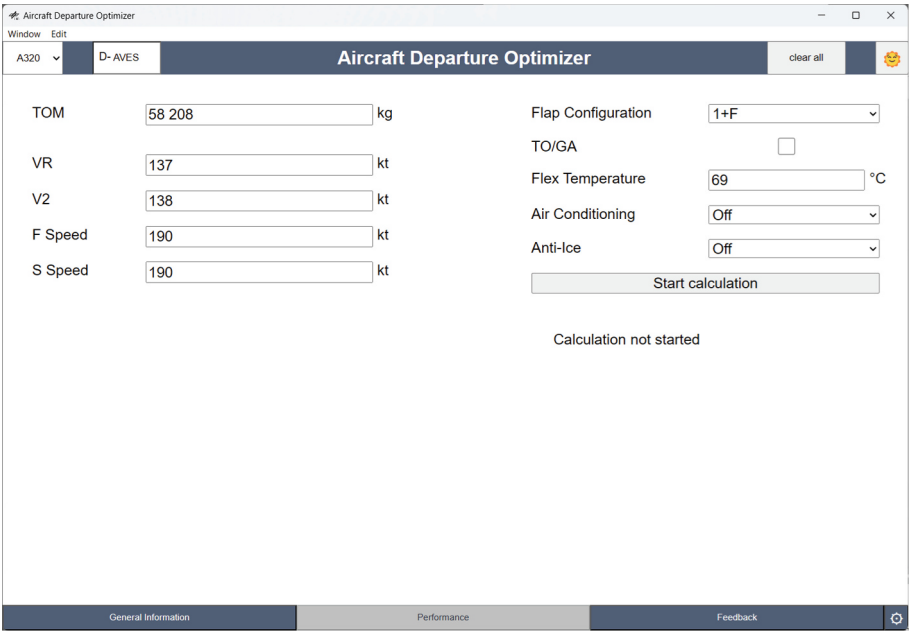

Fig. 5. Aircraft Departure Optimizer (ADO) developed by the DLR: An example of an Electronic Flight Bag (EFB) Departure Noise Assistance Application in early Development Stage.

```
You are a pilot user, as declared in the previous dialog, and
    you want to resolve the following task. Thereby, you are
    not allowed to hallucinate or invent content. If you
   have issues, please state it. The graphical user
   interface is described in the following "elements" JSON
   Object. How to use the existing Options to pursue the
   following goal. Please keep in mind, you may need
   multiple steps to conclude the goal. If this is the case,
    return one step and wait for the new state. Please
   answer in JSON by stating the type, position, function
   and value.

{Elements.json}
Scenario: Please change the values accordingly to
- Aircraft type: A320Neo
- Runway: 18
- TOW: 63t (55 +8(4,4))
...
```

```
At the end of the interaction you want an Acceleration
   Height, Cutback Height and the number of awakenings it
   calculated.
```

Listing 1.1. Example Prompt to the LLM

Although initial tests came up with promising results, due to the hallucination problems of LLMs in general, we ended up using a dedicated OCR-based algorithm for the perception of the UI. Therefore, we first transcribed the user interface using an OCR to a computer readable format and added some element detection libraries. We then prompted the results along with the snapshot to a graphical usage LLM to ensure our algorithm was able to detect all elements. This ensured to be able to reduce the task to a simpler one without interfering with the text loop. Tasks that were too complex produced too much information, without a distinction of their relevance for the decision, where first attempts failed, stopped by a simple watcher and modified by randomly reducing the options to a smaller group. If the proposed options were not helpful, another set would be proposed. As an example, an excerpt of the completely self generated result based on the snapshot from Fig. 5 is given in the following Listing 1.2:

```
{
  "elements": [
...
    {
      "type": "input",
      "position": [10, 100],
      "function": "Takeoff Mass",
      "value": "58208 kg"
    },
    {
      "type": "checkbox",
      "position": [400, 130],
      "function": "TO/GA Selection",
      "value": "Unchecked"
    },
    {
      "type": "dropdown",
      "position": [400, 190],
      "function": "Air Conditioning",
      "value": "Off"
    },
....
  ]
}
```

Listing 1.2. Example of some detected Elements in JSON.

With this input, the LLM was able to resolve partially the task with the following answer. Thereby, it should be noted that the first two results were not sufficient, since a result element was not generated. Instead, the loop corrected

the task, to a smaller task, by temporarily removing some options. In addition, we stated multiple times that no hallucinations etc. were allowed. Each time, the LLM was asked what it planned to perform, until the result was parsable.

```
"result": [
{
  "element": "Takeoff Mass",
  "position": [10, 100],
  "task": "write",
  "value": "63000"
},
```

Listing 1.3. Example Result of the LLM.

After 25 iterations and $\sim$ 2 minutes, the simulated user was able to resolve the task. Since this performance was not good enough, we modified the input multiple times and tried to interpret the issues as well. Thereby, we found out that the more background knowledge we gave, the better the results were and fewer errors were made. With more background knowledge, such as the application description of its goal and general knowledge about departure performance calculations, the algorithm could accomplish the task in 12 loops. However, the hallucination of available buttons increased. As a countermeasure, we primarily stated that the user would be an experienced pilot, by providing some details about how an experienced pilot would iterate, and that it should highlight issues, and recommend solutions. To evaluate the usability, we modified the inputs, according to the knowledge base of the user, and tried multiple scenarios. Furthermore, our error tests with the simulated user were able to detect that one input field didn't handle a false input error correctly.

To get a first feedback on the human factors, we used the System Usability Scale (SUS) by John Brook for usability and the NASA-TLX for workload. We then simply analyzed the log for all scenarios for specific keywords to get workload and compared them with the measured time and the necessary iterations. For example, if an interaction failed due to the moving environment, the physical demand and frustration got higher, but the mental demand got lower. When an approach was unfeasible, the mental and temporal demand, along with the frustration, increased and the performance downshifted. Since the SUS is even more generalized, we asked the simulated user itself. Finally, we tried to get a first idea of how well the test application performed using the reduced set of Nielsen's heuristics [22,23]. Therefore, we simply tracked times and interpreted the created log during the usage. Depending on how often an issue for a scenario encountered, we rated the heuristic worse. In three scenarios, we tracked heuristics, where we simply summarized the max results. The results are indicated by the dots in Table 1.

In addition, we were able to detect that most buttons were a bit too small to be performed in a moving environment where the curvature and acceleration of the simulated mouse gesture abrupt vary. Within 133 tested mouse inputs, 12.7% failed.

Table 1. First Approach for a simulated Usage rated by the Nielsen Heuristic. Higher numbers indicate worse Performances (• represents the simulated User, × represents the real User Group.).

Heuristic	0	1	2	3	4
1. Visibility of System Status	×	•			
2. User Control and Freedom		×•			
5. Error Prevention		•	×		
6. Recognition Rather than Recall		×	•		
9. Help Users Recognize, Diagnose, and Recover from Errors	×		•		

4.1 Comparison to Real User Studies

To get an idea of how well this concept performed, we as well conducted a usability study with 11 appropriate type-rated pilots in a simulator study. The participants had in average 3448 flight hours on the specific aircraft type, where 8 of them were male and 2 female. All participants were asked to perform the same scenarios in the same setting. Thereby, we asked them to solve 3 different tasks and to mention errors they encountered. Afterwards, we asked them also to rate the usage with some general questions, the SUS for usability and the NASA-TLX for workload and analyzed their usage with the Nielsens heurisitc scale to get an insight where possible issues were.

In general, the human users were much slower than the simulated user and needed more time to process changes. In addition, the learning effect, when two scenarios were to be performed in direct succession, was faster for both user types. This indicates that the simulated user may be able to reproduce the learning effects of the human user.

Regarding the error detection, the results were notably different. We detected four issues with the simulated user that were not detected by the human participants. On the other hand, four human-participants (Participant 3, 4, 7, 9) detected issues that were not detected by the algorithm. Interestingly, participant 3 was the first to encounter the same false input error issue. The other ones (specially Participant 1, 2 and 5, 9, 10) didn't encounter this issue. Some errors, such as an input field could not be empty (either 0 or any other number), were not discovered by this approach. Furthermore, some participants stated the usage of this application in a moving environment, due to the element heights could be difficult (Participant 4, 7, 8, 11), which was as well detected by the simulated user.

Regarding the cognitive load and usability, the NASA TLX and the SUS were tracked. The exact results for the eleven human participants and the one simulated user of the NASA-TLX can be derived from the Fig. 6. Since we didn't evaluate multiple simulated users, there is no error bar to indicate. The results indicate an overall low workload, with low demands (Mental: 3,27; Physical: 2,09; Temporal: 2,63) and frustration (Frustration: 1,45). The effort and performance

were equally low, even though a bit higher (Effort: 3,72, Performance: 1,01). The results of the simulated user of the NASA-TLX results were, in general, higher. The frustration and performance was much worse (Frustration: +7,21, Performance: +3,24). On the other hand, the temporal and physical demand were slightly smaller.

The SUS can support these results, since the usability was rated in average around 92.5%. In comparison, the SUS average of the simulated users was 80%. The adapted Nielsen's heuristics were as well analyzed. In Table 1 we marked by × the results stemming from the user survey. Besides the error prevention, the simulated user rating always resulted in a same or slightly worse result. Nevertheless, the heuristic "Help Users Recognize, Diagnose, and Recover from Errors" differs significantly. The human users were able to fulfill their task. When the simulated user interacted with the user interface, some errors were not processable. Therefore, the rating is worse.

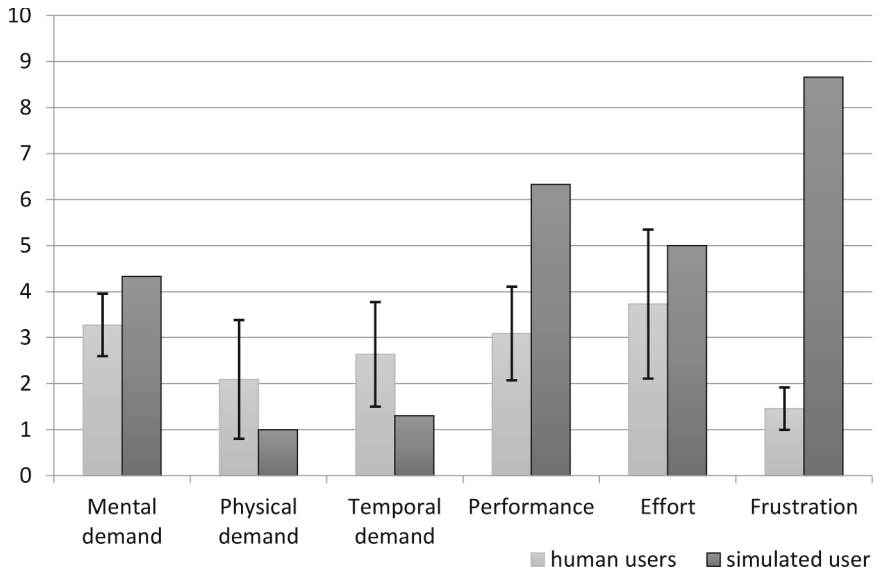

Fig. 6. Raw NASA-TLX Results of the comparison user study.

5 Discussion and Conclusion

This work showed a first example on how a simulated user for UI testing could be realized using LLM, control mechanisms and OCR. Therefore, we implemented a first version and compared the results with some real study results. Thereby, we were able to get comparable results regarding error detection and usability, by the SUS. The workload differed significantly in some respects, e.g., the frustration

and performance, but in general produced comparable but slightly lower results. With these results, a discussion upon portability and applicability is necessary.

The evaluation of gesture behavior presents challenges as it is difficult to detect through traditional automated methods, though it remains theoretically at least performable. The methodology faces limitations in capturing the nuanced user experience, which seems necessary for usability evaluation. A critical comparison emerges between the use of a domain-specific LLM (trained with simpler and more targeted knowledge relevant to the application domain, including standards and states of knowledge) and generalized LLMs. While a specialized LLM could improve domain-specific accuracy, it necessitates significant additional training efforts.

The OCR-based detection of GUI elements works adequately for text-based components but struggles with recognizing graphical or complex interface elements, especially those that require physical interaction, such as buttons or sliders. The integration of image recognition with artificial intelligence alongside OCR shows potential, but its effectiveness needs further research. Thereby, error handling remains a significant challenge as it is application-specific, making generalizable solutions difficult to achieve.

The current approach demonstrated the capability to detect usability errors, but there remains uncertainty regarding the scope and completeness of the detected errors. The lack of insight into the number of missed errors highlights a critical gap that needs addressing for broader applicability. A comparison involving one simulated user to eleven human participants in three scenarios highlights a notable difference in findings, underlining the limitations of this approach in fully replacing human users. Nevertheless, there were some accessibility issues that could have been avoided by using the simulated user early on.

Furthermore, the inability to employ a between-subject design complicates the evaluation, as the differences between the groups of participants make direct comparisons unreliable. Metrics, such as the NASA-TLX for workload evaluation and subjective usability scales like SUS were applied, but defining clear benchmarks for usability and workload remains a complex issue requiring further validation, since the used metrics are not validated for this use case. In addition, we assumed some simulation issues as an influence on the workload, which needs to be further investigated. These limitations underscore the necessity of refining the approach and looking further into the results and metrics.

5.1 Conclusion

Approximately 80% of usability issues are typically identified by the first few human probands in traditional usability testing [24]. This raises whether such issues could be reduced or preemptively identified through an automated simulation approach. The LLM based simulated user framework appears especially effective for highly declarative GUIs, such as those in safety-critical domains, where explicit and structured interaction paths are prevalent. However, the approach has limited applicability to more complex or non-declarative interfaces.

The initial results are promising, offering insights into the potential to simulate and detect simple usability issues and get an initial user experience feedback. Despite these promising results, the framework has notable limitations that warrant further research.

While this concept cannot fully replace human participants, it can serve as a valuable complementary tool. It enables an early analysis of usability issues and raises awareness of overlooked design aspects before human trials. Validating this methodology remains a necessary step, particularly to confirm its effectiveness and its scalability for broader contexts.

References

1. Baron, J.: Thinking and Deciding. Cambridge University Press, 4 edn., October 2006. https://doi.org/10.1017/CBO9780511840265, https://www.cambridge.org/core/product/identifier/9780511840265/type/book
2. Brooke, J.: SUS: A quick and dirty usability scale. Usability Eval. Ind. **189**, November 1995
3. Deng, Y., Xia, C.S., Peng, H., Yang, C., Zhang, L.: Large Language Models are Zero-Shot Fuzzers: Fuzzing Deep-Learning Libraries via Large Language Models, March 2023. https://doi.org/10.48550/arXiv.2212.14834, http://arxiv.org/abs/2212.14834, arXiv:2212.14834 [cs]
4. Diehl, E., Sterman, J.D.: Effects of feedback complexity on dynamic decision making. Organ. Behav. Hum. Decis. Process. **62**(2), 198–215 (1995). https://doi.org/10.1006/obhd.1995.1043, https://www.sciencedirect.com/science/article/pii/S0749597885710436
5. Fellows, L.K.: The cognitive neuroscience of human decision making: a review and conceptual framework. Behav. Cogn. Neurosci. Rev. **3**(3), 159–172 (2004) https://doi.org/10.1177/1534582304273251, https://doi.org/10.1177/1534582304273251, publisher: SAGE Publications
6. Gonzalez, C.: Building human-like artificial agents: a general cognitive algorithm for emulating human decision-making in dynamic environments. Perspect. Psychol. Sci. **19**(5), 860–873 (2024) https://doi.org/10.1177/17456916231196766, publisher: SAGE Publications Inc
7. Gonzalez, C., Fakhari, P., Busemeyer, J.: Dynamic decision making: learning processes and new research directions. Hum. Factors **59**(5), 713–721 (2017) https://doi.org/10.1177/0018720817710347, https://doi.org/10.1177/0018720817710347, publisher: SAGE Publications Inc
8. Hadi, M.U., et al.: A Survey on Large Language Models: Applications, Challenges, Limitations, and Practical Usage. TechRxiv, July 2023. https://doi.org/10.36227/techrxiv.23589741.v1, https://www.authorea.com/doi/full/10.36227/techrxiv.23589741.v1?commit=b1cb46f5b0f749cf5f2f33806f7c124904c14967
9. Hart, S.G., Staveland, L.E.: Development of NASA-TLX (Task Load Index): results of empirical and theoretical research. In: Advances in Psychology, vol. 52, pp. 139–183. Elsevier (1988). https://doi.org/10.1016/S0166-4115(08)62386-9, https://linkinghub.elsevier.com/retrieve/pii/S0166411508623869
10. Hong, W., et al.: CogAgent: a visual language model for GUI agents. In: 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 14281–14290. IEEE, Seattle, June 2024. https://doi.org/10.1109/CVPR52733.2024.01354, https://ieeexplore.ieee.org/document/10655402/

11. Hornbæk, K.: Current practice in measuring usability: Challenges to usability studies and research. Int. J. Hum Comput Stud. **64**(2), 79–102 (2006). https://doi.org/10.1016/j.ijhcs.2005.06.002, https://www.sciencedirect.com/science/article/pii/S1071581905001138
12. Kelso, J.A.S.: Human Motor Behavior: An Introduction. Psychology Press, New York, May 2014. https://doi.org/10.4324/9781315802794
13. Kim, G., Baldi, P., McAleer, S.: Language Models can Solve Computer Tasks. Advances in Neural Information Processing Systems **36**, 39648–39677 (2023). https://proceedings.neurips.cc/paper_files/paper/2023/hash/7cc1005ec73cfbaac9fa21192b622507-Abstract-Conference.html
14. Lee, S., et al.: MobileGPT: augmenting LLM with human-like app memory for mobile task automation. In: Proceedings of the 30th Annual International Conference on Mobile Computing and Networking, ACM MobiCom '24, pp. 1119–1133. Association for Computing Machinery, New York (2024). https://doi.org/10.1145/3636534.3690682, https://dl.acm.org/doi/10.1145/3636534.3690682
15. Lehman, M.M.: Feedback in the software evolution process. Inf. Softw. Technol. **38**(11), 681–686 (1996). https://doi.org/10.1016/0950-5849(96)01121-4, https://www.sciencedirect.com/science/article/pii/0950584996011214
16. Lin, Z., Guan, S., Zhang, W., Zhang, H., Li, Y., Zhang, H.: Towards trustworthy LLMs: a review on debiasing and dehallucinating in large language models. Artif. Intell. Rev. **57**(9), 243 (2024). https://doi.org/10.1007/s10462-024-10896-y
17. Liu, Z., et al.: Make LLM a testing expert: bringing human-like interaction to mobile GUI testing via functionality-aware decisions. In: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering, ICSE '24, pp. 1–13. Association for Computing Machinery, New York, April 2024. https://doi.org/10.1145/3597503.3639180, https://dl.acm.org/doi/10.1145/3597503.3639180
18. MacKenzie, S.: Motor behavior models for human-computer-interaction. In: Carroll, J.M. (ed.) HCI Models, Theories, and Frameworks: Toward a Multidisciplinary Science, pp. 27–54. Elsevier, May 2003, google-Books-ID: gGyEOjkdpbYC
19. Miller, N.L., Crowson, J.J., Narkevicius, J.M.: Human Characteristics and Measures in System Design. In: Booher, H.R. (ed.) Handbook of Human Systems Integration. John Wiley and Sons (2003)
20. Mori, S., Suen, C., Yamamoto, K.: Historical review of OCR research and development. Proc. IEEE **80**(7), 1029–1058 (1992). https://doi.org/10.1109/5.156468, https://ieeexplore.ieee.org/abstract/document/156468, conference Name: Proceedings of the IEEE
21. Nguyen, T.T.H., Jatowt, A., Coustaty, M., Doucet, A.: Survey of post-ocr processing approaches. ACM Comput. Surv. **54**(6), 124:1–124:37 (2021). https://doi.org/10.1145/3453476, https://dl.acm.org/doi/10.1145/3453476
22. Nielsen, J.: Finding usability problems through heuristic evaluation. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '92, pp. 373–380. Association for Computing Machinery, New York, June 1992. https://doi.org/10.1145/142750.142834, https://dl.acm.org/doi/10.1145/142750.142834
23. Nielsen, J.: Enhancing the explanatory power of usability heuristics. In: CHI '94: Proceedings of the SIGCHI Conference on Human Factors in Computing System, CHI '94, pp. 152–158. ACM (1994). https://doi.org/10.1145/191666.191729
24. Parnell, K.J., Banks, V.A., Wynne, R.A., Stanton, N.A., Plant, K.L.: Human Factors on the Flight Deck: A Practical Guide for Design, Modelling and Evaluation. CRC Press, Boca Raton, May 2023. https://doi.org/10.1201/9781003384465

25. Smith, R.: An overview of the tesseract OCR engine. In: Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), vol. 2, pp. 629–633 (2007). https://doi.org/10.1109/ICDAR.2007.4376991, https://ieeexplore.ieee.org/abstract/document/4376991, iSSN: 2379-2140
26. Wang, J., Huang, Y., Chen, C., Liu, Z., Wang, S., Wang, Q.: Software testing with large language models: survey, landscape, and vision. IEEE Trans. Software Eng. **50**(4), 911–936 (2024). https://doi.org/10.1109/TSE.2024.3368208, https://ieeexplore.ieee.org/abstract/document/10440574, conference Name: IEEE Transactions on Software Engineering
27. Wang, L., et al.: User Behavior Simulation with Large Language Model based Agents, February 2024. https://doi.org/10.48550/arXiv.2306.02552, http://arxiv.org/abs/2306.02552, arXiv:2306.02552 [cs]
28. Watkins, K.S., Rose, K.A.: Simulating individual-based movement in dynamic environments. Ecol. Model. **356**, 59–72 (2017). https://doi.org/10.1016/j.ecolmodel.2017.03.025, https://www.sciencedirect.com/science/article/pii/S0304380016307888
29. Xia, C.S., Zhang, L.: Less training, more repairing please: revisiting automated program repair via zero-shot learning. In: Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp. 959–971, November 2022. https://doi.org/10.1145/3540250.3549101, http://arxiv.org/abs/2207.08281, arXiv:2207.08281 [cs]
30. Xiao, X., Wang, X., Cao, Z., Wang, H., Gao, P.: IconIntent: automatic identification of sensitive UI widgets based on icon classification for android apps. In: 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE), pp. 257–268, May 2019. https://doi.org/10.1109/ICSE.2019.00041, https://ieeexplore.ieee.org/abstract/document/8812108, iSSN: 1558-1225
31. Zhang, D., et al.: Mobile-Env: building qualified evaluation benchmarks for LLM-GUI Interaction, June 2024. https://doi.org/10.48550/arXiv.2305.08144, http://arxiv.org/abs/2305.08144, arXiv:2305.08144 [cs]
32. Zhang, E., Wang, X., Gong, P., Lin, Y., Mao, J.: USimAgent: large language models for simulating search users. In: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, pp. 2687–2692. Association for Computing Machinery, New York, July 2024. https://doi.org/10.1145/3626772.3657963, https://dl.acm.org/doi/10.1145/3626772.3657963