KLM-GOMS Detection of Interaction Patterns Through the Execution of Unplanned Tasks

Daniel Cunha[1], Rui P. Duarte[1,2(✉)], and Carlos A. Cunha[1,2]

[1] Polytechnic of Viseu, Viseu, Portugal
{estgv15244,pduarte,cacunha}@estgv.ipv.pt
[2] CISeD – Research Centre in Digital Services, Viseu, Portugal

Abstract. The availability of software applications has contributed to the increase in user demand, which has increased the complexity of these applications. This contributed to the adoption of automation mechanisms for the software testing process, in order to reduce coding errors and shorten the time needed to deploy a new version of the application to the user. Currently, automating the application testing process is a well-established reality and supported by many tools. However, the usability evaluation of an application requires solutions that allow to determine, in advance, the type of improvements that may be necessary in the application without the need for intensive user testing. This work deals with the automatic analysis of the impact on the user of changes in the design of an application, through the implementation of the Keystroke Level Model (KLM). Based on the execution of unplanned user interactions in a web interface, a KLM string is obtained and evaluated, providing a model that converts KLM operators and the execution time of each operator into information for designers. Moreover, performance indicators are obtained and interaction patterns are automatically defined.

Keywords: Keystroke level model · Human-centered computing · Software testing · Interaction design · User interface design

1 Introduction

Graphical User Interfaces (GUI) are available in most modern applications and represent the primary connection between the software and its users. GUI display relevant information and possible actions to users through virtual objects, or widgets, which are graphic elements such as buttons, text or edit boxes that make it intuitive to use software [1,2]. Moreover, an average of 48% of the application source code is used to implement a GUI in order to release a user-friendly software [3], and the user interaction with the software is established by performing sequences of events.

At the user level, the demand for applications is increasing and the interface between the user and the system plays an important role in the acceptance of

© Springer Nature Switzerland AG 2021
O. Gervasi et al. (Eds.): ICCSA 2021, LNCS 12950, pp. 203–219, 2021.
https://doi.org/10.1007/978-3-030-86960-1_15

the systems, as well as user's trust and satisfaction [4]. This demand focuses on usability, performance, quality of service and interface design [5]. When the interface is well understood by the users, a win-win relation is established between the user and the system; otherwise, users tend to become frustrated and more intolerant to errors. According to Hogan, L. [6], most users leave web applications when the loading wait time exceeds 2 s. In turn, when the page load time exceeds 3 s, about 40% of users tend to abandon the application.

Interface design is one of the most critical tasks in User Experience (UX) and User Interface (UI) analysis. It improves UX making the interface more intuitive and simple. That said, it is important to perform UI/UX analysis to predict the user behaviour, and, thus reduce the time spent on each interaction. To achieve this, two approaches can be considered: empirical and analytic. While the first involve users, the second uses techniques to predict user behavior. Predictive models use normalized historical data from several sources, which are processed to obtain an analysis. Example of this are the family of predictive models to compare and evaluate Goals, Operators, Methods and Selection (GOMS) of skilled, error-free performances [7]. These models are focused on the hierarchy of tasks composed of actions and cognitive operators. The simplest model of the GOMS family is the Keystroke Level Model (KLM), which predicts the execution time of a task using mouse and keyboard as its main inputs [8].

One of the current challenges in interface design is to help UI designers to improve their designs without the need to set-up a number of user testing sessions. Nevertheless this is an important part of the design process, it is a time consuming process to produce results. Moreover, UI designers make many small decisions along the way that can impact on the user interaction, and need tools that allow them to measure the quality of their design. According to Al-Megren et al. [9], it is of most importance to create methods that use the KLM model to provide guidance to introduce changes in their design and reduce the time required to complete a task. In this paper we present a KLM approach that uses the interface created by the UI designer to provide feedback about the quality of the design and proposes performance indicators on how to improve the quality of the interaction. To achieve this, we propose an algorithm based on the KLM that analyses a KLM string to detect types of interaction and, from there, identify patterns of behavior that are increasing the user task execution time, resulting in a poor interaction. The major contributions of this paper are a model of interpretation of unplanned actions by users, and performance indicators for the automatic detection of interaction patterns.

The remainder of this paper is organized in the following manner. Section 2 presents the background on KLM by identifying its applications and current trends. Section 3 presents a general view of the approach and the two main modules developed. Section 4 refers to the major conclusions that can be obtained, and provides insights on future work.

2 Background

The KLM model has been extensively used to evaluate time performance of user-computer interaction across different devices. In this section we review the KLM and its applications to related contexts. For further reading, Al-Megren et al. [9] present a survey of the KLM and recommendations are provided.

2.1 Keystroke Level Model

The GOMS model can be defined as an high-level description of the knowledge that a user has to have in order to execute tasks. It relates to the mental processing and hierarchical decomposition of predefined goals. It is a strong reference in the context of behavior analysis in what concerns the prediction of the effective usability of UIs [7].

The KLM models the decomposition of a general task obtained from the GOMS model into unit tasks, at the device level (like mouse clicks or key press). It allows the prediction of user performance, by measuring the time required for a user to complete a task. Proposed by Card et al. [8], a unit task has two parts: *acquisition* of the task and the *execution* of the task. The total time to complete a unit task is given by:

$$T_{task} = T_{acquire} + T_{execute} \tag{1}$$

At the execution level, KLM provides physical, mental and response operators with predefined time values. These operators are defined by a letter and include: K (keystroke), P (point), H (home), D (draw), B (button press), M (mental preparation for action), and R (system response). According to Card et al. [8], K is the mostly frequently used operator and represents the act of keystroke or button press (when considering $K = 0.2\,\text{s}$). P represents pointing to a specific target with a mouse. It's time varies as a function of the distance to the target, and the size of the target, according to Fitts's Law [10]. H is the action of switching the hand between keyboard and mouse. D is associated to the mouse and refers to the action of drawing with it.

To execute the physical operators, the user has to mentally prepare himself for this execution. This is represented by the mental operator, M. Since this operator is not observable from user behaviour, it is based on specific knowledge of user skills and the placement of M's is governed by a set of heuristics [8].

The final operator, R refers to the time required for the system to respond to user's actions and is typically neglected since it is assumed that the system responds in real-time. The KLM process begins with an encoding (named KLM string) containing of the physical operators (K, P, H, B, and D). After this, M's are inserted into the string according to Rule 0. Rules 1 to 4 are then applied to each M to determine if it should be deleted.

The execution time is the sum of the time for each of the operators from the final KLM string:

$$T_{execute} = T_K + T_P + T_H + T_D + T_B + T_M + T_R \tag{2}$$

Despite the advantages of using KLM, it presents several limitations. First, only expert users are considered and does not account for user errors. Second, it considers linear routine tasks; however, users can multi-task or reach a goal by different means. Finally, individual differences are not considered, once the M only assumes a single value; mental workload and fatigue are aspects of M that are not accounted by this model. KLM was validated against observed models to determine prediction accuracy. According to Card [7], the root-mean-square percentage error (RMSPE) of the model is 21%.

2.2 Related Work

Traditional methods used in the usability evaluation of an interface fall into two categories: the subjective opinion of users and experts, (mainly applying questionnaires [11,12] and inspection methods [13,14]), or using objective techniques like rules [15], and analytic modeling [16]. Nevertheless these approaches provide important tools to determine the usability of an UI, there is both cost and time needed to implement users' interaction evaluation with and acceptable coverage, increased by the need to use experts to cover for the user's faults.

Automated usability testing of web applications is a trending topic (web approaches cover 32% of the reviewed literature [17]) which presents several advantages when compared to traditional techniques [17,18]. During the last decades, predictive cognitive models, like KLM, have been applied and extended to several application domains, with focus on mobile applications and IVIS systems. Some of the most representative applications of KLM are the CogTool project [19] (which was extended to web forms by the KLM Form Analyser (KLM-FA) [20]) and the SANLab-CM [21]. CogTool [19] is a tool that can produce quantitative, model-based predictions of skilled performance time from tasks demonstrated on storyboard mockups of a user interface (inspired by the DENIM project [22]). Each state if the interface is represented by a frame in the storyboard. With the storyboard created, the modeler defines the steps required to complete a task, and KLM is used to determine the time to complete a task. CogTool Explorer [23] builds upon CogTool to predict a user's goal-directed exploratory interaction with a website. Current available model-based tools require non-trivial manual work to examine forms. To tackle this problem, KLM-FA [20] extends the capabilities of existing tools for practitioners by focusing specifically on automating the analysis of web forms. On one hand they increased the automation and efficiency for evaluation tasks and minimized the required effort from CogTool. On the other hand they do not account with design support and evaluation. They also accounted for the calculation of Fitts' law, instead of using the fixed value. SANLab-CM [21] differs from these approaches since it incorporates variability as a part of its core paradigm and offers a set of tools designed to visualize and compare critical paths and how they differ within and between models. This variability is related to the execution times of an operator, which can cause the path for tasks completion to fluctuate widely.

Models focused on empirical human interaction were also developed. Instead of focusing on predictive behavior, they focus on the cognitive perception of users employing, to some extent, Artificial Intelligent (AI) techniques. These AI based tools are surveyed by Katsanos et al. [24], and cover information structure elaboration and link's appropriateness evaluation of web applications. Some of the most representative are Bloodhound [25], Method for Evaluating Site Architectures (MESA) [26], Automated Cognitive Walkthrough for the Web (ACWW) [27], InfoScent Evaluator Tool (ISEtool) [28], Combining Link Scent with Navigation Path Relevancy (CoLiDeS+) [29], and at the level of conformance to standards, the Usability Evaluation Framework (USEFul) [30].

The use of agents to extract usability metrics has later been criticized [25] since the agents do not consider the content analysis and other human-related aspects. The InfoScent Bloodhound Simulator is presented, which performs automated usability evaluation of websites through the use of information scent, and produces a usability report for the designer to identify navigation problems. This method requires three inputs: the user tasks identified by keywords, the URL address of each task target page, and the URL of the page in which exploration starts. ISEtool [28] presents an approach similar to Bloodhound and is a semi-automated tool to support the evaluation of information scent of hyperlinks' descriptions in webpages. They assumed that users have a goal and their surfing patterns are guided by information scent. Inputs provided by the designer are a description of a user goal, the URL of the webpage and select the LSA semantic similarity algorithm [31]. MESA [26] simulates navigation in a website by modeling the interoperability between the site's structure, link quality, and cognitive limitations. ACWW [27] provides an analysis of webpages for problems in navigation. It identifies four types of usability problems and produces a prediction of the mean total number of clicks for a user to select the correct link. The designer provides three types of inputs: a description of the user goal, labels of the regions of the webpage and text labels of each region. CoLiDeS+ [29] is a cognitive model of web navigation which describes a real-time step-by-step process based on positive correlations between spatial ability, path adequacy, and task performance to allow users to make navigational decisions. As inputs it requires a detailed description of the user's goal, the heading labels of subregions on each webpage, and text labels of the links in each subregion. Results provide a simulation of link selection and backtracking behavior. USEFul [30] is a web-based, nonpublically available prototype that evaluates the usability of a website by employing the Heuristic Evaluation technique which refers to the set of 242 research-based usability guidelines compiled by the author. Nevertheless, it only considers those that are straightforward to implement. Results were compared to the work of Nielsen and Tahir [32], and their framework detected on average 28.5% more violations than the referred work.

The aforementioned tools present several disadvantages. Both Bloodhound and ISEtool do not model visual search and assume a global evaluation of all hyperlinks in a webpage, whereas both ACWW and CoLiDeS+ ask from the designer to manually parse the webpage into sub regions and do not specify

complex web navigation strategies. On the other hand, predictive tools simulate visual perception and provide greater insight into expected user behavior and accurate predictions of user's performance. According to Chi et al. [25] there are two types of tools that perform automated usability evaluation: those that make use of conformance to standards and those that try to predict the usage of an application. The framework proposed in this paper falls in the second category, since it generates performance indicators from a user interaction, and automatically extracts interaction patterns from a KLM string.

3 Unplanned User Interaction Framework

This section presents the process of parsing an unplanned action into user performance indicators, to assist the designer of interfaces. It presents an high level description of the architecture, and a detailed view of its components.

3.1 High-Level Architecture

At this level, a web application is designed and deployed for tests. Next, a user executes unplanned interactions (associated to a goal) in the application, which allow the extraction of performance indicators used to generate interaction patterns. To achieve this, the model is composed of two core components: the parsing module and the analysis module, as illustrated in Fig. 1.

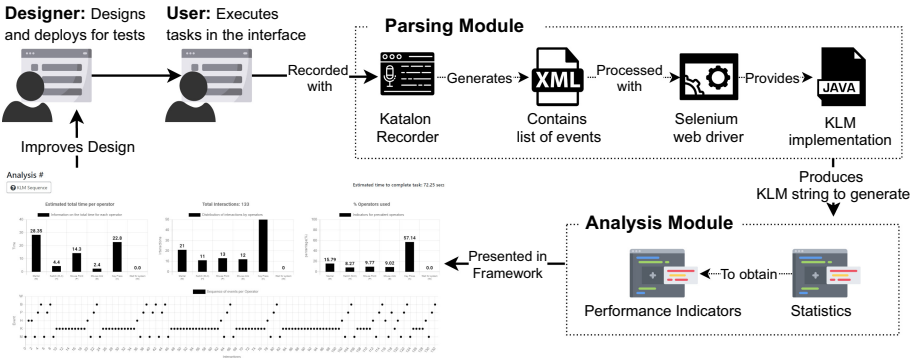


Fig. 1. General view of the framework

The parsing module captures events from the user interaction when executing a task in a web application using Katalon Automation Recorder [33]. This application stores all the interactions in a XML file which is parsed in JAVA using the Selenium WebDriver API [34], to acquire the data from the XML file. Next, the KLM [8] is applied and the corresponding KLM string is generated, resulting in the pre-processing stage of our framework.

The analysis module acts on the KLM string to extract performance indicators related to operators used in the interaction. Since user interaction is unplanned, users can reach a goal of interaction following random paths, and performance indicators are used to automatically determine interaction patterns (mouse based or keyboard based).

3.2 KLM String Extraction

The extraction of the KLM string corresponds to the output of the parsing module. To reach this goal, several steps are considered: record user interaction, process XML file, and generate the KLM string. In the first step, the user interaction is recorded with Katalon Recorder [33]. It is a browser add-on that records actions in a XML file related to elements of web applications. The XML file is composed of a set of tags associated to an event (identified as <*selenese*>) that is composed of the purpose of the action, <*command*>, containing a key or target element in the UI, <*target*>, which is given by a <*value*>. An example is presented in Fig. 2, which refers to the interaction with an email textbox.

```
<selenese>
        <command>click</command>
        <target><![CDATA[name=registrationEmail]]></target>
        <value><![CDATA[]]></value>
</selenese>
<selenese>
        <command>type</command>
        <target><![CDATA[name=registrationEmail]]></target>
        <value><![CDATA[dfghjklsdf@gmail.com]]></value>
</selenese>
```

Fig. 2. Email textbox XML tags: the user clicks the textbox and enters the email

The first <*selenese*> event is associated to the click in the textbox to activate the element, and the second relates to the writing with the keyboard. In the first event, the <*command*> returns the value *click* which corresponds to the user click in the component before writing the email. The <*value*> of the <*target*> is null. In the second event, the value of <*command*> is *type* which refers to the use of keyboard in the web element in the <*target*>. The <*value*> is given by the keys pressed. Once the XML file is generated, the second step initiates. The XML is parsed into the java implementation to extract the tags for KLM processing. This process is carried out with the Selenium WebDriver API [34]. This API provides functions suitable to extract information from the XML file, associate them to KLM operators (see Sect. 2.1), and builds the KLM string. This process is depicted in Fig. 3, and an example of application is described.

Every hierarchical <*selenese*> event is processed to obtain the <*value*> of a specific action. When a user executes a *type* <*command*> in a <*target*> of type *Id*, it is inferred that the user is interacting with a textbox. Next, the

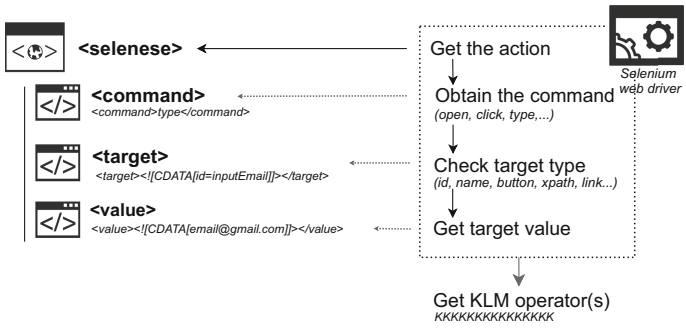


Fig. 3. XML parser: from the XML file to a KLM string

<*value*> *email@gmail.com* is obtained and stored. This <*value*> is of type *text* and is parsed in the KLM as a sequence of 15 operators K (the KLM value of keystroke). This sequence is added to the KLM string and the next <*selenese*> event in the XML file is parsed into the KLM string. When all the events are processed, the pre-final KLM string is created. The final KLM string is generated by applying the rules to insert the mental operator, M, presented in Sect. 2.1.

To validate the accuracy of the KLM implementation, the predictions obtained with the implemented framework were compared with user testing data for the sign-up form of Cable News Network (CNN). Two interaction strategies were considered: (a) mouse based in which a user interacts with the form with a mouse, except for text entries, and (b) keyboard based, in which form fields are reached using the tab key. Ten users were asked to carry out the steps required to sign-up in the aforementioned application. No sequence of steps was defined, except for the goal of the interaction. In the framework, the following assumptions were used: (a) system response time, W, was negligible, (b) the user's hand began on the keyboard before the execution of the task, (c) Fitts' law calculations were enabled. Table 1 presents users' task execution times and the framework predicted times for CNN register form, identified as [S01] in Appendix A, along with the error rate and coefficient of covariation of the RMSE (CV-RMSE).

Table 1. Framework execution times showing means and, in parenthesis, standard deviations

Interaction	Participant's task time (ms)	Framework predicted time (ms)	Error rates of predictions (%)	CV-RMSE of predictions (%)
Keyboard based	24743 (1902.48)	25020	1.12%	7.68%
Mouse based	25679 (3522.02)	29720	15.74%	18.04%

Results show that the mean error of predictions was 8.43%, which is within the 20% margin of error reported by [35]. In general, this framework is an accurate implementation of KLM, which overestimates task time in the mouse based interaction, and underestimates in the keyboard based interaction.

3.3 Performance Indicators and Interaction Patterns

To identify potential problems in a user interaction, the KLM string is parsed into it's three visual representations: the number of operators used in the interaction, the time per operator and the timeline of events. Thus, the KLM string is interpreted as an array, and information is extracted to provide the designer visual indicators of the interaction. Figure 4 depicts an example of transforming a KLM string (*MHMPBMKKKKKKKKKKKKKKKKKKKKKKKHMPBMKKKKKKKK KKKKHMPBMPB*) into readable indicators of user performance.

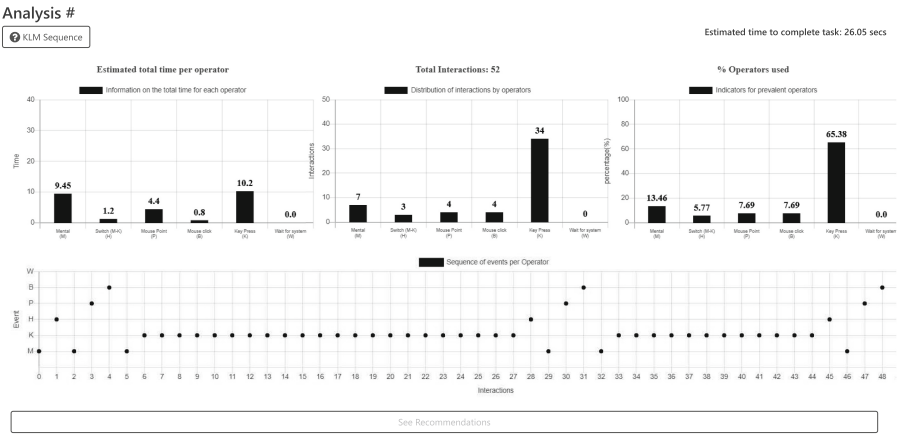


Fig. 4. Conversion of a KLM string into performance indicators for mouse and keyboard interaction

This interaction scenario was based on the task of registering a new user in [S02] using a mixed strategy of interaction (mouse and keyboard). The most prevalent operator is *K*. However, the mental operator, *M*, also consumes 9.45 *secs* of interaction time, which is high, relative to the total time of interaction. This analysis makes possible for the designer to understand that during the interaction the user had to move from the keyboard to the mouse, and again to the keyboard, which lead to an increase in the interaction time. Thus, a message was placed in the interface to the user: "did you know that you can press TAB key to change between fields?". Next, another user was asked to execute the same interaction, with the redesigned interface. As presented in Fig. 5, the user followed a keyboard-based strategy. The KLM string generated by the interaction is *MHMPBMKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKHMPBMPB*, and the performance indicators show that the user reduced the amount on mental and pointing interactions, which lead to a decrease in the total time required to complete a task.

Nevertheless performance indicators provide information of the quality of the interaction, they only provide a view of the task executed by the user. To

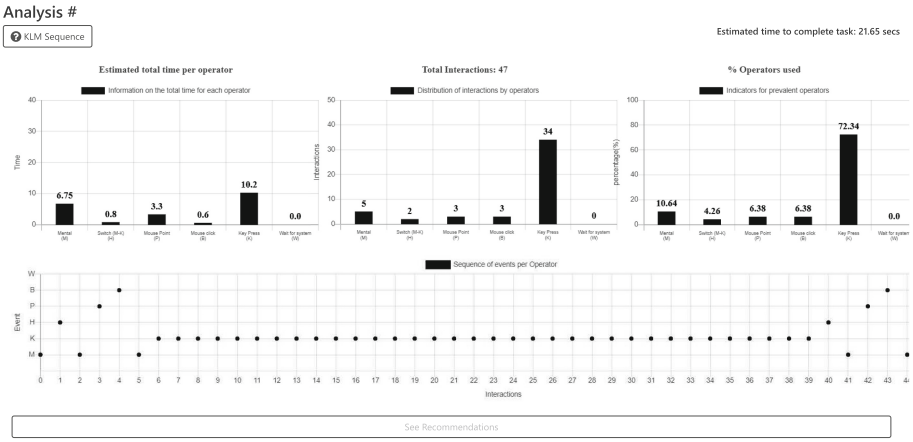


Fig. 5. Conversion of a KLM string into performance indicators for keyboard interaction resulting of changes in the web application

improve on the extraction of features, it is important to automatically detect the type of interaction that the user executed on an web interface. To implement a rule-based method, it is first required to determine patterns from the user interaction strategy. Table 2 and Fig. 6 present the results obtained by applying the interaction strategies to ten web applications (identified in Appendix A).

Results show that the percentage of time used by each operator is related to the interaction strategy. In a mouse-based strategy, the prevalent operators are M and P with 57.44% and 35.81%, in average, of the total time spent in the interaction, respectively. In a keyboard-based interaction, the operators are M (29.01%) and K (50.72%), and in a mixed combination, the prevalent operators are M (37.05%) and K (38.36%). Nevertheless similarities can be found in these two strategies, the increase in M is compensated to a decrease in K, complemented with an increase in H and P. For comparison purposes, the number of operators used (materialized in the percentage of operators) was also considered. Results show that they do not provide a precise indicator to measure the consumption of time in the execution of a task, since many operators do not imply on increase in time (cf. first scenario, where K are used in 64.38% of operators and, in time spent, they nearly correspond to the same time spent as M operator, with only 13.92% of operators used). Considering the patterns, Fig. 6 plots the percentage of time used by each operator in each interaction strategy, and its standard deviation, σ. In order to predict the interval of admissible values of operators for each type of interaction, let $O \in \{M, H, P, B, K\}$ be a general operator resulting from a user interaction, and $O_i, i = 1...n$, the value of the operator presented in Table 2. The rule is to determine $min\{O_i\}$ and $max\{O_i\}$ and verify if $O \in [min_{O_i} - \sigma_O, max_{O_i} + \sigma_O]$. If that occurs, a pattern is established, and the type of interaction is predicted, as presented in Table 3.

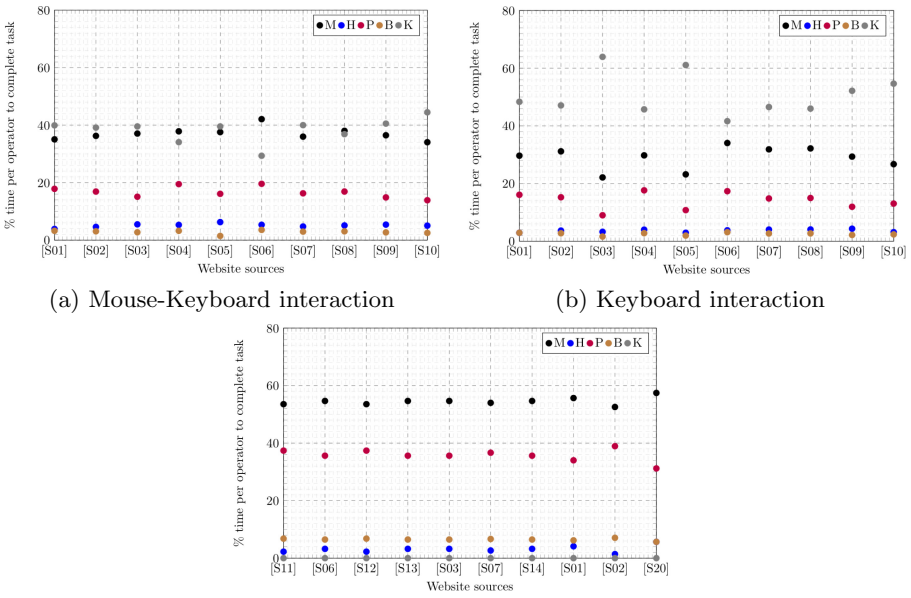
Table 2. Interaction strategies followed in ten sources, and corresponding percentages of time spent and, in brackets, percentage of operators used to complete a task.

	Sources	% of time per operator (% of operators used)				
		M	*H*	*P*	*B*	*K*
Sign-up (mouse & keyboard)	[S01]	35.06 (12.9)	3.89 (4.84)	17.85 (8.06)	3.24 (8.06)	39.93 (66.14)
	[S02]	36.27 (13.46)	4.60 (5.77)	16.89 (7.69)	3.07 (7.69)	39.15 (65.39)
	[S03]	37.08 (13.70)	5.49 (6.85)	15.10 (6.85)	2.74 (6.85)	39.56 (65.75)
	[S04]	37.85 (14.84)	5.31 (7.03)	19.48 (9.38)	3.24 (8.59)	34.09 (60.16)
	[S05]	37.58 (13.77)	5.27 (6.52)	16.11 (7.24)	1.46 (7.25)	39.56 (65.11)
	[S06]	42.09 (17.50)	5.34 (7.50)	19.59 (10.00)	3.56 (10.00)	29.35 (55.00)
	[S07]	36.00 (13.24)	4.74 (5.88)	16.29 (7.35)	2.96 (7.35)	40.00 (66.18)
	[S08]	38.02 (14.47)	5.12 (6.58)	16.90 (7.89)	3.07 (7.89)	36.87 (63.17)
	[S09]	36.48 (13.32)	5.40 (6.67)	14.86 (6.67)	2.70 (6.67)	40.54 (66.67)
	[S10]	34.06 (11.94)	5.04 (5.97)	13.88 (5.97)	2.52 (5.97)	44.47 (70.15)
	Average	**37.05 (13.92)**	**5.02 (6.36)**	**16.70 (7.71)**	**2.86 (7.63)**	**38.36 (64.38)**
Sign-up (keyboard - tab)	[S01]	29.67 (10.00)	2.93 (3.33)	16.11 (6.67)	2.93 (6.67)	48.35 (73.33)
	[S02]	31.17 (10.64)	3.69 (4.26)	15.24 (6.38)	2.77 (6.38)	47.11 (72.34)
	[S03]	22.13 (6.45)	3.27 (3.22)	9.01 (3.23)	1.63 (3.23)	63.93 (83.87)
	[S04]	29.78 (10.28)	4.01 (4.67)	17.65 (7.48)	2.80 (6.54)	45.73 (71.03)
	[S05]	23.19 (6.93)	2.94 (2.97)	10.79 (3.96)	1.96 (3.96)	61.10 (82.18)
	[S06]	34.06 (12.31)	3.78 (4.62)	17.35 (7.69)	3.15 (7.69)	41.64 (67.69)
	[S07]	31.87 (10.94)	4.04 (4.68)	14.83 (6.25)	2.69 (6.25)	46.54 (71.88)
	[S08]	32.19 (11.11)	4.08 (4.76)	14.99 (6.35)	2.72 (6.35)	45.99 (71.43)
	[S09]	29.34 (9.52)	4.34 (4.76)	11.95 (4.76)	2.17 (4.76)	52.17 (76.2)
	[S10]	26.73 (8.47)	3.16 (3.39)	13.06 (5.08)	2.37 (5.08)	54.65 (77.98)
	Average	**29.01 (9.67)**	**3.62 (4.01)**	**14.10 (5.79)**	**2.52 (5.70)**	**50.72 (74.80)**
Navigation (mouse)	[S11]	53.54 (35.00)	2.26 (5.00)	37.39 (30.00)	6.79 (30.00)	0.0 (0.0)
	[S06]	54.65 (35.71)	3.23 (7.14)	35.62 (28.57)	6.47 (28.57)	0.0 (0.0)
	[S12]	53.54 (35.00)	2.26 (5.00)	37.39 (30.00)	6.79 (30.00)	0.0 (0.0)
	[S13]	54.65 (35.71)	3.23 (7.14)	35.62 (28.57)	6.47 (28.57)	0.0 (0.0)
	[S03]	54.65 (35.71)	3.23 (7.14)	35.62 (28.57)	6.47 (28.57)	0.0 (0.0)
	[S07]	54.00 (35.29)	2.66 (5.88)	36.66 (29.41)	6.66 (29.41)	0.0 (0.0)
	[S14]	54.65 (35.71)	3.23 (7.14)	35.62 (28.57)	6.47 (28.57)	0.0 (0.0)
	[S01]	55.67 (36.36)	4.12 (9.09)	34.02 (27.27)	6.18 (27.27)	0.0 (0.0)
	[S02]	52.56 (34.38)	1.41 (3.13)	38.93 (31.25)	7.07 (31.25)	0.0 (0.0)
	[S10]	57.44 (37.5)	5.67 (12.5)	31.20 (25.00)	5.67 (25.00)	0.0 (0.0)
	Average	**57.44 (35.64)**	**3.13 (6.92)**	**35.81 (28.72)**	**6.51 (28.72)**	**0.0 (0.0)**

Table 3. Patterns obtained for each type of interaction

Type of interaction	Operators $\in [min\{O_i\} - \sigma_O, max\{O_i\} + \sigma_O]$				
	M	H	P	B	K
Keyboard	[18.44, 37.76]	[2.44, 4.83]	[6.33, 20.34]	[1.19, 3.60]	[34.82, 70.76]
Mouse	[51.31, 58.71]	[0.31, 6.78]	[29.20, 40.94]	[5.31, 7.44]	[0.0, 0.0]
Mouse and keyboard	[32.02, 44.15]	[3.4, 5.96]	[12.09, 21.39]	[0.92, 4.11]	[25.51, 48.37]

In order to validate these patterns, the three types of interaction were tested in ten different websites. Users were asked to follow one of three interaction strategies: keyboard, mouse, or mouse-keyboard, and the interaction is evaluated to determine if it fits the expected result. Figure 7, depicts the values for the operators within the intervals defined for each pattern of interaction, and the expected results were validated using the patterns defined in Table 3. With this approach, an unplanned interaction from a user can be automatically classified, by measuring the prevalence of operators used within the execution of a task.



(a) Mouse-Keyboard interaction

(b) Keyboard interaction

(c) Mouse interaction

Fig. 6. Sign-up forms used for tests in different scenarios of interaction: (a) $\sigma_M = 2.05$, $\sigma_H = 0.46$, $\sigma_P = 1.79$, $\sigma_B = 0.55$, $\sigma_K = 3.89$; (b) $\sigma_M = 3.69$, $\sigma_H = 0.49$, $\sigma_P = 2.69$, $\sigma_B = 0.45$, $\sigma_K = 6.82$; (c)$\sigma_M = 1.26$, $\sigma_H = 1.1$, $\sigma_P = 2.0$, $\sigma_B = 0.36$, $\sigma_K = 0.0$

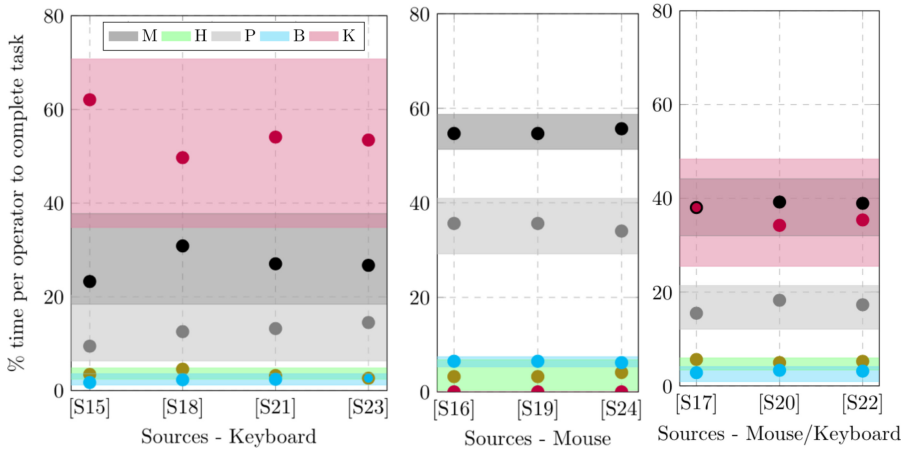


Fig. 7. Representation of scenarios for pattern fitting

4 Conclusions and Future Work

This paper presents a novel approach to assist a designer of interfaces in the quality evaluation of a designed application. It is based on unplanned user interactions that allow the user a greater degree of freedom to complete a task. Most tools available in the literature require pre-processing, for the user to follow a specific path to complete a task. The proposed framework uses KLM model in the calculation of user interaction times with web applications, and the extraction of performance indicators to allow the redesign of interfaces. By simplifying the entry model alluding to the user-oriented task (only the goal of the task to be executed is indicated), the user can execute unplanned actions in the web application in order to achieve the proposed goal. In addition, the type of interaction that the user executes on the web application is automatically detected.

Future work will follow three directions: develop a recommendation system based on the performance indicators extracted from the KLM; capture the timeline of the user facial emotions when executing a task to overlap it with the timeline of the KLM; analyse the impact of the W operator in the overall time required to complete a task. A general purpose recommendation system for designers is a key tool to obtain feedback on interface errors. At this point, this work is able to automatically detect user interactions and provide performance indicators based on patterns of interaction. From these detection, different design guidelines can be provided to designers to improve their UI, based on hybrid recommendation systems [36] that are acquired from performance indicators. Another feature that is planned is the extraction of user emotions in order to be analyze reactions when interacting with the web application to verify if the user's emotions are related to the operators obtained by the KLM. Finally, it is intended to analyze the impact of the W operator of the KLM model on the time of execution of tasks by the user. In the literature, it is suggested that W should be measured. This will allow to identify design patterns that reduce the sensitivity of UX to variations with performance.

Acknowledgements. This work is funded by National Funds through the FCT - Foundation for Science and Technology, I.P., within the scope of the project Ref. UIDB/05583/2020. Furthermore, we would like to thank the Research Centre in Digital Services (CISeD) and the Polytechnic of Viseu for their support.

A List of Sources Used for Tests

Fourteen web applications were used to carry out the tests specified in Sect. 3.3, and are identified in Table 4: (a) mouse interactions only based on navigation; (b) keyboard interactions between fields using the Tab key; or (c) combination of mouse and keyboard interactions. Column (d) identifies web applications used for pattern fitting tests, as presented in Fig. 7.

Table 4. Websites used in tests and corresponding contexts of interaction

Label	Website	(a)	(b)	(c)	(d)
[S01]	CNN (https://edition.cnn.com/)	×	×	×	
[S02]	New York Times (https://www.nytimes.com/)	×	×	×	
[S03]	Walmart (https://www.walmart.com/)	×	×	×	
[S04]	Spotify (https://www.spotify.com/)		×	×	
[S05]	Twitter (https://twitter.com/)		×	×	
[S06]	BBC (https://www.bbc.com/)	×	×	×	
[S07]	Reddit (https://www.reddit.com/)	×	×	×	
[S08]	The Guardian (https://theguardian.com/)		×	×	
[S09]	LinkedIN (https://www.linkedin.com/)		×	×	
[S10]	GitHub (https://github.com/)	×	×	×	
[S11]	Ebay (https://www.ebay.co.uk/)	×			
[S12]	Stack Overflow (https://stackoverflow.com/)	×			
[S13]	Forbes (https://www.forbes.com/)	×			
[S14]	Reuters (https://www.reuters.com/)	×			
[S15]	Farfetch (https://www.farfetch.com/)				×
[S16]	Fox News (https://www.foxnews.com/)				×
[S17]	The Wall Street Journal (https://www.wsj.com/)				×
[S18]	The Telegraph (https://www.telegraph.co.uk/)				×
[S19]	Puma (https://eu.puma.com/)				×
[S20]	Logitech (https://www.logitech.com/)				×
[S21]	Levis (https://www.levi.com/)				×
[S22]	Gant (https://gant.com/)				×
[S23]	El Corte Ingles (https://www.elcorteingles.com/)				×
[S24]	NFL (https://www.nfl.com/)				×

References

1. Ganov, S.R., Killmar, C., Khurshid, S., Perry, D.E.: Test generation for graphical user interfaces based on symbolic execution. In: Proceedings of the 3rd International Workshop on Automation of Software Test, AST 2008, pp. 33–40. Association for Computing Machinery, New York (2008)
2. Memon, A.M.: A comprehensive framework for testing graphical user interfaces. Ph.D., University of Pittsburgh (2001). Advisors: Mary Lou Soffa and Martha Pollack; Committee members: Prof. Rajiv Gupta (University of Arizona), Prof. Adele E. Howe (Colorado State University), Prof. Lori Pollock (University of Delaware)
3. Myers, B.A., Rosson, M.B.: Survey on user interface programming. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 1992, pp. 195–202. Association for Computing Machinery, New York (1992)
4. Pommeranz, A., Broekens, J., Wiggers, P., Brinkman, W.P., Jonker, C.M.: Designing interfaces for explicit preference elicitation: a user-centered investigation of preference representation and elicitation process. User Model. User Adap. Inter. **22**(4–5), 357–397 (2012)
5. Delisle, S., Moulin, B.: User interfaces and help systems: from helplessness to intelligent assistance. Artif. Intell. Rev. **18**(2), 117–157 (2002)
6. Hogan, L.C.: Designing for Performance: Weighing Aesthetics and Speed. O'Reilly Media, Inc., Sebastopol (2014)
7. Card, S.K.: The Psychology of Human-Computer Interaction, pp. 49–51. Lawrence Erlbaum Associates (1983)
8. Card, S.K., Moran, T.P., Newell, A.: The keystroke-level model for user performance time with interactive systems. Commun. ACM **23**(7), 396–410 (1980)
9. Al-Megren, S., Khabti, J., Al-Khalifa, H.S.: A systematic review of modifications and validation methods for the extension of the keystroke-level model. In: Advances in Human-Computer Interaction 2018 (2018)
10. Fitts, P.M.: The information capacity of the human motor system in controlling the amplitude of movement. J. Exp. Psychol. **47**(6), 381 (1954)
11. Callahan, E., Koenemann, J.: A comparative usability evaluation of user interfaces for online product catalog. In: Proceedings of the 2nd ACM Conference on Electronic Commerce, EC 2000, pp. 197–206. Association for Computing Machinery, New York (2000)
12. Paz, F., Paz, F.A., Pow-Sang, J.A.: Evaluation of usability heuristics for transactional web sites: a comparative study. In: Information Technology: New Generations. AISC, vol. 448, pp. 1063–1073. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-32467-8_92
13. Paz, F., Paz, F.A., Villanueva, D., Pow-Sang, J.A.: Heuristic evaluation as a complement to usability testing: a case study in web domain. In: ITNG 2015, pp. 546–551. IEEE Computer Society, USA (2015)
14. Yushiana, M., Rani, W.A.: Heuristic evaluation of interface usability for a web-based OPAC. Library Hi Tech **25**, 538–549 (2007). https://doi.org/10.1108/07378830710840491
15. Lim, C., Song, H.D., Lee, Y.: Improving the usability of the user interface for a digital textbook platform for elementary-school students. Educ. Tech. Res. Dev. **60**(1), 159–173 (2012)
16. Tonn-Eichstädt, H.: Measuring website usability for visually impaired people-a modified GOMS analysis. In: Proceedings of the 8th International ACM SIGACCESS Conference on Computers and Accessibility, ASSETS 2006, pp. 55–62. Association for Computing Machinery, New York (2006)

17. Fernandez, A., Insfran, E., Abrahão, S.: Usability evaluation methods for the web: a systematic mapping study. Inf. Softw. Technol. **53**(8), 789–817 (2011). Aug
18. Bakaev, M., Mamysheva, T., Gaedke, M.: Current trends in automating usability evaluation of websites: can you manage what you can't measure? In: 2016 11th International Forum on Strategic Technology (IFOST), pp. 510–514 (2016)
19. John, B.E., Prevas, K., Salvucci, D.D., Koedinger, K.: Predictive human performance modeling made easy. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 2004, pp. 455–462. Association for Computing Machinery, New York (2004)
20. Katsanos, C., Karousos, N., Tselios, N., Xenos, M., Avouris, N.: KLM form analyzer: automated evaluation of web form filling tasks using human performance models. In: Kotzé, P., Marsden, G., Lindgaard, G., Wesson, J., Winckler, M. (eds.) INTERACT 2013. LNCS, vol. 8118, pp. 530–537. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40480-1_36
21. Patton, E.W., Gray, W.D.: SANLab-CM: a tool for incorporating stochastic operations into activity network modeling. Behav. Res. Meth. **42**(3), 877–883 (2010)
22. Lin, J., Newman, M.W., Hong, J.I., Landay, J.A.: DENIM: an informal tool for early stage web site design. In: CHI '01 Extended Abstracts on Human Factors in Computing Systems, CHI EA 2001, pp. 205–206. Association for Computing Machinery, New York (2001)
23. Teo, L., John, B.E.: CogTool-Explorer: towards a tool for predicting user interaction. In: CHI '08 Extended Abstracts on Human Factors in Computing Systems, CHI EA 2008, pp. 2793–2798. Association for Computing Machinery, New York (2008)
24. Katsanos, C., Tselios, N., Avouris, N.: A survey of tools supporting design and evaluation of websites based on models of human information interaction. Int. J. Artif. Intell. Tools **19**(06), 755–781 (2010)
25. Chi, E.H., et al.: The bloodhound project: automating discovery of web usability issues using the infoscent π simulator. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 2003, pp. 505–512. Association for Computing Machinery, New York (2003)
26. Miller, C.S., Remington, R.W.: Modeling information navigation: implications for information architecture. Hum. Comput. Interact. **19**(3), 225–271 (2004)
27. Blackmon, M.H., Kitajima, M., Polson, P.G.: Tool for accurately predicting website navigation problems, non-problems, problem severity, and effectiveness of repairs. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 2005, pp. 31–40. Association for Computing Machinery, New York (2005)
28. Katsanos, C., Tselios, N., Avouris, N.: Infoscent evaluator: a semi-automated tool to evaluate semantic appropriateness of hyperlinks in a web site. In: Proceedings of the 18th Australia Conference on Computer-Human Interaction: Design: Activities, Artefacts and Environments, OZCHI 2006, pp. 373–376. Association for Computing Machinery, New York (2006)
29. Van Oostendorp, H., Juvina, I.: Using a cognitive model to generate web navigation support. Int. J. Hum. Comput. Stud. **65**(10), 887–897 (2007)
30. Dingli, A., Mifsud, J.: Useful: a framework to mainstream web site usability through automated evaluation. Int. J. Hum. Comput. Interact. (IJHCI) **2**(1), 10 (2011)
31. Landauer, T.K., Dumais, S.T.: A solution to Plato's problem: the latent semantic analysis theory of acquisition, induction, and representation of knowledge. Psychol. Rev. **104**(2), 211 (1997)

32. Nielsen, J., Tahir, M., Tahir, M.: Homepage Usability: 50 Websites Deconstructed, vol. 50. New Riders, Indianapolis (2002)
33. Katalon: Katalon recorder v3.4: A powerful selenium ide alternative (2020). https://www.katalon.com/resources-center/blog/katalon-recorder-v3-4/
34. Selenium: Selenium webdriver (2020). https://www.selenium.dev/projects/
35. Ivory, M.Y., Hearst, M.A.: The state of the art in automating usability evaluation of user interfaces. ACM Comput. Surv. **33**(4), 470–516 (2001).
36. Burke, R.: Hybrid recommender systems: Survey and experiments. User Model. User Adap. Inter. **12**(4), 331–370 (2002)