Detecting and Explaining Usability Issues of Consumer Electronic Products

Dario Benvenuti, Emanuele Buda, Francesca Fraioli,
Andrea Marrella, and Tiziana Catarci

Sapienza Universitá di Roma, Rome, Italy
{d.benvenuti,catarci,marrella}@diag.uniroma1.it
{buda.1775383,fraioli.1696638}@studenti.uniroma1.it

Abstract. Usability of a consumer electronic product (CEP) is one of the most important factors that the users consider in purchasing a CEP as well as functionality, price, etc. This has led many companies to realize new shapes of user interfaces (UIs) and styles of interaction for CEPs, ranging from modern touchscreens to physical controls and displays of any kind. Even if the general feeling is that such increased interactivity may enhance the overall user experience, the side effect is that often a CEP's UI provides too many functions that are difficult to learn and use without referring to the user manual, leading to many usability issues. In this paper, we leverage a case study in the CEPs sector to present a novel log-based evaluation technique in the field of Human-Computer Interaction (HCI). Our technique allows us not only to keep track of the user interactions with a CEP's UI during its daily use, but also to understand what has gone wrong during a user interaction, detecting which user actions have caused usability issues and suggesting explanations for solving them, thus providing a crucial feedback to improve the design of the CEP's UI next version.

Keywords: Usability of Consumer Electronic Products (CEPs); Log Study; User Interface (UI) Log; Interaction model; Trace Alignment; Heuristic Evaluation

1 Introduction

In the last years, there was a rising interest in the integration of Internet-of-Things (IoT) in connected home devices, giving a new lease of life to Consumer Electronic Products (CEPs), which have now functionalities like sensing, actuation, and control [43]. They are often connected to the manufacturer's network via the internet and can send information about product performance, usage trends, and energy consumption. For example, a smart CEP with IoT capabilities like a washing machine has many sensors across its body. It can transmit log data such as water usage, the health of the appliance, ambient and water temperature to the manufacturer. These data are then used by manufacturer designers and software engineers to predict product malfunction, plan for repairs

and update the custom software made, improving at the same time the design of the next product. For this reason, today major CEP companies use to employ expert analysts to reveal important insights from log data [7].

A not yet well explored knowledge that is recorded into log data concerns the concrete user interactions with the User Interface (UI) of a CEP, which could be exploited by expert analysts to compute the usability of a CEP's UI during its daily use. In the Human-Computer Interaction (HCI) field, usability is the key feature to capture the quality of an interaction with a UI in terms of measurable parameters such as time taken to (and learn how to) perform relevant tasks and number of errors made [4].

Since usability of a CEP is one of the most important factors that the users consider in purchasing a CEP as well as functionality, price, etc. [24], many companies started to realize new shapes of UIs and styles of interaction for CEPs, ranging from modern touchscreens to physical controls [16]. Even if the general feeling is that such increased interactivity may enhance the overall user experience, the side effect is that often a CEP's UI provides too many functions – often hidden behind many undifferentiated buttons – that are difficult to learn and use without referring to the user manual, leading to several usability issues [12]. As a matter of fact, some functions remain untouched until the end of the CEP's life, and some end users do not even recognize those functions exist [20].

To measure the usability of a UI, the HCI literature proposes several *user evaluation* techniques (the work [9] identified 95 techniques in 2003), which mainly belong to two categories: *lab studies* and *field studies*. In *lab studies*, participants are brought into a laboratory and asked to perform certain tasks of interest. Analysts can learn a lot about how participants interact with a UI, but the observed behavior happens in a controlled and artificial setting and may not be representative of what would be observed "in the wild" [21]. Alternatively, *field studies* collect data from participants conducting their own activities in their natural environments. Data collected in this way tends to be more authentic than in lab studies, but the presence of an evaluator observing what participants are doing may interfere with the natural flow of the interaction [14]. Although a wide variety of lab and field tests have been developed to measure the usability of a CEP [12, 24, 20], the major obstacle is that the cost and time required to conduct these studies are often too high [10], and delay the time that the CEP is introduced to the market. Note that a late CEP launch in any industry can negatively impact revenues, causing the product to become obsolete faster [41]. Moreover, usability issues can be identified mainly in the "pre-release" stage of the UI, i.e., before the CEP is launched to the market. Consequently, many usability issues remain uncovered or not comprehensively investigated until the "after-release" stage, and companies tend to fix such issues only when they are reported by the end users in form of complaints [11, 48, 18].

To mitigate the above limitations, in this paper we present a novel evaluation technique that exploits log data collected during the after-release stage of a CEP to the automated identification of what goes wrong during the user interactions with the CEP's UI, detecting which user actions have caused usability issues and

suggesting explanations for solving them. Our technique is based on the concepts of *UI logs*, *interaction models*, *trace alignment* and *heuristic evaluation*.

- UI logs include the user actions (from clicks on a touchscreen to the pressure of physical buttons) recorded "in situ" as people interact with the UI of a CEP while executing a relevant task with the CEP itself (e.g., *wash laundry* with a washing machine), uninfluenced by external observers;
- Interaction models describe the expected human-computer dialogs required to properly executing relevant tasks on the UI of a CEP;
- Trace alignment verifies whether the user's "observed" behavior, which is recorded in a specific UI log, matches the "intended" behavior represented as a model of the interaction itself. A perfect alignment between the log and the interaction model is not always possible, thus making deviations be highlighted. Such deviations reflect mistakes or slips made by the user during the interaction with the CEP's UI, i.e., potential usability issues.
- A heuristic evaluation, which is conducted in the context of the CEP's relevant tasks that are under observation, is employed to derive explanations to the usability issues identified through trace alignment.

If compared with existing literature studies on log-based evaluation in HCI [7], which are mainly focused on collecting UI logs for a subsequent usability assessment performed "manually" by expert evaluators, our technique is able to automatically detect usability issues and suggest explanations to fix them directly from the analysis of the UI logs, thus not requiring the intervention of any expert in the after-release stage of a CEP. We have evaluated the effectiveness of our technique through a case study in the CEPs sector, in which the usability of a real microwave has been assessed against 44 potential end users.

The rest of the paper is organized as follows. Section 2 analyzes the previous works dealing with the usability evaluation of CEPs and investigates the existing log based studies in HCI. Section 3 introduces the main peculiarities of the case study, based on the testing of a real microwave oven currently present on the market. Section 4, after providing the relevant background to understand the paper, describes our log-based evaluation technique. Section 5 presents a user experiment performed with real users over the case study of Section 3, in order to assess the effectiveness of our technique. Finally, Section 6 concludes the paper with a critical discussion about the general applicability of the technique.

2 Related Work

Usability evaluation is considered an essential procedure in CEP development. However, planning and conducting such an evaluation requires considerations of a number of factors surrounding the evaluation process including the product, the user, and environmental characteristics, which are difficult to be captured with traditional lab and field studies [1, 15].

For this reason, in 2001 there was an attempt to propose a novel definition of usability and of its main dimensions that could be applied to the case of

CEPs [12]. Usability was defined as "*satisfying the users in terms of both the performance and the image and impression felt by them*", and characterized by the fact that both aspects should be treated equally important in understanding the usability of CEPs. Around this definition, in [24] a structured framework to support analysts to conduct the usability evaluation of a CEP was proposed. In 2006, the work [22] identified issues and actors with a relation to usability in the product development of CEPs. Then, in 2008, the work [20] has proposed a methodology for developing a usability index of CEPs, which consists of classifying usability dimensions, developing usability measures, and building usability index models that can be applied to check the usability of prototypes of a CEP, during their pre-release stage. More recently, in 2017, the work [23] identified practitioner-reported barriers to and enablers of usability in the development of CEPs, where barriers/enablers are conditions in the CEP development process, team, or context that negatively or positively influence the usability of a CEP.

While all the above works have had the merit of delivering useful strategies to approach the issue of measuring usability of a CEP, none of them has gone beyond proposing the use of traditional lab and field usability techniques, which can be considered appropriated to capture the usability of a CEP in its pre-release stage. Conversely, in this paper, *the target was to develop a log-based technique that may act as a valid inspection method to automatically identify and explain usability problems found in a CEP's UI during its after-release stage.*

Over the last years, there was considerable work in HCI on log based evaluations as complementary to traditional lab and field tests [7]. While the majority of literature works are targeted to exploit log analysis to profile the end users of an application for marketing purposes [25, 44] or to enable comparisons between two or more UIs (e.g., A/B testing) [27, 45], there are two relevant works [36, 10] that are closer to our technique. On the one hand, in [36] the authors present a method that supports evaluators to detect repeated patterns of user errors within log files of multiple user interactions. On the other hand, the work [10] proposes extending the Google Analytics features for mobile applications to store specific low-level user actions for logging real use after application release and perform dedicated lab usability testing. However, in both [36] and [10], the burden to evaluate the usability of the UI is left to the evaluators in the after-release stage, which can be a time consuming and error prone task in presence of UI logs keeping track of hundreds or thousands real user interactions. Conversely, in this paper we exploit UI logs for a different yet little-explored challenge, namely the *support of expert analysts in the CEP industry to conduct automated usability evaluation of a CEP during its daily use.* Our technique moves any effort of the evaluators (i.e., the definition of the interaction models and the enactment of the heuristic evaluation) in the pre-release stage. In this way, the technique is able to automatically detect usability issues and suggest explanations to fix them directly in the after-release stage, thus not requiring the intervention of any human expert in this phase.

3 Case Study

As a case study, we consider the working of a real microwave oven currently available on the market (cf. Fig.1(a)), and sold by a well-known CEP company.[1] Let us imagine that the company wants to analyze if the control panel of the oven (cf. Fig.1(b)) has the potential for improvements with regards to its UI design, in view of the realization of a new – more usable – version of the oven.

The control panel provides: *(i)* a display showing the current settings and time; *(ii)* a door opener mechanism; *(iii)* five clickable buttons and a setting knob, which enable to interact with the different oven's functions, as follows:

- **A** - *Function button.* To choose a specific cooking function of the oven.
- **B** - *Defrost button.* To defrost food by weight or time.
- **C** - *Clock / Timer button.* To set the clock or the timer of the oven.
- **D** - *Stop / Clear button.* To deactivate the appliance or delete the cooking settings.
- **E** - *Start / +30 sec button.* To start the appliance or increase the cooking time for 30 seconds at full power.
- **RR/RL** - *Setting knob.* To increase/decrease the cooking time, weight or to activate the auto cooking programmes.

[1] The case study is just an exploration of some possible design issues of a real oven's UI. Since the authors are not affiliated in any way with the CEP company that manufactured the oven, the company name and the oven's model are not disclosed.

(a) Microwave oven (b) Control panel

Fig. 1. Screenshots of the microwave oven investigated in the range of case study

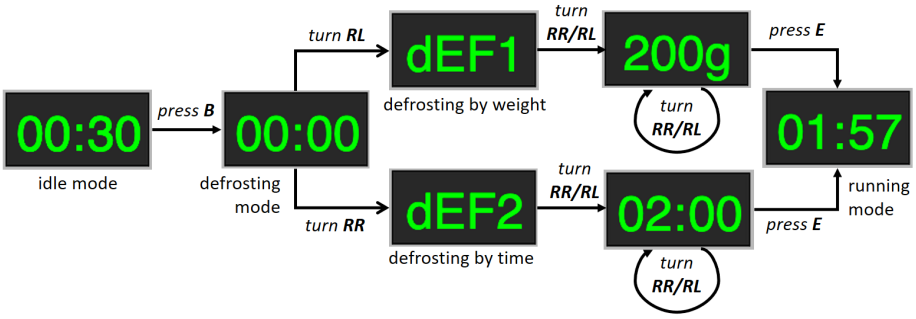

Fig. 2. The flow of user actions needed to start defrosting food with the oven

The microwave oven provides different cooking functions to the end users,[2] such as (among the others) the ability to *defrost frozen foods* without cooking them. To use any of the cooking functions, it is required to press the right combination of buttons in the control panel following a specific sequence. For example, as shown in Fig. 2, to defrost a food the user has to: *(i)* press **B** to switch from the "idle mode" (where it is shown the current time in the oven's display) to the "defrosting mode"; *(ii)* turn the setting knob to the left (**RL**) or to the right (**RR**) to enable defrosting by weight or by time, respectively; *(iii)* turn again the setting knob to the left (**RL**) or to the right (**RR**) as many times as it is needed to set the weight of the food to defrost (for weight defrosting the time is set automatically), or the time of defrosting; *(iv)* press **E** to confirm and activate the microwave.

When the microwave is in "running mode", whatever cooking function has been launched, the display will show a countdown timer. When it expires (display shows "00:00"), an alarm will sound to notify that the cooking function has been successfully completed. At any time, the door opener mechanism can be used to pause the microwave when it is in running mode, i.e., opening the door of the oven does not abort the cooking function in progress.

Given the above scenario, in this case study we assume that the analysts of the CEP company want to assess the usability of the oven's UI (i.e., of its control panel) with respect to the task: "*Defrost a food with the oven by weight or by time*", synthetically called DEFROST FOOD. To this end, the common practice would be to employ a lab or a field study, which requires to involve several users that must be observed by external evaluators over an extended period of time during their interaction with the oven. However, both techniques are expensive in terms of the time they require to collect the data, limiting the number of user tests that can be performed and bounding the scope of the testing activity to the pre-release stage of the CEP.

[2] The complete list of cooking functions provided by the microwave oven can be found in the user manual associated to the oven.

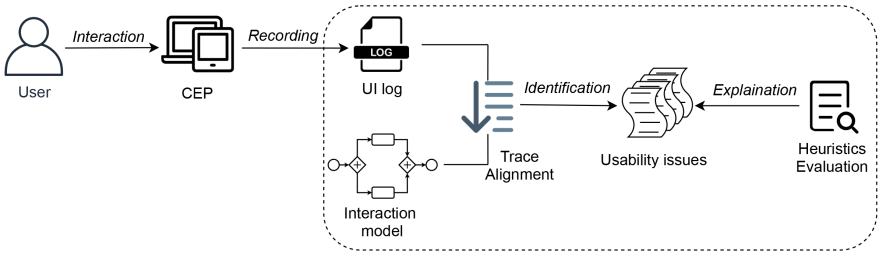

Fig. 3. Overview of the main components of the log-based evaluation technique

In this paper, we leverage the above case study to present a novel evaluation technique that allows us to reason over the concrete user interactions happened with the UI of a CEP (in our case, with the control panel of the microwave oven), and recorded into dedicated UI logs, to identify and explain the usability issues found on the CEP's UI directly during the after-release stage of the CEP.

4 Log-based Evaluation Technique

In this section, we describe the key features of our log-based evaluation technique. To be more precise, as shown in Fig. 3, starting from a *UI log* recorded after many interactions with the UI of a CEP, and an *interaction model* representing the expected human-computer dialogue required to properly executing a relevant task on the CEP itself, our evaluation technique leverages *trace alignment* in Process Mining [26] and a dedicated *heuristic evaluation* [34] to automatically detect and provide explanations to the usability issues found during the interactions with the CEP. In the following, we describe in detail the above "ingredients" needed to concretely employ our technique.

4.1 Collecting User Interface Logs

First, it is necessary to collect a UI log L containing *execution traces* that describe interaction sessions performed by end users during the enactment of a relevant task of interest through the CEP. Such a log can be generated through a massive remote user test, which large companies may periodically conduct with those customers that are known to have purchased the CEP.[3] In fact, nowadays, it is very common that users download a mobile app on their smartphone to register the product ID of the CEP just bought, so that, on the one hand, they can monitor the "health" of the CEP, and – on the other hand – the manufacturer can easily track and contact them if there is a safety alert or CEP recall [19].

To perform a remote evaluation test with a CEP, the end user is instructed via email, or with a notification on the mobile app, to run the CEP in testing mode

[3] Users participating to remote tests are often rewarded with discounts, coupons, etc.

pushing a special combination of buttons on the CEP's UI (usually the CEP returns a visual or acoustic feedback to inform the user that the testing mode has been activated), and then to perform one (or more) suggested task(s) with the CEP. In this time frame, a UI log records all the user actions performed on the CEP's UI, until the user decides to exit from the testing mode (this happens using the same special combination of buttons as before). At this point, the UI log is delivered to the manufacturer through the connection with the mobile app, which acts as a proxy between the CEP and the manufacturer servers.

From a technical point of view, a UI log L is a multi-set of *execution traces* $\sigma_1, ..., \sigma_n \in L$. Each trace $\sigma_i \in L$ consists of a sequence of *user actions* $a_1, a_2, ..., a_m \in Z$, such that $\sigma_i = \langle a_1, a_2, ..., a_m \rangle$ is recorded during one user session, i.e., it is related to the *single execution* of a specific relevant task. Z is the universe of user actions included in L. In the case study of Section 3, we can recognize the set of user actions of interest: $Z = \{A, B, C, D, E, RR, RL\}$. Multiple executions of the same task may consist of the same sequence of actions executed and, hence, result in the same trace. This motivates the definition of a UI log as a multi-set. If we consider our case study, the following $L_1 = [\langle B, RR, RL, E \rangle, \langle B, C, RR, RR \rangle, \langle C, C, C, B, RR, RR \rangle, \langle B, RR, RL, E \rangle, \langle B, B, B, RL, RR, RR, RL, RL, RR, E \rangle]$ is an example of UI log consisting of 5 traces, generated by 5 tests in different user sessions. Within a trace, the concept of time is usually explicitly modeled in a way that user actions in a trace are sorted according to the timestamp of their occurrence.

In this paper, we assume that the user's actions associated to a task executed on the UI are already clustered in execution traces that refer to single enactments of the task itself. This assumption is reasonable, as we envision to collect the user actions performed on a CEP's UI when the CEP is in testing mode, and the user has been asked to perform exactly the task to be tested.

4.2 Defining Interaction Models as Petri nets

Secondly, it is required to formalize the potential dialog between the end user and the CEP by employing dedicated interaction models. An interaction model represents the *expected way* to perform a relevant task on the CEP's UI.

The research literature is rich of notations for expressing human-computer dialogs as interaction models that allow to see at a glance the structure of a user interaction with a UI [39, 1]. Existing notations can be categorized in two main classes: *diagrammatic* and *textual*. Diagrammatic notations include (among the others) various forms of state transition networks (STNs) [47], Petri nets [3], Harel state charts [13], flow charts [1], JSD diagrams [42] and ConcurTaskTrees (CTT) [32]. Textual notations include regular expressions [46], Linear Temporal Logic (LTL) [40], Communicating Sequential Processes (CSPs) [6], GOMS [17], modal action logic [5], BNF and production rules [8].

While there are major differences in expressive power between different notations, an increased expressive power is not always desirable as it may suggest a harder to understand description, i.e., the dialog of a UI can become unmanageable [1]. To guarantee a good trade-off between expressive power and

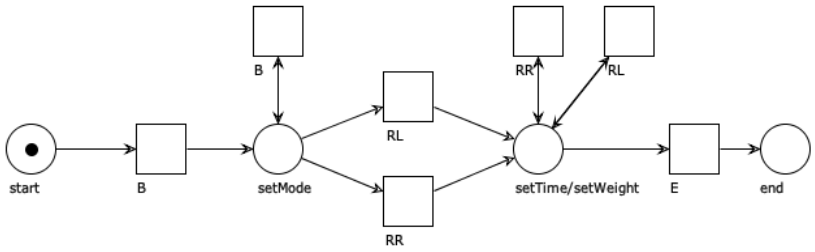

Fig. 4. Interaction model of the task DEFROST FOOD represented as a PN

understandability of the models, to realize our technique we opted for *Petri nets* (PNs) [33], which have a clear semantics and have proven to be adequate for defining interaction models [39, 1, 38]. PNs may contain exclusive choices, parallel branches and loops, allowing the representation of extremely complex behaviours in a compact way and with an exact mathematical definition of their execution semantics.

A PN is a directed bipartite graph with two node types: *places* (represented by circles) and *transitions* (represented by squares) connected via directed arcs. Technically, a PN is a triple (P, T, F) where P and T are the set of *places* and *transitions*, respectively, such that $P \cap T = \emptyset$ and $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation. Fig. 4 illustrates the PN used to represent the interaction model of the task DEFROST FOOD. Transitions are associated with *labels* reflecting the user actions (e.g., buttons clicked, interactions with the setting knob, etc.) required to accomplish the task on the UI of the microwave oven.

Given a transition $t \in T$, ${}^\bullet t$ is used to indicate the set of *input places* of t, which are the places p with a directed arc from p to t (i.e., such that $(p, t) \in F$). Similarly, $t^\bullet$ indicates the set of *output places*, namely the places p with a direct arc from t to p. At any time, a place can contain zero or more *tokens*, drawn as black dots. Any distribution of tokens over the places, formally represented by a total mapping $M : P \mapsto \mathbb{N}$, represents a configuration of the net called a *marking*. The semantics of a PN defines how transitions route tokens through the net changing the number of tokens in places, i.e., the PN marking, so that they correspond to a task execution. In any run of a PN, its marking may change according to the following enablement and firing rules:

- A transition t is *enabled* at a marking m iff each input place contains at least one token: $\forall\ p \in {}^\bullet t,\ m(p) > 0$.
- A transition t can *fire* at a marking m iff it is enabled. As result of firing a transition t, one token is "consumed" from each input place and one is "produced" in each output place. Hence, firing a transition t at marking m leads to a marking m', and this is denoted as $m \xrightarrow{t} m'$.

Fig. 5 illustrates the act of firing for various PN configurations. It is assumed that the firing of a transition is an atomic action that occurs instantaneously and

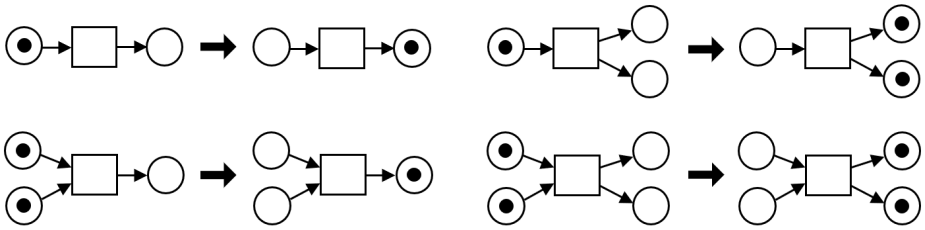

Fig. 5. Example of firing for various PN configurations

can not be interrupted. Within the salient features of PNs is the fact that several enabled transitions can not fire simultaneously, and that an enabled transition is not forced to fire immediately but can do so at a time of its choosing. In the remainder, given a sequence of transition firing $\delta = \langle t_1, \ldots, t_n \rangle \in T^*$, $m_0 \xrightarrow{\delta} m_n$ is used to indicate $m_0 \xrightarrow{t_1} m_1 \xrightarrow{t_2} \ldots \xrightarrow{t_n} m_n$, i.e., m_n is *reachable* from m_0.

Since concrete executions of tasks on a UI have a start and an end, PNs need to be associated with an initial (respectively final) marking, characterized by the presence of one token in at least one of the starting (respectively ending) places of the PN and no tokens in any other place.

All the above features make PNs particularly suitable for modeling concrete interactions with a CEP's UI. Interestingly, an interaction model may allow *different strategies* to perform a relevant task. For example, if we consider the PN in Fig. 4, the number of ways to properly reach the final marking are potentially unbounded (due to the presence of a loop in the model). For instance, the execution traces $[\langle B, RR, RL, E\rangle, \langle B, B, B, RL, RR, RR, RL, RL, RR, E\rangle]$ represent (both) good ways to execute the relevant task underlying the PN in Fig. 4. This basically means that a same task can be completed through different paths of user actions in the UI with equivalent results, as may happen in real UIs. Last but not least, we can define as many interaction models as are the relevant tasks of the CEP to analyze.

In the remainder of this paper, we assume all PNs to be labelled and 1-bounded, also known as *safe*. A PN is 1-bounded if it imposes that the number of tokens in all places is at most 1 in all reachable markings, including the initial one. One-boundness is not a large limitation as the behavior allowed by interaction models can be represented as 1-bounded PNs [1, 38].

4.3 Trace Alignment to detect usability issues

Thirdly, given an interaction model of the task of interest and a UI log associated to it, we can construct the *alignment* between any of the traces extracted from the log and the model. Trace alignment [26] is a conformance checking technique within Process Mining that is employed to *replay* the content of any trace of a log against a model represented as a PN, one action at a time. For each trace in the log, the technique identifies the closest corresponding trace that can be parsed

by the model, i.e., an *alignment*, together with a *fitness* value, which quantifies how much the trace adheres to the model. The fitness value can vary from 0 to 1 (the latter means a perfect matching between the trace and the model.

In this paper, we have customized the technique developed in [26] to construct an alignment between a UI log L and an PN-based interaction model N that exactly pinpoints where deviations occur. To this aim, the user actions in Z need to be related to transitions in the model, and vice versa. For this reason, we can define a function $\ell(T) \in Z$ that maps transitions with user actions in Z.

To establish an alignment between an interaction model and a UI log, we need to relate *moves in the log* to *moves in the model*. However, it may be that some of the moves in the log cannot be mimicked by the model and vice versa. We explicitly denote such "no moves" by $*$. From a formal point of view, *alignment moves* can be defined as follows:

Definition 1 (Alignment Moves). *Let $N = (P, T, F)$ be a PN, L a UI log and Z the universe of user actions included in L. A* legal alignment move *for N and L is represented by a pair $(q_L, q_N) \in (Z \cup \{*\} \times T \cup \{*\}) \setminus \{(*, *)\}$ such that:*

- *(q_L, q_N) is a move in log if $q_L \in Z$ and $q_N = *$,*
- *(q_L, q_N) is a move in model if $q_L = *$ and $q_N \in T$,*
- *(q_L, q_N) is a synchronous move if $q_L \in Z$, $q_N \in T$ and $q_L = \ell(q_N)$*

An *alignment* is a sequence of alignment moves:

Definition 2 (Alignment). *Let $N = (P, T, F)$ be a PN with an initial marking and a final marking denoted with m_i and m_f. Let also L be a UI log. Let Γ_N be the universe of all legal alignment moves for N and L. Let $\sigma_L \in L$ be a log trace. Sequence $\gamma \in \Gamma_N^*$ is an* alignment *of N and σ_L if, ignoring all occurrences of $*$, the projection on the first element yields σ_L and the projection on the second yields a sequence $\delta'' \in T^*$ such that $m_i \xrightarrow{\delta''} m_f$.*

In a nutshell, the alignment activity consists of *replaying* any user action included in a log trace over the interaction model. The output is a pair of aligned traces together with the points of divergence between these two traces. Specifically, a pair shows a trace in the log that does not match exactly a trace in the model, together with the corresponding closest trace produced by the model. For example, Fig. 6 shows a possible alignment for a log trace $\tau_1 = \langle B, C, RR, RR \rangle$ taken by our UI log L_1 and the PN of Fig. 4. Note how moves are represented vertically. For example, as shown in Fig. 6, the first move of γ_1 is (B,B), i.e., a *synchronous move* of B, while the second and fifth move of γ_1 are a move in log and model, respectively.

$$\gamma_1 = \begin{array}{|c|c|c|c|c|} \hline B & C & RR & RR & * \\ \hline B & * & RR & RR & E \\ \hline \end{array}$$

Fig. 6. Alignment of trace $\tau_1 = \langle B, C, RR, RR \rangle$ and the interaction model in Fig. 4.

The presence in an alignment of (non-synchronous) moves in log and model denotes that some actions recorded in the log cannot be matched to any of the actions allowed by the model. As a consequence, the fitness value of γ_1 is lower than 1. To be more specific, a *deviation* can manifest itself in *skipping* actions that have been executed in the log but are not allowed by the model, (e.g., a user pushing the button C that is not needed to activate the defrosting function of the oven), or in *inserting* actions, namely, some actions that should have been executed (i.e., prescribed by the model) but are not observed in the log, e.g., a user that is not able to complete the task DEFROST FOOD since s/he misses to push button E.

If an alignment between a log trace and model contains at least a deviation, it means that the log trace refers to a user interaction that is not compliant with the allowed behavior represented by the model. As a matter of fact, the alignment moves (i.e., skipping or inserting actions) indicate where the interaction is not conforming with the model by pinpointing the deviations that have caused this nonconformity, which is crucial for identifying potential usability issues.

For example, coming back to the alignment in Fig. 6, it is clear that the alignment activity will identify that: *(i)* action C has been executed even if forbidden by the model, and *(ii)* action E is required by the model (even if it does not appear in the trace), and it should have been executed as last action of the trace. The alignment of trace τ_1 with the model will instruct to *skip* action C and *insert* action E, i.e., the *repaired trace* is $\hat{\tau_1} = \langle B, skip(C), RR, RR, add(E)\rangle$. Recovery instructions are labeled with *add* and *skip* to capture those wrong/missing actions that must be skipped/inserted from/into the repaired trace to make it compliant with the model. The analysis of the recovery instructions enables to support an HCI analyst to detect potential usability issues in the UI. For example, considering $\hat{\tau}_1$, it is possible to infer that the user was confused about how to start and conclude the interaction with the the oven to activate the desired cooking function, probably due to the lack of visual cues in the control panel of the oven, or could simply not find the feature.

4.4 Heuristic Evaluation to suggest explanations of usability issues

While trace alignment enables the identification of potential usability issues in the range of a user interaction with a UI, the interpretation and repair of such issues is usually let to HCI expert analysts. In this paper, we relax this assumption relying on heuristic evaluations in HCI, to provide possible explanations to the deviations found through trace alignment.

Specifically, the idea is to involve an expert evaluator that, given a specific task to be tested with the CEP, searches for usability violations in the UI design while simulating its execution, judging its compliance with recognized usability principles (i.e., the heuristics). In the proposed case study, we have relied on the well known Jakob Nielsen 10 heuristics [35]: (H1) visibility of system status through appropriate feedback; (H2) match between system and the real world making information appear with concepts familiar to the user; (H3) user control and freedom to leave the unwanted system's states; (H4) usage of consistency

and standards; (H5) error prevention; (H6) recognition rather than recall of information; (H7) flexibility and efficiency of use allow users to tailor frequent actions; (H8) aesthetic and minimalist design, filtering out any information that is irrelevant; (H9) help users recognize, diagnose, and recover from errors; (H10) provide help and documentation features.

Thus, for any transition (i.e., user action on the UI) belonging to an interaction model represented as a PN, it is asked to an expert evaluator to motivate the reason why a user should select an action different from the one expected at a certain point of the PN execution, relying on the selected heuristic. The results of this activity, which can be performed in the pre-release stage of the CEP under analysis, thus before collecting UI logs, enable to define a clear mapping between potential usability issues hidden within the CEP's UI and their explanations in terms of violated heuristics. Note that the same heuristic can be violated for many reasons, depending by the specific user actions being performed and by the nature of the task to achieve. At this point, any of the identified heuristic violation can be associated to a precise description of the reason of the violation itself. For example, if we consider our case study in Section 3, and specifically the DEFROST FOOD task, the result of the association between the violated heuristics and their explanation in the context of the task is shown in Table 1.

Of course, heuristic evaluation is able to identify potential usability violations, but not to verify if such violations correspond to concrete usability issues in the after-release stage of the CEP. For this reason, it is required to relate heuristic violations found at the outset with deviations detected after the trace alignment step. This can be done by matching the *skip* and *add* actions in an aligned trace with the usability violations found during the heuristic evaluation. For example, given the trace $\tau_2 = \langle C, C, C, B, RR, RR \rangle$ taken by our UI log L_1, the aligned trace is $\hat{\tau_2} = \langle skip(C), skip(C), skip(C), B, RR, RR, add(E) \rangle$. From $\hat{\tau_2}$, it is clear that the correct execution of B is preceded by three wrong occur-

Table 1. An example of mapping between explanations and violated heuristics, produced by an expert evaluator for the task DEFROST FOOD.

Expl. ID	Violated Heuristic	Brief Description
E1	H1	Users do not have enough feedback about their interaction with the oven.
E2	H1	Users do not know if an interaction with the oven has been successfully completed.
E3	H1	The oven's display does not provide any visual cue to understand which button has been pressed.
E4	H2	The symbol associated to Function button in the oven's control panel is not familiar to the users.
E5	H2	The symbol associated to Stop button in the oven's control panel is not familiar to the users.
E6	H2	The symbol associated to Defrost button in the oven's control panel is not familiar to the users.
E7	H2	Defrost modes as shown on the oven's display have names (dEF1/dEF2) not familiar to the users.
E8	H3	Users find it difficult to undo any cooking function started by mistake.
E9	H4	One of the symbols associated to the Setting Knob is associated also to the Defrost button.
E10	H4	The symbol associated to the Function button does not match well with its intended usage.
E11	H4	The symbol associated to Start/+30 sec button does not match well with its intended usage.
E12	H4	The clock symbol associated to the Setting Knob is associated also to the Clock/Timer button.
E13	H5	It is not clear which buttons are useful to activate a cooking function and which are not required.
E14	H5	No confirmation request is asked to the user while pressing any button.
E15	H8	There are too many undifferentiated buttons in the control panel of the oven.
E16	H9	The oven does not show any error message on its display.
E17	H10	The effects of the Function button are not clearly explained in the user manual of the oven.

Table 2. Mapping of violated heuristics and explanations related to a wrong execution of the action B in the context of the task DEFROST FOOD.

Action Expected	B		B	
Action Executed	A		RR or RL, C, D, E	
Mapping	**Explanation ID**	**Violated Heuristic**	**Explanation ID**	**Violated Heuristic**
	E6	H2	E1	H1
	E4	H2		
	E10	H4	E6	H2
	E17	H10		

rences of *C*, which can be interpreted as the violation of the heuristics *H1* and *H2* and explained through *E1* and *E6*, respectively (cf. Table 2). Such explanations are properly described in Table 1 and delivered as an outcome by our technique, together with the deviations (i.e., in this case the *skip* actions found with the trace alignment activity related to them. Note that similar considerations can be done for *add* actions (i.e., moves in model) as well.

It is worth to note that the aim of this paper is to validate the effectiveness of the proposed technique for detecting and explaining usability issues, and not the quality of the specific heuristic employed or the clarity of explanations provided by the expert evaluators. In addition, we point out that there is no restriction to analyze the same UI log relying on many different heuristics (and explanations associated with their violation) to obtain different perspectives of a specific set of usability problems.

5 User Experiments

To validate the effectiveness of our technique over the case study introduced in Section 3, we performed a user experiment involving 44 potential end users (24 females, 20 males) of the microwave oven, selected from a sample of people with an age between 18 and 50 years, that were all familiar (i.e., with a similar level of expertise) with the use of microwaves. Therefore, completely novice users were not considered in the experiment. The users were contacted by broadcasting an email to a large internal university mailing list where we asked them if they wanted to join the experiment and some questions to understand their level of expertise with the use of microwaves. Due to the impossibility of verifying which persons owned the exact version of the oven to be tested, we have implemented a digital clone of the oven using the Processing programming language.[4] The digital clone has been developed relying on the well known Skeuomorphism pattern [37], which is related to the design of UIs that mimic their real-world counterparts in how they appear and/or how the user can interact with them. We notice that the digital clone is not generally requested to make our technique work, as it is thought to be enacted through real-life log data collected using real CEPs. Nonetheless, the presence of the digital clone was required to evaluate the effectiveness of the technique against the case study.

[4] `https://processing.org/`

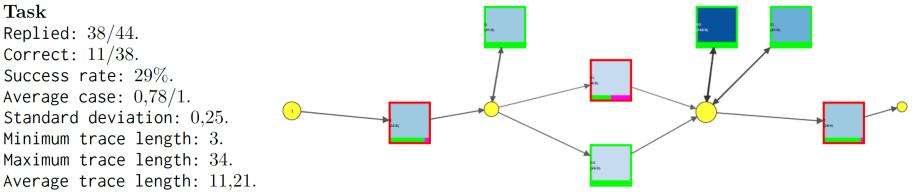

Fig. 7. Overview of the results of the alignment of the task Defrost Food with its related UI log

Then, we contacted via email all the users to involve in the experiment asking them to perform the following activities, in this exact sequence:

1. read the user manual of the oven, only the pages related to the tasks to test;
2. try to run some of the cooking functions of the oven with its digital clone;
3. watch a training video that explains the working of the cooking functions of interest for the test, and that emphasizes the potential mistakes that can be encountered during the interaction with the oven;
4. start the user test.

From a technical point of view, the digital clone was hosted in a dedicated web server reachable by any user involved in the experiment. The interaction with the digital clone was possible using any traditional web browser.

The user test consisted of performing four different tasks (in sequence) of increasing complexity with the digital clone of the oven, including the task Defrost Food, which is described in Section 3. Of course, a separate PN-based interaction model has been defined for any of the tested tasks:

- *Set the clock* of the oven.
- *Grill food*, which enables to heat and brown food quickly. The complexity of this task is comparable to Defrost Food.
- *Set the auto cooking feature*, which enables to register new pre-programmed cooking times and power level based on the type of food to cook.

For the sake of space, in this paper we have analyzed only the results associated with the task Defrost Food because of its average complexity and since it reflects the most frequent task performed by any end user of a microwave.

During the enactment of any user test, we employed a dedicated action logger to collect all the user interactions in form of execution traces recorded into UI logs. Once obtained the UI logs, one per tested task, we ran our trace alignment technique to find deviations between the models and the execution traces in the UI logs. For example, the high-level results concerning the alignment of the task Defrost Food and its associated UI log are described in Fig. 7. Among the various statistics returned, they suggest that 38 out of 44 users tried to perform the task with the oven, but only 11 were able to complete the execution of the task, i.e., to activate the defrosting function. The coloured PN in Fig. 7 can be used to identify which points of the PN were subjected to deviations. Places with

Table 3. Amount of traces in the UI log that violated a specific heuristic in the range of the user action *RL* within the DEFROST FOOD task.

Explanation ID	Low Sev.	Medium Sev.	High Sev.	Catastrophic Sev.
E1	12	10	1	3
E3	12	10	1	3
E9	4	0	0	0
E11	4	3	0	0
E12	6	6	1	0
E13	12	10	1	3
E14	4	0	0	3
E15	0	0	0	3
E16	9	7	1	3
E17	6	1	0	0

the yellow background indicates that at least) a *move in log* deviation occurred before a specific transition, while the green/pink bar placed at the bottom of any transition is used to check the amount of *synchronous moves/moves in model* found during the alignment.

After having found all the deviations (i.e., potential usability issues), the technique associates them with the heuristic violations identified in the pre-release stage of the CEP (see Section 4.4), with the target to relate any deviation to one or more explanations. In Table 3 it is summarized the amount of traces in the UI log that have violated a specific heuristic (and its related explanation) while performing the user action *RL* in the range of the DEFROST FOOD task. Any violation is classified according to a specific degree of severity: 1-Low (1 violation), 2-Medium (2-3 violations), 3-High (4-5 violations), 4-Catastrophic (6+ violations), which is useful to indicate how many times a specific heuristic has been violated in a single execution trace. For example, as shown in Table 3, it is evident that H1 has been violated six or more times in 12 traces, leading to 12 catastrophic violations of H1. The reasons behind such violations are explained by E1 and E3, whose description is reported in Table 1. Of course, the degree of severity of any violation can be customized by an expert evaluator according to the specific task to analyze.

With this knowledge at hand, extracted from the real interactions with the microwave oven, we have derived the three major usability issues found while users interacted with the control panel of the oven to execute the DEFROST FOOD task. Those can be obtained by looking at the explanations associated to the heuristics violated the most times:

- **Ineffective and confusing UI**, related to *E13*, with too many available buttons if compared with the scarcity of cooking functions provided.
- **Absence of feedback**, related to *E1, E3* and *E16*. In the path towards the activation of a cooking function, the interaction with some of the buttons does not provide any feedback on the oven's display, but changes the status of

the interaction. Furthermore, the system does not provide any error message or information about the actions performed previously.
– **Buttons' identifiers are misleading, or too similar between them**, related to *E11* and *E12*. Some of the oven's buttons are associated with symbols that do not help to understand their effects. In addition, some buttons share similar/identical symbols, which make their effect undifferentiated at the eyes of a user.

We have performed the same analysis for any of the other three tested tasks obtaining similar results. The results are also supported by a qualitative evaluation performed after the completion of the user tests. Specifically, we organized a dedicated thinking aloud session by involving 10 further potential end users of the microwave oven (having similar age and characteristics of the users involved in the first test) asking them to execute the four tasks of interest with the digital clone of the microwave, but with an external evaluator observing them. The 10 users were asked to explicitly indicate the usability issues found while interacting with the control panel of the microwave. This allowed us to confirm the validity of the results obtained through our technique, thus certifying it as a valid inspection method to identify and explain usability issues.

6 Discussion and Concluding Remarks

The market for smart consumer electronics has expanded over the last few years and is expected to reach approximately USD 1,787 billion in 2024 owing to evolving consumer experiences and changing lifestyle preferences.[5] For this reason, consumer electronics is one of the major industries getting impacted by the IoT revolution [30]. In this direction, today it is becoming common practice to capture the interactions with a UI during daily use and save them into UI log files for later analysis employing dedicated log studies [7]. In fact, UI logs have the benefit of being easy to capture at scale, enabling to observe even small differences that exist between populations, including unusual behaviour that is hard to capture with the other studies. In this direction, in this paper we have presented a novel log-based evaluation technique that can be successfully employed to automatically detect usability issues of a CEP's UI and suggest useful explanations to support their fixing.

Our technique has been proven to be particularly suitable for evaluating the usability of wizard-based and structured tasks, i.e., with predefined entry and exit points. For this reason, we believe it can be useful for the usability assessment of those categories of CEPs that provide fixed procedures to activate their functions (e.g., domestic appliances, housekeeping tools, etc.). Moreover, the explanations provided to fix the usability issues may support usability experts to realize UIs for next products that are closer to the users' expectations. However, on the other hand, the use of PNs may limit the possibility to model extremely complex behaviours involving many combinations of actions in the

[5] cf. `https://www.zionmarketresearch.com/news/consumer-electronics-market`

context of "less-structured" tasks, such as the ones that are provided by modern audiovisual tools (e.g., smart TVs, game consoles, etc.). This aspect can be potentially mitigated employing less prescriptive modeling notations, such as regular expressions and temporal logics, e.g., see [29].

Another strength of the technique relies on the possibility to be enacted when the user concretely interacts with the CEP in the real user context. Furthermore, for a specific relevant task, different interaction models can be developed to represent the different interaction strategies of novice and experienced users (this can be useful for defining the interactions with less user-friendly CEPs). Finally, the granularity of user actions in interaction models can be customized on the basis of the kind of user logs recorded by the CEPs, i.e., the technique is *scalable*. All these aspects make our technique flexible and customizable for several settings and different types of CEP.

The main limitation of the proposed technique is that the tasks to be tested have to be predetermined and known, since they require to be explicitly modeled through PN-based interaction models. However, although the need to explicitly define interaction models may require some extra modeling effort, we believe that the overhead is compensated by the possibility of detecting the usability issues and their explanations in the after-release stage of the CEP's life-cycle, where it is complex to perform any kind of usability evaluation that is different from the traditional lab and field studies. We think that the above limitation can be mitigated by employing algorithms for PN discovery (cf. [2]), which would allow us to automatically derive the structure of interaction models from the UI logs related to different tasks' executions, thus simplifying the modeling effort.

We note that the proposed technique can be customized with minor modifications to accept in input interaction models defined with other flow-based modeling languages than PNs. In fact, it is undoubtable that there are aspects of the interaction that can not be formalized through PNs, including the objects manipulated by user actions and the representation of the users roles (that can be relevant for certain kinds of CEP). They can certainly be modeled by richer languages, e.g., the standard language CTT (ConcurTaskTrees [32]) for designing interaction sequences. However, in this paper we did not aim at demonstrating that PNs are suitable for modeling human-computer dialogs. In fact, despite of their simplicity, it has been already proven that PNs are sufficiently adequate to model the key aspects of an interaction [39, 1, 38, 28]. The same reasoning can be applied to clarify that, for the explanations of the usability issues, different heuristic evaluation methods than the one of Nielsen can be employed without affecting the effectiveness of the technique.

We conclude by emphasizing that our technique takes inspiration from conformance checking techniques in the Process Mining field [26]. Such techniques are employed to detect deviations in the execution of real-world business processes, whose structure is more complex than the one of tasks in CEPs [31]. This further emphasizes the practical relevance of our technique in the HCI field.

Acknowledgments. This work has been supported by the "Dipartimento di Eccellenza" grant, the H2020 project DataCloud and the Sapienza grant BPbots.

References

1. Alan, D., Janet, F., Gregory, A., Russell, B.: Human-Computer Interaction. Pearson (2004)
2. Augusto, A., Conforti, R., Dumas, M., Rosa, M.L., Maggi, F.M., Marrella, A., Mecella, M., Soo, A.: Automated Discovery of Process Models from Event Logs: Review and Benchmark. IEEE Transactions on Knowledge and Data Engineering **31**(4), 686–705 (2018)
3. Bastide, R., Palanque, P.A., Sy, O., Le, D.H., Navarre, D.: Petri Net Based Behavioural Specification of CORBA Systems. In: 20th International Conference on Application and Theory of Petri Nets. pp. 66–85. Springer-Verlag (1999)
4. Benyon, D.: Designing interactive systems: A comprehensive guide to HCI, UX and interaction design. Pearson (2014)
5. Campos, J.C., Sousa, M., Alves, M.C.B., Harrison, M.D.: Formal Verification of a Space System's User Interface with the IVY Workbench. IEEE Transactions on Human-Machine Systems **46**(2), 303–316 (2016)
6. Dignum, M.: A Model for Organizational Interaction: Based on Agents, Founded in Logic. Ph.D. thesis, Utrecht University (2004)
7. Dumais, S., Jeffries, R., Russell, D.M., Tang, D., Teevan, J.: Understanding User Behavior Through Log Data and Analysis. In: Ways of Knowing in HCI, pp. 349–372. Springer New York (2014)
8. Feary, M.S.: A Toolset for Supporting Iterative Human Automation: Interaction in Design. NASA Ames Research Center, Tech. Rep. 20100012861 (2010)
9. Ferre, X., Juristo, N., Moreno, A.M.: Improving software engineering practice with HCI aspects. In: International Conference on Software Engineering Research and Applications (SERA'03). pp. 349–363. Springer, Berlin, Heidelberg (2003)
10. Ferre, X., Villalba, E., Julio, H., Zhu, H.: Extending Mobile App Analytics for Usability Test Logging. In: 16th IFIP Conference on Human-Computer Interaction (INTERACT'17). pp. 114–131. Springer International Publishing (2017)
11. Fu, B., Lin, J., Li, L., Faloutsos, C., Hong, J., Sadeh, N.: Why people hate your app: Making sense of user feedback in a mobile app store. In: 19th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining. pp. 1276–1284 (2013)
12. Han, S.H., Yun, M.H., Kwahk, J., Hong, S.W.: Usability of consumer electronic products. International Journal of Industrial Ergonomics **28**(3-4), 143–151 (2001)
13. Harel, D.: Statecharts: A visual formalism for complex systems. Science of computer programming **8**(3), 231–274 (1987)
14. Hartson, H.R., Andre, T.S., Williges, R.C.: Criteria for evaluating usability evaluation methods. Int. Journal of Human-Computer Interaction **13**(4), 373–410 (2001)
15. Humayoun, S.R., Catarci, T., de Leoni, M., Marrella, A., Mecella, M., Bortenschlager, M., Steinmann, R.: The WORKPAD User Interface and Methodology: Developing Smart and Effective Mobile Applications for Emergency Operators. In: 5th International Conference on Universal Access in Human-Computer Interaction. Applications and Services (UAHCI '09). pp. 343–352 (2009)
16. Janlert, L.E., Stolterman, E.: Things that keep us busy: The elements of interaction. MIT Press (2017)
17. John, B.E., Kieras, D.E.: The GOMS Family of User Interface Analysis Techniques: Comparison and Contrast. ACM Trans. Comput.-Hum. Interact. **3**(4), 320–351 (1996)
18. Joung, J., Jung, K., Ko, S., Kim, K.: Customer Complaints Analysis Using Text Mining and Outcome-Driven Innovation Method for Market-Oriented Product Development. Sustainability **11**(1), 1–14 (2019)

19. Kao, C.K., Liebovitz, D.M.: Consumer Mobile Health Apps: Current State, Barriers, and Future Directions. Journal of the American Academy of Physical Medicine and Rehabilitation **9**(5), 106–115 (2017)
20. Kim, J., Han, S.H.: A methodology for developing a usability index of consumer electronic products. International Journal of Industrial Ergonomics **38**(3-4), 333–345 (2008)
21. Kjeldskov, J., Skov, M.B.: Was it worth the hassle?: Ten years of mobile HCI research discussions on lab and field evaluations. In: 16th Int. Conf. on Human-Computer Interaction with Mobile Devices & Services (MobileHCI '14). pp. 43–52. ACM (2014)
22. van Kuijk, J., Christiaans, H., Kanis, H., van Eijk, D.: Usability in the Development of Consumer Electronics: Issues and Actors. In: 16th World Congress on Ergonomics (IEA'06) (2006)
23. van Kuijk, J., Kanis, H., Christiaans, H., van Eijk, D.: Barriers to and Enablers of Usability in Electronic Consumer Product Development: A Multiple Case Study. Human–Computer Interaction **32**(1), 1–71 (2017)
24. Kwahk, J., Han, S.H.: A methodology for evaluating the usability of audiovisual consumer electronic products. Applied Ergonomics **33**(5), 419–431 (2002)
25. Lau, T., Horvitz, E.: Patterns of Search: Analyzing and Modeling Web Query Refinement. In: UM99 User Modeling, pp. 119–128. Springer (1999)
26. de Leoni, M., Marrella, A.: Aligning Real Process Executions and Prescriptive Process Models through Automated Planning. Expert Systems with Applications **82**, 162 – 183 (2017)
27. Lettner, F., Holzmann, C.: Automated and unsupervised user interaction logging as basis for usability evaluation of mobile applications. In: 10th Int. Conf. on Advances in Mobile Computing & Multimedia (MoMM '12). pp. 118–127. ACM (2012)
28. Marrella, A., Catarci, T.: Measuring the Learnability of Interactive Systems Using a Petri Net Based Approach. In: Designing Interactive Systems Conference, (DIS '18). pp. 1309–1319 (2018)
29. Marrella, A., Ferro, L.S., Catarci, T.: An approach to identifying what has gone wrong in a user interaction. In: 17th IFIP TC 13 International Conference on Human-Computer Interaction (INTERACT'19). pp. 361–370 (2019)
30. Marrella, A., Mecella, M., Russo, A.: Collaboration on-the-field: Suggestions and beyond. In: 8th International Conference on Information Systems for Crisis Response and Management (2011)
31. Marrella, A., Mecella, M., Sardiña, S.: Supporting adaptiveness of cyber-physical processes through action-based formalisms. AI Commun. **31**(1), 47–74 (2018)
32. Mori, G., Paternò, F., Santoro, C.: CTTE: Support for Developing and Analyzing Task Models for Interactive System Design. IEEE Trans. Sof. Eng. **28**(8), 797 – 813 (2002)
33. Murata, T.: Petri nets: Properties, analysis and applications. Proc. of the IEEE **77**(4), 541 – 580 (1989)
34. Nielsen, J.: Finding usability problems through heuristic evaluation. In: SIGCHI conference on Human Factors in Computing Systems (CHI'92). pp. 373–380 (1992)
35. Nielsen, J.: 10 Usability Heuristics for User Interface Design. Nielsen Norman Group (1994), `https://www.nngroup.com/articles/ten-usability-heuristics/`
36. Okada, H., Asahi, T.: Guitester: A Log-Based Usability Testing Tool for Graphical User Interfaces. IEICE Transactions on Information and Systems **82**(6), 1030–1041 (1999)

37. Page, T.: Skeuomorphism or flat design: future directions in mobile device User Interface (UI) design education. International Journal of Mobile Learning and Organisation **8**(2), 130–142 (2014)
38. Palanque, P.A., Bastide, R.: Petri Net Based Design of User-Driven Interfaces Using the Interactive Cooperative Objects Formalism. In: Interactive Systems: Design, Specification, and Verification. pp. 383–400. Springer (1995)
39. Paterno, F.: Model-Based Design and Evaluation of Interactive Applications. Springer-Verlag, 1st edn. (1999)
40. Pnueli, A.: The temporal logic of programs. In: 18th Annual IEEE Symposium on Foundations of Computer Science. IEEE (1977)
41. Supply & Demand Chain Executive: What Are Late Product Launches Really Costing You? (2017), `https://www.sdcexec.com/sourcing-procurement/article/20985884/`
42. Sutcliffe, A.G., Wang, I.: Integrating Human Computer Interaction with Jackson System Development. The Computer Journal **34**(2) (1991)
43. Thapliyal, H.: Internet of things-based consumer electronics: Reviewing existing consumer electronic devices, systems, and platforms and exploring new research paradigms. IEEE Consumer Electronics Magazine **7**(1), 66–67 (2017)
44. Tyler, S.K., Teevan, J.: Large Scale Query Log Analysis of Re-Finding. In: Third ACM Int. Conf. on Web search and data mining. pp. 191–200 (2010)
45. Valle, T., Prata, W., et al.: Automated Usability Tests for Mobile Devices through Live Emotions Logging. In: 17th Int. Conf. on Human-Computer Interaction with Mobile Devices and Services Adjunct (MobileHCI'15). pp. 636–643. ACM (2015)
46. Van Den Bos, J., Plasmeijer, M.J., Hartel, P.H.: Input–Output Tools: A Language Facility for Interactive and Real-Time Systems. IEEE Trans. Sof. Eng. **9**(3), 247–259 (1983)
47. Wasserman, A.I.: Extending State Transition Diagrams for the Specification of Human-Computer Interaction. IEEE Trans. on Soft. Eng. (8), 699–713 (1985)
48. Zhang, X., Qiao, Z., Tang, L., Fan, W., Fox, E., Wang, G.: Identifying Product Defects from User Complaints: A Probabilistic Defect Model. Decision Support and Analytics (SIGDSA'16) (2016)