Reinforcement Learning-Based Framework for the Intelligent Adaptation of User Interfaces

Daniel Gaspar-Figueiredo
Universitat Politècnica de València &
ITI
Valencia, Spain
dagasfi@epsa.upv.es

Marta Fernández-Diego
Universitat Politècnica de València
Valencia, Spain
marferdi@omp.upv.es

Ruben Nuredini
Heilbronn University of Applied
Sciences
Heilbronn, Germany
ruben.nuredini@hs-heilbronn.de

Silvia Abrahão
Universitat Politècnica de València
Valencia, Spain
sabrahao@dsic.upv.es

Emilio Insfrán
Universitat Politècnica de València
Valencia, Spain
einsfran@dsic.upv.es

ABSTRACT

Adapting the user interface (UI) of software systems to meet the needs and preferences of users is a complex task. The main challenge is to provide the appropriate adaptations at the appropriate time to offer value to end-users. Recent advances in Machine Learning (ML) techniques may provide effective means to support the adaptation process. In this paper, we instantiate a reference framework for Intelligent User Interface Adaptation by using Reinforcement Learning (RL) as the ML component to adapt user interfaces and ultimately improving the overall User Experience (UX). By using RL, the system is able to learn from past adaptations to improve the decision-making capabilities. Moreover, assessing the success of such adaptations remains a challenge. To overcome this issue, we propose to use predictive Human-Computer Interaction (HCI) models to evaluate the outcome of each action (*i.e.,* adaptations) performed by the RL agent. In addition, we present an implementation of the instantiated framework, which is an extension of OpenAI Gym, that serves as a toolkit for developing and comparing RL algorithms. This Gym environment is highly configurable and extensible to other UI adaptation contexts. The evaluation results show that our RL-based framework can successfully train RL agents able to learn how to adapt UIs in a specific context to maximize the user engagement by using an HCI model as rewards predictor.

CCS CONCEPTS

• **Software and its engineering** → **Software design engineering**; • **Human-centered computing** → *User interface design.*

KEYWORDS

Adaptive User Interfaces, Reinforcement Learning, Human-Computer Interaction

1 INTRODUCTION

Adapting user interfaces (UIs) to the dynamic needs and preferences of users taking into account the various contexts of use by suggesting changes at the right time and place is a major challenge in software systems. The main goal is to provide timely and contextually relevant adaptations that significantly improve the User Experience (UX). Recent advances in Machine Learning (ML) have introduced promising ways to enhance the adaptation process [1]. Thus, in this paper, we explore the use of RL as the ML component of the conceptual framework for Intelligent User Interface Adaptation proposed in a previous work [1]. In this context, Reinforcement Learning (RL) techniques can be used as an instrument for step-wise adaptations of the UI based on user's interactions. This adaptation process will continuously align the interface with user's preferences improving the overall UX. Additionally, the inherent "trial-and-error" nature of RL methods as well as the mechanism for penalizing mistakes and rewarding successes, provides better understanding of users' preferences.

Since RL is based on learning by success and failure, quantifying success in the context of UI adaptation is not a trivial task [25]. Successful adaptation can be interpreted in different ways, for example, one that improves user efficiency, one that improves user engagement, or a combination of both. But how to assess these metrics and even how to combine them is far from straightforward. Therefore, in this paper we propose the integration of predictive Human-Computer Interaction (HCI) models to assist in this regard. These models can provide an indicator of the success of adaptations by measuring certain aspects of user interaction and evaluating the impact of adaptation actions on the overall UX. As an initial approximation, our HCI model focus on predicting user engagement.

Finally, we present an implementation of the extended framework, based on OpenAI Gym. This implementation serves as a configurable toolkit that allows developers to create, on the one

hand, the definition of the adaptive capabilities of their UI and the contextual information they can monitor and, on the other hand, to create and compare RL algorithms for the adaptation of the UI for the context they have defined. The configurable and extensible nature of the implementation ensures scalability for various scenarios.

Moreover, we evaluated this implementation by instantiating a specific case and running the evaluation in simulated environments. This experiment, though simulated, serves as a proof of concept, demonstrating the feasibility and adaptability of the proposed approach. Simulated data utilized in the experiment can be readily replaced with real-world or synthesized data, showcasing the flexibility and generalizability of the framework. The capability to adapt to different datasets while maintaining the same methodology highlights the versatility and potential real-world applicability of the extended framework.

This paper is organized as follows: In Section 2, we review related works to contextualize our contribution. Section 3 introduces the extended Intelligent User Interface Adaptation Framework, providing the conceptual basis. The implementation details of our RL agent for UI adaptation are presented in Section 4. Section 5 offers an evaluation of the framework's performance. Finally, Section 6 concludes the paper, summarizing key findings and suggesting potential avenues for future research.

2 RELATED WORK

Adaptive systems and adaptive user interfaces (AUIs) have become crucial in modern software applications to tackle usability problems [2]. These systems can modify aspects of their structure, functionality, or interface to meet the varying and evolving requirements of individual users or user groups over time [27]. However, providing appropriate adaptations, deciding when to present and display them, and ensuring they add value to end-users remains challenging for adaptive systems and AUIs. Different computational approaches to the problem of AUIs have been studied such as rule-based systems, heuristics, bandits, Bayesian optimisation, and supervised learning [26].

Research on AUI design has focused on the development of adaptation rules. These rules have traditionally been created with the help of UX experts' or system designers' knowledge [15] Mezhoudi and Vanderdonckt [19]. Moreover, most adaptive systems have relied on users directly stating their preferences. However, some facets of user preferences manifest through their behavioral patterns rather than through self-inspection. Recent approaches are increasingly considering implicit aspects of the user, such as their cognitive processing capabilities and the user's physiological state [9]. In the context of adaptive menus, systems still follow a heuristic approach where adaptations are selected based on manually-encoded rules that exploit data such as click frequency, visit duration or recency [24].

More recently, ML has enabled the automatic deduction of such adaptation rules from the users interaction data with the system. This automatic deduction process is performed using various ML techniques and algorithms [16]. Learning is recognized as a key capability for adaptive systems. For well-defined environments

where the user state is highly predictive of adequate adaptation, the problem can be approached as a supervised learning problem.

In the more general situation where nor the user state is trivially known nor it is highly predictive of adequate adaptation, a RL approach is preferred. The multi-armed bandit problem is a classic RL problem that exemplifies the exploration–exploitation trade-off dilemma. Additionally, Bayesian optimization is used for problems with continuous parameters and an infinite number of potential options. In the context of AUI, such approaches offer a new paradigm for designing UIs in collaboration with Artificial Intelligence and user data [18] [10]. However, they have been proven successful in simple adaptation problems, such as recommendations and the calibration of interface parameters.

The special case of the RL problem in which the next state is not dependent on the action taken (*i.e.,* bandit problem) is not appropriate as regards learning policies for sequences of adaptations in which rewards are not immediately achievable. Problems that do require this kind of long-term planning should be solved with other RL algorithms. For example, Monte Carlo Tree Search (MCTS) has been proposed as a promising technique for the development of adaptive menu search interfaces [26]. In this context of menu searching in UI, Todi et al. [26] used predictive HCI models to predict rewards for each state during simulations. Since online simulations can be computationally expensive, a pretrained value network was used to directly obtain value estimates for unexplored states. Training data for this neural network was generated using the predictive HCI models. The authors showed that while the computation time increases drastically with simulations as search depth increases, it remains constant with the neural network approach without interfering much in the overall success rate (92.7% with model-based simulation vs. 89.6%).

Traditionally, AUIs have focused adaptations narrowly on specific UI elements rather than holistic changes across the entire system. According to the user's needs, preferences, and context, UI elements such as menus, buttons, bars, icons can be modified by adjusting font size, layout, color, contrast, theme. While this approach can efficiently resolve isolated usability problems, it disregards aspects of end-user's interaction that fall within the realm of UX. This inherent complexity of AUIs is imposing considerable challenges. Thus, UI adaptation should be considered as a multi-factorial problem in order to avoid improvements along some dimensions (e.g. user performance) at the expense of degradations along others (e.g. cognitive destabilisation) [1]. A related challenge is the resolution of conflicting UI adaptation alternatives. Once again, ML techniques could guide the multi-criteria decision-making process to identify the UI adaptation closest to the end-user's goals and RL emerges as promising.

In fact, our purpose in this paper is not focused on a specific UI element and on a specific task, for example on menus to speed up searches as in [26], but on any adaptable UI element taking into account the needs and preferences of users. In our framework, these adaptable UI elements can be easily added and removed as well as their different configurations so, we provide a holistic view of the adaptation process of the UI. Unlike [26] where the implementation of the RL environment is customized for the case of menu searches, our vision is to provide an environment that can be reusable for other purposes. Implementing it with OpenAI Gym enables this.
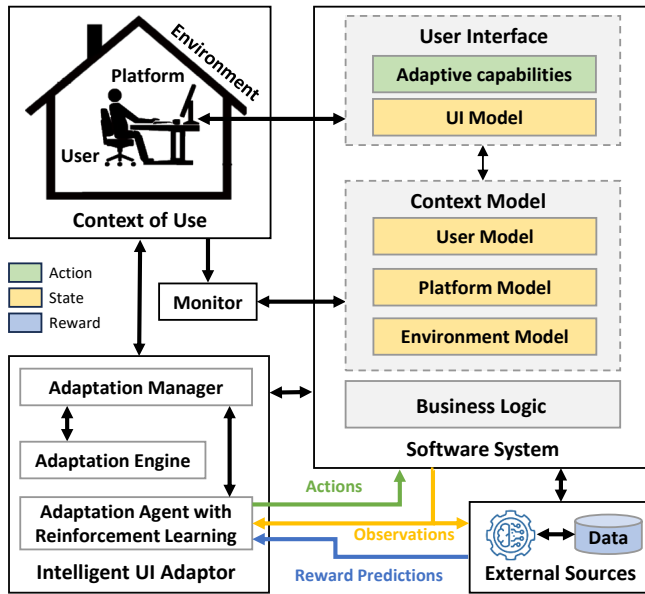
Figure 1: User Interface Adaptation framework using Reinforcement Learning. This is an extension from the original conceptual framework [1].

Even our proposal will facilitate the integration of any RL algorithm to provide full modularity.

3 INTELLIGENT USER INTERFACE ADAPTATION FRAMEWORK

In this paper we extend the conceptual framework proposed by Abrahão et al. [1] by specifying what kind of ML techniques are going to be used and by implementing a first proof of concept. Since the conceptual framework is very complex and covers many aspects, we decided to implement only the mandatory elements in order to make a proof of concept. The Figure 1 shows the four parts of the generic framework (software system, context of use, intelligent UI adapter, and external sources) and how they are connected to each other.

The Software System includes a semantic core component that contains the *Business Logic* functions specific to the application domain. The Software System also includes a *User Interface*, which is responsible for presenting the functionality and data provided by the semantic core to the end-users. This UI has some certain *Adaptive Capabilities*, for example, redistributing elements or modifying the content displayed [12]. Moreover, this UI should be designed and defined as a *UI Model*, for example, it could be defined using the Cameleon Reference Framework [6].

This software is intended to be run on a platform, in an environment and by a user. In order to represent this, the *Context of Use* represents the dynamic environment where end-users engage with the software system. It covers user-specific, platform-related and environment-related factors. This contextual information is then sensed by a *Monitor* that senses and abstracts relevant data into the *Context Model*. It can take into account many factors to

get a better understanding on who, where and on what conditions the software is running. As an example, the *User Model* could take into account factors such as emotional states, personality traits and other characteristics [14]. The Platform Model could include certain information about the device(s) where the Software is running on, and the Environment Model could include information regarding the surroundings where user and platform are located at. This contextual understanding enables adaptive changes in the UI, responding to variations in the individual characteristics, platform specifications, or user's environment.

The conceptual framework proposed by Abrahão et al. [1] defines multiple components in the *Intelligent UI Adaptor*, but only few of them are mandatory to conduct the main adaptation process. In this paper, we focus only on the *Adaptation Manager*, *Adaptation Engine* and we extend the *Adaptation Machine Learning* component to be an *Adaptation Agent with Reinforcement Learning*:

Adaptation Manager: The Adaptation Manager is responsible for coordinating and managing the whole adaptation process. It also provides a UI that allows the end-user to interactively access and update adaptation parameters, review adaptation operations, and make decisions regarding the adaptation process. The Adaptation Manager also executes the adaptation logic contained in the *Adaptation Engine* and coordinates the transition between the status before and after adaptation. Additionally, it can integrate ML techniques to monitor and recommend adaptation operations based on other factors such as contextual changes and user interaction history.

Adaptation Engine: The Adaptation Engine contains the adaptation logic, which refers to the algorithm(s) used to perform the UI adaptation. This can include probabilistic-based models, rule-based approaches, case-based reasoning, logic-based methods, ontology-based techniques, evidence-based approaches, fuzzy logic, and more. The Adaptation Engine is responsible for processing the adaptation logic based on the current context and user requirements to generate adaptation proposals or decisions.

Adaptation Agent with Reinforcement Learning: The Adaptation Agent with Reinforcement Learning operates within the framework of a Markov Decision Process (MDP) to continually monitor the adaptation process, assimilating feedback from successful adaptations and evolving user preferences over time. The adaptation agent extracts essential decision-making information from the software system, namely state (observation) information from the *Context Model* and the *UI Model* and the actions from the *UI Adaptive capabilities*. This agent operates on a reward-based mechanism, where *external sources* contribute as reward predictors by taking advantage of other ML algorithms to process the data obtained by other sources. The historical data information is stored in the Data component so the RL agent can make more informed decisions. This user-centred reward mechanism guides the learning process, guiding the Adaptation Agent towards actions that positively impact UX.

In the initial framework proposal, several properties were outlined, including the definition of the responsibility for adaptation and the underlying reasons for adaptation decisions. However, in our contribution we have chosen to centralise the responsibility for adaptation exclusively on the RL agent. This decision was made to simplify the proof of concept and to highlight the autonomous

learning capabilities of the RL approach. It is worth mentioning that collaborative adaptation strategies will be further investigated, exploring synergies between users and the RL agent as future work. This collaborative approach is likely to involve shared decision-making, allowing users to contribute to the adaptation process, providing valuable insights and fostering a more interactive and user-centred adaptive system.

4 REINFORCEMENT LEARNING AGENT TO ADAPT USER INTERFACES

In this section we define the conceptualisation and implementation of our RL agent for adapting UIs. The conceptualisation is designed to cover a broad spectrum of states and actions to ensure flexibility and extensibility to multiple contexts. Subsequently, we present a practical example to demonstrate the instantiation of this conceptual framework, providing information on configuration and implementation. Finally, in the next section we will use this instantiation to evaluate the framework.

4.1 Markov Decision Process Definition

A Markov Decision Process (MDP) provides a mathematical framework to model decision-making processes in stochastic and dynamic environments, where outcomes are only partially influenced by the decisions that the agent takes. At its core, an MDP comprises states, actions, transition probabilities and rewards that guide the decision-making of an agent. These components collectively enable the agent to learn optimal strategies over time, making MDP a suitable approach for modeling and solving decision problems such as deciding which UI adaptations to perform and when.

4.1.1 States. The concept of states within the MDP framework encapsulate everything that may influence the decision-making process of an intelligent agent. In the case at hand, this entails the contextual information about the User, Platform and Environment and also the various UI design parameters.

In our framework, we represent the states using a *MultiDiscrete* space to deal with different dimensions of information independently. The agent receives this state representation in the form of a vector that includes multiple dimensions, with each dimension assigned discrete values. By adopting this approach, the agent navigates and adapts within a well-organised state space, capturing the various factors that influence UI adaptations. This structured representation improves the comprehensibility and facilitates the configuration of the process.

One illustrative example of changes in the state is when a user switches from a mobile to a desktop device (Platform change) while using an application. In this case, the state would show information changes such as screen size, device capabilities, operating system, and many others. Another example is when a user transitions from a quiet office environment to a noisy coffee shop (Environment change). Here the state would reflect factors like ambient noise and ambient light levels to allow adaptations of the UI for better visibility or interaction. Moreover, if the user states some preferences about the UI configuration at a specific moment or reflects certain emotions while interacting with a software, which can both change over time (User change), these changes should also be updated in the state.

4.1.2 Actions. Actions cover a wide range of possibilities, from alterations in UI elements to more complex modifications in the overall UI design. These may include, but are not limited to, changes in colour schemes, repositioning of UI elements, resizing fonts, changing the content or even the navigation flow on the software [12]. These actions allows our system to respond intelligently to changes in user preferences, contextual variations and evolving environmental conditions.

4.1.3 Transition probabilities. It provides the probability of transitioning from one state to another state after performing adaptation. However, RL does not require explicit specification of the transition probabilities to solve MDP.

4.1.4 Reward. Defining a reward is usually a complex task, but even more for AUIs, due to both the nature of user interactions and the dynamic nature of interface adaptations. We recognise two key facets when defining the reward function: generality, which reflects general trends or preferences observed among users, and individual preferences, which take into account the unique choices of each user. To address these aspects in our reward, we have formulated the following reward function:

$$R = (1 - \sigma) \cdot G + \sigma \cdot I \tag{1}$$

Where, G represents the general trends from different types of users and I represents the individual preferences of each user. On the one hand, G could represent the general tendency of users to prefer dark-themed interfaces [11], based on extensive data analysis of user interactions. On the other hand, I would encompass each user's unique preferences, such as preferring a larger font sizes or a specific colour scheme. Both G and I are normalized to a range between 0 and 1. Then, the parameter σ, ranging from 0 to 1 too, allows us to adjust the balance of reward; for example, if σ is closer to 0, generality is emphasised, while if σ is close to 1, individual preferences are given more weight. Since every component is ranged from 0 to 1, the resulting reward is also ranged from 0 to 1. Thus, a reward closer to 1 means that the action (or adaptation) has been successful.

4.2 Implementation

In our framework, the RL agent continuously monitors user interactions, learning from successful adaptations and recommends personalised adjustments. To implement the RL component, we opted for OpenAI Gym [5], a widely used toolkit designed to easily develop environments through a standardised interface and compare RL algorithms from the ones supported.

Specifically, we created a customised OpenAI Gym environment for the AUI problem. This environment contains the representation of the states and actions and the design of the reward of AUIs, all required for our RL agent. To ensure compatibility with a variety of AUIs, our framework provides a configurable environment. This adaptability is facilitated through a configuration file, which allows developers to define the contextual information, (*i.e.,* User, Platform and Environment data) and the specific UI design information, in order to adjust the framework to their AUI. Moreover, developers can define the set of actions that the UI can take in response to dynamic contextual changes. Furthermore, the configurability covers

also the design of a reward model, enabling developers to shape the system's learning process by specifying the rewards associated with the adaptation outcomes.

On the other hand, our framework provides connectivity with AUIs through API calls. When a software system incorporates adaptive functionalities, these should be easily accessible through well-defined API calls allowing the intelligent adapter, which can operate remotely, to activate or use the adaptive functions of the AUI. Developers can specify API calls directly within the configuration file to articulate the exact functionalities and actions that the intelligent agent can call on the AUI. This approach ensures that the adaptive process is not limited only to a specific environment, but can dynamically interact with and influence external AUIs based on the specified API calls. The complete implementation is available at https://github.com/RESQUELAB/RL-UIAdaptation

4.3 Setup

Since our MDP definition is general to the AUI problem, it can be instantiated with different specific states, actions and reward sources. We define in this section an example of MDP specification that will be used to evaluate our framework.

4.3.1 States. The state includes both, the UI design information and the contextual (User, Platform and, Environment) information. In our first prototype, simple but able to demonstrate the functionality of the framework, only the UI design features and the User preferences will be considered. Future iterations of the framework will include the variability over the Platform and Environment.

With regard to the UI design information that we considered for this evaluation, we focused on four variables: *Layout*, *Theme*, *Font size* and *Information display*. These variables represent the structure, appearance and contents of the interface. The layout variable can take the values *list* and a range of *grid* layouts, from *2-column grid* to *5-column grid* changing how the elements distribute over the interface, the *theme* can take the values *light* and *dark*, the *font size* can be *small*, *default* or *big* and, *information display* can be *show*, *partial*, and *hide*, which will show, show partially or hide some information to the users. On the user side, we took into account changes in *preferences*. The user's preferences can take the values of the design of the UIs. Given the four UI design variables (layout, theme, font size, and information display), each with multiple possible values, the total number of possible combinations can be determined by multiplying the number of options for each variable, resulting in $5 \cdot 2 \cdot 3 \cdot 3 = 90$ combinations. Moreover, we must consider the user preferences with the same possible values as for the UI design. Consequently, this leads to $(5 \cdot 2 \cdot 3 \cdot 3)^2 = 8100$ possible states in this context.

Future evaluations may include additional variables in both UI design and contextual representations to comprehensively assess the adaptability of the framework across various setups.

4.3.2 Actions. The actions considered in this MDP instantiation correspond to changes in the UI design variables. Specifically, we consider actions to distribute elements in a *0) list*, or *1) grid* layout with 2-columns, *2) grid* layout with 3-columns, *3) grid* layout with 4-columns and *4) grid* layout with 5-columns; to activate the *5) light* or *6) dark* themes, to change how the characters are displayed on the UI

with a *7) small*, *8) default* or *9) big* font sizes, and change the amount of information displayed on the screen by *10) showing 11) showing partially* and *12) hiding* the information. Additionally, we introduce a *13) No operate* action, allowing the agent not to make any change. This action acknowledges that, in some situations, maintaining the current UI configuration might be the optimal decision for the agent. To simplify the representation and for compatibility with OpenAI Gym interface, we map these actions to numerical values ranging from 0 to 13, resulting in a total of 14 actions that the agent can take.

4.3.3 Reward. The reward proposed in the Section 4.1.4 is defined by Equation 1. This Equation has two main components, the Generality (*G*) and Individuality (*I*). However, this equation does not specify the source or the way in which these components are derived. In this MDP instantiation, we rely on a predictive HCI model to obtain the *G* component. Then, for the *I* component, we calculate the alignment between the user preferences and the actual UI design configuration. This measures how similar the preferences are compared to the UI design.

On the one hand, with regard to the predictive HCI model, an experiment was conducted involving 25 master students to record their interactions with various UI configurations. This experiment consisted in recording every interaction the participants did while using an e-commerce catalogue. We asked the participants to proceed with the purchase of some products using different configurations of UIs. Specifically, we registered interactions with the UI combinations of layout and theme (*i.e.,* layout in grid or list and theme in dark or light). This experiment provided a data set of how many clicks, scrolls and events that the participants did for each of the configurations. This dataset was then used to calculate the engagement levels [3, 7, 17] associated to the various UI configurations that the participants were using.

However, training a high-quality HCI model requires a substantial number of examples to ensure good performance on unseen, future data. The processes involved, such as recruiting participants, data collection, and labeling, are typically time-consuming and expensive. In response to these challenges, we applied data augmentation techniques to enhance the dataset obtained from the experiment. This approach allowed us to artificially increase the amount of interaction data, addressing the limitations posed by a small dataset. Specifically, we employed SMOTE [8], a data augmentation technique, to generate additional instances, resulting in a more comprehensive dataset for training the predictive HCI model. In the end, we obtained 310 samples of interaction data. Then we trained a *Random Forest Regressor* model capable of predicting a user's engagement level based on the UI configuration. We evaluated the accuracy of the model and we obtained a mean squared error of 0.055.

On the other hand, to calculate the alignment of the *I* component, we created a *simulated user* which has a set of predefined preferences. The alignment measure consists of comparing the preferences of the user with the actual configuration of the UI design. This distance assesses the degree of overlapping or agreement between the user's predefined preferences and the actual state of the UI. A higher alignment score implies a closer alignment, indicating

that the UI design is more consistent with the preferences of the user.

4.3.4 Algorithm. In our prototype, we selected Q-Learning from OpenAI Gym's suite of RL algorithms due to its simplicity, effectiveness with discrete actions, and widespread use as a baseline algorithm. Q-Learning is an off-policy algorithm which aims to determine the optimal next action based on the current state to maximize the accumulated rewards. The Q-value, denoted as $Q(s, a)$, represents the estimate of the cumulative future rewards anticipated by the agent in state s and taking action a. The Q-Learning update equation (Equation 2) refines this estimate over time:

$$Q(s, a) = (1 - \alpha) \cdot Q(S_t, A_t) + \alpha \cdot (R_{t+1} + \gamma \cdot \max_a Q(s_{t+1}, a)) \quad (2)$$

Here, $Q(S_t, A_t)$ represents the current estimate of the Q-value, where α is the Learning Rate. The update term, $\alpha \cdot (R_{t+1} + \gamma \cdot \max_a Q(s_{t+1}, a))$, incorporates the Learning Rate, immediate reward (R_{t+1}), Discount Factor (γ) for future rewards, and the maximum Q-value for the next state s_{t+1}. This equation guides how the agent refines its understanding of Q-values in order to make more informed decisions over time.

In this context, *episodes* represent complete iterations during training, with each *step* corresponding to an individual action or decision made by the agent (*i.e.,* adaptations). Equation 2 guides the agent in refining its understanding of Q-values over steps within an episode.

Additionally, the Q-Learning algorithm starts with no knowledge and balances exploration and exploitation with the *Exploration Factor* (ϵ). As episodes progress, ϵ is gradually reduced, shifting the agent from exploration to exploitation based on accumulated knowledge.

5 EVALUATION

According to the Goal-Question-Metric (GQM) template for goal definition [4], the goal of this study is to *analyze* the Reinforcement Learning agent *with the purpose to* evaluate its effectiveness *with respect to* the ability to learn to adapt the UI *from the point-of-view of* researchers when training RL agents *in the context of* simulated environments, with UI design and User preferences variability and reward obtained through an HCI model trained with historical interaction data obtained from a previous experiment.

This evaluation is conducted through simulations to assess the framework's performance in a controlled environment. The simulations are executed on a computer with the following specifications: Intel Core i9-13900KF, 64 GB DDR5, NVIDIA GeForce RTX 4090. The simulations are set up with the specifications defined in Section 4.3. These simulations aim to provide insights into the framework's capabilities and limitations, for further development and improvement. It's important to note that while simulations serve as a valuable starting point, future work should extend this evaluation process to include human interactions.

5.1 Configuration Parameters of the RL Algorithm

The Equation 2 presents multiple parameters that influence on how the agent will perform its learning. Our goal is to ensure optimal balance between exploration and exploitation in the learning process. The choice of parameter values is based on published experiences applying Q-Learning to tasks with comparable complexity. Furthermore, expert recommendations played a pivotal role in refining and finalizing these values:

- Number of episodes (60000): The total number of iterations or episodes the Q-Learning algorithm will go through during training.
- Learning Rate ($\alpha = 0.90$): A relatively high Learning Rate allows the agent to adapt quickly to new information. A higher Learning Rate ensures that the agent incorporates recent experiences effectively.
- Discount Factor ($\gamma = 0.90$): The Discount Factor determines the weight given to future rewards. A value of 0.90 prioritizes future rewards while still considering short-term consequences.
- Exploration Factor, ϵ: Has an initial value of 1 with linear decay in the first half of the training process (30000 episodes) until its value reaches 0.1. This decay heuristics promotes more exploration in the early stages and exploitation in the later stages of training. The minimum value of 0.1 allows the agent to still explore sometimes while exploiting.

Finally, we aim to explore the trade-off between the G and I in Equation 1, where we set σ to various values [0, 0.25, 0.50, 0.75, 1], offering insights into the system's response to different levels of personalisation.

5.2 Evaluation Metrics

In order to evaluate the ability to learn and the efficiency of the agent used for this framework implementation, we have selected the following metrics, which are commonly used to assess RL agents [20, 21]:

- *Number of steps*: It reflects the total count of adaptations taken by the RL agent to achieve a fully adapted UI. The purpose of tracking the number of steps is to assess the efficiency and effectiveness of the learning process. Ideally, a well-performing agent should learn to adapt the UI minimizing the number of steps required for a complete adaptation. A higher number of steps may indicate suboptimal learning.
- *Episode Score*: The episode score represents the cumulative reward obtained throughout a single episode. The episode score reflects the effectiveness of the agent's actions during that episode. Higher episode scores indicate better performance, as they suggest that the agent has made decisions leading to more favorable outcomes.

To address the inherent noise and fluctuations in the learning process visualization for both the *Number of steps* and *Episode Score*, we incorporated an anti-jittering [22] parameter. This parameter involves recording average values at regular intervals. In our case, we calculate the moving average over a window of size 150 episodes,

providing a stable and representative trend for assessing the performance of the RL agent's learning. This approach ensures that both metrics are presented in a smoothed manner, making it easier to interpret the overall learning progress.

In addition to the commonly used metrics in evaluating RL agents, we propose an additional metric:

- *Alignment Score*: The alignment score measures how well the RL agent's adaptive decisions align with user preferences, providing insights into personalized UI adaptation effectiveness. It's normalized from 0 to 1, calculated as:

$$Alignment = \frac{MaxAlignment - MismatchedAttributes}{MaxAlignment}$$

Where $MaxAlignment$ is the total number of attributes in the user preferences, and
$MismatchedAttributes$ is the count of attributes where the user preferences differ from the UI design.

5.3 Results

In this section, we distinguish between evaluation during learning which is computed over the course of training and evaluation after learning, which is evaluated on a fixed policy after the agent has been trained. During the learning process (*i.e.,* training), evaluation provides insights into how the RL agent evolves over time, demonstrating its ability to adapt and optimize UI configurations. Subsequently, once the agent is trained, the evaluation process (after learning) assesses the agent's performance on simulated users with specific preferences.

5.3.1 Learning Process. In the learning process, we ran 60000 episodes for each of the different values for σ to train the RL agent. We included an extra reward of 1 if the agent finished the whole adaptation process in 4 or less steps. Since there are four types of actions, we presumed that optimal adaptation can be achieved in four steps.

Each episode started with the creation of a new random initial state. Consequently, a new simulated context was generated in each episode, with a random user with specific preferences and new UI design features selected from the available options. The agent's goal was to adapt the UI to maximize the reward, which depended on the σ value.

As illustrated in Figure 2 *a)* and *b)*, the agent demonstrates convergence across various values of σ, transitioning from an exploration phase (depicted in yellow) to an exploitation phase (depicted in green) after the initial 30,000 episodes. However, it's worth noting that for $\sigma = 0.5$ and $\sigma = 0.75$, complete convergence may not have been achieved, suggesting ongoing exploration and refinement of adaptive strategies even in later episodes.

The analysis of the *number of steps*, as shown in Figure 2 *a)*, reveals that the agent efficiently completes episodes in 3 to 4 steps across most scenarios, indicating an adaptation strategy focused on essential changes. However, for $\sigma = 0.5$ and $\sigma = 0.75$, the agent required an additional step, taking 4 to 5 steps to conclude the episodes. This deviation suggests that when there is a balance between generality and individuality leaning towards individuality may lead to increased complexity in the adaptation process.
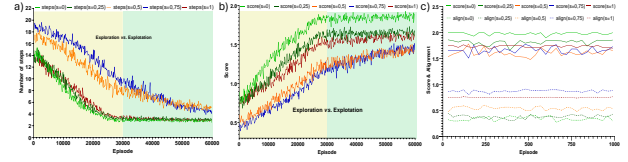

Figure 2: RL agent learning process. a) The number of steps needed to finish an episode decreases over time; b) The score increases over time and the agent converges to an optimal solution; c) RL agent evaluation process.

Meanwhile, examination of the *score* obtained over time, as depicted in Figure 2 *b)*, offers insights into the agent's learning progress and performance. Across most scenarios, a steady growth trajectory is observed, indicating that the agent improves its performance over time. However, notable differences in the score are evident depending on the σ value. For $\sigma = 0$ and $\sigma = 0.25$, where the reward is predominantly influenced by generality, higher and more stable rewards are achieved. This observation suggests that prioritizing generality in the reward function leads to more consistent performance and effective adaptation strategies. Conversely, for intermediate values of σ, such as $\sigma = 0.5$ or $\sigma = 0.75$, where individuality is prioritized, the reward tends to be lower and less stable. However, in the case of $\sigma = 1$, the agent also obtains a high and stable reward. This phenomenon highlights the challenge of balancing generality and individuality in the adaptation process, as reflected in the fluctuating nature of the score. Moreover, the influence of σ on the reward function underscores the importance of carefully selecting and tuning this parameter to optimize the agent's performance in adapting user interfaces.

5.3.2 Evaluation Process. We used the previously trained agent in a series of 1000 episodes to evaluate its performance. Similar to the learning process, each episode in this phase involved creating simulated users with random preferences and starting the adaptation process from a randomly generated initial UI configuration.

Figure 2 c) illustrates the agent's performance in terms of score and alignment across the 1000 episodes for each value of σ. We observe that *alignment* increases as σ emphasizes individuality, with the highest alignment values observed for σ values of 1 and 0.75. Conversely, the lowest alignment values are associated with σ values of 0 and 0.25, while $\sigma = 0.5$ exhibits intermediate alignment, as anticipated. Regarding the *score*, notably, $\sigma = 0.5$ and $\sigma = 0.75$ exhibit more fluctuations compared to other values, suggesting potential for further refinement and convergence through parameter adjustments in future simulations. We find that σ values of 0 and 0.25 yield the highest scores. This indicates that emphasizing generality in the reward function leads to higher cumulative rewards. However, it is noteworthy that all σ values achieve good scores, with values consistently exceeding 1.5 across all scenarios. Despite some variations, these results demonstrate the effectiveness of the agent in adapting user interfaces across different σ values.

5.4 Threats to Validity

In this section, we discuss the threats to validity of our research, following the guidelines from [28]:

Internal Validity: One potential threat is that states or transitions remain unexplored during the agent's learning process. We deliberately constrained the size of the state space. This deliberate choice to limit the complexity of the state space facilitated more efficient experimentation and ensured thorough exploration of all possible transitions within the Q-table. Another internal threat concerns reward definition, which may not perfectly quantify adaptation success. However, we designed a generic reward adaptable to each context. The use of a predictive HCI model for the generalistic (G) part of the reward introduces an internal challenge. The accuracy of the HCI model depends on the dataset quality; our model, trained with data augmented using the SMOTE method from 25 participants, resulting in a dataset of 310 samples, achieved a mean squared error of 0, 055. Despite the limited participant number and data quantity may impact HCI model generalization.

External Validity: An external threat comes from the simplicity of our contextual representation, focusing only on user changes and neglecting platform and environmental variations. Generalizing our results to more complex contexts may be limited. To enhance external validity, future experiments should include broader contextual factors, ensuring a more comprehensive evaluation of our RL-based UI adaptation framework. Additionally, relying on the HCI model to predict user engagement introduces another external threat. While our current experiments use a simplified context representation, extending the model to predict additional aspects like user satisfaction or cognitive load could enrich contextual information. This improvement would offer a more complete representation of system state variability, contributing to a more realistic assessment. Furthermore, our choice of the Q-Learning algorithm introduces a potential external threat. While Q-Learning is well-established, exploring alternative RL algorithms, such as Proximal Policy Optimization (PPO) [23] or Monte Carlo Tree Search (MCTS), in future research may provide a broader understanding of the framework's performance across different algorithmic approaches.

6 CONCLUSIONS AND FUTURE WORK

This paper presents an instantiation of a reference framework for Intelligent User Interface Adaptation utilizing Reinforcement Learning as the machine learning component. The framework aims to adapt user interfaces, ultimately enhancing the overall User Experience. By formulating the problem using MDP, we provide a comprehensive representation of the adaptation process. Our framework offers high configurability and expandability, facilitating customization for specific use cases and evolving requirements. We introduce a holistic reward function capturing both general trends and individual user preferences, alongside a flexible σ parameter for fine-tuning adaptation. Simulated evaluations offer valuable insights into framework performance, but real-world validation is indeed needed. Future work includes exploring alternative reward mechanisms, such as human feedback [13] and algorithms, such as PPO and MCTS, enhancing state coverage, and improving predictive HCI models for better context representation.

REFERENCES

[1] Silvia Abrahão, Emilio Insfran, Arthur Sluÿters, and Jean Vanderdonckt. 2021. Model-based intelligent user interface adaptation: challenges and future directions. *Software and Systems Modeling* 20, 5 (2021), 1335–1349. https://doi.org/10.1007/s10270-021-00909-7

[2] Pierre A. Akiki, Arosha K. Bandara, and Yijun Yu. 2014. Adaptive Model-Driven User Interface Development Systems. *Comput. Surveys* 47, 1, Article 9 (may 2014), 33 pages. https://doi.org/10.1145/2597999

[3] Eduardo Barbaro, Eoin Martino Grua, Ivano Malavolta, Mirjana Stercevic, Esther Weusthof, and Jeroen van den Hoven. 2020. Modelling and predicting User Engagement in mobile applications. *Data Science* 3, 2 (2020), 61–77.

[4] Victor R. Basili, Gianluigi Caldiera, and H. Dieter Rombach. 1994. The Goal Question Metric Approach.

[5] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym. *arXiv preprint arXiv:1606.01540* (2016).

[6] Gaëlle Calvary, Joëlle Coutaz, David Thevenin, Quentin Limbourg, Laurent Bouillon, and Jean Vanderdonckt. 2003. A Unifying Reference Framework for multi-target user interfaces. *Interacting with Computers* 15, 3 (06 2003), 289–308. https://doi.org/10.1016/S0953-5438(03)00010-9

[7] Jonathan Carlton, Andy Brown, Caroline Jay, and John Keane. 2021. Using interaction data to predict engagement with interactive media. In *Proceedings of the 29th ACM International Conference on Multimedia*. 1258–1266.

[8] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16 (2002), 321–357.

[9] Francesco Chiossi, Johannes Zagermann, Jakob Karolus, Nils Rodrigues, Priscilla Balestrucci, Daniel Weiskopf, Benedikt Ehinger, Tiare Feuchtner, Harald Reiterer, Lewis L. Chuang, Marc Ernst, Andreas Bulling, Sven Mayer, and Albrecht Schmidt. 2022. Adapting visualizations and interfaces to the user. *it - Information Technology* 64, 4-5 (2022), 133–143. https://doi.org/10.1515/itit-2022-0035

[10] John J. Dudley, Jason T. Jacques, and Per Ola Kristensson. 2019. Crowdsourcing Interface Feature Design with Bayesian Optimization. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1—-12. https://doi.org/10.1145/3290605.3300482

[11] Henriette Eisfeld and Felix Kristallovich. 2020. The rise of dark mode: A qualitative study of an emerging user interface design trend.

[12] Murielle Florins and Jean Vanderdonckt. 2004. Graceful degradation of user interfaces as a design method for multiplatform systems. In *Proceedings of the 9th International Conference on Intelligent User Interfaces* (Funchal, Madeira, Portugal) *(IUI '04)*. Association for Computing Machinery, New York, NY, USA, 140–147. https://doi.org/10.1145/964442.964469

[13] Daniel Gaspar-Figueiredo, Silvia Abrahão, Marta Fernández-Diego, and Emilio Insfran. 2023. A Comparative Study on Reward Models for UI Adaptation with Reinforcement Learning. arXiv:2308.13937 [cs.SE]

[14] Dominik Heckmann, Tim Schwartz, Boris Brandherm, Michael Schmitz, and Margeritta von Wilamowitz-Moellendorff. 2005. Gumo–general user model ontology. In *User Modeling 2005: 10th International Conference, UM 2005, Edinburgh, Scotland, UK, July 24-29, 2005. Proceedings 10*. Springer, 428–432.

[15] Jamil Hussain, Anees Ul Hassan, Hafiz Syed Muhammad Bilal, Rahman Ali, Muhammad Afzal, Shujaat Hussain, Jaehun Bang, Oresti Banos, and Sungyoung Lee. 2018. Model-based adaptive user interface based on context and user experience evaluation. *Journal on multimodal user interfaces* 12, 1 (1 March 2018), 1–16. https://doi.org/10.1007/s12193-018-0258-2

[16] Pat Langley. 1997. Machine learning for adaptive user interfaces. In *KI-97: Advances in Artificial Intelligence*, Gerhard Brewka, Christopher Habel, and Bernhard Nebel (Eds.). Springer, Berlin, Heidelberg, 53–62.

[17] Janette Lehmann, Mounia Lalmas, Elad Yom-Tov, and Georges Dupret. 2012. Models of User Engagement. In *User Modeling, Adaptation, and Personalization*, Judith Masthoff, Bamshad Mobasher, Michel C. Desmarais, and Roger Nkambou (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 164–175.

[18] J. Derek Lomas, Jodi Forlizzi, Nikhil Poonwala, Nirmal Patel, Sharan Shodhan, Kishan Patel, Kenneth R. Koedinger, and Emma Brunskill. 2016. Interface Design Optimization as a Multi-Armed Bandit Problem. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, Jofish Kaye, Allison Druin, Cliff Lampe, Dan Morris, and Juan Pablo Hourcade (Eds.). ACM, San Jose, CA,

USA, 4142–4153. https://doi.org/10.1145/2858036.2858425

[19] Nesrine Mezhoudi and Jean Vanderdonckt. 2021. Toward a Task-driven Intelligent GUI Adaptation by Mixed-initiative. *International Journal of Human–Computer Interaction* 37, 5 (2021), 445–458. https://doi.org/10.1080/10447318.2020.1824742

[20] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. arXiv:1312.5602 [cs.LG]

[21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (01 Feb 2015), 529–533. https://doi.org/10.1038/nature14236

[22] Vitchyr H Pong, Ashvin V Nair, Laura M Smith, Catherine Huang, and Sergey Levine. 2022. Offline Meta-Reinforcement Learning with Online Self-Supervision. In *Proceedings of the 39th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 162)*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (Eds.). PMLR, 17811–17829. https://proceedings.mlr.press/v162/pong22a.html

[23] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347 [cs.LG]

[24] S. Shrestha, P. Poudel, S. Adhikari, and I. Adhikari. 2022. Adaptive menu: A review of adaptive user interface. *Trends in Computer Science and Information Technology* 7, 3 (2022), 103–106. https://doi.org/10.17352/tcsit.000059

[25] Richard S Sutton, Andrew G Barto, et al. 1998. *Introduction to reinforcement learning*. Vol. 135. MIT press Cambridge.

[26] Kashyap Todi, Gilles Bailly, Luis Leiva, and Antti Oulasvirta. 2021. Adapting User Interfaces with Model-Based Reinforcement Learning. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 573, 13 pages. https://doi.org/10.1145/3411764.3445497

[27] Gianni Viano, Andrea Parodi, James Alty, Chris Khalil, Inaki Angulo, Daniele Biglino, Michel Crampes, Christophe Vaudry, Veronique Daurensan, and Philippe Lachaud. 2000. Adaptive User Interface for Process Control Based on Multi-Agent Approach. In *Proceedings of the Working Conference on Advanced Visual Interfaces* (Palermo, Italy) *(AVI '00)*. Association for Computing Machinery, New York, NY, USA, 201–204. https://doi.org/10.1145/345513.345316

[28] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Empirical Strategies*. Springer Berlin Heidelberg, Berlin, Heidelberg, 9–36. https://doi.org/10.1007/978-3-642-29044-2_2