International Journal of Human–Computer Interaction

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/hihc20

Toward a Task-driven Intelligent GUI Adaptation by Mixed-initiative

Nesrine Mezhoudi & Jean Vanderdonckt

To cite this article: Nesrine Mezhoudi & Jean Vanderdonckt (2020): Toward a Task-driven Intelligent GUI Adaptation by Mixed-initiative, International Journal of Human–Computer Interaction, DOI: 10.1080/10447318.2020.1824742

To link to this article: https://doi.org/10.1080/10447318.2020.1824742

Published online: 02 Oct 2020.

INTERNATIONAL JOURNAL OF HUMAN–COMPUTER INTERACTION
https://doi.org/10.1080/10447318.2020.1824742

Toward a Task-driven Intelligent GUI Adaptation by Mixed-initiative

Nesrine Mezhoudi [a] and Jean Vanderdonckt [b]

[a]Department of Computer Information Systems, College of Computer Science and Information Technology, Imam Abdulrahman Bin Faisal University, Dammam, Saudi Arabia; [b]Louvain Research Institute in Management and Organizations, Université Catholique de Louvain, Louvain-la-Neuve, Belgium

ABSTRACT

Adapting the user interface (UI) to the changing context of use is intended to support the interaction effectiveness and sustain UI usability. However, designing and/or processing UIs adaptation at design time does not encompass real situation requirements. Adaptation should have a cross-cutting and low-cost impact on software patterning and appearance with regard to the situation and the ambient-context. To meet this requirement, we present TADAP proposal for run-time adaptive and adaptable UI based user feedbacks and machine learning. It allows a task-driven adaptation of the user interface (UI) at runtime by mixed-initiative. The particularity of TADAP is the utilization of Machine Learning potential to support context-aware runtime adaptation within model-driven UI. Further, TADAP allows the UI adaptation by mixed-initiative (User and System) considering the user preferences (implicit and explicit) during an interaction. Such a mixed-initiative runtime UI-adaptation tool provides recommendations on how to personalize the UI. Further, it has the ability to track real-time users' interventions and learn their preferences. Diverse tests were performed and showed TADAP as a promising initiative for intelligent model-driven UI adaptation.

1. Introduction

With today's growing use of smart connected devices everywhere, interacting with Graphical User Interfaces (GUI) seeps into most of our daily tasks. Therefore, adapting the UIs intelligently to fit all requirements of the diverse context of use and meet user preferences is mandatory. Interfaces are required to survive changes in their context to enhance the user's control over tasks and improve their experience, throughout smoothing their interaction and reducing their errors. Accordingly, adaptation plays a principal role in the success of interaction systems by allowing systems to be accessible and easily manageable at runtime for different users. The emerging smart environments' appliances exhibit similar properties in an effort to provide end-user customizability and extensibility. Runtime adaptation has become readily available in an adaptive way. These facilities enable systems to progress without recompilation, by generating managing and executing adaptation decisions at runtime.

Divers adaptation techniques were identified in the literature ranging from the adaptability to the adaptivity, along with systems mixing both techniques. Systems are recognized adaptable if they permit their end-users to adjust a selection of system parameters and adapt their behavior accordingly. While adaptive systems are expected to adapt to the users automatically based on the system's assumptions about user needs (Oppermann & Rasher, 1997). Both adaptability and adaptivity features can be incorporated in systems at different levels of functionality and representation with varying effectiveness. Earlier, systems often need recompilation for upgrades, which incur increased cost, delay, and risk. Hence, in the current computational landscape of runtime, pervasiveness, and context-awareness, the support of runtime adaptation becomes the crucial requirement to handle varying resources, changing user needs, and system faults. Thus, models driven engineering involved runtime aspect, and then models are no longer limited to use during the first cycle of development (i.e at the design, implementation, and deployment), but then models are provided with dynamic behavior and were evolved at runtime (Criado et al., 2012; Ghiani et al., 2017). The use of such dynamic models permits one to make reconfiguration decisions based on a global perspective of the running system, apply analytic models to determine correct adaptation strategies, and test the effectiveness of adaptations through continuous system monitoring (Criado et al., 2012; Hussain et al., 2018). However, an efficient implementation of adaptation, that considers changing user preferences and takes several contexts into account at runtime is still a challenge. Since the user represents the most extensive and complex dimension of context, accordingly the user involvement is a crucial requirement to improve the usability of a UI.

To that end, Machine Learning (ML) as a field supporting the solution of complex problems comes in to provide meaningful help (Alpaydin, 2004; Barber, 2010; Bishop, 2006, Mezhoudi, 2013). Throughout several techniques, ML put

CONTACT Nesrine Mezhoudi Nesrine.mezhoudi@gmail.com Department of Computer Information Systems, College of Computer Science and Information Technology, Imam Abdulrahman Bin Faisal University, Dammam 31441, Saudi Arabia

forward an incredible opportunity to advance UI contextualization and convey the adaptation within UI in the executing environment; so far few works that effectively involve ML techniques in finding the beneficial adaptation methods and to solve context recognition questions.

In this paper, we present TADAP a task-based runtime adaptation of the user interface. TADAP approach follows a model-based approach generating UI from high abstract specification to more concrete ones. It considers an abstract task-oriented specification (task tree) together with an abstract UI model specification (AUI) in order to remain model-based and to allow for more flexibility in generating UIs. TADAP is distinguished by an intelligent runtime adaptation embedded in model-based approaches. Moreover, assuming machine learning appliances benefits and supports a variety of interactions with the user, our approach operates a different type of feedback during the learning process.

The rest of this paper is organized as follows; first, we provide a comparative review of the state of art outlining adaptation coverage of existing model-based approaches. Afterward, we present our proposal for the runtime UI generation and adaptation process of TADAP taken together with the task model, platform, and user interventions for adaptation practices. TADAP proposal and its adaptation algorithms have been applied and tested throughout a demonstrative case study. Finally, we evaluate the tool via an empirical evaluation and end-user testing.

2. Related works and challenges

There already exist a lot of approaches dealing with models for the user interface dating back to the 1980s. Such model-based approaches are aimed to user interface creation and adaptation at runtime as well as design-time (Criado et al., 2012; Mezhoudi, 2013; Paterno et al., 2008; Vanderdonckt & Nguyen, 2019). Models were mainly adopted as User Interface Description Languages (UIDLs) to define technology and modality independent UI. Several UIDLs (UsiXml, UIML, XIML, etc.) exist, commonly UsiXml is considered to have the most comprehensive metamodel complying with the CAMELEON reference framework. Additionally, it is possible to define mappings and transformations between the various levels of abstraction (Tasks Model, Abstract UI, Concrete UI, and Final UI) (Akiki et al., 2016Enes Yigitbas et al., 2017; Vanderdonckt & Nguyen, 2019).

Model-based approaches generally use model-driven engineering (MDE) techniques for the generation of the UI. Addressing issues relating to the simplification of the process of UI creation and providing an infrastructure to allow applications to run on different platforms with different capabilities were a common purpose (Gajos et al., 2006). To adapt the UI to their context of use is an ultimate requirement, different adaptation rules are predefined in accordance with context features. Assuming the diversity of interaction's platforms and environments, it is obvious to accommodate different and heterogeneous contextual constraints. Adaptations involve all abstraction levels of model-based UI. Whereas optimizing the UI by accommodating context requirements can result in an adverse change in performance when it doesn't meet user choice (Rosman et al., 2014). Therefore, user preferences still the most relevant constraint to define adaptation. And then human interventions are typically needed to verify and/or correct the result of such adaptation (Mezhoudi & Vanderdonckt, 2015). According to (Rosman et al., 2014) the customization of adaptation decisions is made more complex by the way in which users learn and the extent to which history can contribute to their choice behavior. These purposes require a more refined user model that supports the optimization process. To that end, different approaches have been proposed addressing adaptation problems; and introduced the context information at different levels. Almost all of them stimulate adaptation via an adaptive behavior (Blumendorf et al., 2010; Bodart et al., 1995; Breiner et al., 2011, 2009; Chu et al., 2004; Clerks et al., 2004; Criado et al., 2012; Eisenstein. et al., 2000; Ghiani et al., 2017; Mitrovic et al., 2005, 2007). The primary goal is to ensure pervasive property for user interfaces and having the ability to change during the runtime of the interactive application due to a contextual change.

Different adaptation purposes and context features were addressed; In Roam (Chu et al., 2004), the authors apply models at runtime to build multi-platform adaptation. Then applications allow a user to transfer/migrate a running application between varied devices effortlessly. Adaptation in (Clerks et al., 2004; Mitrovic et al., 2005, 2007) considered user preferences, and then such approaches supported the creation of context-sensitive user interfaces. Usually, the task models are used for improved understanding of the logic of utilization, and then support considering usability guidelines during the design phase. Task models can also support usability evaluation during execution. For instance, TRIDENT (Bodart et al., 1995), Roam (Chu et al., 2004), DynaMo-AID (Clerks et al., 2004), ADUS (Mitrovic et al., 2005, 2007), Teresa (Paterno et al., 2008) Breiner (Breiner et al., 2011, 2009), Criado (Criado et al., 2012) and (Blumendorf et al., 2010) derive a context-sensitive UI from a task task-oriented languages which allow for greater flexibility in generating user interfaces from the abstract specification. Commonly, existing systems provided some mechanism to automatically generate user interfaces and using a simple rule-based approach, where each type of data was matched with precisely one type of interactors. TRIDENT (Bodart et al., 1995) was probably the first system to take more complex context information into account when generating user interfaces. It provides a decision tree that takes into account a broad set of discriminants and represents progress toward automated user interface design. TIMM (Eisenstein. et al., 2000) capitalizes on an adaptive algorithm allowing automated model-based interface design. The adaptive behavior focuses on predicting user behavior using Markov-based models. ADUS (Mitrovic et al., 2005, 2007) capitalizes on monitoring transparently the user behavior at run-time, learn and anticipate potential user actions throughout interface accordingly. A different advanced adaptation was defined by Supple (Gajos et al., 2006), It defines the interface generation as a discrete constrained optimization problem and solves it with a branch-and-bound algorithm using constraint propagation. Supple supports both adaptation modes for personalizing generated interfaces, and convoy automatic user-driven customization. However, the supple generation process does not

benefit a task specification and then it addressed mainly direct manipulation systems. Most of the recent work aims at considering end-users feedback during interaction for adaptation. Egoki system (Gamecho et al., 2015) focused on the accessibility of the UI throughout adopting a model-based approach for generating adaptive user-centered UIs. The system is aimed at automatically tailoring UIs to the end-user with special needs (physical, sensory, and cognitive disabilities). Egoki system has issues with the model's generation and the design of the final UI (Gamecho et al., 2015).

The user context and user experience were the key evaluation factors for (Hussain et al., 2018), A-UI/UX-A presented an authoring tool supporting an allowing the adaptation of user interface based on diverse factors, such as user disabilities and environmental factors. WiSel employs an adaptation approach for context-aware selection of widgets. It consists of a mixed-initiative adaptation approach capitalizing on user feedbacks and addressing usability threats (Mezhoudi et al., 2015). (Enes Yigitbas et al., 2017; Ruiz-López et al., 2013; Yigitbas et al., 2020) addressed self-adaptive model-driven user interfaces. Commonly they focused on a model-driven engineering approach to generate context-aware self-adaptation mechanisms. (Yigitbas et al., 2020) distinguishes by supporting the automatic generation of context services to monitor and detect context information changes.

The major dilemma of existing UI development approaches is the utilization of a restricted set of pre-defined contexts and adaptation rules. Such rule-based approaches performed adaptation at runtime. In practice, adaptation is executed once the rules that satisfy the conditions are executed. Recent rule-based systems consider advanced adaptation rules to support complex context-of-use situations (Yigitbas et al., 2020). However, despite the advantages of rule-based systems (flexible, easy to integrate, support context etc.), their responsiveness for context changes is limited. One of the main challenges in a rule-based system is that adaptation rules have variable precision and usefulness for end users. Thereby, monitoring and adjusting the rules helps in maintaining the precision of adaptation of the system and user satisfaction.

Mostly adaptation rules are static, and their modification is costly or impossible. However, AI-powered adaptations are more context-driven and responsive. Intelligent systems are intended to make decisions based on the real context as well as user information and user interactions. For instance, ML technology goes further and allows the systems to learn from the end-user interactions, and feedback, and make adaptation decisions accordingly.

Further, previous adaptive systems implementation has a flexibility issue since the adaptation code is mostly entwined with the functional core of the application. However, there is a need to design loosely coupled systems allowing the support of new context-of-use, and then extend the functional core and re-adapt the generated UIs accordingly. We argue that separating the UI code from the core logic of the application would improve flexibility and speed up the development of adaptive behavior. Further, with respect to existing approaches dealing with UI models at runtime, there still lacks a common understanding and a suitable methodology for the definition of runtime models. The commonly raised questions are about the connection of the models of design time to runtime architecture, beside synchronization and validation of adaptation decision on models at runtime.

This work involves studying how to implement and manage novel interactions in intelligent interfaces by capitalizing on ML techniques and user feedback. The focus of TADAP proposal is to address the context-awareness requirement in the runtime application definition. Generated UI should be smartly reactive and provide a suitable and optimized interaction scenario depending on context-related events. TADAP is particularly interested in the UI generation from the task model, considering user-centeredness, and investigates the application of ML techniques in the support of context-aware adaptation. TADAP operates different types of user feedbacks (implicit and explicit) to learn, validate, and decide adaptation. We assume that monitoring user interaction increases the system predictability to perform adaptations fitting the end-user expectations. In addition to reducing the potential for not expected adaptions and user frustration, which lead to the interaction downfall.

The characterization of the stated approaches is presented in (Table 1). We identify considered abstraction levels for the UI specification according to the spinal column of model-based UI (Task, Abstract, Concrete, and final). Then applied adaptation is characterized in terms of type (adaptive and/or adaptable) and execution time (runtime or design time). Moreover, as user involvement is an ultimate requirement for adaptation, we identify the support of end-users during the whole adaptation process through feedbacks. Both implicit and implicit feedback exhibits different characteristics of users' preferences with both pros and cons. The feedback allows evaluating the adaptation decisions taken rather than instructs by giving the correct choice. This is what creates the need for active exploration of users' reactions through their explicit opinions besides implicit one and user behavior.

3. The TADAP method

Quite a few of existing tools involve adaptivity within a model-driven engineering perspective by rendering interfaces on multiple types of platforms, or by considering specific user capacity and disability. TADAP proposal distinguishes by the particularity of imposing a specific reflection on the architectural model of adaptation in order to guarantee a harmonious integration between the UI adaptation and the UI generation at runtime.

(1) The system is intended to produce an adaptive UI, considering the user feedbacks beside the context of use. This feedback will be gathered implicitly and explicitly via user intervention, behavior and interaction history, then must participate immediately in the customization of the interface.
(2) Furthermore, TADAP provides a controllability feature-allowing the user to customize the UI via an explicit adjustment of adaptation parameters. Afterward, we present an extensive description of TADAP (Task-Based Adaptation) depicting a 'smart' task-based UI generator enabling extensive (user-) and (system-) initiated adaptations at runtime.

Table 1. Characterization of existing approaches regarding supported models, adaptation strategies, AI/machine learning use and feedbacks support.

Approaches (●) supported (◐) Partially supported (○) not supported (-) not specified	**Supported UI models**			**Adaptation**		**AI / ML**	**Runtime**	**Feedback support**	
	Abstraction levels	Final	Context	Adaptive	Adaptable			Implicit	Explicit
Trident-1995	●	-	●	●	-	●	○	○	○
TIMM-2000	◐	-	◐	●	-	●	●	●	○
Roam-04	◐	●	◐	●	-	○	●	○	○
Dynamo-Aid-04	◐	●	◐	●	-	-	●	○	○
Supple-2006	◐	●	-	●	●	○	●	○	○
ADUS-2007	◐	●	-	●	-	●	●	●	○
Breiner-2009	◐	●	-	●	-	○	●	○	○
MARIA -09	◐	●	◐	●	-	○	●	○	○
Blumendorf-10	◐	●	-	●	-	○	●	○	○
Criado-12	◐	-	-	●	-	○	●	-	-
MYUI-12	◐	-	-	●	-	○	●	●	●
Egoki--15	◐	●	●		●	○	○	-	-
Wisel-15	◐	●	●	●	●	●	●	●	●
CEDAR-16	●	-	-	●	●	○	-	○	●
AUI-UXA--18	○	●	●	●	-	○	●	●	●
MoCaDIX-19	●	●	◐	◐	-	○	○	○	
TADAP	●	●	●	●	●	●	●	●	●

Table 2. ANOVA test results on the average rate of Workload consuming.

ANOVA table	SS	DF	MS	F (DFn, DFd)	P value
Treatment (between columns)	1142	3	380.5	F (3, 44) = 3.136	P = .0348
Residual (within columns)	5338	44	121.3		
Total	6480	47			

Table 3. ANOVA parameters.

ANOVA summary	
F	3.136
P value	0.0348
P value summary	*
Are differences among statistically significant? ($P < .05$)	Yes
R square	0.1762

Table 4. ANOVA test results on the average rate of workload consuming for explicitly and implicitly adapted sessions.

ANOVA table	SS	DF	MS	F (DFn, DFd)	P value
Treatment (between columns)	986.7	2	493.4	F (2, 49) = 4.378	P = .0178
Residual (within columns)	5522	49	112.7		
Total	6509	51			

3.1. The supported context

Given that design-time generated UIs fail to provide the required flexibility, it is required to consider the evolution of the context data and to involve end-user during the time of use. The principal challenge is to ensure that adaptation decisions are computed dynamically considering the actual

Table 5. ANOVA test results on the average rate of Workload consuming for explicitly and implicitly adapted sessions.

Tukey's multiple comparisons test	Mean Diff.	95% CI of diff.	Significant?	Summary
NotAdapted vs. ImplicitlyAdapted	6.660	−2.175 to 15.50	No	ns
NotAdapted vs. ExplicitlyAdapted	−3.256	−12.98 to 6.466	No	ns
ImplicitlyAdapted vs. ExplicityAdapted	−9.917	−18.36 to −1.472	Yes	*

context of use circumstances and that UI changes are executed at run-time without interruption.

The definition of the context of use involves different parameters that have an impact on the system use such as place, device, time, tasks, interaction history, user preferences, and other social aspects. Artificial intelligence and ML techniques support the consideration context parameters for adaptation (Genaro Motti et al., 2012). TADAP focused on a set of parameters for the adaptation purpose such as; (1) the interaction platform, which is involved in terms of screen size, this parameter will contribute the adaptation decision to avoid encumbering GUIs. (2) The Task model in terms of completeness, task dependency and redundancy as well as the interaction history and task fulfillment. Further, it considers (3) the end user preferences as the main adaptation feature to sustain UI usability. User preferences are gathered during interaction through their feedbacks expressed explicitly and implicitly (Mezhoudi et al., 2015, Genaro Motti et al., 2012).

Accordingly, TADAP recognizes context variations <platform, user preferences, tasks>, identifies appropriate adaptation decisions, and generate new adapted UI during the same interactive session.

The runtime context-awareness is carried by the concept of 'at run-time triggered reification process' started on the user or the system request. The UI is then re-reified partially, only not accomplished tasks, taking into account context changes. New adaptation decisions will be applied for the following screens using the partial reification processes ensuring local adaptation. Despite the 'locality' of adaptation, previous containers (displaying performed tasks) are supported in adapting and designing the next one. TADAP architecture aims at avoiding the adaptation of previously filed actions while creating the next container. Only tasks that appeared in previous containers, but not accomplished will be considered for adaptation and displayed in the next reified UI container. Previously displayed UI container keeps their initial setting to ensure coherence and avoid user's disruption. Giving that the reifications process based on adaptation decisions results in plenty of options, the selection of the appropriate adapted solutions need to be guided by quality measures. To handle the discrimination of a better solution, we define different scores that are computed for each result in order to find the best one according to an assigned score function.

Figure 1 depicts TADAP adaptation scenario through the partial reification concept. The process points out a real-time adaptation using partial reification triggered during one interactive session. Initial generated FUI 1 support mainly the platform. Then once the interaction is started, the user-action-prediction module is triggered and a partial reification of an adapted FUI2 will take place considering new context data<tasks, user preferences>. Adaptations are availed in the abstract UI level 'Abstract Container2′.

To ensure such adaptation we proceed to Markov Model for the user behavior monitoring and the prediction of user behavior. We can resume the process of user behavior prediction with the Markov chain in three steps; first creating and monitoring sequences of actions (tasks), then learning an N order Markov chain model (or establishing all the order from one to N), and finally predicting the next most likely user action in view of his interaction history. Accordingly, the adaptation approach outlines the advantages of:

- Involving agents: System, User.
- Learning system (Adaptation process takes into account "nearly" directly the evolution of the collected data and the context of use)
- Enabling and facilitating controllability
- Enhancing the granularity of adaptation

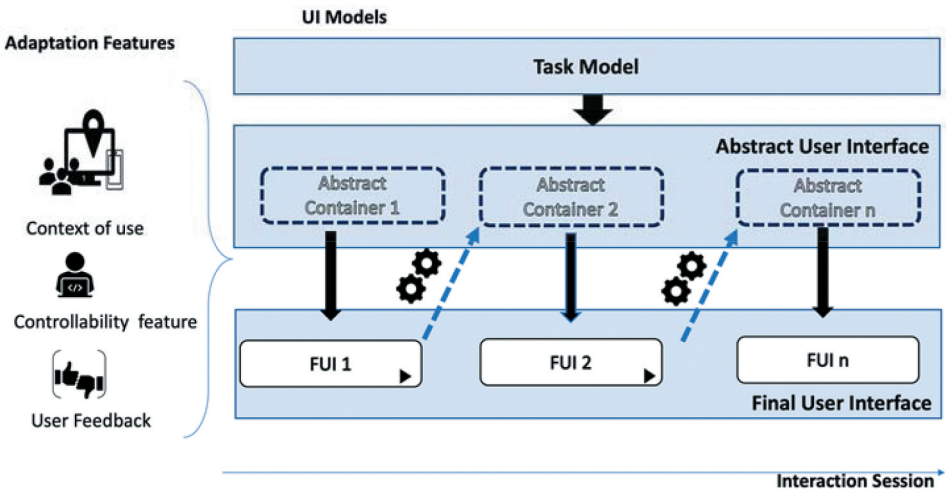

Figure 1. TADAP adaptation scenario.

3.2. Runtime adaptability: The controllability feature

TADAP supports user controllability on the UI and the whole adaptation process and affords the interaction with a controllability feature. A controllability feature is an extension of the final UI, where a scoreboard of reflected adaptation parameter is displayed allowing the adjustment of their relevance for adaptation. Parameters control consists of adjusting the weight attributed to each sub-score to balance their importance in deriving the final score. The considered parameters are determined by the task model completeness, monitoring user behavior, feedbacks, graphical appurtenance of widgets, and the platform weight. Accordingly, the controllability feature displays a list of tunable parameters, allowing the end-user to define the weights associated with each sub-score in order to balance its importance in the final score. Considered parameters revealed in Figure 2 are defined as follows:

- The accuracy of considering the completeness of the task model
- The accuracy of user guidance by behavior monitoring, which is used for the score of actions given the user behavior prediction (u.b.p.).
- The level of consideration of Feedbacks.
- This locality provides an opportunity to consider or not the content of the previous containers in the designing of the next one.
- Concerning actions that appear in previous containers, but which are not filed.
- Graphical connection between widgets implementing actions belonging to the same parent task in this reification process.
- The platform weight that defines the maximal number of elements displayed on the screen.

On the left-bottom border, a preview of the best-scored AUI models is shown. By moving the mouse on these buttons, the preview is displayed in the centric frame replacing the scoreboard. A simple click on the preferred preview triggers the reification of associated AUI to end the adaptation.

3.3. Runtime intelligent adaptivity: Learning and predicting user behavior

The main benefits of gathering interaction data are making adaptation decisions and helping users to better realize their tasks by predicting their behavior. Various Machine Learning techniques were deployed in different ways to support predictive approaches. It determines future users' actions by considering the user profile or model that links information about the user or the task to expectations concerning behavior. Markov chain was widely used for several prediction related issues. Likewise, TADAP deploys Markov models for predicting the action a user will take next given already performed sequence of actions. Markov models are represented by three parameters < A, S, T > Where:

A: set of all potential actions that can be accomplished by the user;

S: set of all potential states established by Markov model;

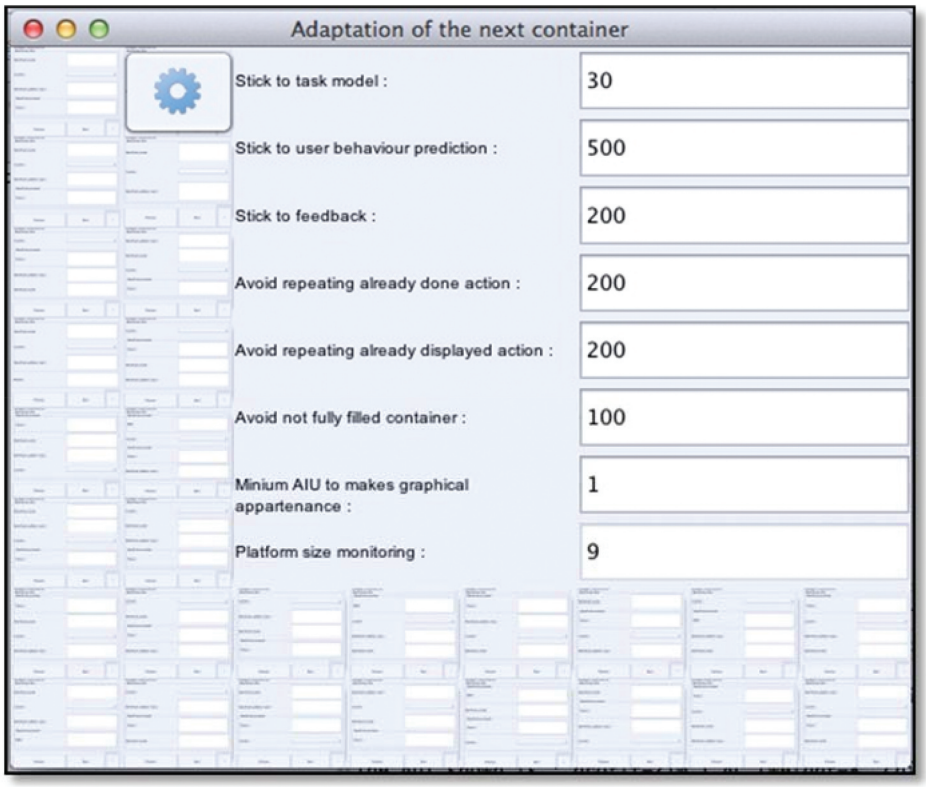

Figure 2. GUI associated to the controllability feature.

T: Transition Probability Matrix (TPM),

where tij: = probability (Performed action = j) when process state = i.

In a more complicated model, the predictions are computed by looking at more than one action performed by the user. And then the approach is generalized to the Kth-order Markov model, which computes the predictions by looking at the last K actions performed by the user. However, these higher-order models intensify the weaknesses associated with high state-space complexity, reduced coverage, and even poor prediction accuracy. To address this weakness, the Longest Repeating Subsequence (LRS) was combined with a Markov chain is used in order to decrease both spatial and temporal complexity. These complexities will be more important if we only use the Markov chain with entire sequences. According to (Mitrovic et al., 2005) the accuracies are quite the same in both cases. The Longest Repeating Subsequence (Pitcow and pirolli 99) is defined as: (1) The subsequence is a set of consecutive actions (follow each other's in time) from a sequence, (2) The subsequence is repeated more than T time in the sequences (generally T = 1), (3) We only keep the longest subsequence (the LRS that are contained by bigger LRS can be pruned).

In that order, combining both methods result in a quite intuitive technique that fits well with the problem of "User behavior prediction" which aims to predict potential actions given previously accomplished ones. Furthermore, in our case there is no need for various types of features to predict the next actions (like the case for weather prediction, speech recognition etc.), only "actions" feature type is considered. Accordingly, this prediction does not need complex optimization techniques like Gaussian or Neural Network with gradient descent at the learning step. The prediction time (use the model, query prediction based on history) is quick since we just have to take the higher probabilities given the history. Further such prediction technique allows us to pass up the problem of contradictory rules, widely faced by rule-based approaches.

TADAP uses the user behavior prediction to improve the arrangement of abstract interaction units at the abstract user interface level. Then all data collected by implicit feedback are used into a tool called user behavior prediction. This tool uses a machine learning technique based on statistics to predict the next action(s) that will be accomplished by the user given the previous ones he filled. TADAP carries out prediction via a UserActionPrediction class which can be seen as a dynamic extension of the context of use where the data are processed to extract more useful data. This class needs an instance of "ActionMonitoringDB" as "raw material" and also takes as parameter the Markov order.

Accordingly, the process of user behavior prediction with the Markov chain considers probabilistic and statistical models rather than deterministic rules. It consists of three steps:

- First, generating and monitoring sequences of actions.
- Then learning an N order Markov chain model,
- Finally predict the next most probable action of the user thanks to its history of immediate action.

3.4. Runtime intelligent adaptivity: Modeling cost functions

As was pointed out, the differentiation of quality of generated interfaces is determined through a cost function, which provides quantitative metrics to enhance both the consistency and optimality of the selected solutions. Accordingly, it should be a common approach in the establishment of relevant UI in different abstraction levels. For TADAP, defined scores are computed at runtime to be an input to determine the next adaptation decisions. First considered score gauging a UI, labeled 'ContentScore'. This metric concerns composite abstract Interaction Unit (IU) (containers) without considering the arrangement of simple IU inside. A clustering algorithm groups objects (action) according to their features in clusters basing on similarities.

A clustering algorithm minimizes the distance intra-class (distance between actions composing a cluster) and maximizes the distance inter-class (between the clusters). In our case, the clustering algorithm identifies UI containers and compute similarity regarding the minimum distance to reach a common parent in the task tree (ai,aj), this score increases when the actions of the container are close together hierarchically in the task model. Then the score could be computed considering the structure of the Task Model (TM), the users' feedback ranking the solution and the cohesion of prediction.

A first variant of the ContentScore is computed according to the TM structure in terms of tasks hierarchy, relationship and weights. The measure is determined as follows:

$$ContentScore_{TM}(CompoundIU) = \sum_{ti,tj}^{Tasks} (TaskWeight/Min(DistanceToReachCommonarent(ti,tj))) \quad (1)$$

Where: t: subtasks of a common parent

Taskweight: value assigned by considering task type

DistanceToreachCommonParent(ti,tj): distance between nodes to reach the common Parent of ti,tj.

An alternative score of actions more complex is regarding the user behavior prediction (u.b.p.) is defined. The user behavior prediction module is not formed to compute the "container content score" since this technique takes by definition the ordering of the actions in account to evaluate the next most probable actions. There is thus an underlying notion of a sequence. To meet this requirement and remove the influence of the ordering of actions, we proceed to simulate all possible history and compute all probabilities of the next task, and then we keep the well-ranked probability for each action. Formally the score of action is computed as in formula (2). Then the content score is calculated according to the formula (3).

$$OrderIndepProbability(CompoundIU) = Max(S \in SimulatedSequences)(\prod_{(t,h)\in S} P(t\backslash h) \quad (2)$$

$$ContentScore_{Prediction}(CompoundIU) = \sum_{t}^{tasks} (OrderIndepProbability(t) * UbPfeatureweight) \quad (3)$$

Where: t: subtasks of a common parent, h: history of fulfilling task, UbPfeatureweight: weight of user behavior prediction.

An additional relevant metric is the Ordering score. Which considers mainly the position of the action inside the container; in consequence, it's more restrictive and easier to compute regarding user behavior prediction. We have just to consider the ordering of the actions inside the AUI model and consider their positions. The score is then more restrictive than above. The score of actions given the task model is computed in the following way:

$$OrdringScore_{TaskModel}(Action) = (i * taskmodelfeatureweight | max\_iactions[0,i].sublistOf(DFS(tm))) \quad (4)$$

Where DFS(tm) generates a sequence of actions by exploring the task tree in depth first search. Thus, the more the sequence of actions inside the container respects the order defined by the task model, the more the score increase. As well as the content score, different variants of 'Ordering Score' could be computed according to the considered feature. For instance, regarding the Task Model, an AUI could be evaluated for the degree of appropriateness and conformity to the task's structure regarding both order and hierarchy. The score variant of actions given the user behavior prediction follows the next formula:

$$OrdringScore_{Prediction}(Action) = \sum_{a,h}^{fulfilledactions+actions} P(a|h) \quad (5)$$

The main advantage of a clustering algorithm is its scalability for considering new characteristics as new dimensions, besides the possibility of weighting different features in order to discriminate valuable one according to their relevance. Such scalability supports continuous adaption and improvement through data preparation without the need for designer input to adjust adaptation rules.

For an easier understanding of the score computation module, a functional analysis is illustrated in Figure 3. All computed scores are assigned to an object instantiated from the Score class to keep track of the subscore for debugging or analysis purposes. A subscore is a part of the score and it is computed by using a specific feature, such as feedback or behavior prediction.

3.5. Model-based runtime adaptation: Reification

In this section, we put forward a simulation of the adaptation process execution (see Figure 4). We consider a simple task model as input, operating the reification and triggering the adaptation process.

The task tree is shown in Figure 4a. The execution of the reification module invokes the method "triggerAdaptation()" in charge of adapting and generation the first container. By the same trigger, all possible partial AUI are reified and scored via the method "calculateScore()" detailed above. The high scored AUI is reified and displayed to the user. Next Figure 4b illustrates a partial simulation of the AUI reification for the mentioned task tree.

The simulation describes different potential AUI and computes their scores. The highest scored AUI is reified into FUI and displayed for the user (see Figure 4c). In our case, the second option was selected. The user should be able to trigger the controllability feature for handling the adaptation.

4. TADAP experiment

TADAP proposal is implemented in Java and provided with a simple module for final user interface (FUI) reification in order to show results. A simple task tree sample of banc transfer operation steps and the original interface generated during the first session before gathering the user feedbacks are shown in Figure 5.

4.1. Method

Then we suppose a list of executed sequences throughout diverse interactive sessions. Afterward, a new interface version is shown according to the new adaptation setting. As the system is expected to monitor all accomplished sequences, and implicitly gather additional contextual data affording the user behavior prediction. We execute five different sequences

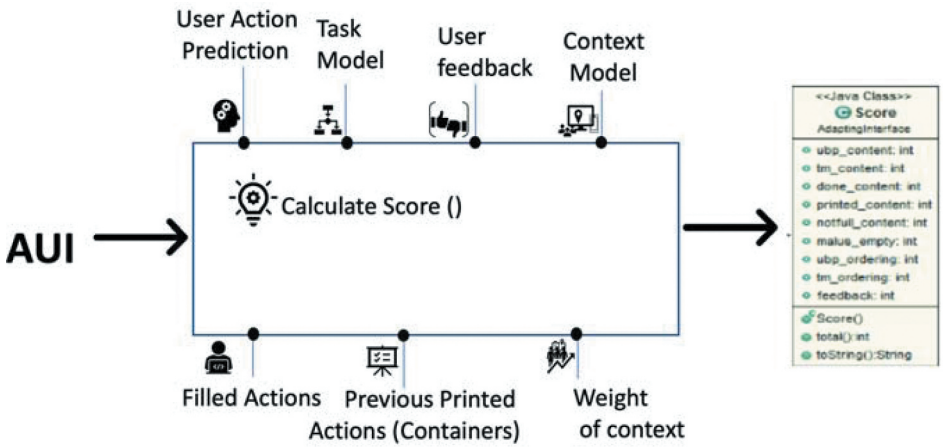

Figure 3. Input -Output of calculating score module.

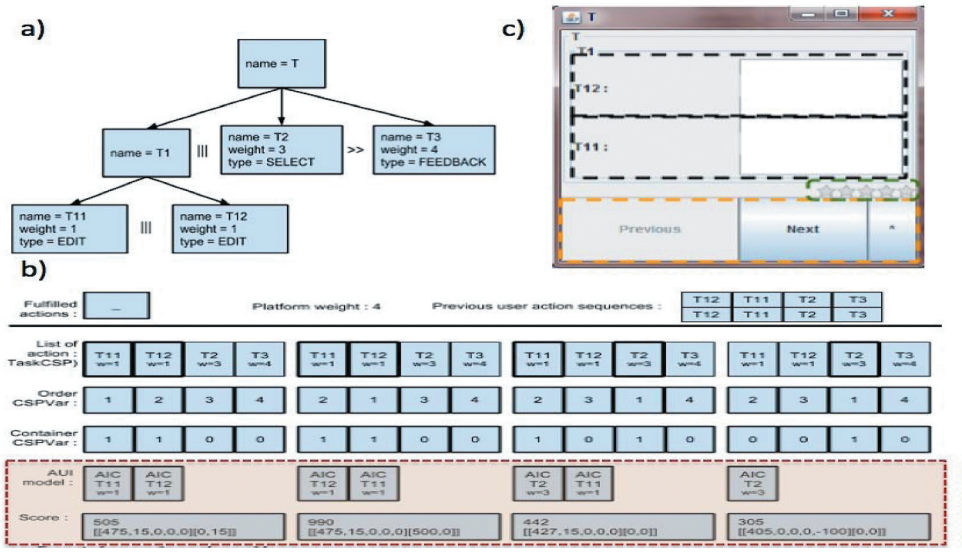

Figure 4. (a) Simple Task Model. (b) The simulation of the reification process. (c) The displayed interface.

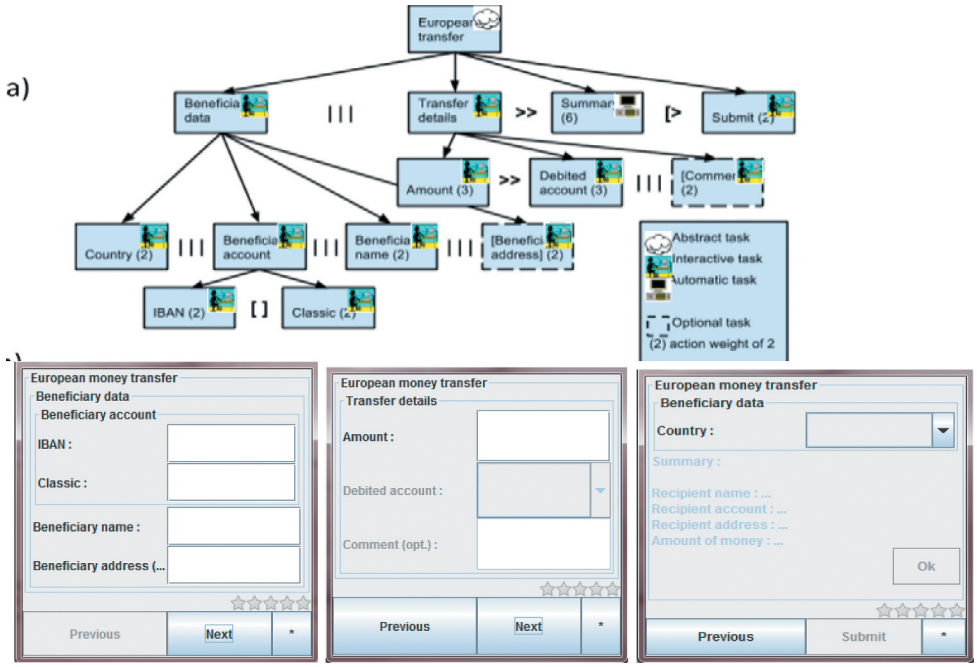

Figure 5. (a) The MoneyTansfer Task model. (b) First autonomous adapted UI: (user-feedback independent).

to perform both classic and IBAN transfers by changing the order of filling data. Then new adapted UI versions are reified in accordance with the partial reification concept depicted above. The reification module classes manage two reification methods, one dealing with the whole interfaces (getModels()), and one other dealing with partial interfaces (getPartialModels()). Partial reification requires more data, specific to the level of abstraction.

These data belong to the implementation of the adaptation architecture and they are not needed in case of whole interface reification, thus they are not specified in the abstract class "AdaptingInterface". These data are obtained from a specialized class implementing the methods needed. The class is a particular architecture adaptation extending "AdaptingInterfaceArchII". This specialized class will receive the context of use and the task model initialized by the "main" method. The constructor will initialize the reification classes: "ReifiTMtoAUIM" and "ReifiAUIMtoFUIM". It will also initialize the "UserActionPrediction" tool to use it in the computing of the AUI scores.

Moreover, the system provides the user with the controllability feature allowing the adjusting and the supervision of the interface adaptation during the interactive session (Figure 2). Figure 6c illustrates a scenario for using the controllability feature.

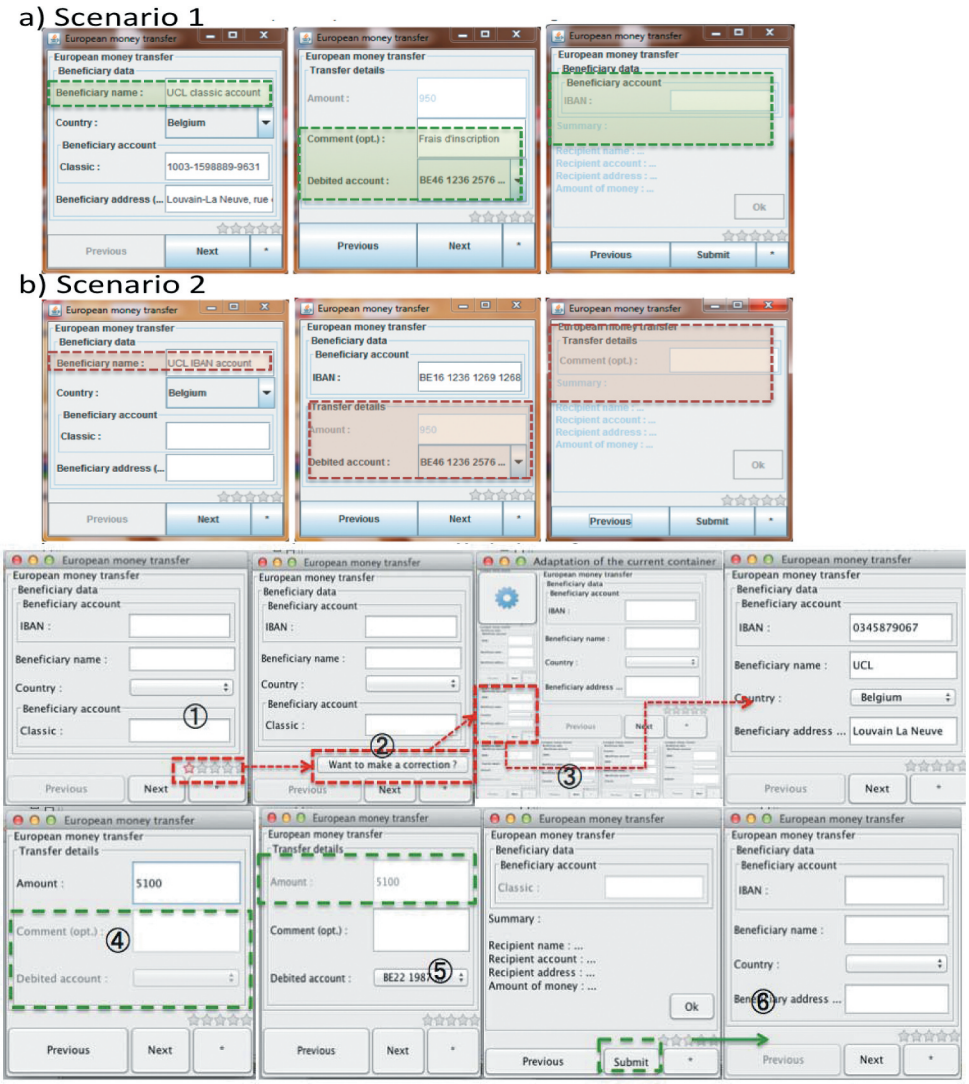

Figure 6. (a) Scenario accomplishing a classic transfer. (b) Scenario of an IBAN transfers realization. (c) Scenario invoking the controllability feature.

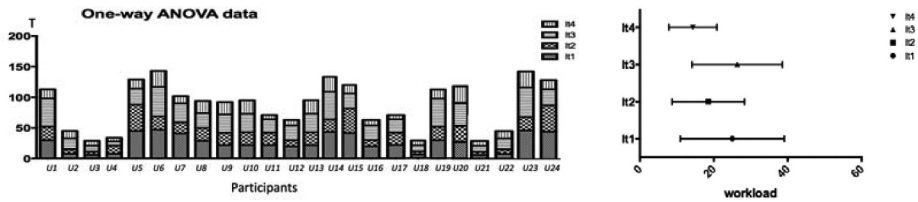

Figure 7. (a). Average Time-consuming by users. (b) Average Time-consuming tasks for different iterations.

- First, the user demotes the displayed interface by giving a low score and proposing corrections.
- Then the controllability feature is shown with a different alternative proposition for the current window.
- The user chooses a new interface description intended to be the most context-aware.
- Based on this choice, the system re-computes adaptations for the next containers.
- The system allows the user to evaluate the adaptation. This option is implemented simply via a feedback star rating widget. This feedback is used to answer at the open question "What do you think about the container?".

The widget can be viewed in all containers (Figure 7c). This kind of feedback is summative allowing a formative evaluation of the container.

4.2. The Users experience

According to Nielsen guide to usability testing; "the strength of the thinking-aloud method is to show what the users are

doing and why they are doing it while they are doing it in order to avoid later rationalizations" (Nielsen et al., 2002). We decided to consider elicited verbal data since we believed that this would provide us with valuable information on the interactive session condition.

This decision relies on the results of a recent research addressing the suitability of the concurrent think-aloud protocol to evaluate usability problems (Fan et al. 2019). Fan et al. (2019) concluded that verbalization and speech features alone are reliable features to identify usability problems. However additive values may be enabled by supporting additional modalities such as facial expression and body language.

Further, several works have demonstrated the effectiveness of the think-aloud protocol for usability testing. It was stated as highly useful for gaining an understanding of interactive information behavior (Holzinger, 2005; Mingming Fan et al., 2019; Nielsen et al., 2002; Sharon McDonald et al., 2012; Van Velsen et al., 2008). Van Velsen et al. (2008) recommend CTA for testing a working prototype and highlight its ability to support the evaluations that address both usability and adaptation output.

Experiment: During the study, 24 volunteer subjects (50% women, 50%men), from different countries (Belgium, Italy, Tunisia, Brazil, Syria, Algeria, France, Vietnam and China) and working in different domains such as mechanic, biology and computer sciences performed the experiment. The average age was 29 years, and all participants were eligible and familiar with the computer and they are all fluent French and English speakers.

Participants were asked to perform the Concurrent Think-Aloud CTA protocol following Ericsson and Simon definition (Ericsson & Simon, 1993).

It consists of verbalizing what they are doing as they are doing it. CTA is the predominantly used method in usability testing due to its performance and easiness (Sharon McDonald et al., 2012). We followed Ericsson and Simon's guidelines during testing sessions and did not probe or intervene except to remind users to keep speaking if they keep silent for a long period (Ericsson & Simon, 1993). However, they are not able to ask any questions about the observation, while still undertaking the task. Further, and in order to surmount the CTA issues with usability, we followed (Mingming Fan et al., 2019) recommendations and guidelines. Also, we have found it useful for the participant to show a short (50 second) demo of thinking aloud session when conducting a simple login scenario. In this demo video, the participant verbalized her interaction and feeling about the UI besides telling her action. The video minimizes the time required to introduce the experience and provides a concrete example of how to express their interaction with the system (Mingming Fan et al., 2019).

Each participant was asked to do four operations with the interface, supposed to be generated and re-adapted regarding his or her conduct during the interaction, along with audio recordings of the users thinking aloud while performing tasks.

The target is to assess the TADAP prototype's usability, first by observing the user behavior in performing tasks while operating the proposed adaptation approach. which allow to identify potential usability issues during performing the list of tasks. Further, the CTA allows to address and to evaluate the system in term of appropriateness of adaptation, User behavior, User performance, User experience etc. (Van Velsen et al., 2008)

Further, we focus on evaluating the interaction throughout analyzing the audio records to report the user experience and reflecting the user's performance when using Tadapt's generated interfaces. Participants were requested to describe their actions without giving explanations participants merely report how they go about completing a task.

4.3. Results

Once we have all participants' records, we start the analysis by identifying where, within it, there are users sounds. We have a measurement of audio amplitude against time, which isn't very easy to analyze.

In our evaluation, we focused on the user in order to detect potential usability issues. The main addressed concept is user performance when accomplishing the test. The literature defines user performance in terms of Learnability, acceptability, and Ease of use (Van Velsen et al., 2008). The ease of use is mainly reflected by the number of successful phases completions (task success).

Learnability and adaptation-appropriateness are revealed by task completion, durations when executing tasks.

By the same token, Observing the user behavior allows detecting usability issues and faced problems; through observation, through verbalization or through a combination of observation and verbalization. The evaluation of end-users' performances with TADAP showed an effective full completion since all participants were able to complete all phases however their performance varies. The next step consists of comparing user performances through verbalization. We began by identifying periods where there are certain efforts when users are talking. Based on the setting of the experience' protocol described above, the verbalized interaction enables us to process the operative interaction time and detect where within the audio there is a sound (user is interacting). We trim silence from records; this function consists in determining a minimum volume (amplitude) threshold and duration (<20 dB for more than 2 seconds) and then simply does a volume analysis of the waveform looking for areas that meet these criteria.

As we said above, participants were asked to accomplish four "Money transfer" scenario resuming the two phases of the experiment.

The first phase: consist of two interactive sessions accomplishing the full transfer task. It is aimed at verifying that the subject could recognize changes in the UI depending on the followed sequence when establishing the task. During the first 2 iterations, the user doesn't intervene in adapting the UI. TADAP adaptive behavior is intended to generate user interfaces that are expected to be the fastest for a particular participant to utilize. At this phase, the supported customizations regard the interface arrangement that varies depending on human errors e.g., displaying the next container without filling all information; and also depending on the order of tasks. Such adaptations decisions are implicitly invoked. For a typical scenario where forms are fulfilled in a systematized way, the user will not perceive

differences. *Second phase*: The second step of the experiment consisted in using explicit feedbacks to manage adaptation within the same interactive session. During the third iteration, we asked users to provide explicit feedback by ranking the UI and/or using the controllability feature when realizing the transfer scenario. The goal was to allow each subject to explicit his preferences for adaptation in order to reduce the interaction load during the last iteration. In this step, we were especially interested in minimizing the period of the final iteration. We selected the "Money transfer" scenario since it is a simple daily life task. This choice is aimed at reducing learning difficulties that participants may experience with a new interaction scenario. Together with the illustrative video, the difficulties became insignificant the user's learning curve is almost flat.

The trend is clear that the rate of time-consuming is lower in an adapted interface. To make this observation statically significant, the ANOVA 'One-way analysis of variance' test is conducted on the value. The following table shows the result of ANOVA test produced by the statistics tool GraphPadPrism (Table 3).

We were expecting subjects to be aware of variances during interaction according to the adaptation decisions deployed during the different phases of the experience (i.e the workload during second and fourth iterations is lower than the first one). There was a significant effect of adaptations on workload and time-consuming at the p <.05 level for the three conditions [F(3, 44) = 3.136 p = .0348] (Table 2).

Since significant variance between the performance (workload) means of different iterations was showed. The rate of task's duration was significantly different, and we can confidently state that workload decrease with adaptations accordingly TADAP approach contributes the ease of use' usability factor.

In a second phase, we analyze the same set of data by considering three groups of interaction session; the first group corresponds to interaction session with not adapted UI, the second considers the session where adaptations where triggered explicitly, and the last group consider only the session where the adaptation was performed in an implicit way. We expect a significant variance between performances during an interactive session with implicitly adapted UI, session without adaptations and session where adaptation was invoked explicitly. Similarly, there was a significant effect of adaptations on performance in term of workload and duration at the $p < .05$ level for the three conditions [F(2, 49) = 4.378 p = .0178] (Table 4).

Because we have found a statistically significant result in this example, we needed to compute a post hoc test. We selected the Tukey post hoc test. This test is designed to compare each of the different iterations. We compare the means of three interaction groups (not adapted, implicitly adapted and explicitly adapted) (via one-way ANOVA, multiple comparisons considering iteration 1 as the control group and alpha = 0.05 (Table 5).

Post hoc comparisons using the Tukey HSD test indicated that the mean score for the implicitly and explicitly adapted interactions was significantly different however no significant difference between adapted and not adapted UI (Figure 8).

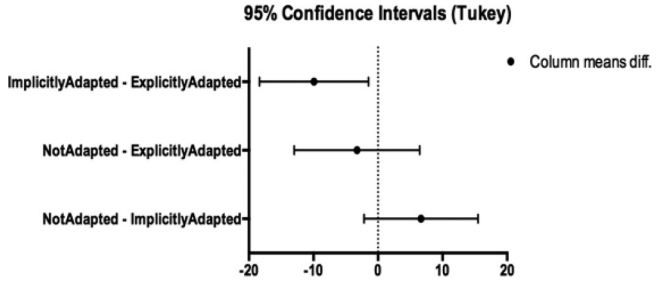

Figure 8. Tukey: difference between group means.

4.4. Discussion

In (Figure 7) we show the detail of time-consumption (duration) of the test steps; we can observe that the use of both adaptation approaches reduces the total time spent by users to complete the task. The average performance (reflecting the user workload, time-consumption) is higher for the third iteration since an additional task was executed when adjusting the UI throughout the controllability feature.

The predictive algorithm, used by TADAP, utilizes the data obtained from monitoring interaction sequence to predict the users' next most probable action. The resulting UIs are generated faster and consequent windows are adapted immediately with only one reification. In the consecutive version, with predictive features, interface elements are defined according to the monitored user interaction sequence.

We note that Pearson's r for the correlation between iterations was close to 1, which is called a positive correlation. In our example, our Pearson's r mean was of 0.7 was positive. Likewise, the results were positive for the correlation between not-adapted and adapted sessions (Table 6). We conclude that the amount of time-consuming during iterations is proportional to the subject due to the variety of profiles and skills.

In the second phase, we can observe that due to the controllability features the interactive session was longer (Figure 7b), however better results are obtained for the following session as more accurate adaptations were applied based on explicitly gathered users' feedbacks. Greater improvement might be obtained by adjusting the parameter setting that produces the best results when customizing adaptation. Otherwise, it is clearly visible that during the fourth iteration all participants were faster in fulfilling the test (Figure 7b). Compared to rule-based approaches, ML based adaptation are flexible and responsive. This reflexibility supports continuous improvements of interaction performance since adaptation decisions taken by the system using the learning technique. ML techniques allow the system to learn the best adaptations in various situations without decreasing the system's real-time quality.

Further, we found that there is an opportunity to achieve significant improvement by mixing adaptive and adaptable behaviors during interaction with individuals. Both adaptability and adaptivity focused on pursuing users' interfaces that

Table 6. Correlation table of one-way ANOVA data.

Correlation Tabular results	NotAdapted vs. ImplicitlyAdapted	NotAdapted vs. ExplicitlyAdapted
Pearson r	Y	Y
r	0.7783	0.7110
95% confidence interval	0.3981 to 0.9303	0.2631 to 0.9068
R squared	0.6058	0.5055

generate personalized GUIs instead of treating all users the same. According to results obtained in the experiments, it can also be concluded that the TADAP is effective in identifying user preferences by both adaptation approaches. These findings reinforce once again the need for accurate prediction of user preferences and appropriate adaptations.

Further, we assume that the perceived improvements of interaction are due to the responsiveness of adaptation decisions. An ML-based adaptation allows the adjustment of adaptation with regards to change contrarily to the rules-based system adaptation that offers a kind of "fixed" intelligence. However, ML-based adaptation decisions are not entirely clear compared to rule-based adaptations. Further work could be elaborated at different levels of evaluation of the context support and adaptation-context suitability.

5. Conclusion

This work presents a model-based environment for the generation of the context-aware graphical user interfaces at runtime. A partial reification technique is introduced to support MB GUI adaptation at runtime. It consists of the adaptation of potential abstract user interface (container) and then predicting an optimized adjustment of the next container by capitalizing on machine learning techniques and user feedback.

At the analytical level, we have classified the literature of the model-based UI reification algorithm and discussed the relevance of prospecting AUI generation. At the methodological level, we introduced different metrics used to assist and control the generation of the AUI. Then, we detailed the reification process and availed ML technique. We use a machine learning technique based on statistics in order to predict the next user-task(s) that will be accomplished by the user given the previous ones he filled. Accordingly, adaptations were based on explicit and implicit feedback besides the user behavior prediction. The considered context of use covers all data specific to the user and some platform descriptors. The main data specific to the platform is the size of the displayed window for the interface. Generated UIs adjust their dimensions based on the device that accesses them in a responsive way. Concerning feedback, all the data captured from interaction might adjust the container concerned by the feedback, or influence more globally the adaptation process. Moreover, we may consider the user behavior prediction to predict the content of the widget based on the previous data filled by the user. This kind of tool looks like an enhanced auto compilation tool. It could be interesting to deal with some kind of data processing/filtering adaptation mixed with container content adaptation. Right now, we are generating all the possible solutions, and then we select the bests according to the score. It would be efficient to improve an existing solution with a local search technique in order to reach a better one without generating all the solutions by brute force as we did. A local search algorithm can be deployed for that. The algorithm was validated via a case study and results were evaluated and discussed according to the Layout appropriateness metric and user evaluation. Performance and usability results demonstrate that there are significant benefits through predicting future states of user interaction. The results of the usability survey show that users perceive a system more useful when it follows their preferences.

Acknowledgments

The authors would like to thank the anonymous referees for their valuable comments and helpful suggestions. They also thank all the participants involved in the studies and experiments.

ORCID

Nesrine Mezhoudi http://orcid.org/0000-0002-6580-6369
Jean Vanderdonckt http://orcid.org/0000-0003-3275-3333

References

Akiki, P. A., Bandara, A. K., & Yu, Y. (2016). Engineering adaptive model-driven user interfaces. *IEEE Transactions on Software Engineering*, *42*(12), 1118–1147. https://doi.org/10.1109/TSE.2016.2553035

Alpaydin, E. (2004, October). *Introduction to machine learning*. The MIT Press.

Barber, D. (2010). *Bayesian reasoning and machine learning*. Cambridge University Press.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Blumendorf, M., Lehmann, G., & Albayrak, S. (2010). Bridging models and systems at runtime to build adaptive user interfaces. In *Proceedings of the 2nd ACM SIGCHI symposium on Engineering interactive computing systems (EICS '10)*. Association for Computing Machinery, New York, NY, USA, 9–18. https://doi.org/10.1145/1822018.1822022

Bodart, F., Hennebert, A. M., Leheureux, J. M., & Vanderdonckt, J. (1995). A model-based approach to presentation: A continuum from task analysis to prototype. In *Interactive Systems: Design, Specification, and Verification* (pp. 77–94). Springer, Berlin, HeHeidelberg.

Breiner, K., Bizik, K., Rauch, T., Seissler, M., Meixner, G., & Diebold, P. (2011, July). Automatic adaptation of user workflows within model-based user interface generation during runtime on the example of the SmartMote. In *International Conference on Human-Computer Interaction* (pp. 165–174). Springer, Berlin, Heidelberg.

Breiner, K., Maschino, O., Görlich, D., & Meixner, G. (2009). Towards automatically interfacing application services integrated in an automated model-based user interface generation process. In: *Proceedings 14th international conference on intelligent user interfaces IUI 09*, Sanibel Island, Florida, USA.

Chu, H. H., Song, H., Wong, C., Kurakake, S., & Katagiri, M. (1955). Roam, a seamless application framework. *Journal of Systems and Software*, 69(3), 209–226.

Clerckx, T., Luyten, K., & Coninx, K. (2014, July). DynaMo-AID: A design process and a runtime architecture for dynamic model-based user interface development. In *IFIP International Conference on Engineering for Human-Computer Interaction* (pp. 77–95). Springer, Berlin, Heidelberg.

Criado, J., Iribarne, L., Troya, J., & Vallecillo, A. (2012, September). An mde approach for runtime monitoring and adapting component-based systems: Application to wimp user interface architectures. In *2012 38th Euromicro Conference on Software Engineering and Advanced Applications* (pp. 150–157). IEEE. https://doi.org/10.1109/SEAA.2012.27

Eisenstein, J., & Puerta, A. (2000, January). Adaptation in automated user-interface design. In *Proceedings of the 5th international conference on Intelligent user interfaces* (pp. 74-81). https://doi.org/10.1145/325737.325787

Ericsson, K., & Simon, H. (1993). *Protocol analysis: verbal reports as data* (2nd ed.). MIT Press.

Fan, M., Lin, J., Chung, C., & Truong, K. N. (2019). Concurrent think-aloud verbalizations and usability problems. *ACM Transactions on Computer-Human Interaction* (TOCHI), 26(5), 1–35.

Gajos, K. Z., Czerwinski, M., Tan, D. S., & Weld, D. S. (2006). Exploring the design space for adaptive graphical user interfaces. In *Proccedding of the conference on advance visual international AVI '06*, (pp. 201–208). NY, USA: ACM.

Gamecho, B., Minón, R., Aizpurua, A., Cearreta, I., Arrue, M., Garay-Vitoria, N., & Abascal, J. (2015). Automatic generation of tailored accessible user interfaces for ubiquitous services. *IEEE Transactions on Human-Machine Systems*, *45*(5), 612–623. https://doi.org/10.1109/THMS.2014.2384452

Genaro Motti, V., Mezhoudi, N., & Vanderdonckt, J. (2012).). Machine Learning in the Support of Context-Aware Adaptation. In *Workshop On Context-Aware Adaptation of Service Front-Ends (Pisa, 13/11/2012). In: Francisco Javier Caminero Gil, Fabio Paternò, Jean Vanderdonckt, Proceedings of the workshop on context-aware adaptation of service front-ends*

Ghiani, G., Manca, M., Paternò, F., & Santoro, C. (2017). Personalization of context-dependent applications through trigger-action rules. *ACM Transactions on Computer-Human Interaction (TOCHI)*, *24*(2), 14. https://doi.org/10.1145/3057861

Harvey Motulsky, Comprehensive Analysis and Powerful Statistics, Simplified, http://www.graphpad.com/scientific-software/prism/

Holzinger, A. (2005). Usability engineering methods for software developers. *Communications of the ACM*, *48*(1), 71–74. https://doi.org/10.1145/1039539.1039541

Hussain, J., Hassan, A. U., Bilal, H. S. M., Ali, R., Afzal, M., Hussain, S., ... & Lee, S. (2018). Model-based adaptive user interface based on context and user experience evaluation. *Journal on Multimodal User Interfaces*, 12(1), 1–16. https://doi.org/10.1007/s12193-018-0258-2

Mezhoudi, N. (2013). *User interface adaptation based on machine learning and user feedback* (Intelligent User OInterface IUI.13).

Mezhoudi, N., Khaddam, I., & Vanderdonckt, J. (2015) Wisel: A mixed initiative approach for widget selection. In *Proceedings of the 2015 conference on research in adaptive and convergent systems*, (pp 349–356). ACM.

Mezhoudi, N., & Vanderdonckt, J. (2015, March). A user's feedback ontology for context-aware interaction. In *2015 2nd world symposium on web applications and networking (WSWAN)* (pp. 1–7). IEEE.

Mitrovic, N., Royo, J. A., & Mena, E. (2007). Performance analysis of an adaptive user interface system based on mobile agents. In *Handbook of research on user interface design and evaluation for mobile technology*.

Mitrovic, N., Royo, J. A. A., & Mena, E. (2005). Adaptive user interfaces based on mobile agents: Monitoring the behavior of users in a wireless environment. In *I symposium on ubiquitous computing and ambient intelligence, Spain*. Thomson- Paraninfo.

Nielsen, J., Clemmensen, T., & Yssing, C. (2002). Getting access to what goes on in people"s heads? – Reflections on the think-aloud technique. In *Proceedings of 2nd NordiCHI conference, Aarhus, Denmark*, (pp. 101–110). NY: ACM.

Oppermann, R., & Rasher, R. (1997). Adaptability and adaptivity in learning systems. *Knowledge Transfer*, *2*, 173–179.

Paterno, F., Santoro, C., Mantyjarvi, J., Mori, G., & Sansone, D. (2008). Authoring pervasive multimodal user interfaces. *International Journal of Web Engineering and Technology*, *4*(2), 235–261. https://doi.org/10.1504/IJWET.2008.018099

Rosman, B., Ramamoorthy, S., Mahmud, M. H., & Kohli, P. (2014). On user behaviour adaptation under interface change. In *Proceedings of the 19th international conference on Intelligent User Interfaces* (pp. 273–278) .

Ruiz-López, T., Rodríguez-Domínguez, C., Rodríguez-Fórtiz, M. J., Ochoa, S. F., & Garrido, J. L.: Context-aware self-adaptations: From requirements specification to code generation. In: *Ubiquitous computing and ambient intelligence. context-awareness and context-driven interaction—7th International Conference, UCAmI 2013*, (pp. 46–53). December 2 –6, 2013. Carrillo, Costa Rica. Proceedings.

Sharon McDonald, H., Edwards, M., & Tingting, Z. (2012). Exploring think-alouds in usability testing: An international survey. *IEEE Transactions on Professional Communication*, *55*(1), 2–19. https://doi.org/10.1109/TPC.2011.2182569

Van Velsen, L., Van Der Geest, T., Klaassen, R., & Steehouder, M. (2008). User-centered evaluation of adaptive and adaptable systems: A literature review. *The Knowledge Engineering Review*, *23*(3), 261–281. https://doi.org/10.1017/S0269888908001379

Vanderdonckt, J., & Nguyen, T.-D. (2019). MoCaDiX: Designing cross-device user interfaces of an information system based on its class diagram. *Proceedings of the ACM on human-computer interaction 3*, (p. 17). EICS.

Yigitbas, E., Jovanovikj, I., Biermeier, K., Sauer, S., & Engels, G. (2020). Integrated model-driven development of self-adaptive user interfaces. *Software and Systems Modeling*, 1–25.

Yigitbas, E., Sauer, S., & Engels, G. 2017. Adapt-UI: An IDE supporting model-driven development of self-adaptive UIs. In *Proceedings of the ACM SIGCHI symposium on engineering interactive computing systems (EICS '17)*, (pp. 99–104). New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/3102113.3102144)

About the Authors

Nesrine Mezhoudi is an Assistant Professorat the IT Department at Imam Abdulrahman Bin Faisal University. Sheholds a Ph.D. from the Catholic University of Louvain, Belgium, and a master'sdegree in computer science from the University of Gabes. Her researchesinterests are Human-Computer Interaction, Intelligent User Interfaces,Model-Based approaches.

Jean Vanderdonckt is Full Professor of Computer Science at Louvain School of Management, Université catholique de Louvain,Belgium. He received M.Sc. degrees in mathematics, in computer science, and the Ph. D. degree from the University of Namur,Belgium. His research interests include HCI, engineering interactive systems, intelligent user interfaces, usability engineering.