IBIMA Publishing
Communications of the IBIMA
http://ibimapublishing.com/articles/CIBIMA/2018/281881/
Vol. 2018 (2018), Article ID 281881, 18 pages, ISSEN : 1943-7765
DOI: 10.5171/2018.281881

Research Article

User Interface Adaptation based on a Business Rules Management System and Machine Learning

Nadia Ghaibi[1], Olfa Dâassi[2] and Leila Jemni Ben Ayed[3]

[1]University of Sfax, Sfax, Tunisia

[2]University of Carthage, Tunis, Tunisia

[3]University of Manouba, Tunis, Tunisia

Correspondence should be addressed to: Nadia Ghaibi; nadia.ghaibi@gmail.com

Received date: 10 November 2017; Accepted date: 19 December 2017;

Published date: 22 October 2018

Academic Editor: Sana Bouajaja

Abstract

In ubiquitous computing environments, computers are embedded into everyday lives to provide information anywhere and at any time. Pervasive computing assists us in our everyday activities, operating invisibly in the background, regardless of our location or devices. Therefore, ubiquitous computing requires discovering new Human-Computer Interaction methods helping users to easily interact with ubiquitous systems whatever is the perceived context situation. However, designing context-aware applications that are able to adapt to context instability is a recurring problem that requires special attention from researchers in the HCI community. On the one hand, it is not trivial for a designer to specify how UIs should adapt, and on the other hand it is very hard to predict the context of use changes and accordingly to construct adaptive UIs that match users expectations. In this paper, we present a Model Driving Engineering (MDE) technique to design UIs that automatically adapt to the perceived context situation while the designer and eventually the end user still have full control over the adaptation during runtime. This technique is supported by a conceptual framework and a graphical tool that lower the threshold for designers as well as developers to design adaptive user interfaces, modelize context situations, edit adaptation rules and manage the adaptation process.

Keywords: User Interface, adaptation, accessibility, context of use, Model-Driven approach, Drools.

Cite this Article as: Nadia Ghaibi, Olfa Dâassi and Leila Jemni Ben Ayed (2018)," User Interface Adaptation based on a Business Rules Management System and Machine Learning", Communications of the IBIMA, Vol. 2018 (2018), Article ID 281881, DOI: 10.5171/2018.281881

Introduction

Emerging ubiquitous environments deliver products and services that are more and more sophisticated creating competitive challenges for computer engineers particularly the Human-Computer Interaction (HCI) community. Well-designed User Interfaces (UIs) could fail to satisfy end-users if they do not cover the unpredictable dynamic variation of the context of use. UIs should be accessed through several devices such as smart phones, notebooks and tablets and used by people with different skills and abilities. This implies the necessity of qualifying UIs with flexibility and adaptability to increase their level of accessibility and accommodate the context requirements.

During the last decade, several successful solutions have been proposed to deal with this issue. Some research studies like Zouhaier et al (2015), Peissner et al (2012) and Miñón et al (2013), address the problem at the design-time and aim to generate multiple variants of a UI for multiple target devices and users. Chen et al (2014), Criado et al (2015) and Mezhoudi et al (2015) focus on a fully dynamic approach where the adaptation behavior is performed at runtime. However, most of these solutions neither end-user nor designer takes part in the adaptation process in order to select relevant adaptation rule or strategy to be performed.

In our work, we adopt a Model Driven Development approach to generate Final User Interface (FUI) adapted to a specific context's situation. In addition, the adaptation specifications are integrated in the generation process of the Cameleon Reference framework (CRF), proposed by Calvary et al (2003), in order to ensure that UI consistency is preserved even when making changes at any level representation and regenerating the interface. We also take into account that the designer should be a full partner throughout the process. Therefore, our solution consists of a model-based toolkit offering graphic tools for:

- generating UIs from tasks level to the final one,
- managing context's situation parameters,
- managing adaptation rules
- launching adaptation process

Thanks to these tools, developers have full control over the adaptation process. They can configure the interface components according to the user preferences or capabilities. If the resulted interface does not satisfy the initial specifications, the process can be rerun with different parameters until the result is satisfactory.

On the other hand, Model Driven approaches aim to deal with the complexity of interactive systems and decrease the effort needed to develop UIs. They put emphasis on models with different levels of abstraction which can be obtained successively through model to model and/or model to code transformations. We exploit these benefits not only at design and generation stage but also to achieve context-dependent customization. Therefore, we define models to describe context components (user, platform, environment) as well as rules adaptation. This can be considered as one of the advantages of this work since rules edition was not addressed by the major of the existing approaches. In fact, in most cases, rules are hard coded or not extensible. Besides, it happens that the designer defines various rules for the same context situation. In this case, conflicts can occur and affect the consistency requirements of the UI or even disable other adaptations. By conflicts we mean situations in which adaptation behavior would have no sense following some operations that impact each other (for example, adding and then deleting the same component), or cause an inconsistent state for the system. To deal with this issue, we proceeded to consolidate our solution by a Business Rules Management System (BRMS) that

supports the execution of rules and conflicts management, an aspect that has not been deeply addressed in previous research works.

This article presents our approach to handle the adaptation challenge: first, we present DNA, our model-based toolkit to dynamically generate context-dependent UIs. We principally focus on the adaptation stage of the development process and we show how designers can easily define context situations, manage a database of adaptation rules, select rules that are suitable to current situation and finally launch the adaptation process. Second, generated UIs are at the same time adaptive: for any change in the context of use, related models are instantly updated and the FUI is regenerated. Through using DNA we have generated multi-language UIs (Java and HTML5 versions) and we have tested their adaptability based on a runtime context simulator.

The remainder of this paper is structured as follows: in section 2 we analyse some related works. Section 3 discusses our approach and describes the DNA toolkit and shows how it can be very useful for designers. Section 4 illustrates the adaptation process and explains how generated UIs can adapt themselves dynamically to the context variation. Finally, we conclude and outline our future work.

Related Works

Integrating Adaptation in the UI Development Process

Regarding adapted UIs to specific context's situations or to users with special needs, most works had integrated adaptation in the development process by transforming models changeable according to the context parameters.

Bouchelligua et al (2010) proposes an approach that allows the adaptation to the context of use based on parameterized transformation. Accordingly, each transformation requires as an input a context model and a UI model and generates as an output the adapted one. This work models context's elements and offers a formal description for UIs models based on Business Process Modeling notation (BPMN). Nevertheless, it supports adaptation only at Abstract and Concrete UI's levels and merely at design time. Besides, no tool is developed for this approach thus far.

Ditto for Zouhaier et al (2015), who proposes a generic solution for developing accessible UIs applications based on parameterized MDE-transformations. From an application model, which provides an abstract view of the user interface, and an ontology model that describes the accessibility context, the system applies a reverse engineering and series of transformations in order to adapt the AUI. Once done, the FUI is regenerated. The solution aims to incorporate the modality of interaction into adaptation process and provide the system with more flexibility to make it possible choosing desirable modality into the UI, but no tools supporting this approach have yet been developed.

Peissner et al (2012) propose MyUI that provides customized user interfaces that are adapted to a broad range of user's needs, different devices and various context conditions by adapting their presentation layout and modalities, and the navigation paths. Controllability is partially covered by the system in the sense that it asks the user to accept or reject the adaptation. MyUI also supports adaptation at runtime through an automated and rule-based UI generation process, which uses bundles of patterns supporting different user's needs or context conditions. According to the user's profile or the device profile, the most suitable pattern is selected and applied. Finally, this work is distinguished by developing a toolkit that supports developers in the creation of adaptive applications. It helps them specify the application properties, explore the available patterns and select those needed for the interaction situation.

Otherwise, in the research study of Miñón (2015), the adaptation process is based on a repository, which stores rules devoted for

people with disabilities. These adaptation rules are integrated into the development process at both design time and runtime. At the design time, the designer can select adaptation rule dedicated for the disability in question and the UI abstraction level and launch the generation process. At the runtime, the same procedure is followed but automatically without the intervention of either the designer or the end-user. Additionally, it should be noted that rules conflicts are resolved by ordering them by a granularity level assigned to each of them. By granularity level, authors of this work denote parts of the UI that can be affected by the adaptation such as the entire application, the presentation format, a group of elements or only a single element.

Akiki et al (2016) present Role-Based UI Simplification (RBUIS) as a system for implementing adaptive UIs by applying two types of adaption named feature-set minimization and layout optimization. The first one disables the UI components that are not required by particular end-users. Afterwards, layout optimization adapts UI presentation to fit the context of use situation. This approach being based on a reference-architecture called the Cedar Architecture supports any changes to the environment, the end user's profile, and the targeted platform. The relevant adaptive behavior can be applied on any UI model and then transferred to UI-presenter, which is responsible for rendering the FUI. To resolve the probable conflicts between adaptation rules, a trade-off manager is used to select those meeting adaptation constraints as much as possible. Another important point to consider in this work is providing end-users with the possibility to reverse the adaptation behavior or even choose other possible alternatives.

Within this background, we notice that a common trend in the design of UIs adaptation is to integrate adaptation requirements into the development process and transfer them to the FUI. However, particular importance should be accorded to: 1) a support tool for allowing designers to develop UIs and adapt them with the proposed adaptation technique 2) the adaptation rules design in order to ensure consistency within UI models and avoid eventually conflicts between them 3) the human involvement in the development process.

Designing Rules Adaptation

Existing approaches can substantially differ about adaptation features, but they all agree that rules design and structure are the key to rise to this challenge. However, rules are often hard-coded or designed in ad-hoc manner, making updating or reusing them a very difficult task. This is e.g. the case for Barrera et al (2014), Desruelle et al (2016) and Mezhoudi (2015). Recently, studies have been more interested in modelling rules adaptation in order to ensure efficiency and ease of use. For example, López-Jaquero et al (2009) define rules are defined using a metamodel and specified in terms of the context of use events and the UI model transformations needed to be applied. The rules design is supported by T:XML, a graphical tool that makes it easy to the designers modelling adaptation rules and automatically generating code for the transformation language. Inspired by the AGG tool proposed by Taentzer in 2000, these transformations are specified in UsiXML language and based on graph grammar. Barrera et al (2014) present Tukuchiy, a rule-based framework that generates dynamic UIs while preserving usability criteria. Several techniques of HCI are mapped to adaptation concepts in order to adjust them to different users and contexts. Rules are hard-coded but supported by a conceptual integration model called Runa-Kamachiy wich provides a taxonomy of adaptation concepts describing adaptation nature and process according to the user's profile, its context of use, its tasks and the UI model. According to Miñón et al (2015) aforementioned, the adaptation rules are defined using the Advanced Adaptation Logic Description Language (AAL-DL) that can be applied to UIs described in others MDE languages, consistent with the CRF. For example, context-aware applications, which are described in MARIA and developed by Paternò et al (2009), can benefit from AAL-DL adaptation model.

Each rule is structured in terms of event, condition and action. It can be performed in both design time and runtime. In the design time, it is directly integrated in the UI by the designer while at runtime, it is triggered by an event. If the condition is satisfied, the action is then executed. Elsewhere, Zheng et al (2016) use a graphical representation to model adaptation rules, inspired from the decision table reasoning. It consists of an adaptation tree structured on two types of nodes: condition node and conclusion node. The first one checks the context conditions whereas the second one represents the action to be achieved. Each path in the adaption tree, from the root to the leaf, presents an adaption rule. Thereby, to make the relevant decision, the adaptation tree splits the nodes on all available variables (context conditions) and then selects the split, which results in most checked sub-nodes.

To conclude, available research and designers' opinion commonly point to at least three adaptation aspects to be considered (i) a conceptual framework for modelling adaptation rules (ii) an adaptation manager for making decision and executing the associated behaviour (iii) a graphical tool for supporting the specification of the adaptation rules by designers even if they lack knowledge about accessibility requirements. Moreover, it helps them to control the adaptation process.

In the following section, we present our solution for adapting UIs to the context of use, which takes into account these three aspects of adaptation.

Solution Overview

The approach proposed in this paper takes advantage of the MDE engineering by applying different types of adaptation on the various abstraction levels of the CRF before deriving the FUI that will be delivered to the end-user. By applying this approach, we are covering almost all the adaptation requirements previously discussed and we are supporting the integration of adaptation rules in the development process of accessible UIs at both design time and runtime. The proposed theoretical architecture illustrated by figure 1 includes four modules relaying to the adaptation system:

(1) *UIs Models*: these modules are specifying the UI at each level of abstraction of the CRF and lead a transformational method allowing progressive generation of the FUI from the task model.
(2) *The Adaptation Manager*: adds and upgrades rules defined by the designer, evaluates the current context information and fires rules satisfying the context parameters. The specification of this process relies on an adaptation metamodel.
(3) *Usability*: typically, the adaptation system is called to respect the usability guidelines especially the user satisfaction and the level of controllability.
(4) *Machine Learning*: this module is intended to support the system in learning new rules based on the user preferences.

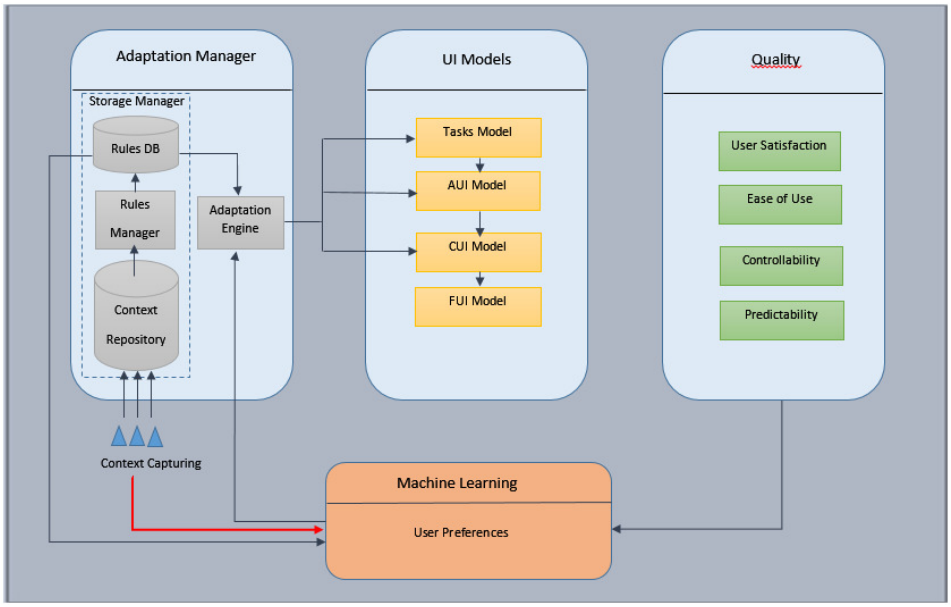

Fig. 1: Global theoretical architecture for adaptation.

A Metamodel For UI Adaptation Rules

The adaptation model specifies the appropriate reaction that can occur and that has an impact on the UI. According to the metamodel depicted by figure 2, an adaptation rule is structured upon two parts. The left part (the green block) refers to events that can trigger the adaptation. An event may be internal to the system or a change in the contextual situation either set by the designer ("ContextVariable") or detected by a sensor ("SensorVariable"). *AdaptationRule* (yellow block) contains Meta information about the rule: its identifier, the policy expresses the level of controllability of the adaptation process, the strategy and the priority value. The right part (the red block) specifies the action to be performed and the UI models involved in the adaptation.

We plan various actions in accordance with the selected strategy and the involved UI model. Some of these actions aim to update the task model but most of them act on the CUI model. At this level, we mainly focus on the modalities of the UI in order to provide to the system with the relevant modality according to the end-user‘s disability. In fact, we think that multimodal processing is preferred by end-users as it offers more natural interaction with the computer. Multimodal interactions are based on CARE properties (Complementarily, Assignation, Redundancy and Equivalence) that deal with modalities combination from the user perspective.

The main advantage of this metamodel is that it can be extended by other classes describing the adaptation features or behaviors. It is also valid at runtime as well as at design time.

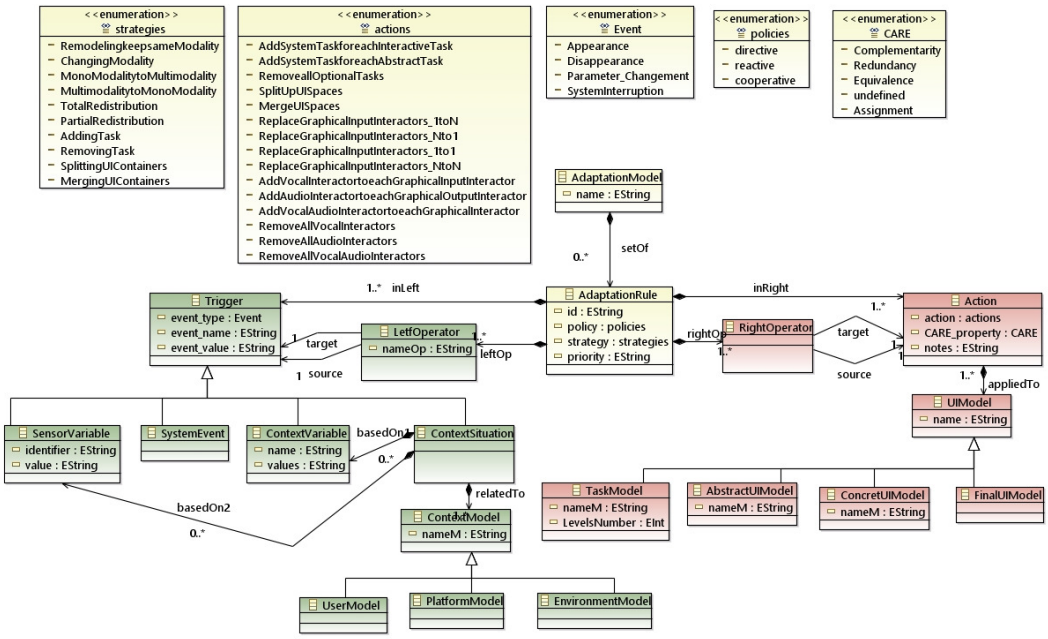

Fig. 2: The adaptation metamodel

The Adaptation Manager and Processing

The adaptation manager handles the whole adaptation process. It is particularly composed of two main modules: the Storage Manager and the Adaptation Engine.

The Storage Manager: stores and manages the adaptation rules in a relational Database (MS Access Database) to make it easier for the designers to add new rules or update the existing ones. The same rules defined at the design time can be used when needed at runtime. These rules are then sent to the Adaptation Engine that will achieve succeeding steps of the process.

The Adaptation Engine: due to the large amounts of data that may be treated, decision making becomes more and more difficult. Possible conflicts can also occur between some rules that refer to the same context situation but imply different actions. Thus, we find more interesting to use an inference engine that ensures the performance and even the consistency of the adaptation behavior. A comparison study between some inference engines prompted us to choose the Drools Rules Engine (DRE).

Drools is a BRMS that uses the rule-based approach to implement Expert Systems. It can deal with a large number of rules and facts. Using an adapted implementation of the Rete Algorithm, it matches facts and rules in order to infer conclusions, which results in actions. To be read by the DRE, rules templates must be written using a domain-specific language (DSL). Therefore, in order to hide the complexity of the language and make the definition of rules' templates and their maintenance easier even for non-domain experts, we implemented a module that allows their automatic generation. It consists of two types of MDE transformations: model-to-model transformations that create rules models conforming to our adaptation metamodel, from data retrieved from the

Database; and model-to-code transformations that generate the Drools' rules from the rules models. Figure 3 shows an example of a generated Drools rule's template that defines a rule called "UserPref". According to the user's profile, chiefly his age and his background-color preferences, this rule updates the CUI model by modifying the font size of interactors' text as well as the background color of the container (the container can be a frame, a window or a web page, according to the target language of the FUI).

```
import NewConcretModel.*;
import NewConcretModel.impl.*;
import usermodel.*;
import usermodel.impl.*;                    The facts
rule "UserPref"
salience 80
    when
            $user : usermodel.User($preference : getPreferences(), $age : getAge())
            $pref : ColorPreference($bg : backgroundColor, $font: fontColor) from $preference
            $cui : NewConcretModel.ConcretUIModel($elem : getElmt2())
            $frm : Frame() from $elem                                  The condition
    then
            $frm.setBackgroundColor($bg);
            $frm.setFontColor($font);
            if($age.equals("Between 46 and 60 years"))
            $frm.setFontSize(SizeofFont.get(1));
            else if($age.equals("Over to 60 years"))
            $frm.setFontSize(SizeofFont.get(3));                       The Action
    update($cui);
end
```

Fig. 3: **An Example of a Drools' rule template**

As illustrated by figure 3, each rule has a salience value. It is a form of priority where rules with higher salience values are given higher priority when ordered in the Activation queue.

Based on a foreword chaining system, we feed the knowledge base with two types of data: the context's models data that matches with the left hand part of a rule and the UI's models that will receive inferred conclusions.

When the adaptation process is launched, the Drools engine loads his working memory by facts and then proceeds by evaluating the rules. Only those that their condition part is evaluated to true are fired. If more than one rule matches a fact, they will be sorted by salience values. If multiple rules have the same salience, the DRE guarantee executes them at the same level.

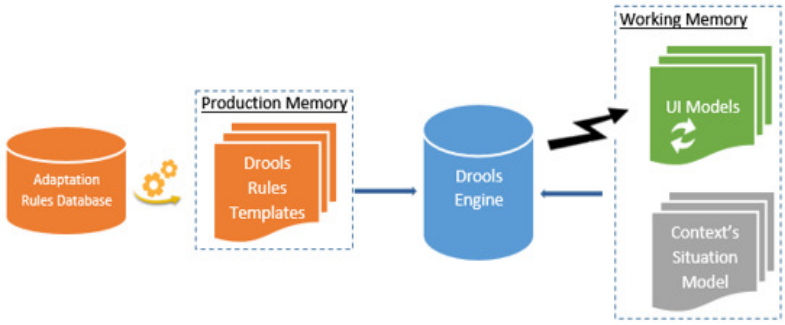

Fig. 4: the adaptation Manager

DNA: A Toolkit to Support the Rule-Based Adaptation Process

To validate our proposal, we have developed the DNA toolkit that supports the Design, the geNeration and the Adaptation of UIs pursuant to CRF specification. It aims to reduce the effort needed by developers to build UIs for interactive systems by enabling the creation of the UI models through a set of graphical editors as well as ensuring the automatic generation of these models from each level of abstraction to other levels. The resulted FUIs can then be adjusted to specific users' profiles or context situations. The user's profile essentially identifies the user's age, his disabilities if they exist, his domain expertise's level, and his preferences in modality of interaction, font colors, background colors, etc.

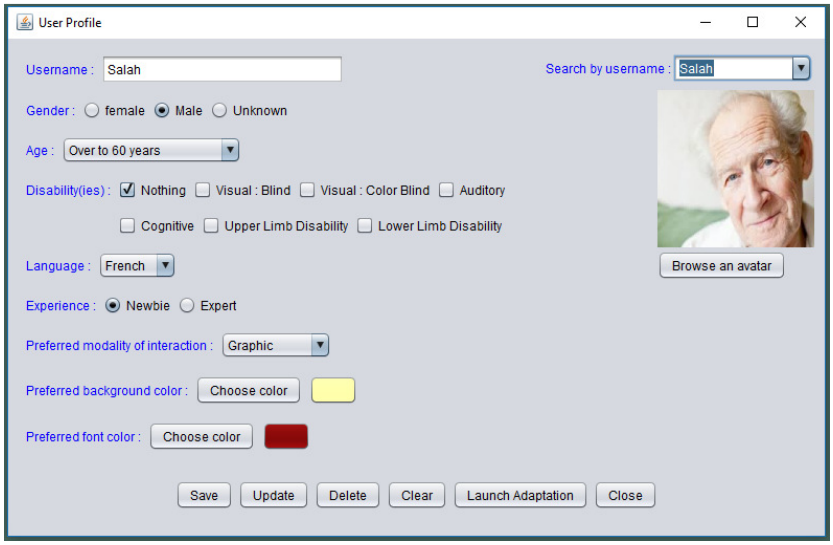

Fig. 5: The user profile interface (Mr. Salah's profile).

The context situation describes the current conditions of use of the system. It includes parameters referring to:

(5) the end-user of the system: such as the disability type if it exists, the domain experience's level, the mobility state and the current activity
(6) the platform used for the interaction: such as the screen size, the battery level and the multimedia equipment
(7) the physical environment where the interaction will take place: such as the noise level, the light, the time and the current location.

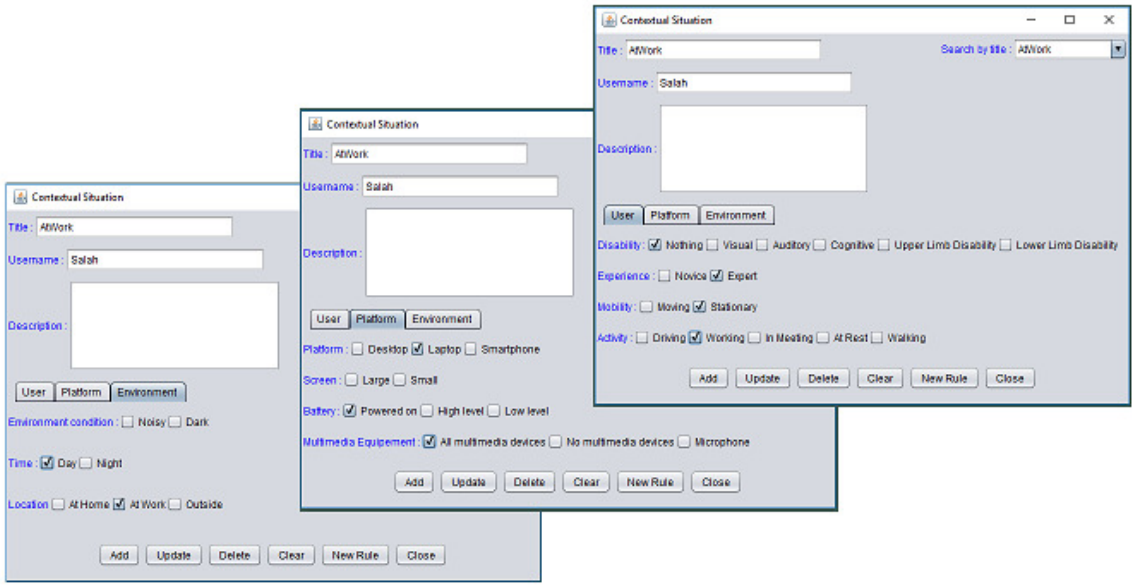

Fig. 6: The context situation interface ("AtWork" situation).

Adaptation rules can be edited according to the defined context situations. Each rule has a priority value and is specified by the adopted strategy, the target UI model, the action to be performed and the CARE property for only actions that will be applied on the CUI model. For example, the adaptation rule "AtWorkRule" shown in figure 7 has 60 as priority level and is triggered when the context situation "AtWork" is selected (see figure 6). As it is based on the "Removing Task" strategy, this rule acts on the Task Model by removing all the optional tasks.

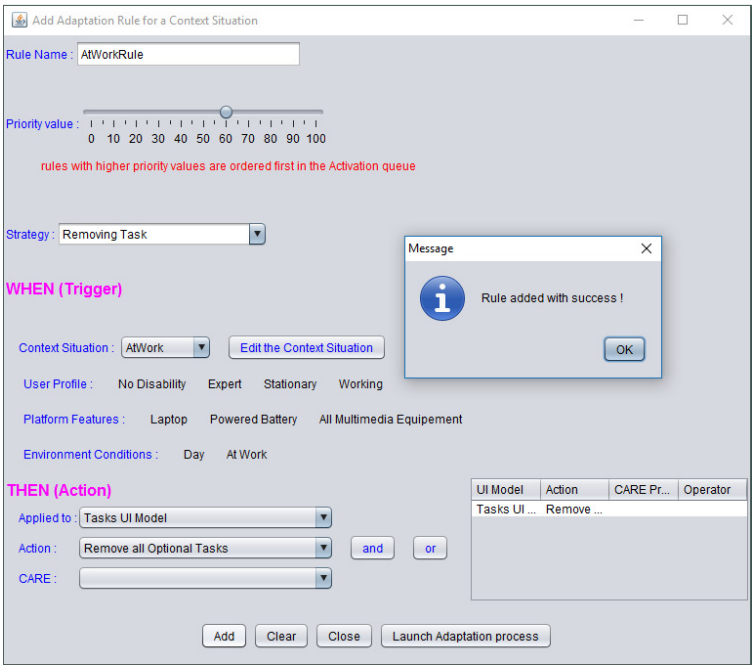

Fig. 7: An interface for adding new rule for a specific context situation.

Machine Learning In the Support of UI Adaptation

In general, rule-based systems can be used to perform UI adaptation at design-time and make automated decision about their behavior according to different context situations. They are attractive because they are simple to manage and to understand. However, they face some issues. For example, they are unable to adapt to new data and need to be under expert control. Furthermore, rules cannot be learned or derived from the training data and remained fixed overtime. As result, we can say that these systems will correctly work only if we know all situations under which decisions should be made beforehand, that is at design time. Therefore, we need to adopt a solution that will help them to recognize new context situations and build suitable rules on the fly. Machine Learning (ML) techniques fit perfectly with this issue and seem to be a significant potential to convey the UI adaptation at runtime. The use of ML to deal with context-awareness goes back to the late 1990s with Vanderdonckt (1995) who applied a knowledge-based system to conduct the automatic selection of the most appropriate widget to build a GUI. Later, ML was increasingly used to predict the user behavior such as in Mitrovic research works (2007) and Liu et. al (2003), according to interaction traces like in Karami et al (2014), user profiles (Wassermann et al, 2011) or user feedback (Mezhoudi, 2013). Based on this briefly review of literature, we decided to find a satisfactory technique for combining rule-based systems with ML. Our approach aims to perform a user-centred adaptation by involving preferences into the learning phase.

Therefore, rules are memorized to be used later as a guideline about changes that can occur in the context. Stored rules can give a pretty-high level of confidence to expect a specific system reaction whenever context parameters change. This way of reasoning is similar to a k-Nearest Neighbors algorithm (KNN algorithm). Whenever a new situation occurs, it scans through all past experiences (the stored rules or samples in our case) and looks up the k closest ones. Accordingly, if we want to predict the suitable adaptation action for a new context situation, we should take the majority vote of all k neighbors. We are

here faced to a multi-label classification problem where we expect to predict more than one class labels at the same time. At this stage, the user preferences are introduced into the prediction process. The action of the system will be selected among actions relatives to the retained rules by the KNN algorithm according to their preferences scores. Each action (class) is associated with a score for every user individually, which represents the users' individual preferences for adaptation behavior. For example, if k=5 and in 3 of most similar rules the action *x* is chosen and in 2 rules the action *y* is chosen, then in this case, the user's preferences score will be a decisive criterion for choosing the suitable action of the system. As a simple example, following (figure 8) is a spread of colored points (green, orange, blue, yellow) referring to four possible adaptation actions that can be performed by the system. For a new expected action (black point), the most suitable class label (action) for the current context is determined by looking up the classes of the 5-nearest neighbors (k = 5). Here, two orange points and three blue points are the nearest. Since the orange points refer to a preferred action for the user e.g. have the highest score, then the decision would be orange (the red-framed point).

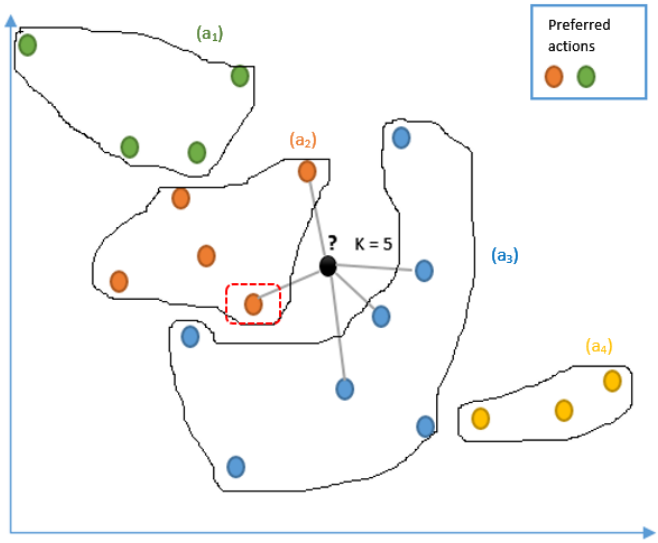

Fig. 8: The KNN classifier

More formally, let us define rules according to "condition → action" paradigm. Thus, a rule is formatted as: if <attributes> then predict <class>, where <attributes> is a conjunction of selectors referring to the context's parameters and <class> refers to the adaptation action. Selectors are represented by (at_i,v_i) where $at_i \in AT$ (AT is the set of attributes) and $v_i \in D$ is its value. For example, the attributes "Disability", "Experience" and "Activity" are related to the user profile. Each attribute has a set of (nominal or numeric) values (e.g. the "Experience" value ∈ {expert,novice}). Globally, the context parameters are presented by $S = \{(at_1, at_2, \ldots, at_n) ; at_i \in AT\}$. Adopting this terminology, the adapted KNN algorithm is written as follow:

Nadia Ghaibi, Olfa Dâassi and Leila Jemni Ben Ayed (2018), Communications of the IBIMA,
DOI: 10.5171/2018.281881

Algorithm : Adapted KNN Classifier

Input $\mathcal{D}$: Dataset
$\mathcal{A}$: class labels of D
d_i: a sample
x: unknown sample
Output a: predicted class label

Begin

1 $\mathcal{P}reprocess$ ($\mathcal{D}$) ; // $\mathcal{P}reprocess$ prepares the dataset for the classification (normalize the data to the range [0,1]

2 **for** i = 1 to m do // m is the samples number
$dis \leftarrow \mathcal{D}istance$ (d_i , x) ; // dis is a set containing the distance between di and x
end for

3 $I \leftarrow \mathcal{S}mallest$ (dis , k) ; // I is a set containing indices of samples having k smallest distances in Dis

4 $j \leftarrow \mathcal{H}ighestScore$ (I) ; // $\mathcal{H}ighestScore$ determines indice of the sample in I having the highest score

5 **return** $\mathcal{L}abel$ (j) ;

End

Fig. 9: Pseudocode for the adaptation ML algorithm based on the KNN classifier (AML-KNN).

The objective of this algorithm is to achieve user satisfaction by getting subsequent adaptation results, which will be closer to its preferences. More precisely, user satisfaction can be reached when the system's adaptation behavior mismatches the user's expectations. User preferences are reified as different weights of the classes so that a class with higher weight will more probably be satisfied. Different methods of rating such as Ordinal rating (Star-rating), Interval rating (numeric value) or Nominal rating (like relevant, "irrelevant" can compute user preferences. In our work, we adopt the second method. Our approach allows users to implicitly interact with the learning models in order to ensure that practice adaptation results do not deviate from their preferences. Therefore, we have modeled the adaptation learning problem as a multi-label classification problem where several adaptation actions can be considered by the system but only the one that simultaneously is major voted and have the highest score referring to user preferences will be performed. To validate our approach, we have implemented this algorithm and compared its performances with those of the SVM and the AdaBoost algorithms as they recommended for multi-label classification problems. First experimental results show that our approach can achieve increased prediction performance with respect to precision, recall and F-measure. Figure 10 illustrates the comparison results for a 10-cross validation.

Evaluation criterion	AML-KNN	SVM	AdaBoost.M1
precision	0,692	0,562	0,420
recall	0,687	0,578	0,490
F-measure	0,688	0,548	0,416
Accuracy	0.686	0.578	0.489

Fig. 10: Experimental results for AML-KNN, SVM and AdaBoost ML algorithms.

A Case Study: Accident Report

To evaluate the efficacy of DNA to support the adaptation of generated UIs at design time, we will set up here an adaptation scenario and we will explain its results through a case study. The scenario is the following. Mr. Salah uses his car every day to go to the office. This morning, he hadn't paid attention while driving and unfortunately, he got into a car accident! His car was damaged and he needs to report that to his insurance company. Fig. 11 shows the interface he used.

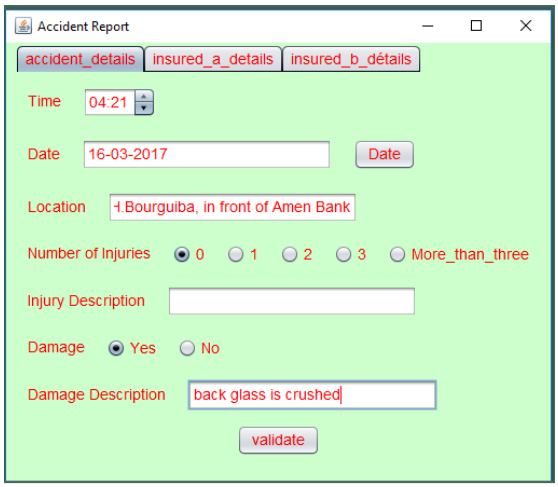

Fig. 11: The accident report interface as generated by DNA.

Other customized versions were also delivered to some users to fit more their profiles. The illustration below gives you an idea about the application interface once adapted to Mr. Salah Profile (see figure 5 for Mr. Salah Profile).

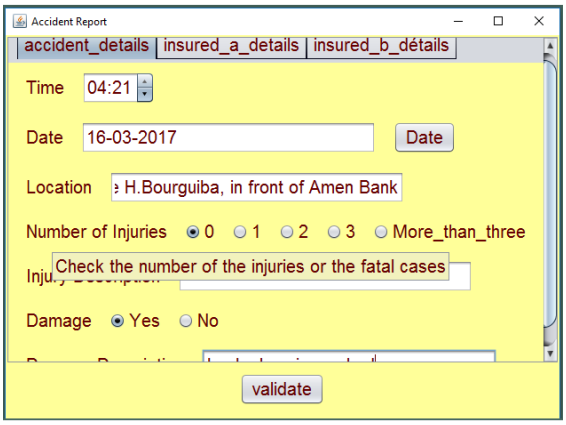

Fig. 12: Adapted UI to a user profile

Three actions of adaptation are illustrated by the figure above: 1) according to the Mr.'s age, labels take large sizes. Therefore, scrollpanes are added to provide a scrollable view of the interface 2) background color and font color are customized according to his preferences 3) due to his low level of expertise, tooltips are added giving him some help to fill in the form fields.

Otherwise, to prove the validation of our approach, we have also tested the execution of some adaptation rules. We give below the rules descriptions and the respective resulted UIs (Fig. 13 a), b)).

Rule 1

Id: "AtWork"
Condition: the user location is "At Work"
Action: remove all optional tasks
Description: the adaptation engine will browse the Tasks Model in order to remove all the optional tasks.

Rule 2

Id: "ScreenSizeRule"
Condition: small screen (smartphone)
Action: replace N interactors by 1
Description: the RadioButtons respective to the "Number of Injuries" and the "Damage" are replaced by two ComboBox having the same options.

Rule 3

Id: "DrivingRule"
Condition: the user activity is "Driving"
Action: add vocal/audio interactors.
Description: according to this rule, the adaptation engine will assign to each input interactor a vocal input recorder and to each output interactor an audio output component.

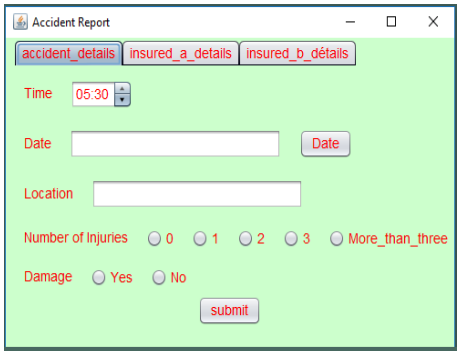

Fig 13. a) The resulted Interface after the execution of Rule 1.

Fig 13. b) The resulted Interface after the execution of Rule 2.

According to the results reported by the case study above, we can say that our approach has achieved its objectives. The developed tool allows users to generate customized UIs and then adapt them according to the end-user profile or the context situation. Although several functionalities are still missed, they appear as a useful support for creating adaptation rules and managing the adaptation process. All the context models are covered and the adaptation can be applied at all the levels of abstraction of the CRF. Besides, multiple strategies are proposed: remodelling (monomodal or multimodal), adding or removing tasks… Rules conflicts are also partially resolved by ordering rules using the priority value assigned to each rule. Other aspects that are worth mentioning are the simplicity and the completeness of the tool. Regarding simplicity, it is easy enough to be used even by people who are not HCI experts. Concerning completeness, it allows designers, through simple actions, to define context situations and update them, manage adaptation rules with the specification of the conditions and the actions parts, launch the adaptation process and regenerate the interface. Thus, they don't need any extra tools to achieve the processing. However, our work needs to be improved to correct some shortcomings such as the limited number of actions that can be applied on the UI models and the limited role of the end-user during the process.

Conclusion

This work proposes a Model-Driven method to generate adapted and customized UIs according to user preferences and/or capabilities. Two main contributions of this work can be concluded. The first one is to provide the end-user with full control of the adaptation process by editing and selecting the relevant rules for each contextual situation or a user profile. The second contribution is to use a BRMS as an adaptation engine which provides an automated conflict resolution via priorities and inference mechanism. Thus, this work highlights several interesting aspects:

(8) It is reusable and extensible: due to the adoption of a model-driven approach, existing transformation rules can be reused for a wide variety of contexts. Designers can also extend metamodels by new concepts without any need to modify the rest of code.
(9) It is complete: our solution provides designers with all the

required tools to automatically generate high-performed interfaces: various model transformations allowing the generation of UIs from any level of abstraction of the CRF, a module for context designing, a module for adaptation rules editing and several ways to launch the adaptation process (in addition to graphical editors for designing UI models, which we haven't discussed in this paper).

(10) It is useful: the case study we have carried out to test our tool has proved the efficiency of our approach for generating context-dependant UIs.

We will dedicate the subsequent step of our work to invite users to test and evaluate our environment. Based on their recommendations, we will proceed to improve the tool and add further features. We will mainly focus on the usability of the tool, the runtime aspect of the adaptation and the user controllability of the whole process.

References

1. Akiki, P. A., Bandara, A. K., and Yu, Y. 2016. Engineering Adaptive Model-Driven User Interfaces. *IEEE Transactions on Software Engineering*, *42*(12), 1118-1147.

2. Barrera, L. F., Ramos, A. C., Florez-Valencia, L., Pavlich-Mariscal, J. A., and Mejia-Molina, N. A. 2014. Integrating Adaptation and HCI Concepts to Support Usability in User Interfaces-A Rule-based Approach. In WEBIST (2) (pp. 82-90).

3. Bouchelligua, W., Mahfoudhi, A., Benammar, L., Rebai, S., and Abed, M. 2010. 'An MDE approach for user interface adaptation to the context of use'. Proceedings of the International Conference on Human-Centred Software Engineering (pp. 62-78). Springer Berlin Heidelberg.

4. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., and Vanderdonckt, J. 2003. 'A unifying reference framework for multi-target user interfaces'. *Interacting with computers*, 15(3), 289-308.

5. Chen, B., Peng, X., Yu, Y., Nuseibeh, B., and Zhao, W. 2014. 'Self-adaptation through incremental generative model transformations at runtime'. Proceedings of the 36th International Conference on Software Engineering (pp. 676-687). ACM.

6. Criado, J., Rodríguez-Gracia, D., Iribarne, L., and Padilla, N. 2015. 'Toward the adaptation of component-based architectures by model transformation: behind smart user interfaces'. *Software: Practice and Experience*, 45(12), 1677-1718.

7. Desruelle, H., Isenberg, S., Botsikas, A., Vergori, P., and Gielen, F. 2016. 'Accessible user interface support for multi-device ubiquitous applications: architectural modifiability considerations'. *Universal Access in the Information Society*, 15(1), 5-19.

8. Karami, A. B., Sehaba, K., and Encelle, B. 'Apprentissage de connaissances d'adaptation à partir des feedbacks des utilisateurs'. In In Proceedings of the IC-25èmes Journées francophones d'Ingénierie des Connaissances (pp. 125-136).

9. Liu, J., C.K. Wong, andK.K. Hui. 'An Adaptive User Interface Based on Personalized Learning'. *IEEE Intellignt Systems* (2003),52-57.

10. López-Jaquero, V., Montero, F., and Real, F. 2009. 'Designing user interface adaptation rules with T: XML'. Proceedings of the 14th international conference on

Intelligent user interfaces (pp. 383-388). ACM.

11. Mezhoudi, N. 2013, March. 'User interface adaptation based on user feedback and machine learning'. In Proceedings of the companion publication of the 2013 international conference on Intelligent user interfaces companion (pp. 25-28). ACM.

12. Mezhoudi, N., PerezMedina, J. L., and Vanderdonckt, J. 2015. 'Towards a Conceptual Model for UIs Context-Aware Adaptation'. Proceedings of the 2nd World Congress on Multimedia and Computer Science.

13. Miñón, R., Paternò, F., and Arrue, M. 2013. 'An environment for designing and sharing adaptation rules for accessible applications'. Proceedings of the 5th ACM SIGCHI symposium on Engineering interactive computing systems (pp. 43-48). ACM.

14. Miñón, R., Paternò, F., Arrue, M., and Abascal, J. 2015. 'Integrating adaptation rules for people with special needs in model-based UI development process'. *Universal Access in the Information Society*, 15(1), 153-168.

15. Mitrovic , N. Royo, J.A. and. Mena, E. '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, 2007.

16. Paternò, F., Santoro, C., and Spano, L.D. 2009. 'MARIA: A Universal Language for Service-Oriented Applications in Ubiquitous Environment'. *ACM Transactions on Computer-Human Interaction*, Vol.16, N.4, 19:1-19:30.

17. Peissner, M., Häbe, D., Janssen, D., and Sellner, T. 2012. 'MyUI: generating accessible user interfaces from multimodal design patterns'. Proceedings of the 4th ACM SIGCHI symposium.

18. Taentzer, G. 2000. 'AGG: A Tool Environment for Algebraic Graph Transformation'. Proceedings of the International Workshop on Applications of Graph Transformations with industrial Relevance. LNCS, vol. 1779. Springer, London, 481-488.

19. Vanderdonckt, J. 1995. 'Knowledge-based systems for automated user interface generation: the TRIDENT experience'. In Proceedings of the CHI'95 (Vol. 95).

20. Wassermann, B., & Zimmermann, G. 2011. 'User profile matching: A statistical approach'. In Proceedings of the CENTRIC 2011, The fourth international conference on advances in human-oriented and personalized mechanisms, technologies, and services (pp. 60-63).

21. Zheng, M., Xu, Q., and Fan, H. 2016. 'Modeling The Adaption Rule in Context-aware Systems'. *arXiv preprint arXiv:1609.01614*.

22. Zouhaier, L., Hlaoui, Y. B., and Ayed, L. J. B. 2015. 'A model driven approach for improving the generation of accessible user interfaces'. In Software Technologies (ICSOFT), 2015 Proceedings of the 10th International Joint Conference on (Vol. 2, pp. 1-6). IEEE.