Context-Awareness Meta-model for User Interface Runtime Adaptation

Nesrine Mezhoudi, Jorge Luis Perez Medina and Iyad Khaddam
{nesrine.mezhoudi, jorge.perezmedina, iyad.khaddam}@uclouvain.be
Université Catholique of Louvain, Belgium

Summary
Effective adaptation of User Interfaces (UI) is still a main requirement to improve system usability and enhance the user experience. The heterogeneity in contexts of use augmented the complexity of such a task. Several requirements should be accommodated in order to meets users' expectations. Design time adaptations are no more sufficient to guarantee context-awareness. A user satisfaction's shortcoming is still revealed backed by the lack of support of user's preferences and interventions at run-time, which decreases user-centeredness and usability. In order to improve the UI contextualization and user-centeredness, deeper research on how to adapt efficiently and effectively the UI at runtime should be directed.
This paper proposes a Context-Awareness Model (CAM) that consists of modeling UI, context and adaptation by addressing user's involvement at runtime. The CAM model is aimed to support user interface designers to develop and conceptualize system that accommodate context-awareness, and user-centeredness requirements.
Key words:
Framework, runtime adaptation, context-awareness, user involvment.

1. Introduction

Current advancements in the technological landscape and their rapid growth are creating competitive challenges, as well as new opportunities for HCI communities. Such a progress seems promising for the user interface (UI) to offer tailored interfaces and interaction scenario that correspond to end-user's specific expectations and preferences. Accordingly, adaptation approaches are evolving with technologies with the purpose to increase user's satisfaction and enhanced interaction experience.

Adaptation of UI is an active domain of research in HCI. Adaptation methods are evolving to fulfill new requirements to increase the UI efficiency. By attempting to cut with earlier interfaces that often needed recompilation for upgrades, which incurred increased cost, delay, and risk, UIs shift to a runtime paradigm. UIs turn out to be adaptive rather than being user-centered and carry out adaptation in accordance with the end-user preferences as well as the context of use.

Model based user interface benefits were widely discussed in the literature [2, 3, 14, 19]. The advantageous cost's reduction and facility of interchange are challenges the HCI community. The aim is to develop UIs with higher usability and enhanced interaction. However, such solutions lack the runtime context-awareness.

Most of existing approaches follow the Cameleon Reference Framework (CRF)[3]. (CRF) considers that once the abstract specification is defined, several instantiations could be derived. In the same way, contexts of use were defined through predefined meta-models (abstraction). However such a solution should be enhanced to support runtime context-awareness and user intervention in order to improve their usability levels and meet present-day requirements.

The purpose of this research is to support runtime adaptation while considering user intervention by means of model-based UI reification at runtime. We achieve this propose by proposing a state transition process, that is conceptualized in the model, that enhances context-awareness runtime adaptation.

This paper is structured as follows: section one presents a review of existing works on adaptation and system context-awareness.
Section two describes CAM: a conceptual model supporting UI runtime context awareness and end-users involvement.
Section three shows two implementations for a car rental case study. The first implementation is a Flippable UI for internationalization developed in accordance with UsiXML project specifications [23]. The second implementation demonstrates an adaptive UI. Finally, we conclude and explain future work in section four.

2. Related works and key challenges

2.1 Adaptation frameworks

Different theoretical frameworks and models to support systematic context-awareness are developed by researchers. These works aim at supporting design decisions and requirements for different contexts of use. In the following,

DOI : 02.IJSE.2015.1.2

we conduct a comparative analysis based on the following criteria:

- UI models employed: levels of abstractions and supported models at each level in the framework.
- Support for transformation engines: capability to generate final user interfaces from UI models.
- Support for the context of use: it denotes what dimensions of the context of use are supported. The context of use is a triplet: platform, user and environment [23].
- Adaptation model: does the framework propose an adaptation model?
- User involvement: Does the framework/model address the user involvement in the adaptation process/model?
- Adaptation autonomy: refers to the level in which adaptation is implemented, i.e., designed applications do not perform adaptation at all, adaptable applications rely on users to trigger and perform the adaptation, adaptive systems rely on the adaptation to be automatically performed, and self-modifying means evolutionary systems able to adapt their own adaptation engines.
- Adaptation techniques: does the framework/model explicit the supported adaptation techniques? (Adaptable/ adaptive/ mixed initiative/ run time/design time).

CRF [3] was introduced as a unified approach to structure model-based UI. It provides a unified understanding of context-sensitive UIs rather than a prescription of various ways or methods of tackling different steps of development. CRF outlines four abstraction levels beside the different transformations between models. However there is no consideration for user involvement and feedbacks. As well, adaptation concerns were not fully supported. Although the model driven engineering of Cameleon allow different types of adaptive behavior to be implemented such as: Using the task model to adapt the feature-set and using the concrete UI model to adapt the layout [1].

Knutov [14] suggests a general-purpose adaptive hypermedia AH framework (GAF) providing reference architecture and defining system criteria to distinguish between adaptation elements. It provides a modular structure to enhance adaptation of web-based systems capabilities.

GAF provides a basic understanding of adaptation questions. Supported adaptation concepts are determined through a composition of the system layers. Mainly three abstraction levels are supported: task, abstract and the presentation models, besides the context models.

Moreover, GAF provides an Adaptation Model that refers to the Search Engine and Ranking mechanisms. The adaptation autonomy is limited to the UI adjustment in two ways with regards to user observations: user model update and system adaptation in which the adaptation is performed (adaptation of presentation, content or navigation) utilizing the state of the user model.

UsiXml [23,2] Supports a MDE approach and covers all CRF models. UsiXml adaptations are focused on the platform model. Users are supported through stereotypes, however there is no involvement of users during adaptation. Regarding adaptation concerns, there are no information about deployed adaptation models and autonomy.

Motti [19] proposes TriPlet, a computational framework that covers a broad view to support the implementation of multi-dimensional CAA. Three conceptual methods (CADS [19], CARF [19] and CAMM [19]) have been integrated within a general computational framework that considers, in a structured way, both context information and adaptation concepts [20].

CAMM cover different UI models, context of use and adaptation models. However transformation engines were not supported as well as user involvement. Sottet proposal [21] is close to CAMM and considers same models. However, many concepts were supported through generic classes. The framework supports transformations and generation of UIs.

Ganneau [10] and Karen [12] proposals were oriented to adaptation concerns. The proposal of [10] consists of a meta-model for adaptation rules supporting adaptation models and different adaptation techniques. Adaptation autonomy is not directly supported, however the meta-model outlines two event types triggering adaptation: context change and system event. On the other hand, there is no support of user involvement.

Karen framework [12] is aimed to categorize the two key elements of an adaptive system: the aspects of automated systems open to adaptation (Taxonomy of Adaptations) and the methods to trigger those adaptations (Taxonomy of Triggers). Adaptation autonomy is not directly supported. The framework taxonomies provides a systematic way to organize research on specific adaptations or triggers.

Despite the broad scope of frameworks, their extensible facets can lead to an incoherent instantiation in addition to confused synchronization of different supported aspects.

Table 1:This table represents the analysis of related works based on Adaptation concepts and concerns. The dimensions were ranked regarding their support "●" fully explicit support, "◖" implicit support, and empty " " when it is not supported or there is no information about.

Related works	Adaptation concepts								Adaptation concerns		
	UI models				*Context model*			*User Involvement (feedbacks)*	*Adaptation model*	*Adaptation autonomy*	*Adaptation technique*
	Task	AUI	CUI	Transformation	User	Platform	Environment				
Cameleon[1]	●	●	●	●	●	●	●				◖
Usixml[24]	●	●	●	●	●	●	●		●		●
GAF (Knutov) [2,7]	●	◖	●		●	●	●	◖	●	◖	◖
Triplet[4]	●	●	●		●	●	●		●		●
Karen [7]									●	◖	●
Sottet[15]	●	●	●	●	●	●	●		●		
Ganneau[14]					◖	◖	◖		●	◖	●
CAM	●	●	●	●	●	●	●	●	●	●	●

2.2 Adaptation requirements

Several analyses and studies targeted adaptive systems from a different point of view, most of them focused on the dimensions of adaptation in systems and are specific to distinctive domains such as: medical [12, 15, 17] (medical, hypermedia).

One of the most underlined issues with adaptation is the lack of user-centeredness. A user-centered adaptation is a key to reach user's satisfaction, improve usability and upgrade the UI quality. Considering the user dimension during adaptation might be addressed through user profiles, however ontological user representation is not enough to define user preferences and needs. Context-awareness requires an enhanced consideration of user preferences, needs and expectations at runtime. Such shortcoming should be overcome through enhancing the end user involvement within the adaptation process.

Moreover, most commonly cited issues with adaptive UI are the lack of predictability, control, and privacy [6, 16], mainly because UI adaptations consider prior interaction knowledge (explicit context, domain models) [4, 6]. Such criteria were of paramount importance to the assessment of the literature and expected to contribute the improvement of their success: Controllability, Predictability and Transparency.

The controllability (defined as interface's customizability) represents the capacity and tolerance of system to support user-initiated customization of the interface. Accordingly users have the opportunities to prevent or actively accept adaptations, and to undo or override adaptations (cf. [24]). Many works (e.g. [5]) argues for providing users full control over automatic adaptations as a major requirement of acceptable adaptive systems.

The predictability (interface's non-perceptiveness) focuses on the extent to which past and present interface allows user to determine the outcome of future interactions, it is about actions and effects. Gajos [9] considers that an adaptive system is predictable if it follows a strategy users can easily model in their heads, and then he evaluates predictability effects on user's satisfaction.

The transparency (comprehensibility) concerns the honesty of the system. It presents the capacity of user to understand adaptation and interpret perceived information.

All above-mentioned criteria agreed on the fact that successful interaction must not result in a confusing situation and should avoid the trouble of losing control over the user interface for end-users. Users must be at the heart of adaptation. Their involvement could be achieved by providing non-technical designers and typical users with user-friendly techniques for managing interfaces depending on their aptitudes.

To our knowledge, there are no frameworks that match with user-centeredness and agile principles (such as: incremental, iterativity, user-centered) for adaptation. Most of them were focused mainly on the conventional adaptation mode or consider just some fragments. For instance, to adapt a UI to a user model without being user centered [3, 8, 10, 21]. Moreover supporting recent and usual adaptation strategies from different perspectives allowing full understanding and comparison of techniques is still partially fulfilled.

An iterative progressive adaptation enhanced by intelligent techniques can meet this shortcoming. The intent is to advance the adaptations and provide systems with the ability to learn and build novel knowledge in an incremental way with regard to changes in the context.

3. A Conceptual Model for runtime Context-awareness

So far, we focused on two paths to discuss the context of the study and to outline the need for advancing adaptation topics and presenting Context-Awareness Model (CAM).

Several models address the adaptation process. Most of them conceptualize adaptation rules in a specific way [10] such as the conceptual model proposed in [21]. Broader conceptual models are presented by [20]. They cover context, adapter, model and adaptation rules.

The reviewed literature allowed an analysis of involved concepts and their characterization. Along with above detailed researches, we propose a conceptual model for adaptation aimed to cover main involved features. The model (figure 2) is aimed at supporting an explicit, comprehensible and complete configuration of adaptation concerns and allowing advances and improvements. It is intended to cover the whole involved concepts and determines their relationship and dependency.

Three main packages were identified to distinguish involved classes belonging to different adaptation dimensions (figure 1).

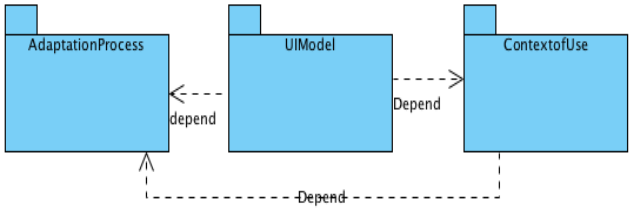

Fig. 1 Adaptation main concepts

The Adaptation package is the heart and engine of contextualization, linking all key elements involved for UI adaptation. The "UIModel" package defines the user interface independently of both adaptation and the context of use. The context of use package corresponds adaptation triggers and all contextual factors. In what follows a detailed description of involved elements is presented.

3.1 The Adaptation Package

The model of the adaptation package (figure 2) establishes the adaptation as a model separate from context and interface definitions. This dimension includes all classes related to the adaptation itself. It is intended to give an abstract conceptualization for the adaptation process in term of UI states and transitions.

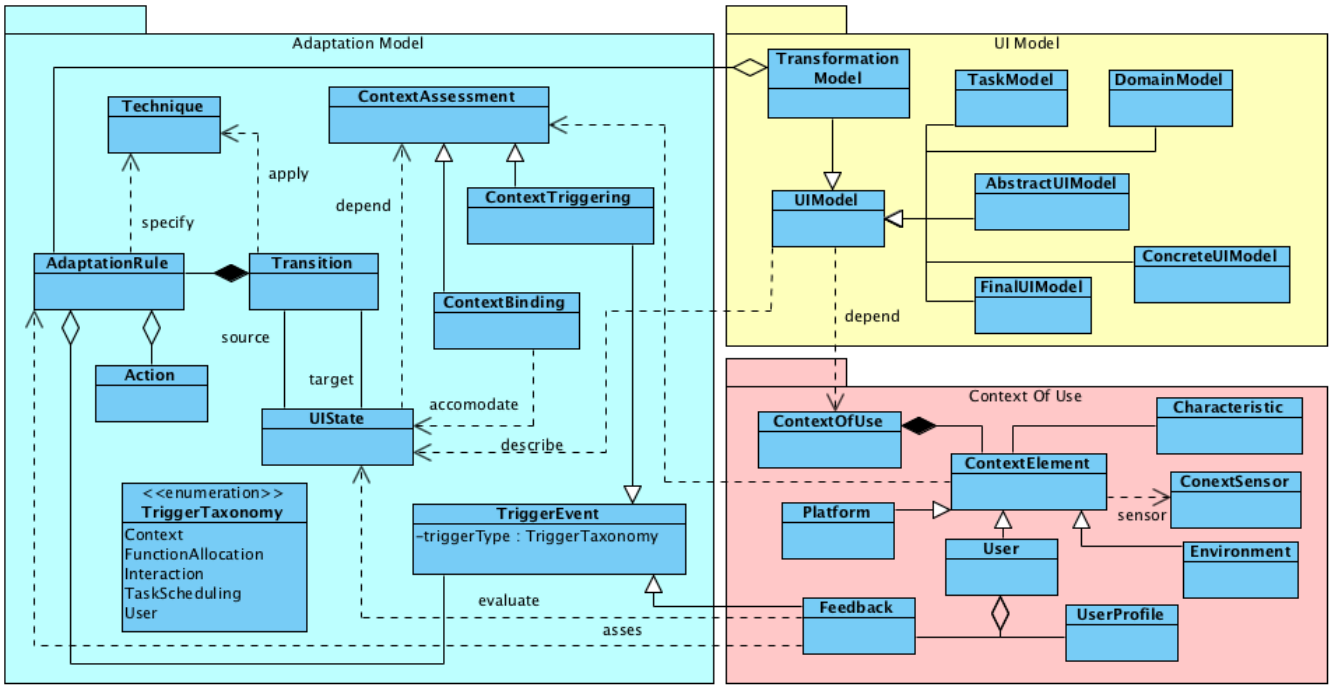

Fig. 2 An Unified Adaptation model for runtime context-awareness

A "UIState" remains a characterization of a UI model consistent with a context assessment. The state terms values of UI attributes with consideration of the context of use. For instance at a concrete abstraction level, for a phone device context the choice interaction unit for a values number up to 30 is assessed to a Drop-down list.

UI adapted features (for instance, Interactors, Task, AbstractUnit, Widgets, etc.) depend of the considered abstraction levels from defined UI Models and the values depend on the current context. The "UIState" changes during an adaptation process through a set of "Transitions" recapitalizing the interface changes. A transition presents a set of adaptation rules targeting a set of UI attributes and accommodating a context change.

The "AdaptationRule" is a part of the "Transformation Model" which consists on different mapping models such as reification, translation, reflexion [3]. It consists of one or more "TriggerEvent" initiating an adaptation and a set of action performed to change the "UIState". For example, we can imagine that an end-user could have an explicity control on the UI definition via his feedback.

The adaptation could be also triggered automatically based on an autonomous decision making process regarding a context assessments.

3.2 The Context Of Use package

The Context of Use has been modeled as a specialization of User, Platform and Environment. The figure 2 gives an overview of the main entities modeled by the Context of Use. These entities are intended to identify attributes and proprieties influencing the adaptation process and providing a trigger event for an adaptation.

The "ContextElement" class determines the set of descriptors that can be considered to define context dimensions; in some cases of adaptive UI, features values are determined via "ContextSensors". As the context is a composition of information gathered regarding different dimensions, it contributes to the definition of adaptation rules conditions. The "ContextElement" defines the context of use as well they present the trigger for all "AdaptationRules".

The "UserModel" class is expanded with the "Feedback" class and the "UserProfile" class. The "Feedback" class defines the evaluated behaviors of the user during interaction. It is aimed to enhance the user involvement during the adaptation.

The "UserProfile" class (figure 3) has been modeled as a composition of Language, Knowledge, Country and PreferredRepresentationStyle.

The "Language" class consists of the base language used by the user. The "knowledge" class defines expertise level of user. This class can be used to organize the information on the interface. For instance, and advanced user might require less guidance to accomplish the tasks. Instead a novice user will require a friendlier interface that will support and guide them to the accomplishment of tasks.

The "Preferred representation style" can be video, text and/or audio. Theses preferences help to the system to determinate the best adaptation of the information. The meta-model outline an enumeration stating potential styles.

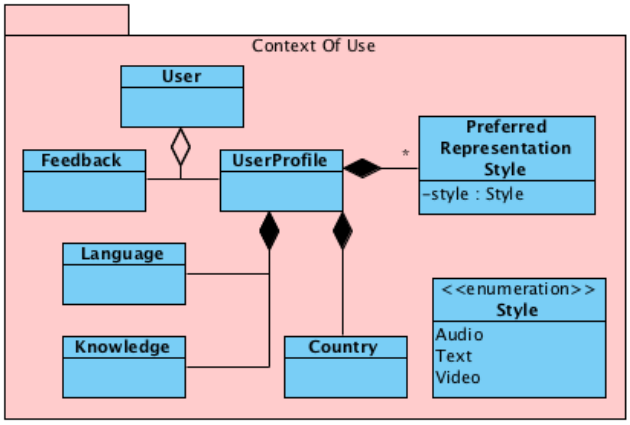

Fig. 3 The User Model

The "Feedback" class as a "ContextElement" allows handling adaptations priorities that must be assigned to prevent conflicts, for instance user feedbacks could be considered to evaluate an adaptation rule and promote or demote it. The "Feedback" is involved for different purposes, for instance control trigger and/or evaluate adaptation decisions. This class is an aggregation of the user model; it is a specialization of the "TriggerEvent" class as well as the "ContextTriggering". The "Feedbacks" class is intended to assess the "UIState" that depends on the current context of use defined by "ContextBinding" class. An adaptation is triggered by a change in this context surrounding the interaction.

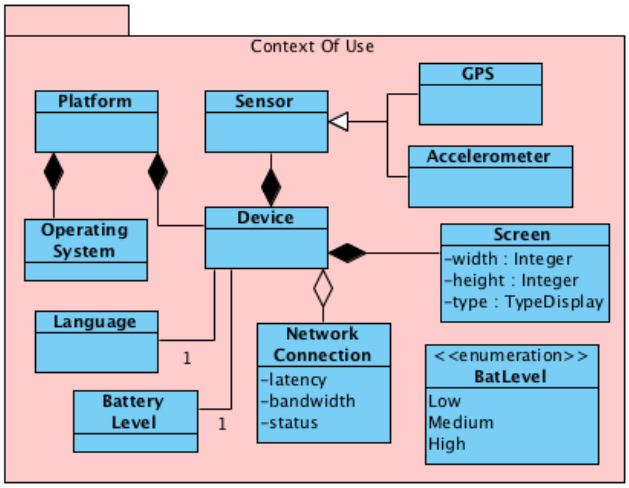

Fig. 4 The Platform Model

The "Platform" class determines the set of information that can be considered to define the hardware used by the user. Figure 4 gives an overview of the main entities modeled by the Platform. The root entity is the "Platform" class with is linked to the Operating System and Device.

The information considered includes the characteristics of the Device and the operating system used to access the

application. The "Device" considers integrate sensors, the screen size, the battery level, the language and the network providing the connection. For instance, a GPS that permits to acquire the geographical coordinates of the user. In the case of a battery low level, the adaptation can't be considerate as a multimedia element.

The environment model (figure 5) provides the characteristics of the environment in which user interact with the device. The environment can be represented as different aspects (Time, Date, Noise Level, Movement Status, Language, Weather, Direction and Location) considered by [22].

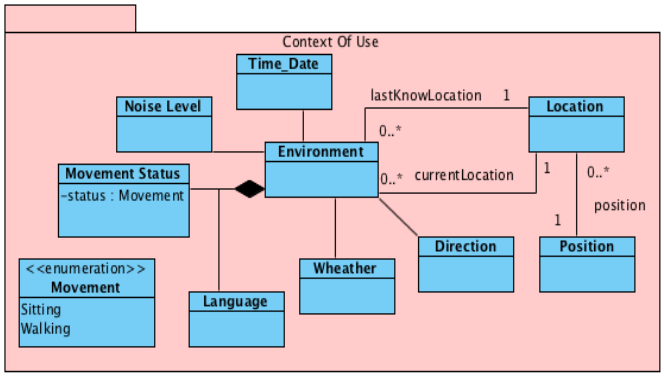

Fig. 5 The Environment Model

The climate conditions like the "Weather" can determinate how the information can be presented on the screen. The "Location" of the user and the "Noise Level" determine the more adequate type of interaction to the user.

3.3 The UI Model Package

The proposal of the "UIModel" package can be related to any UI approach, such as Model-based approach showing a combination of UI models defined in the Cameleon reference frameworks [3] and PIM model which could be considered a model-based approach considering only the Final UI Model.

The "UIModel" is decomposed mainly into four step organizing four abstraction levels:

- "TaskModel" providing a goal-oriented description of interactive systems suitable for reviewing temporal relationships between tasks, and their decomposition into elementary tasks.
- "AbstractUIModel" outlining an expression of a UI in terms of interaction units without making any reference to implementation.
- "ConcreteUIModel" presenting the UI in term of concrete interaction object that are modality-dependent, but implementation technology independent,
- "FinalUIModel" that represents the final implementation realized in a programing language.

The "UIModel" support the execution of model throughout transformation. A "TransformationModel" links involved models during generation process. Commonly transformation was merged with adaptation, however, we argue for separating them improve the transformer engine performance and reduce their complexity. [24] and [25] identifies tree main transformation: reification: form a high abstraction level to more concrete one , abstraction: from an level to more abstract one (reverse engineering) and translation wich regards transformation, within the same abstraction level.

3.4 Examples

In this section, we illustrate by few concrete examples some relevant aspects of the above detailed meta-model. The context of use is represented by a set of categorized context elements: the user, the platform and the environment. Some context elements could be assessed via sensors such as environment luminosity, platform screen size user state etc. Assessment of context elements could be considered as a context binding allowing to recognize a specific context accommodated by a particular UI state. In the typical case, the context assessment acts in the role of a triggering event. For instance the detection of a low battery level on the device, trigger an economy mode defining a particular UI state. The transition to the economy state is assured via a set of adaptation rules. Such as:

If battery level < 15% Then (1) reduce screen brightness / change the background color and (2) deactivate Wi-Fi / Bluetooth, Etc.

Further, the proposed meta-model considers an additional relevant trigger event: the user feedback recognizing the user feedback and interventions during interaction. Such feedbacks are valuable for UI and interaction context-awareness because it incorporates user preferences and needs during execution. The CAM considers user's feedback as part of the user modeling. However supporting feedbacks cannot be merged with the user profile, since feedbacks has a dynamic aspect that is likely to change frequently during the use.

The support for the user's feedback is illustrated in the following scenario:

Applying adaptation rules mentioned above regarding the battery level, the user should be asked to approve modifications before applying the rules. The rules represent a static aspect of the adaptation, while asking the user for his feedback represents the run-time aspect. Feedbacks assess adaptation rules; in this case user's feedbacks are considered as promoting/demoting factors for adaptation rules.

Moreover feedbacks could be used as an evaluation for a UI state, at this level user provide information about their

preference and their satisfaction that could allows the refinement of adaptation in case of intelligent systems.

4. Implementation

In this section, we account for the convenience and applicability of the above detailed model. We outline two implementations of the model presented in the previous section. The first implementation regards a Flippable UI for internationalization developed in accordance with UsiXML project specifications [13]. The second implementation consists on an adaptive UI.

```
<contextModel id="CarRental-contextModel_14"
              name="CarRental-contextModel">
    <context id="CarRental-context-ar_AR_14"
             name="CarRental-context-ar_AR">
        <userStereotype id="CarRental-star_AR_14"
          language="ar_AR" stereotypeName="CarRental-star_AR"/>
        <platform id="CarRental-platform_14"
          name="CarRental-platform"/>
        <environment id="CarRental-env_14" name="CarRental-env"/>
    </context>
    <context id="CarRental-context-en_US_14"
             name="CarRental-context-en_US">
        <userStereotype id="CarRental-sten_US_14"
          language="en_US" stereotypeName="CarRental-sten_US"/>
        <platform id="CarRental-platform_14"
          name="CarRental-platform"/>
        <environment id="CarRental-env_14" name="CarRental-env"/>
    </context>
</contextModel>
```

Fig. 6 An XMI Context Model instantiation

Both implementations show a car rental case study. It serves as a preliminary guideline to work on a common scenario. A set of key functional requirements must be considered for implementing the car rental example.

The users must be able to:

- Select the city of interest to pick up the car;
- Specify the period for the car rental;
- Access a set of possible cars and select one;
- See details about selected car;
- Access and select additional car features (e.g. GPS);
- Provide personal information before renting the car;
- Access details about the car rental before submitting the request;
- Change the car rental parameters anytime before confirming the rental.

At a first time, the adaptation focuses on user-related contextual facts, specifically the user's culture. Figure 6 shows an XMI instantiating the contexts of use. Each context instantiation considers the triplet user, platform and the environment.

Adaptations are outlined via a set of transformations that consists of a models transformation at the Concrete UI levels. The implementation of adaptation rules was based on the Java Expert System Shell (Jess) [18]. Jess is an open rule-based engine integrated in the Java platform. An adaptation rule consisted of two main parts: a condition denoting the trigger event and an action. Figure 7 outlines an example of an adaptation rule that defines different facts related to cultures adaptations.

```
(defrule adaptGraphicalCioThatHasResources_toCulture
    "For each fact of type JessGraphicalCioType that has a resource, adapt to culture"
    (declare (no-loop TRUE))
    ?jessCio<- (JessGraphicalCioType (id ?gCioId)(graphicCio ?gCio) (dir ?gCioDir))
    ?jessResource<-(JessResource{cioId == ?gCioId} (resource ?res))
    ?ctx <- (Context)
    =>
    (call ?gCio setContent ?res.content)

    (if (= ?gCioDir nil) then
        (bind ?platform ?ctx.platform)

        (call ?gCio setDir ?platform.dir)
    )

    (modify ?jessCio (isAdapted TRUE))
    (update)

;    (printout t "Adapt a CioType " ?gCio.id "." crlf)
)
```

Fig. 7 A Jess Adaptation Rule example

The implementation architecture looks is depicted in figure 8. The CUI model is instantiated into a UIState. This UI state is firstly transformed into FUI (Java Swing). A change in the context may trigger an adaptation. This adaptation modifies the UIState and thus leads to another transformation to a final UI. The newly transformed FUI represents the adapted version that accommodates the change in the context.

The example follows the unified adaptation model in figure 2. It works as follows: the user sends his feedback through the handler tool (depicted in figure 10, to the left).

The user chooses to change the culture (explicitly set in his profile). This feedback is a TriggerEvent to an AdaptationRule. In the case of passing from a western to Arabic culture, two rules are triggered: translate the language and change the UI layout direction. In the later case, both rules are triggered and conduct a transition on the UI state to a state that is assessed by the ContextAssessment for Arabic culture.

The complete process is depicted in figure 9 illustrating the distribution of the process activities on the three packages of the unified adaptation model.

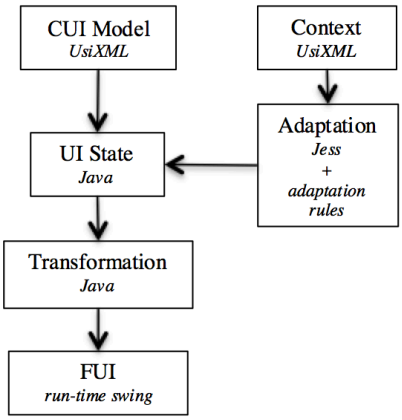

Fig. 8 The implementation architecture

The figure 10 shows a visualization of the execution of adaptation. Adaptations are triggered explicitly by end-users via a control panel. The control panel consists of an implementation of users' feedbacks aimed at adapting the interface regarding theirs evaluation. In the picture at le left area, we present the control panel allowing the manipulation of adaptation.

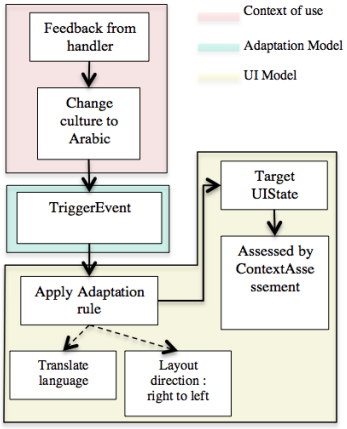

Fig. 9 Case study adaptation steps

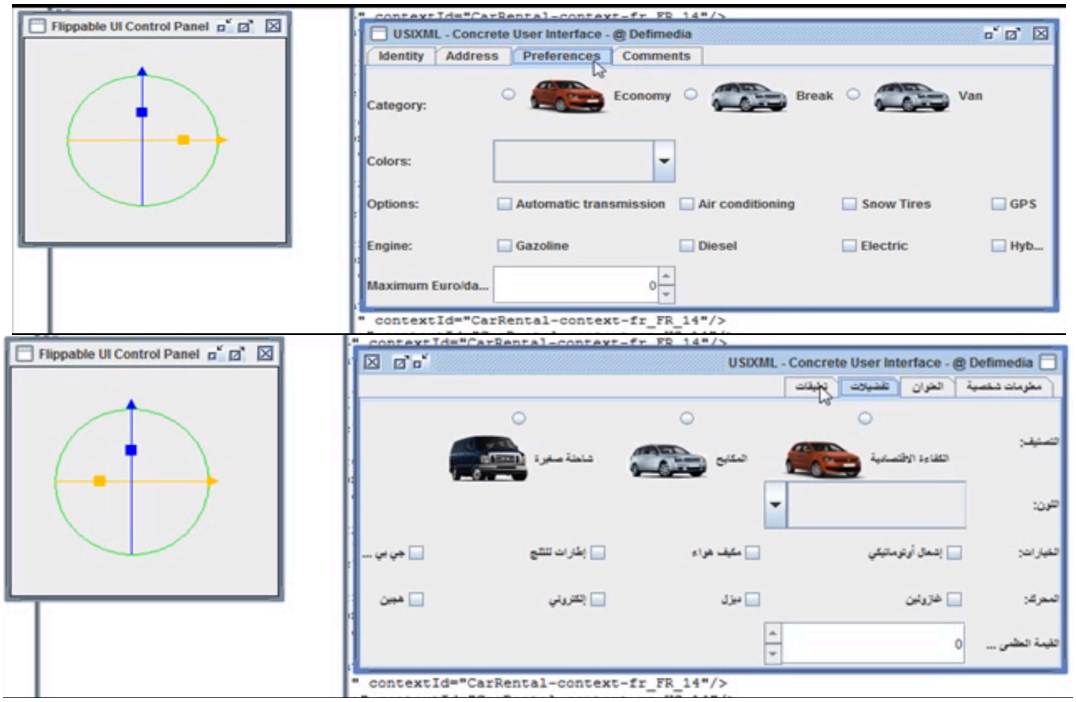

Fig. 10 The Flippable Car Rental example

By moving the handles horizontally, the end user is able to manipulate the geometry in order to adapt the UI.

In this case, the adaptation consists on reversing the UI, and accordingly to change the UI language and orientation regarding cogitated culture facts. In the right area the picture shows how adaptations are visualized in the graphical UI.

A further illustrative mock-up was implemented considering specific contexts, aspects and sharing the same theoretical models. The prototype is based on the same case study presented in the above illustration. It shows another situation for adaptation that considers the platform of interaction. Two platforms were considered: desktop and smartphone.

User can interact with both applications. The screen of the device is the main aspect that permits activate the adaptation of the application. The figure 9 shows the desktop version.

To shift from one platform to another adaptation rules are defined to meet different platform requirement. When a consistent set of requirements are identified, the adaptation process should carry out suitable actions to meet these requirements.

Two levels are considered to accomplish adaptation:

(1) At the interactive level, the interaction workload has been restructured by selecting interaction objects with higher guidance and accessibility. For instance the select color task (illustrated in the green box) is represented by a dropdown list to replace the selection list. This choice is justified by the adequacy with the screen size, the dropdown list allow more visibility and avoid the scrolling to visualize next tasks.

(2) At the presentation level, the formatting instructions have been suitably rewritten for the device. The arrangement and position of interface's elements is defined with regards to the propriety of each device. Such as, for the category specification task, and the disposition of tabs (Horizontal-Vertical).

Next, figure 12 shows the screen layout after the adaptation process. It illustrates how the elements were adapted. For instance, the menu identified on the figure 11 by the red box has been changed by a scrollable menu (see figure 10). Also the select color task represented by the select list (identified by the green box) has been changed by a dropdown list. Both adaptations are realized at runtime according to the screen size of device used by the user.

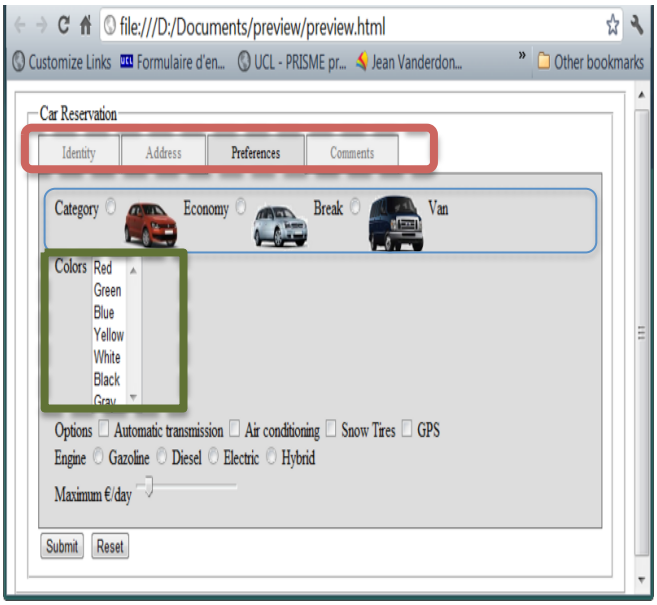

Fig. 11 Car Rental case study for desktop

Fig. 12 Car Rental case study for smartphone

5. Conclusion

This article presents a Conceptual Model for Agile Adaptation designed to supporting an explicit,

comprehensible and complete configuration of adaptation concerns at runtime. It permits developing adaptive and adaptable user interfaces supporting end-user involvement.

The proposed model involves users and end-users for adaptation triggering. It is instantiated via a flippable UI allowing users to adapt the UI at runtime by a control panel.

We will consider realize a methodological framework that considers structural and procedural views. As well we will take into account the study of different solutions to analyze and evaluate the information capture by the model to produce and present the UI adaptation.

A platform prototype for the implementation of runtime context-aware adaptation is foreseen to validate the model. With this prototype, we will be able to easily evaluate the interest and the usability of our proposal by conducing user experiments.

Acknowledgements

We are grateful to the "Jounum Project", the Louvain School Management and the Catholic University Of Louvain for their financial support.

References

[1] Akiki, P. A., Bandara, A. K., & Yu, Y. (2015). Adaptive model-driven user interface development systems. ACM Computing Surveys, 47(1), In-press.

[2] Assad, M., Carmichael, D. J., Kay, J. and Kummerfeld, B. PersonisAD: distributed, active, scrutable model framework for context-aware services. International conference on Pervasive computing (PERVASIVE'07). Springer-Verlag, Berlin, Heidelberg, (2007), pp 55-72.

[3] Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J. A Unifying Reference Framework for Multi-Target User Interfaces. Interacting with Computers 15, 3 (June 2003), pp. 289-308.

[4] Chang, K., Hightower, J., and Kveton, B. Inferring Identity Using Accelerometers in Television Remote Controls. In Pervasive (2009), 151-7.

[5] Evers, Christoph, Kniewel, Romy, Geihs, Kurt, et al. Achieving user participation for adaptive applications. In Proc. Ubiquitous Computing and Ambient Intelligence. Springer Berlin Heidelberg, 2012. p. 200-207.

[6] Order of references

[7] Findlater, L., and McGrenere, J. Impact of screen size on performance, awareness, and user satisfaction with adaptive graphical user interfaces. Proc. SIGCHI Conference on Human Factors in Computing Systems CHI, ACM Press (2008), 1247-1256.

[8] Froehlich, J. et al. UbiGreen: investigating a mobile tool for tracking and supporting green transportation habits. In CHI (2009), 1043-1052.

[9] Fox, D., Sillito, J., & Maurer, F. Agile methods and user-centered design: How these two methodologies are being successfully integrated in industry. In Agile, 2008. AGILE'08. Conference (2008, August), pp. 63-72.

[10] Gajos, K. Z., Everitt, K., Tan, D. S., Czerwinski, M., & Weld, D. S. Predictability and accuracy in adaptive user interfaces. In Proc. the SIGCHI Conf. on Human Factors in Computing Systems, ACM, 2008. p. 1271-1274.

[11] Ganneau, V., Calvary, G., & Demumieux, R. (2007, November). Métamodèle de règles d'adaptation pour la plasticité des interfaces homme-machine. In Proceedings of the 19th International Conference of the Association Francophone d'Interaction Homme-Machine (pp. 91-98). ACM.

[12] Hill, Ernest Friedman. Jess in Action: Java Rule-Based Systems. Manning Publications Co., Greenwich, CT, USA, 2003.

[13] Karen M. Feigh, Michael C. Dorneich, Caroline C. Hayes,. Toward a Characterization of Adaptive Systems Framework for Researchers and System Designer In. Human Factors: The Journal of the Human Factors and Ergonomics Society (2012) 1008-1024.

[14] Khaddam, I., & Vanderdonckt, J.Flippable User Interfaces for Internationalizatio. Third international conference on effective interactive computing systems (EICS) 2011, Pisa, Italy.

[15] Knutov, E., De Bra, P., Pechenizkiy, M. Generic Adaptation framework: a process-Oriented perspective. Journal of Digital Information, (2012).

[16] Knutov, E., De Bra, P., Pechenizkiy, M. AH—12 years later: a comprehensive survey of adaptive hypermedia methods and techniques. New Rev. Hyperme'd. Multime'd. 15(2009), 5–38.

[17] Lavie, Talia and Meyer, Joachim. Benefits and costs of adaptive user interfaces. International Journal of Human-Computer Studies, 2010, vol. 68, no 8, p. 508-524.

[18] Lim, B. Y., Dey, A. K. & Avrahami, D. Why and why not explanations improve the intelligibility of context-aware intelligent systems. CHI (2009), 2119-2128.

[19] Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., & López-Jaquero, V. (2005). USIXML: a language supporting multi-path development of user interfaces. In Engineering human computer interaction and interactive systems (pp. 200-220). Springer Berlin Heidelberg.

[20] Motti, V.. A computational framework for multi-dimensional context-aware adaptation. In Proceedings of the 3rd ACM SIGCHI symposium on engineering interactive computing systems (2011, June) pp. 315-318.

[21] Motti, V,. Mezhoudi, N, Vanderdonckt, J (Machine Learning in the Support of Context-Aware Adaptation. Proceedings of the Workshop on Context-Aware Adaptation of Service Front-Ends (2012).

[22] Sottet, J. S., Ganneau, V., Calvary, G., Coutaz, J., Demeure, A., Favre, J. M., & Demumieux, R. Model-driven adaptation for plastic user interfaces. In Human-Computer Interaction–INTERACT (2007). pp. 397-410. Springer Berlin Heidelberg.

[23] Sotsenko A., Jansen M., Milrad M. Discussion About Contextualization of Learning Objects. In: Proceedings of the 12th World Conference on Mobile and Contextual Learning, 2013.

[24] UsiXML User Interface eXtensible Markup Language: http://www.w3.org/2005/Incubator/model-based-ui/wiki/UsiXML

[25] Vanderdonckt, J. M., & Bodart, F. Encapsulating knowledge for intelligent automatic interaction objects selection. In Proc. INTERACT and CHI conf. on Human factors in computing systems 1993.

Nesrine Mezhoudi she is a Ph.D. student at Université catholique of Louvain (UCL) and member of Lilab (Louvain Interaction Lab). She's currently investigating intelligent UI adaptation and recommendation systems. She worked as a research assistant for the EU-funded FP7 Serenoa project and Jounum project. She collaborated with the working group of W3C on the standardization of Model-based User Interfaces (MBUI).

Jorge Luis Pérez Medina received the M.S. degree in Computer Science from the Centrocidendal Lisandro Alvarado University at Barquisimeto, Venezuela in 2004. He also received the Ph.D. in Computer Science degree from the Grenoble University, France in 2010. He is an IT researcher with a wide knowledge in Computer Science. Passionate about design based on models, processes, Java-based technologies and Human-Computer Interaction. He capitalizes a long track record on academics environments and a strong passion for Modeling Tools and Software Development.

Iyad Khaddam He is a Ph.D. student at Université catholique de Louvain (UCL), and a member of LILab (Louvain Interaction Lab). He's a teaching assistant at UCL. My research focuses on investigating cultural adaptation of user interfaces.