The PDA-LPA Design Space for User Interface Adaptation

Sarah Bouzit[2], Gaëlle Calvary, Joëlle Coutaz
Université Grenoble Alpes, LIG
F-38000 Grenoble (France)
{sarah.bouzit, gaelle.calvary, joelle.coutaz}@imag.fr

Denis Chêne, Eric Petit,
[2]Orange Labs,
28 chemin du Vieux Chêne,
F-38240 Meylan (France),
{sarah.bouzit, denis.chene, eric.petit}@orange.com

Jean Vanderdonckt
Université catholique de Louvain,
Louvain School of Management,
B-1348 Louvain-la-Neuve (Belgium)
jean.vanderdonckt@uclouvain.be

***Abstract*—This paper presents a design space for engineering adaptive user interfaces throughout the user interface development life cycle in order to describe any adaptation technique, adaptable or adaptive, to compare two or more techniques, and to generate new, perhaps unprecedented, techniques. Grounded in the theory of psychological perception, this design space structures the adaptation life cycle into two regulation loops between the user and the system: a perception-decision-action (PDA) loop for both the system and the user, and a learning-prediction-adaptation (LPA) for supporting the adaptation, this last being particularly expressive for adaptivity. This PDA-LPA design space enables defining properties for assessing the quality of these loops between the system and the end-user. This design space of is instantiated on two advanced adaptive user interfaces: adaptive user interfaces based on machine learning and adaptive layouts. This design space provides new insights for considering adaptivity design options.**

Keywords— Adaptation of user interfaces, adaptive user interfaces; design space exploration; quality properties; intelligent user interfaces.

I. Introduction

User Interface adaptation consists in modifying the User Interface (UI) of an interactive system to satisfy specific requirements. Adaptation falls into two categories depending on whom, the system and/or the end-user, is involved in the adaptation process [20,23]: *adaptability* refers to the end-user's ability to adapt the UI [41] whereas *adaptivity* refers to the system's ability to perform UI adaptation [14]. *Mixed-initiative adaptation* [7] occurs when both the end-user and the system collaborate in the adaptation process. UI adaptation has been extensively investigated [20], researched [34], developed [28], and tested [38] with the ultimate goal of optimizing the overall end-users' experience by increasing end-users' performance and/or preferences [11], by reducing task completion time and error rate [5], by improving the subjective user's satisfaction, her learning curve [25].

The challenge is to suggest the right adaptation at the right time at the right place for making it valuable to the end-user [23]. Otherwise, adaptation will be prone to several limitations that could become a major impediment to the expected benefits, if not thwart them [25]: *risk of misfit* (the end-user's needs are incorrectly captured or interpreted), *user cognitive disruption* (the end-user is disrupted by the adaptation), *lack of prediction* (the end-user does not know when and how the UI will be adapted by the system), *lack of explanation* (the end-user is not informed the reasons that triggered adaptation), *lack of user involvement* (the end-user does not have the opportunity to participate actively in the adaptation process), and *risk for privacy* (the system maintains personal information that the user wishes to keep private). "The field of adaptive systems is infamous for its lack of standards, or even commonly accepted approaches" [33]. Surveys of UI adaptation have been published [6,14,18,28,34] as attempts to synthesize adaptation concepts, methods, and tools. However, most of them is technology driven, limited in scope, or surpassed by recent technical progress, which makes them inappropriate for covering the most recent adaptation techniques as well as for exploring alternatives in a systematic and structured manner.

Design spaces are useful conceptual tools for supporting systematic reasoning. A design space is a multidimensional ensemble where a dimension identifies a key issue and for each issue, makes explicit the alternatives available. Design spaces have been used extensively in Human-Computer Interaction (HCI), from software architecture modelling for interactive systems [24] to multimodal user interfaces [12], to device modelling [10]. They have provided the HCI community with shared vocabularies for describing and understanding multiple aspects of interactive systems. On the other hand, "UI adaptation" still lacks a design space that would serve as a reference framework to reason about UI adaptation at the appropriate level of details.

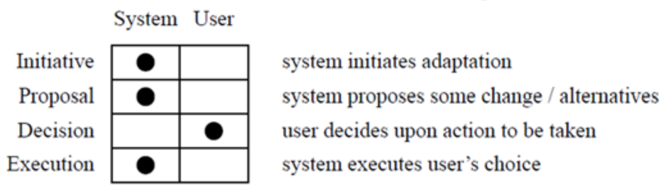
Existing taxonomies and frameworks for reasoning about UI adaption either are too high level to inform UI designers and developers in a useful and effective manner, or they ignore the role of the end-user in the adaptation process, or even do not consider the usability of the adaptation process. In turn, we propose a new design space for UI adaptation that covers both the human and the system as two symmetrical and interacting loops where each loop is decomposed into 6 stages informed by cognitive psychology: a Perception-Decision-Action (PDA) sequence enriched with a Learning-Prediction-Adaptation (LPA) sequence. In addition, the quality of the interaction that takes place between the human PDA-LPA cycle and that of the system cycle are characterized with properties that refine the usability of the adaptation process at multiple levels of granularity: at the loops level or at the stage levels. As a result, designers and developers of UI adaptation can reason at several levels of abstraction. Additionally, the stages of the PDA-LPA loops provide the basis for architectural decomposition.

The article is structured in the following way: Section 2 provides some background on design spaces and frameworks for adaptation, Section 3 defines the PDA and LPA loops as well as a series of quality properties derived from them. Section 4 compares two adaptive systems based on this design space. Section 5 concludes the paper by presenting some future avenues.

II. Background on User Interface Adaptation

Pioneering work on UI adaptation started with **Browne *et al.*** [6]. These authors have used Moran's Command Language Grammar (CLG) [10] that structures the specification of a UI into distinct aspects, from tasks and abstract concepts to syntactic and physical components. They conclude that CLG major strength for UI adaptation is the Principle of Separation of Concerns. Although this principle is enforced in CLG, it is not obvious how to easily propagate all specifications aspects into the final code. In addition, Browne et al. observe that CLG has very limited facilities for expressing UI presentation and behaviour, thus raising the need for improving its expressiveness.

Dieterich *et al.* taxonomy [14] has long been considered as a seminal reference for classifying different types of adaptation configurations and techniques. Based on the analysis of more than 200 articles covering UI adaptation, Dieterich *et al.* propose a four stage classification: *initiative* (which entity involved in the interaction process expresses the intention to perform adaptation), *proposal* (if a need for adaptation arises, what proposals could be applied successfully given the current situation), *decision* (which adaptation proposal best fit the requirements), and *execution* (the adaptation technique previously chosen is finally enacted). Given these four stages, the authors have classified every system with adaptation capabilities according to the actors involved at each stage. Fig. 1 presents an adaptation configuration in which the system recognizes the need for adaptation, proposes some alternatives, the end-user then selects the appropriate alternative that is finally executed by the system. Any other combination of this distribution could be imagined.



	System	User	
Initiative	●		system initiates adaptation
Proposal	●		system proposes some change / alternatives
Decision		●	user decides upon action to be taken
Execution	●		system executes user's choice

Fig. 1. An adaptation configuration in Dieterich's terms.

McKinley's taxonomy [28] addresses software compositional adaptation that exchanges algorithmic or structural parts of the system with others to improve the system's fit to its current context of use, which is considered as the reference for adaptation [9]. This kind of adaptation is based on the separation of concerns between the functional behaviour of the system and cross-cutting concerns, as well as on computational reflection that provides a vehicle to query the different aspects of a system, on component-based design practices that enable the development of the different parts of a system separately, and on a middleware that usually provides the compositional capabilities.

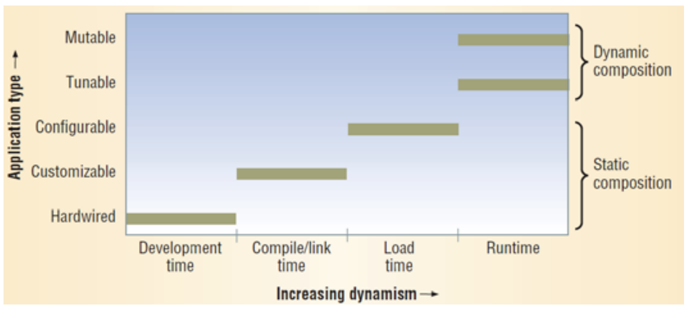
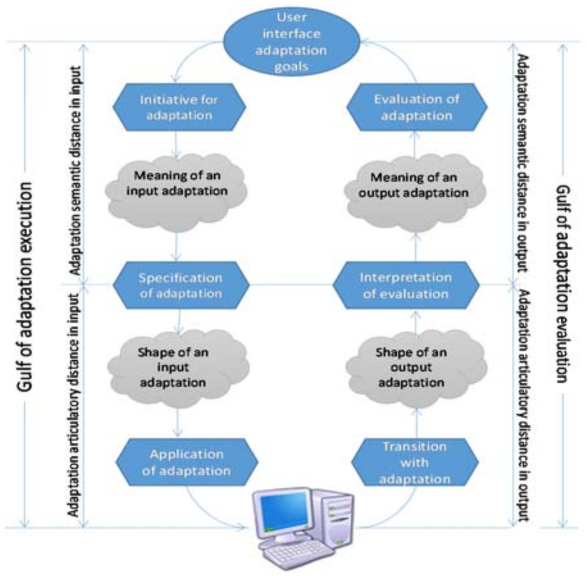


Fig. 2. Adaptation time according to McKinley's taxonomy [28].

The taxonomy is structured along three dimensions:

1. *How to adapt*: the process can be carried out by different entities: a human (e.g., a developer, a system administrator), a component loader, a run-time system, or a meta-object.
2. *Where to adapt*: it describes where, in the system, the adaptation code is inserted. The most common approach is to place the code in the middleware, although extensible operating systems have also been used.
3. *When to adapt*: the adaptation time (Fig. 2) is *static*, respectively *dynamic*, when it takes place at design, prototyping, develop, compile, link, or load time, respectively at run time. In addition, adaptation is *hardwired* (when UI adaptation is embedded in the code of the interactive application, or in the UI code), *customizable* (when UI adaptation enables some pre-computed freedom), *configurable* (when the UI adaptation technique can be configured before execution), *tunable* (when UI adaptation can fine-tune the UI at run time without modifying its code), or *mutable* (when UI adaptation subsumes the run time code modification of the interactive application, as in generative programming [36]).

Although this taxonomy is applicable to an entire interactive application, the three dimensions (i.e. how, where, when) are particularly constructive for a design space: 'how to adapt' corresponds roughly to the proposal stage of Dieterich's taxonomy, 'where to adapt' is implicitly included in the execution stage of Dieterich's taxonomy, whereas 'when to adapt' is not covered by Dieterich's taxonomy. Therefore, McKinley's taxonomy complements Dieterich's proposal, but contrarily to the ISATINE framework [27], they altogether do not cover the adaptation life cycle in sufficient detail to reason about the design and implementation of UI adaptation techniques.



User interface adaptation goals
Initiative for adaptation
Evaluation of adaptation
Meaning of an input adaptation
Meaning of an output adaptation
Specification of adaptation
Interpretation of evaluation
Shape of an input adaptation
Shape of an output adaptation
Application of adaptation
Transition with adaptation
Gulf of adaptation execution
Adaptation semantic distance in input
Adaptation articulatory distance in input
Gulf of adaptation evaluation
Adaptation semantic distance in output
Adaptation articulatory distance in output

Fig. 3. The ISATINE framework for UI adaptation [27].

The ISATINE framework (Fig. 3) [29] structures the adaptation life cycle of a UI into the exact same stages as the seven stages identified by Norman to model human mental steps when performing a task with a system [32]. Inspired by Norman's gulf of execution and gulf of evaluation, ISATINE includes the gulf of adaptation execution and the gulf of adaptation evaluation.

As shown on the left end side of Fig.3, the gulf of adaptation execution covers the stages that the adaptation process goes through to adapt. The gulf of adaptation evaluation, shown on the right end side of Fig. 3, includes the stages involved in the assessment of the outputs produced by the execution gulf. As in Norman's mental model of action [32], the process starts by stating some particular adaptation goal which in turn triggers an initiative for adaptation whose meaning has to be specified, then transformed into a concrete plan before being applied. Once the adaptation has been completed, a transition takes place until interpretation of the adaptation is reached, and evaluated with respect to the initial goal. Feedback may be interpreted *positively* (i.e., the adaptation goal has been fulfilled) or *negatively* (i.e. the adaptation goal has not been achieved) [30]. In the latter case, a new adaptation loop is initiated. Any stage can be carried out by any combination of end-users, system, and other stakeholders.

While more expressivity is gained, the ISATINE framework still misses important aspects: it details the stages of the adaptation process only with respect to the end-user's viewpoint, it does not provide any insight from the system viewpoint. In particular, it does not separate the part related to the main interaction from the adaptation itself; it does not express how adaptivity is achieved; and it does not support the expression of the quality of the adaptation process. Therefore, the 'when to adapt' is covered as well as the 'where to adapt', but not 'how to adapt'. For the aforementioned reasons, we would like to come up with a design space for adaptation that could anchor techniques selected for supporting 'how to adapt' when they are required.

III. The PDA-LPA Design Space

A. Requirements for a Design Space for UI Adaptation

Any design space for adaptive UIs, as any other design space or model, should serve three virtues [3]:

1. A *descriptive virtue*: any adaptive UI should be described completely, consistently, and unequivocally.
2. A *comparative virtue*: any set of adaptive UIs should be made comparable according to the criteria defined in the design space. This requirement supports sound comparative analyses of techniques as well as rigorous benchmarking. When two adaptive UIs are compared in the design space, their coverage can be highlighted, thus enabling an objective identification of their respective strengths and weaknesses, which is referred to as commensurability.
3. An *exploratory virtue*: every dimension of the design space can be systematically explored in order to identify where existing techniques are located, where they are strong or weak, where new opportunities emerge, and where underexplored portions remain to be investigated, if not discovered yet. Such underexplored portions could be subject to a theoretical analysis to determine their feasibility or a prototypical implementation to identify its potential benefits.

Beyond these three virtues, a design space for UI adaptation should cover the co-evolution that occurs between the end-user and the system. As end-users adapt themselves while interacting with the system, the same applies to the system for anticipating end-users actions and for facilitating the interaction. Therefore, two loops must be considered: one for the main UI interaction and one for the adaptation of this UI.

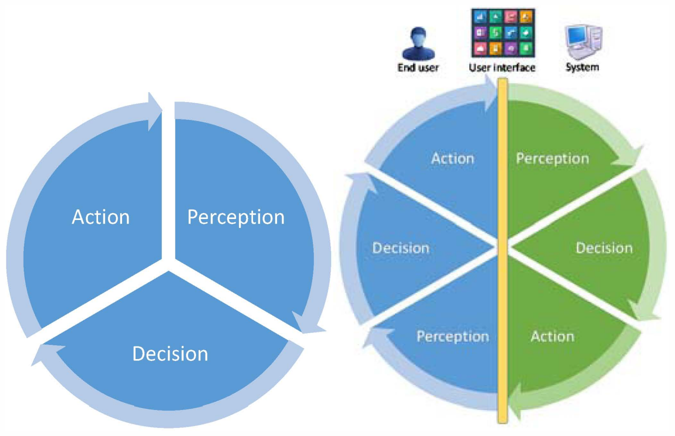


Fig. 4. The Perception-Decision-Action (PDA) cycle in cognitive psychology (a) and its application in HCI (b).

Cognitive psychology [22,32] as well as human decision theory argue that perception, decision, and action are the key activities that humans use to make sense of, and to interact with the environment [9]. Any interactive task thus involves: *perception* when the end-user has to perceive the basic properties of the context of use, inkling the UI, *decision* when the end-user has to make a choice between competing options based on her own interpretation, and *action* when a particular action should take place after a decision has been made. The end-user follows a series of **Perception-Decision-Action** (PDA) cycles (Fig. 4a) until the task is completed. While some researchers claim that these three types of actions are intertwined [15] and could be linked by bidirectional arrows indicating the non-consecutive interaction [20], we prefer to keep the concision of the initial PDA cycle because it equally applies to the UI: the system has to perceive what the user is actually doing (through sensors, user-generated events), to decide what next actions will be undertaken (e.g., through a dialog controller), and to execute the required actions. Hence, the user-system interaction could be represented as a sequence of 2 PDA cycles (Fig. 4b).

In addition to the interaction PDA cycle, we propose the **Learning-Prediction-Adaptation** (LPA) cycle to convey the human-system co-evolution phenomenon mentioned above. The end-user gains experience from the previous PDA cycle and learns. This in turn facilitates the prediction of what to do next for adapting herself to the system. On the system side, the system learns from the end-user's actions, creates and maintains this knowledge for predicting what to offer next to the end-user, and for applying the appropriate adaptation technique. Adaptation cycles cease when either the user or the system decides to suspend the adaptation process. Otherwise, there would be a risk of infinite adaptation without any convergence towards a stable state. Usually, the end-user takes this decision.

B. Design Space Definition

The design space for UI adaptation resulting from these requirements is illustrated in Fig. 5: the end-user's side, graphically depicted in blue, is made up of two cycles (one PDA cycle for the interaction and one LPA cycle for human adaptation) as well as the system, graphically represented in green (one PDA cycle for controlling the system according to the business logic and one LPA cycle for system adaptation).

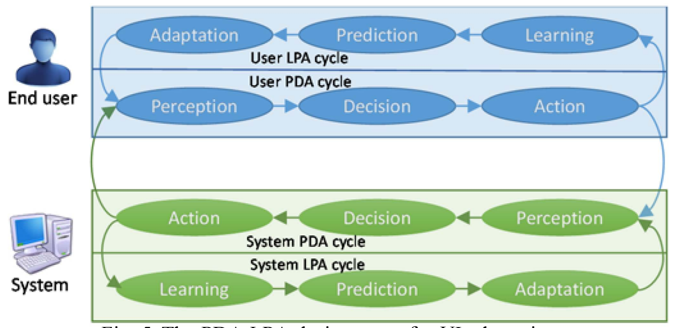


Fig. 5. The PDA-LPA design space for UI adaptation.

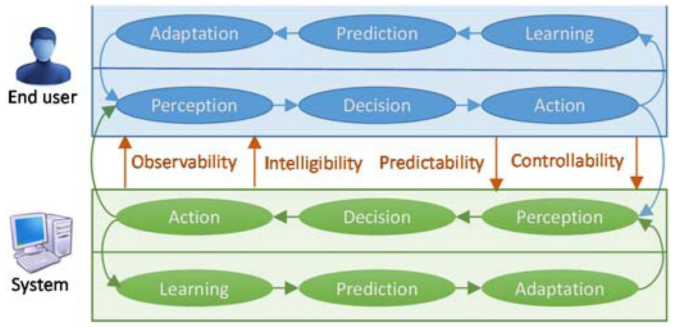


Fig. 6. Global quality properties.

The resulting PDA-LPA design space for UI adaptation therefore makes it explicit the activities from both sides (user and system) while differentiating the main interaction process from the adaptation process. When no system adaptation exists, the LPA-system sequence disappears. When no user adaptation is achieved, the LPA-user sequence disappears. An adaptable UI could be represented equally without the LPA-system sequence since the end-user is in charge of adapting the system. An adaptive UI includes the 4 sequences shown in Fig. 5.

C. Quality Properties

Any adaptation could be described in the terms of the PDA-LPA design space as follows: which technique is promoted in the interactive system for supporting perception, decision, action, learning, prediction, and adaptation when it occurs. This should respect the aforementioned descriptive virtue. In order to respect the comparative virtue, quality properties could be defined on top of the design space, thus revisiting and enriching existing quality properties for interactive systems [21] and simultaneously introducing new abilities [41] for the adaptation. Quality properties in our design space fall into two categories: *system* (software) *quality properties* when the property could be defined as an ability ensured by the system in a user-independent way, or *user quality properties* when the property could be defined as an ability offered to the user, which is therefore user-specific. A quality property is said to be *shared* when both the end-user and the system collaborate to ensure the property. Fig. 6 depicts a first series of global quality properties (i.e., between user and system considered as a whole):

- *Observability*: refers to the system's ability to make perceivable its state in a relevant way for the end-user. This subsumes the interaction (*system observability*) and/or the adaptation (*adaptation observability*). Adaptation observability is key for the end-user to perceive what the system is doing to support adaptivity: in *Learning* (to perceive what the system is currently learning or has learnt from the user, such as user events, actions), in *Prediction* (to perceive that the system is able to predict an adaptation), and in *Adaptation* itself (to render that an adaptation is taking place, such as with animated transition showing how a UI progressively evolves when subject to adaptation [13]). Observability is a generalization of immediate *visibility* since several techniques could ensure observability [21]: *traceability* (step-by-step perceivability) [31] or *browsability* (visibility by navigation on-demand): what is not immediately perceivable could be made browsable on demand. *Browsability* refers to the system's ability for the user to explore the system state by the way of articulatory tasks (i.e., tasks that do not modify the state of the functional core, such as scrolling, zooming in, zooming out, navigating). Some forms of observability are graphically depicted in Fig. 7: learning observability from the learning activity to the end-user, prediction observability from the prediction activity to the end-user, and adaptation observability for the final adaptation executed. Each activity could be subject to observability independently of the others: it is not because the adaptation is observable that its learning and its prediction should be also observable. On the opposite, these three activities are rarely subject to observability, apart from showing the results of the adaptation to the end-user.

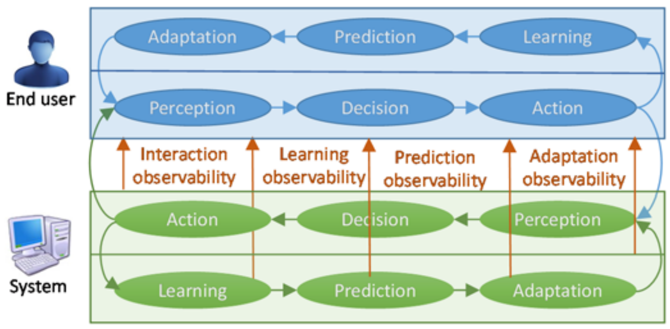


Fig. 7. Various forms of observability.

- *Intelligibility*: refers to the system's ability to communicate to the end-user how the interaction and/or the adaptation processes are conducted in a meaningful and representative way [4]. As intelligibility subsumes observability [39] (before making the adaptation intelligible, it should be made observable), it could be similarly decomposed into *interaction intelligibility* and *adaptation intelligibility*. Intelligibility is key for the end-user to understand the system and the adaptation engine, while they are running. For a context-aware system, intelligibility refers to the system's ability to present itself in a convenient manner, the way the context is perceived and its behaviour depending on significant changes over time [22]. Intelligibility could be ensured by different ways [4]: *explainability* (the adaptation is explained), *continuity* (the adaptation process is continuously rendered) [19], *honesty* [21], or *transparency* (which each adaptation is rendered univocally to the end-user to avoid the aforementioned limitations of adaptivity) [8]. The end-user can understand all the better that she is able to observe the system thanks to observability. *Honesty* refers to the system's ability to achieve two aims: to make the real system state observable to the end-user (which is challenging due to latency and lag [13]) and to make this state accurately [19] interpreted by the end-user.

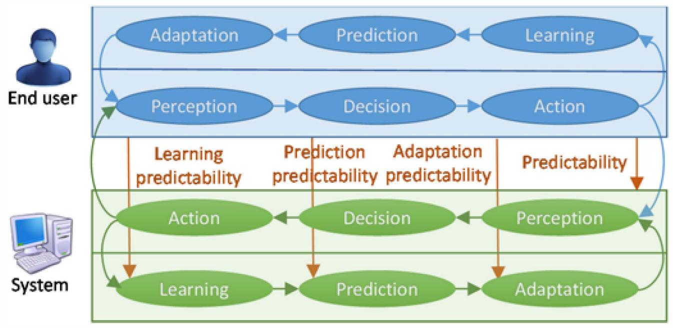


Fig. 8. Various forms of predictability.

- *Predictability*: refers to the user's ability to predict future system actions for supporting the interaction, the adaptation, or both based on past corresponding actions [19]. Regarding the adaptation predictability (see Fig. 8 for various forms of this property), the user should to some extent predict the behaviour of the adaptivity algorithm based on past adaptivity actions, which may subsume controllability: the end-user can predict all the better that she is under control of the system thanks to controllability. Accuracy of the prediction positively impacts the performance and thus subjective satisfaction of the end-user when adaptivity is properly achieved [17,19], which heavily depends on the prediction method.

- *Controllability*: refers to the user's ability to control the system interaction (*interaction controllability*), the system adaptation (*adaptation controllability*), or both depending on which cycles are concerned: the system PDA cycle, the system LPA cycle, or both. Controllability could cover any interactive UI aspect in principle. When the end-user has no control, the interaction and/or the adaptation cycles are entirely initiated and controlled by the system through *self-controlling, self-regulation*, or *self-adaptation* [8,33]. Adaptation controllability is essential to enable the end-user to be actively involved in any adaptation activity: in Learning (to specify to the system what is allowed to capture, interpret, and learn from the user for ensuring privacy), in Prediction (to control the parameters used for predicting the adaptation, e.g., via machine learning), and in Adaptation (to assess adaptation proposals, to accept a relevant adaptation –*true positives*– or reject irrelevant adaptations resulting from wrong predictions –*false positives*–). For instance, a prediction window could display menu items predicted by the system adaptivity technique which may result into a correct item selection (if the item was desired by the end-user) or undisplaying the prediction window and selecting in the initial menu (if them item does not belong to the prediction window) [5].

We defined a global quality property as a quality property between the end-user and the system taken as a whole: from/to PDA-LPA cycle from one entity to the other. By analogy, we define a local quality property as a quality property between one step of the PDA-LPA cycle from one entity and the corresponding step in the other PDA-LPA cycle.

Fig. 9 graphically depicts a series of local quality properties ranging from the end-user to the system (E2S):

- *Awareness*: refers to the user's ability to perceive (hence, the perception activity is concerned) how the interaction, the adaptation, or both are occurring in the system. Awareness could be supported in several ways, such as context-awareness [21] (what are the contextual conditions that are estimated significant enough to trigger a change of context), action awareness (what are the actions undertaken by the system), decision awareness (how the system decides which adaptation), and perception awareness (how the system perceive the context of use). Awareness is positively influenced by corresponding system properties, such as observability and honesty: the more the system is observable and honest in what is observed, the more the user is capable of being aware of the system state, information, and actions.
- *Decidability*: refers to the user's ability to decide (hence, the decision activity is concerned) what to do (in the interaction cycle) and/or how to adapt herself (in the adaptation cycle). In case of a system with internal control, there is no possibility for the user to decide anything; on the opposite side of the continuum, the user may decide everything in case of a full external locus of control. Between these two extremes, *mixed-initiative* [8] prompts the user with several options on what actions to undertake next, on which adaptation could take place so that the user may decide while knowing the potentially positive and negative consequences of this decision.
- *Triggerability*: refers to the user's ability to trigger (hence, the action activity is concerned) the actions she wants (in the interaction cycle) and/or the options needed for an appropriate adaptation. In adaptability for instance, the user is given the opportunity to adapt some UI features (e.g., icons, menu shortcuts, toolbar contents and position) at any time. Some adaptation may be allowed or forbidden depending on the context of use or because the consequences will be beneficial or fatal for the end-user. Deactivation of actions and adaptations is a typical example for revealing or hiding (un)triggerable actions. Smart menus displaying first basic menu items, and then progressively more complex items as the end-user's experience grows, is another form of triggerability.

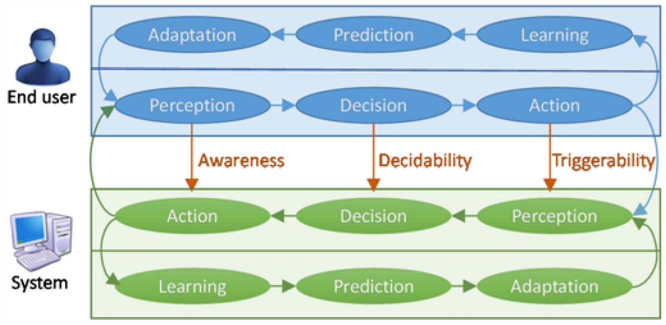


Fig. 9. End-user to System (E2S) local quality properties.

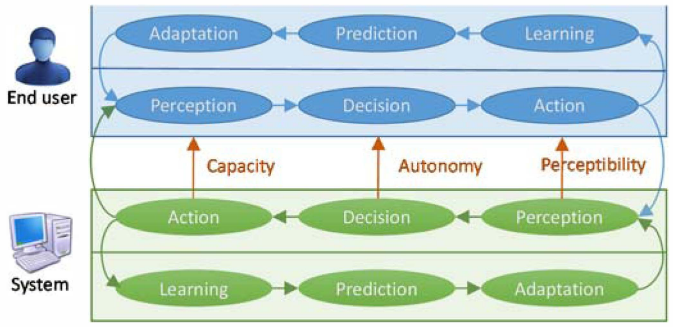


Fig. 10. System to End-user (S2E) local quality properties.

Fig. 10 graphically depicts a series of local quality properties ranging from the system to the end-user (S2E):

- *Capacity*: refers to the system's ability to execute (hence, the action activity is concerned) either domain actions (belonging to the business domain of activity) or adaptation actions or both [41]. The capacity for adaptation reveals the power of the adaptation techniques offered to the end-user in terms of applicability. The end-user may want to trigger some adaptation (in the triggerability), but the system has no capacity for carrying out the required adaptation. This may represent a mismatch between the adaptation goals and the system capacity to achieve them.
- *Autonomy*: refers to the system's ability to decide (hence, the decision activity is concerned) either domain actions or adaptation actions or both. Autonomy may be governed by the locus of control: if adaptation actions are made available, the system may have the permission to decide which one to apply with or without the consent of the end-user. When the system is given the full autonomy, the locus of control is completely internal. When the system has no autonomy, the locus of control is completely external. For instance, menu items in a pull-down menu could be re-arranged by the end-users (adaptable menu) or by the system (adaptive menu) or both (mixed-initiative menu) [17]. Agent technology is recognized for having autonomous agents that carry out tasks for the end-user almost automatically, without the direct intervention of end-users or others, sometimes even without any observability and controllability.
- *Perceptibility*: refers to the system's ability to perceive (hence, the perception activity is concerned) domains and/or adaptation actions achieved by the end-user. These actions could be interactive tasks in the course of the interactive system itself or manual tasks that are outside the system, but yet perceivable (e.g., thanks to camera-based computer vision, situation and activity detection).

Capacity, autonomy, and perceptibility are three local quality properties expressing system abilities. In order to refine these properties in terms of potential benefits for the end-user, further sub-properties could be also introduced:

- *Accuracy*: refers to the degree to which perceptibility is achieved. When an adaptive system accurately perceives the user and probes her context of use (e.g., by detecting the platform used in which location), it provides end-users with some comfort and trust and could reinforce the end-user's feeling that the system is performing accurate actions. For instance, predicting frequently used menu items on a smartphone should be accurate to be accepted by the user [5].
- *Adequacy*: refers to the degree to which autonomy is achieved. When an adaptive system adequately decides an adaptation that is considered suitable for the end-user, it provides end-users with some subjective satisfaction [25].
- *Stability*: refers to the degree to which capacity is achieved. When an adaptive system has to ensure a high degree of stability in the interface adaptation, it means that it should be capable of ensuring consistent adaptation actions that are subject to a smooth transition from the status before adaptation to the status after adaptation.

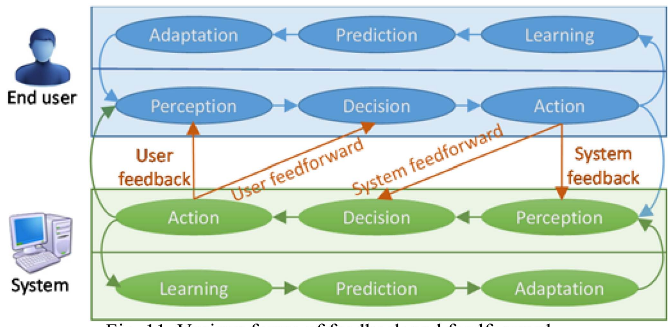


Fig. 11. Various forms of feedback and feedforward.

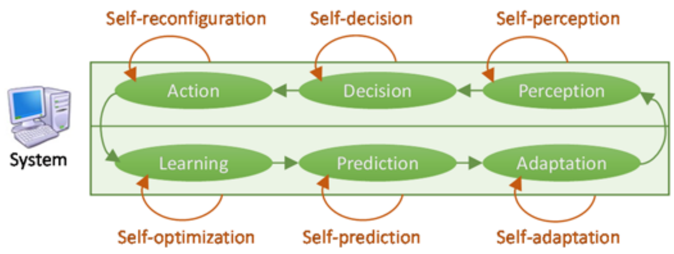


Fig. 12. Various properties of self-management.

Some classical properties could be represented on top of the design space, such as various forms of feedback (Fig. 11) [30,35] and self-management properties (Fig. 12) [33]:

- *Feedback*: refers to any entity's ability to provide the other entity with any information in return to an action executed by the other entity, whether it is the user or the system [15]. When the end-user executes an action (in her PDA cycle), the system needs to perceive it (in its PDA cycle) and the user should be provided with some immediate feedback (e.g., an information message, a process running), which is covered by system feedback since the system is responsible. When the system executes an action (in its PDA cycle), the end-user should be notified as well, which is covered by system feedback by the end-user (e.g., by a progress bar, by acknowledging a command, by accepting an adaptation proposal). Whereas the feedback always occurs after an action has been carried out, preferably immediately after termination, the next property should occur even before any action could be undertaken or while an action is being formulated by the end-user. For instance, a pen-based gesture could display some feedback indicating the recognition results before initiating the corresponding command.
- *Feedforward*: refers to any entity's ability to provide the other entity with any information before this last entity will execute any action [15]. *User feedforward* occurs when the system produces any action just before the end-user will do her task, thus helping her to decide whether this task is indeed appropriate. For instance, the system shows possible pen-based gestures while the user is producing them, thus providing feedback before the final gesture is produced. On the contrary, *system feedforward* occurs when the end-user produces any action just before the system will initiate a task. For instance, a wizard may be redirected by a user action while running, thus dispatching to another branch.

Self-management in general refers to a set of system abilities to deal itself with various issues without requesting any external operation from the end-user or another entity [8,33]. These abilities could be again structured along the cycles:

To:	From:	User						System					
		Per.	Dec.	Act.	Lea.	Pre.	Ada	Per.	Dec.	Act.	Lea.	Pre.	Ada.
User	Perception	Self-consciousness and behavior						Perception observability and intelligibility	Decision observability and intelligibility	Action observability and intelligibility	Learning observability and intelligibility	Prediction observability and intelligibility	Adaptation observability and intelligibility
	Decision												
	Action												
	Learning												
	Predicting												
	Adaptation												
System	Perception	Perception predictability and controllability						Self-perception					Self-management
	Decision	Decision predictability and controllability						Self-decision					
	Action	Action predictability and controllability						Self-reconfiguration					
	Learning	Learning predictability and controllability						Self-optimization					
	Predicting	Predicting predictability and controllability						Self-prediction					
	Adaptation	Adaptation predictability and controllability						Self-adaptation					

Table 1. Property mappings between end-user and system.

- *Self-reconfiguration*: refers to the system's ability to reconfigure itself by undertaking appropriate actions. A newly discovered service is added. Self-protection occurs when an adaptive UI is required to protect itself from attacks and end-users who inadvertently make software changes and errors.
- *Self-decision*: refers to the system's ability to decide itself to initiate any action. For instance, dynamic programming exhibits the capability of the system to generate new rules which, when triggered offer new decision opportunities.
- *Self-perception*: refers to the system's ability to perceive its own functioning. For instance, fault-tolerant system may detect this it is no longer properly working and re-initialize.
- *Self-optimization*: refers to the system's ability to learn itself from its own knowledge in order to optimize its functioning. An adaptive system must improve its learning and rules applied for adaptation for providing good predictions.
- *Self-prediction*: refers to the system's ability to predict itself when a self-adaptation may occur. For instance, a context-aware UI could probe the context of use to deduce that an adaptation of its behaviour is likely to happen if the context of use is continuously and regularly accessed.
- *Self-adaptation*: refers to the system's ability to adapt itself depending on new requirements, whether they are functional or non-functional requirements.

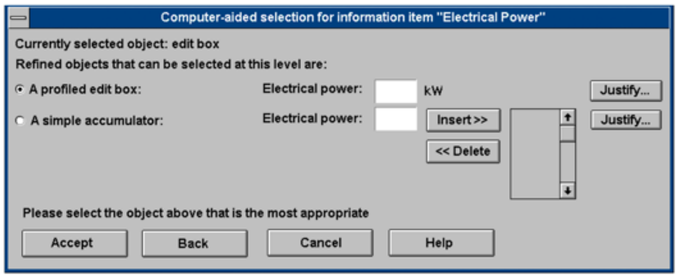
The adaptivity relevance strongly depends on the performance of the predictability. The challenge is to make the adaptation predictable to the end-user while maintain accuracy [19]. For this purpose, the system is required to be stable [18] in order to help users understanding the system and creating a mental model of the UI. In addition, the system should be transparent [23] by explaining the proposed prediction. Also, transparency is required about its learning as well as about the decision made and the chosen format of prediction presentation. Table 1 provides an overview of the quality properties arranged by origin (i.e., initiated by the end-user or the system) to destination (i.e., targeted to the end-user or the system). Self-management properties are merged in the bottom-right quadrant since they are valid for the whole system. Each self-* property always goes from the system to the corresponding step in the cycle, e.g., self-perception to denote the system ability to perceive itself. Properties from and to the end-user are relevant to psychology and philosophy: they are merged in the top-left quadrant since they are out of the scope of this paper.

IV. Comparing Two Adaptive User Interfaces

This section will describe two adaptive UIs developed according to the design space (descriptive virtue [3]) and compare them to each other (comparative virtue [3]).

A. Adaptive UI with Intelligent Widget Selection

We developed FFUI, an environment for form filling with an adaptive UI augmented with machine learning on how widgets are selected. The process is structured as follows: the designer creates a form based on standard widgets and stores its definition in a XML (UsiXML) file that is then interpreted at the client side. At any time, the user could enter in an End-User Development (EUD) mode where each form widget could be edited. Depending on the data type, the number of possible values, the number of domain values, and the data semantics, the system relies on a *decision tree* to select automatically the most appropriate widget for each form field, which the user may change. For example, in order to enter the electrical power of a train expressed in kW unit, a profiled edit box could be selected or a simple accumulator where possible values are progressively added to a list box for future usage (Fig. 13). The "Justify" button, when clicked, reaches the conclusion in the decision tree in order to understand which rule has been fired (Fig. 14).


Computer-aided selection for information item "Electrical Power"

Currently selected object: edit box
Refined objects that can be selected at this level are:

A profiled edit box: Electrical power: kW

A simple accumulator: Electrical power:

Please select the object above that is the most appropriate

Fig. 13. Changing the widget selection for a form field.

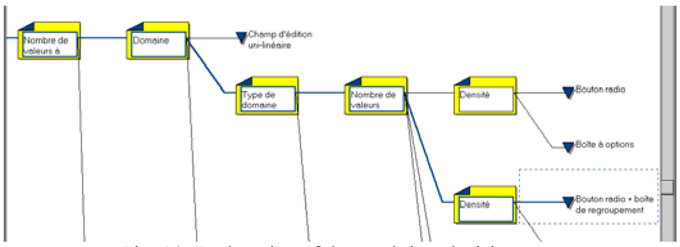


Fig. 14. Explanation of the result in a decision tree.

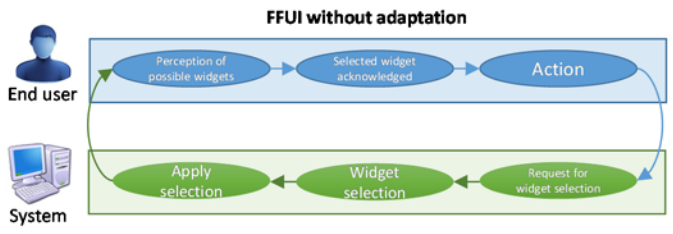
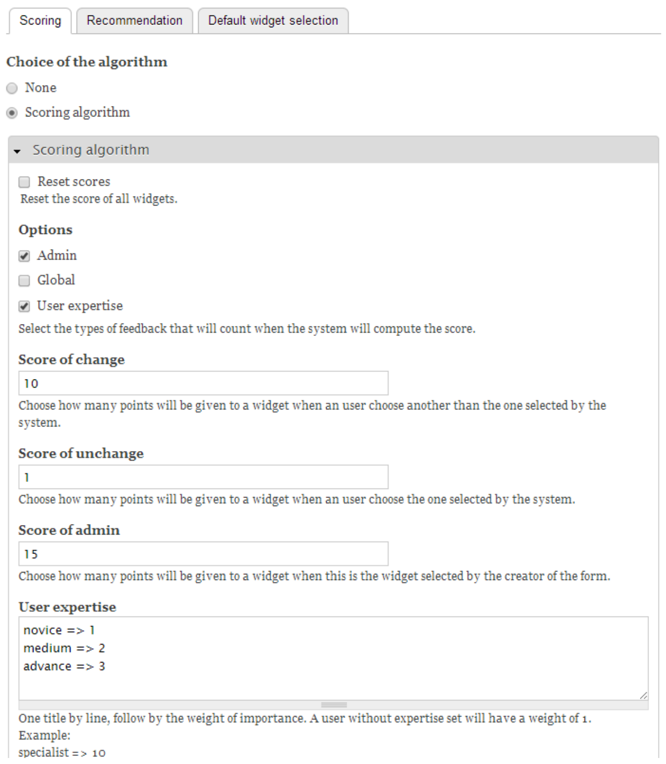

FFUI without adaptation
End user
System
Perception of possible widgets
Selected widget acknowledged
Action
Apply selection
Widget selection
Request for widget selection

Fig. 15. PDA cycles for FFUI without any adaption.

Until now, FFUI does not perform any adaptation: widgets selected for each form data do not change over time. Fig. 15 represents this situation based on the design space: when the system is requested to select a widget according to meta-data (system perception), the decision tree is activated in order to perform the widget selection (system decision) and apply it (system action). The only thing that the end-user could do is to ask the system why a particular widget has been selected (user perception) and acknowledge it (user decision). Consequently, observability is limited to interaction and ensured through browsability since on-demand selection is enabled (Fig. 13). Intelligibility is ensured through explainability (Fig. 14). We hope that predictability is reinforced by consistency since the system always selects the same widget in the same circumstances, thus enabling the end-user to perform some inference. Unfortunately, controllability is non-existent since self-controlling is achieved: the system does not enable the end-user to change any selected widget. In order to better support these quality properties, FFUI has been augmented with a Machine Learning (ML) technique enabling the system to learn which widget should be selected under which circumstances by letting three stakeholders to hear their voice: the end-user with her own reasons and preferences, the designer who is assumed to be a usability expert, and the developer who is responsible for implementing the selected widget.



Choice of the algorithm

None [✓ Scoring algorithm]

Scoring algorithm

☐ Reset scores
Reset the score of all widgets.

Options

☑ Admin
☐ Global
☑ User expertise

Select the types of feedback that will count when the system will compute the score.

Score of change

10

Choose how many points will be given to a widget when an user choose another than the one selected by the system.

Score of unchange

1

Choose how many points will be given to a widget when an user choose the one selected by the system.

Score of admin

15

Choose how many points will be given to a widget when this is the widget selected by the creator of the form.

User expertise

```
novice => 1
medium => 2
advance => 3
```

One title by line, follow by the weight of importance. A user without expertise set will have a weight of 1.
Example:
specialist => 10

Fig. 16. The default scoring function.

The ML algorithm is based on a scoring function determining the weight of each widget candidate. Initially, each widget candidate (e.g., a profiled edit box vs an accumulator in Fig. 13) is assigned to a default score according to a scoring function which is defined as (Fig. 16):

$$\text{Default Score}_{(\text{Widget})} = P{*}SC + D{*}SU$$

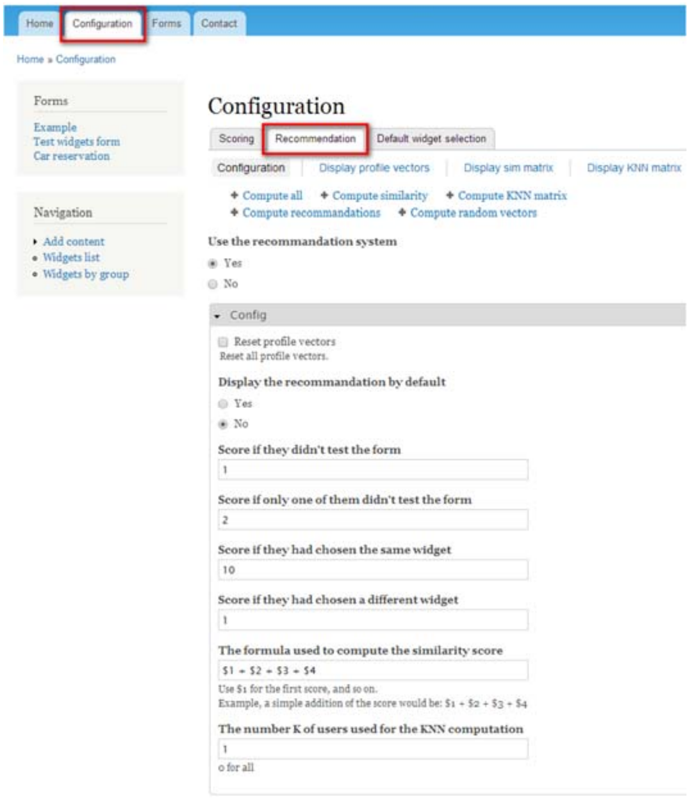
where *SC* is the score of change which defines the additional weight assigned to a widget after being selected by a user or a designer for a specified form. Such score allows the promoting/demoting of widgets with regard to users and experts.

SU is the score of unchanged which defines the interest accorded to the system choice in term of rewarding a well-behaved recommendation within a reinforcement-learning paradigm.

P denotes the number of times that the widget is selected by the end-user without being displayed by default ($W_{Selected} = W_{Default}$).

D denotes the number of times that the widget selected by the end-user is the displayed one ($W_{Selected} \neq W_{Default}$).

Various people may assign a different score to the same widget in the same circumstances, thus explaining why there is a need to consider the end-user's choice or preference, but also the designer's choice (which is assumed to be more experienced), and also the administrator's choice (which is supposed to be the developer knowing which widget is available when). Other actors could play a similar role in this function, such as members of a crowd for crowd-based adaptation of UIs [31].



Configuration

Use the recommandation system

[✓ Yes] No

Config

☐ Reset profile vectors
Reset all profile vectors.

Display the recommandation by default

Yes [✓ No]

Score if they didn't test the form

1

Score if only one of them didn't test the form

2

Score if they had chosen the same widget

10

Score if they had chosen a different widget

1

The formula used to compute the similarity score

```
$1 + $2 + $3 + $4
```

Use $1 for the first score, and so on.
Example, a simple addition of the score would be: $1 + $2 + $3 + $4

The number K of users used for the KNN computation

1

0 for all

Fig. 17. The scoring algorithm configuration

In order to integrate all these choices, a global scoring function is defined for recommending any widget (Fig. 17) [29]:

$$\text{Score}_{(\text{Widget})} = P*SC+D*SU+T*SG+f(w,SA)$$

where *f(w,SA)* determines whether the selected widget matches the designer recommendation and *T* is the total number of widget selections.

SA is the designer's score assigned to the widget in this context based on usability engineering and guidelines, such as graceful degradation rules.

SG is the global score based on previous options.

The global scoring function is continuously computed while the end-user is interacting with the form. Of course, the scoring function can be tailored to another formula (Fig. 17) whether a most accurate schema could be determined and the recommendation can be activated or deactivated. Each end-user action is recorded in a log file that is attached to any particular widget of any particular form, thus making it unique for each context. In FFUI, every user action (A_u) is recorded in a log file (P_s) that automatically triggers the computation (A_s) of a new score. Since this function is always computed for every widget affected by the action, the system decision (D_s) is bypassed. The end-user may perceive the widgets candidates (P_u) by entering in FFUI design mode (like in Fig. 14) by pressing a control key, then decide whether a new widget should be selected (D_u). This decision in then translated into a new action (A_u).

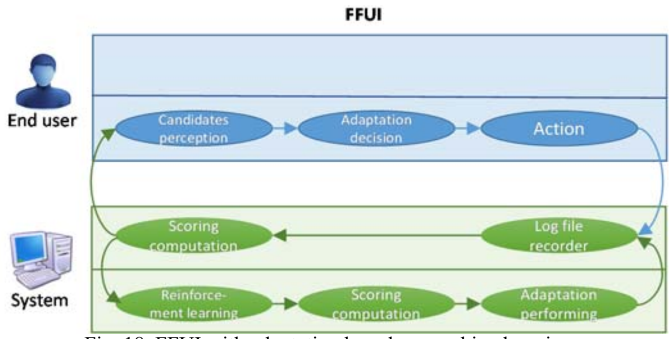


Fig. 18. FFUI with adaptation based on machine learning.

B. Adaptive GUI layout

We developed TAsk-BAsed Design Adaptive (TABADA), a software enabling end-users to carry out interactive tasks as explicitly defined in a task model with run-time adaptive layout based on machine learning. Our approach follows existing model-based work specifying the UI at a higher level of abstraction [9,37]. We consider task-oriented language (task tree) and an Abstract UI specification (AUI) in order to remain model-based and to allow for a greater flexibility in generating UIs from abstract levels [9]. TABADA is distinctive by the use of machine learning techniques for runtime adaptation and their deployment together within a model based approaches. TABADA exploits the user behaviour 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 module called user behaviour predictor. This module uses a machine learning technique based on statistics in order to predict the next action(s) that will be accomplished by the user given the previous ones he filled. The prediction is implemented via a UserActionPrediction class, and can be seen as an 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. The process of user behaviour predictor is based on Markov chains as follows:

- Generating and monitoring sequences of actions based on various parameters.
- Learning an *n* order Markov chain model (or all the order from one to *n*).
- Predict the next most probable action of the user thanks to its history of immediate action.

When TABADA is executed, a first by-default GUI layout is generated. All end-users actions are then recorded (e.g., filling in a field, selecting a new tab, making a choice) so as to feed the ML algorithm. Based on (un)used parts of the task model and on interaction traces, TABADA computes the most probable interaction paths. At any time, the end-user can stop the system and ask for alternate adapted GUI layout that better suit her task. This new user action enters in a new cycle, thus triggering a new score computation (A_s), which reinforces the learning (L_s): the system learns that the end-user has preferred to rely on another widget (SC is updated) or not (SU is updated), which is reflected in a new score computation (P_s). The system then applies the adaptation (A_s) by saving the widget current state, by substituting the old widget by the new one [18], and by restoring the current state into the newly selected widget. A new loop is then initiated.

In TABADA, every end-user action (e.g., using a particular widget, navigating between views, changing the value of a field) is recorded by a sequence monitor (P_s) which records all sequences on top on an internally-maintained task model, which is unfortunately invisible to the end-user (Fig. 17). TABADA does not decide to perform any adaptation (no D_s) and always automatically generates a series of alternate layouts based on previously recorded sequences (A_s). These alternate layouts could be made observable on-demand by the end-user (P_u), among which the end-user may pick one layout or not (D_u), and requests (A_u) the system to switch to this alternate layout. This request is in turn recorded in the system (P_s), which refreshes the generation of sequences (A_s) and updates the Markov chain accordingly (L_s) that again identifies the most probable sequences (P_s). A new loop is generated and so on. Note that in this case, there is a minimal LPD for the end-user since she was conscious of the new layout selected based on previous task sequences, thus learning (L_u). This may help her to predict how a new layout could be computed (P_u).

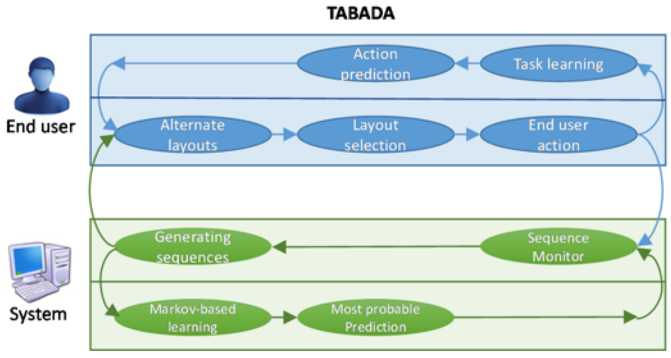


Fig. 19. The Design Space for TABADA.

C. *Comparison of two Adaptive User Interfaces*

Table 2 provides an overview of how both FFUI and TABADA are addressing the quality properties. For both FFUI and TABADA, *observability* is assessed as medium since system and adaptation observability are ensured by browsability: the end-user does not immediately see that there is an adaptivity process ongoing, but could access to the adaptivity control panel by clicking on a control key in FFUI or on an icon in TABADA.

Quality property		FFUI	TABADA
Global	Observability	◑	◑
	Intelligibility	◑	●
	Controllability	◔	◕
	Predictability	○	◔
Local	Awareness	◔	◔
	Decidability	◕	●
	Triggerability	●	●
Local	Capacity	●	●
	Autonomy	○	○
	Perceptibility	◕	◕
Local	Accuracy	●	●
	Adequacy	?	?
	Stability	●	●
Self-management	Self-reconfiguration	○	○
	Self-decision	○	○
	Self-perception	◔	◔
	Self-adaptation	○	○
	Self-prediction	◕	◕
	Self-optimisation	○	○

Table 2. Comparison of FFUI and TABADA in terms of quality properties.

Some users even do not notice the control key and the icon. Nothing else informs the end-user that some adaptivity is ensured by the system, thus suggesting that this properly should be largely improved.

Intelligibility is assessed as superior in TABADA than in FFUI because the alternate layouts are directly rendered, thus giving the end-user a real preview before adaptation, as opposed to only a widget candidate in FFUI that is not rendered in a potential new UI. Some explainability is possible in FFUI though. *Controllability* is limited in FFUI because only the choices suggested by the system are proposed, thus keeping very limited control over these choices. In contrast, TABADA presents the six most probable layouts among which the user can choose and give access to a large set of alternate layouts on demand, potentially all possible combinations.

Predictability is limited in FFUI because the system does not deliver any information that would help the end-user to imagine which kind of widget would result from the adaptivity process. TABADA is slightly better because sequences of actions followed by the users are represented, thus giving some hint on how they are built. Similarly, predictability is certainly a property that deserves a better attention to improve the end-users' ability to trigger adaptation.

For both systems, *awareness* is very limited for the same reasons as in observability: the user cannot be really aware of the adaptivity process since there is no rendering of this process while running.

Decidability is almost excellent for FFUI (not all choices could be made, only among the possible widget candidates) and excellent for TABADA since all possible layouts could be selected, although this could induce a long navigation between the alternate layouts.

Triggerability is assessed as maximal for both since each candidate, once selected, is immediately incorporated in the new UI, which explains why *capacity* is also assessed as maximum since both systems have the ability to build the new layout and to apply substitution. *Autonomy* is assessed as non-existent since only the end-user can make a decision on which widget or layout should be kept. Both systems have no initiative and no decision to trigger any adaptation. *Perceptibility* is assessed as very good since both systems are able to perceive what the user is doing by capturing and recording her actions into an internal log file. *Accuracy* is assessed as excellent in both cases since the new adapted layout will be exactly the one built with the new adaptation decision. *Adequacy* cannot be estimated at this stage because it should require a user experiment to determine whether the adaptation proposals are adequate enough for the end-user. What could be captured however is the extent to which the system proposes candidates that are accepted or rejected by the end-user. *Stability* is always ensured because the same algorithm is always applied in both cases, although we could imagine that different algorithms for adaptation (e.g., based on different machine learning algorithms) could be competing.

Self-management properties are almost not ensured since both systems apply adaptation only after an end-user decision. However, both systems are able to perceive their own state (*self-perception*) and are still able to make predictions at any time (*self-prediction*) without waiting for the end-user, but without any ability to apply what has been predicted, hence *self-optimisation* is non-existent.

In conclusion, several quality properties are affected by the constraints imposed by the system: only the user can make a decision of what prediction could be applied. The system cannot take any decision, therefore preventing mixed-initiative. This generates a new suggestion for both systems: how and when to delegate adaptivity to the system and/or embark into a conversation between the user and the system to decide which candidate is the most appropriate based on respective knowledges.

D. *How to Use the PDA-LPA Design Space*

In this section, we elaborate some guidelines on how to practically use the PDA-LPA design space for designing adaptation for interactive systems.

Guideline 1. Balance the support of the 4 global properties. The most important PDA-LPA quality properties are the four global properties: observability, intelligibility, predictability, and controllability. It is more important to design UI adaptation in order to cover a minimal support of the four properties simultaneously in a balanced way than focusing on one property only. If one or two properties are largely satisfied with little or no support for the other ones, the general LPA process will not deliver its full potential. The level of support between the four properties should be minimal, but balanced. They could also imply some sequence; observability should be thought first, otherwise little or no intelligibility. Similarly, if it is not predictable, it will hardly become controllable.

Guideline 2. Invent a manipulable representation. In order to properly support the four global properties, a representation of the artefacts subject to adaptation should be invented so that it could be manipulated throughout the properties. This representation should be preferably adequate for direct manipulation, but not necessarily: if the representation of the artefact subject to adaptation could be directly manipulated, which is usually more complex to implement, the end-user will see it (observability), understand it (intelligibility-provided that the corresponding metaphor is straightforward), perhaps predict it (predictability), and control it (controllability). If no such representation exists, the end-user should develop a mental model of the adaptation process which will be very hard to complete. The fact that the representation is directly or indirectly manipulable counts less than its existence. Three types of representation may considered: *external* if the representation is the one seen by the end-user, *internal* if the representation that is managed by the software implemented by the developer, or *conceptual* if the representation is an abstraction of the artefact introduced by the designer to simplify the realm of the artefact. For instance, adaptive menus for smartphones [5] invoke a prediction window displaying the most likely to be selected items (Fig. 20).

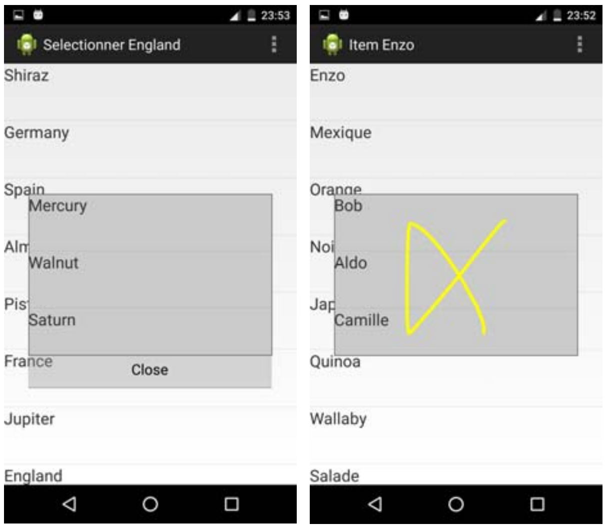


Fig. 20. The Prediction window for adaptive menus.

Guideline 3. Maximise the controllability. End-user hate to lose control over any process, unless they prefer to delegate the responsibility of a task to a particular agent. Therefore, controllability is key [13]: as soon as the representation is invented, feedback on this representation should be envisioned so that the system could express its actions as well as the end-user. For instance, the end-user could select any menu item from the prediction window in Fig. 20 either in graphical or in tactile mode or close it graphically or by gesture. Not all actions should be however subject to control: it is important to identify which adaptation action should be subject to a corresponding user interface action in order to reflect its control. Some experience shows that the more control is offered to end-users to see how adaptivity is achieved, the less they require it over time [13]: an animated transition for explaining an adaptivity could be played slowly the first time and quicker the next times. Or even no longer need after several animations have been operated.

Guideline 4. Refine global properties by local properties afterwards. Considering the local quality properties as well as their sub-properties should come after supporting the global quality properties. For instance, ensuring observability counts more than knowing how to support it, by traceability or by browsability. For instance, in FFUI, observability is ensured first by browsability (Fig. 13), then by traceability (Fig. 14), with a direct positive impact on explainability. In the adaptive menus for smartphone (Fig. 20), observability is immediately ensured through direct observation: no browsability. The more observable an adaptation is, the more intelligible it becomes; the more intelligible an adaptation is, the more it supports predictability. The more predictable an adaptation is, the easier the controllability could be managed.

Guideline 5. Explicitly cover the LPA cycle. Several techniques for adaptation have been reported to cover the PDA cycle [14], but no study exists today that produced an inventory of similar techniques for covering the LPA cycle where the learning is key. Consequently, the focus has been often emphasized on the adaptation (PDA) cycle, and less on the learning (LPA) cycle. This last one should be explicitly covered, preferably with the same minimal level of support that has been used for the PDA cycle. For instance, in FFUI with machine learning, the scoring function (Fig. 17) could be entirely redefined, thus offering a basis for maximal observability and controllability of the learning (LPA) cycle. In TABADA however, different layouts could be produced, each along with its explanation, but the process to obtain them is not observable, therefore not controllable. The final choice remains controllable fortunately.

Guideline 6. Prioritize local quality properties. Three sets of local quality properties have been introduced, that are usually considered by decreasing order of importance. This ordering may be different depending on the type of interactive application, thus requiring a prioritization scheme.

V. CONCLUSION

In this paper, we presented a design space for user interface adaptation that departs from existing design spaces (e.g., [6,8,14, 18,34]) in terms of quality properties defined from a software viewpoint (often called "*ilities*" since most of them finish with this suffix expressing some ability-based behaviour [41]) instead of a set of independent, not interconnected properties (e.g., stability, visibility). This design space is explicitly based on the Perception-Decision-Action (PDA) cycle coming from cognitive psychology [32], which is itself augmented by a second cycle Learning-Prediction-Action (LPA). This design space supports the three expected virtues [3]: descriptive, comparative, and generative.

Future avenues of this work include: *(i)* the introduction of time to qualify the time constraints between the PDA-LPA steps and to determine, which is the most appropriate moment to ensure them, not always continuously; *(ii)* the conducting of a Systematic Literature Review (SLR) on a series of papers like in [14], but on a recent base of references; and *(iii)* the refining of the Prediction step into three sub-steps according to Endsley' model of situational awareness [16] where situation awareness is decomposed into perception, comprehension, and projection.

REFERENCES

[1] S. Abrahão, E. Iborra, and J. Vanderdonckt, “Usability Evaluation of User Interfaces Generated with a Model-Driven Architecture Tool,” in Law, E., Hvannberg, E., and Cockton, G. (eds.), "Maturing Usability: Quality in Software, Interaction and Value", Chapter 1, HCI Series, Vol. 10, London: Springer, pp. 3–32, 2008.

[2] L. Arhippainen, T. Rantakokko, and R. Tähti, M., “Navigation with an Adaptive Mobile Map-Application: User Experiences of Gesture- and Context-Sensitiveness,” in Proc. of 2nd Int. Symposium on Ubiquitous Computing Systems UCS’2005 (Tokyo, Nov. 8-9, 2004), Lecture Notes in Computer Science, vol. 3598, Berlin: Springer, pp. 62–73, 2004.

[3] M. Beaudouin-Lafon, “Designing interaction, not interfaces,” in Proc. of the ACM Working Conf. on Advanced Visual Interfaces AVI’2004 (Gallipoli, May 25-28, 2004), New York: ACM Press, pp. 15–22, 2004.

[4] V. Belloti and K. Edwards, “Intelligibility and Accountability: Human-Considerations in Context-Aware Systems,” Human-Computer Interaction, vol. 16, no. 2, pp. 193–212, 2001.

[5] S. Bouzit, G. Calvary, D. Chêne, and J. Vanderdonckt, “Step-by-Step and Shortcut Menus: A Comparison of two Adaptive Menus for Smartphones,” in Proc. of 30th British Human Computer Interaction Conference BHCI’2016 (Bournemouth, July 2016), Electronic Workshops in Computing, BISL, 2016, http://ewic.bcs.org/content/ConWebDoc/56904.

[6] D. Browne, P. Totterdell, and M. Norman, “Adaptive user interfaces,” London: Academic Press Ltd., 1990.

[7] A. Bunt, C. Conati, and J. McGrenere, “A Mixed-Initiative Approach to Interface Personalization,” AI Magazine, vol. 30, no. 4, pp. 58–64, 2009.

[8] Y. Brun, R. Desmarais, K. Geihs, M. Litoiu, A. Lopes, M. Shaw, and M. Smit, “A Design Space for Self-Adaptive Systems,” in Self-Adaptive Systems, R. de Lemos et al. (Eds.), Lecture Notes in Computing Systems, vol. 7475, Berlin: Springer, pp. 33–50, 2013.

[9] G. Calvary, J. Coutaz, D. Thevenin, Q. Limbourg, L. Bouillon, and J. Vanderdonckt, “A Unifying Reference Framework for Multi-Target User Interfaces”, Int. with Comp., vol. 15, no. 3, pp. 289–308, Nov. 2003.

[10] S.K. Card, T. Moran, and A. Newell, “The keystroke-level model for user performance time with interactive systems,” Communications of the ACM, vol. 23, no. 7, pp. 398–400, 1980.

[11] A. Cockburn, C. Gutwin, and S. Greenbrerg, “A Predictive Model of Menu Performance,” in Proc. of ACM Int. Conf. on Human Aspects in Computing Systems CHI’2007 (San José, April 30-May 3, 2007), New York: ACM Press, pp. 627–636, 2007.

[12] J. Coutaz, L. Nigay, D. Salber, A. Blandford, J. May, and R., Young, “Four easy pieces for assessing the usability of multimodal interaction: the CARE properties,” in Proc. of IFIP TC 13 Int. Conf. on Human-Computer Interaction INTERACT’95 (Lillehammer, June 27-29, 1995), London: Chapman & Hall, 1995, pp. 115–120.

[13] C.-E. Dessart, V. Genaro Motti, and J. Vanderdonckt, “Showing user interface adaptivity by animated transitions,” in Proc. of ACM Conf. on Engineering Interactive Computing Systems EICS’2011 (Pisa, June 13-16, 2011). New York: ACM Press, pp. 95–104, 2011.

[14] H. Dieterich, U. Malinowski, T. Kuhme, and M. Schneider-Hufschmidt, “State of the art in adaptive user interfaces,” in Adaptive User Interfaces Principles and Practice, Schneider-Hufschmidt, M., Kuhme, T., Malinowski, U. (Eds.), Amsterdam: Elsevier Sci. Pub., pp. 13–48, 1994.

[15] T. Djajadiningrat, K. Overbeeke, and S. Wensveen, “But how, Donald, tell us how?: on the creation of meaning in interaction design through feedforward and inherent feedback,” in Proc. of ACM Conf. on Designing Interactive Systems DIS’2002 (London, 2002), New York: ACM Press, pp. 285–291, 2002.

[16] M.R. Endsley, “Toward a Theory of Situation Awareness in Dynamic Systems,” Human Factors Journal, vol. 37, no. 1, pp. 32–64, 1995.

[17] L. Findlater and J. Mc Grenere, “A comparison of static, adaptive, and adaptable menus,” in Proc. of ACM Int. Conf. on Human factors in computing systems CHI’2004 (Vienna, April 24-29, 2004), New York: ACM Press, pp. 89–96, 2004.

[18] K.Z. Gajos, M. Czerwinski, D.S. Tan, and D.S. Weld, “Exploring the Design Space for Adaptive Graphical User Interfaces,” in Proc. of the ACM Working Conference on Advanced Visual Interfaces AVI’2006 (Venice, May 23-26, 2006), New York: ACM Press, pp. 201–208, 2006.

[19] K.Z. Gajos, K., Everitt, D.S. Tan, M. Czerwinski, and D.S. Weld, “Predictability and accuracy in adaptive interfaces,” in Proc. of ACM Int. Conf. on Human Aspects in Computing Systems CHI’2008 (Florence, April 5-10, 2008), New York: ACM Press, pp. 1271–1274, 2008.

[20] V. Genaro Motti and J. Vanderdonckt, “A Computational Framework for Context-aware Adaptation of User Interfaces,” in Proc. of 7th Int. Conf. on Research Challenges in Information Science RCIS’2013 (Paris, 29-31 May 2013), Piscataway: IEEE Press, pp. 1–12, 2013.

[21] Ch. Gram and G. Cockton (Eds), “Design Principles for Interactive Software,” Chapman & Hall: London, 1996.

[22] R. Hartson, “Cognitive, physical, sensory, and functional affordances in interaction design,” Behaviour & Information Technology, vol. 22, no. 5, pp. 315–338, 2003.

[23] A. Jameson, “Understanding and Dealing with Usability Side Effects of Intelligent Processing,” AI Magazine, vol. 30, no. 4, 2009, pp. 23-40.

[24] T.G. Lane, “Studying Software Architecture Through Design Spaces and Rules,” Technical Report CMU/SEI Report Number: CMU/SEI-90-TR-018, Carnegie Mellon University, Software Engineering Institute, 1990.

[25] T. Lavie and J. Meyer, “Benefits and costs of adaptive user interfaces,” International Journal of Human-Computer Studies, vol. 68, pp. 508–524.

[26] D.S. Lee and W.C. Yoon, “Quantitative results assessing design issues of selection-supportive menus,” International Journal of Industrial Ergonomics, vol. 33, no. 1, pp. 41–52, 2004.

[27] V. López-Jaquero, J. Vanderdonckt, F. Montero, and P. González, “Towards an Extended Model of User Interface Adaptation: the ISATINE framework,” in Proc. of IFIP WG2.7/13.4 10th Conf. on Engineering Human Computer Interaction EIS’2007 (Salamanca, 22-24 March 2007), J. Gulliksen, M.B. Harning, Ph. Palanque (Eds.), Lecture Notes in Comp. Science, Vol. 4940, Berlin: Springer, pp. 374–392, 2008.

[28] P.K. McKinley, S.M. Sadjadi, E.P. Kasten, and B.H. Cheng,“Composing Adaptive Software,” IEEE Computer, vol. 37, no. 7, pp. 56–64, 2004.

[29] N. Mezhoudi, I. Khaddam, and J. Vanderdonckt, “WiSel: a mixed initiative approach for widget selection,” in Proc. of the 2015 ACM Conf. on research in adaptive and convergent systems RACS’2015 (Prague, October 9-12, 2015), New York: ACM Press, pp. 349–356, 2015.

[30] N. Mezhoudi and J. Vanderdonckt, “A User Feedback Ontology for Context-aware Interaction,” Proc. of the 2nd IEEE World Symposium on Web Applications and Networking WSWAN’2015 (Sousse, March 2015, 21-23), Piscataway: IEEE Computer Society Press, 2015, pp. 1–7.

[31] M. Nebeling, M. Speicher, and M.C. Norrie, “CrowdAdapt: enabling crowdsourced web page adaptation for individual viewing conditions and preferences,” in Proc. of ACM Symp. on Engineering Interactive Computing Systems EICS’2013, New York: ACM Press, pp. 23–32.

[32] D.A. Norman, The design of everyday things. Doubleday, NY, 1988.

[33] A. Paramythis, “Towards Self-Regulating Adaptive Systems,” in Proc. of the Annual Workshop of the SIG Adaptivity and User Modeling in Interactive Systems of the German Informatics Society ABIS’2004, Berlin: Springer, pp. 57–63, 2007.

[34] A. Paramythis, S. Weibelzahl, and J. Masthoff, “Layered evaluation of interactive adaptive systems: framework and formative methods”, User Modeling and User-Adapted Interaction, vol. 20, pp. 383–453, 2010.

[35] D.L Scapin and J.M.C Bastien, “Ergonomic criteria for evaluating the ergonomic quality of interactive systems,” Behaviour & Information Technology, vol. 16, no. 4/5, pp. 220–231, 1997.

[36] M. Schlee and J. Vanderdonckt, “Generative Programming of Graphical User Interfaces,” in Proc. of 7th Int. Working Conf. on Advanced Visual Interfaces AVI’2004 (Gallipoli, May 25-28, 2004), New York: ACM Press, pp. 403–406, 2004.

[37] J.-S. Sottet, V. Ganneau, G. Calvary, J. Coutaz, A. Demeure, J.-M. Favre, and R. Demumieux, “Model-Driven Adaptation for Plastic User Interfaces,” in Proc. of IFIP TC13 Int. Conf. on Human-Computer Interaction INTERACT’2007, Berlin: Springer, pp. 397–410, 2007.

[38] T. Tsandilas and M.C. Schraefel, “An empirical assessment of adaptation techniques,” in Proc. of ACM Int. Conf. on Human Aspects in Computing Systems CHI’2005 (Portland, April 2-7, 2005), Ext. Abstracts, New York: ACM Press, pp. 2009–2012, 2005.

[39] J. Vermeulen, “Improving intelligibility and control in UbiComp,” in Proc. of the 12th ACM Int. Conf. on Ubiquitous Computing UbiComp’2010, New York: ACM Press, pp. 485-488, 2010.

[40] D. Weld, C. Anderson, P. Domingos, O. Etzioni, T. Lau, K. Gajos, and S. Wolfman, “Automatically personalizing user interfaces,” in Proc. of Proceedings of the 18th international joint conference on Artificial intelligence IJCAI’2003 (Acapulco, August 9-15, 2003), San Francisco: Morgan Kaufmann, pp. 1613–1619, 2003.

[41] J.O. Wobbrock, S.K. Kane, K.Z. Gajos, S. Harada, and J. Froehlich, “Ability-based design: Concept, principles and examples,” ACM Transactions on Accessible Computing, vol. 3, no. 3, pp. 1–27, April 2011.