Contextual Adaptation of User Interfaces

Gaëlle Calvary[0000-0003-1680-4575] and
Alaa Eddine Anis Sahraoui[0009-0001-5402-9605]

Abstract Since the inception of interactive systems, it has been clearly stated that user interfaces need to be adapted to their users in context, adaptation being either engineered at design-time or computed at run-time. Experience shows that adapting user interfaces dynamically is not risk-free: while there may be expected advantages, they may be outweighed or even prevented by several unfavorable consequences, the primary one coming from the adaptation process itself. There may be a cognitive disturbance that occurs in the user's mind when the interface changes compared to its original state. This chapter first outlines the adaptation life cycle's seven stages before discussing some of the challenges in adapting user interfaces generally, and adaptive user interfaces specifically. To review the state-of-the-art in UI adaptation, this chapter breaks down the definition of an adaptation operation using the Event-Condition-Action paradigm, qualified by the properties it is expected to satisfy. Thereby, the four parts are the Event that initiates the adaptation, the Condition the adaptation is applicable under, the Adaptation to be applied, and the overall worth of the operation in terms of its quality properties. For each part, we examine and discuss a series of representative works to improve our general understanding of how adaptation works and should be implemented. In the end, it explores a couple of adaptation-related topics that indicate a promising future.
A playlist of videos showing adaptive user interfaces is also available to review research and innovation in user interfaces adaptation.

Gaëlle Calvary
Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG, Grenoble, France,
* Institute of Engineering Univ. Grenoble Alpes
e-mail: Gaelle.Calvary@grenoble-inp.fr

Alaa Eddine Anis Sahraoui
Université catholique de Louvain, LouRIM, Belgium
e-mail: alaa.sahraoui@uclouvain.be

1 Introduction

Adapting User Interfaces (UI) to their contexts of use aims at meeting both evolving requirements, such as the needs, desires, and preferences of individual users or user groups, and evolving resources, such as the available interaction devices in the surroundings, that all may be dependent on the current situation. There are two categories of adaptation, whether the end user or the application is responsible for the modification [42]. *Adaptability* refers to the end user's ability to adjust the UI, while *adaptivity* or *self-adaptation* denotes the application's ability to perform UI adjustments [18]. *Adaptive User Interfaces* (AUIs) refer to user interfaces benefiting from adaptivity. *Personalisation* or *customisation*, a subset of adaptivity, focuses on adapting the UI contents, presentation, and behaviour based solely on end-user data, such as personal traits [42]. When data comes from external sources, like other user groups, it results in *recommendations* instead. *Mixed-initiative adaptation* occurs when both the end user and the application collaborate to make adaptations [50].

The ultimate goal of any UI adaptation is to make the UI better from the end user perspective to satisfy him/her as much as possible, by optimising factors such as efficiency (e.g., reducing the task execution time and error rates or improving the learning curve), effectiveness (e.g., by ensuring full task completion), and subjective user satisfaction. This goal is commendable, but it carries some risk because adaptation is multifactorial and not all its impacts have been thoroughly researched. Benefiting from the adaptation resulting from the improvement of certain factors comes at the price of suffering the resulting disadvantages (e.g., user disruption [51]), which may have an impact on factors other than those initially considered.

The challenge lies in suggesting and/or executing the appropriate adaptation at the right time and place to provide value to the end user [4]. Otherwise, adaptation may encounter limitations that hinder expected benefits [59], including the risk of misfit, user cognitive disruption, lack of prediction, lack of explanation, lack of user involvement, and privacy risks [33]. Therefore, adapting UIs means responding adequately to the following questions of the Quintilian hexameter [72]:

- *What to adapt?* Specifies the type of resource or aspect that is subject to adaptation: contents (e.g., text, images, videos), presentation (e.g., format of UI elements, position, size, arrangement), or behaviour (e.g., navigational flow, activation, and deactivation of features, promotion or demotion of widgets [17] - see Figure 17). For example, Figure 1 shows an adaptation of a weather forecast application ① with more data depending on the user's location ②. Figure 2 shows an adaptation of the presentation where the background colour is adapted from daytime ① to nighttime ②. Figure 3 shows three examples of behaviour adaptation depending on desktop ①, tablet ②, and smartphone ③.
- *Why to adapt?* Specifies the main goals that the adaptation should reach and satisfy, preferably expressed as software qualities (e.g., support the transition from a novice user to an expert user, maximise the learnability) and explains how the adaptation can contribute to achieving these goals (e.g., this adaptation is expected to reduce the cognitive load).

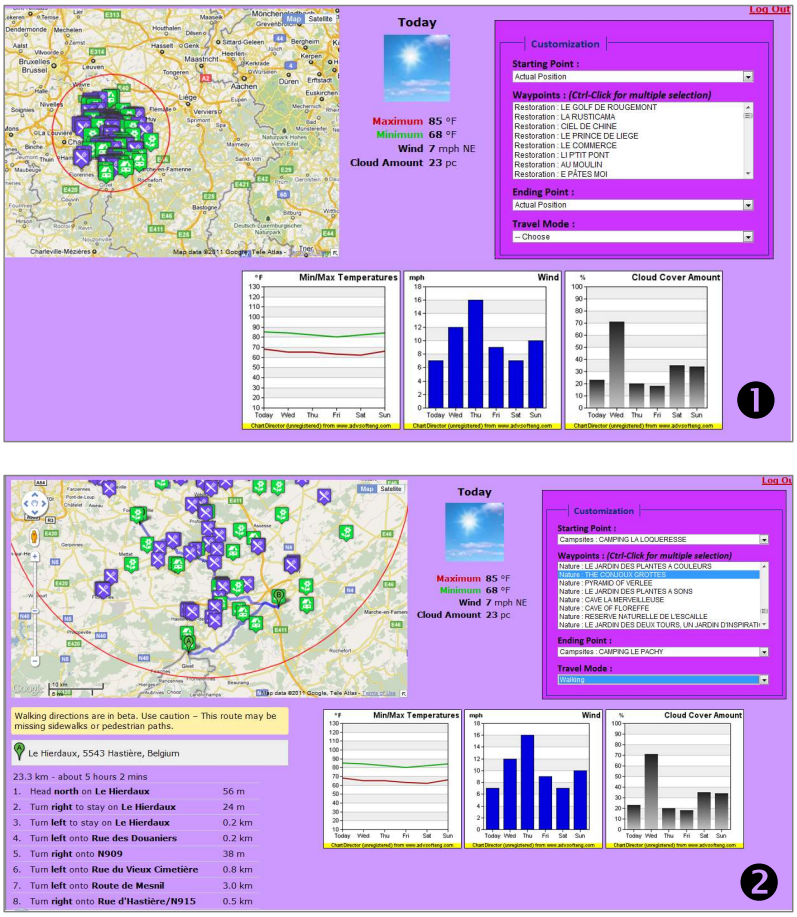


Fig. 1: Example of contents adaptation: before ① and after ② (Application Walk-Aware, by courtesy of Luc Ponsard).

- *How to adapt?* Specifies according to which method, technique, algorithm, or strategy the adaptation will be performed (e.g., a technique of changing the video quality, a decision tree to select the most preferred widget or menu type). This chapter will define an adaptation operation to specify how an adaptation action could be operationalised based on its event and condition for a certain value.
- *With regard to what?* Specifies contextual information related to the user (e.g., preferences, skills, level of experience) and interactive tasks (e.g., what type of task is subject to adaptation), to the computing platform (e.g., device characteristics, platform features, screen resolution, interaction capabilities), and to the environment (e.g., location, level of light, noise, stress) with regard to which the adaptation is justified and performed. For example, adapt the UI to colour-blind users, to a set of devices, or to a noisy environment.

- *Who is controlling the adaptation?* Specifies the actor that initiates, triggers, or is in charge of each stage of the adaptation (e.g., the end user, the application, the UI, a third party such as a proxy, or any combination). This chapter addresses this question by referring to the seven stages of the life cycle.
- *When to adapt?* Specifies the temporal state in which the adaptation is performed (e.g., at design time, run time, or compilation time). For example, the adaptation moment is at run-time whether it is for adaptability or adaptivity.
- *Where to adapt?* Specifies the physical location where the adaptation is computed (e.g., based on the software architecture at the client, at the proxy, or on the server). For example, an adaptation can be performed on the server side when its logic is computationally demanding and cannot be operated at the UI level.

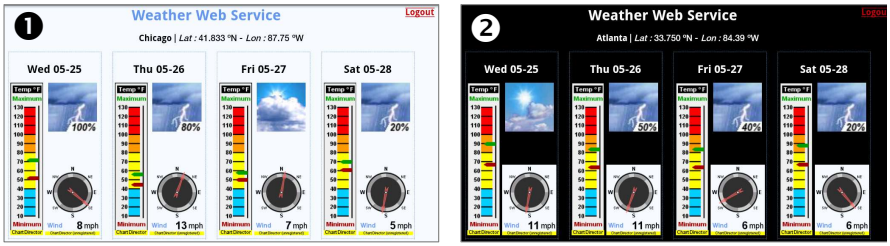


Fig. 2: Example of Presentation adaptation: daylight ① and nightlight ② (Application WalkAware, by courtesy of Luc Ponsard).

2 The User Interface Adaptation Life Cycle

The UI adaptation life cycle can be decomposed into seven stages [1, 64] (Figure 4):

1. *Specification of the adaptation goals*: this first stage states the goals that UI adaptation should pursue, maintain, and update by any entity involved, such as the user (U), the application or system (S), a third-party (T), or any combination of them. Various contextual aspects are considered, such as the user profile, the task at hand, the computational platform (both hardware and software), and the entire physical and organisational environment in which the task is performed. The goals fall into three categories: self-expressed, application-expressed (locally or remotely), or according to their location. When the application is tolerant of faults depending on conditions, a machine-expressed goal is specified.
2. *Specification of initiative for adaptation*: this second stage specifies which entity is responsible for initiating the adaptation: the user (explicit initiative), the application (implicit initiative that detects a change in the context of use requiring adaptation), or both jointly (as a decision taken by the entities in control). When the machine initiates an adaptation that the user subsequently cancels, a mixed initiative occurs [50]. This stage is further refined into an adaptation request,

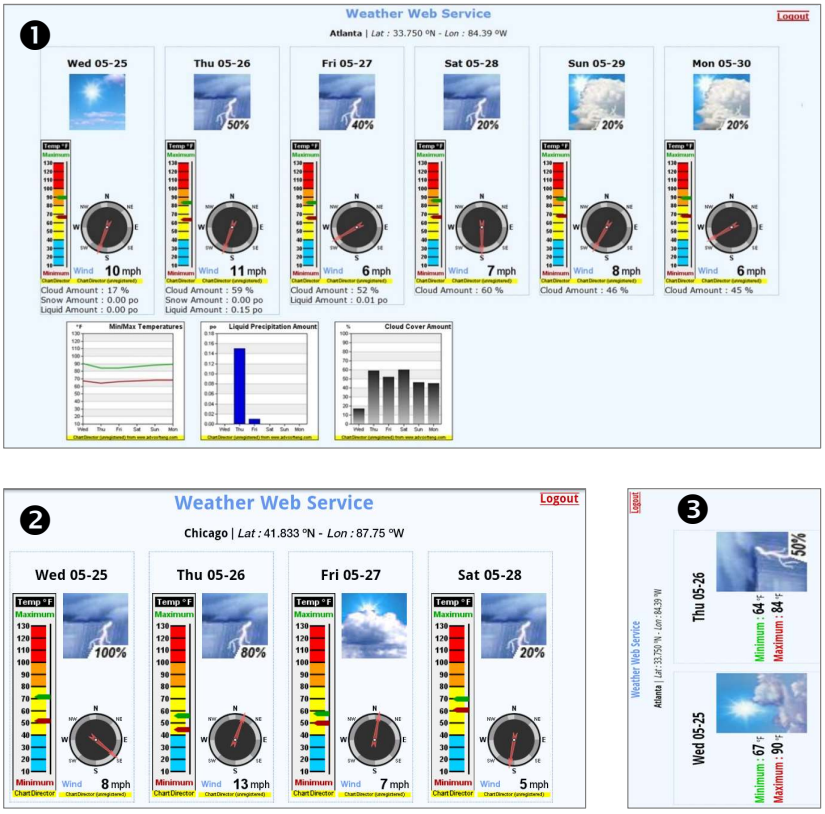


Fig. 3: Example of behaviour adaptation: desktop ①, tablet ②, and smartphone ③ (Application WalkAware, by courtesy of Luc Ponsard).

a detection of an adaptation need, and a notification for an adaptation request, depending on their location.

3. *Specification of adaptation*: this third stage specifies which entity is responsible for deciding that an adaptation should occur. After the adaptation is initiated, a set of adaptation proposals is created that contains zero, one, or many proposals. These proposals can be further elaborated by the user, the application, or a third party, such as a broker. If the application is responsible for creating adaptation proposals, then adaptation rules generate proposals and even infer new rules. Once the set of proposals is obtained, a decision must be made to accept (e.g., which adaptation proposal or which ones are the most appropriate), to reject (e.g., if no proposal corresponds to the goals), to cancel (e.g., no adaptation is finally needed), to defer (e.g., by the next session), or to propose a new set.
4. *Application of adaptation*: this fourth stage specifies which entity is responsible for applying the adaptation decided in the previous stage, if any. Since the adaptation will be applied to the UI, it should incorporate a mechanism that

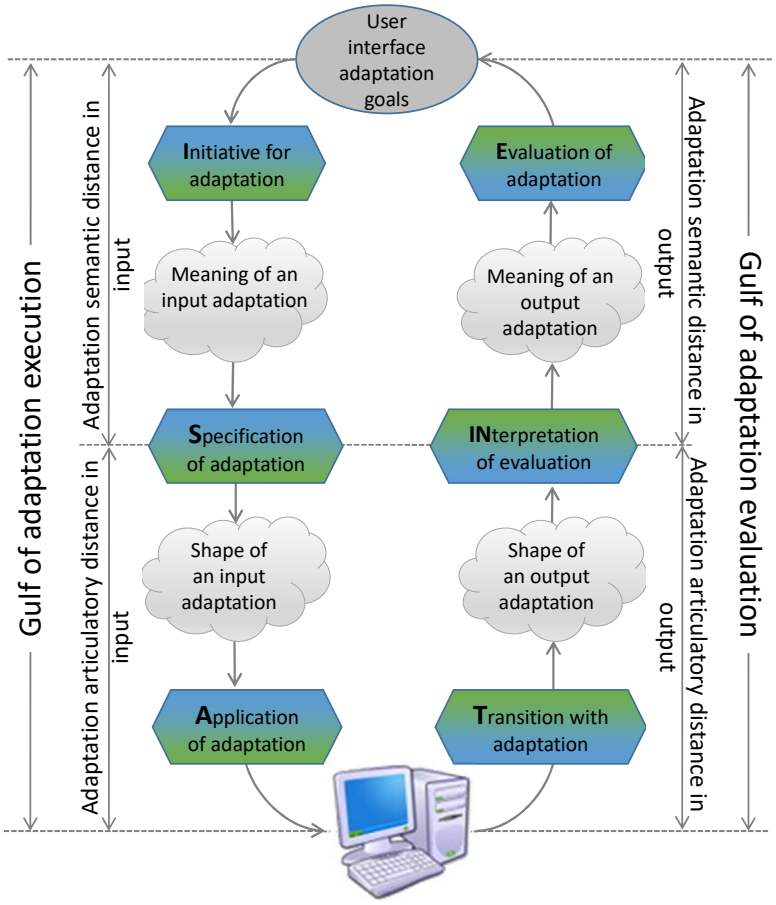


Fig. 4: The seven stages of user interface adaptation (Illustration from ISATINE [64]), by courtesy of Jean Vanderdonckt).

supports adaptation (e.g., via an API for parameterisation [24]). The user adapts the UI (e.g., through UI options, personalisation) or the application does it on behalf of the user. For example, transformations apply various presentation adaptations of a GUI to multiple platforms with various resolutions [40, 8].

5. *Transition with adaptation*: this fifth stage specifies which entity is responsible for ensuring a smooth transition between the UIs before and after adaptation. For example, an application uses some visualisation techniques to present the intermediary steps executed (e.g., by progressive rendering [83] or by animated transition, by morphing [30]).
6. *Interpretation of evaluation*: this sixth stage specifies which entity is responsible for providing information meaningful for understanding the adaptation. When the application performs some adaptation without any explanation, the user does

not necessarily understand why this type of adaptation has been performed. Conversely, when the user has performed some adaptation, the user should tell the application how to interpret this adaptation. For example, a Machine Learning (ML) algorithm first proposes some adaptation to be applied [34]; if this adaptation does not correspond to the goals, the user produces an alternate adaptation and informs the application how to incorporate this new scheme for future use. The machine updates the knowledge base by interpreting this explanation. User-control of the adaptation is key and can be supported by unsupervised learning [35].

7. *Evaluation of adaptation*: this seventh stage specifies which entity is responsible for evaluating the quality of the adaptation performed to check whether the initially-specified goals are met. For example, if the application maintains some internal plan of goals, it should update this plan according to the adaptations applied so far. If the goals are in the users' mind, they could also be evaluated with respect to what has been performed in the previous stages. In this case, the explanation of the adaptation contributes to updating the goals' status too. For example, Figure 5 shows how a previously-applied adaptation can be evaluated: the adaptation operation applied is displayed and subject to evaluation by the system itself (based on internal analysis) or by the end-user, which can be qualitative (e.g., by a rating bar) or quantitative (e.g., by a rating scale).

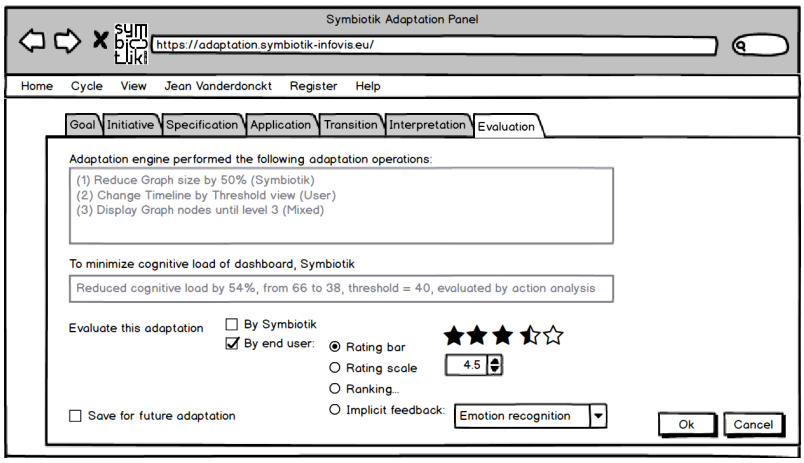


Fig. 5: Example of adaptation evaluation in Symbiotik: by the user and/or the system (Illustration owned by the second author).

3 Targeted Literature Review on User Interface Adaptation

Despite the significant progress in AUIs, several challenges persist, which complicate the broader adoption of these systems and their development. Instead of perform-

ing a systematic literature review, this section prefers a targeted literature review that summarises the literature on UI adaptation, considering the most relevant and representative references in the domain.

3.1 Existing and Future Challenges in User Interface Adaptation

This section therefore discusses a limited set of challenges that are still active and open today for UI adaptation.

C_1. **Complexity in user modelling.** Building precise and thorough user models that can anticipate and adjust to each user's demands is still challenging. These models must account for the diversity of human behaviour and preferences, which are shaped by a wide range of elements such as the user's present emotional state, personal experience, and cultural background. Because of this complexity, models frequently oversimplify the user or become too complicated to be used in real-time adaptivity [75]. Several factors influence the complexity in user modelling. Human behaviour is intrinsically highly variable and influenced by numerous external and internal factors. Capturing this variability in a user model requires sophisticated algorithms and extensive datasets [5]. Various user modelling approaches, such as GOMS models, cognitive architectures, grammar-based models, and application-specific models, have varied levels of complexity and are appropriate for different types of applications [23]. The complexity increases while selecting and applying the appropriate strategy in the appropriate situation. End-users interact with AUIs in various contexts of use [19] that change over time, such as different tasks, devices, platforms, and environments [20]. Adapting to these dynamic contexts of use adds another layer of complexity [26]. Cultural background, personal experiences, and personal traits [42] significantly influence user preferences and interactions. Models must be sensitive to these differences to provide meaningful adaptations [23]. User models used for adaptation may initially be correct or not and become outdated over time due to changes in external factors. Identifying, correcting, and updating these models while the context is evolving is even more complex. Mechanisms need to be in place to detect and address inaccuracies in the user models [5]. Environmental factors in the context of use are often underestimated if not overlooked. The environment in which the user operates [32], including factors like traffic, noise, light, pressure, and the presence of other users, especially in collaborative configurations, significantly impacts user behaviour and the effectiveness of AUIs.
Due to all these reasons, obtaining an accurate AUI is no longer a matter of predefined adaptation rules, heuristics, and models, but a matter of taking the evolution into play.

C_2. **Balancing adaptation with user control.** While AUIs aim to provide a highly adapted user experience [1], there is a fine line between helpful adaptation

and perceived intrusion. Users may feel a loss of control over the UI when changes are too frequent, too disruptive [95], unexpected [59], or disturbing [51]. Ensuring that adaptations enhance user experience without undermining user autonomy is a crucial design consideration that remains largely unresolved. A totally automatic adaptation is certainly disrupting; a totally manual adaptation is probably even more disrupting as the adaptability cost is too expensive for the end-user. The aim of possessing a good AUI is to seem finally as if adaptation lies in mixed-initiative [50] control.

For example, MyUI [79] tested the effectiveness and acceptability of adaptation patterns in different cost-benefit situations and for different users. The patterns turn out to increase the transparency and controllability [42] of adaptations as soon as they are under user control. Preference and acceptance of the patterns depend on the cost-benefit condition. In the same vein, Scaler [35] supports user control by unsupervised learning by enabling end-users to personalise the form widgets based on their preferences or based on the system.

C_3. **Integration with existing systems.** Integrating advanced adaptive mechanisms into existing technological ecosystems poses technical and operational challenges. Compatibility with legacy systems, scalability, and maintaining consistent performance across different platforms are significant hurdles for developers of AUIs [1]. Software engineering techniques, such as reverse engineering of UIs [13] are helpful, but will always require some manual code tweaking.

C_4. **Privacy and security.** To provide personalised [75] and context-aware services [27], AUIs collect a variety of user data, such as demographic data, location history, preferences, and even psychological states. Deep data collecting, however, presents privacy issues [33, 49] because private data may be exposed and misused. For example, an AUI can improperly determine a user's home location based on repeated visits [10]. Additionally, AUIs influence user behaviour through subtly nudging them, such as encouraging in-app purchases during decision-weariness-prone moments. This is especially true when it comes to games that use subtle cues to encourage in-app purchases, which is particularly remarkable in some games [93]. This could force end-users to buy at times when they are most likely to be vulnerable, including when they are tired from making decisions. This is an illustration of indirect manipulation of autonomy, in which the interface is made to sway the choices made by the user according to information gathered about their actions and emotional state [49]. Ethical design principles should put user liberty and privacy first to reduce these hazards and protect personal data [106].

Differential privacy is one promising strategy that preserves individual privacy while enabling user data analysis. Differential privacy lowers the likelihood of privacy breaches by adding "noise" to the data, which makes it harder to identify particular users [33]. Blockchain technology can also be used to improve data usage accountability and transparency. Users can have more insight into how and by whom their data is being used by keeping track of data transactions on a decentralised ledger. This helps foster trust and prevent data misuse [106].

C_5. **Handling wrong, outdated, and inadequate data in adaptivity.** AUIs heavily rely on user models that are intended to anticipate and react to unique demands, preferences [36], and behaviours [24]. These models frequently encounter difficulties because of errors resulting from a variety of sources [1]. User preferences and behaviours may not be fully captured, which might result in incorrect assumptions when building user models. For example, Netflix's recommendation engine begins with user inputs, such as preferred genres, and improves over time by analysing movie viewing patterns to provide more precise recommendations [45]. To avoid these models becoming out of date, they should be regularly updated with new data to preserve the relevance and responsiveness of AUIs. For this purpose, methods like data mining and categorisation ensure that the AUI adapts to suit the user's evolving needs [54].
For example, in a smart home setting, an AUI may adjust lighting settings in response to user behaviour. If the user's schedule changes, the model may become unaligned. The system can determine the requirement for model upgrades by identifying more frequent manual illumination adjustments [66]. Furthermore, due to privacy restrictions, incomplete data collection, or difficulties in predicting specific behaviours, the data available for creating user models may occasionally be insufficient, thus resulting in user dissatisfaction, decreased efficiency, and possible disengagement [1].

3.2 Recent Advances in User Interface Adaptation

In response to the aforementioned challenges, recent advances in technology and methodology have been made to enhance the functionality and applicability of AUIs:

A_1. **Enhanced machine learning techniques.** Recent developments in ML, Deep Learning (DL), and neural networks have provided new ways to process and analyse large datasets more effectively for AUIs. These advancements enable more nuanced understanding and prediction of user behaviour, improving the accuracy of user models. For example, techniques like reinforcement learning [95, 44] dynamically adapt AUIs based on user interactions, learning from each user's preferences to optimise the UI configuration over time. This way of supporting adaptation induces two sets of operations between the end-user and the AUI [16]: a Perception-Decision-Action (PDA) cycle for the user followed by a Learning-Prediction-Action (LPA) cycle for the AUI.

A_2. **Multimodal interaction.** Advancements in multimodal interfaces, which combine inputs from various sources, such as voice, touch, and even gaze, offer new ways to understand user intentions and context more holistically [12]. These AUIs can adapt more effectively to the user's current state and preferences by integrating different types of data, leading to a more intuitive and seamless user experience. In particular, the modality can be adapted to environmental conditions [56] while allowing the end user to choose among the available modalities which is the most adapted. For example, polymodal menus [15] enable end-

users to select menu items graphically, vocally, or gesturally depending on their preferences and their context of use.

A_3. **Context-aware adaptation.** Progress in sensor technology and context-aware computing allows AUIs to adapt not only to the user, but also to the entire context of use at once [72, 103, 104]. AUIs can now consider factors such as location, time of day, and even the presence of other people to tailor user experiences. For example, a UI might simplify its options and enlarge buttons when it detects that the user is in a moving vehicle.

A_4. **User-centred design approaches.** There is a growing emphasis on involving users directly in the adaptation process. User-centred design approaches [101] that allow users to set preferences on how and when the UI should adapt are being explored to balance automation with user control. This approach not only addresses the challenge of maintaining user agency but also helps in fine-tuning the system according to individual preferences. Although AUIs have made remarkable progress, they continue to evolve in response to technological advances and user feedback. Overcoming existing challenges and effectively leveraging recent advances will be key to developing more intuitive, efficient, and user-friendly AUIs in the future. These advances promise not only to enhance user interaction but also to pave the way for UIs that are truly responsive to the complex and dynamic nature of human needs and behaviours.

3.3 Computational Approaches to Adaptive User Interfaces

Using modern computational techniques, AUIs dynamically adjust user experiences in real-time, adapting to users' changing choices, actions, and environmental situations. The need for these UIs is growing as systems become more complicated and as users demand more customized experiences. Incorporating developed computational methods [55] into the comprehension, creation, and modification of UIs not only improves the adaptability and responsiveness of these UIs, but also addresses important issues in human-computer interaction (HCI), including context awareness, usability, and accessibility. In order to enable AUIs, a number of generations of computational approaches [55] from rule-based systems to machine learning algorithms and model-based techniques [52] have been studied over time.

3.3.1 Model-Based Development of Adaptive User Interfaces

Model-based development (MBD) [34, 29, 52] provides a structured approach to designing AUIs that can dynamically adjust to the user's context of use. By defining high-level abstractions that represent the core functionalities and user interactions, MBD allows for the creation of flexible and adaptable systems. This section explores how MBD facilitates the development of AUIs and examines case studies to illustrate its practical applications. MBD focuses on creating a set of models that define the UI

at various levels of abstraction, from user interactions down to the specific elements of the UI design. These models include:

1. *Domain models*: Define the data and logic specific to the application domain, ensuring that the UI aligns with the business requirements [91] and needs [45].
2. *User models*: Capture information about the users, their preferences, and behaviours to tailor the UI accordingly. These models make it possible to modify the UI in real-time, improving usability and satisfaction by customising the experience to each user's requirements. For example, the "Mining Minds" platform adapts its UI based on user context, such as increasing the font size for visually impaired users or switching to graphical modes [52].
3. *Interaction models*: Specify how users will interact with the system, detailing the flow and dynamic aspects of the UI. For example, interaction models can be constantly modified according to user circumstances and preferences using a Rule-Based User Interface (RBUI) method [3]. RBUI, which uses a rule-based framework, allows real-time UI adaptation by considering many variables, including user experience, device capabilities, and ambient conditions. This method closely adheres to MBD principles while also improving the user experience and facilitating a more customised interaction flow.
4. *Presentation models*: Describe the UI rendering (e.g., the graphical components in the case of graphical UIs) while making sure that they are consistent and easy to use on various platforms [8] and UI types [67]. These models are made to adapt dynamically to changes in context, such as different screen sizes and resolutions, while also retaining a consistent look and feel when users switch between different devices, like desktops, tablets, and smartphones. For example, *graceful degradation* exploits a presentation model to adapt an existing UI for a certain platform to a new UI for a smaller platform (Figure 6) [37]. The opposite operation is called *progressive enhancement*. This flexibility offers a productive user experience, especially when the AUI instantly adapts to changing user preferences or environmental circumstances. For example, the UI of a travel planning application adapts based on the user's context [1]: a business traveler in a busy airport using a smartphone might see a simplified layout with larger buttons to facilitate quick interactions, while a tourist using a laptop in a quiet hotel room might experience a more detailed UI, taking advantage of the larger screen and calmer environment.

By abstracting these elements, MBD supports the generation of UIs that can adapt to different user requirements and operating environments without significant rework. This approach not only enhances the adaptability of the UI but also promotes reusability and scalability. The MBD advantages include:

- *Consistency across platforms*: MBD allows developers to maintain a consistent user experience across various platforms by adapting the base model to fit different device specifications [40].
- *Efficiency in development*: by abstracting the UI into models, changes can be implemented more quickly and propagated across platforms automatically [71].

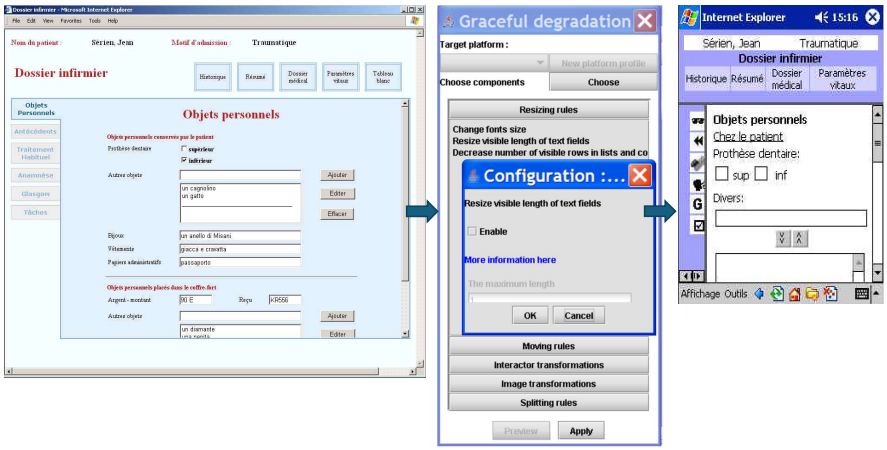


Fig. 6: Graceful degradation of a desktop UI to a mobile [6] (Illustration from ARTHUR project, by courtesy of Jean Vanderdonckt).

.

- *Improved adaptability*: with models that dynamically adjust based on user data and context, the UI can respond in real-time to changes in user behaviour or environmental conditions, such as ubiquitous environments [76].
- *Model reusability*: once some UI or user aspects have been captured in the models, any model excerpt can be reused in other contexts of use that share some common characteristics [28]. For example, the nomadic gestures [100] can be reused from one context to another.

The main MBD limitations are:

- *Required abstraction by modelling*: the big win with MBD is that, when the model(s) change, the UI should change accordingly. This approach always requires abstracting various UI aspects and user parameters into a set of models, an approach that is not straightforward for any designer or developer [80, 81].
- *Dependence on rendering engines*: UI rendering engines can produce final AUIs either by code generation or by model parsing and interpretation. In both cases, the quality of produced AUIs heavily depends on the power of such engines and on adaptation rules or operations [2].
- *Increased complexity*: the MBD process adds another layer of complexity on top of the adaptation process, but this is probably inevitable since adaptation cannot happen without such mechanisms [54].

3.3.2 Model-Driven Engineering of Adaptive User Interfaces

Model-Driven Engineering (MDE) is a computational approach that focuses on creating and exploiting models, which are representations of the knowledge and

know-how in specific areas. In the context of AUIs, MDE allows developers to abstract and automate the generation of UIs from high-level models. These models describe the UI independent of technology, enabling the adaptive system to instantiate UIs tailored to different devices, users, or surroundings.

For example, Yigitbas et al. [104] discuss how MDE can facilitate the development of UIs that automatically adjust to user-specific data and context changes without manual intervention. By employing a model-driven approach, developers can ensure consistency across different platforms while accommodating individual user needs and preferences.

3.3.3 Model-Free Approaches of Adaptive User Interfaces

Model-free approaches do not assume to have any particular underlying model. For example, ACE (Adaptive CSS-oriented Engine) [60] automatically adapts presentation properties of widgets in a web page based on designer's input and interaction history: adapt the font size of title, labels, and contents, changing the margins, increase the surface of push buttons to enable touch-based interaction. Adaptation of presentation properties can also be generated, either completely or partially unsupervised, according to the collective behaviour of a set of web page users [61, 62]. While a representation of the UI elements subject to adaptation exists in both cases, there is no explicit user or complete UI model to support adaptation.

3.3.4 Artificial Intelligence, Machine Learning, and Deep Learning of AUIs

AI and ML are at the forefront of driving adaptive capabilities in UIs. These technologies enable systems to learn from data, identify patterns, and make decisions with minimal human intervention. In AUIs, machine learning algorithms can predict user behavior, adapt interfaces to new contexts, and even anticipate future needs by analyzing historical data.

The application of AI in AUIs often involves natural language processing (NLP), image recognition, and predictive analytics. For example, NLP can interpret and respond to user inputs in natural language, enhancing UI accessibility and ease of use. More recently, Reinforcement Learning (RL) of AUIs for Accessibility Context [105] exploits three Knowledge Layers depending on the target adaptation: the Disability Knowledge Layer learns the behaviour of the structure of the UI depending on the disability profile, the Modality Knowledge Layer learns adaptation facilities based on the couple (UI, profile), and the Platform Knowledge Layer exploits platform knowledge to learn adaptation facilities. RL can also train agents to learn how to adapt UIs in a specific context of use to maximise user engagement (which can be measured by electroencephalography [43]) by using an interaction model in a reward function [44]. MARLUI [58] engaged a user agent that mimics a real user and learns to interact via point-and-click actions with a UI agent that learns UI adaptations, to maximise efficiency by observing the user agent's behaviour.

Context-aware computing is crucial for the development of AUIs that can intelligently [54] adapt to the user environment [32]. This approach utilizes sensors, data from the device, and other sources to detect and interpret the surrounding physical and digital environment. Then it adjusts the UI accordingly, addressing factors such as location, time, ambient conditions, and even social context.

For example, an AUI might enlarge buttons on a mobile device when it detects that the user is in a shaky environment, such as riding in a vehicle. Similarly, it might switch to a dark mode in low-light conditions to reduce eye strain.

Adaptive algorithms are designed to modify and control their behaviour based on feedback or changes in their operating environment. In AUIs, these algorithms adjust the UI in response to user interactions. They manage the layout, content, and functionality of the UI based on real-time data to optimise the user experience.

For example, an adaptive algorithm might alter the UI layout when it detects that the user frequently accesses certain features but ignores others [84]. It could bring those frequently used features to a more prominent position within the UI. For example, AUI layout and widgets constraints feed a fuzzy constraint satisfaction problem to dynamically adapt them in a graphical UI depending on the screen resolution and the window size (Figure 7) [102].

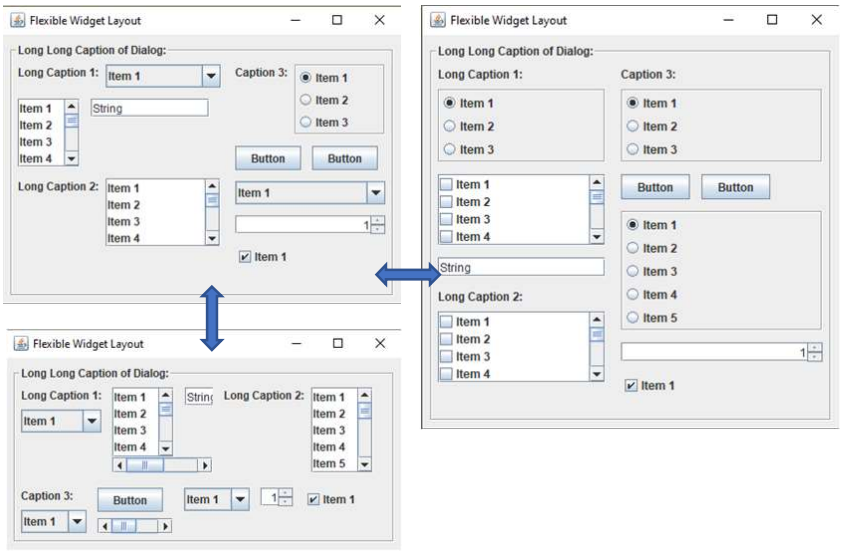


Fig. 7: Adaptive layout and widgets depending on screen resolution [102] (Illustration by courtesy of Takuto Yanagida).

The use of computational design in AUIs represents a significant shift towards more dynamic, responsive, and personalised user experiences. By harnessing model-driven engineering, artificial intelligence, context-aware computing, user modelling, and adaptive algorithms, designers can create UIs that not only meet the diverse needs

of users but also anticipate and adapt to changes in user behaviour and environmental conditions. These technologies foster a deeper integration of human factors into system design, leading to UIs that are not only more functional but also more intuitive and satisfying to use. As these computational techniques continue to evolve, the potential for creating truly intelligent and adaptive UIs will expand, offering new possibilities to improve interaction.

4 A Comprehensive Problem Space for Engineering User Interface Adaptation

In the context of UI adaptation, there are many design spaces [4, 16, 99], taxonomies [31, 57], and frameworks [19, 64, 74, 32], each shedding light on different perspectives based on the authors' concerns. Here, we adopt a comprehensive approach for the specification of UI adaptation, while also considering the quality of adaptation. We propose a dual problem space model called HEMISPHERES [22], designed for the engineering of "plastic" interactive systems [89, 90, 96, 97], i.e. systems whose adaptation preserves quality in use over the evolution of the context of use. This problem space separates concerns into two areas:

- The specification of the adaptation operation (e.g., "If the battery gets low, then migrate the UI to the nearest platform");
- The specification of the adaptation implementation (e.g., including when and how to adapt).

The right hemisphere (Adaptation operation) and left hemisphere (Life cycle) are responsible for these concerns.

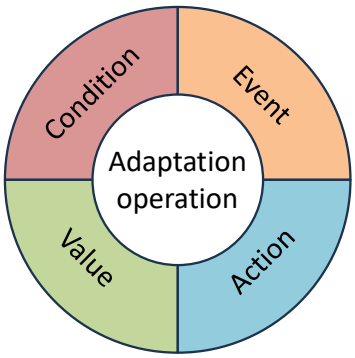


Fig. 8: The four quadrants of an adaptation operation: CONDITION, EVENT, ACTION, and VALUE (Illustration owned by the authors).

4.1 The Right Hemisphere for the Specification of Adaptation Operations

The right hemisphere examines the specification of *adaptation operations*, which under condition associate a response to a change in the context of use, aiming to preserve a certain set of values, such as quality factors, altogether contributing to the system's worth. The overall format of an adaptation operation is: On Event, If Condition, Then Action for Value(s). For example, in the adaptation operation "If the laptop battery gets low, when a smartphone becomes available around, then propose migrating the UI from the laptop to the smartphone":

- The Condition concerns the battery state of the laptop: "If the battery gets low";
- The Event occurs when the context of use changes with the arrival of the smartphone: "when a smartphone becomes available around";
- The Action proposed in response to this contextual change response is a UI migration: "then propose migrating the UI from the laptop to the smartphone");
- The Value is implicit: the interaction continuity should be preserved.

Note that UI migration [46, 39] consists of transferring a particular UI from one platform to another while preserving its interaction state. The key concepts of Condition, Event, Action, and Value (Figure 8) are classical Event-Condition-Action (ECA) rules enhanced with the notion of Value. We will now further examine, define, and develop the four quadrants of Figure 8.

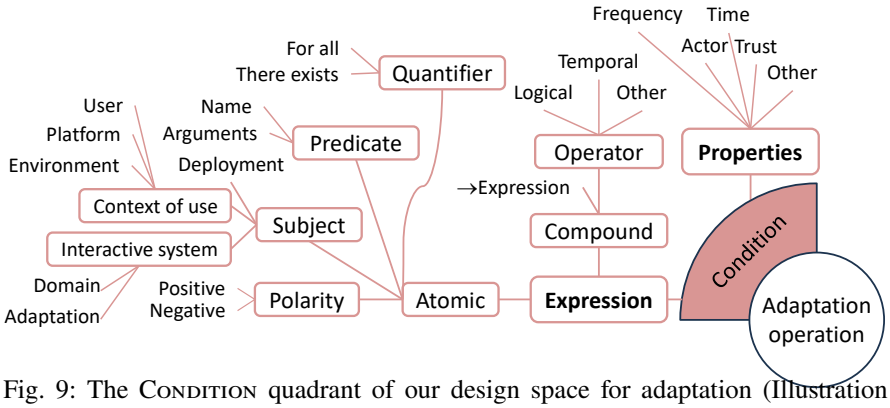


Fig. 9: The Condition quadrant of our design space for adaptation (Illustration owned by the authors).

4.1.1 Condition

The Condition relates to any observable aspect of the interactive ecosystem, which encompasses any variable of the context of use (e.g.,, user and task, platform/device, and environment) [19, 20]. If the condition is not met, the adaptation operation

cannot be triggered. The CONDITION is expressed as either *atomic* when it consists of only one term or *compound* when it is composed of several terms (Figure 9). Examples of atomic conditions include: "if the user is colour-blind", "if there is no ongoing task", "if the PC battery gets low", "if there is a mobile device near the user", "if the computing platform is a public display", "if the physical environment is shaky", "if the organisational environment is hierarchical".

An atomic condition concerns an observable (its subject) that can belong to the interactive system in its business domain of application (e.g., adapt the UI on business processes [92]) or adaptation parts, the context of use with its components USER, PLATFORM, and ENVIRONMENT, or the deployment of the interactive system in its context of use. The condition is expressed as a formula of the first-order logic PREDICATE evaluated with potential arguments and their names, such as isLOW(BATTERY) or isNEAR(USER). The expression can involve quantifiers (e.g., "For all", "There exists") that may be positive or negative.

A COMPOSITE CONDITION is a combination of conditions (atomic or composite) using operators. For example, "If the user's smartphone is switched on AND there is no ongoing task on it", "If the laptop battery has dropped AND the user turns on his/her mobile phone OR PDA". These examples implement two classic types of operators in task modeling [76]: logical and temporal. Temporal operators allow reasoning about the interaction history. Other types of operator could be imagined.

The condition can be decorated with PROPERTIES which can also be reasoned about (e.g., by property-based reasoning [11]), such as "If condition C is frequent", "If menu item I has a high probability of selection". Relevant properties include: the actor who evaluated the condition, the time of condition evaluation (Time), the confidence factor in the condition's evaluation, and the frequency of the condition being met. This detailed structure is crucial for creating effective adaptation operations that can be effectively and efficiently implemented to ensure AUIs under varying conditions.

4.1.2 Event

An EVENT identifies what has changed in the context of use and justifies an action in response to this change. Key elements include (Figure 10):

- The VARIABLE of change: the change can concern the user, the platform, or the environment. For example, if the user turns on a smartphone, the variable is the platform, which is now enriched with this device.
- The NATURE of the variation: in this case, it is the arrival (Addition) of a new platform.
- The INITIATOR of the variation: here, it is the user.

The EVENT can also have properties such as the confidence level in perception, the actor of perception, or the frequency of the event. Reasoning can be applied to these properties. For example, the adaptation operation "If the user is in front of the PC and the confidence factor for this event is 100%, then...". The variables identify what has changed in the context of use. According to the ontology of Crowley et al.

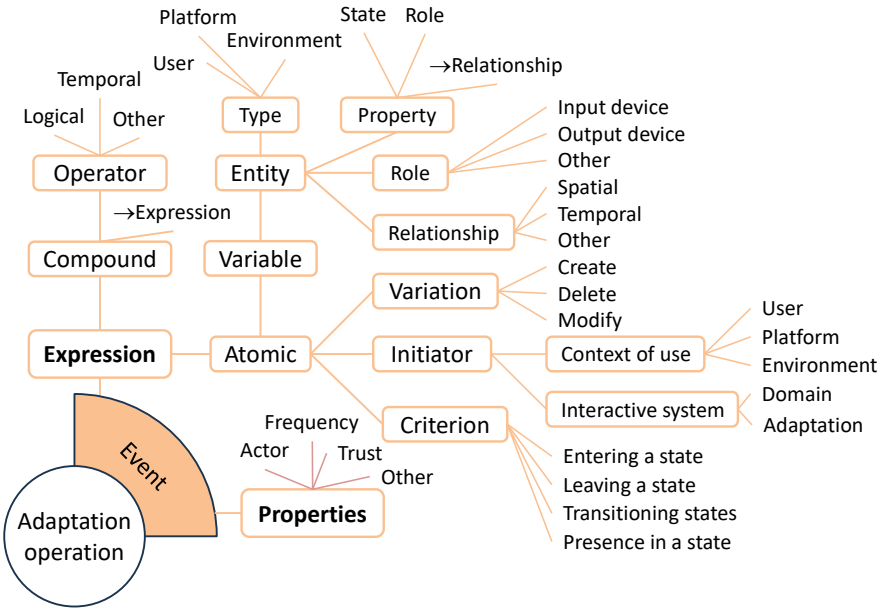


Fig. 10: The EVENT quadrant of our design space for adaptation (Illustration owned by the authors).

(Crowley:2002), four types of changes can occur depending on whether they affect ENTITIES, ROLES, RELATIONS, or ASSOCIATIONS between entities, roles, and relations:

Fig. 11: Selection of menu item adapted to the hand [7] (Illustration by courtesy of Peter Antoniac).

- ENTITIES: they pertain to the living or inanimate physical world. An entity is a grouping of observables. By measuring observables, the application can detect the presence of physical world entities, recognise them (e.g., by gesture recognition through surfaces [88]), track them, and determine the value of their attributes and relationships. We generalise this to all observable components of the context of use: user, platform, and environment (TYPE).

- ROLES: they are the functions held by these entities. The roles of input and output devices are predominant. In the past, these roles were assigned to the traditional screen with the keyboard and mouse trio. Today, they are replaced by any physical entity that has the right properties and is observable by the system. For example, a hand, traditionally used as a pointing device, becomes a display surface [7]. Its proximity to the user and the elongated shape of the fingers make it suitable for menu display (Figure 11). A single entity can have multiple roles. In contrast, a role can be held by several entities simultaneously.
- RELATIONS: they pertain to a set of entities. They can be spatial (e.g., "the user is near the platform"), temporal (e.g., "the device will be available during 5 minutes"), functional (e.g., "the sun illuminates the plant". Identifying the variables means locating the change in terms of entities (i.e., state, roles, or relations) as well as the set of roles and relations held.
- VARIATION: it characterizes the type of change. According to Crowley's ontology [27], the notions of context of use (or usage context) and situation are distinguished. A CONTEXT OF USE is defined by a set of roles and relations held by a set of entities. A context change occurs when a role or relation appears (CREATE), is modified (MODIFY), or disappears (DELETE). A SITUATION is characterised by a mapping between entities, roles, and relations. A situation change occurs when these mappings change (CREATE or DELETE associations between entities and roles/relations) or when entities appear (CREATE) or disappear (DELETE). MODIFY is planned to integrate changes in the state of entities into the reflection if necessary. This ontology allows the establishment of a graph of contexts and situations. An event is implicitly defined as an arc of this graph. There are two types of events depending on whether they are *intra-context* (i.e., between situations within the same context) or *inter-contexts* (i.e., between situations of different contexts).

Schmidt [85] defines the notion of an event: it is not necessarily the conjunction of an exit from one node (context or situation) and an entry into another node. An event can be more atomic, an exit from a node, an entry into a node, or even the presence in a node. Thus, we distinguish atomic events from compound events. Similarly to compound conditions, compound events use operators, particularly logical operators.

The INITIATOR of the change identifies the primary responsible agent for the variation. This information can be relevant to avoid conflicting with user actions. For example, if the user has moved the UI to a less visible area of the screen, it would be inappropriate to recentralise the UI in response. The initiator can be the interactive system or its usage context. In the interactive system, we distinguish between the business parts and the adaptation parts. For example, the interactive system DIFFIE initiates the content adaptation and highlights the results since the last visit of the user (Figure 12). For the context of use, we refer to its three components: user, platform, and environment [20]. By identifying these changes and their properties, the adaptation can better respond to the dynamic conditions in which an AUI operates, ensuring continued usability and satisfaction.

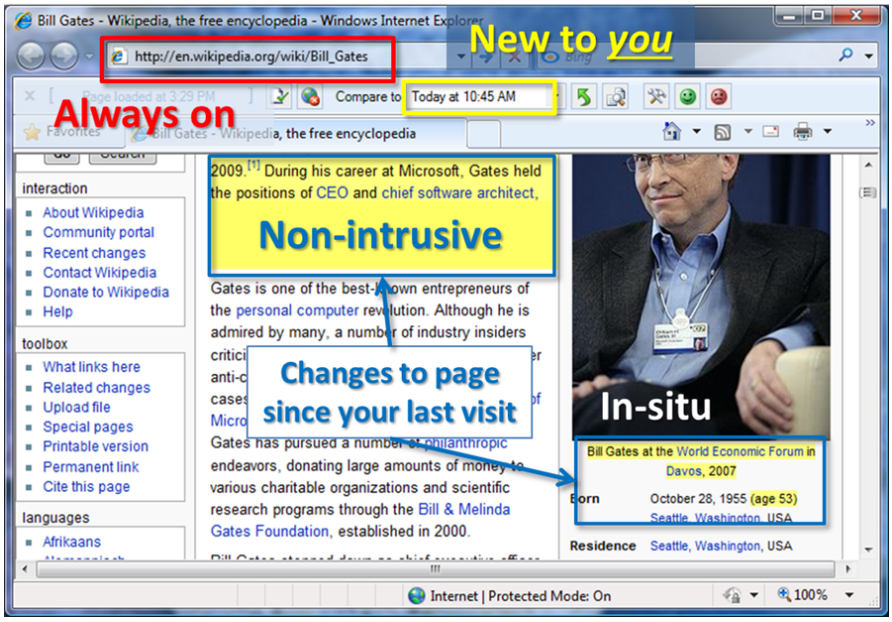


Fig. 12: DIFFIE adapts the content and highlights the changes since the last visit [94] (Illustration by courtesy of Jaime Teevan).

4.1.3 Action

The ACTION specifies the reaction to implement following a change in the context of use (Figure 13). Examples of atomic actions include: "Migrate the UI to the nearest platform" [39], "Remove infrequent tasks" [37], "Execute a specific UI" [19], "Prioritise remodeling" [97], "Eliminate automatic distribution" [46], "Declare that the user is tired", "Add a specific adaptation operation on top of existing ones", "Execute the first applicable operation possible".

Compound actions can also be imagined by combining actions (atomic or compound) through operators (e.g., logical or temporal). Actions, whether atomic or compound, can be decorated with PROPERTIES. Relevant properties include specifying the actor responsible for their execution and the timing of their execution (TIME). An atomic action is issued with a certain FORCE: the action is either PROPOSED or IMPOSED. Actions are governed by control policies, such as "Execute all imposed actions". CREATE, DELETE, and MODIFY are action types for control policies. These are STATEMENT actions as opposed to STRATEGIES.

STRATEGIES manipulate STATEMENTS: they filter or weight them. Four types of statements are defined: POLICIES, FACTS, GUIDELINES, and TARGET INTERVENTIONS. TARGET INTERVENTIONS act on the context of use (e.g., "turning on the light") or on the interactive system. For the interactive system, we functionally refine the target into either the business functions (i.e., related to the application domain) or the

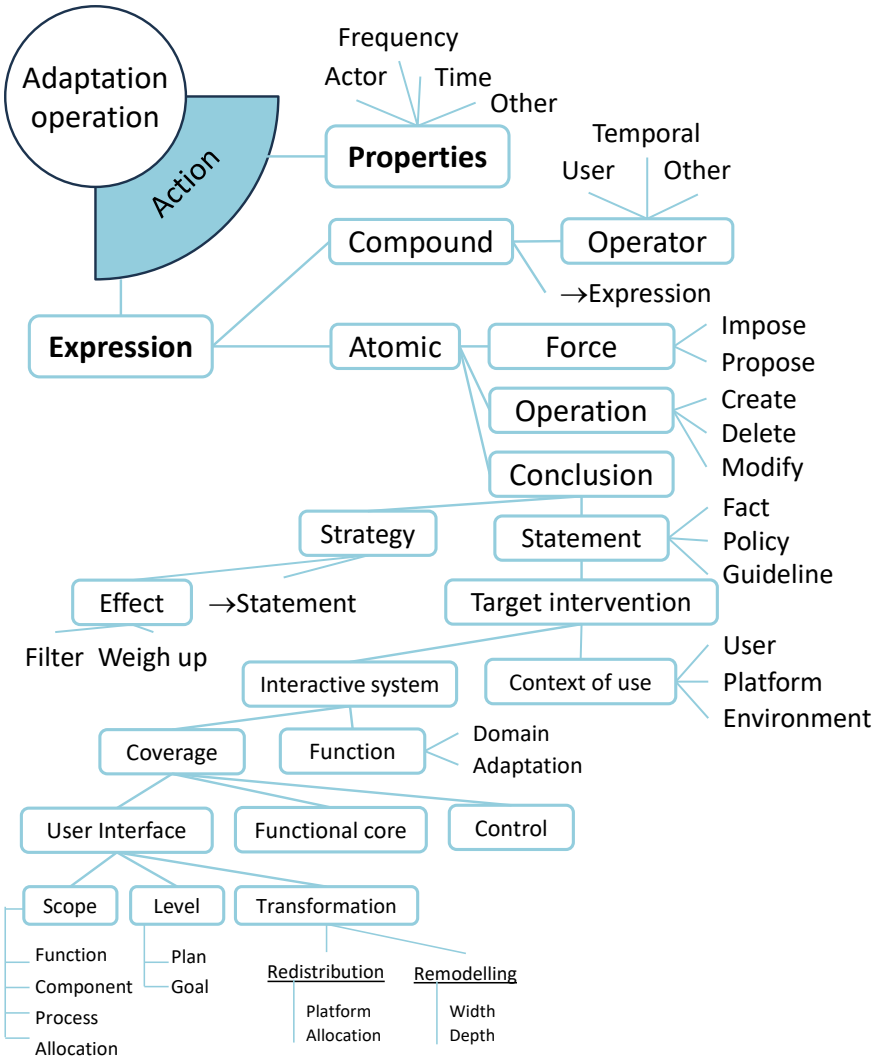


Fig. 13: The ACTION quadrant of our design space for adaptation (Illustration owned by the authors).

adaptation itself. In turn, the functional COVERAGE needs to be specified in terms of the components being modified (i.e., UI, functional core, and/or dialog control). For the FUNCTIONAL CORE, we simply nuance whether the adaptation concerns the intrinsic functional core or its deployment in the context of use.

Interventions in the UI can be explored from the perspective of software architecture (SCOPE): which elements (i.e., functions, components, processes) and allocations between elements (i.e., functions to components, components to pro-

cesses, processes to physical resources) are concerned. The considered elements and allocations depend on the adaptation level [97]: REMODELING acts on a constant distribution state of the UI on interaction resources, unlike REDISTRIBUTION. Remodeling and redistribution can be specified in terms of PLAN or GOAL. The goal sets the objective to be achieved (e.g., "migrate the UI") without imposing any conceptual or implementation solution, unlike the plan, where all degrees of freedom are fixed.

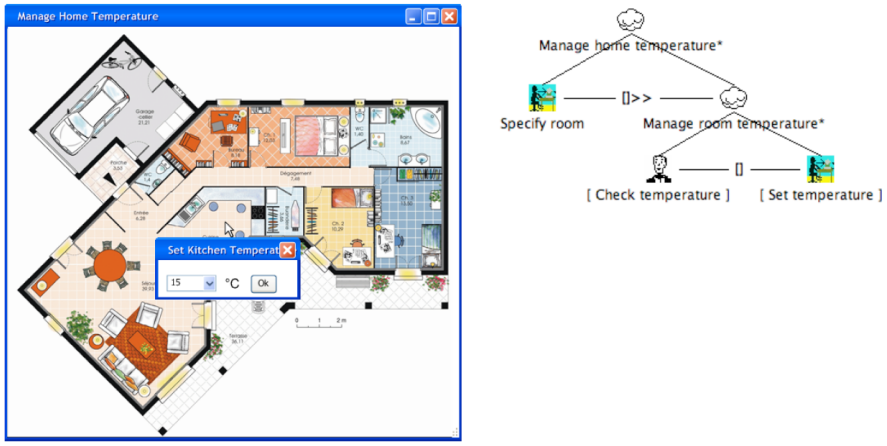


Fig. 14: Example of a task model for home automation and its related final UI [89] (Illustration owned by the first author).

Research in plasticity [19] began studying remodeling on various abstraction levels such as the task level (user tasks [69] and concepts [96]), the abstract UI level, the concrete UI level, and the final UI level.

The TASK LEVEL describes the user's task and domain concepts independently of any representation and implementation. ConcurTaskTree and its language [76], UML class diagrams of OWL ontologies, are usually considered for task and domain modeling, respectively. They can be decorated with properties such as frequency, iteration, and optionality. For example, Figure 14 illustrates task modelling for a home automation application. The user controls the home temperature ("Manage home temperature") by iteratively (i.e., decoration "*") handling the different rooms of the house. Handling a room involves specifying a command ("Specify command") and optionally checking the feedback ("Check feedback" with its optionality decoration). Specifying a command involves specifying the room of interest ("Specify room") and the action to apply: either check its temperature ("Check temperature") or change its temperature ("Set temperature"). User Interface Description Languages (UIDLs), such as UsiXML [63] or MariaXML [76] capture these aspects in a Domain Specific Language.

The FINAL UI LEVEL corresponds to the particular implementation of the UI for a given platform in any programming or markup language. For example, Figure 15

performs adaptation at the final UI level by adapting the layout and its widgets depending on the length and height of the window, but also depending on the different time zones when the user needs to fly between two areas.

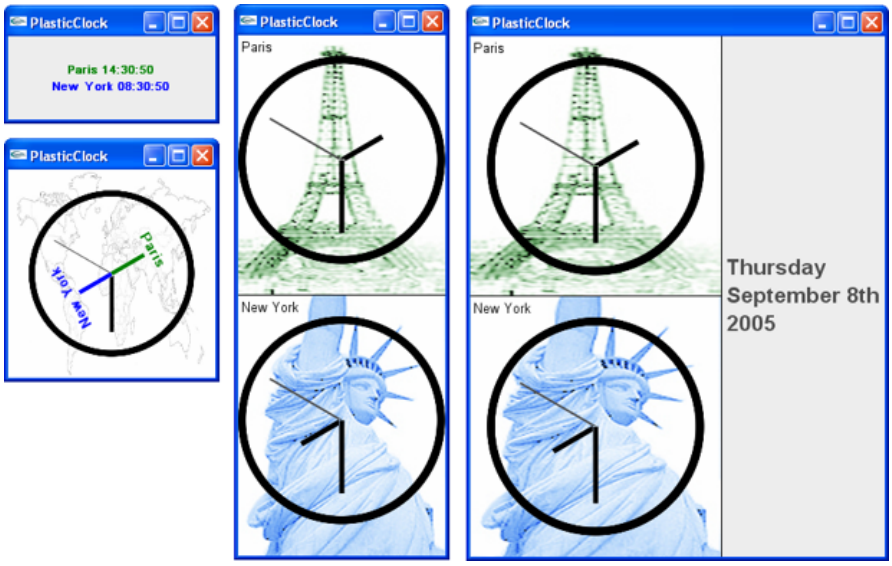


Fig. 15: PLASTICLOCK: the final UI changes its layout and widgets depending on the window dimensions and time zone [21] (Illustrations owned by the first author).

4.1.4 Value

In economics, VALUE measures the ratio between a benefit and a cost, which is suitable for adaptation, which always induces some benefits and drawbacks for a cost [59]. In this vein, we consider two elements: the application benefit (which we generalise into any positive or negative effect) and the implementation cost related to the execution of the adaptation operation. The cost is practical, as opposed to the effect, which is theoretical. The effect captures the expected outcomes of the adaptation operation by formulating properties expressed in a given reference frame:

- ENSURED by the application of the adaptation operation: they were not ensured before the application and they become so after the adaptation operation is completed. For example, task continuity is ensured after migration.
- PRESERVED: they were initially satisfied and remained so. For example, a domain concept is observable [24] and is still observable after adaptation.
- IMPROVED: they were partially satisfied and are now improved. For example, the cognitive load in terms of informational density is improved by moving to a large screen or multiple displays [87].

- DEGRADED: they were partially satisfied and are now deteriorated. For example, graceful degradation [37] to a small screen increases navigation among tasks and increases workload in terms of physical actions.
- REMOVED: they were initially satisfied and they are no longer satisfied. For example, the removal of user guidance due to a lack of display surface will no longer satisfy this property.

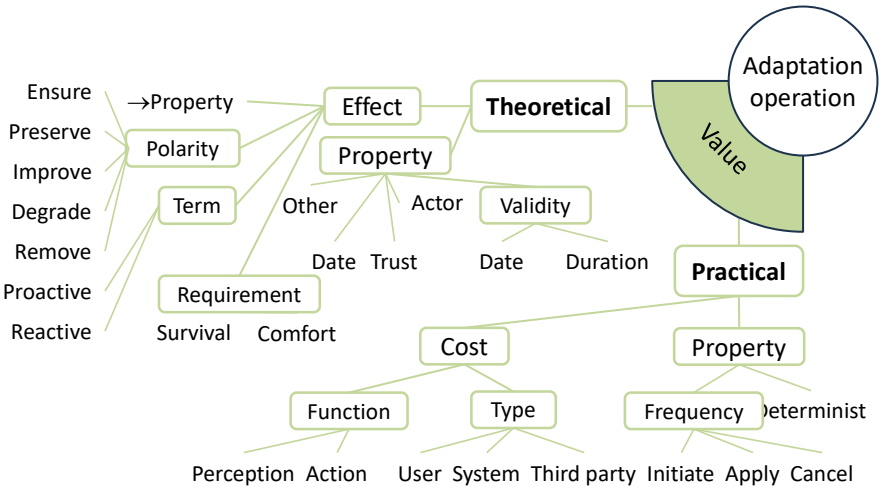


Fig. 16: The VALUE quadrant of our design space for adaptation (Illustration owned by the authors).

The EFFECT can be expressed more globally by applying the adaptation operation: does the operation pertain to the survival of the interactive system or the user's comfort? If it is about user comfort, is it functional, i.e., related to the system's utility, or extra-functional related to the usability of the adapted UI and/or the meta-UI [26] (defined as the UI governing the UI adaptation, later renamed into extra-UI [68]).

Whether the adaptation operation pertains to the survival of the interactive system or user comfort, it can be applied PROACTIVELY (i.e., in anticipation of a change of context), or REACTIVELY (i.e., to face a reality). We call this dimension the TERM.

The THEORETICAL VALUE can be decorated with PROPERTIES that are useful for reasoning about the adaptation. We identify as relevant the VALIDITY of the adaptation operation (i.e., expiration date or duration), the TRUST placed in the operation, its ACTOR, and its DATE OF ISSUE (e.g., a recent form adaptation is more trusted than an old one [35]). Such values can also incorporate software quality factors [53].

The PRACTICAL VALUE considers the cost of applying the operation. This cost is measured from a system perspective (e.g., by estimating digital and physical resources required for calculation, communication, and interaction) but also from a human (e.g., perceptual, cognitive, and motor load) for PERCEPTION (condition and event) and ACTION. Estimating the cost is a challenging aspect of UI adaptation.

The practical value can also be decorated with properties that are evaluated by practice: the frequency of triggering the adaptation, its application and cancellation frequencies [35], its determinism, which integrates the level of abstraction of the operation. For example, icons in a toolbar (Figure 17) can be promoted (made larger) or demoted (made smaller) depending on their probability of being selected [17]. This probability can be computed by recency, frequency, recurrence, importance, or any combination of them [98, 99]. The adaptation operations being defined, the next section studies their life cycle.

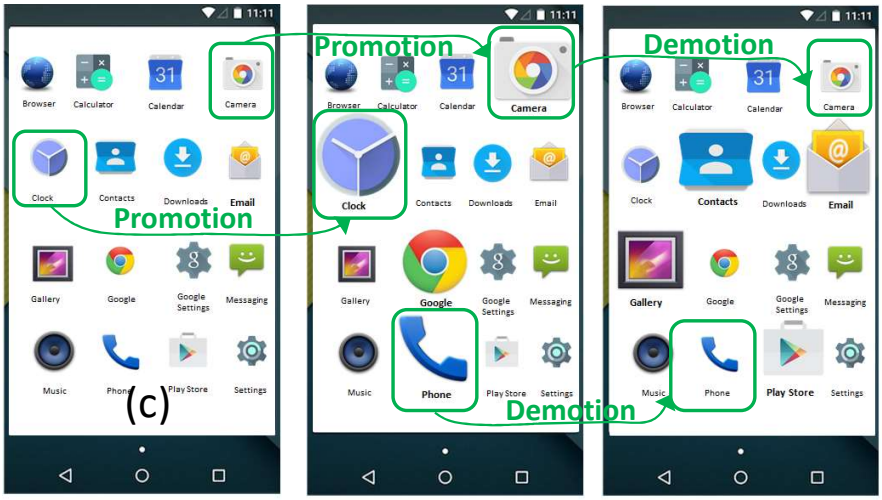


Fig. 17: Promotion or demotion of icons depending on their probability of use [17] (Illustrations owned by the first author).

4.2 Categories of Adaptation Operations

While any adaptation operation can be defined according to the terms defined in Section 4, some frequent categories of adaptation operations emerge according to the effect types they produce [78, 37]. We detail them in the next sub-sections.

4.2.1 Property-Changing Operations in Adaptation

These operations adjust the properties of the interface without replacing or splitting elements, allowing for smooth transitions between different states of a UI based on real-time conditions. Property-changing operations are essential for dynamically modifying the presentation and behaviour of UI elements in response to changes in

context or user interactions. A common example of a property-changing operation can be observed in the calculator interface on a smartphone, which dynamically adjusts based on the device's orientation property.

Example: smartphone calculator. When a smartphone is held vertically in portrait mode, the calculator default configuration displays only simple arithmetic operations (**??**). When the smartphone is turned horizontally in landscape mode, it provides more space to display sophisticated scientific functions, such as trigonometric functions (**??**). The underlying components remain the same, but the orientation affects how they are displayed and arranged. This modification is a representative example of an adaptation operation that modifies a property. In contrast, MiniAba [84] enables the end users to (un)select any subset of arithmetic, trigonometric, or advanced functions they want and to recompile the project to get an adapted version.

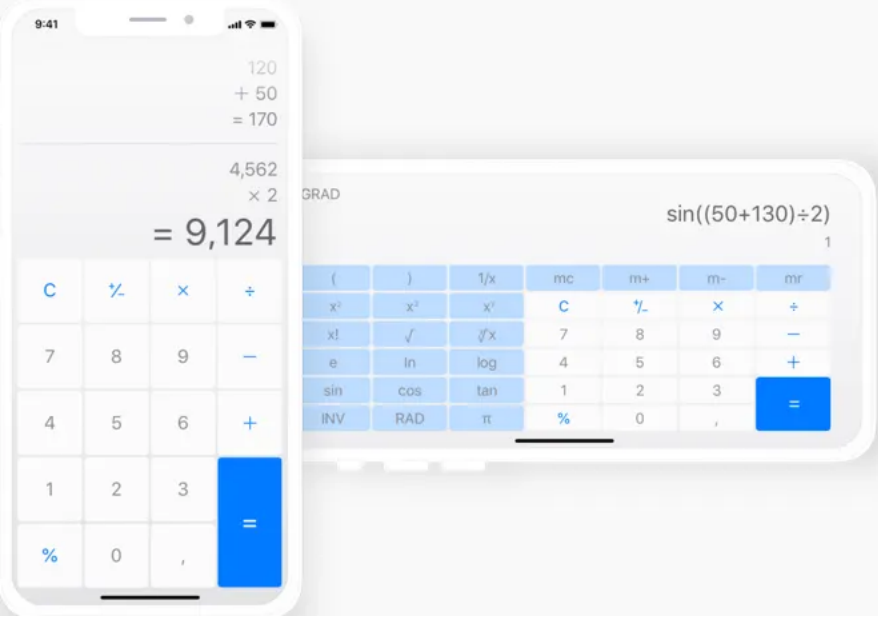


Fig. 18: Comparison of smartphone calculator in different modes: (a) portrait and (b) landscape (Illustrations by courtesy of Charlotte Doughty).

This adaptation operation can be represented as follows:
On Change(Orientation)
If Screen.Orientation=vertical
Then (Screen.orientation=horizontal And Display(FullCalculator))
For Supporting expert mode.

1. EVENT: the user rotates the smartphone from portrait to landscape mode, where the rotation sensor in the device detects that the phone has been turned.
2. CONDITION: the device orientation changes from vertical (portrait) to horizontal (landscape). The rotation must be 90° to transition the screen from portrait to landscape.
3. ACTION: the calculator UI reveals advanced functions (scientific mode), modifying the layout and visibility of the interface elements and dynamically adjusting its layout to display more advanced functions like trigonometric operators, exponential functions, and memory functions. In this case, the UI structure remains the same, but its presentation properties (such as visible elements and screen space allocation) are modified to accommodate the additional functions.
4. VALUE: to enhance user interaction by dynamically displaying advanced functions, improving efficiency for expert users who need scientific calculations.

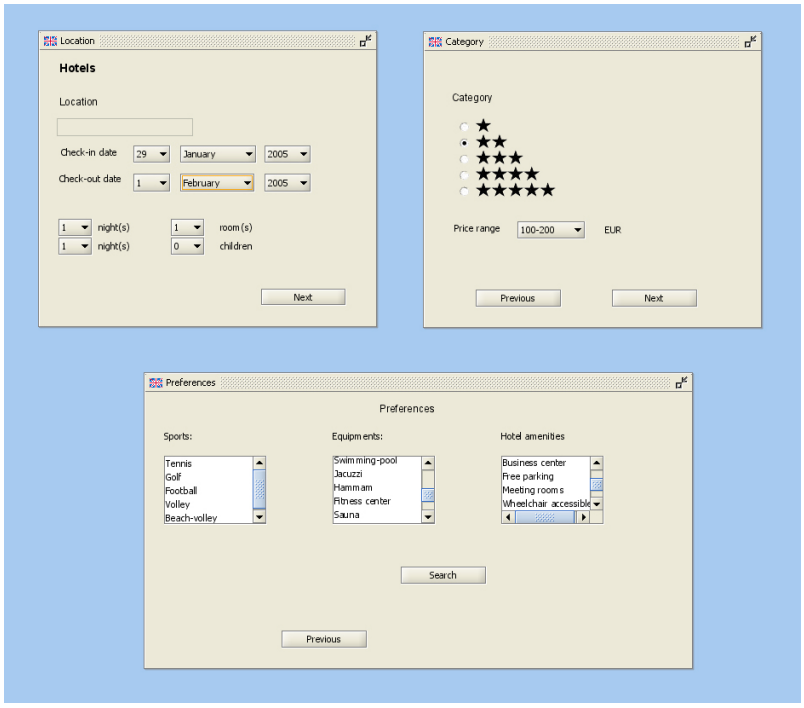
4.2.2 Splitting Operations

Splitting operations divide a UI into smaller, manageable components, elements, or interaction spaces to adapt to devices with different display capabilities. These operations ensure the usability and consistency [8] across multiple platforms [38]:

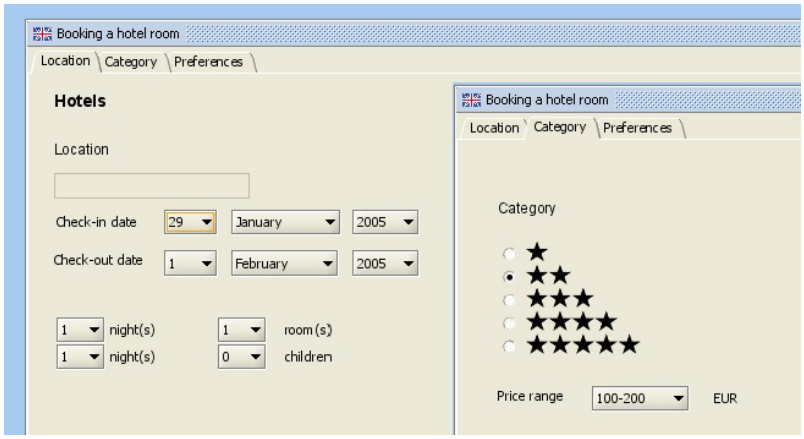
1. *Class 1: Split in sequential operators*: if some tasks are arranged in a sequence (e.g., Step 1 → Step 2 → Step 3), the UI before each sequential task can be split to create separate a screen or a page for each step, such as in a wizard widget.
2. *Class 2: Split before optional tasks*: if a sequence includes optional tasks, the UI can be split before the optional task to avoid users see unnecessary options unless needed.
3. *Class 3: Use concurrent operators when sequential splitting is not possible*: if sequential splitting is not adequate or possible, the UI can be split using concurrent operators, which indicate tasks that can occur in any order.
4. *Class 4: Split at the highest level*: when multiple levels of tasks can be split, choose the highest level. This keeps semantically related tasks together.
5. *Class 5: Distribution of tasks*: when splitting occurs under a higher-priority operator, ensure that necessary tasks, like a "Cancel" option, are available in all resulting interaction spaces.

Example: Book a hotel room. A form used to book a hotel room usually consists of specifying three parts: the *location* with the arrival and departure dates and the number of guests, the selection of the hotel *category*, and a selection of hotel *preferences*. This form can be split into three independent windows (Class 3) as illustrated in Figure 19a or collected in a tabbed dialog box (Class 4)) as reproduced in Figure 19b. When the first task related to user information (e.g., user identification) is completed, the subsequent task of specifying booking details, as represented above, and payment information can be represented as follows:

ON CHANGE(SCREEN.SIZE)



(a) A hotel room in three separate windows.



(b) A hotel room in a tabbed dialog box.

Fig. 19: Splitting rules applied to a hotel room booking form [38] (Illustrations by courtesy of Jean Vanderdonckt).

If Screen.Size = Small And Task.Operator = Sequential (e.g., User Information → Booking Details → Payment Information)

Then Split Interaction.Space Before Task "Booking Details"

And Create Interaction.Space "Booking Details"

For Consistency

The task model could also comprise an optional task where the guest can specify any special request, such as a non-smoking room with wi-fi access. This optional task is therefore subject to the following splitting rule (Class 2):

On Change(Screen.Size)

If Screen.Size = Small and Task.Type ("Special Requests") = Optional

Then Split Interaction.Space Before Task="Special Requests"

Create Interaction.Space ("Special Requests")

For Optimization

This adaptation operation ensures that the optional task "Special Requests" is displayed only when necessary to optimize the screen real estate when the screen resolution is constrained. This is part of the overall UI transformation shown in Figure 19b, where navigation is adapted:

On Change(Screen.Size)

If Screen.Size = Small And Task.operator = Concurrent (e.g., User Information || Payment Information)

Then Create Interaction.Space ("User Information", "Payment Information") and Create Navigation.Link ("User Information", "Payment Information")

For Reducing(CognitiveLoad).

4.2.3 Replacement Operations

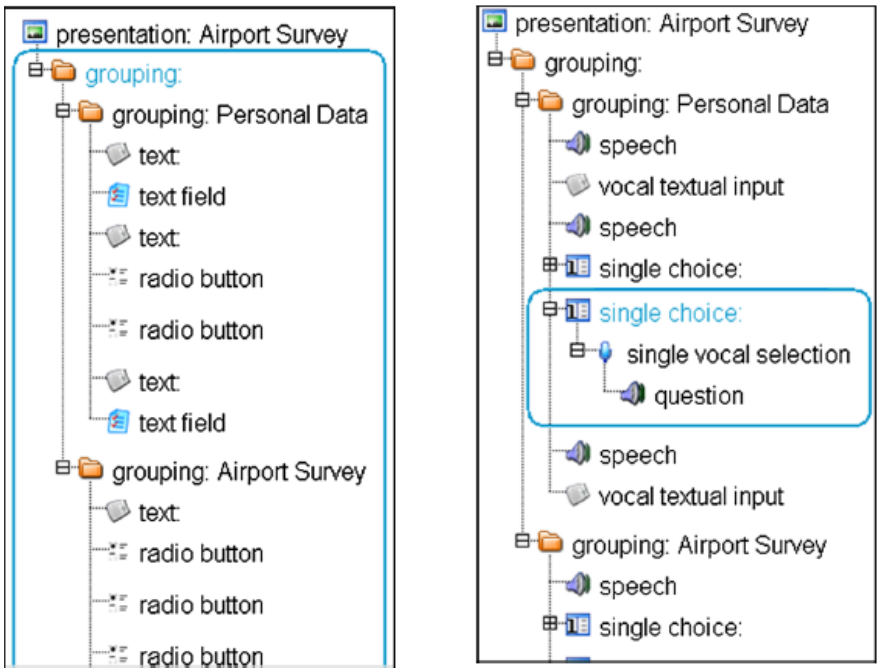
To address the needs of users with disabilities, replacement operations entail switching out certain types of UI elements with others. By offering alternate modes of interaction for the same task or action, the expected value concerns accessibility. For example, motor or visually impaired users who need to interact with push buttons, perform text input, and select options in a list, could experience trouble. Replacing these elements or enriching them with voice commands or audio feedback should support a better user experience for them [70]. Figure 20a shows the hierarchy of UI elements for entering personal data through such buttons, text fields, and radio buttons to support graphical interaction. Figure 20b shows the adapted UI hierarchy where graphical elements have been replaced by speech recognition, vocal textual input and selection with audio prompts to support vocal interaction. This adaptation operation can be represented as follows:

On Detect(User.Disability)=true — This event triggers the operation whenever a user's disability is detected, such as when the user logs in or the context changes.

IF UI.ELEMENTS=GRAPHICAL — The condition checks if the current UI elements are graphical. This means the user interface contains some elements like buttons, text fields, or icons that may not be accessible to users with certain disabilities.

THEN REPLACE(UI.ELEMENTS, GRAPHICAL, VOCAL) AND CREATE(AUDIOCOMMANDS) — This action replaces the graphical UI elements with vocal elements and audio commands. The user can now interact with the interface using voice commands and receive audio feedback instead of visual prompts.

FOR ACCESSIBILITY



(a) For graphical interaction.

(b) For vocal interaction.

Fig. 20: Hierarchy of UI elements to input personal data [77]: (a) before adaptation, (b) after adaptation (Illustrations by courtesy of Fabio Paternò).

4.2.4 Removal Operations

Removal operations remove UI elements at run-time for several reasons: they are irrelevant, inappropriate, unnecessary for particular user groups or use cases, or consuming too many resources, such as screen space. Removal operations could physically remove these UI elements, momentarily hide them, or disable them. In contrast to replacement operations, which replace UI elements with alternatives,

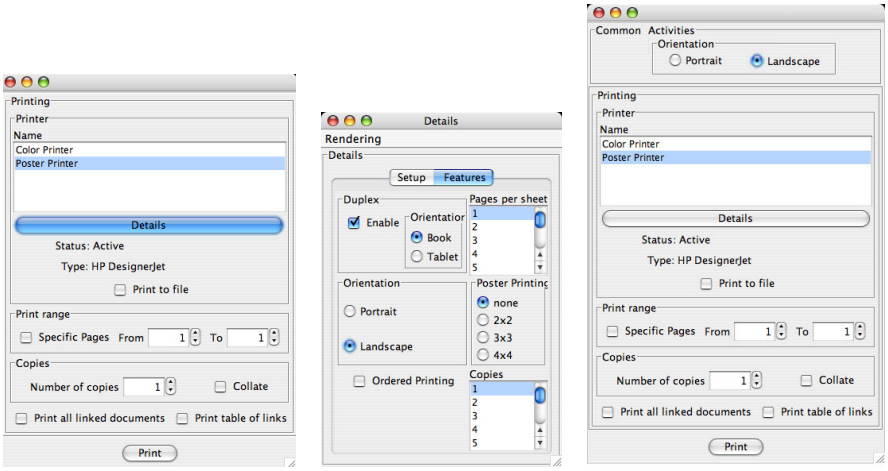


Fig. 21: (a) Original print dialog box with multiple steps to access printing orientation. (b) Adapted dialog box with simplified access to printing orientation, removing unnecessary steps [41] (Illustrations by courtesy of Krzysztof Gajos).

removal operations simply delete them, either momentarily or permanently. For example, SUPPLE [41] dynamically adapts UIs based on user behaviour and needs. In Figure 21, a print dialog box is adapted to prioritize frequently used options and remove less relevant components. Figure 21 demonstrates how the UI is adapted to reduce complexity. In the original print dialog box (Figure 21a), changing the print orientation from portrait to landscape requires navigating through multiple steps and options, which can be difficult for users with motor impairments or limited cognitive abilities. This complexity is not only time-consuming, but also increases the risk of errors during interaction. The adapted interface (Figure 21b) simplifies this process by removing unnecessary steps and presenting the landscape printing option directly on the main screen, which reduces the need for complex navigation and makes the UI more intuitive and accessible.

By focusing on the most commonly used functions and removing less relevant options, SUPPLE tailors the UI to the user's needs, resulting in a streamlined interaction experience. This adaptation operation can be represented as follows:

ON USER.ACCESS("LANDSCAPEOPTION") AND FREQUENCY("LANDSCAPEOPTION") > THRESHOLD
IF UI.CONTAINS("INTERMEDIATESTEPS")
THEN UI.REMOVE("INTERMEDIATESTEPS") AND UI.DISPLAY("LANDSCAPEOPTION", "MAINSCREEN")
ON USER.PREFERENCE("SIMPLIFIEDUI")
IF UI.CONTAINS("RARELYUSEDOPTIONS")
THEN UI.REMOVE("RARELYUSEDOPTIONS")
FOR REDUCING(TASKTIME)

- Event: the system detects frequent access to the landscape printing function by comparing it to a reference threshold.
- Condition: the "Print" dialog box includes multiple steps for accessing this function.
- Action: the system removes intermediate steps and presents the landscape printing option by default, simplifying the interaction.
- Value: the adaptation aims at simplifying the UI by reducing the task completion time.

This operation dynamically simplifies the user interface based on the user's capabilities, ensuring a more accessible and user-friendly experience.

4.3 The Left Hemisphere for the Specification of the Adaptation Implementation

The left hemisphere identifies four stages in the life cycle of an adaptation operation (Figure 22): the definition of an adaptation operation, its execution, its evaluation and, finally, the consolidation of experience gained with this adaptation. For each stage, we are encouraged to answer the same questions of the Quintilian hexameter [72], as outlined in section 1.

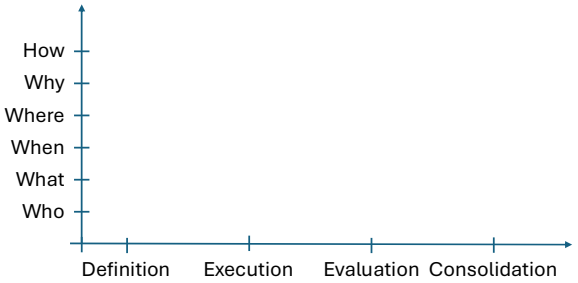


Fig. 22: Left hemisphere: stages of adaptation life cycle (Illustration owned by the authors).

4.3.1 Definition of the Adaptation Operation

The *definition* of the adaptation operation is hereby referred to as any specification, possibly by code generation or interpretation of the specification, of the precise instructions to be performed by an adaptation operation. This specification is aimed at enabling the concrete execution of an operation in the next stage. This should not be confused with the specification of adaptation (Figure 22), which specifies

which entity is responsible for deciding an adaptation, whereas this definition is aimed at building the adaptation operation itself. This encompasses the research, the design, and the coding of an operation. In regard to *What?*, we distinguish the specification of the overall goals of the operation (e.g., minimize the cognitive load) and the specification of the operation itself (e.g., "Remove optional widgets"). By overall goals, we may consider the specification of the VALUE to be guaranteed and the domains or zones of plasticity [25] to be guaranteed for this value [19]. Specifications can be written by (*Who?*) experts in human factors or adaptation, the designer, the end user, the interactive system, any third party, or any combination of them. For example, SCALER [35] enables the definition of an adaptation operation by configuring and balancing weights (Figure 23) specified by a designer, a developer, and the end-user, either in isolation one by one or from a crowd, as suggested by Nichols et al. [73].

The specification tool (*How?*) depends, of course, on the actor in charge of specification. Adaptation operations can be defined according to a wide range of tools, spanning from hard-coded environments (e.g., the graceful degradation plug-in offers three families of adaptation operations that are all predefined and coded in Java [37] and knowledge bases (e.g., operations for adapting a graphical UI to a mobile device in [34] are stored in a knowledge base that is processed by an inference engine) to specification environments (e.g., operations are defined in a domain specific language) and model-driven architectures [89, 90].

Depending on the tools (*How?*), the actor (*Who?*), the definition can be achieved at different times (*When?*): design time, linking time, compilation time, installation or deployment of the interactive system, its execution and inter-session, requiring the interactive system to be stopped and then restarted. Operations can be stored (*Where?*) within the interactive system itself (internal) or externally, in configuration files for example.

4.3.2 Execution of the Adaptation Operation

Adaptation is a seven-stage process (Figure 22) involving (*What?*) detecting the change in the context of use to take the initiative for adaptation (stage 2), to specify which adaptation (stage 3) and apply it (stage 4, which corresponds to the actual *execution* of the adaptation operation). These functions require mechanisms that are (*Where?*) internal or external to the interactive system and its AUI. These functions can be carried out by four types of actors (*Who?*): the end-user, another user (e.g., a controller), the interactive system, or another system. We identify five key moments for their execution (*When?*): either during execution (e.g., at any time or at precise moments), the granularity of the stage can range from the physical action to the session, via the elementary task and the compound task. Depending on the support tool, past actions may be lost or retained.

Support tools (*How?*) can be generic and reusable, or integrated into the interactive system, and consequently non-reusable. They can be equipped with an extra-UI [26] (formerly known as meta-UI), which is itself a UI that ensures ob-

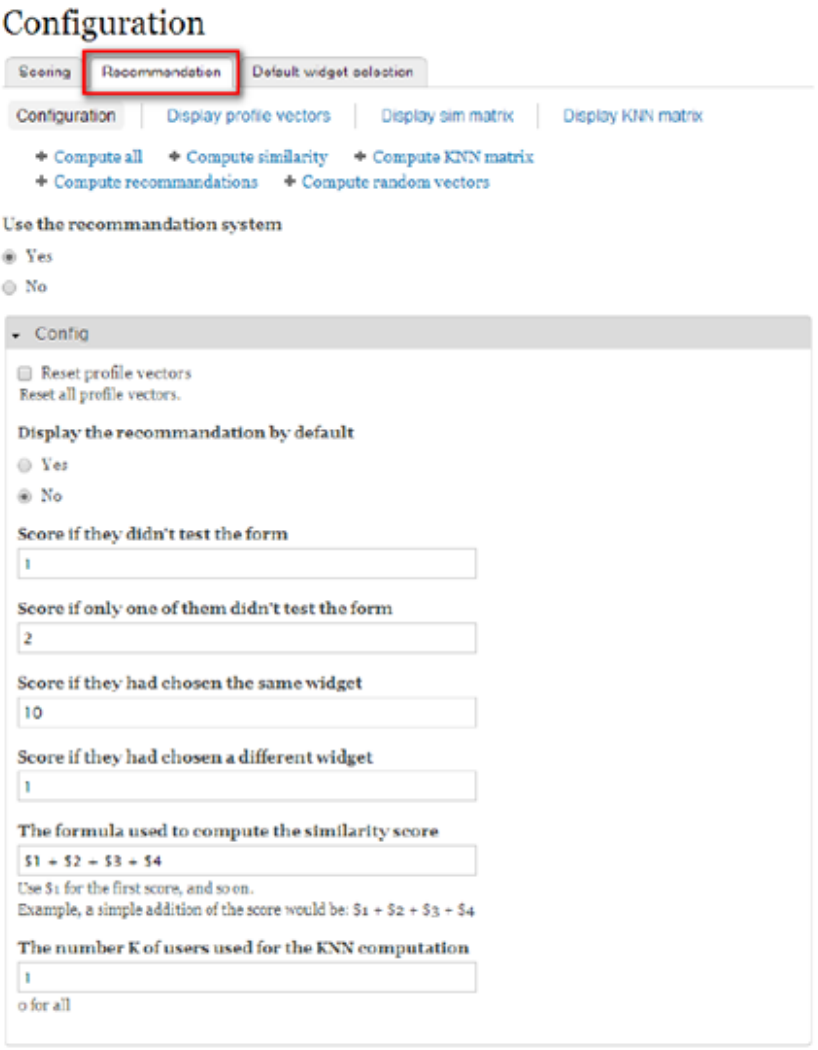


Fig. 23: Configuration of parameters for adaptivity by unsupervised learning [35] (Illustration by courtesy of Diego Eloi).

servability and control of the adaptation [16]. This extra-UI may require resources (e.g., UI components, libraries, perception services, agents, inference engines) that are either prefabricated or generated at runtime [12]. Their location is internal or external to the interactive system and their availability can be perennial (e.g., beyond adaptation) or temporary (e.g., erased after adaptation).

4.3.3 Evaluation of the Adaptation Operation

The *evaluation* is a process that critically examines to what extent the adaptation operation has been effective and efficient, which involves collecting and analysing data and outcomes resulting from its execution and outcomes. Its purpose is to assess the overall quality of an adaptation operation to improve its effectiveness and to inform future executions. We identify five actors (*Who?*) likely to be involved in evaluation: an evaluation specialist, a designer, the end-user, and the interactive system or any third party, provided that they hold the computational capabilities to perform such an evaluation. The evaluation can cover the four stages: the definition of an adaptation operation, its execution, its evaluation (which would then be a recursive process: how to evaluate the evaluation) and its consolidation of experience. For example, Figure 5 shows how an adaptation operation can be evaluated by the system itself (e.g., by performing some automatic evaluation) or by the end user qualitatively (e.g., using a rating bar) or quantitatively (e.g., using a rating scale). In the past, automatic UI evaluation has been explored, such as by guideline review [9] or agent analysis [65]. More recently, eye tracking and emotion recognition represent attractive candidates to determine to what extent the end-user is happy or not with an adaptation [48].

Evaluation can be carried out in the very true context of use or controlled in a specific environment, such as in a usability laboratory. It can be carried out on the fly or off-line (*When?*). For example, captured on-the-fly, the first impression that an end user produces is indicative of the graphical UI being good or bad [48]. Mechanisms for showing how adaptivity has been performed (e.g., Figure 24 shows adaptivity at run-time using an animated transition that preserves the context of use) and for explaining it are welcome.

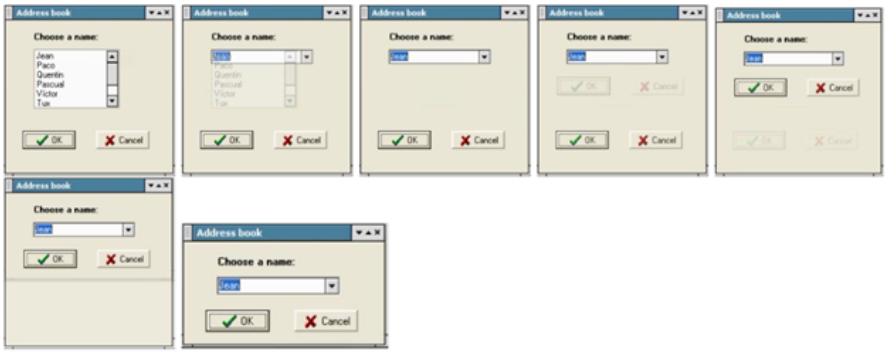


Fig. 24: Animated transition showing the various stages of adaptivity [30] (Illustrations by courtesy of Charles-Eric Dessart).

4.3.4 Consolidation of Experience

Consolidation of the experience gained with the adaptation operation is a new dimension that is required for future use by learning what happened (e.g., by keeping an appropriate operation and discarding an inappropriate one [105]). Five players are likely to be involved (*Who?*): an expert in knowledge management or computational approaches [55], an experienced designer, the end user, the interactive system and its AUI or any other party. Consolidation can be general, covering the value of operations (weak or strong), or limited to values only (*What?*). Consolidation requires software mechanisms (*How?*), e.g., ML/DL/RL tools [16, 44, 95], which can be external or embedded in the interactive system itself. The support offered can range from simple memorisation of experiences to the deduction of new adaptation operations. Results can be organized or not, justified or not, stored internally or externally to the interactive system (*Where?*), and consolidated on-the-fly or off-line (*When?*).

5 Practical Lessons from Experience

Existing methods for designing UIs of interactive systems (e.g., [24, 47]) still apply to AUIs, but with the need to pay attention to the context of use and the quality of the use. For both of them, it must be understood that they may result either from user-centered requirements elicited during the Analysis phase (see the Problem part of the design process in Figure 25) or from design choices made by the practitioner (see the Solution part of the design process in Figure 25). Several lessons can be drawn from experience. **Lesson n°1** recommends considering explicitly both the context of use and the quality in use, together with their rationale. User-centered requirements "must" be satisfied, while practitioner's decisions "should" be preserved for interaction continuity. Once the context of use is determined and the quality in use is decided accordingly, **lesson from experience n°2** recommends reasoning for the transitions, considering the change from one context of use to another. The **lesson from experience n°3** recommends designing a UI first for the most constrained context of use, i.e., the one combining the most constrained user, platform, and environment. This "reference UI" could then be considered for designing UIs for other contexts of use by progressively relaxing, which is referred to as *progressive enhancement*, the inverse of *graceful degradation* [37]. Serna et al. [86] demonstrates that this improves the quality of use of the resulting UIs. Consequently, **lesson from experience n°4** consists of using the AUIs as a means for quality, i.e., to consider very constrained contexts of use, even if they are not targeted, just for quality.

6 Conclusion and Perspectives

This paper provides an overview of UI adaptation since its inception until recent days, by focusing mainly on adaptive UIs, where adaptation is ensured by the interactive system or application itself, as opposed to adaptable UIs (where the end-user is responsible for adapting the UI) and mixed-initiative UIs (where the system and the end-user collaborate to ensure the adaptation).

For this purpose, we discuss the traditional questions of adaptation (i.e., what to adapt, why to adapt, how to adapt, with regard to what, who controls the adaptation, when to adapt, and where to adapt) that need to be answered when adaptation should be implemented in a UI. We then presented a generic UI adaptation life cycle, which can be decomposed into seven stages (e.g., ranging from UI adaptation goals to evaluation of adaptation). A targeted literature review resulted in a discussion of existing and future challenges encountered in adaptation and in recent advances in the domain, such as those induced by computational approaches. We then presented a comprehensive problem space for UI adaptation made up of two hemispheres: one for the specification of the adaptation operation (which is the cornerstone of adaptivity) and one for the specification of the implementation of adaptation. The first hemisphere is further decomposed into four quadrants: condition, event, action, and value. The second hemisphere is further divided into four stages: definition, execution, evaluation, and consolidation. From experience, we believe that these structuring elements inform and guide the engineering of adaptive UIs.

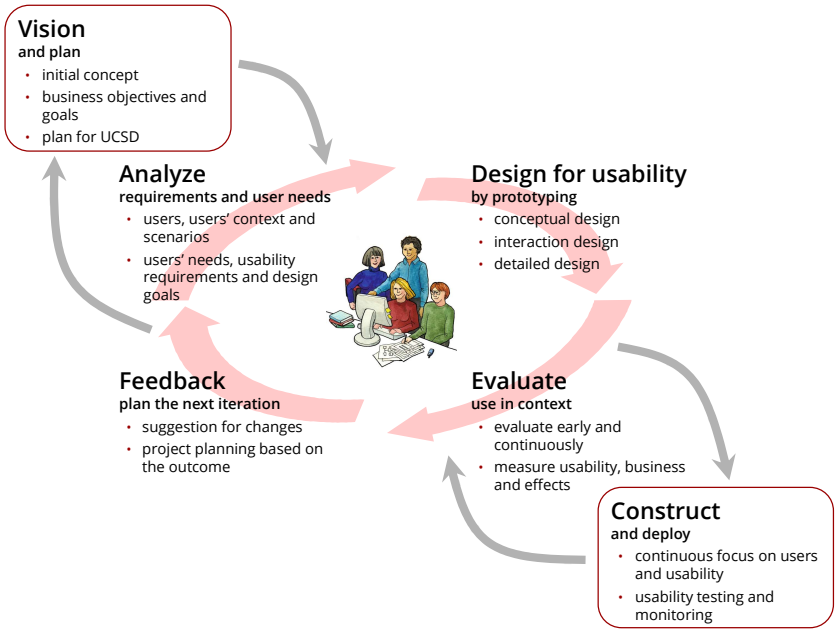


Fig. 25: Gulliksen's method for designing UIs, adapted from [47] (Illustration by courtesy of Jan Gulliksen and Bengt Göransson).

The greatest challenge posed by adaptation, now and in the future, is that of the evolution of adaptation. Adaptivity is inevitably an evolutionary process for both the end-user and the interactive system. Users evolve in their experience, the way they perform interactive tasks, and in their preferences. The environment in which they evolve also evolves inexorably. The interactive system cannot remain unaffected by these changes. To date, numerous methods have been applied to compensate for this lack of evolution, mainly by adding new adaptation rules, modifying and improving their application strategies, or taking better account of user parameters. This compensation remains insufficient to draw relevant and useful conclusions about the future of the interactive system.

That is why machine learning, in general, [66] and reinforcement learning in particular offer many suitable advantages, enabling us to take into account how the UI evolves in its context of use. The most recent advances [44, 58, 95, 105, 69, 35] are heading in this direction, and are already taking constructive steps towards overcoming this lack of consideration for evolution. A second potential avenue is to determine to what extent adaptation can be fragmented in time and space (e.g., Todi et al. [95] gradually transform graphical adaptive menus, Bouzit et al. [14] suggest step-by-step progress and adaptation in mobile menus) to deliver an adaptation experience that would reduce adverse effects [59], such as disruption [51].

We hope that the future will allow us to determine the best methods, techniques, and algorithms to ensure the best possible adaptivity. Meanwhile, lessons from experience are reported to reconcile generic know-how in engineering HCI and specific advances in AUIs.

Acknowledgements Alaa Sahraoui is supported by the EU EIC Pathfinder-Awareness Inside challenge Symbiotik project (1 Oct. 2022-30 Sep. 2026) under Grant no. 101071147.

References

1. Abrahão, S., Insfrán, E., Sluÿters, A. & Vanderdonckt, J. Model-based intelligent user interface adaptation: challenges and future directions. *Software and System Modelling*. **20**, 1335-1349 (2021). https://doi.org/10.1007/s10270-021-00909-7
2. Akiki, P., Bandara, A. & Yu, Y. Adaptive Model-Driven User Interface Development Systems. *ACM Computing Surveys*. **47**, *9*, 1-33 (2014). https://doi.org/10.1145/2597999
3. Akiki, P., Bandara, A. & Yu, Y. Engineering Adaptive Model-Driven User Interfaces. *IEEE Trans. Software Eng.*. **42**, 1118-1147 (2016). https://doi.org/10.1109/TSE.2016.2553035
4. Alvarez-Cortes, V., Zarate, V., Ramirez Uresti, J. & Zayas, B. Current Challenges and Applications for Adaptive User Interfaces. *Human-Computer Interaction, Inaki Maurtua (Ed.), IntechOpen*. pp. 49-68. https://doi.org/10.5772/7745
5. Al Seraj, M., Pastel, R., & Al-Hasan, M. A Survey on User Modeling in HCI. *Computer Applications: an International Journal (CAIJ)*, **5**, *1*, 21-28 (February 2018). https://doi.org/10.5121/caij.2018.5102
6. Amouh, T., Gemo, M., Macq, B., Vanderdonckt, J., Wahed El Gariani, A., Reynaert, M.S., Stamatakis, L., & Thys, F. Versatile clinical information system design for emer-

gency departments. *IEEE Trans. Inf. Technol. Biomed.*. **9**, *2*. pp. 174-183 (2005), https://ieeexplore.ieee.org/document/1435415
7. Antoniac, P., Pulli, P., Kuroda, T., Bendas, D., Hickey, S. & Sasaki, H. Wireless User Perspectives in Europe: HandSmart Mediaphone Interface. *Wirel. Pers. Commun.*. **22**, 161-174 (2002). https://doi.org/10.1023/A:1019960305038
8. Aquino, N., Vanderdonckt, J., Condori-Fernández, N., Dieste Tubío & Pastor, O. Usability evaluation of multi-device/platform user interfaces generated by model-driven engineering. *Proceedings of the ACM International Symposium on Empirical Software Engineering and Measurement, ESEM 2010, 16-17 September 2010, Bolzano/Bozen, Italy*. (2010). https://doi.org/10.1145/1852786.1852826
9. Beirekdar, A., Keita, M., Noirhomme, M., Randolet, F., Vanderdonckt, J. & Mariage, C. Flexible Reporting for Automated Usability and Accessibility Evaluation of Web Sites. *Proceedings of IFIP TC13 Int. Conf. on Human-Computer Interaction, INTERACT 2005, Rome, Italy, September 12-16, 2005*, Lecture Notes in Computer Science, Vol. 3585. pp. 281-294 (2005). https://doi.org/10.1007/11555261_25
10. Beresford, A.R. & Stajano, F. Location privacy in pervasive computing. *IEEE Pervasive Computing*, **2**, *1*, 46-55 (2003). https://doi.org/10.1109/MPRV.2003.1186725
11. Blouin, A., Morin, B., Beaudoux, O., Nain, G., Albers, P. & Jézéquel, J. Combining Aspect-Oriented Modeling with Property-Based Reasoning to Improve User Interface Adaptation. *Proceedings of the 3rd ACM ACM Symposium on Engineering Interactive Computing Systems EICS '11*. pp. 85-94 (2011). https://doi.org/10.1145/1996461.1996500
12. Blumendorf, M., Lehmann, G. & Albayrak, S. Bridging models and systems at runtime to build adaptive user interfaces. *Proceedings of the 2nd ACM Symposium on Engineering Interactive Computing System, EICS 2010, Berlin, Germany, June 19-23, 2010*. pp. 9-18 (2010). https://doi.org/10.1145/1822018.1822022
13. Bouillon, L., Limbourg, Q., Vanderdonckt, J. & Michotte, B. Reverse engineering of Web pages based on derivations and transformations. *Proc. of Third Latin American Web Congress LA-Web '05*. pp. 11- (2005). https://doi.org/10.1109/LAWEB.2005.29
14. Bouzit, S., Calvary, G., Chêne, D. & Vanderdonckt, J. A Comparison of Shortcut and Step-by-Step Adaptive Menus for Smartphones. *Proceedings of the 30th International BCS Human Computer Interaction Conference, BCS HCI '16, Bournemouth University, Poole, UK, 11-15 July 2016, Electronic Workshops in Computing*. pp. 1-12 (2016). http://doi.org/10.14236/ewic/HCI2016.26
15. Bouzit, S., Calvary, G., Chêne, D. & Vanderdonckt, J. Polymodal Menus: A Model-based Approach for Designing Multimodal Adaptive Menus for Small Screens. *Proc. ACM Hum. Comput. Interact.*. **1**, *15*, 1-19 (2017). https://doi.org/10.1145/3099585
16. Bouzit, S., Calvary, G., Coutaz, J., Chêne, D., Petit, E. & Vanderdonckt, J. The PDA-LPA Design Space for User Interface Adaptation. *Proc. of the 11th IEEE Int. Conf. on Research Challenges in Information Science, RCIS '17*. pp. 353-364 (2017). https://doi.org/10.1109/RCIS.2017.7956559
17. Bouzit, S., Calvary, G., Chêne, D. & Vanderdonckt, J. Interface adaptivity by widget promotion/demotion. *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems, EICS '19, Valencia, Spain, June 18-21, 2019*. pp. 18:1-18:6 (2019). https://doi.org/10.1145/3319499.3328237
18. Browne, D., Totterdell, P. & Norman, M. Adaptive User Interfaces. Computers and People Series. Academic Press, London, UK (1990). https://shop.elsevier.com/books/adaptive-user-interfaces/browne/978-0-12-137755-7
19. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Souchon, N., Bouillon, L., Florins, M. & Vanderdonckt, J. Plasticity of User Interfaces: A Revised Reference Framework. *Proceedings of The First International Workshop on Task Models and Diagrams For User Interface Design*. pp. 127-134 (2002). https://doi.org/10.5555/646617.697235
20. 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**, 289-308 (2003). https://doi.org/10.1016/S0953-5438(03)00010-9

21. Calvary, G., Coutaz, J., Dâassi, O., Balme, L., & Demeure, A. Towards a New Generation of Widgets for Supporting Software Plasticity: The "Comet". *Proceedings of Joint Working Conferences on Engineering Human Computer Interaction and Interactive Systems, EHCI-DSVIS '04, Hamburg, Germany, July 11-13, 2004*. Lecture Notes in Computer Science, *3425*. pp. 306-324 (2004), https://doi.org/10.1007/11431879_21
22. Calvary, G. Plasticité des Interfaces Homme-Machine. *Habilitation à Diriger des Recherches – Spécialité Informatique*. Université Joseph Fourier (2007). http://iihm.imag.fr/publs/2007/HDR-calvary.pdf
23. Castillejo, E., Almeida, A., & López-de-Ipiña, D. Modelling users, context and devices for adaptive user interface systems. *International Journal of Pervasive Computing and Communications*, **10**, *1*, 69-91 (2014). https://doi.org/10.1108/IJPCC-09-2013-0028
24. Cockton, G. Some Critical Remarks on Abstractions for Adaptable Dialogue Managers. *Proceedings of 3rd Conference of the British Computer Society Human-Computer Interaction Specialist Group, People and Computers III, BCS-HCI '87, University of Exeter, UK, 7-11 September 1987*. pp. 325-343 (1987). https://researchportal.northumbria.ac.uk/en/publications/some-critical-remarks-on-abstractions-for-adaptable-dialogue-mana
25. Collignon, B., Vanderdonckt, J. & Calvary, G. An intelligent editor for multi-presentation user interfaces. *Proceedings of the ACM Symposium on Applied Computing, SAC 2008, Fortaleza, Ceara, Brazil, March 16-20, 2008*. pp. 1634-1641 (2008). https://doi.org/10.1145/1363686.1364072
26. Coutaz, J. Meta-User Interfaces for Ambient Spaces. *Proceedings of Int. Conf. on Task Models and Diagrams for User Interface Design, TAMODIA 2006, Lecture Notes in Computer Science, vol. 4385, Springer, Berlin*. pp. 1-15 (2007). https://doi.org/10.1007/978-3-540-70816-2_1
27. Crowley, J., Coutaz, J., Rey, G. & Reignier, P. Perceptual Components for Context Aware Computing. *Proceedings of UbiComp 2002: Ubiquitous Computing, Lecture Notes in Computer Science, vol. 2498, Springer, Berlin*. pp. 117-134 (2002). https://doi.org/10.1007/3-540-45809-3_9
28. Delgado, A., Estepa, A., Troyano, J. & Estepa, R. Reusing UI elements with Model-Based User Interface Development. *International Journal of Human-Computer Studies*. **86** pp. 48-62 (2016). https://www.sciencedirect.com/science/article/pii/S1071581915001470
29. Derakhshanmanesh, M., Ebert, J., Grieger, M. & Engels, G. Model-integrating Development of Software Systems: a Flexible Component-based Approach. *Software and Systems Modeling*. **18**, 2557-2586 (2019). https://link.springer.com/article/10.1007/s10270-018-0682-5
30. Dessart, C., Genaro Motti, V. & Vanderdonckt, J. Showing User Interface Adaptivity by Animated Transitions. *Proceedings of the 3rd ACM SIGCHI Symposium on Engineering Interactive Computing Systems EICS 2011*. pp. 95-104 (2011). http://doi.acm.org/10.1145/1996461.1996501
31. Dieterich, H., Malinowski, U., Kuhme, T. & Schneider-Hufschmidt, M. State of the art in adaptive user interfaces. *Adaptive User Interfaces Principles and Practice*. pp. 13-48 (1994). https://www.elsevier.com/books/adaptive-user-interfaces/schneider-hufschmidt/978-0-444-81545-3
32. Dubiel, M., Yilma, B., Latifzadeh, K. & Leiva, L. A Contextual Framework for Adaptive User Interfaces: Modelling the Interaction Environment. *CoRR*. **abs/2203.16882** (2022). https://doi.org/10.48550/arXiv.2203.16882
33. Dwork, C., & Roth, A. (2014). The Algorithmic Foundations of Differential Privacy. *Foundations and Trends in Theoretical Computer Science*, **9**, *3–4*, 211-407. http://dx.doi.org/10.1561/0400000042
34. Eisenstein, J., Vanderdonckt, J. & Puerta, A. Adapting to mobile contexts with user-interface modeling. *Proc. Of The 3rd IEEE Workshop On Mobile Computing Systems And Applications, IEEE Press*. pp. 83-92 (2000). https://doi.org/10.1109/MCSA.2000.895384

35. Eloi, D., Sahraoui, A., Vanderdonckt, J. & Leiva A., L. User-controlled Form Adaptation by Unsupervised Learning. *Adjunct Proceedings of the 2024 Nordic Conference on Human-Computer Interaction (NordiCHI Adjunct 2024), October 13–16, 2024, Uppsala, Sweden*. (2024). http://doi.acm.org/10.1145/3677045.3685431
36. Eslami, M., Firoozabadi, M. & Homayounvala, E. User Preferences for Adaptive User Interfaces in Health Information Systems. *Universal Access in Information Society*. **17**, 875-883 (2018), https://doi.org/10.1007/s10209-017-0569-1
37. Florins, M. & Vanderdonckt, J. Graceful degradation of user interfaces as a design method for multiplatform systems. *Proceedings of the 9th International Conference on Intelligent User Interfaces, IUI 2004, Funchal, Madeira, Portugal, January 13-16, 2004*. pp. 140-147 (2004). https://doi.org/10.1145/964442.964469
38. Florins, M., Montero Simarro, F., Vanderdonckt, J., & Michotte, B. Splitting rules for graceful degradation of user interfaces. *Proceedings of the ACM Conference on Advanced visual interfaces, AVI 2006, Venezia, Italy, May 23-26, 2006*. pp. 59-66 (2006). https://doi.org/10.1145/1133265.1133276
39. Frosini, L. & Paternò, F. User interface distribution in multi-device and multi-user environments with dynamically migrating engines. *Proceedings of ACM SIGCHI Symposium on Engineering Interactive Computing Systems, EICS'14, Rome, Italy, June 17-20, 2014*. pp. 55-64 (2014). https://doi.org/10.1145/2607023.2607032
40. Furtado, E., Furtado, V., Silva, W., Rodrigues, D., Silva Taddeo, L., Limbourg, Q. & Vanderdonckt, J. An Ontology-Based Method for Designing Multiple User Interfaces. *Proceedings of Int. Workshop on Multiple User Interfaces*. (2001). https://www.researchgate.net/publication/2567741_An_Ontology-Based_Method_for_Universal_Design_of_User_Interfaces
41. Gajos, K., Weld, D. & Wobbrock, J. Automatically generating personalized user interfaces with Supple. *Artif. Intell.*. **174**, 910-950 (2010), https://doi.org/10.1016/j.artint.2010.05.005
42. Gajos, K. & Chauncey, K. The Influence of Personality Traits and Cognitive Load on the Use of Adaptive User Interfaces. *Proceedings of the 22nd International Conference On Intelligent User Interfaces*. pp. 301-306 (2017). http://doi.acm.org/10.1145/3025171.3025192
43. Gaspar-Figueiredo, D., Abrahão, S., Insfrán, E. & Vanderdonckt, J. Measuring User Experience of Adaptive User Interfaces using EEG: A Replication Study. *Proceedings of The 27th International Conference on Evaluation and Assessment In Software Engineering*. pp. 52-61 (2023). https://doi.org/10.1145/3593434.3593452
44. Gaspar-Figueiredo, D., Fernández-Diego, M., Nuredini, R., Abrahão, S. & Insfrán, E. Reinforcement Learning-Based Framework for the Intelligent Adaptation of User Interfaces. *Companion Proceedings of the 16th ACM SIGCHI Symposium On Engineering Interactive Computing Systems, EICS Companion 2024, Cagliari, Italy, June 24-28, 2024*. pp. 40-48 (2024). https://doi.org/10.1145/3660515.3661329
45. Gomez-Uribe, C. A., & Hunt, N. The Netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems (TMIS)*. **6**, *4*, 1-19 (2015). https://doi.org/10.1145/2843948
46. Grolaux, D., Roy, P. & Vanderdonckt, J. Migratable User Interfaces: Beyond Migratory Interfaces. *Proceedings of the 1st Annual International Conference on Mobile And Ubiquitous Systems, MobiQuitous 2004, Networking And Services, 22-25 August 2004, Cambridge, MA, USA*. pp. 422-430 (2004). https://doi.org/10.1109/MOBIQ.2004.1331749
47. Gulliksen, J., Göransson, B., Boivie, I., Blomkvist, S., Persson, J. & Cajander, A. Key principles for user-centred systems design. *Behaviour and Information Technology, 22.6*. pp. 397-409 (2003).
48. Haddad, S., Latifzadeh, K., Duraisamy, S., Vanderdonckt, J., Daassi, O., Belghith, S. & Leiva, L. Good GUIs, Bad GUIs: Affective Evaluation of Graphical User Interfaces. *Proceedings of the 32nd ACM Conference on User Modeling, Adaptation and Personalization, UMAP 2024*. pp. 232-243 (2024). https://doi.org/10.1145/3627043.3659549
49. Hazard, C.J. & Singh, M.P. Privacy Risks in Intelligent User Interfaces. *IEEE Internet Computing*, **20**, *6*, 57-61 (2016). https://doi.org/10.1109/MIC.2016.116

50. Horvitz, E. Principles of Mixed-Initiative User Interfaces. *Proceeding of the ACM Int. Conf. on Human Factors in Computing Systems, CHI 1999*. pp. 159-166 (1999). http://doi.acm.org/10.1145/302979.303030
51. Hui, B., Partridge, G. & Boutilier, C. A Probabilistic Mental Model for Estimating Disruption. *Proceedings of the 14th ACM International Conference on Intelligent User Interfaces, IUI 2009*. pp. 287-296 (2009). https://doi.org/10.1145/1502650.1502691
52. Hussain, J., Ul Hassan, A., Bilal, H.S.M., Ali, R., Afzal, M., Hussain, S., Bang, J., Banos, O. & Lee, S. Model-based adaptive user interface based on context and user experience evaluation. *Journal on Multimodal User Interfaces*. **12**, 1-16 (2018). https://doi.org/10.1007/s12193-018-0258-2
53. ISO ISO/IEC 25010 - Software Quality Product Standard. International Standard Organization, Geneva (2019). https://iso25000.com/index.php/en/iso-25000-standards/iso-25010?limit=3%5C&limitstart=0
54. Jalil, N. Introduction to Intelligent User Interfaces (IUIs). In *Software Usability*, L. M. Castro, D. Cabrero, & R. Heimgärtner (Eds.). IntechOpen, Rijeka, Chapter 8, (2021). https://doi.org/10.5772/intechopen.97789
55. Jiang, Y., Lu, Y., Nichols, J., Stuerzlinger, W., Yu, C., Lutteroth, C., Li, Y., Kumar, R. & Li, T. Computational Approaches for Understanding, Generating, and Adapting User Interfaces. *Proceedings of ACM International Conference on Human Factors in Computing Systems - Extended Abstracts*. pp. 74:1-74:6 (2022). https://doi.org/10.1145/3491101.3504030
56. Josifovska, K., Yigitbas, E. & Engels, G. A Digital Twin-Based Multi-modal UI Adaptation Framework for Assistance Systems in Industry 4.0. *Human-Computer Interaction. Design Practice in Contemporary Societies*. pp. 398-409 (2019). https://link.springer.com/chapter/10.1007/978-3-030-22636-7_30
57. Kühme, T., Dieterich, H., Malinowski, U. & Schneider-Hufschmidt, M. Approaches to Adaptivity in User Interface Technology: Survey and Taxonomy. *Proceedings Of The IFIP TC2/WG2.7 Working Conference On Engineering For Human-Computer Interaction*. pp. 225-252 (1992). http://dl.acm.org/citation.cfm?id=647103.717564
58. Langerak, T., Christen, S., Albaba, M., Gebhardt, C., Holz, C. & Hilliges, O. MARLUI: Multi-Agent Reinforcement Learning for Adaptive Point-and-Click UIs. *Proc. ACM Hum.-Comput. Interact.*. **8**, *6* (2024). https://doi.org/10.1145/3661147
59. Lavie, T. & Meyer, J. Benefits and costs of adaptive user interfaces. *International Journal of Human-Computer Studies*. **68**, 508-524 (2010). http://www.sciencedirect.com/science/article/pii/S1071581910000145
60. Leiva, L. Restyling website design via touch-based interactions. *Proceedings of the 13th International Conference on Human-Computer Interaction with Mobile Devices and Services, MobileHCI 2011*. pp. 599-604 (2011). https://doi.org/10.1145/2037373.2037467
61. Leiva, L. Automatic web design refinements based on collective user behavior. *Proceedings of ACM International Conference on Human Factors in Computing Systems, CHI '12 Extended Abstracts*. pp. 1607-1612 (2012). https://doi.org/10.1145/2212776.2223680
62. Leiva, L. Responsive snippets: adaptive skim-reading for mobile devices. *Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct, MobileHCI 2018, Barcelona, Spain, September 03-06, 2018*. pp. 327-331 (2018). https://doi.org/10.1145/3236112.3236159
63. Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L. & Florins, M. UsiXML: A User Interface Description Language Supporting Multiple Levels of Independence. *Proceedings Of Workshops in Connection With the 4th International Conference On Web Engineering (ICWE '04). Engineering Advanced Web Applications*. pp. 325-338 (2004). https://www.pst.ifi.lmu.de/~baumeist/icwe/ws/ws4/DIWE2004-Limbourg.pdf
64. López-Jaquero, V., Vanderdonckt, J., Simarro, F. & González, P. Towards an Extended Model of User Interface Adaptation: The ISATINE Framework. *Proc. Of The Joint Working Conferences On Engineering Interactive Systems, EIS'07-EHCI'07-DSV-IS'07-HCSE'07, Salamanca, Spain, March 22-24, 2007, Lecture NOtes in Computer Science, Vol. 4940*, pp. 374-392 (2007). https://link.springer.com/chapter/10.1007/978-3-540-92698-6_23

65. López-Jaquero, V., Simarro, F. & González, P. AB-HCI: an interface multi-agent system to support human-centred computing. *IET Software*. **3**, 14-25 (2009). https://doi.org/10.1049/iet-sen:20070108
66. Mahdavinejad, M. S., Rezvan, M., Barekatain, M., Adibi, P., Barnaghi, P., & Sheth, A. P. Machine learning for internet of things data analysis: a survey. *Digital Communications and Networks*. **4**, *3*, 161-175 (2018). https://doi.org/10.1016/j.dcan.2017.10.002
67. Martinez-Ruiz, F., Arteaga, J., Vanderdonckt, J., González-Calleros, J. & González, R. A first draft of a Model-driven Method for Designing Graphical User Interfaces of Rich Internet Applications. *Proceedings of the Fourth Latin American Web Congress (LA-Web 2006), 25-27 October 2006, Cholula, Puebla, Mexico*. pp. 32-38 (2006). https://doi.org/10.1109/LA-WEB.2006.1
68. Melchior, J., Vanderdonckt, J. & Roy, P. A Comparative Evaluation of User Preferences for Extra-User Interfaces. *Int. J. Hum. Comput. Interact.*. **28**, 760-767 (2012). https://doi.org/10.1080/10447318.2012.715544
69. Mezhoudi, N. & Vanderdonckt, J. Toward a Task-driven Intelligent GUI Adaptation by Mixed-initiative. *Int. J. Hum. Comput. Interact.*. **37**, 445-458 (2021). https://doi.org/10.1080/10447318.2020.1824742
70. Miñón, R., Paternò, F. Arrue, M., & Abascal, J. Integrating adaptation rules for people with special needs in model-based UI development process. *Universal Access in the Information Society*. **15**, 1, 153-168 (2016). https://doi.org/10.1007/s10209-015-0406-3
71. Montero, F., López-Jaquero, V., Vanderdonckt, J., González, P., Lozano, M. & Limbourg, Q. Solving the Mapping Problem in User Interface Design by Seamless Integration in IdealXML. *Proc. of Int. Workshop on Design, Specification, and Verification of Interactive Systems, DSV-IS 2005*. Gilroy, S.W., Harrison, M.D. (Eds). Lecture Notes in Computer Science, Vol. 3941. Springer, Berlin, pp. 161-172 (2006). https://doi.org/10.1007/11752707_14
72. Motti, V. & Vanderdonckt, J. A computational framework for context-aware adaptation of user interfaces. *IEEE 7th International Conference On Research Challenges In Information Science, RCIS 2013, Paris, France, May 29-31, 2013*. pp. 1-12 (2013). https://doi.org/10.1109/RCIS.2013.6577709
73. Nichols, J. Using the Crowd to Understand and Adapt User Interfaces. *Proceedings of the 5th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. pp. 1-2 (2013), http://doi.acm.org/10.1145/2494603.2480344
74. Nivethika, M., Vithiya, I., Anntharshika, S. & Deegalla, S. Personalized and adaptive user interface framework for mobile application. *Proc. of Int. Conf. on Advances in Computing, Communications and Informatics*. pp. 1913-1918 (2013). https://doi.org/ICACCI.2013.6637474
75. Ontanon, S. & Zhu, J. The Personalization Paradox: the Conflict between Accurate User Models and Personalized Adaptive Systems. *Companion Proceedings of the 26th ACM International Conference on Intelligent User Interfaces, IUI '21*. pp. 64-66 (2021). https://doi.org/10.1145/3397482.3450734
76. Paternò, F., Santoro, C. & Spano, L. MARIA: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments. *ACM Trans. Comput. Hum. Interact.*. **16**, *19*, 1-30 (2009). https://doi.org/10.1145/1614390.1614394
77. Paternò, F., Santoro, C. & Spano, L. Engineering the authoring of usable service front ends. *Journal of Systems and Software*. **84**, *10*, 1806-1822 (2011). https://doi.org/10.1016/j.jss.2011.05.025
78. Paternò, F. User Interface Design Adaptation. Interaction Design Foundation - IxDF. https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/user-interface-design-adaptation
79. Peissner, M. & Edlin-White, R. User Control in Adaptive User Interfaces for Accessibility. *Proceedings of IFIP TC 13 Int. Conf. on Human-Computer Interaction, INTERACT '13*. Lecture Notes in Computer Science, Vol. 8117. pp. 623-640 (2013). https://doi.org/10.1007/978-3-642-40483-2_44

80. Puerta, A. The MECANO Project: Comprehensive and Integrated Support for Model-Based Interface Development. *Computer-Aided Design Of User Interfaces I, J. Vanderdonckt (Ed.), Proceedings of The Second International Workshop On Computer-Aided Design Of User Interfaces, CADUI 1996, June 5-7, 1996, Namur, Belgium*. pp. 19-36 (1996). http://www.usixml.org/en/puerta-a-the-mecano-project-comprehensive-and-integrated-support-for-model-based-interface-development.html?IDC=465&IDD=2565
81. Puerta, A. A Model-Based Interface Development Environment. *IEEE Software*. **14**, 40-47 (1997). https://doi.org/10.1109/52.595902
82. Raneburger, D., Popp, R. & Vanderdonckt, J. An automated layout approach for model-driven WIMP-UI generation. *Proceedings of ACM SIGCHI Symposium On Engineering Interactive Computing Systems, EICS'12, Copenhagen, Denmark, June 25-28, 2012*. pp. 91-100 (2012). https://doi.org/10.1145/2305484.2305501
83. Rogers, S., Fiechter, C. & Langley, P. An Adaptive Interactive Agent for Route Advice. *Proceedings of the Third Annual Conference On Autonomous Agents, AGENTS 1999, Seattle, WA, USA, May 1-5, 1999*. pp. 198-205 (1999). https://doi.org/10.1145/301136.301193
84. Schlee, M. & Vanderdonckt, J. Generative Programming of Graphical User Interfaces. *Proceedings of the Working Conference On Advanced Visual Interfaces, AVI 2004, Gallipoli, Italy, May 25-28, 2004*. pp. 403-406 (2004), https://doi.org/10.1145/989863.989936
85. Schmidt, A. Implicit Human Computer Interaction Through Context. *Pers. Ubiquitous Comput.*. **4**, 191-199 (2000). https://doi.org/10.1007/BF01324126
86. Serna, A., Calvary, G. & Scapin, D. How Assessing Plasticity Design Choices Can Improve UI Quality: A Case Study. *Proceeding of the second ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS 2010)*(2010).
87. Sluÿters, A., Vanderdonckt, J. & Vatavu, R. Engineering Slidable Graphical User Interfaces with Slime. *Proc. ACM Human-Computer Interaction*. **5**, *6* (2021). https://doi.org/10.1145/3457147
88. Sluÿters, A., Lambot, S., Vanderdonckt, J. & Villarreal-Narvaez, S. Analysis of User-Defined Radar-Based Hand Gestures Sensed Through Multiple Materials. *IEEE Access*. **12**, 27895-27917 (2024). https://doi.org/10.1109/ACCESS.2024.3366667
89. Sottet, J., Ganneau, V., Calvary, G., Coutaz, J., Demeure, A., Favre, J. & Demumieux, R. Model-Driven Adaptation for Plastic User Interfaces. *Human-Computer Interaction, Proc. of the 11th IFIP TC 13 International Conference, INTERACT 2007, Rio De Janeiro, Brazil, September 10-14, 2007*. Lecture NOtes in Computer Science, Vol. 4662. pp. 397-410 (2007). https://doi.org/10.1007/978-3-540-74796-3%5C_38
90. Sottet, J., Calvary, G., Coutaz, J. & Favre, J. A Model-Driven Engineering Approach for the Usability of Plastic User Interfaces. *Engineering Interactive Systems*. pp. 140-157 (2008). https://doi.org/10.1007/978-3-540-92698-6_9
91. Stocq, J. & Vanderdonckt, J. A domain model-driven approach for producing user interfaces to multi-platform information systems. *Proceedings of the Working Conference On Advanced Visual Interfaces, AVI 2004, Gallipoli, Italy, May 25-28, 2004*. pp. 395-398 (2004). https://doi.org/10.1145/989863.989934
92. Sousa, K., Filho, H., Vanderdonckt, J., Rogier, E. & Vandermeulen, J. User interface derivation from business processes: a model-driven approach for organizational engineering. *Proceedings of the 2008 ACM Symposium on Applied Computing, SAC 2008, Fortaleza, Ceara, Brazil, March 16-20, 2008*. pp. 553-560 (2008). https://doi.org/10.1145/1363686.1363821
93. Teachout, B.R. Gotta collect it all: Surveillance law lessons of Pokemon Go. *Stan. L. Rev. Online*, **69**, *83* (2016). https://www.stanfordlawreview.org/online/gotta-collect-it-all/
94. Teevan, J., Dumais, S., Liebling, D. & Hughes, R. Changing How People View Changes on the Web. *Proceedings of the 22nd Annual ACM Symposium On User Interface Software And Technology, UIST 2009*. pp. 237-246 (2009). https://doi.org/10.1145/1622176.1622221

95. Todi, K., Bailly, G., Leiva, L. & Oulasvirta, A. Adapting User Interfaces with Model-Based Reinforcement Learning. *Proceedings of the ACM Conference on Human Factors in Computing Systems, CHI 2021*. (2021). https://doi.org/10.1145/3411764.3445497
96. Vanderdonckt, J. & González-Calleros, J. Task-Driven Plasticity: One Step Forward with UbiDraw. *Engineering Interactive Systems, Proc. of Second Conference on Human-Centered Software Engineering, HCSE 2008, and 7th International Workshop on Task Models and Diagrams, TAMODIA 2008, Pisa, Italy, September 25-26, 200*. Lecture Notes in Computer Science, Vol. 5247. pp. 181-196 (2008). https://doi.org/10.1007/978-3-540-85992-5%5C_16
97. Vanderdonckt, J., Calvary, G., Coutaz, J. & Stanciulescu, A. Multimodality for Plastic User Interfaces: Models, Methods, and Principles. *Multimodal User Interfaces: From Signals to Interaction*. 61-84 (2008). https://doi.org/10.1007/978-3-540-78345-9_4
98. Vanderdonckt, J., Bouzit, S., Calvary, G. & Chêne, D. Cloud Menus: a Circular Adaptive Menu for Small Screens. *Proceedings Of The 23rd International Conference on Intelligent User Interfaces, IUI 2018, Tokyo, Japan, March 07-11, 2018*. pp. 317-328 (2018). https://doi.org/10.1145/3172944.3172975
99. Vanderdonckt, J., Bouzit, S., Calvary, G. & Chêne, D. Exploring a Design Space of Graphical Adaptive Menus: Normal vs. Small Screens. *ACM Trans. Interact. Intell. Syst.*. **10**, *7* (2019), https://doi.org/10.1145/3237190
100. Vatavu, R. Nomadic gestures: A technique for reusing gesture commands for frequent ambient interactions. *J. Ambient Intell. Smart Environ.*. **4**, 79-93 (2012), https://doi.org/10.3233/AIS-2012-0137
101. Velsen, L., Geest, T., Klaassen, R. & Steehouder, M. User-centered Evaluation of Adaptive and Adaptable Systems: a Literature Review. *Knowledge Engineering Review*. **23**, 261-281 (2008), https://www.cambridge.org/core/journals/knowledge-engineering-review/article/abs/usercentered-evaluation-of-adaptive-and-adaptable-systems-a-literature-review/C77A0D4AE8BAF5808E55214884245965
102. Yanagida, T., Vanderdonckt, J. & Burny, N. Adaptive GUI Layout by Satisfying Fuzzy Constraints. *Companion Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems, EICS 2023, Swansea, United Kingdom, June 27-30, 2023*. pp. 70-72 (2023). https://doi.org/10.1145/3596454.3597190
103. Yigitbas, E. & Sauer, S. Engineering Context-Adaptive UIs for Task-Continuous Cross-Channel Applications. *Human-Centered And Error-Resilient Systems Development, Proceedings of IFIP WG 13.2/13.5 Joint Working Conference HCSE 2016 and HESSD 2016 Stockholm, Sweden, August 29-31, 2016*. pp. 281-300 (2016), https://doi.org/10.1007/978-3-319-44902-9%5C_18
104. Yigitbas, E., Jovanovikj, I., Biermeier, K., Sauer, S. & Engels, G. Integrated model-driven development of self-adaptive user interfaces. *Softw. Syst. Model.*. **19**, 1057-1081 (2020), https://doi.org/10.1007/s10270-020-00777-7
105. Zouhaier, L., Hlaoui, Y. & Ayed, L. A Reinforcement Learning Based Approach of Context-driven Adaptive User Interfaces. *Proceedings Of IEEE 45th Annual Computers, Software, and Applications Conference*. pp. 1463-1468 (2021). https://ieeexplore.ieee.org/document/9529780
106. Zyskind, G., Nathan, O., & Pentland, A. Decentralizing privacy: Using blockchain to protect personal data. *Proceedings of IEEE Security and Privacy Workshops*. pp. 180-184 (2015). https://doi.org/10.1109/SPW.2015.27