Software and Systems Modeling (2021) 20:1335–1349
https://doi.org/10.1007/s10270-021-00909-7

EXPERT VOICE

Model-based intelligent user interface adaptation: challenges and future directions

Silvia Abrahão[1] · Emilio Insfran[1] · Arthur Sluÿters[2] · Jean Vanderdonckt[2]

Received: 21 May 2021 / Revised: 28 June 2021 / Accepted: 29 June 2021 / Published online: 16 July 2021

Abstract
Adapting the user interface of a software system to the requirements of the context of use continues to be a major challenge, particularly when users become more demanding in terms of adaptation quality. A considerable number of methods have, over the past three decades, provided some form of modelling with which to support user interface adaptation. There is, however, a crucial issue as regards in analysing the concepts, the underlying knowledge, and the user experience afforded by these methods as regards comparing their benefits and shortcomings. These methods are so numerous that positioning a new method in the state of the art is challenging. This paper, therefore, defines a conceptual reference framework for intelligent user interface adaptation containing a set of conceptual adaptation properties that are useful for model-based user interface adaptation. The objective of this set of properties is to understand any method, to compare various methods and to generate new ideas for adaptation. We also analyse the opportunities that machine learning techniques could provide for data processing and analysis in this context, and identify some open challenges in order to guarantee an appropriate user experience for end-users. The relevant literature and our experience in research and industrial collaboration have been used as the basis on which to propose future directions in which these challenges can be addressed.

Keywords Context of use · Intelligent user interface · Machine learning · Model-based software engineering · Model-driven engineering · User interface adaptation · Conceptual reference framework

1 Introduction

User interface (UI) adaptation consists of modifying a software system's UI in order to satisfy requirements, such as the needs, wishes, and preferences of a particular user or a group of users. Adaptation falls into two categories depending on which the system or end-user is responsible for making the adaptation [15]: *adaptability* refers to the end-user's ability to adapt the UI, whereas *adaptivity* or *self-adaptation* refers to the system's ability to perform UI adaptation. *Personalisation* is a particular form of adaptivity, usually for the UI contents, that is based on data originating solely from the end-user, such as personal traits [15]. When the data originate from sources that are external to the end-user, such as other user groups, *recommendation* occurs instead. *Mixed-initiative adaptation* [17] occurs when both the end-user and the system collaborate in order to make the adaptation.

UI adaptation should ultimately serve the end-user's benefit, by optimising factors contributing to the end-user's experience. For example, the objective of UI adaptation could be to increase efficiency (by reducing task completion time and error rate or by improving the learning curve), to ensure effectiveness (by guaranteeing full task completion) or to improve the subjective user's satisfaction, but could also be related to other factors, such as hedonic value or user disruption [18].

The challenge is to suggest the right adaptation at the right time in the right place in order to make it valuable for the end-user [4]. Otherwise, adaptation will be prone to limitations that could impede the expected benefits [21], if not thwart them: risk of misfit (the end-user's needs are incorrectly captured or interpreted), user cognitive disruption (the end-user is disrupted by the adaptation), lack of prediction (the end-user does not know when and how the adaptation will take

Communicated by Bernhard Rumpe.

✉ Silvia Abrahão
sabrahao@dsic.upv.es

[1] Universitat Politècnica de València, ES, Valencia, Spain

[2] Université catholique de Louvain, BE, Ottignies-Louvain-la-Neuve, Belgium

place), lack of explanation (the end-user is not informed of the reasons for adaptation), lack of user involvement (the end-user does not have the opportunity to participate actively in the adaptation process), and risks as regards privacy (the system maintains personal information that the user wishes to keep private).

A number of model-based approaches with which to address these challenges have been proposed to support UI adaptation by the human–computer interaction (HCI) and software engineering (SE) communities. However, no study that summarises the current knowledge, reasoning, and experience gained by these approaches, along with their benefits and limitations, currently exists. These aspects are so numerous that positioning any new approach with respect to the prior work is difficult to achieve. Surveys of UI adaptation [2,13,20] synthesise adaptation concepts, methods, and tools. Most of them are, however, technology driven, limited in scope, or largely surpassed by recent technical progress, which makes them incomplete as regards covering the most recent adaptation approaches or exploring alternatives in a structured manner.

In this paper, we, therefore, present a conceptual reference framework for model-based intelligent UI adaptation that contains a set of conceptual adaptation properties. These properties are structured around the Quintilian questions—what, why, how, to what, who, when, and where—posed for model-based UI adaptation. The objective of these conceptual properties is to facilitate the understanding and comparison of adaptation capabilities, in addition to their integration into the model-based or model-driven engineering of user interfaces of software systems, such as interactive applications, websites and desktop applications. These properties also help to identify open challenges and generate new ideas. In particular, progress in artificial intelligence (AI) and, more specifically, machine learning (ML), provides useful ways in which to support adaptation more effectively. We, therefore, analyse some opportunities that these fields may bring to model-based UI adaptation.

In Sect. 2, we present the current state of UI adaptation, while in Sect. 3, we define the conceptual framework for UI adaptation in order to locate the conceptual properties that support adaptation with respect to the Quintilian questions. These properties target the needs of two major stakeholder groups: they help system engineers to incorporate suitable UI adaptation mechanisms into model-based development more systematically, and they help practitioners to understand and compare UI adaptation methods. We conclude this paper in Sect. 4, with a call for action that includes a discussion of open challenges and future directions.

2 Current state of model-based UI adaptation

Pioneering work on UI adaptation started with Browne et al. [8], who used Moran's Command Language Grammar (CLG) to structure UI specifications into distinct aspects, ranging from tasks and abstract concepts to syntactic and physical components. These authors concluded that the major strength of CLG as regards UI adaptation is the principle of separation of concerns. Although this principle is enforced in CLG, it is not obvious how to easily propagate all specification aspects into the final code. These authors additionally state that CLG has very limited facilities with which to express UI presentation and behaviour.

Dieterich et al.'s taxonomy [13] has long been considered a seminal reference when classifying different types of adaptation configurations and methods. This taxonomy was obtained after analysing more than 200 papers, after which the UI adaptation methods found were structured in four stages: (1) the *initiative*, which specifies the entity, end-user or system that expresses the intention to perform adaptation; (2) the *proposal*, which suggests those proposals that could be applied to adaptation, given the current context of use; (3) the *decision*, which specifies those adaptation proposals that best fit the requirements imposed by the context of use, and (4) the *execution*, which is responsible for enacting the adaptation method previously decided on.

However, López-Jaquero et al. [23] identified some shortcomings of this taxonomy: it does not support an explicit collaboration between entities (i.e. the user and the system, or even a third party) and it is restricted to the execution only. These authors specialised Norman's theory of action in the ISATINE framework, which structures the UI adaptation into seven stages describing how the adaptation is carried out and by whom, thus addressing some of the Quintilian questions. The UI adaptation is understood to be a sequence of seven stages (Fig. 1, in which the user's parts are depicted in blue, while the system parts are depicted in green): (1) an entity obtained from UI adaptation goals that is formally expressed in the system or informally maintained in the end-user's head is established; (2) this entity takes the initiative in order to start a UI adaptation; (3) based on this input, some UI adaptation is subject to a specification so as to enable it to express how the adaptation will be conducted; (4) the UI adaptation selected is then applied; (5) a transition from an initial state before adaptation to a final state after adaptation is subsequently ensured in order to preserve continuity; (6) the results of this output are then subjected to interpretation by an entity based on the feedback provided by the system, and (7) the interpretation eventually leads to an evaluation of whether the initial goals established for the adaptation are (partially or totally) met. Depending on this evaluation, a new cycle could be initiated until the final goals are achieved.

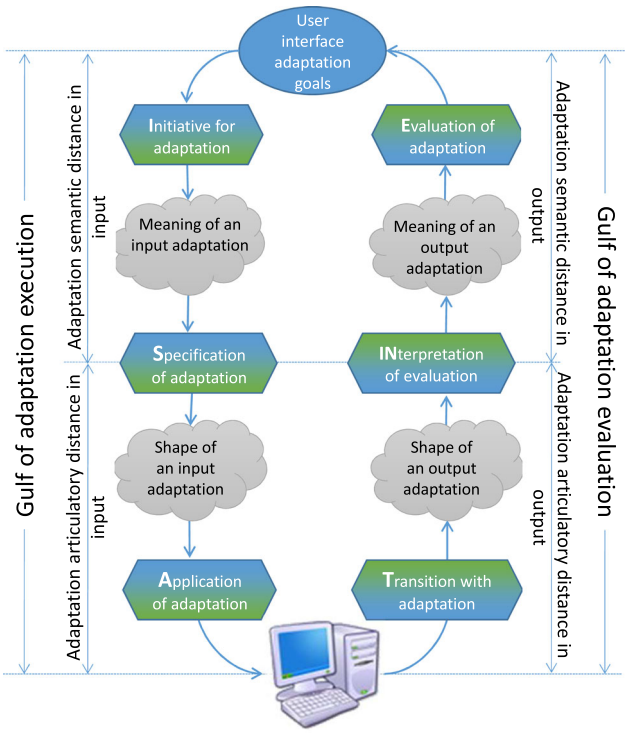


Fig. 1 The seven stages of the ISATINE framework

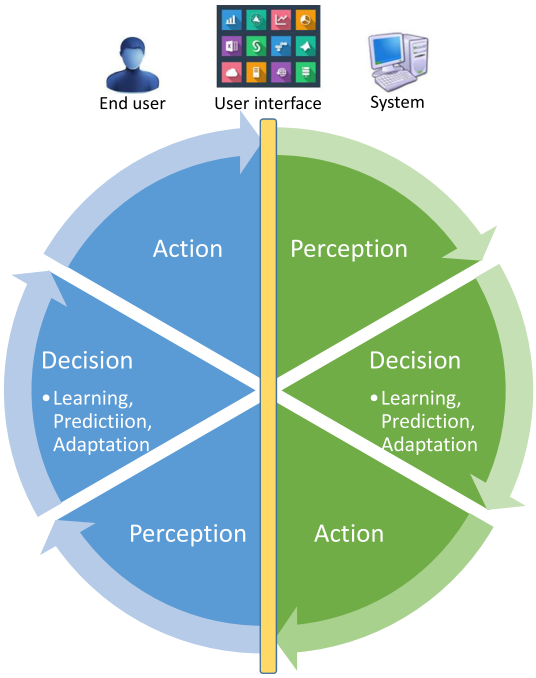


Fig. 2 User interface adaptation life cycle

Paramythis et al. [32] proposed a framework that can be used to guide a layered evaluation of adaptive interactive systems. This approach decomposes the system into layers (i.e. collect input data, interpret the collected data, model the current state of the "world", decide upon adaptation and apply adaptation) that can be evaluated independently using a set of formative methods. The authors then addressed the aforementioned Quintilian questions for web sites and hypermedia systems, but not for any type of UI.

The taxonomy proposed by McKinley et al. [25] addresses how software could be adapted by employing a composition in which algorithmic or structural parts of the system are exchanged for others in order to improve the system's fit to its current context of use. This adaptation is based on the separation of concerns into the functional behaviour of the system and cross-cutting concerns, and on computational reflections expressing different aspects of a system, component-based design practices that enable the development of the different parts of a system separately, and a middleware that usually provides the compositional capabilities. The taxonomy is structured in three dimensions: "how to adapt", which corresponds roughly to Dieterich's proposal stage, "where to adapt", which is implicitly included in the execution stage, and "when to adapt", which was not originally covered, signifying that McKinley's taxonomy complements Dieterich's taxonomy.

Several surveys on UI adaptation have been published in order to synthesise adaptation concepts, methods and tools. For example, Van Velsen et al. [41] presented a systematic literature review on the user-centred evaluation of adaptive and adaptable systems. Akiki et al. [2] presented a qualitative study analysing adaptive UI development systems that employed a model-driven engineering approach for UI adaptation. Motti et al. [28] conducted a survey investigating whether and how practices concerning model-based context-aware adaptation are perceived and adopted by practitioners. While most stakeholders recognise the relevance and benefits of the adaptation methods, they are still not considered or partially adopted during software development.

Finally, Fig. 2 depicts the PDA-LDA cycle employed to structure the UI adaptation according to the theory of control perspective [7]: each entity, the end-user (depicted in blue) or the system (depicted in green) enters a cycle of three stages: the *perception* (P) of the context before adaptation, the *decision* (D) to adapt and the *action* (A) taken in order to adapt. Unlike other frameworks that emphasise the adaptation steps, this cycle acknowledges that both the end-user and the system act symmetrically with these stages, which should be covered to some extent.

UI adaptation is, therefore, treated independently of any implementation method, which is desirable. However, in the context of model-based/driven engineering as a particular method for adaptation, there are few or no explicit recommendations on how to structure software components in order to support intelligent UI adaptation. The existence of some

models, such as the user model, is frequently mentioned, but their structure and usage are not made sufficiently explicit to help modellers and developers to implement adaptation [1].

On the one hand, UI adaptation methods have been investigated in human–computer interaction (HCI), but without making the means required to practically implement them sufficiently explicit.

On the other, the software engineering (SE) community has advanced as regards principles and technologies with which to support the PDA-LDA cycle on the system side (e.g. the MAPE-K adaptation loop, models at runtime), but often relegates the perception, decision and action stages on the end-user side, typically addressed in HCI, to a secondary role.

3 Conceptual framework and properties for UI adaptation

In this section, we introduce a conceptual reference framework with intelligent model-based UI adaptation. The purpose of the framework is twofold: (1) to help software engineers to properly decompose the application into layers and modules that are suitable for supporting model-based UI adaptation, and (2) to provide a property-based classification of existing approaches in order to identify some trends and research directions.

3.1 The conceptual reference framework

Figure 3 depicts our conceptual framework with which to support intelligent UI adaptation based on MDE principles (e.g. abstraction, separation of concerns, automation) and technologies (e.g. modelling and metamodelling, model transformation). This framework is decomposed into four parts:

- The *context of use*, which represents the actor(s) that interact with their platform or device in any physical environment [9]. For example, a business person interacting with a smartphone in a busy airport represents a context of use that is radically different from a tourist browsing a laptop in a hotel. In order to support context-aware UI adaptation, a *context probe* (e.g. a camera, a sensor) senses it and abstracts relevant data into useful model fragments corresponding to the context of use: a *user model* captures all data pertaining to the end-user (e.g. gender, age, interaction history, abilities, preferences, emotional state, past experience), a *platform model* captures data that are useful as regards characterising the target platform (e.g. screen resolution, sizes, interaction capabilities, CPU availability), and an *environment model* captures any environmental data that could influence the UI (e.g. location, stationary vs. mobile conditions, light, noise level, physical configuration, organisational and psycho-social constraints). For example, Figs. 4 and 5 reproduce two UIs of a trip planner in two different contexts of use.
- The *software system*, which is usually an interactive application, consists of the semantic core component that contains the business logic functions appertaining to the application domain. These functions are executed from the intelligent UI of this software, which could exploit up to four models [9]: a "task and domain" model captures domain abstractions, usually in the form of an object-oriented diagram or a UML class diagram or any kind of domain model, while the "task" model captures how the end-user sees the interaction with the semantic core, independently of any implementation or technology.[1] The task and domain models are transformed into an "abstract UI" model[2] for a given context of use, but still without making any assumptions about the interaction modality and target platform. The "abstract UI" becomes a "concrete UI" when design options are decided for a particular modality, e.g. graphical, vocal, tactile, haptic or tangible, and for a particular platform, e.g. a smartphone, a tablet, a wall screen. A final UI is obtained from this concrete UI model by means of either model interpretation or model-to-code transformation. Any transformation between two levels can exploit the contents of the context model. Examples of these are: a forward engineering transformation (depicted as a downwards arrow in the software system), a reverse engineering transformation (depicted as an upwards arrow), or a self-modification (depicted as a loop), all of which can generate a subsequent model based on the previous one by exploiting the context model.
- The *Intelligent UI adaptor*, which consists of six components, two of which are mandatory. At the core is the *adaptation manager*, which is responsible for performing any complete adaptation process from its initiation to its completion, such as according to the ISATINE framework. This manager, therefore, stores and maintains *adaptation parameters*, which regulate the adaptation process with variable parameters, such as the level of automation, the frequency, the priority of adaptation rules or the preferred adaptation strategies. Adaptation parameters can be application-independent, such as the level of automation, or application-dependent, such as those shown in Figs. 6 and 7. The adaptation manager has its own UI, denominated as an *adaptation manager UI*,

[1] See W3C recommendation for task model at https://www.w3.org/TR/task-models/.

[2] See W3C recommendation for abstract UI model at https://www.w3.org/TR/abstract-ui/.

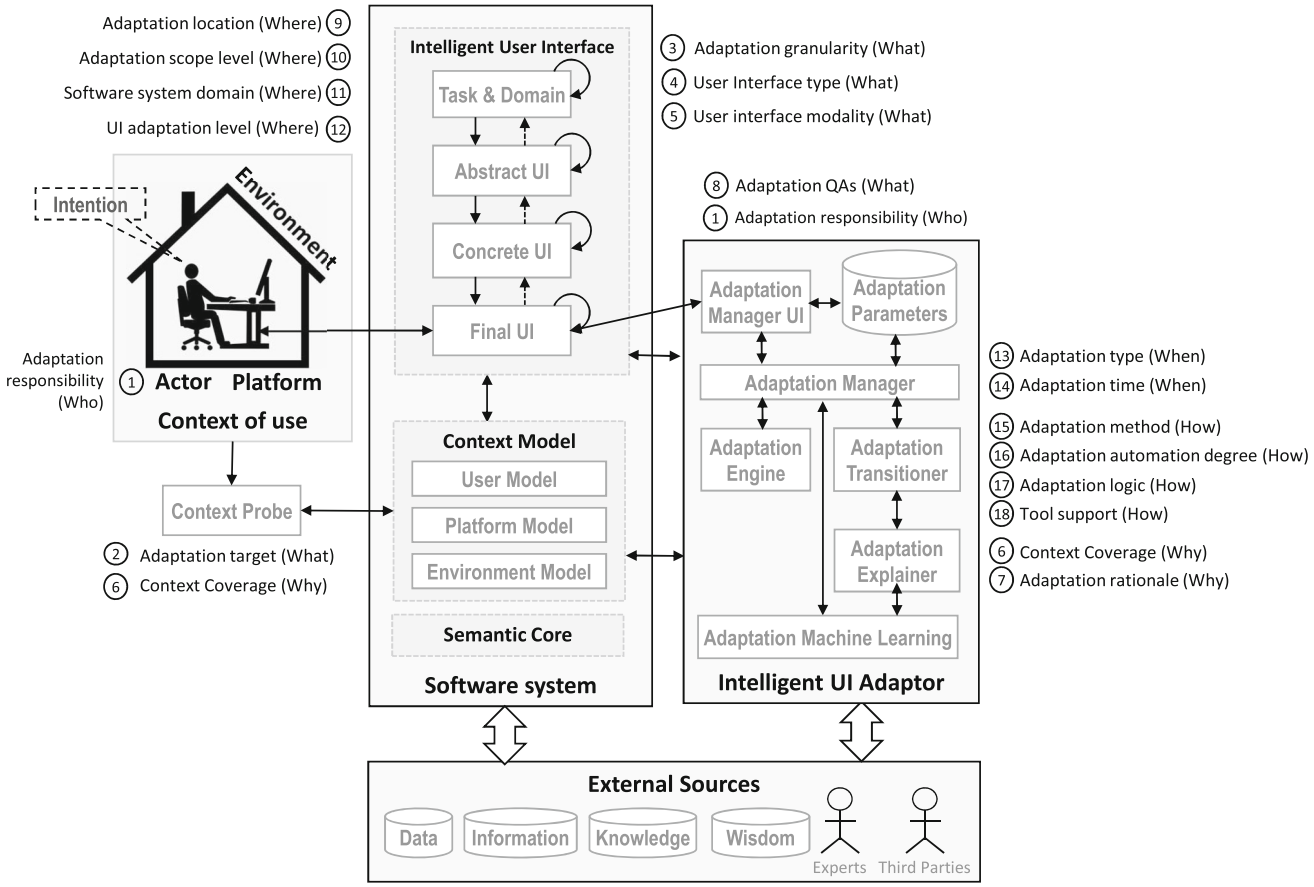


Fig. 3 Conceptual reference framework for intelligent UI adaptation highlighting four main components and their related conceptual adaptation properties

which is sometimes referred to as a meta-UI (the UI above the original UI [11]) or extra-UI (the UI external to the UI of the application [26]). This UI enables the end-user to access and update the adaptation parameters and to conduct the whole adaptation process interactively so as to specifically perform adaptation operations, review them, accept them or reject them. In order to clearly differentiate the UI of the adaptation manager from that of the software system, it should be located in a separate location. Depending on the parameters, the adaptation manager executes the adaptation logic contained in the *adaptation engine*, which is usually implemented in the form of adaptation rules. The adaptation manager can call the *adaptation transitioner* in order to convey to the end-user the transition between the status before and after adaptation. For example, animated transitions [12] apply morphing techniques to show this step and to preserve continuity (Fig. 8). If necessary, the transitioner provides the end-user with information on why, how and when the adaptation is performed by requesting the *adaptation explainer*, which is responsible for explaining and justifying why any adaptation proposal or step will be executed [16]. Finally, an adaptation machine learning system can monitor the whole process over time, learn what the good adaptations are or which are preferred by the end-user, and recommend them in the future [7]. For example, TADAP [27] suggests adaptation operations based on the user's interaction history that the end-user can accept, reject, or re-parameterise by employing Hidden Markov Chains.

– The *external sources* contain any form of information that can be exploited in order to support and improve the adaptation process: data concerning individual items, information for semantically related data, knowledge gained from exploiting the information within a certain domain, and wisdom when knowledge can be reproduced in different domains. These sources are typically held by agents that are external to the software system, such as experts, brokers, recommenders, or any third party source. For example, when no adaptation proposal can be obtained, an external source may be required in order to attain one.

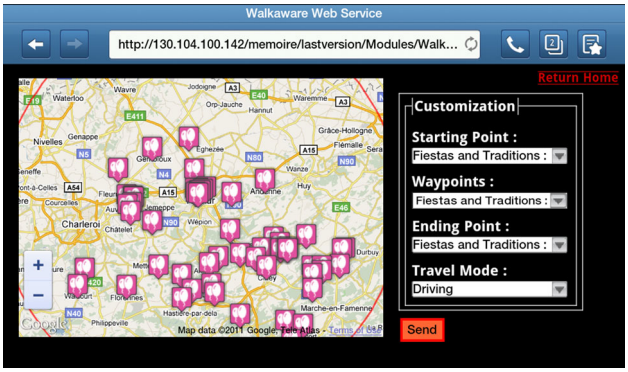


Fig. 4 UI adapted to a first context of use: a tourist equipped with a tablet in dark conditions

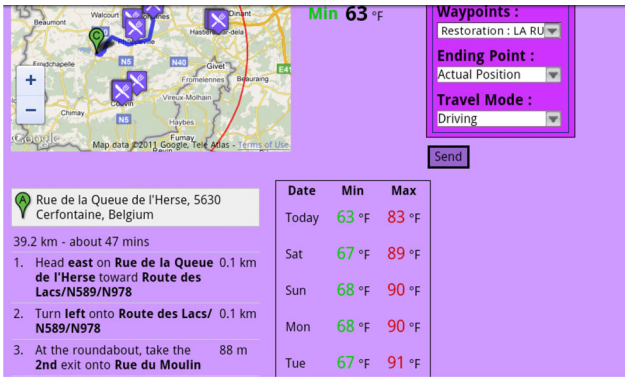


Fig. 5 UI adapted to a second context of use: a car driver using a car-embedded browser during daylight to obtain a weather forecast

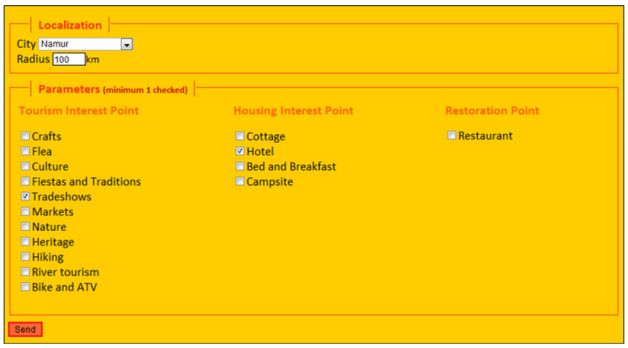


Fig. 6 Cultural adaptation parameters

3.2 The conceptual properties for UI adaptation

The framework enables the location and definition of conceptual properties for model-based UI adaptation. Some properties were taken from existing frameworks, and others are based on our own experience of the topic. These properties can be structured according to the Quintilian questions [28,32]:

Fig. 7 Weather forecast adaptation parameters

- **Who**: which actor triggers, initiates or is in charge of each adaptation step? The end-user, designer, developer, system, UI, an expert, any third party or an external source?
- **What**: what is adapted in the UI? The presentation, navigation behaviour or contents?
- **Why**: what are the main adaptation goals, expressed as external software quality attributes (e.g. usability, performance) or internal software quality attributes (e.g. predictability, learnability [7])?
- **How**: what development methods, techniques, and strategies are exploited to drive the adaptation?
- **To what**: what contextual information is used to justify and define the adaptation? For example, the application resources subject to adaptation based mainly on user, platform or environment?
- **When**: in what state does adaptation occurs?: design time, linking time, compilation time, run time, or a combination of them?
- **Where**: where in the interactive application does the adaptation take place, based on the software architecture?: the client, a proxy, or a server?

3.2.1 Who

This dimension refers to the actor that is responsible for carrying out the adaptation process from beginning to end. This actor could be the end-user, the software system itself (and by metonymy, its UI), or a third party, such as an external broker, the designer, the developer, any external stakeholder, or the crowd [29]. This dimension is defined by the following property:

① **Adaptation responsibility**, which can be shared between different actors and may vary depending on the adaptation steps that are carried out. According to the ISATINE framework [23], the adaptation steps followed by a particular approach can be grouped according to their impact on the execution and evaluation degree of the approach. The *execu-*

Fig. 8 Animated transition from the initial state before adaptation to the final state after adaptation [12]

tion degree can be assessed by analysing who performs the following three steps:

- *Initiative*, which refers to who detects that there is a need to adapt the user interface. The adaptation process can usually be initiated by the user (U), the system (S) or a third party (T). For instance, the user can trigger an adaptation by selecting an element in the user interface or the system can decide that an adaptation is needed by inferring it from a change in the context of use.
- *Decision*, which refers to who makes the decision to adapt the UI (i.e. the user, the system, or third party). The decision is concerned with the identification of what adaptation proposals best fit the need for the adaptation detected.
- *Application*, which refers to who is responsible for applying the adaptation, i.e. U, S, or T, or any combination.

The *evaluation degree* can be assessed by analysing who performs the following three steps:

- *Transition*, which refers to how the transition is performed from the original UI to that which is adapted. This criterion indicates whether the end-user is able to perceive how the adaptation is conducted, i.e. whether the user is aware of the intermediate steps taken when adapting the user interface.
- *Interpretation*, which refers to the user's ability to understand both the adaptation results and the adaptation execution itself.
- *User feedback*, which refers to the ability of the approach to provide feedback about the quality of the adaptation.

3.2.2 What

This dimension refers to what is adapted, which is further characterised through the use of five conceptual properties:

② **Adaptation target**, which refers to which UI part is subject to adaptation: its presentation (layout), its dynamic behaviour, its navigation, its contents (e.g. text, images, videos), or any combination. For example, after identifying the end-user, Diffie [37], highlights parts of a website that have changed since the last visit.

③ **Adaptation granularity**, which refers to the smallest UI unit that is subject to adaptation. The adaptation unit could cover the UI presentation, the dialogue, the navigational flow or the contents. The following units may be subject to adaptation:

- *Within widgets*: the adaptation is applicable within an interaction object. For example, a list box is replaced with a drop-down list (Fig. 8).
- *Across widgets*, within a group: the adaptation is applicable to widgets within the same group.
- *Across groups within a container*: the adaptation is applicable to groups of widgets within the same container.
- *Across containers, within a software system*: the adaptation is applicable to all groups within a software system, e.g. a single application.
- *Across software systems*: the adaptation is applicable to software systems. For example, a particular adaptation is always applied to all applications used by a person.

A model-based approach may support the adaptation of one or more UI units. For example, Sottet et al. [36] support adaptations across different interaction objects (widgets), across groups within a container, and across containers, within a software system.

④ **UI Type**, which refers to the UI type that is subject to adaptation depending on its interaction modality:

- *Graphical*: concerns only the graphical part.
- *Vocal*: concerns only the vocal part.
- *Tactile*: concerns only the tactile part.
- *Gestural*: concerns only the gestural part.
- *Haptic*: concerns only the haptic modality.

For example, a rich internet UI is rendered as a vectorial graphical interface [24]. Nomadic gestures [40] adapt command gestures for a particular user that are transferable from one software system to another.

⑤ **UI Modality**, which expresses how many modalities are incorporated into the UI adaptation, as follows:

- *Monomodal*: the approach supports only the adaptation of a single UI type.

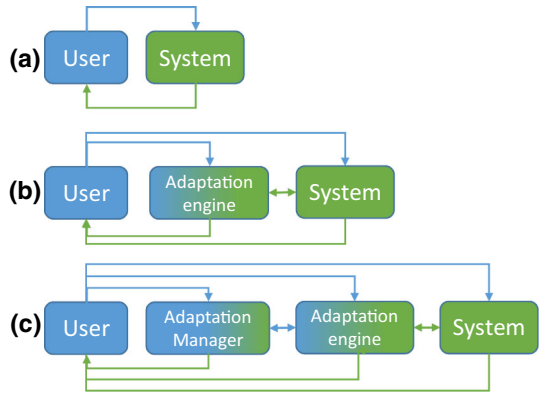


Fig. 9 Feedback loops in adaptation: **a** one with the system without adaptation, **b** two with an adaptation engine, **c** three with an adaptation manager

– *Bimodal*: the approach supports only the adaptation of two UI types together.
– *Multimodal*: the approach supports the adaptation of several UI types combined.

⑥**Context Coverage**, which expresses which part of the context model is exploited for UI adaptation: the user model (e.g. user profile, preferences, goals, tasks, emotional state, physical state), the platform model (e.g. screen resolution, browser, battery) and/or the environment model (e.g. location, noise, light). For example, a business traveller who rents a car via a smartphone in a noisy airport is considered as one context of use, and a tourist who book a car on a laptop while sitting on a sofa at home is considered as another context of use. Figure 4 covers the three models: the user who is a tourist, the platform detected as a tablet and the environment detecting dark conditions. Any variation in the model involved in the context model can initiate a contextual variation that will or will not be reflected via a UI adaptation. A small contextual variation could be considered not sufficiently significant enough to trigger a UI adaptation, and it is not always desirable or advisable to perform such an adaptation for every slight contextual perturbation.

3.2.3 Why

This dimension is concerned with justifying the reasons why a UI adaptation is carried out. This depends on the user's goals and is defined by two properties:

⑦ **Adaptation rationale**, which refers to the reason why the adaptation is required and, more specifically what the new requirements that need to be satisfied through the adaptation are. For example, an end-user expressing a preference for data selection rather than data input will see some sort of UI adaptation based on this preference.

⑧ **Adaptation QAs**, which refer to the quality attribute(s) that should be impacted by the UI adaptation process. For example, the ISO/IEC 25010 standard for software product quality [19] can be used as a reference to specify the quality attributes to be guaranteed or improved by any UI adaptation, such as usability, UI aesthetics, flexibility and portability. Since the UI adaptation is, in principle, performed for the ultimate benefit of the end-user, and not necessarily the software system, quality attributes such as accessibility and continuity are often oriented towards the end-user. UI plasticity [10] also represents a frequent quality attribute, as it expresses the ability of a UI to adapt itself depending on contextual variations while preserving usability.

3.2.4 Where

This dimension is concerned with where the adaptation takes place and is defined by four properties:

⑨ **Adaptation location**, which refers to the physical location of the intelligent UI adaptor (Fig. 3) in the overall architecture of the software system as follows:

– *Client-side*: when located inside in the software.
– *Server-side*: when located outside the software system, which is typically the case in cloud computing.
– *Proxy-side*: when encapsulated in a proxy component to ensure some independence. For example, Fig. 5 depicts a UI in which weather forecasts were retrieved from a web service in XML and fed back into a proxy to decide how to present these data based on the adaptation parameters specified in Fig. 7. Locating this adaptation strategy inside the software would create a certain dependence between the UI and the web service.

The adaptation location directly influences how feedback loops are introduced into the components (Fig. 9). A software system devoid of adaptation (Fig. 9a) benefits from retroactive feedback only between the software system and the end-user. A software system with an adaptation engine (Fig. 9b) has two feedback loops: between the user and the adaptation engine and between the user and the system.

Finally, three feedback loops are possible for an intelligent UI (Fig. 9c), which require the system to be decoupled from its intelligent UI. Current research efforts in the SE community [42] [3] are focused on providing strategies and facilities with which to support UI adaptation based on one control loop with an adaptation engine, and there is a shortage of approaches that support more intelligent strategies based on two control loops with an adaptation manager.

⑩ **Adaptation scope level**, which refers to the level at which the adaptation process occurs, which is based on the three levels proposed by Nierstrasz and Meijler [30]:

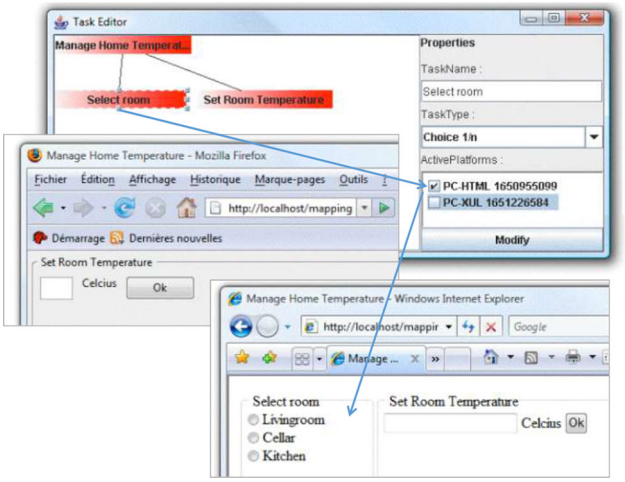


Fig. 10 An adaptation manager with which to distribute tasks to platforms [36]

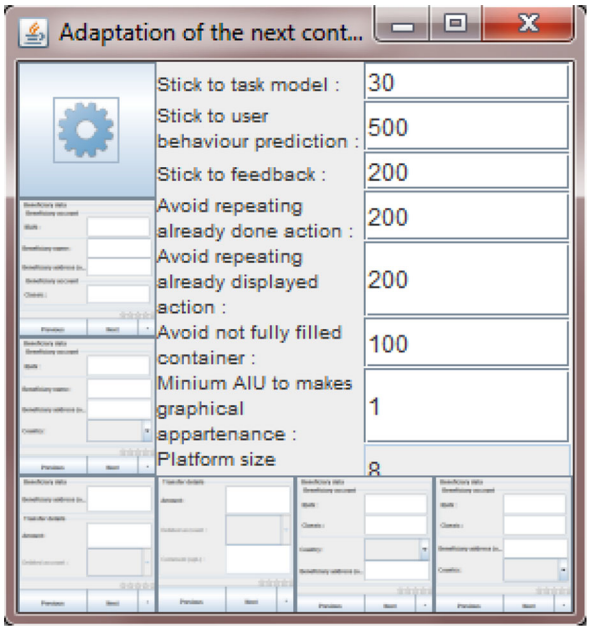


Fig. 11 Adaptation parameters for the task model

- *Framework level*, when the process occurs at the level of a generic software architecture together with a set of generic software components that may be used to create specific software architectures. For example, Nivethika et al. [31] developed a UI adaptive framework by exploiting an inference engine with the purpose of adapting a UI based on user actions in one application to be propagated to other applications.
- *Class level*, when the process occurs at the level of the components belonging to specific software systems or provided by frameworks for this purpose. For example, Yigitbas et al. [42] presented a model-driven approach for self-adaptive UIs that is applied at the model level of a particular UI. It supports the specification of an adaptation model that contains abstract UI adaptation rules in alignment with the IFML abstract UI modelling language.
- *Instance level*, when the process occurs at the level of a running software system. For example, Akiki et al. [3] presented a model-driven UI adaptation approach that is applied only at the instance level of a particular UI.

⑪ **UI Adaptation level**, which refers to the abstraction level defined in the Cameleon Reference Framework (CRF) [9] at which the adaptation occurs as represented in the intelligent UI (Fig. 3): "task and concepts", "abstract UI", "concrete UI", or "final UI", or several levels simultaneously. CRF is a unified UI reference framework with which to develop multi-target UIs that is particularly suitable for an MDE approach [14]:

- *Task/domain model*: when a task model and/or a domain model are exploited in order to perform UI adaptation. For example, TADAP [27] maintains a task model of the end-user's activity and suggests a final UI adaptation depending on its parameters (Fig. 11). UbiDraw [39] consists of a vectorial drawing application that adapts its UI by displaying, undisplaying, resizing, and relocating tool bars and icons according to the current user's task, task frequency, criticality, importance, or the user's preference for a particular task. The domain model denotes the application's universe of discourse and can typically be represented using a UML class diagram. Figure 10 depicts a UI of the adaptation manager enabling the end-user to distribute tasks (represented in a task model) to various platforms (represented in platform models), such as an HTML UI to one browser and another XUL UI to another browser with another rendering.
- *Abstract user interface model*: this specifies the user's interactions with the UI without making any reference to any specific technology (i.e. modality). This model is typically represented with a User Interface Description Language (UIDL).
- *Concrete user interface model*: this specifies the user's interactions with the UI with explicit reference to a specific technology, e.g. a graphical UI for a website or a vocal UI on a smartphone. For example, ReversiXML [6] reverse engineers the HTML code of a web page into a concrete UI model that is then derived for another platform.
- *Final user interface model*: this represents the actual UI produced by any rendering engine, i.e. by interpretation or by model-to-code generation.

⑫ **Adaptation Domain**, which refers to the domain of human activity in which the adaptation takes place. A model-based UI adaptation approach can be general purpose (independent of the application domain) or devised for a specific domain (e.g. smart home, Internet-of-things, ambient assisted living, smart cities, ERP system). We believe that the application domain may influence the *adaptation rationale* or the *adaptation QAs* that should be ensured by a particular approach. For example, the objective of Akiki et al.'s approach [3] is to improve the UI usability of enterprise applications, such as ERP systems, by providing end-users with a minimal feature-set and an optimal layout.

3.2.5 When

This dimension is concerned with when the adaptation takes place. This decision is not trivial, since the frequency of adaptation affects the system usability. It is defined by two properties:

⑬ **Adaptation type**. The UI adaptation type is said to be *static* when its process takes place during design (e.g. prototyping, sketching), development (e.g. compile), link or load time, *dynamic*, when its process takes place during runtime, or *hybrid* when both are combined. For example, in the Yigitbas et al. approach [42] a rule-based execution environment supports the UI adaptation at runtime.

⑭ **Adaptation time**, which refers to the exact moment of time at which the UI adaptation occurs, which could be at one specific moment (*single-step*) or distributed throughout several moments of time (*multi-step*). In order to further characterise this conceptual property, we rely on the adaptation dimensions proposed by McKinley et al. [25], which result from a survey of adaptive systems. It is said to be *hardwired* (when the UI adaptation is embedded in the code of the software application, typically the UI code), *customisable* (when the UI adaptation enables some degree of pre-computed freedom), *configurable* (when the UI adaptation technique could be configured before executing it), *tunable* (when the UI adaptation technique could fine-tune the UI at run-time without modifying its code), or *mutable* (when the UI adaptation technique subsumes the run-time code modification of the software system, namely the UI code). McKinley et al. [25] mention that hardwired, customisable, and configurable cases are static, while tunable and mutable cases are, by definition, dynamic. For example, MiniAba [34] uses generative programming to automatically regenerate a new C++ project from dynamic specifications, which are thus dynamic and mutable.

3.2.6 How

This dimension is concerned with how the UI adaptation is performed. One critical issue is to what extent the software system can access the various aforementioned models to optimise the UI adaptation and to exploit them. It is characterised by five properties:

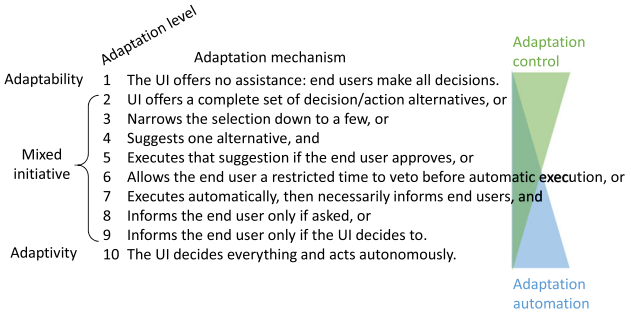


Fig. 12 Adaptation automation levels (AAL)

⑮ **Adaptation method**, which refers to the software engineering method used to adapt the UI. An adaptation method can be model-based/driven or it can be combined with other methods, such as aspect-oriented modelling, component-based design, computation reflection (i.e. a programme's ability to reason about, and possibly alter, its own behaviour), dynamic interconnection, higher-order functional composition, higher-order modelling, macro-command expansion, mashup, modelling or programming by example, syntactical expansion of parameterised component. This paves the way towards investigating the effectiveness of combining these techniques with the purpose of improving the existing model-based UI adaptation approaches. For example, Blouin et al. [5] presented an approach that combines aspect-oriented modelling with property-based reasoning to control complex and dynamic user interface adaptations. The encapsulation of variable parts of interactive systems into aspects permits the dynamic adaptation of user interfaces, and the tagging of UI components and context models with QoS properties allows the reasoner to select the aspects best suited to the current context.

⑯ **Adaptation automation degree**, which refers to the level to which the UI adaptation is automated. There is a wide range of possible adaptation levels between adaptability (when UI adaptation is performed entirely manually by the end-user) and adaptivity (when UI adaptation is performed entirely by the system), which we defined as follows based on [33] (see Fig. 12):

- *Level 1. Adaptability (fully manual)*: the UI adaptation is performed entirely by the end-user.
- *Level 2. Proposability*: the intelligent UI manager proposes certain decisions that should be made in order to execute actions towards UI adaptation to be performed by the system and the end-user decides.
- *Level 3. Narrowing*: the intelligent UI manager sorts the proposed decisions according to certain criteria to

facilitate the end-users' decision. For example, Fig. 11 proposes a suite of six new layouts in decreasing order of performance based on past user actions and parameters.
- *Level 4. Identification*: the intelligent UI manager identifies the best decision for the user to make from among all the proposals.
- *Level 5. Execution*: the intelligent UI manager executes the decision made by the end-user. For example, the end-user selects one of the new layouts presented in Fig. 11 to replace the existing one.
- *Level 6. Restriction*: the intelligent UI manager postpones the UI adaptation for a certain amount of time. If the end-user does not react, the UI adaptation will be processed as suggested. Otherwise, the end-user should use the adaptation manager UI to specify which actions to take.
- *Level 7. Information*: the intelligent UI manager performs the UI adaptation and triggers the adaptation transitioner and/or explainer in order to inform the end-user of this decision.
- *Level 8. On-demand*: the intelligent UI manager performs the UI adaptation and triggers the adaptation transitioner and/or explainer only if the end-user demands it.
- *Level 9. Self-explanation*: the intelligent UI manager performs the UI adaptation and triggers the adaptation transitioner and/or explainer when it decides to do so.
- *Level 10. Adaptivity/self-adaptation*: the intelligent UI manager performs the UI adaptation entirely automatically without any user intervention.

Levels 2 to 9 represent various cases of mixed-initiative adaptation. While these levels cover a wide range of automation levels, they mainly relegate the end-user to a secondary role of decision maker. These levels should, therefore, be accompanied by appropriate actions that the end-user should take within the adaptation manager UI, which should offer more high-level actions to support UI adaptation. These levels are cumulative, thus requiring a sophisticated adaptation manager.

To be more practical, we suggest distributing the mixed initiative between the end-user, the system, and any third party according to the seven stages of adaptation (Fig. 1): goal, initiative, specification, application, transition, interpretation, and evaluation. For example, AB-HCI [22] supports a mixed initiative for the three steps belonging to the gulf of execution, i.e. from initiative to application, but not the subsequent stages belonging to the gulf of evaluation. Each stage is managed through a particular agent in a multi-agent architecture which adequately distributes responsibilities.

An alternate characterisation of the adaptation automation degree could balance the UI adaptation with equal responsibility (when the UI adaptation is performed equally by the end-user and the adaptation manager), with more user involvement (when the UI adaptation is mostly performed by the user) or less involvement (when the UI adaptation is mostly performed by the adaptation manager). These cases should cover various degrees of user involvement depending on her willingness to drive the process and the knowledge required for this purpose. Most existing model-based/driven UI adaptation approaches do not properly involve the end-user during the adaptation process. Moreover, most of them use only data or information as external sources. There is consequently a shortage of approaches that use knowledge and wisdom to drive the adaptation process.

⑰ **Adaptation logic**, which refers to the algorithm(s) used to perform the UI adaptation. Typical examples of adaptation algorithms are:

- *Probabilistic-based*: the adaptation logic is performed by a probabilistic model (e.g Bayesian network).
- *Rule-based*: the adaptation logic is performed by rules (e.g. Event-Condition-Action (ECA) rules in the *adaptation engine* shown in Fig. 3). For example, Yigitbas et al. [43] presented a model-based approach with which to build context-adaptive UIs based on an adaptation model containing ECA rules.
- *Case-based*: the adaptation logic is based on case-based reasoning.
- *Logic-based*: the adaptation logic is based on logic (e.g. first-order predicate logic).
- *Ontology-based*: the adaptation logic is based on an ontology (e.g. a domain ontology).
- *Evidence-based*: the adaptation logic is performed by an evidence theory.
- *Fuzzy-based*: the adaptation logic is performed by a fuzzy approach (e.g. fuzzy sets).

⑱ **Tool support**, which refers to the automation level provided by the UI *adaptation manager*. We define the following levels:

- *Level 0. Not automated*: all steps of the UI adaptation are performed manually.
- *Level 1. Partially automated*: one or more steps of the UI adaptation are supported by the Adaptation Manager and can be manipulated via its adaptation manager UI.
- *Level 2. Fully automated*: all steps of the UI adaptation are supported by the Adaptation Manager. For example, SLIME [35] adapts a final UI, in a completely automatic manner, in order to avoid the awkward problem posed by the bezels of contiguous displays (Fig. 13).

Fig. 13 Dual monitor UI adaptation [35]

4 Opportunities for the modelling community

The majority of the challenges related to the adaptation of software systems addressed by the modelling community in the last two decades have been mostly of a technical nature. However, in order to attain the full potential of the conceptual reference framework and its properties, the community should also address challenges that arise from the intersection of SE and HCI related to the engineering of human-centred aspects, along with other challenges that may appear as a result of the combined use of MDE and AI.

In the following, we explain some key properties of the framework and discuss the major challenges that need to be addressed in order to take advantage of the opportunities provided by model-based intelligent UI adaptation:

1. **Take advantage of user models.** Many aspects can be captured in a user model such as gender, age, emotions, personality, language, culture, and physical and mental impairments, all of which play an essential role in intelligent UI adaptation. How can we effectively and precisely capture these aspects and use them to propose appropriate UI adaptation? There is no shortage of means to probe the user as more sensors become affordable. For example, wearable devices capture biometric data and external sensors acquire data on the user's behaviour in order to discover adaptation patterns. The question is not so much how to probe the user, but how to make real use of the information obtained while respecting privacy. Another challenge is related to the integration of the user model with other models (e.g. user interface model, context model) in order to ensure the traceability and consistency of these models during the adaptation process.
2. **Towards "grey models".** The MDE community has long sought the ultimately expressive models and adaptation engines that would optimise UI adaptation in most contexts of use, thus producing "white boxes" that can be parsed, analysed, and reasoned about. On the other side, the AI community investigates ML techniques that are based solely on users' data, thus producing "black boxes" that cannot be scrutinised in order to understand them. Why not mix the best of both fields by feeding classical or new models with users' data abstracted by means of ML techniques [27], thus obtaining "grey models"? A representative example is a model-based reinforcement learning approach proposed by Todi et al. [38], which plans a sequence of adaptation steps (instead of a one-shot adaptation) and exploits a model to assess the cost/benefit ratio.
3. **Towards a systematic exploration of adaptation automation degrees.** While Fig. 1 decomposes the UI adaptation into stages to be explicitly supported by tools, Fig. 12 suggests the application of various degrees of automation. Very few of these degrees have been investigated to date when performing model-based/driven UI adaptation and there is a lack of knowledge on how and when to apply them. These challenges open up new research directions to be explored in the future, including the definition of strategies with which to progressively increase the level of automation and 'intelligence' of the Adaptation Engine by exploiting the data, information, knowledge and wisdom captured from specific domains.
4. **Keeping the Human-in-the-loop paradigm**. The aforementioned challenges will never be properly addressed if the end-user is not actively involved, and not just passively grazing between adaptation steps. The application of ML techniques could be structured on the basis of a "Perception-Decision-Action (PDA)" cycle (Fig. 2): the UI adaptation manager uses all available means to perceive/sense the user and probe her context of use in order to suggest and make an appropriate decision for UI adaptation that could be undertaken in various mixed-initiative configurations. Similarly, the end-user should also enter a second PDA scheme in which the UI adaptation is adequately perceived, thus triggering certain human decisions and executing corresponding actions, and a new cycle then starts over. Adaptive systems and models at runtime are core enabling techniques behind an intelligent UI. Models at runtime has been successfully used to automatically reflect changes from a system into changes in models, and vice versa. However, as human cognition is involved, the traditional adaptation in which all the variabilities are pre-defined is not sufficient. We should turn to intelligent UIs that can learn how to adapt to different users based on a user model that captures the user preferences, style of interaction, expertise, emo-

tions, etc. However, models at runtime is rarely applied to such models, and this may be challenging.

5. **Relying on software co-evolution.** Changes in software resulting from UI adaptation go far beyond merely modifying the UI and could potentially impact on any component of the software system or the others represented in Fig. 3. For example, how can we align user interface changes with changes in the software architecture and vice versa? There is a need to formalise these changes in order to reason about them for purposes such as maintainability, traceability, etc. The field of software evolution has an established tradition as regards formalising these aspects, but rarely as regards UI aspects. When the UI comes into play, software evolution should upgrade to software co-evolution in which changes on both sides, the user interface and the software system, should be formalised.
6. **Considering adaptation as a multi-factorial problem.** Since many contextual aspects could influence the quality of UI adaptation, multiple quality factors (e.g. adaptation QAs, adaptation automation degree, the user's characteristics) should be considered together in the same multi-factorial problem. Improving user performance could come about at the expense of cognitive destabilisation. Another challenge is related to the analysis and resolution of conflicting UI adaptation alternatives. In this context, ML techniques could be used to support the decision making as regards the selection of the best UI adaptation that is closer to the end-user's intention.

The aforementioned suggestions represent opportunities for the modelling community to leverage UI adaptation of software systems by investigating some new avenues. A limitation of this approach is that the conceptual reference framework does not provide any prioritisation of its key features and how they should be explored. In addition, it is impossible to consider them all together, although they are intertwined. We need, therefore, to investigate trade-offs and dependencies among the different properties and their levels to get a better understanding of the potential of the proposed framework. Software co-evolution, as suggested, is a form of keeping the human-in-the-loop paradigm when the UI should be adapted as much as possible as a collaboration between the end-user, the system, and any third party, especially when no consensus is reached between the end-user and the system.

Some properties of the reference framework present a level-wise assessment which represents particular capabilities of an approach to support intelligent UI adaptation, which increases the higher the level. More efforts are needed to validate more thoroughly the property levels for a wider set of existing adaptation approaches. In addition, instantiating the framework to specific model-based adaptation scenarios and building prototypes of the main components of the reference framework (e.g. adaptation engine, adaptation transitioner, adaptation machine learning, adaptation explainer) would allow us to get further insights.

The authors of this expert voice trust that the proposed framework for model-based intelligent user interface adaptation will serve as a call for action that could lead to research initiatives by the modelling community.

Acknowledgements This work is supported by the Spanish Ministry of Science, Innovation, and Universities under Grant No.: TIN2017-84550-R, Adapt@Cloud Project and by the Generalitat Valenciana under Grant No.: AICO/2020/113, UX-Adapt Project. Arthur Sluÿters is funded by the "Fonds de la Recherche Scientifique - FNRS" under Grant n40001931.

References

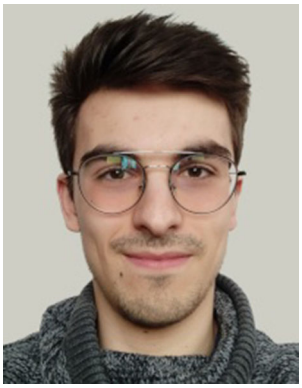
1. Abrahão, S., Bourdeleau, F., Cheng, B.H.C., Kokaly, S., Paige, R.F., Störrle, H., Whittle, J.: User experience for model-driven engineering: Challenges and future directions. In: Proceedings of the 20th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS 2017, Austin, TX, USA, September 17-22, 2017, pp. 229–236. IEEE Computer Society (2017). https://doi.org/10.1109/MODELS.2017.5
2. Akiki, P.A., Bandara, A.K., Yu, Y.: Adaptive model-driven user interface development systems. ACM Comput. Surv. **47**(1), 91–933 (2014). https://doi.org/10.1145/2597999
3. Akiki, P.A., Bandara, A.K., Yu, Y.: Engineering adaptive model-driven user interfaces. IEEE Trans. Softw. Eng. **42**(12), 1118–1147 (2016). https://doi.org/10.1109/TSE.2016.2553035
4. Alvarez-Cortes, V., Zarate, V.H., Ramirez Uresti, J.A., Zayas, B.E.: Current challenges and applications for adaptive user interfaces. In: I. Maurtua (ed.) Human-Computer Interaction, chap. 3, pp. 49–68. IntechOpen, London, UK (2009). https://doi.org/10.5772/7745. https://www.intechopen.com/books/human-computer-interaction/current-challenges-and-applications-for-adaptive-user-interfaces
5. Blouin, A., Morin, B., Beaudoux, O., Nain, G., Albers, P., Jézéquel, J.M.: Combining aspect-oriented modeling with property-based reasoning to improve user interface adaptation. In: Proceedings of the 3rd ACM SIGCHI Symposium on Engineering Interactive Computing Systems, EICS '11, p. 85–94. Association for Computing Machinery, New York, NY, USA (2011). https://doi.org/10.1145/1996461.1996500
6. Bouillon, L., Limbourg, Q., Vanderdonckt, J., Michotte, B.: Reverse engineering of web pages based on derivations and transformations. In: Proceedings of Third Latin American Web Congress, LA-WEB '05, pp. 11. IEEE Computer Society Press,

Piscataway, USA (2005). https://doi.org/10.1109/LAWEB.2005.29
7. Bouzit, S., Calvary, G., Coutaz, J., Chêne, D., Petit, E., Vanderdonckt, J.: The PDA-LPA design space for user interface adaptation. In: Proceedings of the 11th IEEE International Conference on Research Challenges in Information Science, RCIS '17, pp. 353–364. IEEE Press, Hoboken, New Jersey, USA (2017). https://doi.org/10.1109/RCIS.2017.7956559
8. Browne, D., Totterdell, P., Norman, M. (eds.): Adaptive User Interfaces. Computers and People Series. Academic Press, London, UK (1990)
9. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J.: A unifying reference framework for multi-target user interfaces. Interact. Comput. **15**(3), 289–308 (2003). https://doi.org/10.1016/S0953-5438(03)00010-9
10. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Souchon, N., Bouillon, L., Florins, M., Vanderdonckt, J.: Plasticity of user interfaces: A revised reference framework. In: Proceedings of the First International Workshop on Task Models and Diagrams for User Interface Design, TAMODIA '02, p. 127–134. INFOREC Publishing House Bucharest (2002). https://doi.org/10.5555/646617.697235
11. Coutaz, J.: Meta-user interfaces for ambient spaces. In: Coninx, K., Luyten, K., Schneider, K.A. (eds.) Task Models and Diagrams for Users Interface Design, pp. 1–15. Springer, Berlin (2007)
12. Dessart, C.E., Genaro Motti, V., Vanderdonckt, J.: Showing user interface adaptivity by animated transitions. In: Proceedings of the 3rd ACM SIGCHI Symposium on Engineering Interactive Computing Systems, EICS '11, pp. 95–104. ACM, New York, NY, USA (2011). https://doi.org/10.1145/1996461.1996501
13. Dieterich, H., Malinowski, U., Kuhme, T., Schneider-Hufschmidt, M.: State of the art in adaptive user interfaces. In: M. Schneider-Hufschmidt, T. Kuhme, U. Malinowski (eds.) Adaptive User Interfaces Principles and Practice, chap. 10, pp. 13–48. Elsevier Science Publishers, Amsterdam (1994). https://www.elsevier.com/books/adaptive-user-interfaces/schneider-hufschmidt/978-0-444-81545-3
14. Furtado, E., Furtado, V., Silva, W.B., Rodrigues, D.W.T., da Silva Taddeo, L., Limbourg, Q., Vanderdonckt, J.: An ontology-based method for designing multiple user interfaces. In: Proceedings of International Workshop on Multiple User Interfaces, MUI' 01 (2001). https://www.researchgate.net/publication/2567741_An_Ontology-Based_Method_for_Universal_Design_of_User_Interfaces
15. Gajos, K.Z., Chauncey, K.: The influence of personality traits and cognitive load on the use of adaptive user interfaces. In: Proceedings of the 22Nd International Conference on Intelligent User Interfaces, IUI '17, pp. 301–306. ACM, New York, NY, USA (2017). https://doi.org/10.1145/3025171.3025192
16. García Frey, A., Calvary, G., Dupuy-Chessa, S., Mandran, N.: Model-based self-explanatory uis for free, but are they valuable? In: P. Kotzé, G. Marsden, G. Lindgaard, J. Wesson, M. Winckler (eds.) Human-Computer Interaction–INTERACT 2013–14th IFIP TC 13 International Conference, Cape Town, South Africa, September 2-6, 2013, Proceedings, Part III, *Lecture Notes in Computer Science*, vol. 8119, pp. 144–161. Springer (2013). https://doi.org/10.1007/978-3-642-40477-1_9
17. Horvitz, E.: Principles of mixed-initiative user interfaces. In: Proceeding of the ACM International Conference on Human Factors in Computing Systems, CHI '99, pp. 159–166. ACM, New York, NY, USA (1999). https://doi.org/10.1145/302979.303030
18. Hui, B., Partridge, G., Boutilier, C.: A probabilistic mental model for estimating disruption. In: Proceedings of the 14th International Conference on Intelligent User Interfaces, IUI '09, p. 287–296. Association for Computing Machinery, New York, NY, USA (2009). https://doi.org/10.1145/1502650.1502691
19. ISO: ISO/IEC 25010: Software Quality Product Standard. standard, International Standard Organization, Geneva (2019). https://iso25000.com/index.php/en/iso-25000-standards/iso-25010?limit=3&limitstart=0
20. Kühme, T., Dieterich, H., Malinowski, U., Schneider-Hufschmidt, M.: Approaches to adaptivity in user interface technology: Survey and taxonomy. In: Proceedings of the IFIP TC2/WG2.7 Working Conference on Engineering for Human-Computer Interaction, pp. 225–252. North-Holland Publishing Co., Amsterdam, The Netherlands, The Netherlands (1992). https://doi.org/10.5555/647103.717564. http://dl.acm.org/citation.cfm?id=647103.717564
21. Lavie, T., Meyer, J.: Benefits and costs of adaptive user interfaces. Int. J. Human Comput. Stud. **68**(8), 508–524 (2010) https://doi.org/10.1016/j.ijhcs.2010.01.004. http://www.sciencedirect.com/science/article/pii/S1071581910000145
22. López-Jaquero, V., Simarro, F.M., González, P.: AB-HCI: an interface multi-agent system to support human-centred computing. IET Softw. **3**(1), 14–25 (2009). https://doi.org/10.1049/iet-sen:20070108
23. López-Jaquero, V., Vanderdonckt, J., Simarro, F.M., González, P.: Towards an extended model of user interface adaptation: The ISATINE framework. In: J. Gulliksen, M.B. Harning, P.A. Palanque, G.C. van der Veer, J. Wesson (eds.) Proceedings 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. Springer (2007). https://doi.org/10.1007/978-3-540-92698-6_23. https://link.springer.com/chapter/10.1007/978-3-540-92698-6_23
24. Martínez-Ruiz, F.J., Arteaga, J.M., Vanderdonckt, J., González-Calleros, J.M., González, R.M.: A first draft of a model-driven method for designing graphical user interfaces of rich internet applications. In: J.A. Sánchez (ed.) Fourth Latin American Web Congress (LA-Web 2006), 25–27 October 2006, Cholula, Puebla, Mexico, pp. 32–38. IEEE Computer Society (2006). https://doi.org/10.1109/LA-WEB.2006.1
25. McKinley, P.K., Sadjadi, S.M., Kasten, E.P., Cheng, B.H.C.: Composing adaptive software. Computer **37**(7), 56–64 (2004). https://doi.org/10.1109/MC.2004.48
26. Melchior, J., Vanderdonckt, J., Roy, P.V.: A comparative evaluation of user preferences for extra-user interfaces. Int. J. Hum. Comput. Interact. **28**(11), 760–767 (2012). https://doi.org/10.1080/10447318.2012.715544
27. Mezhoudi, N., Vanderdonckt, J.: Toward a task-driven intelligent GUI adaptation by mixed-initiative. Int. J. Hum. Comput. Interact. (2020). https://doi.org/10.1080/10447318.2020.1824742
28. Motti, V.G., Vanderdonckt, J.: A computational framework for context-aware adaptation of user interfaces. In: Proceedings of the 7th IEEE International Conference on Research Challenges in Information Science, RCIS '13, pp. 1–12 (2013). https://doi.org/10.1109/RCIS.2013.6577709
29. Nichols, J.: Using the crowd to understand and adapt user interfaces. In: Proceedings of the 5th ACM SIGCHI Symposium on Engineering Interactive Computing Systems, EICS '13, pp. 1–2. ACM, New York, NY, USA (2013). https://doi.org/10.1145/2494603.2480344
30. Nierstrasz, O., Meijler, T.D.: Research directions in software composition. ACM Comput. Surv. **27**(2), 262–264 (1995). https://doi.org/10.1145/210376.210389
31. Nivethika, M., Vithiya, I., Anntharshika, S., Deegalla, S.: Personalized and adaptive user interface framework for mobile application. In: Proceedings of International Conference on Advances in Computing, Communications and Informatics, ICACCI '13, pp. 1913–1918. IEEE Press, Piscataway, USA (2013). https://doi.org/10.1109/ICACCI.2013.6637474

32. Paramythis, A., Weibelzahl, S., Masthoff, J.: Layered evaluation of interactive adaptive systems: framework and formative methods. User Model. User Adapt. Interact. **20**(5), 383–453 (2010). https://doi.org/10.1007/s11257-010-9082-4
33. Parasuraman, R., Riley, V.: Humans and automation: use, misuse, disuse, abuse. Hum. Fact. **39**(2), 230–253 (1997). https://doi.org/10.1518/001872097778543886
34. Schlee, M., Vanderdonckt, J.: Generative programming of graphical user interfaces. In: Proceedings of the Working Conference on Advanced Visual Interfaces, AVI '04, p. 403–406. Association for Computing Machinery, New York, NY, USA (2004). https://doi.org/10.1145/989863.989936
35. Sluÿters, A., Vanderdonckt, J., Vatavu, R.D.: Engineering slidable graphical user interfaces with slime. Proc. ACM Hum. Comput. Interact. (2021). https://doi.org/10.1145/3457147
36. Sottet, J.S., Calvary, G., Coutaz, J., Favre, J.M.: A model-driven engineering approach for the usability of plastic user interfaces. In: Gulliksen, J., Harning, M.B., Palanque, P., van der Veer, G.C., Wesson, J. (eds.) Engineering Interactive Systems, pp. 140–157. Springer, Berlin (2008)
37. Teevan, J., Dumais, S.T., Liebling, D.J., Hughes, R.L.: Changing how people view changes on the web. In: Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology, UIST '09, p. 237–246. Association for Computing Machinery, New York, NY, USA (2009). https://doi.org/10.1145/1622176.1622221
38. Todi, K., Bailly, G., Leiva, L., Oulasvirta, A.: Adapting user interfaces with model-based reinforcement learning. In: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, CHI '21. Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3411764.3445497
39. Vanderdonckt, J., González-Calleros, J.M.: Task-driven plasticity: One step forward with ubidraw. In: P. Forbrig, F. Paternò (eds.) Engineering Interactive Systems, Proceedings 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. Springer (2008). https://doi.org/10.1007/978-3-540-85992-5_16
40. Vatavu, R.: Nomadic gestures: A technique for reusing gesture commands for frequent ambient interactions. J. Ambient Intell. Smart Environ. **4**(2), 79–93 (2012). https://doi.org/10.3233/AIS-2012-0137
41. van Velsen, L., van der Geest, T., Klaassen, R., Steehouder, M.F.: User-centered evaluation of adaptive and adaptable systems: a literature review. Knowl. Eng. Rev. **23**(3), 261–281 (2008)https://doi.org/10.1017/S0269888908001379. https://www.cambridge.org/core/journals/knowledge-engineering-review/article/abs/usercentered-evaluation-of-adaptive-and-adaptable-systems-a-literature-review/C77A0D4AE8BAF5808E55214884245965
42. Yigitbas, E., Jovanovikj, I., Biermeier, K., Sauer, S., Engels, G.: Integrated model-driven development of self-adaptive user interfaces. Softw. Syst. Model. **19**(5), 1057–1081 (2020). https://doi.org/10.1007/s10270-020-00777-7
43. Yigitbas, E., Sauer, S.: Engineering context-adaptive UIs for task-continuous cross-channel applications. In: Human-Centered and Error-Resilient Systems Development—IFIP WG 13.2/13.5 Joint Working Conference HCSE 2016 and HESSD 2016 Stockholm, Sweden, August 29–31, 2016, Proceedings, pp. 281–300. Springer (2016). https://doi.org/10.1007/978-3-319-44902-9_18

Silvia Abrahão is an Associate Professor at Universitat Politécnica de Valéncia, Spain. Her research interests include quality assurance in model-driven engineering, empirical assessment of software modeling approaches, model-driven cloud services development and monitoring, and the integration of usability into software development. Contact her at sabrahao@dsic.upv.es

Emilio Insfran is an Associate Professor at Universitat Politécnica de Valéncia, Spain. His research interests include requirements engineering, model-driven engineering, DevOps, and cloud services development and evaluation. Contact him at einsfran@dsic.upv.es



Arthur Sluÿters is a PhD student in Computer Science at Université catholique de Louvain, Belgium, where he is an "aspirant FNRS" under contract no. 1.A434.21. His research interests include intelligent user interfaces (IUI), gesture recognition, gestural user interfaces, and radar-based interaction. Contact him at arthur.sluyters@uclouvain.be

Jean Vanderdonckt is a Full Professor at Université catholique de Louvain, Belgium, where he leads the Louvain Interaction Lab. His research interests include engineering of interactive systems (EICS), intelligent user interfaces (IUI), multimodal systems such as gesture-based, information systems, and model-based/driven engineering of user interfaces. Contact him at jean.vanderdonckt@uclouvain.be