Showing User Interface Adaptivity by Animated Transitions

Charles-Eric Dessart, Vivian Genaro Motti, and Jean Vanderdonckt
Université catholique de Louvain, Louvain School of Management
Louvain Interaction Laboratory, Place des Doyens, 1 – B-1348 Louvain-la-Neuve (Belgium)
{vivian.genaromotti, jean.vanderdonckt}@uclouvain.be – Phone: +32 10 478525

ABSTRACT
In order to reduce the inevitable end user disruption and cognitive perturbation induced by adapting a graphical user interface, the results of the adaptation could be conveyed to the end user by animating a transition scenario showing the evolution from the user interface before adaptation to the user interface after adaptation. A transition scenario consists of a sequence of adaptation operations (e.g., set/change a property of a widget, replace a widget by another, resize a widget) belonging to a catalogue of operations defined as an Extended Backus-Naur Form grammar. Each transition operation has a range from a single widget (e.g., this "Ok" button) to a selection of widgets based on a selector mechanism (e.g., all validation widgets of this family of interfaces). A transition scenario is built either automatically by any adaptation algorithm or interactively by a specific editor for designers. An animator then executes the animation scenario by parsing each adaptation operation one by one or in a grouped mode and by rendering them by an animated transition on a user interface model. The type (e.g., wipe, box in, box out) and parameters (e.g., animation speed, pace, direction) of each animated transition have been selected based on usability guidelines for animation. A user study suggests that a transition scenario reinforces understandability and trust, while still suffering from lag.

Author Keywords
Adaptation, adaptivity, animation, transition operation, selection mechanism, transition operation, visual transition.

General Terms
Design, Experimentation, Human Factors, Verification.

ACM Classification Keywords
D2.2 [**Software Engineering**]: Design Tools and Techniques – *Modules and interfaces; user interfaces*. D2.m [**Software Engineering**]: Miscellaneous – Rapid Prototyping; reusable software. H.5.1 [**Information interfaces and presentation**]: Multimedia Information Systems – *Animations*. H5.2 [**Information interfaces and presentation**]: User Interfaces – *User-centered design*.

INTRODUCTION
User Interface (UI) adaptation typically consists in modifying parts or whole of a particular interface in order to address specific needs required by an end user or a category of end users. Adaptation falls into two categories depending on who is in control of the adaptation process [7,11,24]: *adaptability* refers to as the ability of the end user to adapt the UI, *adaptivity* refers to as the ability of the system to adapt the UI. Mixed-initiative adaptation exists when both the end user and the system cooperate towards the UI adaptation goal. Adaptivity, although expensive to develop, has demonstrated several benefits [27] and is largely used in a wide range of domains of human activity, such as ambient intelligence [13], automotive [30], electronic commerce [33], algorithmic [26], and information systems [11].

Some of the main shortcomings of adaptivity are [8,27]: *end user disruption* caused by a behavior that is unexpected by the end user and *cognitive perturbation* when the end user, confronted to a new UI, must reconcile with this UI by imagining the correspondence between the UI before and after adaptation. Between the UI before adaptation and the UI after adaptation, there is nothing than a big whole, thus reinforcing the cognitive perturbation. Cognitive psychology [19] refers to this phenomenon as "cognitive destabilization", meaning that any user is mentally destabilized when confronted with anything unexpected, unprecedented, or unpredicted contents. The end user remains in this stage of cognitive destabilization until a "re-stabilization" restores a relation between the past and the newly presented contents. The end user does not suffer from these shortcomings in adaptability since the end user remains in control (therefore knowing what she is doing), as opposed to the system is in control in adaptivity (therefore the end user does not know what the system is doing). In order to address this challenge, animated transitions are applied to showing how the adaptivity process has been conducted in order to explain to the end user what has been adapted, and perhaps why.

The remainder of this paper is structured as follows: the next section reports on some related work. Then, the full process of adaptation by animated transitions is introduced, motivated, and defined. The software architecture supporting the implementation of animating transition is explained, and exemplified. A user study is then conducted in order to determine what the impact of animated transition over the end user is. Finally, a conclusion delivers the main points of this research and presents some future avenues.

RELATED WORK
Animated transitions and support for adaptivity are two main fields of research that are related to this work since its originality lies in considering the former for the latter.

EICS'11, June 13–16, 2011, Pisa, Italy.

Animated transitions

Animation [1, 35] has been widely used as a general technique for supporting end users in understanding different types of contents: evolution of a dynamic process (e.g., a mechanical process) [34, rule 2.4*18], a chronological sequence of events over time (e.g. a country demography) [12] or complex graphics (e.g., earth rotation) [39] and statistics [20]. It has also been used to represent various types of relations between elements [34, rule 2.4*19], such as sequences, important [35] or spatial connections [3], causal relations [41], for organizing diagrams [5], and for searching in 3D treemaps where task times and user performance were improved [4]. Small animated icons could convey functionality better than static icons of the same size [1].

Animated transitions [2,4,20] in interactive systems are aimed at conveying to the end user a transition between states, views or scenes, e.g., to foster a smooth transition between two scenes, menus [22] or images [23]. Animated transitions improve feedback on users' actions [31], to notify display changes [29], and to improve situation awareness in a distributed environment. Sliding and blinking animated transitions were used to convey a context change on a menu [22] or images [23] on a mobile device with a positive impact on perception and conception of change.

Cartoons-inspired visual effects [10,28,38] have also been added to achieve a more realistic, if not lifelike, visual effect in the transition. Animated help better explains a GUI usage to the end user [36].

In order to be effective and usable, animated transitions need to be carefully designed as they are subject to a series of intrinsic shortcomings: they may require more cognitive workload than static images [41], animation is always the first display element attracting the end user's attention [21] whatever the animation goal is, they may cause user distraction [41], their duration always induce some lag [35], the animated objects should not exceed a certain threshold [9]. To minimize lag, an animated transition should be fast, but not too fast, otherwise the end user may completely overlook the animated transition. Typical duration may range from 300 msecs to several secs, depending on the complexity of the transition and other user- and situation-specific factors such as familiarity, expectation, attentiveness, and perceptual abilities that are difficult to predict [2].

Animated transitions may induce a significantly positive impact on understanding display changes, whether it is for notifying value changes in controls of a GUI [2], for updated contents in a web page [37], such as web navigation [14] or for evolving data in a dynamic display [12]. Different techniques support end users in perceiving and understanding screen changes, mainly based on animation between states [22,23], perhaps supplemented by sound [32].

Mnemonic rendering [6] consists of an image-based technique that buffers all changes of a fast-changing dynamic display and restitutes these changes under the end user's control via a memory jog. *DiffIE* consists of highlighting web page contents that have been updated since last visit [37]. A positive value has been demonstrated on how people interact with the web page and understand their contents. For instance, some users confessed they initially perceived some contents as static although they were dynamic. *Phosphor widgets* rely on afterglow effect in order to leave some visual reminiscence of changes of values of widgets (e.g., the value change of a slider, the check/uncheck of a check box, a new selection in a radio box). Rhetorical Structure Theory (RST) is exploited to apply Flash multimedia animated transitions on web pages to explain how web navigation has been transformed [14]. *Differentiated transitions* [31,32] are animated transitions that support explaining a process over time in a way that is reflected in the visual effect. For instance, the transfer time, the network bandwidth, and the file size are explicitly represented in an animated transition depicting a file transfer. RST, respectively Mnemonic rendering, force end users to wait for, respectively to replay, the display changes, thus inducing some lag [35]. DiffIE does not induce such a drawback (since the highlighting is almost instantaneous), nor Phosphor widgets (since the afterglow effect does not stop user in their tasks). Differentiated transitions actually animate the task while being executed [31], thus not representing any hindrance for achieving the end user's task. The aforementioned techniques certainly contribute to improving the perception of display changes over time, but they do not address the perception of UI adaptation over time, even if UI adaptation could be considered as a certain type of screen change. More importantly, they are not capable of recording the adaptation process to replay or explain it afterwards. RST [14] is the only one applying animated transitions on an abstract UI description of a web page. In our work, the animated transitions are applied on a general-purpose GUI model, but could be equally interpreted with any similar User Interface Description Language (UIDL).

Support for adaptivity of user interfaces

Adaptivity has been subject to many pieces of work that lead to a recognition of a series of benefits vs. costs [7,11,17]. In particular, adaptive UIs are able to optimize task completion time and rate [27], to induce a positive impact on accuracy [18], human performance [15,25], predictability [18], situation awareness [15,25] and workload [25]. Adaptivity has also been revealed effective when the UI should be adapted to the constraints imposed by any loss of screen resolution [15], like on mobile devices [16].

In this work, animated transitions show to the end user how an adaptivity process has led to an adapted GUI. It is expected that all benefits of animated transitions will establish a feeling of continuity between the UI before and after adaptation, thus impacting the end user's disruption and the cognitive perturbation discussed in the introduction. To our knowledge, this combination remains unexplored.

WHICH TRANSITION FOR WHICH ADAPTATION?

In order to address the problem of determining which animated transitions are considered adequate to mimic an adaptation operation, this section first provides a catalogue of such adaptation operations to be supported by the animator. It

then reviews usability guidelines and cognitive psychology principles for animation for establishing a mapping between adaptation operations and animated transitions.

A Catalogue of Adaptation Operations

Adaptation operation is hereby defined as any transformation performed on any UI element in order to adapt the UI for the ultimate benefit of an end user interacting in a certain context of use. Such adaptation operations may involve a series of actions that are intended to obtain a certain global effect on the initial UI before adaptation until the final UI after adaptation is obtained (Fig. 1). Each adaptation operation produces a *transient UI being adapted* (Fig. 1), which consists in an intermediary UI stage during adaptation. Usually, the end user does not perceive any of these transient UIs, being presented only with the initial and the final UIs, which cause the end user disruption and the cognitive perturbation. The whole sequence of adaptation operations conducted for the UI adaptation is called the *adaptation scenario*, that could involve a wide spectrum of adaptation operations which fall into five categories [15,16,17]:

1. *Resizing operations*: are aimed at changing a widget size in order to optimize screen real estate, aesthetics, and visual design [40]. For instance, an edit field could be enlarged/shortened in height and/or length to take less space and to be subject to various alignments.
2. *Relocating operations*: are aimed at changing a widget location in order to reduce the screen space consumption. For instance, "Ok", "Cancel", and "Help" push buttons could be relocated to the bottom of a dialog box.
3. *Widget transformations*: are aimed at replacing one or a group of widgets by another widget or another group of widgets ensuring the same task, perhaps with some degradation [16]. For instance, an accumulator that consists of list boxes with possible values and chosen values could be replaced by a multi-selection list, which could be in turn replaced by a multi-selection drop-down list.
4. *Image transformations*: are aimed at changing the size, surface, and quality of an image in order to accommodate the constraints imposed by the new context of use, namely the display/platforms constraints.
5. *Splitting rules*: are aimed at dividing one or a group of widgets into one or several other groups of widgets that will be displayed separately. For instance, a dialog box is split into two tabs in a tabbed dialog box.

A single adaptation operation could be performed on a single UI element in isolation (e.g., resizing an individual or a compound widget) or several related UI elements concurrently (e.g., resizing a group of aligned edit fields). Therefore, we will define an *Adaptation Operation Language* (AOL) for expressing one adaptation operation on one element at a time first and then, this will be generalized to several UI elements together.

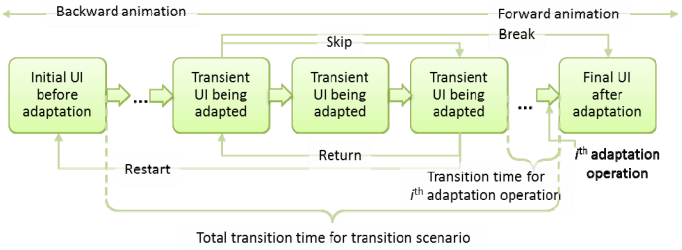

Figure 1. Timeline of the animation process.

Adaptation Operation Language. We now provide a catalogue of adaptation operations belonging addresses the five aforementioned categories. For this purpose, each adaptation operation is defined in an Extended Backus-Naur Form (EBNF) format to form a grammar. In this notation, brackets indicate an optional section, while parentheses denote a simple choice in a set of possible values.

SET <Element.property> TO {value, percentage}: assigns a value to a widget property or a percentage of the actual value. For instance, SET "pushButton_1.height" TO 10 will resize the push button to a height of 10 units while SET "pushButton_1.height" TO +10 increases its height by 10%.

DISPLAY <Element> [AT x,y]: displays a UI element whose identifier is provided at a *x,y* location where *x* and and *y* are integer positions (e.g., in characters or pixels). For instance, DISPLAY "pushButton_1" AT 1,1 will display an identified push button at coordinates 1,1 on a designated display. UNDISPLAY <Element> [AT x,y] is the inverse operation. DISPLAY <Message> [AT x,y] displays a provided message.

MOVE <Element> TO x,y [IN n steps]: moves a UI element to a new location indicated by its coordinates *x* and *y*, possibly in a fixed amount of steps.

REPLACE <Element1> BY <Element2>: replaces a widget Element1 by another one Element2. Sometimes the replacement widget could be determined after an adaptation algorithm, thus giving the following definition: REPLACE <Element1> BY <AdaptationAlgo:>. This mechanism is similar for image transformations: images are usually transformed by local or remote algorithms (e.g., for resizing, converting, cropping, clipping, repurposing), thus giving the following definition: TRANSFORM <Image1> BY <ImageAlgo:URL>.

DISTRIBUTE <Elements> INTO <Containers> [BY <DistribAlgo:URL>: computes a distribution of a series of UI Elments into a series of UI Containers, possibly by calling an external algorithm, local or remote.

Selection mechanism. In the above definitions of adaptation operations, only one UI element is provided as parameter at a time. Obviously, an adaptation operation could have a scope of several UI elements together. For this purpose, a selection mechanism is introduced that defines a scope of UI Elements that could serve as a parameter. A *Selector* consists of a defi-

nition of the UI Element types to which the adaptation operation applies, and a series of property declarations that define the operations. Four major types of selector scope are considered that replace <Element> or <Elements> fields in the previous definitions:

1. universalSelector: applies the adaptation operation to all UI elements belonging to the current GUI of concern. For instance, SET "universalSelector.backgroundColor" TO "Ivory" will change the background color of the entire GUI into ivory.
2. elementTypeSelector: applies the adaptation operation to all elements belonging to the selector's type (e.g., all containers, all list boxes). For instance, SET "elementTypeSelection.foregroundColor=pushButton" TO "lightGrey" will set the foreground color of all push buttons of the current UI to light grey.
3. classSelector: applies the adaptation operation to all elements belonging to the selector's type whose definition makes them part of the class (e.g., all containers having an ID greater or equal to "CC2", all list boxes having more than 10 items).
4. idSelector: applies the template to only one element belonging to the GUI of concern: the one whose id attribute matches the string contained in the parameter. The idSelector is used by default and should not be necessarily specified.

Animated transitions for an adaptation operation

On the one hand, usability guidelines [1,9,10,15,21,28,35, 40,41] exist that recommend an animated transition for a particular usage that has been proved effective and/or efficient to some extent. On the other hand, cognitive psychology provided a series of high-level principles that could be converted into design guidelines. For instance, the visual animation dynamicity should be appropriate to the animated transition: "wipe from left" is considered less disruptive when explaining a process that is demonstrated from left to write, other animations like "appear", "fall from top" are considered too disruptive and/or too visually impactful. "Venetian blinds" should be used when the process evolves to a significantly different stage, which is not appropriate for a local change. In order to decide which animated transition is appropriate for which adaptation operation, some major animated transitions are defined in Table 1 and classified into five families in Table 2 that will then be used in establishing mappings summarized in Table 3. Presentation software [20,26] and animation [1,10,21, 35] have introduce a large amount of varied animated transitions. Therefore, animations selected in Table 1 have been chosen according to the following criteria: they are the most frequently used techniques that are described in a consistent way throughout the literature, they are easy to implement, they convey a message that is simple enough to be understood while being flexible enough to allow some variation. In order to group these selected animated transitions, we clustered them into five families based on visual properties [41] (e.g., visual differentiation, clarity, density) based on the literature [28,35,40] (Table 2):

Icon	***Name*: definition**
	*Horizontal scroll from right*: to display the next element from a sequence of UI elements
	*Horizontal scroll from left*: to display the previous element from a sequence of UI elements
	*Vertical scroll from bottom*: to proceed with a step-by-step reasoning, a continuous subject or a long passing over, or a movement
	*Vertical scroll from top*: to move back in a step-by-step reasoning, a continuous subject or a long passing over, or a movement
	*Diagonal replacement from top/bottom left corner*: to go back to the previous page or Screen or UI element
	*Diagonal replacement from top/bottom right corner*: to move to next page or screen or UI element
	*Venetian blinds*: to present a completely different topic, to provide a feeling of coordinated time, to convey a significant transition
	*Bam door close*: to close a transient screen (e.g., an information screen, the About… splash screen), to close a current scene, to signify game over
	*Bam door open*: to open a transient screen, to initiate a new step, to open a new window or UI element, to launch a game, a simulation
	*Iris open*: to show more detailed information about a particular topic
	*Iris close*: to show more general information about a particular topic

Table 1. Definitions of some major animated transitions.

F1	Scroll, Diagonal replacement, Wipe
F2	Checkers, lines, columns, blinds, bam door open/close
F3	Cover, uncover
F4	Open, close, Box in, Box out, Iris open/close
F5	Cutting, Black transition

Table 2. Five families of animated transitions.

1. *F1 family* gathers animated transitions that simply recover the old element by a new element (i.e, in our context any UI element, but in general, it could be any graphical object of a display or an entire display such as a graphic, a presentation slide, or an overhead). The main variation lies in the way the new element is presented with respect to the old one, which is usually the direction or the shape of the animated transition.
2. *F2 family* gathers animated transitions that divide the old element into regions that are further subject to partial overlapping when transitioning to the new element.
3. *F3 family* gathers animated transitions that present the new element on top of the old element by moving it in some way. The new element is therefore perceived as it "flies" over the old element.

4. *F4 family* gathers animated transitions of type double "blinds" or "windows". The new element is divided into two regions and progressively appears on top of the old element because the blinds have been opened or closed.
5. *F5 family* gathers specific animated transitions that do not induce any movement or overlapping of the new element, but that simply makes the old element disappearing for the new element by a sharp visual effect.

Table 3 motivates the selection of animated transition for each adaptation operation that was previously defined. Animated transitions from F5 should be reserved for highly-changing regions of the display. Per se, there is no direct adaptation operation that is directly appropriate to this kind of transition, except the complete display/replacement of a significant region. For the moment, this animated transition was not incorporated in the Animator for this reason, but this may change depending on users' feedback. We hereby define a *transition scenario* as a sequence of adaptation operations rendered by animated transitions based on Table 3.

USER CONTROL ACTIONS

The critical success factor for an animation beyond its appropriateness (as discussed in the previous section) resides in the user's capability to govern the pace and duration of the animation. This is also applicable to our animated transitions in the transition scenario. In order to provide some user control over the whole animation process, thus keeping control over the total transition time of the animation scenario (Fig. 1), the user may want to operate some actions either in the *forward animation* (e.g., to understand the evolution of the adaptation process) or in the *backward animation* (e.g., to come back to a previously applied animated transition). These actions are made available in the Animator through keyboard shortcuts as follows:

- *Skip* (Pg Dn): terminates the current animated transition and skips to the next one in the transition scenario. This user action is motivated by the end user need to stop an animated transition as soon as it is understood by users.
- *Break* (End): terminates the current transition scenario. This is probably the most important user action since the end user should be able to terminate the animation at any time, as recommended by Smith & Mosier [34].
- *Return* (Pg Up): escapes from the current animated transition and returns to the previous one in the transition scenario. This user action is motivated by the end user need to come back to a previously animated stage when there is a disruption in the understanding.
- *Restart* (Home): starts again the current transition scenario from the first animated transition. This user action is motivated by the end user need to replay entirely the transition scenario in case of misunderstanding.

Adaptation operation	Animation family, animated transition with justification
SET that modifies the length of a UI element into a larger value (absolute or relative)	Horizontal scroll/wipe from left (F1): this operation minimizes the visual change since only the right part resulting from the enlarging is changing. For edit fields, for instance, this is particularly appropriate because it gives the feeling that the field is really expanding
SET that modifies the height of a UI element into a larger value (absolute or relative)	Vertical scroll/wipe from bottom (F1): this operation minimizes the visual change since only the right part resulting from the enlarging is changing
DISPLAY that displays a new UI element at a certain position	Uncover (F3), Box out (F4), or Iris open (F4): these operations all induce a progressive display of a new UI element at once, thus creating the illusion that it is coming from the empty.
UNDISPLAY that undisplays a new UI element at a certain position	Cover (F3), Box in (F4), or Iris close (F4): these operations all induce a progressive disappearing of a existing UI element at once, thus creating the illusion that it is shrunk to an empty/white region.
REPLACE that substitutes a UI element by another one	Bam door open (F2): this operation affects the entire visual aspect of the previous one and the new one.
DISTRIBUTE that computes a distribution of a series of UI Elments into a series of UI Containers	Bam door open (F2) or Iris open (F4): these operations enable the visualization of an entire group at once, instead of showing every little display change individually
MOVE that moves a UI element to a new location indicated by its coordinates x and y, possibly in a fixed amount of steps	Ideally, the UI movement could be represented by an animation depicting the movement itself. But practically, this would induce a very long animation, thus increasing again the lag. Therefore, we preferred to adopt a disappearing of the UI element from its original location and an appearing to its target location. Depending on these locations, vertical, horizontal or diagonal replacements (F1) are selected. For instance, when a UI element disappears from a top left location to a bottom right location, a diagonal replacement from top/bottom left corner is selected, thus creating the illusion that the element moves from one location to another. Consistently with this direction, when a UI should only move linearly (either vertically or horizontally), a vertical/horizontal scroll is selected instead.

Table 3. Mapping table between adaptation operation and animated transition.

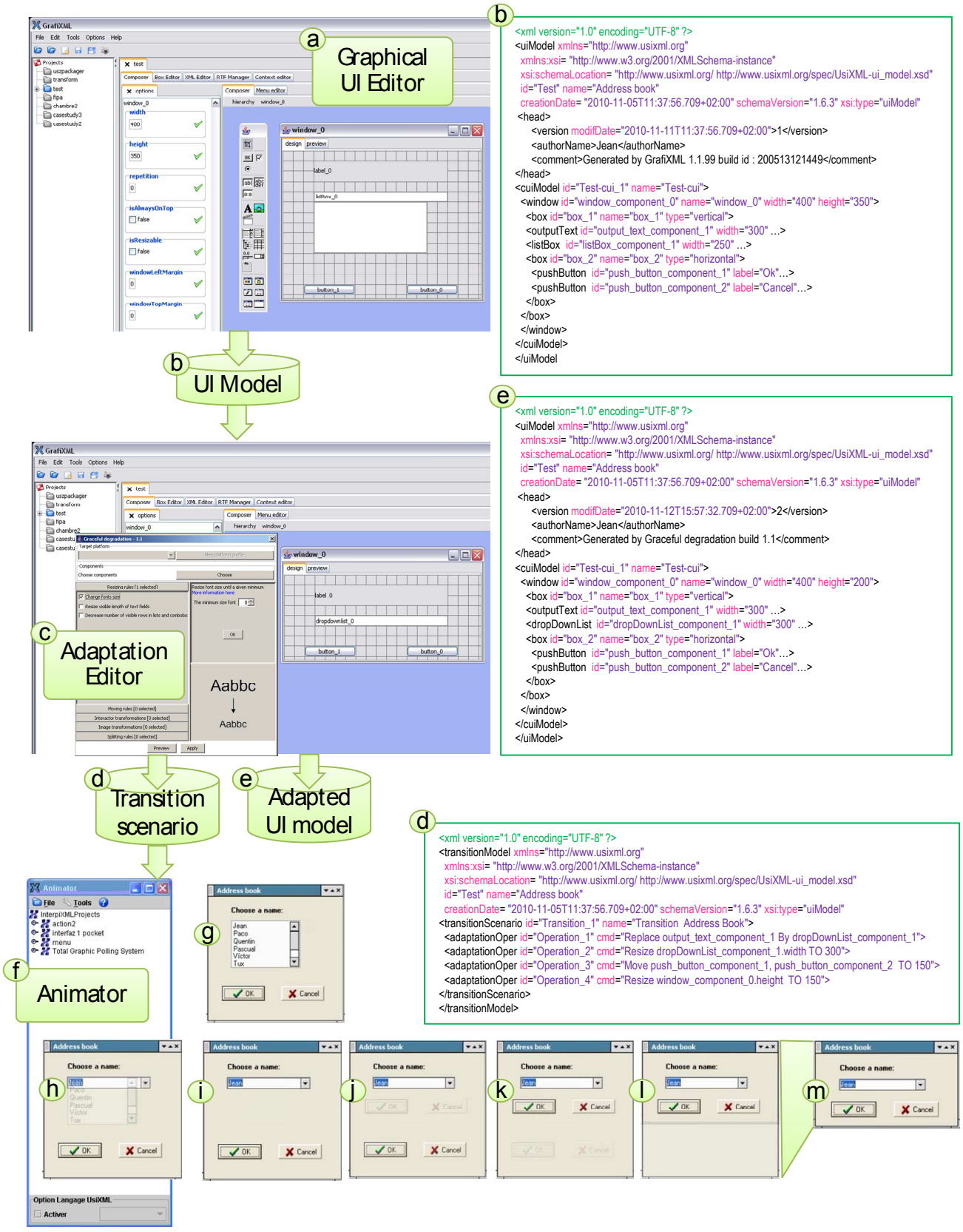

Figure 2. Process of adaptation rendered by a transition scenario.

- *User-Break* during *n* msecs or *n* secs, until *m* next transition/end (Space with repetition): stops momentarily the animated transition. This user action is motivated by the end user need to pause or stop as long as the space bar is pressed, depressed so as to allow time enough to understand the adaptation operation being animated.
- *Acceleration* (CTRL+A): increases the speed of the animation scenario. This user action is motivated by the end user need to speed up the animation pace when there is no problem of understanding when the understanding of the animated transitions is fine-grained or obvious.
- *Deceleration* (CTRL+D): decreases the speed of the animation scenario. This user action is motivated by the end user need to slow down the animation pace when there is a need to allow more time for understanding of the animated transition.

PRODUCING A TRANSITION SCENARIO

After having defined adaptation operations and animated transitions that are adequate for conveying the message of a particular adaptation operation, the process of adaptation rendered by a transition scenario (Fig. 2) is now explained, along with the implementation of software that supports it.

Step1: Producing a User Interface Model.

Fig. 2a reproduces a screen shot of a *Graphical UI Editor* with which the designer can edit the initial UI before adaptation. As in any UI builder, the designer drags widgets from a palette and drops them onto a working surface area where they can be assembled, grouped, and aligned. The GrafiXML was developed that exports the results of the design phase into a Concrete User Interface (CUI) model (Fig. 2b) that is stored in UsiXML, a XML-compliant UIDL that is partially reproduced in Fig. 2b. A CUI model basically consists here of a recursive hierarchy of containers and widgets that are expressed independently of any programming or makup language. The editor today consists of about 20,000 LOC implemented in Java 1.5 with various libraries (e.g., Castor, Jakarta, Jdom, LiquidINF, Looks, Xalan, and Xerces) and stores models in a MySQL V5.0 database. This editor today supports two UIDLs: UsiXML (http://www. usixml.org) and XAML (http://archive.msdn.microsoft. com/XAML/), and could be extended to other UIDLs through a set of XSLT transformations provided that equivalent concepts exist.

Step 2: Producing an Adapted User Interface Model.

Fig. 2c reproduces a screen shot of an *Adaptation Editor* with which the designer can apply any adaptation operation defined in the aforementioned catalogue on the initial UI in order to obtain the final UI after adaptation. For this purpose, control panels are provided to let the designer applying any adaptation operation desired on the UI being designed in the Graphical UI Editor. Any such operation, once executed, is stored in a log file. Each line of the log file is an instruction compatible with the EBNF format for adaptation operations. All lines of adaptation then form a transition scenario stored in an independent XML file for a transition model (Fig. 2d). "Undo", respectively "Redo" operations cancels, respectively duplicates, the last operation in the file. Fig. 3 reproduces another panel of this Adaptation Editor in which the designer is applying a selection of UI elements based on the selection mechanisms introduced in order to apply widget substitution. The *Adaptation Editor* consists of 2,600 LOC implemented in Java 1.5 with some libraries (Jdom, JSearch, Xalan, and Xerces). The adapted UI is maintained in an adapted UI model (Fig. 2e).

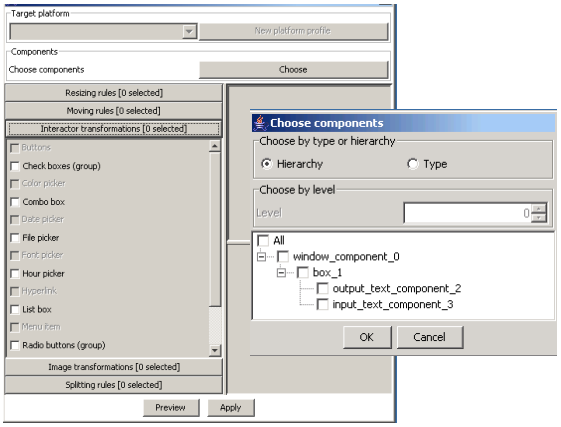

Figure 3. Some control panels of the adaptation editor.

Step 3: Rendering a Transition Scenario.

Fig. 2f reproduces a screen shot of an *Adaptation Editor* that opens a transition scenario to be applied to a UI model. For this purpose, the transition scenario file (Fig. 2d) is parsed and animated transitions corresponding to each line of the scenario (according to Table 3) is produced, equipped with the user actions described in the previous section. An animation is then produced that shows the transition from the UI before adaptation until the UI after adaptation is reached (Fig. 2g to m). The *Animator* consists of 1,100 LOC implemented in Microsoft Expression Studio. This environment has been selected for the following reasons: it is already compliant with XAML, a XML-compliant UIDL for CUI; all UI elements of a GUI expressed in XAML are vector-based and logical operations could be performed on them; animated transitions of Tables 1 and 3 are already built-in with some options (like speed, duration); MS Expression Studio comprises five products: Expression Blend (for building GUIs for Silverlight, Windows, and Surface), Expression Blend SketchFlow (for prototyping these GUIs), Expression Web (for building Web GUIs), Expression Design (for creating graphic assets for the Web or Silverlight, Windows, and Surface), and Expression Encoder (for preparing video assets for the Web or Silverlight, Windows, and Surface). In our case, we used Expression design to develop the animated transitions based on the AOL defined previously and Expression Blend for the Animator itself.

In the next subsections, we examine when adaptation operations could be grouped together in order to reduce the animation duration while not decreasing its main quality. Animations could then be executed in series or in parallel.

Grouping similar adaptation operations

On the one hand, grouping similar adaptation operations into one single animated transition instead of playing the same animated transition several times for several similar adaptation operations makes sense. This would decrease animation duration (thus reducing animation lag [35]) and produce a global animation at once (thus improving the understandability of the whole adaptation process executed through the transition scenario). For instance, instead of moving up two horizontally aligned push buttons one after another (as in Fig. 2j, k), they could be moved all at once. Similarly, a same adaptation operation performed on a series of physically adjacent widgets could lead to a grouped animated transition: right resizing a column of edit fields could be done at once in one single animated transition. *On the other hand*, grouping similar adaptation operations should consider human limitations: no occlusion, no overlapping should be induced; the cognitive load of the animated transition should be minimized; the amount of widgets subject to animation should be reduced. Psychophysics research has revealed that average end users cannot track more than five objects in movement [9]. Therefore, the EBNF allows specifying grouped adaptation operations, but it does not check whether any such limitation occurs.

Grouping dissimilar adaptation operations

It is also possible to consider grouping dissimilar adaptation operations under certain conditions. Typically, such a grouping could be made possible when several different animated transitions affect the same widget but in different ways. For instance, replacing a list box by a drop down list while enlarging the resulting widget is acceptable as long as the associated animated transitions affect different portions of the same widget or non-overlapping regions in the same container. In contrast to similar operations where all operations could be performed at once, in this case the transitions cannot be executed all at once, but in a way that could be perceived together, e.g. by fading in/out. This situation is acceptable provided that the amount of transitions per widget does not exceed a certain threshold.

USER STUDY ON SUBJECTIVE PERCEPTION

After describing how a transition scenario could be dynamically produced at run-time, we report on a user study on the subjective perception of end users when confronted to a transition scenario of animated transitions showing UI adaptivity. The purpose of the study was to examine whether the transition scenario helps users to understanding it.

Participants

We conducted a user trial of 20 users (6 female, 14 male) who were recruited from a database of volunteers coming from different disciplines (e.g., marketing, finance, pharmacy, medicine) and having different ages and background.

Method

The participant's task was to watch 3 transition scenarios: a *personal information form* as a simple scenario to foster initial understanding of the whole process; an *address book* adapted for a PDA (Fig. 2g to m) as a moderately-complex scenario to illustrate other adaptation operations, and the *connexion* between the two previous ones as a more complex example in order to illustration dialogue and navigation. Then participants had to demonstrate their appreciation of the animation process by answering a questionnaire made up of a section of 12 closed questions and 3 open questions (i.e., what are the aspects that you liked the most, what are the aspects that you disliked the most, what do you suggest in order to improve the quality of the animation). In the closed part of the questionnaire, we asked the participants to respond to a series of positive statements on a scale of one to five (1 = strongly disagree, five = strongly agree). The first two statements on the questionnaire tested user satisfaction with the two interfaces. The statements were:

1. I liked the animation process
2. I liked the animation interface
3. I preferred the animation over no animation at all
4. The animation is easy to use
5. The animation is easy to control
6. The animation is easy to understand
7. The animation is easy to follow
8. The animation is easy to progress (forward an.)
9. The animation is easy to revert (backward an.)
10. The animation represents the adaptation
11. The animation is fast
12. I would recommend using the animation

Results and Discussion

The cumulated histogram in Figure 4 summarizes the responses to the statements included in the questionnaire. The distribution for statement #1 revealed that nobody had a negative feeling about having an animation of the transition scenario (neither orange nor red areas). But some participants were concerned about the Animator UI: the distribution of responses for statement #2 shows this, while the preference (statement #3) follows a similar trend. Participants appear, however, to show a preference for the animation over no animation at all ($p = 0.031$ for a one-tail t-test with 19 degrees of freedom). But this does not mean that the animation should always come automatically, as suggested in statement #4: participants seemed to appreciate the animation effects, but do not appreciate the time consumed by the animation, especially when the total animation time is long. Rather, they prefer to keep control over the transition scenario with user actions, but it turns out that they do not know exactly what user action to undertake since they do not know what the next adaptation operations are.

Forward animation (statement #8) is perceived in a better way that the backward animation (statement #9). The last statement (#12) on the questionnaire verifies the results of the global perception responses by asking the participants to respond to a recommendation statement: three quarters of the participants were confident in recommending the animation transition as a mechanism for showing the adaptation. These results are more moderate than the initial statements.

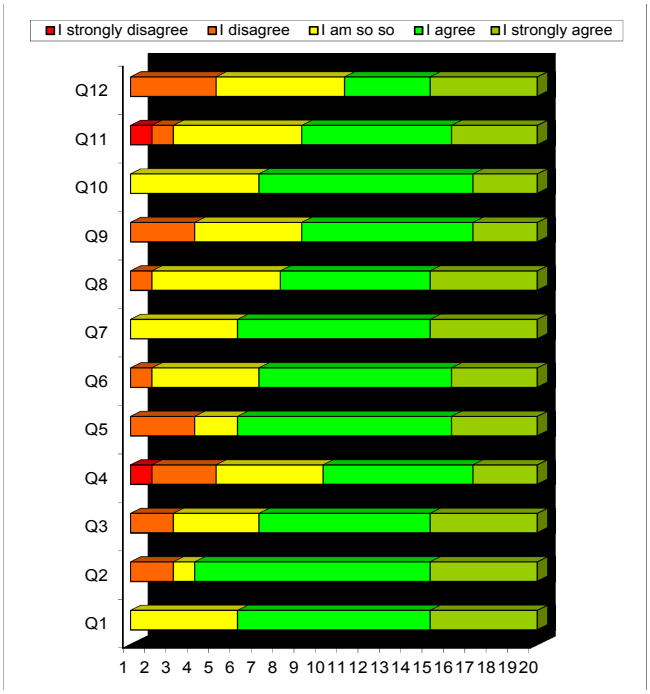

Figure 4. Distribution of participants's responses.

When asked to freely comment on potential improvements to the transition scenario, users had several suggestions. Six of the twenty participants suggested some mechanisms for grouping more animated transitions together while animating the scenario. Several participants recommended finding out such mechanisms in order to reduce the total time. Therefore, the lag problem [35] is still important. Participants however recognized that the animation is adequately shown by the animation (statement #10), which is confirmed by several informal comments. Participants perceive less disruption since there is a transition between the UIs before and after adaptation and felt less perturbation. In addition, some participants confessed that they felt more trust in the system when an animation shows the adaptation, but that this could be reinforced by on-demand explanation. They also said that, if they see some transition like that for one or two UIs subject to adaptation, they would trust more the system and ask less the animation in the future.

CONCLUSION

This paper presented a method for showing the adaptation process to the end user by animating its transition scenario from the UI before adaptation until the UI after adaptation is reached. A user study was conducted to determine what the subjective perception of the transition scenario on end users was. This study revealed some advantages (e.g., global appreciating, a perception that the end user disruption and the cognitive perturbation were reduced, increase of trust), but also some shortcomings to be addressed in the future (e.g., enabling faster animation, including on-demand explanation of why this or that adaptation operation has been executed), better capabilities to bypass, group or compact some adaptation operations. In the near future, conducting an experimental study to determine the exact cognitive load of each animated transition and their adequacy with respect to the adaptation operation would be welcome. In this work, we only established such an adequacy based on cognitive psychology and usability guidelines for animation, which is a qualitative approach. A quantitative approach is a desirable for the next step, although different factors may influence these results that are hard to quantify.

ACKNOWLEDGMENTS

The authors would like to acknowledge the support of the ITEA2-Call3-2008026 USIXML (User Interface extensible Markup Language) European project and its support by Région Wallonne DGO6 as well as the FP7-ICT5-258030 SERENOA (Multidimensional context-aware adaptation of Service Front-ends) project supported by the European Commission. The authors would like also to thank the anonymous reviewers, particularly the one who has been the most criticizing, thus triggering new directions to explore and to generalize the work already done.

REFERENCES

1. Baecker, R. and Small, I. Animation at the interface. In: B. Laurel (Ed.), *The Art of Human-Computer Interface Design*. Addison-Wesley, New York (1990).
2. Baudisch, P., Tan, D., Collomb, M., Robbins, D., Hinckley, K., Agrawala, M., Zhao, S., and Ramos, G. Phosphor: Explaining Transitions in the User Interface Using Afterglow Effects. In *Proc. of ACM Symposium on User Interface Software Technology UIST'2006* (Montreux, October 15-18, 2006). ACM Press, New York (2006), pp. 169-178.
3. Bederson, B.B. and Boltman, A. Does Animation Help Users Build Mental Maps of Spatial Information? In *Proc. of IEEE Symposium on Information Visualization InfoVis'99*. IEEE Computer Society Press, Los Alamitos (1999), pp. 28–35.
4. Bladh, T., Carr, D.A., and Kljun, M. The Effect of Animated Transitions on User Navigation in 3D Tree-Maps. In *Proc. of the 9th Int. Conf. on Information Visualization InfoVis'2005*. IEEE Computer Society, Los Alamitos (2005), pp. 297–305.
5. Bétrancourt, M. and Tversky, B. Animation, can it facilitate? *Int. J. of Human Computer Studies 57*, 4 (2002), pp. 247–262.
6. Bezerianos, A., Dragicevic, P., Balakrishnan, R. Mnemonic Rendering: An Image-Based Approach for Exposing Hidden Changes in Dynamic Displays. In *Proc. of ACM Symposium on User Interface Software Technology UIST'2006* (Montreux, Oct. 15-18, 2006). ACM Press (2006), pp. 159-168.
7. Browne, D., Totterdell, P., and Norman, M. (Eds.). *Adaptive User Interfaces*. Computers and People Series. Academic Press, Harcourt Brace Jovanovich Publishers, London (1990).
8. Bunt, A., Conati, C., and McGrenere, J. What role can adaptive support play in an adaptable system? In *Proc.of the 9th Int. Conf. on Intelligent User Interfaces IUI'2004* (Funchal, Jan. 13-16, 2004). ACM Press, NY (2004), pp. 117-124.
9. Cavanagh, P. and Alvarez, G. Tracking multiple targets with multifocal attention. *Trends in Cognitive Science, 9*, 7 (July 2005), pp. 249–354.
10. Chang, B.-W. and Ungar, D. Animation: From Cartoon to User Interface. In *Proc. of ACM Symposium on User Interface Software Technology UIST'93* (Atlanta, November 3-5, 1993). ACM Press, New York (1993) pp. 45–55.
11. Dieterich, H., Malinowski, U., Kuhme, T., and Schneider-Hufschmidt, M. State of the art in adaptive user interfaces. In: Schneider-Hufschmidt, M., Kuhme, T., Malinowski, U. (Eds.),

Adaptive User Interfaces Principles and Practice. Elsevier Science Publishers B.V., Amsterdam (1993), pp. 13–48.
12. Dunn, C. The Use of Real-Time Simulation by Means of Animation Film as an Analytical Design Tool in Certain Spatio-Temporal Situations. *Ergonomics, 16* (1973), pp. 515–519.
13. Escribano, J.G., Manrique, G.M., Haya Coll, P.A. iFaces: Adaptive User Interfaces for Ambient Intelligence. In *Proc. of IADIS Int. Conf. on Interfaces and Human Computer Interaction IHCI'2008* (Amsterdam, July 25-28, 2008). InderScience.
14. Fialho, A.T.S. and Schwabe, D. Enriching Hypermedia Application Interfaces. In *Proc. of 7th Int. Conf. on Web Engineering ICWE'2007* (Como, July 16-20, 2007). LNCS, Vol. 4607. Springer-Verlag, Berlin (2007), pp. 188-193.
15. Findlater, L. and McGrenere, J. Impact of screen size on performance, awareness, and user satisfaction with adaptive graphical user interfaces. In *Proc. of the 26th ACM Conf. on Human Factors in Computing Systems CHI'2008* (Florence, April 2008). ACM Press, New York (2008), pp. 1247–1256.
16. Florins, M., Montero, F., Vanderdonckt, J., and Michotte, B. Splitting Rules for Graceful Degradation of User Interfaces. In *Proc. of 8th Int. Working Conference on Advanced Visual Interfaces AVI'2006* (Venezia, 23-26 May 2006). ACM Press, New York (2006), pp. 59–66.
17. Gajos, K.Z., Czerwinski, M., Tan, D.S., and Weld, D.S.. Exploring the design space for adaptive graphical user interfaces. In *Proc. of 8th Int. Working Conference on Advanced Visual Interfaces AVI'2006* (Venezia, 23-26 May 2006). ACM Press, New York (2006), pp. 201–208.
18. Gajos, K.Z., Everitt, K., Tan, D.S., Czerwinski, M., and Weld, D.S. Predictability and accuracy in adaptive user interfaces. In *Proc. of the ACM Conf. on Human Factors in Computing Systems CHI'2008* (Florence, April 5-10, 2008). ACM Press, New York (2008), pp. 1271–1274.
19. Gardiner, M. and Christie, B. *Applying Cognitive Psychology to User Interface Design*. John Wiley, New York (1987).
20. Heer, J. and Robertson, G. Animated Transitions in Statistical Data Graphics. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (Nov. 2007), pp.1240-1247.
21. Hong, W., Thong, J.Y.L., and Tam, K.-Y. Does Animation Attract Online Users' Attention? The Effects of Flash on Information Search Performance and Perceptions. *Information Systems Research 15*, 1 (2004), pp. 60–86.
22. Huhtala, J., Mäntyjärvi, J., Ahtinen, A., Ventä, L., and Isomursu, M. Animated Transitions for Adaptive Small Size Mobile Menus. In *Proc. of the 12th IFIP TC 13 Int. Conf. on Human-Computer Interaction Interact'2009* (Uppsala, August 24-28, 2009). Lecture Notes in Computer Science, Vol. 5726, Springer-Verlag, Berlin (2009), pp. 772-781.
23. Huhtala, J., Sarjanoja, A.-H., Mäntyjärvi, J., Isomursu, M. and Häkkilä, J. Animated UI transitions and perception of time: a user study on animated effects on a mobile screen. In *Proc. of ACM Conf. on Human Aspects in Computing Systems CHI'2010*. ACM Press, New York (2010), pp. 1339–1342.
24. Jameson, A. Adaptive Interfaces and Agents. In: Jacko, J.A., Sears, A. (Eds.), *Human–Computer Interface Handbook*. Lawrence Erlbaum, Mahwah (2003), pp. 305–330.
25. Kaber, D.B. and Endsley, M.R. The effects of level of automation and adaptive automation on human performance, situation awareness and workload in a dynamic control task. *Theoretical Issues in Ergonomics Science 5*, 2 (2004), pp. 113–153.
26. Kerren, A, Stasko, J., Algorithm Animation, Introduction of Software Visualization, State of the Art Survey. LNCS, Vol. 2269. Springer-Verlag, Berlin (2002), pp. 1-15.
27. Lavie, T. and Meyer, J. Benefits and costs of adaptive user interfaces. *Int. J. of Hum.-Comp. Stud.*, *68* (2010), pp. 508–524.
28. May, J., Dean, M.P., and Barnard, P.J. Using Film Cutting Techniques in Interface Design. In *Human-Computer Interaction*, Vol. 18, Lawrence Erlbaum Ass. (2003), pp. 325–372.
29. Rensink, R.A., O'Regan, J.K., and Clark, J.J. To see or not to see: the need for attention to perceive changes in scenes. *Psychological Science 8*, 8 (1997), pp. 368–373.
30. Rogers, S., Fiechter, C.N., and Thompson, C. Adaptive user interfaces for automotive environments. In *Proc. of the IEEE Symposium on Intelligent Vehicles Dearborn*. IEEE Computer Society Press, Los Alamitos (2000), pp. 662–667.
31. Schlienger, C., Dragicevic, P., Ollagnon, C., and Chatty, S. Les transitions visuelles différenciées : principes et applications. In *Proc. of IHM'2006* (Montréal, 18-21 April 2006). ACM Int. Series, Vol. 133 (2006), pp. 59–66.
32. Schlienger, C., Conversy, S., Chatty, S., Anquetil, M., and Mertz, Ch. Improving Users' Comprehension of Changes with Animation and Sound: An Empirical Assessment. In *Proc. of Interact'2007* (Rio de Janeiro, 2007). LNCS, Vol. 4662, Springer-Verlag, Berlin (2007), pp. 207–220.
33. Sherman, R., Alpert, J.K., Karat, C., Carolyn, B., and Vergo, J. User attitudes regarding a user-adaptive e-commerce web site. *User Modeling and User-adaptive Interaction 13*, 4 (2003), pp. 373–396.
34. Smith, S.L. and Mosier, J.N. Design guidelines for the user interface software. Technical Report ESD-TR-86-278 (NTIS No. AD A177198), U.S. Air Force Electronic Systems Division, Hanscom Air Force Base, Massachusetts (1986).
35. Stasko, J. Animation in User Interfaces: Principles and Techniques. In *Proc. of User Interface Software '93*, pp. 81–101.
36. Sukaviriya, P. and Foley, J. Coupling a User Interface Framework with Automatic Generation of Context Sensitive Animated Help. In *Proc. of ACM Symposium on User Interface Software Technology UIST'90* (Snowbird, Oct. 1990). ACM Press, New York (1990), pp. 152–166.
37. Teevan, J., Dumais, S.T., Liebling, D.J., and Hughes, R. A Longitudinal Study of How Highlighting Web Content Change Affects People's Web Interactions. In *Proc. of ACM Conf. on Human Aspects in Computing Systems CHI'2010*. ACM Press, New York (2010), pp. 1353-1356.
38. Thomas, B.H. and Calder, P. Applying Cartoon Animation Techniques to Graphical User Interfaces. *ACM Trans. on Computer-Human Interaction 8*, 3 (Sept. 2001), pp. 198–222.
39. Tucker, J.B. Computer Graphics Achieves New Realism. *High Technology* (June 1984), pp. 40–53.
40. Vanderdonckt, J. and Gillo, X. Visual Techniques for Traditional and Multimedia Layouts. In *Proc. of 2nd ACM Workshop on Advanced Visual Interfaces AVI'94* (Bari, 1-4 June 1994), ACM Press, New York (1994), pp. 95–104.
41. Ware, C., Neufeld, E. and Bartram, L. Visualizing Causal Relations. In: *Proc. of IEEE Symposium on Information Visualization InfoVis'99*. IEEE Computer Society Press, Los Alamitos (1999), pp. 39–42.