# Animated Transitions for Empowering Interactive Information Systems

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

In Memoriam: Paul Vanderdonckt (8 May 1933 – 31 March 2012)

***Abstract*—Animated transitions are widely used in many different domains of human activity, ranging from cartoons and movies to computer science for powerfully conveying a message more effectively and efficiently about a phenomenon of interest. This paper reviews a series of techniques for defining, analyzing, and exploiting animated transitions in different types of interactive information systems. A general conceptual model is provided that explicitly links a model of an interactive information system, its model elements and relationships to animated transitions in order to adequately reflect any change of the model into animated transitions. Two instantiations of this conceptual framework are discussed: animated transitions for representing adaptation of the graphical user interface of an interactive system, along with its implementation; and animated transitions between user interface views during development life cycle.**

***Keywords- animation, animated transition, context-aware adaptation, information system, model-based approach, model-driven engineering, web engineering***

## I. Introduction

Animation is defined in the Free Merriam Webster dictionary as follows [http://www.merriam-webster.com/dictionary/animating]: “Animation is the act of animate, which is: to give spirit and support to; to give life to, to give vigor or zest to; to move to action; to make or design in such a way as to create apparently spontaneous lifelike movement, to produce in the form of an animated cartoon” [51]. Indeed, the verb “animate” come from its Latin root *animatus*, past participle of *animare*, which means “to give life to”. Model animation could therefore be interpreted as the act to give life cto a model in such a way as to create natural movement [48].

An *animated transition* is a particular animation intended to graphically ensure a smooth transition between an initial state of a system, whether it is computer-based or not, and a final state of this system [15,16]. An animated transition is typically aimed at representing a transition between two states by an animation so as to give life to this change of state. In the domain of graphics for instance, an animated transition could depict how a particular graphic is transformed into another one [25] over time to depict the change of the data over time on time lines [20], such as demographic evolution or cartoons [46,47].

In the context of this paper, we consider that an animated transition is a particular animation intended to ensure a smooth transition between two states of an interactive information system. Animated transitions [2,6,44] in interactive systems are aimed at conveying to the end user a transition between states, views or scenes [16,18], e.g., to foster a smooth transition between two scenes [6], menus [27] or images [28]. Animated transitions improve feedback on users’ actions [24], notify display changes [40], and improve situation awareness in a distributed environment [39]. Animated transitions exist in many different domains of application such as, but not limited to: air-traffic control [31,39], data and decision making [14], documentation [12,14], graphics, information visualization, map navigation [6], mobile computing [27,28], model-based design [18], model-driven engineering [35,49], system simulation [19], textual documents [14], user interface [11], dependable systems [33], visual design [48], and zooming interfaces [42]. Before examining the implications of animated transitions, a conceptual framework is introduced that identifies the main concepts of interest in order to specify, design, and implement animated transitions for an interactive system.

## II. A Conceptual Model for Animated Transitions

Figure 1 graphically depicts an overview of our conceptual model [1] for animated transitions in interactive systems. It could still be applied in principle in any methodological approach for information systems engineering.

A *model* of an interactive information system consists of a series of model elements that characterizes the model in itself. It is then built as a hierarchical decomposition of model elements into several refined sub-concepts appearing at the decomposition sub-layers [34,49]. Any *model element* of interest can then be in turn captured by an identifier, a name, a definition, and properties of interest gathered in a given characterization. Model elements are linked together via *model relationships* of a certain type and whose statement could be provided. Typical relationships include: decomposition, abstraction, reification, reflection, cross-cutting, translation, and mappings (e.g., mappings resulting from applying model transformations in model-driven engineering). Any model, model element, or model relationship may consist of *attribute*, which is characterized by a name, a definition, a data type, a definition of its domain of values, and potential *constraints*. Usually, a model is represented through a *model view*, that could be graphical, textual or both (i.e., bimodal). Each view is of the following type: *conceptual* (if the view reflects an abstract model), *internal* (if the view reflects the code that is internal to the interactive system) or *external* (if the view reflects the code in another way than its internal representation). Each view could hold different levels of fidelity (e.g., low-fidelity, mid-fidelity, high-fidelity) and different levels of details (e.g., low, medium, high) depending of the context of use.

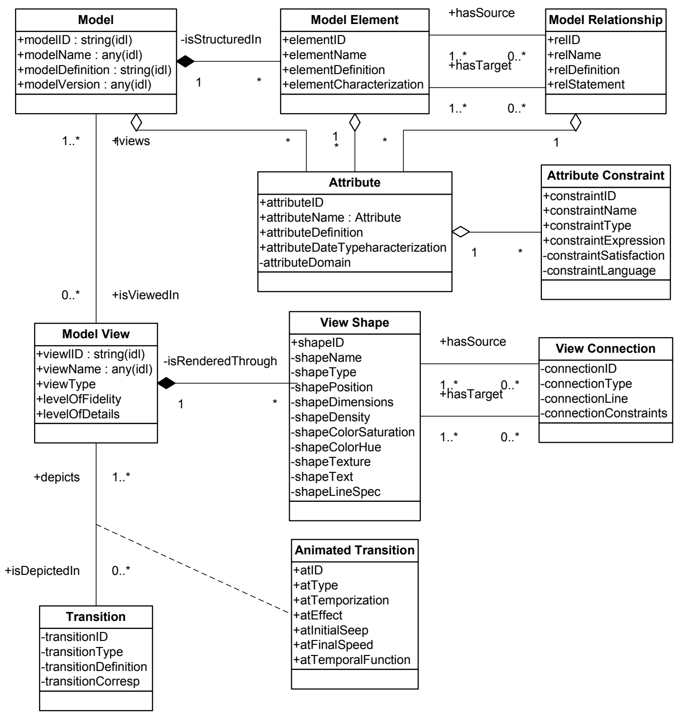


**Figure 1**. Definition of a general-purpose conceptual model.

A model view is rendered through a series of *view shapes* (e.g., line, polygon, graphic, text or any combination of them), each shape being characterized by different attributes (e.g., position, length, height, density, texture, color). These attributes could be selected depending on the level of precision required by the model view: Figure 2 lists major attributes by decreasing order of precision in order to represent a model attribute depending of its type (e.g., quantitative, ordinal or nominal). Figure 2 shows that in all cases the position of view elements is always the most precise way to represent model attributes. A transition between views, whether they are for the same model or for different models, is then ensured through an animated transition that maps the view elements (corresponding to respective model attributes) of the different model views. For example, a transformation [34,35,36] from a CIM model to a PIM model could be ensured through animated transitions and similarly from a PIM model to a PSM model, and from a PSM model to code. Similarly in Human-Computer Interaction (HCI), animated transitions could be used from task and concepts to abstract user interface, from abstract user interface to concrete user interface, and from concrete user interface to final code. Since it is not mandatory to process the development life cycle through all these levels, it is also possible to imagine animated transitions between non-subsequent levels of abstraction, such as from CIM to PSM, PIM to code, or CIM to code. In HCI, it could be from task and concepts to concrete user interface, from abstract user interface directly to final code.

Depending on the type of model attribute, on the type of corresponding view shape (e.g., a rectangle, a circle, a line), different animated transitions could be imagined such as: text-to-text, text-to-position, text-to-dimension, text-to-color, text-to-shape or reciprocal animated transitions [16].

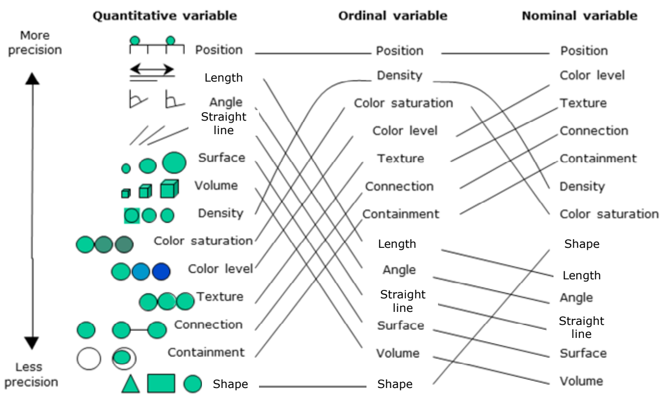


**Figure 2**. Representation style depending on the level of precision and variable type (adapted from [30]).

## III. An Instantiation of the Conceptual Framework

In this section, a particular instantiation of the conceptual framework introduced in Section 2 is provided to the area of adaptation of a Graphical User Interface (GUI) of a web application, with some review of related work. Adaptation of a web application typically falls into two categories depending on who is in control of the adaptation process [13]: *adaptivity* when the web application is responsible for adapting itself [9,10], *adaptable* when the end user is responsible for adapting the web application by means provided by this application [7], and *mixed-initiative* when the responsibility is shared between the web application and the end user [13]. Adapting a web application obviously represents an important opportunity for several stakeholders [8,10,37,43]:

- *End users*: the adaptation is in principle always undertaken in order to improve the global quality of a web application for the ultimate benefit of the end user, preferably depending on the context of use so that to obtain a context-aware (or context-sensitive) web application. Adaptation can be effectively and efficiently applied to a wide variety of human activity domains. Potential benefits include improving [29]: usability, user experience, navigation, task completion time,…
- *Designers*: several methods exist that support designers in conducting web engineering (e.g., [23,37,43] all provide extensive and interesting comparison and survey of major web engineering methods), but only some of them support adaptation explicitly, with varying levels of granularity, context-awareness [43].
- *Developers*: adaptation can be developed for many different types of web applications ranging from simple HTML pages [8] until Rich Internet Applications (RIAs) [22]; several User Interface Description Languages (UIDLs) [49] could provide developers with developing facilities for producing various interfaces for various contexts of use from a set of models [34]; the complexity of software architectures for supporting adaptation could vary depending on the sophistication of context-awareness [4], thus making it more complex for developers [50,52].

Beyond the aforementioned aspects, end users may suffer from several intrinsic drawbacks of adaptation that are hard to overcome, one of the most important being the *end user disruption* [13,29]: there is a discontinuity between the web application before and after the adaptation process, there is nothing between the initial and final states, thus inducing a cognitive de-stabilization that may prevent end users from accepting, using, and benefiting from the adaptation. We are intended to demonstrate by animated transitions how adaptation has been achieved on a web application in such a way that any web engineering method or adaptation development practice could benefit from this process to minimize the disruption.

### *A. Related Work in Adaptation of User Interfaces*

*Adaptation* has been subject to extensive investigation that leads to recognizing a series of benefits vs. costs [8,9,10,13, 29]: adaptive UIs are able to optimize task completion time and rate, they induce a positive impact on accuracy, human performance, predictability, situation awareness, and cognitive workload. Adaptivity has also been revealed effective when the UI should be adapted to the constraints imposed by any loss of screen resolution [32], like on mobile devices [28]. Browne *et al.* [7] as well as Dieterich *et al.* [13] are among the first surveys of adaptation processes at large, not just for web applications; They demonstrate that the coverage of four adaptation steps by adaptation methods is largely varying: *initiative* (who is taking the initiative of adaptation), *selection* (who selected the appropriate adaptation operations), *decision* (who decides to apply the adaptation operations), and *execution* (who executes the adaptation operations). In this paper, we assume that selection and decision have already been on what adaptation operations should be executed, but we prefer to demonstrate these operations during their execution. Brusilovsky *et al.* have introduced a taxonomy of adaptation operations for hypermedia applications that have been refined and expanded several times [8]: adaptation can be applied to any element for either presentation or dialog (including navigation) or both. Again, this paper is not intended to contribute to defining adaptation operations (since it is properly done in [8]), but to rely on them to map them onto appropriate animated transitions.

*DiffIE* [45] highlights web page contents that have been updated since last visit, thus inducing a positive impact on how people interact with the web page and understand their contents, and perceived these contents as dynamic (Figure 3a). It is particularly appreciated by end users to identify what has changed in a web page in order to handle the most recent data.

*Phosphor widgets* apply afterglow effects [3] to foster some visual reminiscence of changes of values of widgets on a screen (e.g., the value change of a slider – Figure 3b). For instance, if the current value of a widget has changed, an animated transition is applied to this widget to reflect this change. This does not stop the current end user's task, but may attract her attention [26] to a focus that is not the current one.

*Differentiated transitions* [40,41] are animated transitions explaining a dynamic process over time, e.g., an animated transition animates the transfer time, the network bandwidth, and the file size when a file is transferred from one location to another (Figure 3c).

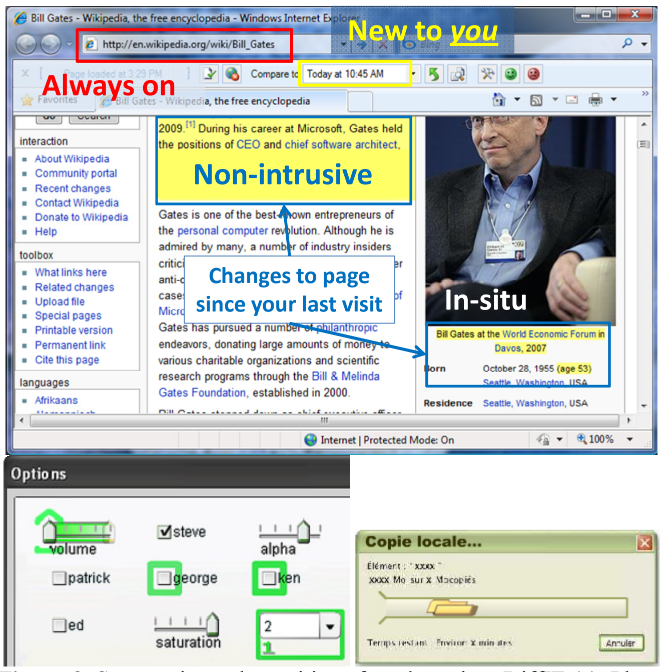


**Figure 3.** Some animated transitions for adaptation: DiffIE (a), Phosphor widgets (b), Differentiated transitions (c).

Fialho & Schwabe [21] enrich the user experience of web applications by applying a visual rhetoric as a way to set the effects presented by animation, as well their sequence and duration. In order to capture the dynamic aspects of a widget, an Abstract Widgets Ontology (AWO) was extended to include the following classes: *Transition* (for representing a state change), *RhetoricalStructure* (for animating a transition or in response to an event), and *Decoration* (for animating a widget change). The Decoration class is further refined into the following elements: *InsertElement* (for introducing a new element), *RemoveElement* (for removing an element from the destination state), *MatchElements* (for matching the parameters of an element in the current state and another in the destination state), *TradeElements* (for performing a transformation of an element in the current state into another in the destination state), and *EmphasizeElement* (for highlighting an element subject to an action). This work is the closest to the one presented in this paper: while it represents changes in the model based on a visual rhetoric, our paper focuses on animated transitions for web adaptation.

Programming animated transitions by hand is possible but complex without a framework [50,52], for instance in JavaScript for HTML, using some APIs or libraries, but this task assumes a development effort whose cost could be considered far superior to the benefit of using the animations. This is why an animation engine will be considered in this paper in order to reduce programming efforts to its minimum. For example, the LWUIT library [38] supports the implementation of various animated transitions from one form to the next one to be displayed, based on mechanisms provided by the interface *Animation* (which renders widgets animatable) and the classes *Motion* (which enables object motion), *Transition* (which ensures animated transition from one form to another), *CommonTransitions* (which provides Slide and Fade visual effects), and *Transition3D* (which provides 3D visual effects such as Cube, Fly in, and Rotate). Such a library indeed reduces the development cost by offering basic classes for developing animated transitions. But a significant effort remains for connecting these classes to adaptation process. It makes sense to elaborate a model-based approach for dealing with animated transitions so that each animated transition is properly assigned to an adaptation operation until ultimately there is no more any development effort required to deploy the solution. The next section introduces the conceptual model of animated transitions for depicting an adaptation of a graphical user interface for a web application.

### *B. Conceptual Model of Animated Transitions for Adaptation of a Graphical User Interface*

Figure 4 provides a UML Class Diagram that gives an overview of the conceptual model used for demonstrating web adaptation operations by executing animated transition. The Concrete User Interface (CUI) [49] of a web application is realized so that it can be submitted to adaptation operations that are in turn mapped onto appropriate animated transitions. These various aspects are further detailed in the next sub-sections.

#### *1) Definition of the Concrete User Interface*

A *Concrete User Interface* (CUI) is defined as a model capturing abstractions of a Graphical User Interface (GUI) for a given interaction modality (e.g., the graphical modality), but independently of any implementation technology. Several similar models exist to describe the UI of a web application [37,43]. User Interface eXtensible Markup Language (UsiXML) [49] was selected in our case in order to model a CUI for various reasons: availability, compliance with the Cameleon Reference Framework (CRF), openness, definition of its semantics via UML Class Diagrams, availability of concept definitions, and transformation between models. But other UIDLs could be equally selected provided that equivalent concepts exist as counterparts. At first glance, there is no restriction of the present work to be ported to another UIDL or web engineering methods as long as they already incorporate equivalent abstractions. Since it is not the goal of this paper to detail a UsiXML CUI, we refer to www.usixml.org for full documentation. A brief overview is only given here: a CUI consists of a hierarchical decomposition of concrete containers, perhaps arranged together via spacing individual components. Each container is in turn decomposed into concrete individual component, an abstraction for widgets with several subclasses.

Our reference model here is the CUI model that captures an abstraction of GUI widgets of a web application independently of any implementation. The corresponding model view is defined by a Look & Feel (L&F) of these widgets. Therefore, each view shape is regulated by the rendering of the corresponding widget in the computing platform used. This does not assume that there is a restriction on these representations. Simply, we need to ensure an animated transition of view shapes corresponding to widgets subject to adaptation. For instance, the view shape corresponding to an edit field belonging to the model consists of a rectangular area, whose style is defined through attributes such as line style, foreground and background colors, textures, etc. If this widget is adapted into another one, say a combination box, then an animated transition could show how this view shape is transformed into another view shape, corresponding to a combination box.

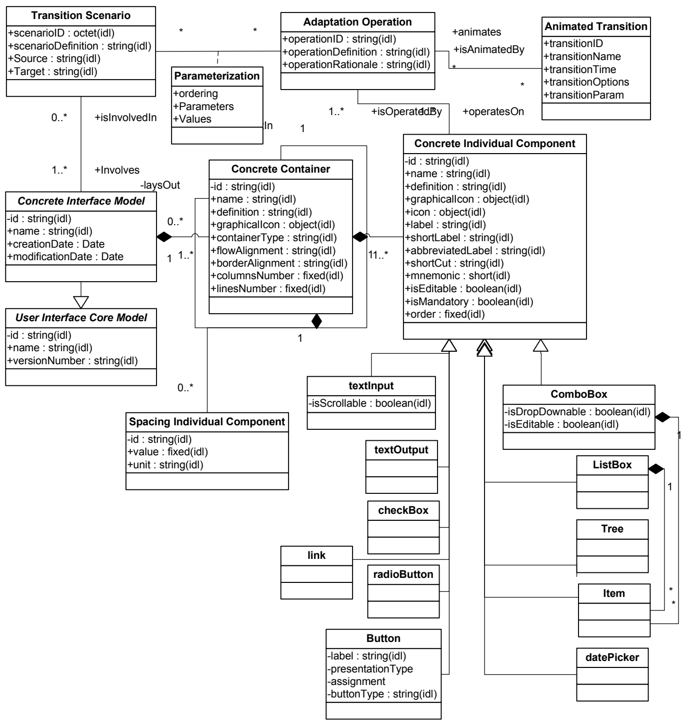


**Figure 4.** The conceptual model used for demonstrating web adaptation.

#### *2) Definition of Adaptation Operations and Transition Scenario*

An *Adaptation operation* is hereby defined as any transformation operated on any web page element (modeled as a concrete individual component) in order to adapt the page for the ultimate benefit of an end user interacting in a certain context of use, that is itself characterized as a triple (*User*, *Platform*, *Environment*) [32]. 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.

Each adaptation operation [15] produces a *transient web page being adapted* (Figure 5), which consists in an intermediary web page stage during adaptation. Usually, the end user does not perceive any of these transient web pages since they are subject to transformation. End users are only presented with the initial and the final web pages, which cause the end user disruption and the cognitive perturbation. The whole sequence of adaptation operations conducted for the web page adaptation is called the *transition scenario* that involves a wide spectrum of adaptation operations which fall into six categories [15]:

1. *Resizing operations*: are aimed at changing a widget size in order to optimize screen real estate, aesthetics, and visual design. For instance, an edit field could be expanded or reduced in order to accommodate screen resolution.
2. *Relocating operations*: are aimed at changing a widget location in order to accommodate constraints imposed by the context of use, like screen resolution. For instance, “Ok”, “Cancel”, and “Help” are relocated to the bottom of a web page.
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. 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. For instance, cropping or reformatting image processing techniques could produce an image suitable for the new context.
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.
6. *Global replacement*: is aimed at representing the results of a general-purpose adaptation algorithm that cannot be decomposed into a series of elementary adaptation operations. For instance, a semantic algorithm can change the contents.

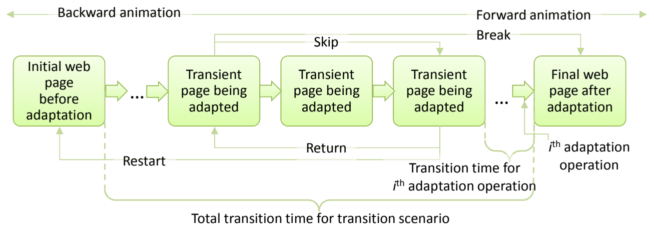

Backward animation
Forward animation
Skip
Break
Initial web page before adaptation
Transient page being adapted
Transient page being adapted
Transient page being adapted
Final web page after adaptation
Return
Restart
$i^{th}$ adaptation operation
Transition time for $i^{th}$ adaptation operation
Total transition time for transition scenario

**Figure 5.** The transition scenario between the initial web page and the final web page.

Therefore, a *transition scenario* is hereby defined as a series of adaptation operations between a source and a target CUI model (involving one or many containers and individual components) based on particular parameterization (Figure 4). Indeed, a single adaptation operation could be performed on a single page element in isolation (e.g., resizing an individual or a compound widget) or several page elements concurrently (e.g., resizing a group of aligned edit fields)

In order to capture a transition scenario in a logical way, a catalogue of adaptation operations addressing the six aforementioned categories is provided. This idea is inspired from Web Adaptation Language (WAL) [9], which consists of an imperative language for describing the adaptation logic of a web application that is parsed by an adaptation engine. The positive consequence of this is that the adaptation logic could be defined independently of its development. 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. Supported adaptation operations are as follows:

SET <Element.property> TO {value, percentage}: assigns a value to any element property or a percentage of the actual value. For instance, SET "Button_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%. As a shortcut, EXPAND "pushButton_1.height" OF 10 represents the same operation while CONTRACT "pushButton_1.height" OF 10 represents the inverse operation.

DISPLAY <Element> [AT x,y]: displays any identified element at a *x,y* location where *x* and and *y* are integer positions (e.g., in characters or pixels). For instance, DISPLAY "Button_1" AT 2,3 displays an identified push button at coordinates 2,3 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 any element to a new location indicated by its coordinates *x* and *y*, possibly in a fixed amount of steps (by default, one).

CHANGEBOX <Element> TO <Container.x,Container.y> [IN n steps]: changes any element from a previous container to a new container within which *x* and *y* specify its coordinates, possibly in a fixed amount of steps. It is possible to change the size of containers by the following operations: CHANGECOLUMN <Container> BY {value} and CHANGEROW <Container> BY {value}.

ROTATE <Element> BY {value} [IN n steps]: rotates any element by a certain amount of degrees, possibly in a fixed amount of steps.

REPLACE <Element1> BY <Element2>: replaces any element Element1 by another one Element2. Sometimes the replacement element could be determined after an adaptation algorithm (6th category above), 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 <Image-Algo:URL>.

DISTRIBUTE <Elements> INTO <Containers> [BY <Distrib-Algo:URL>: computes a distribution of a series of Elements into a series of Concrete Containers, possibly by calling an external algorithm, local or remote.

In the above definitions of adaptation operations, only one web page element is provided as parameter at a time. Obviously, an adaptation operation could consider several elements together. For this purpose, a selection mechanism is introduced that defines the scope of possible elements as a parameter. A *Selector* consists of a definition of the web page types to which the adaptation operation is applied. Four major types of selector may affect <Element> or <Elements> fields in the definitions:

1. universalSelector: applies the adaptation operation to all UI elements belonging to the current GUI of concern. For instance, SET "universalSelector.backgroundColor" TO "Grey" will change the background color of the entire GUI into grey.
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=Button" TO "lightGrey" will set the foreground color of all push buttons of the current web

page 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.

The above catalogue of adaptation operations mainly cover simple attributes of elements included in a web page. These attributes fall into two categories: attributes that have a visual impact (e.g., color, size) and attributes that do not have any visual impact (e.g., change of default value). This catalog is mainly inspired by adaptation operations defined in Brusilovsky *et al.* taxonomy [8], but is not expected to cover all possible adaptation operations possible. The next section explains how one or many animated transitions (along with their parameters) could animate a particular adaptation operation (Figure 2). Attributes with visual impact are therefore first-class citizens to consider associated animations, but attributes with non-visual impact could be equally considered. However, an animated transition is probably not the best way to convey such a change, thus motivating the need for considering some generalization of the approach with other means than animated transitions.

## C. *Mapping Adaptation Operations onto Animated Transitions*

Table 1 provides an overview of major animated transitions gathered from the literature [2,6,44] and are classified according to their goals. More sophisticated animated transitions could be also considered, but they may induce some 'lag' problem [44] due to their cognitive load and/or time required to run the animated transition.

Table 2 considers a sub-set of possible animated transitions for each major adaptation operation. When alternate candidates exist for one adaptation operation, we justify the priority assigned to each animated transition with respect to its goal. This priority is maintained in a configuration file that can be edited separately. We now give some example of some adaptation operations.

*Label contraction*. When a web page should be contracted, e.g., for being viewed on a small screen, it needs to be compacted as much as possible. For this purpose, label contraction consists in replacing the long label identifying an element (typically, an edit field or a list box) by a shorter version, if any. For instance, "Department" may be contracted successively into "Dept.", a frequently used abbreviation, or "Dep." if this is acceptable for the end user. This process is captured by: SET Label.name TO Label.ShortName. Since it is more important to convey the contraction process than the replacement of the label by its short label, "Horizontal scroll from left" erases the long label from its end to the beginning while replacing its by its short version, thus giving the illusion of contraction.

| **Icon** | ***Name*: definition** |
|---|---|
| ← | *Horizontal scroll from right*: to display the next element from a sequence of elements |
| → | *Horizontal scroll from left*: to display the previous element from a sequence of 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 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.** Major animated transitions.

*Label expansion*. When a web page should be displayed on a large screen, like a public display or a wall screen, there is room to have the full version of the label. This process is captured by: SET Label.name TO Label.LongName and is animated by a "Horizontal scroll from right" to convey the illusion of expansion.

*Edit field move*. When reformatted, a web page often involves moving elements within a box. For instance, an edit field could be moved horizontally, vertically or diagonally: MOVE textInput_1 TO 1,10. According to Table 2, the ideal animated transition would be an object motion according to a motion path that is computed at run-time between the two centers and a real movement operated between these two locations. The drawback of this animated transition is its complexity, its possible overlapping of other elements if intercepted by the motion path, and the time required to animate the process, thus posing the 'lag' problem again upfront. Therefore, we prefer to apply other animated transitions depending on the movement type, such as the following ones:

*Horizontal move*: three alternate transitions could be considered: (a1) erase old element first, present the new after; (a2) present the new first, erase old after; (a3) erase old element while presenting the new one simultaneously. We justify our decision here based on the cognitive load induced by the animated transition: *low* when one element is involved at a time in a logical order (a1), *medium* when one element is involved, but not in a logical order (a2), or *high* when two elements are involved simultaneously, thus causing two foci of attention for the end user. "Bam door close" is chosen for the first element while "Bam door open" is chosen for the second one.

| **Adaptation operation** | **Animation family, animated transition with justification** |
|---|---|
| **SET** that modifies the length of any element into a larger value (absolute or relative) | *Horizontal scroll/wipe from left*: 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 any element into a larger value (absolute or relative) | *Vertical scroll/wipe from bottom*: this operation minimizes the visual change since only the right part resulting from the enlarging is changing |
| **DISPLAY** that displays a new element at a certain position | *Uncover, Box out, or Iris open*: 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 an element from a certain position | *Cover, Box in, or Iris close*: 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 an element by another one | *Bam door open*: this operation affects the entire visual aspect of the previous one and the new one. |
| **DISTRIBUTE** that computes a distribution of a series of Elements into a series of Containers | *Bam door open or Iris open*: these operations enable the visualization of an entire group at once, instead of showing every little display change individually |
| **MOVE** that moves an element to a new location indicated by its coordinates *x* and *y*, possibly in a fixed amount of steps | Ideally, the 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 problem [24]. Therefore, we preferred to adopt a disappearing of an element from its original location and an appearing to its target location. Depending on these locations, vertical, horizontal or diagonal replacements are selected. For instance, when an 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 web page should only move linearly (either vertically or horizontally), a vertical/horizontal scroll is selected instead. |

**Table 2.** Mapping table between adaptation operations and animated transitions.

*Vertical move*: the reasoning is achieved by similarity to the horizontal move. “Bam door close” and “Bam door open” for the two locations.

*Diagonal move*: could be achieved by combining the above transitions.

*Widget move*. Individual components (as represented in Fig. 2) are equally submitted to the same animated transitions, e.g., MOVE datePicker_2 TO 2,36.

*Link promotion*. Link promotion [9] consists of putting upward a link from a group of links in order to reflect its change of state, such as recency, frequency of use, recommendation. According to Schlienger *et al.* [40], the best animated transition would be to open a free slot for the promoted link while pushing down the rest of the links, then move the promoted link up. In order to foster simplicity and avoid the ‘lag’ problem [44], this could be captured alternatively by the following sequence of adaptation operations: UNDISPLAY Link_4 AT 4,1; MOVE Link_1,Link_2, Link_3 TO 2,1;3,1;4,1 ; DISPLAY Link_4 AT 1,1. This solution does not induce any overlapping.

*Link demotion*. Link demotion [9] consists of putting backward a link from a group of links in order to reflect its change of state, such as obsolescence, reduction of interest or decrease of recommendation level. Link demotion could be again replaced by: UNDISPLAY Link_1 AT 1,1; MOVE Link_2,Link_3, Link_4 TO 1,1;2,1;3,1 ; DISPLAY Link_1 AT 4,1.

*Display web element*. If a new widget should appear, e.g., though a DISPLAY listBox_3 AT 2,2, the following animated transitions could be considered: Appear (fast, but not very progressive), Bam Door Open (slow, but progression is uniform), or Dissolve in (slowest, but progression is random and independent of the widget shape). Due to these considerations, we rank them in this priority order: “bam door open”, “appear”, and “Dissolve in”. The UNDISPLAY operation is similar.

*Widget substitution*. Substituting a widget by another one often arises when contextual conditions are considered, e.g., REPLACE inputText_1 BY comboBox_1 could be executed as soon as the value domain is known. The following animated transitions could be considered: Discover left (the old widget is replaced by the new one from left to right), Wipe right (the old widget is wiped from right and replaced by the new one simultaneously), or Box out (the old widget is globally replaced by the new one, whatever the widget types and shapes are). Other more cognitively expensive animated transition could be considered like Swirl, but the visual impact is stronger, which may be not desirable given that several animated transitions should occur.

Note that *DiffIE* [45] could be considered as a particular case of our conceptual framework: new textual contents could be animated by REPLACE label BY newLabel, SET newLabel.BackgroundColor TO “Yellow”.

## *D. Implementation of the Adaptation Animation Process*

This section motivates and describes the implementation of an Adaptation Animator that relies on the conceptual model introduced before to run a transition scenario (Figure 6).

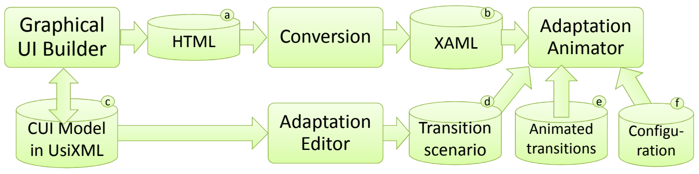


**Figure 6.** The software architecture for the animation process.

### 1) *Software architecture*

A *Graphical User Interface Builder*, i.e. GrafiXML [32] was developed that exports the results of the design phase into a Concrete User Interface (CUI) model stored in a UsiXML file (Figure 6c) and can automatically generate HTML code (Figure 6a) corresponding to this CUI model. This initial HTML is then incorporated in the web application and is converted into Microsoft XAML UIDL (Figure 6b) by XSLT style sheets. The GUI builder today consists of about 21,600 LOC implemented in Java 1.5 with various libraries (e.g., Castor, Jakarta, Jdom, LiquidINF, Looks, Xalan, and Xerces).

An *Adaptation Editor* enables the designer to apply any adaptation operation defined in the catalogue on the initial web page in order to obtain the final one after adaptation. For this purpose, control panels are provided to let the designer applying any adaptation operation desired by a set of rules corresponding to the five first categories introduced. Any such executed operation is added in the log file of the Transition Scenario (Figure 6d). The Adaptation Editor has been implemented as a Java plug-in for GrafiXML and todays consists of about 1,300 LOC.

The *Adaptation Animator* then parses the transition scenario on the XAML file by animating *the* transition contained in the definition file, whose priority is based on the configuration file (e.g., when several alternative animated transitions could occur). The Adaptation Animator then renders the graphical animation while enabling the end user to control it with different actions based on Figure 5, such as "go next", "go previous", "restart", "suspend/resume", "go to end".

The adaptation animator today consists of 1,100 LOC implemented in Microsoft Expression Studio. This environment has been selected for the following reasons: it is already equipped with XAML, a XML-compliant UIDL for CUI; all elements of a XAML-compliant GUI are vector-based and logical operations could be then performed on them in a logical way since they are treated as simple vectorial graphical objects, some animated transitions of Table 2 are already built-in with some options (like speed [17], 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 aforementioned operations and Expression Blend for the Animator itself.

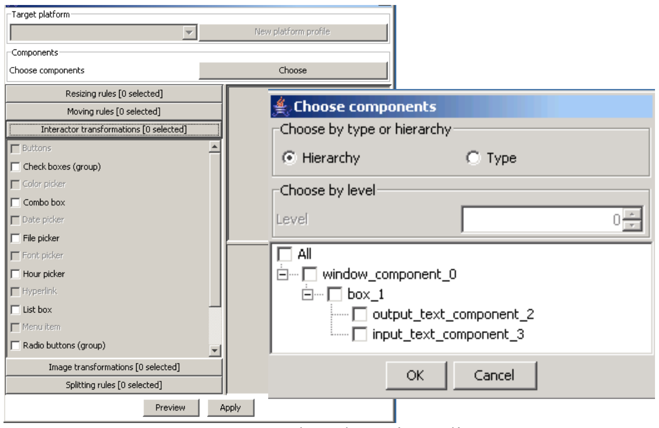


**Figure 7.** The adaptation editor.

### 2) *Possible paths*

Figure 7 depicts that several paths are possible in order to obtain the rendering of animated transitions on a web page. The classical path would be to produce the HTML code from a CUI model, thus forcing the designer to rely on such an editor. It could be imagined though that any HTML web page could be converted to XAML (Fig. 4b) so as to benefit from the adaptation animator. In this case, an HTML page could be also reverse engineered into a CUI model stored in UsiXML [49], thus entering the external page in the loop. In the same way, any XAML-based UI could be also transformed into UsiXML thanks to XSLT style sheets, thus enabling designers to consider many entry points.

Here are some links of videos capturing the resulting animation for two web pages:

- A simple login page: http://www.youtube.com/watch?v=2mvvTL1yYBA
- A polling system: http://www.youtube.com/watch?v=2ZViwktUbhU
- An address book: http://www.youtube.com/watch?v=8MxokT-GCMY

## E. *Generalization to other interaction modalities*

Animated transitions were considered as the focus of this paper in order to demonstrate how adaptation operations have been conducted. Animated transitions are of course not the only possibility. Bernsen's taxonomy [5] identified 24 potential unimodal output modalities of interaction for rendering some information to the end user. Animated transitions are just one of them. Based on this taxonomy, some other significant interaction modalities could be considered that require further investigation:

- *Textual rendering*: a textual statement explains the rationale behind an adaptation operation or algorithm, which is particularly appropriate when non-visual attributes are of concern. This textual statement could be rendered immediately in terms of the command language resulting from the EBNF grammar introduced previously, in terms of log files with time stamping (explaining which operation was executed when), or in natural language by generating automatically a sentence corresponding to the command language, also based on reasoning explanation.

- *Graphical rendering*: an icon assigned to each adaptation operation or an image showing the main steps of the adaptation process could be used.
- *Animation*: a dedicated animation for each adaptation operation, a graphical morphing between the web page before and after adaptation, or an avatar pointing to the web page region that is subject to adaptation and explaining it.
- *Sound*: a short vocal synthesis of the adaptation operation or a sound expressing the progression as used in [41].
- *Hypermedia linking*: a link could be provided to the end user to access a knowledge base containing the definitions of adaptation operations, a reference–based training could provide on-line access to documentation, help messages, Frequently Asked Questions (FAQs), a discussion forum or even a physical person.
- *Combination*: several of the above techniques could be considered together as long as the cognitive load does not become prohibitive and as long as the 'lag' problem [44] could be overcome by end user control.

### F. *Discussion of Animated Transitions*

In this section, we presented a conceptual model for mapping web adaptation operations onto a set of potential animated transitions (an instantiation of the general conceptual model introduced in Section 2) that is used by the Transition Animator, a program that executes a transition scenario based on these animated transitions. It is expected that this demonstration will reduce the end user disruption by establishing a visual bridge between the web page before and after adaptation, thus supporting a transition during the adaptation execution [13]. User actions can speed up or slow down the animation process when they want, thus providing them with a mean to reduce the 'lag' problem [44]. Preliminary results suggest the following conclusions: in the beginning, end users require the demonstration to be executed step by step in order to understand the full adaptation process. This could be augmented by providing them with some rationale explaining them the underlying guideline, rule or heuristic that has been used for this purpose. This rationale could be based on theory of argumentation and associated to any animated transition so that end user could ask explanation on-demand exactly as in Artificial Intelligence (AI): end users tend to accept the reasoning of an expert system as soon as they can browse the knowledge used for the reasoning. It is expected to have a similar conclusion here. What has been observed is that after some time, end users prefer to speed up the process, not just to reduce the 'lag' problem, but because they feel convinced that there is indeed an obvious reasoning behind the adaptation that could be demonstrated on-demand, and not an obscure mechanism that escapes from their control (or illusion of control). Therefore, a first line of future research will be dedicated to investigating theory of argumentation for presenting the end user with explanation, perhaps with different output interaction modalities [5]. A second potential avenue will be to conduct a user study in order to determine the user preference for an animated transition for each adaptation operation and to compare these results with those resulting from the theoretical reasoning that was held based on the cognitive load.

## IV. A SECOND INSTANTIATION

In order to introduce a second instantiation of the general-purpose conceptual framework introduced in Section 2, this section introduces, defines, and explains the various steps required to establish an animated transition between UI views. Figure 8 shows intermediate steps of an animated transition between a conceptual and an external view, then between an internal view and an external view. This last transition will be now discussed in the next sub-sections, while the latter is similar in principle.

### A. *Step 1. Define the External View.*

From its definition, the external view is interpreted as the final GUI with the L&F belonging to its computing platform. Therefore, the external view will consist of any runnable GUI in any platform that could be expressed in terms of widgets.

### B. *Step 2. Define the Internal View*

From its definition, the internal view is considered as the developer's view in which the UI code or description is manipulated. Today, several UIDLs, such as XWT (http://wiki.eclipse.org/E4/XWT), XIML (www.ximl.org), or UIML (www.uiml.org), allows describing a GUI. In our case, UsiXML was selected, but other UIDLs could be used without changing the principles.

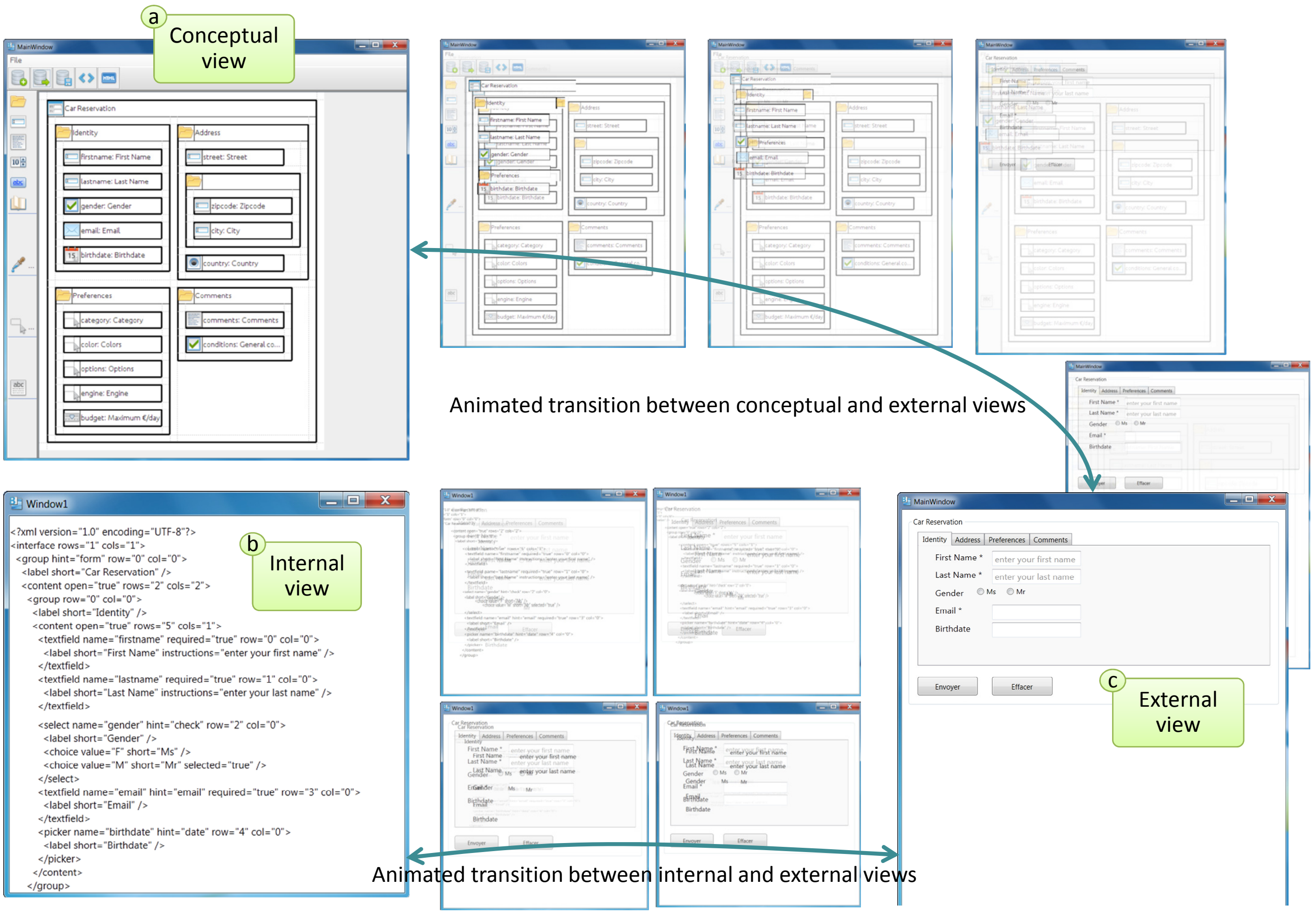
### C. *Step 3. Define the Mapping between Views*

In order to define a mapping between UI views, say for instance here from the internal view to the external view, a correspondence should be established and maintained between elements belonging to the internal view and elements belonging to the external view. This mapping is structured according to the following format: *Mapping M: series of pairs (variable name, value)*➥*set of instructions on the widgets of the external view.*

### D. *Step 4. Derive the Transition from the Mapping Definition*

A transition is hereby defined as the logical way to transform the input of a mapping into its output depending on their respective data type (e.g., text, color, shape). A transition is therefore encoded by an identifier, a name, a list of synonyms, a description, a transition type (e.g., text-to-text, text-to-color, text-to-shape), and a transition cardinality that is defined:

- *One to one*: one element belonging to the initial view (the internal view in our running example) is mapped onto one element belonging to the final view (the external view in our running example). E.g., assign a label to a widget.
- *One to many*: one element belonging to the initial view is mapped onto many elements belonging to the final view. For example, create an instance of a widget type or assign the same foreground color to a set of widget instances.
- *Many to one*: many elements belonging to the initial view are mapped onto one element belonging to the final view. For example, the foreground color of one widget in a web page is determined by considering HTML code and CSS.
- *Many to many*: many elements belonging to the initial view are mapped onto many elements belonging to the final view. For example, HTML and CSS together determine the border color of several widgets included in a container.


a
Conceptual view
Animated transition between conceptual and external views
b
Internal view
c
External view
Animated transition between internal and external views

```
<?xml version="1.0" encoding="UTF-8"?>
<interface rows="1" cols="1">
 <group hint="form" row="0" col="0">
  <label short="Car Reservation" />
  <content open="true" rows="2" cols="2">
   <group row="0" col="0">
    <label short="Identity" />
    <content open="true" rows="5" cols="1">
     <textfield name="firstname" required="true" row="0" col="0">
      <label short="First Name" instructions="enter your first name" />
     </textfield>
     <textfield name="lastname" required="true" row="1" col="0">
      <label short="Last Name" instructions="enter your last name" />
     </textfield>

     <select name="gender" hint="check" row="2" col="0">
      <label short="Gender" />
      <choice value="F" short="Ms" />
      <choice value="M" short="Mr" selected="true" />
     </select>
     <textfield name="email" hint="email" required="true" row="3" col="0">
      <label short="Email" />
     </textfield>
     <picker name="birthdate" hint="date" row="4" col="0">
      <label short="Birthdate" />
     </picker>
    </content>
   </group>
```

**Figure 8**. Starting point, intermediate steps, and ending points of animated transitions between user interface views.

## V. Conclusion and Future Work

This paper presented a general-purpose conceptual framework for implementing animated transition between various model views, which are structured into view shapes linked by view connections. Each model view is aimed at representing any model, its model elements, and model relationships that link these elements. For a single model, one or many model views could be defined with mappings between. An animated transition is then defined as an animation between the view shapes of one or many model views corresponding to this or these models. Two instantiations of this general-purpose conceptual framework are presented: one for user interface adaptation for web application and one for transition between UI views (conceptual, internal, and external) during development.

## Acknowledgments

This paper is dedicated in memoriam of Paul Vanderdonckt, retired principal controller at Société Anonyme Belge de Constructions Aéronautiques (SABCA, www.sabca.be), the father of the author, for the so numerous gifts received.

The author would like also to acknowledge the support of the ITEA2-Call3-2008026 UsiXML project supported by Région Wallonne DGO6 as well as the FP7-ICT5-258030 SERENOA project supported by the European Commission.

## References

[1] N. Aquino, J. Vanderdonckt, I. Panach, and O. Pastor, "Conceptual Modelling of Interaction," Chapter 3, in Handbook of Conceptual Modelling: Theory, Practice, and Research Challenges, D. Embley and B. Thalheim, Eds. Berlin: Springer-Verlag, 2011, pp. 335–358.

[2] R. Baecker and I. Small, "Animation at the interface," in The Art of Human-Computer Interface Design, B. Laurel, Ed., New York: Addison-Wesley, 1990.

[3] P. Baudisch, D. Tan, M. Collomb, D. Robbins, K. Hinckley, M. Agrawala, S. Zhao, and G. Ramos, "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). New York: ACM Press, 2006, pp. 169–178.

[4] H. Baumeister, A. Knapp, N. Koch, and G. Zhang, "Modelling Adaptivity with Aspects," in: Proc. of ICWE'2005. LNCS, Vol. 3579. Berlin: Springer, 2005, pp. 406-416.

[5] N.O. Bernsen, "Multimodality in Language and Speech Systems - from theory to design support Tool," in Multimodality in Language and Speech Systems, Granström, B., Ed. Dordrecht: Kluwer Academic Publishers, 2002

[6] T. Bladh, D.A. Carr, and M. Kljun, "The Effect of Animated Transitions on User Navigation in 3D Tree-Maps," in Proc. of the 9th Int. Conf. on Information Visualization InfoVis'2005 (Minneapolis, October 23-25, 2005). Washington: IEEE Computer Society, 2005, pp. 297–305.

[7] D. Browne, P. Totterdell, and M. Norman, M. (Eds.), "Adaptive User Interfaces," Computers and People Series. London: Harcourt Brace Jovanovich Publishers, 1990.

[8] P. Brusilovsky, A. Kobsa, and W. Nejdl, W. (Eds.), “The Adaptive Web, Methods and Strategies of Web Personalization,” LNCS, Vol. 4321. Berlin: Springer, 2007.

[9] S. Casteleyn, O. De Troyer, and S. Brockmans, “Design Time Support for Adaptive Behavior in Web Sites,” in Proc. of ACM Symposium on Applied Computing SAC’2003 (Melbourne, March 9-12, 2003). New York: ACM Press, 2003, pp. 1222–1228.

[10] S. Casteleyn, F. Daniel, P. Dolog, M. Matera, G.-J. Houben, and O. De Troyer, Eds., Proc. of the 2nd Int. Workshop on Adaptation and Evolution in Web Systems Engineering AEWSE’2007 (Como, July 19, 2007). CEUR Workshop Proceedings, Vol. 267, 2007.

[11] B.-W. Chang and D. Ungar, “Animation: From Cartoon to User Interface,” in Proc. of ACM Symposium on User Interface Software Technology UIST’93 (Atlanta, November 3-5, 1993). New York: ACM Press, 1993, pp. 45–55.

[12] B.-W. Chang, J.D. Mackinlay, P.T. Zellweger, and T. Igarashi, “A negotiation architecture for fluid documents,” in Proc. of the 11th Annual ACM Symposium on User Interface Software and Technology UIST’98 (San Francisco, November 1-4, 1998). New York: ACM Press, 1998, pp. 123–132.

[13] H. Dieterich, U. Malinowski, T. Kuhme, and M. Schneider-Hufschmidt, “State of the art in adaptive user interfaces,” in Adaptive User Interfaces Principles and Practice, Schneider-Hufschmidt, M., Kuhme, T., Malinowski, U., Eds. Amsterdam: Elsevier Science Publishers B.V., 1993, pp. 13–48.

[14] F. Chevalier, P. Dragicevic, A. Bezerianos, and J.-D. Fekete, “Using Text Animated Transitions to Support Navigation in Document Histories,” in Proc. of ACM Conf. on Human Aspects in Computing Systems CHI’2010 (Atlanta, April 10-15, 2010). New York: ACM Press, 2010, pp. 683–692.

[15] Ch.-E. Dessart, V. Motti, and J. Vanderdonckt, “Showing User Interface Adaptivity by Animated Transitions,” in Proc. of 3rd ACM Symposium on Engineering Interactive Computing Systems EICS’2011 (Pisa, 13-16 June 2011). New York: ACM Press, 2011, pp. 95–104.

[16] Ch.-E. Dessart, V. Motti, and J. Vanderdonckt, “Animated Transitions between User Interface Views”, in Proc. of ACM Int. Working Conf. on Visual User Interfaces AVI’2012 (Capri, May 21-25, 2012). New York: ACM Press, 2012.

[17] P. Dragicevic, A. Bezerianos, W. Javed, N. Elmqvist, and J.-D. Fekete, “Temporal Distortion for Animated Transitions,” in Proc. of ACM Conf. on Human Aspects in Computing Systems CHI’2011 (Vancouver, May 7-12, 2011). New York: ACM Press, 2011, pp. 2009–2018.

[18] P. Dragicevic, S. Huot, and F. Chevalier, “Gliimpse: Animating from Markup Code to Rendered Documents and Vice Versa,” in Proc. of 24th ACM Symposium on User Interface Software and Technology UIST’2011 (Santa Barbara, October 16-19, 2011). New York: ACM Press, 2011, pp. 257-262.

[19] C. Dunn, “The Use of Real-Time Simulation by Means of Animation Film as an Analytical Design Tool in Certain Spatio-Temporal Situations,” Ergonomics, vol. 16, 1973, pp. 515–519.

[20] S. Eick, J. Steffen, and E.S. Jr. Seesoft, “A tool for visualizing line oriented software statistics,” IEEE Trans. on Software Engineering, vol. 18, no. 11, 1992, pp. 957–968.

[21] A.T.S. Fialho and D. Schwabe, “Enriching Hypermedia Application Interfaces,” in Proc. of 7th Int. Conf. on Web Engineering ICWE’2007 (Como, July 16-20, 2007). Lecture Notes in Computer Science, vol. 4607. Berlin: Springer-Verlag, 2007, pp. 188–193.

[22] I. Garrigós, S., Meliá, and S. Casteleyn, “Adapting the Presentation Layer in Rich Internet Applications,” in Proc. of ICWE’2009. Berlin: Springer, 2009, pp. 292–299.

[23] J.M. Gómez and T. Tran, “A Survey on Approaches to Adaptation on the Web,” in Emerging Topics and Technologies in Information Systems, M.D. Lytras, Ordóñez de Pablos, P., Eds. Hershey: IGI Global, Hershey, 2009, pp. 136–152.

[24] C. Gonzalez, “Does animation in user interfaces improve decision making?,” in Proc. of ACM Conf. on Human Aspects in Computing Systems CHI’1996 (Vancouver, April 13-18, 1996). New York: ACM Press, 1996, pp. 27–34.

[25] J. Heer and G. Robertson, “Animated Transitions in Statistical Data Graphics,” IEEE Trans. on Visualization and Computer Graphics, vol. no. 6, Nov. 2007, pp.1240–1247.

[26] W. Hong, J.Y.L. Thong, and K.-Y. Tam, “Does Animation Attract Online Users’ Attention? The Effects of Flash on Information Search Performance and Perceptions,” Information Systems Research, vol. 15, no. 1, 2004, pp. 60–86.

[27] J. Huhtala, J. Mäntyjärvi, A. Ahtinen, L. Ventä, and M. Isomursu, “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. Heidelberg: Springer, 2009, pp. 772–781.

[28] J. Huhtala, A.-H. Sarjanoja, J. Mäntyjärvi, M. Isomursu, and J. Häkkilä, “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 (Atlanta, April 10-15, 2010). New York: ACM Press, 2010, pp. 1339–1342.

[29] T. Lavie and J. Meyer, “Benefits and costs of adaptive user interfaces,” International Journal of Human-Computer Studies, 68 (2010), pp. 508–524.

[30] J. Mackinlay, “Automating the Design of Graphical Presentations of Relational Information,” ACM Trans. on Graphics, vol. 5, no. 2, April 1986, pp. 110–141.

[31] Ch. Mertz, S. Chatty, and J.-L. Vinot, “The influence of design techniques on user interfaces: the DigiStrips experiment for air traffic control,” in Proc. of HCI-Aero’2000 (Toulouse, September 2000).

[32] B. Michotte, J. Vanderdonckt, “GrafiXML, A Multi-Target User Interface Builder based on UsiXML,” in Proc. of 4th Int. Conf. on Autonomic and Autonomous Systems ICAS’2008 (Gosier, 16-21 March 2008). Los Alamitos: IEEE Computer Society Press, 2008, pp. 15–22.

[33] T. Mirlacher, Modeling Animations for Dependable Interactive Applications, Proc. of 3rd ACM Symposium on Engineering Interactive Computing Systems EICS’2011 (Pisa, 13-16 June 2011). New York: ACM Press, 2011, pp. 319–322.

[34] O. Pastor, “Generating User Interfaces From Conceptual Models: A Model-Transformation Based Approach,” in Proc. of 4th Int. Conf. of Computer-Aided Design of User Interfaces CADUI’2002 (Valenciennes, 15-17 May 2002), Ch. Kolski, J. Vanderdonckt, Eds. Dordrecht: Kluwer Academics, 2006, pp. 1–14.

[35] O. Pastor and J.C. Molina, “MDA in Practice: a Software Production Environment Based on Conceptual Modelling”, Berlin: Springer, 2008.

[36] I. Pederiva, J. Vanderdonckt, S. España, I. Panach, and O. Pastor, “The Beautification Process in Model-Driven Engineering of User Interfaces,” in Proc. of 11th IFIP TC 13 Int. Conf. on Human-Computer Interaction INTERACT’2007 (Rio de Janeiro, September 10-14, 2007). Lecture Notes in Computer Science, vol. 4662. Berlin: Springer, 2007, pp. 409–422.

[37] G. Rossi, O. Pastor, D. Schwabe, and L. Olsina, Eds., “Web Engineering: Modelling and Implementing Web Applications,” Human-Computer Interaction Series. Berlin: Springer, 2008.

[38] B. Sarkar, “LWUIT 1.1 for Java ME Developers,” Packt Publishing, 2009.

[39] C. Schlienger and M. Anquetil, “A formal Proof of Animation and Sound Benefit in ATC/ATM User Interfaces: Improving Controller's Situation Awareness,” in Proc. of INO’2006 Workshop (Bretigny, December 5–7, 2006)

[40] C. Schlienger, P. Dragicevic, C. Ollagnon, and S. Chatty, “Les transitions visuelles différenciées : principes et applications,” in Proc. of IHM’2006 (Montréal, April 18-21, 2006). ACM Int. Series, vol. 133. New York: ACM Press, 2006, pp. 59–66.

[41] C. Schlienger, S. Conversy, S. Chatty, M. Anquetil, and Ch. Mertz, “Improving Users’ Comprehension of Changes with Animation and Sound: An Empirical Assessment,” in Proc. of 11th IFIP TC 13 Int. Conf. on Human-Computer Interaction INTERACT’2007 (Rio de Janeiro, September 10-14, 2007). Lecture Notes in Computer Science, vol. 4662. Berlin: Springer-Verlag, 2007, pp. 207–220.

[42] M. Shanmugasundaram and P. Irani, “The effect of animated transitions in zooming interfaces”. in Proc. of ACM Conf. on Advanced Visual Interfaces AVI’2008. New York: ACM Press, 2008, pp. 396–399.

[43] W. Schwinger, et al., “A survey on web modeling approaches for ubiquitous web applications,” Int. Journal of Web Information System 4, 3 (2008), pp. 234–305.

[44] J. Stasko, “Animation in User Interfaces: Principles and Techniques,” in Proc. of ACM Symposium on User Interface Software Technology UIST‘1993. New York: ACM Press, 1993, pp. 81–101.

[45] J. Teevan, S.T. Dumais, D.J. Liebling, and R. Hughes, “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 (Atlanta, April 10-15, 2010). New York: ACM Press, 2010, pp. 1353–1356.

[46] B.H. Thomas and P. Calder, “Applying Cartoon Animation Techniques to Graphical User Interfaces,” ACM Trans. on Computer-Human Interaction, vol. 8, no. 3, September 2001, pp. 198–222.

[47] J.B. Tucker, “Computer Graphics Achieves New Realism,” High Technology, June 1984, pp. 40–53.

[48] J. Vanderdonckt and X. Gillo, “Visual Techniques for Traditional and Multimedia Layouts,” in Proc. of 2nd ACM Workshop on Advanced Visual Interfaces AVI’1994 (Bari, June 1-4, 1994). New York: ACM Press, 1994, pp. 95–104.

[49] J. Vanderdonckt, Model-Driven Engineering of User Interfaces: Promises, Successes, and Failures,” in Proc. of 5th Annual Romanian Conf. on Human-Computer Interaction ROCHI’2008 (Iasi, September 18-19, 2008), S. Buraga, I. Juvina, Eds., Bucharest: Matrix ROM, 2008, pp. 1–10.

[50] D. Vodislav, “A visual programming model for user interface animation,” in Proc. of IEEE Symposium on Visual Languages InfoVis’97 1997, pp. 344–351.

[51] R. Williams, “Techniques d’animation pour le dessin animé, l’animation 3D et le jeu video”. Eyrolles, Paris, 2009. Translated from R. Williams, “The Animator’s Survival Kit,” Faber and Faber ltd, 2009.

[52] R.C. Zeleznik, D. Brookshire Conner, M.M. Wloka, D.G. Aliaga, N.T. Huang, P.M. Hubbard, B. Knep, H. Kaufman, J.F. Hughes, and A. van Dam, “An Object-Oriented Framework for Integration of Interactive Animated Techniques,” in Proc. of SIGGRAPH’91. Computer Graphics, vol. 25, no. 4, July 1991, pp. 105-112.