Journal of Web Engineering, Vol. 16, No.7&8 (2017) 371-394

AN SMIL-TIMESHEETS BASED TEMPORAL BEHAVIOR MODEL FOR THE VISUAL DEVELOPMENT OF WEB USER INTERFACES

MARINO LINAJE, JUAN CARLOS PRECIADO, ROBERTO RODRIGUEZ-ECHEVERRIA, JOSÉ MARÍA CONEJERO AND FERNANDO SÁNCHEZ-FIGUEROA

Quercus Software Engineering Group. School of Technology. University of Extremadura, Spain
{mlinaje,jcpreciado,rre,chemacm,fernando}@unex.es

Received August 18, 2016
Revised March 30, 2017

Temporal behaviors are being incorporated into the user interfaces of Web applications making them look more and more like multimedia applications, the so-called Rich Internet Application (RIA) user interfaces. Due to RIA complexity, some research communities have proposed models to ease its development. However, there is a gap to cover between formal temporal relationships and the current state of the art in the RIA model-driven development techniques. The purpose of this paper is to specify a temporal behavioral model for data-intensive RIA user interfaces with three main objectives. The first one is that the model must be usable by non-experts in engineering specifications (e.g., Web designers). The second one is that the model must be suitable to be implemented in a CASE tool integrating temporal behaviors in the RIA model driven development workflow. The third one is that the temporal behaviors specified must run in current Web browsers. The approach here presented is based on SMIL Timesheets, a standard that can be used as a foundation to extend RIA user interface model driven proposals.

Key words: Design tools and techniques, Web Engineering, Rich Internet Applications, Multimedia temporal relationships, SMIL Timesheets
Communicated by: B. White & G. Rossi

1 Introduction

Currently, we can appreciate how temporal behaviors have been incorporated into the User Interfaces (UIs) of Web applications making them look more and more like multimedia applications. These multimedia features, together with others such as asynchronous communications or client processing capacities, have led to which is known as Rich Internet Applications (RIAs) [52]. According to Pleub et al. [41], their key difference is the increased dynamics of the UI, where elements can be modified according to the current context, and the tighter integration of these elements with the application logic. Thus, RIAs can be considered as multimedia highly interactive Web applications.

The complexity of RIAs development forwarded the necessity of models and methodologies to cope with them [42]. Many of the current systematic approaches use model-driven development techniques, allowing designers to focus on abstractions more than on implementation details [52]. The Web Engineering field provides Web models (e.g. WebML, UWE, OO-H, OO-HDM [47]) that are able to support the systematic development of data-intensive Web applications. The HCI field provides

UI description languages and models (e.g., UIML or UsiXML [15]) to develop multi-device and multi-target UIs, being most of the current proposals based on the Cameleon framework [9]. The Multimedia field also provides models (e.g., MML [39]) for the specification of multimedia applications. However, all these models are far from the RIA designer's expectations since they do not cover most of the new UI requirements appearing in RIAs detailed in [42].

In order to lay a bridge between the Web, HCI and multimedia fields, RUX-Method [25] (Rich User eXperience Method, now on RUX) was specified. Some pieces of RUX-Method such as its description language have been presented to the HCI community (e.g., [24]) and other pieces, such as the connection with Web models, have been presented to the Web Engineering community (e.g., [23]). However, the RUX temporal behavioral model (graphical and textual) has never been detailed. In this paper, we focus on the UI temporal behaviors coming from the Multimedia field that are highlighting the limits of RIAs Web Engineering and HCI methodologies. Since the main focus of Web Models are data-intensive RIAs [52], this kind of applications will be the target of our research. Being data-intensive RIAs (e.g., Gmail, YouTube, etc.) not characterized by complex animations, it is not required to incorporate the whole set of possibilities that are available in complex multimedia applications [12].

The temporal behavioral model proposed has three main objectives. These objectives arise from the practical experience of Homeria Open solutions[a], a University of Extremadura spin-off company, dealing with academic and non-academic customers during the last 8 years. The first objective is that these models must be usable by non-experts in engineering specifications e.g., web designers. The second objective is that these models can be specified by means of an authoring model driven development tool. The third objective is that the applications specified using the temporal behavior model must run in current Web browsers.

Firstly, to be usable by non-experts, we start from the assumption that SMIL documents are difficult to read and not free from inconsistencies [14]. As in the case of end-user design for authoring tools [38], the proposal will try to simplify temporal behaviors management through graphical visualization.

Secondly, to maximize compatibility when implementing the temporal behavior model in an authoring tool, our research is based on SMIL Timesheets [54], a declarative standard timing language. SMIL timesheets are based on a limited subset of SMIL 3.0, so some multimedia features are missed [19]. However, its expressiveness is enough for our target applications. SMIL Timesheets was selected due to its simplicity and connectivity facilities even when some adaptations were required. For practical issues, we will use RUX-Method [26] to detail the adaptations over a real model driven development method. Hopefully, some of the adaptations proposed here for SMIL Timesheets will help to improve its interoperability with other languages and specifications in future versions.

For the third objective, we will use a feature of model driven methods called transformations. This transformation will allow us to generate the final code for the temporal behaviors specified and run them without web browsers plug-ins or custom extensions. While browsers render and run RIAs, not all RIA renderers support SMIL [10]. Furthermore, for performance issues temporal behaviors are being coded by professionals at Homeria using CSS or JavaScript code.

[a] http://www.homeria.com

This paper is organized as follows: in Section 2 RUX-Method is briefly introduced. Section 3 is focused on the Temporal Presentation model. Section 4 is devoted to the related work. In Section 5 the implementation details are shown and in Section 6 users' evaluations are detailed. Finally, conclusions and future work are presented in Section 7.

2 RUX-Method brief introduction

RUX is a model driven method which supports the design of multimedia, multimodal and multidevice interactive Web 2.0 UIs (i.e., RIAs).

RUX is organized into four design levels (see Figure 1): Concepts and Tasks, Abstract UI, Concrete UI and Final UI. It takes Concepts and Tasks (i.e., data and business logic) from the underlying Web model while the rest of the levels are mainly based on RUX Components. RUX UI Component specification is stored in the Component Library, which also stores how the transformations among components of different levels will be performed. Each RUX Component is hierarchically defined, so all of them share properties, events, and methods with their common ancestors. The properties take their values from expressions, being: a constant, a value of a property from the same or other Component (except the same Component and the same property), a device capability, a value coming from the underlying business logic or an OCL operation that involves any of the previous ones. Components may have a visual UI representation (e.g., a button) or not (e.g., background music player).

In RUX, the Abstract UI provides a conceptual representation of the UI with all the features that are common to all the RIA devices and development platforms, without any kind of spatial, look&feel or behavior dependency (Figure 2). This RUX UI level works with the following conceptual components: views (for any different types of containers like alternative, hierarchical, etc.), media (divided into continuous and discrete for different type of media like text, video, etc.) and connectors (to connect the User Interface with the business logic of the underlying Web model to e.g., filter or personalize the multimedia content to be delivered [46]). Those media attached to a connector are data-driven while those which are not attached are typically used just for decoration.

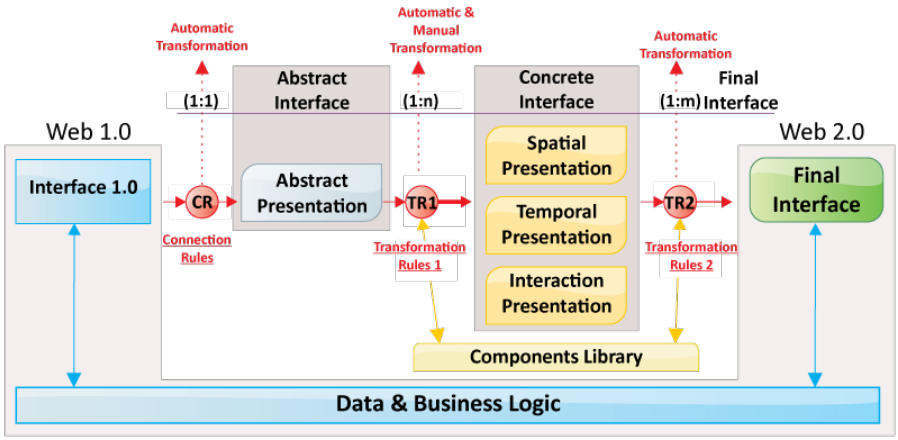

Figure 1 RUX-Method architecture overview.

The Concrete UI specializes the UI model for a specific device or set of devices. The Concrete UI is divided into three Presentation levels in order to provide a deeper separation of concerns: Spatial, Temporal, and Interaction Presentation. Spatial Presentation allows the spatial arrangement of the UI and the look&feel to be specified. Temporal Presentation allows the specification of those behaviors which require temporal synchronization(s). Interaction Presentation allows modeling the behaviors that the user produces through events over the UI. Both, Temporal and Interaction Presentations are able to produce events. These events can trigger handlers, which are the way to specify actions in RUX (e.g., trigger a business operation at server-side, change a property of a UI component, etc.) under conditions (e.g. only when the button visibility property is set to true). Temporal behaviors are related to a UI element (e.g., a window that performs an animation when maximized or minimized). So, temporal behaviors may be related to interaction behaviors such as "click" (e.g., maximize button). But temporal behaviors could also be related to other behaviors, e.g. when the application starts and finishes according to a duration, or when certain value is reached, or when the application is closed.

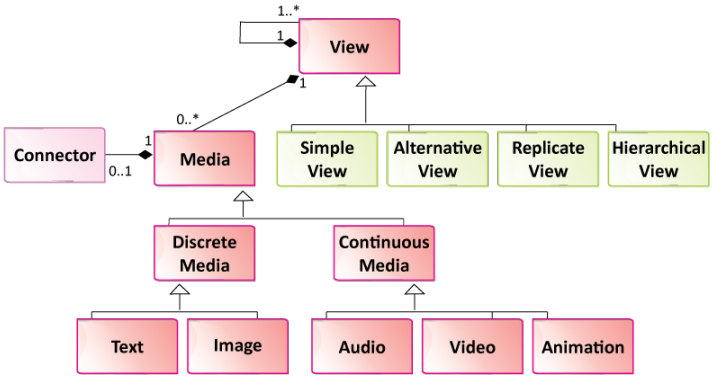

Figure 2 RUX-Method Abstract User Interface model hierarchy.

There are two adaptation phases in RUX according to the UI levels defined above: one to adapt Web contents and business logic from the underlying Web model to RUX Abstract UI (marked CR in Figure 1) and the one that fits this Abstract UI to one or more particular devices and grants a right access to the business logic (marked TR1 in Figure 1). The RUX process ends with the Final UI specification which corresponds to the model-to-text transformation (TR2 in Figure 1) i.e., the code generation phase. RUX-Tool [23] is the RUX-Method CASE authoring tool.

3 RUX-Method Temporal Presentation

RUX incorporates the Temporal Presentation to establish temporal relationships between one or more Concrete UI components as specified by [22]. This fact allows the specification of temporal relations such as "*c* begins before *d* but after *a* and *b*". In RUX the Temporal Presentation can affect whatever UI element since the whole UI is based on components.

The Temporal Presentation allows altering three kinds of properties: spatial, style and internal (self-component) behavior. When the Temporal Presentation alters spatial properties (fixed by the Spatial Presentation), they are spatial animations. In RUX there are two types of them: movement (when the animation affects to position properties e.g., left) and transformation (when the animation affects size properties such as height). The data-type of a property in a component specification defines the valid values and, for the RUX Temporal Presentation, it affects the possibility to interpolate values in animations. E.g., "visible" property may be valuated "true" or "false" so continuous animation will not be possible, but "opacity" may be valuated from 0 to 100 allowing a continuous animation.

The Temporal Presentation also allows calling to the underlying business logic by scheduling the triggering of operations using temporal events (e.g., performing a query to the business logic every ten seconds).

The properties and the declarative specification of temporal relations logic presented here is based on the SMIL Timesheets 1.0 specification after some adaptations in order to cope with the model-driven development of RIAs followed by RUX. SMIL Timesheets declarative XML-based nature should facilitate the connection with some model-based approaches, being many of them already represented in XML [48], as in the case for RUX.

Here, we detail which pieces of SMIL Timesheets are not required, those ones adopted as they are, those ones adapted and also some additions in order to simplify the development of RIA UIs. Even when SMIL Timesheets is incorporated here to RUX for practical issues, many of the adaptations should be necessary for other UI model-driven approaches and some advices are provided in this sense for other researchers.

The abstract syntax of the temporal presentation language is defined as an Ecore metamodel which is partially shown in Fig.3. This metamodel resembles the definition of Triads and Quartets presented below. Its primary element is TemporalBehaviour which encapsulates all the commonalities shared between Triads and Quartets.

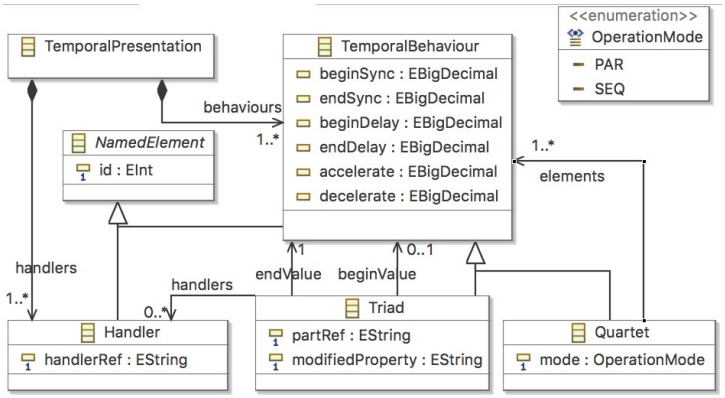

Figure 3 RUX-Method Abstract User Interface model hierarchy.

3.1. Temporal Presentation definitions

Some concepts must be introduced before going into details in the following sections.

Definition 1. **Temporal Presentation elements.** All those items that compose the Temporal Presentation. They can be Concrete Interface Components or groups of them, which are instantiated in the Concrete UI description.

Definition 2. **Real using time.** Linearly elapsed time from the application start until the user closes it. It includes all the pauses, repetitions or whatever possible alteration in the execution of the temporal relationships logic (originated by the user interaction or by the temporal behavior).

Definition 3. **Predefined using time.** Duration of a temporal behavior established on one or more temporal presentation elements.

Based on the last two definitions, the real using time will always be equal or higher than the predefined using time. In this document, the millisecond will be used as time unit. However, this fact does not limit the expressive power of RUX. E.g., for those RIA platforms based on frames, it is possible to perform conversions between units (that, for instance, an authoring tool could provide). This relationship is important when Transformation Rules 2 (TR2) are triggered to generate the code for the final UI (e.g., Adobe Flash).

Definition 4. **Moment.** It defines the state at a given instant for all the UI elements.

Definition 5. **Period.** Time elapsed between two moments. It is useful to indicate the temporal behavior between two different moments.

3.2. Triads and quartets

The temporal relationships are specified by means of triads and quartets to simplify the number of elements originally available in SMIL Timesheets. Triads are used to define the temporal relationships that affect a single element (E). Quartets are used to specify complex relationships that affect a group of elements (G) that must be synchronized.

It is possible to attach triads and quartets to any moment during the real using time, but it is also possible to define them without this attachment and use them when required (e.g., being triggered by and active or passive behavior).

Triads and quartets presented in Sections 3.2.1 and 3.2.2, respectively, are abstractions to represent the most basic elements mainly for timing and synchronization. In order to provide e.g., animations, a more detailed specification is required (e.g., which property of the component changes). Being RUX a component based approach; these details are provided as properties (Section 3.4).

Even when it is not explained here due to its complexity and not being the objective of this paper, triads and quartets provide the basis for the formal specification of temporal relationships and have been used to check the coverage of Allen et al. temporal relationships [3]. In the future, they will be used to check spatial/temporal constraints [28].

3.2.1 Triads

They have the following structure:

$$E[E^0 : V_O, E^{00} : V_F, \{handler : V_E\}] \quad (1)$$

where:

***Definition 6*. E.** Temporal Presentation element target of the temporal logic indicated.

***Definition 7*. E^0.** Temporal presentation element related to *E* to start its temporal behavior in a synchronized way.

***Definition 8*. V_O.** Numeric value representing the delay in the start of element *E* regarding the start of element *E'*.

***Definition 9*. E^{00}.** Temporal Presentation element with which *E* is related to, in order to finish its temporal behavior in a synchronized way.

***Definition 10*. V_F.** Numeric value representing the delay in the end of temporal element *E* regarding the end of element *E'*. It can contain also the value END allowing temporal behaviors until the end of the real using time.

***Definition 11*. handler.** Reference to a handler in a set of defined handlers in RUX-Method.

***Definition 12*. V_E.** Delay when launching the handler from the beginning of element E.

E, *E'* and *E"* may refer or not to the same element. The couple *handler* : V_E appears between brackets indicating that it is a set, which can be empty (i.e., no handler to be triggered) or not. It can contain different couples separated by commas to indicate the triggering of different handlers in different moments during the period in which the temporal behavior of *E* is specified. Taking into account that the couple is compulsorily tied to the predefined temporal event *E* being affected by its temporal definition, the launching of the handler is also affected by such situation in a way that, if *E* is repeated in time, the launching of the handler will be too.

3.2.2 Quartets

A quartet has the next structure:

$$G[E^0 : V_O, M, E^{00} : V_F, \{element\}] \quad (2)$$

where:

Definition 13. **G.** Group of temporal elements, the target of the indicated temporal logic.

Definition 14. **E^0.** Temporal presentation element related to *G* to start its temporal behavior in a synchronized way.

Definition 15. **V_O.** Numeric value representing the initial delay of element *G* regarding the start of element *E'*.

Definition 16. **M.** Execution mode, which can be PAR where the children elements of *G* behaves in parallel (default value), or SEQ, where the children elements of *G* behaves sequentially.

Definition 17. **E^{00}.** Element related to *G* that finishes its temporal behavior in a synchronized way with *G*.

Definition 18. **V_F.** Delay in the end of *G* regarding the end of element *E"*. It can contain the value LOOP(n) to indicate that temporal behavior of *G* will be repeated *n* times. The valid units are: 1) natural numbers to represent a certain number of repetitions; 2) an absolute value indicating the duration of the repetitions e.g. milliseconds; 3) *END*, indicating that it will be repeated until the end of the real using time (or the end of the predefined using time when the quartet is included in another quartet). This allows establishing a group of animations e.g., until the application finishes.

Definition 19. **element.** It appears between brackets, indicating that it is a set of elements. This set cannot be empty. It will contain more than one element to indicate the temporal and synchronization relationships between them (triads and quartets).

3.2.3 Additional concepts

Due to the implicit values available within the temporal behavior, it is possible to simplify the temporal notation in some cases as detailed as follows. For the specification of triads and quartets grouped into quartets and when the designer does not wish to specify a delay in the starting and/or ending of the temporal behavior, it is not mandatory to indicate the starting values for their temporal behavior (i.e., $E^0 : V_O$) and/or their finalization values (i.e., $E^{00} : V_F$). In these cases, start and/or end values are represented by two hyphens (i.e., –).

3.3. Methods and events

Temporal relationships (i.e., triads and quartets) are considered in RUX as a kind of continuous media Component (Figure 2), thus containing properties, methods and events in the same way that other components defined in the Component Library.

The methods and events provided by RUX are inspired by those described in [3, 18], but avoiding the information already provided in other UI levels of RUX. Notwithstanding, and due to the definition of components in RUX which make them extensible, it is possible to include new methods, events, and properties to specify temporal conditions that are not initially considered at any time to the Component Library.

Topic	SMIL Timesheets 1.0	RUX-Method
	Design-time rationale	
Structure	*(X)HTML*	*Abstract UI model (Views and Media)*
Layout and Styling	*CSS*	*Spatial Presentation model*
User interaction and data model	*XForms*	*User Interaction: Interaction Presentation* *Data model: (connectors: Abstract UI)*
Temporal behaviors	*SMIL Timesheets*	*Temporal Presentation*
	Run-time rationale	
Structure	*(X)HTML*	*Multi-platform (e.g., XHTML, XAML)*
Layout and Styling	*CSS*	*Multi-platform (e.g., CSS, skins)*
User interaction and data model	*XHTML forms + JavaScript*	*Multi-platform (e.g., XHTML+JavaScript, Flash)*
Temporal behaviors	*SMIL Timesheets + JavaScript*	*Multi-platform (e.g., JavaScript, ActionScript)*

Table 1 SMIL Timesheets and RUX-Method rationale

The set of methods considered for temporal relationships logic is:

- play: makes the temporal behavior to start running from the initial moment indicated on its predefined using time
- pause: stops the execution of temporal behavior in the current moment
- resume: the execution of the temporal behavior continues from the current moment
- stop: the execution of the temporal behavior stops, going back to the initial moment indicated in its predefined using time
- In addition, the temporal relationships logic has the following events:
- onPlay: activated when a temporal behavior begins to execute
- onPause: activated when a temporal behavior stops at a given moment before its predefined final
- onResume: activated when a temporal behavior resumes its execution after being paused
- onStop: activated when a temporal behavior stops its execution

With these methods and events, it is possible to create the definition of behaviors such as ”when the temporal behavior T1 is resumed, turn the audio volume down progressively (starting temporal behavior T2)”.

3.4. Methods and events

This section outlines the adoption and adaptation of SMIL Timesheets in RUX.

3.4.1 Rationale

First of all, the rationale of SMIL Timesheets and RUX is different. Table 1 summarizes both approaches at design- and run-time, using the original table presented in the SMIL Timesheets specification [55] but removing the "Vector Graphics" row (for SVG), because it makes no sense here specifying media support (e.g., the type of images, audio or video formats).

Differences at design-time between SMIL Timesheets and RUX can be summarized as follows: while SMIL Timesheets is focused on the combination with other W3C standard languages (i.e., XHTML for the structure, CSS for layout and styling, etc.,), RUX provides models for each topic that are not technology dependent and also a way to interconnect all these models and transform them (i.e., a method).

At run-time, SMIL Timesheets approach typically requires an XForms application server and an SMIL Timesheets JavaScript engine because XForms and SMIL are not widely supported by current browsers. On the contrary, RUX is able to provide at run-time multiple native platforms without emulation or interpretation, using code generation techniques targeting AJAX or Flash RIA platforms among others. Obviously, the former exposes a slower UI responsiveness that makes it not optimal in some scenarios.

RUX Temporal Presentation does not include all SMIL Timesheets elements. SMIL Timesheets incorporates a prefetching mechanism for media that is not used by RUX. The reason is that, as stated in Section 1, RIAs are able to load any piece of the UI, not only media but also business logic and data at any moment. In RUX these pieces are loaded by default on user's demand, but prefetching can also be specified to be according to e.g., the typical navigation path to achieve a better user experience. Prefetching availability depends on the type of the Concrete UI Component because it is specified as a property. SMIL Timesheets (like SMIL) suppose visibility conditions for temporal behaviors. As an example, readers could take a look at the first code example at [54]. This is quite ambiguous from the RUX point of view, where displayable and non-displayable components (those elements without spatial and look-and-feel properties) can be combined to create a UI. In RUX non-displayable components could be used to e.g., play music in the background using an audio component without controls. Non-displayable components with temporal behavior could be used to establish connections between the UI and the business logic e.g., to delay n seconds a business logic call or to check new data at server side every five seconds. Due to the model-to-code transformation that RUX performs, ambiguity is an undesired feature when full automatic code generation is a goal. SMIL also includes some shorthands (e.g., set, AnimationMotion or AnimationColor) to specify temporal behaviors with less code in the notation, but these shorthands are not required by RUX, indeed they could lead to increase the complexity of the code generator(s).

Some SMIL Timesheets elements and attributes are adapted to fit with the RUX Temporal Presentation. SMIL Timesheets "src" and "media" attributes available in the timesheet node are not necessary for RUX. The former because all the models related to the RUX Concrete UI are specified in the same file in order to maintain inter-model constraints among them, using key IDs. The latter because RUX specifies a UI for a single device or a set of devices with common features (e.g., display size, media capabilities) using a different approach. For further information, the reader may be interested in [27]. Regarding the selection mechanism introduced by SMIL Timesheets, "select"

attribute value can be multi-valuated using CSS2 selectors. In RUX it can only take one value that must be the id of a Concrete UI Component. For the same reason, some attributes e.g., "beginInc", "index" and "indexStart" are meaningless in RUX. Indeed, both, SMIL Timesheets and RUX provide other mechanisms to deal with temporal behaviors that affect more than one UI element (e.g., par, seq in SMIL Timesheets, quartets in RUX).

RUX Temporal Presentation includes the acceleration/deceleration system for animations provided by the temporal manipulation module in SMIL [59] that is not available in SMIL Timesheets. This allows creating smooth animations that are not constant in time. For this purpose, it is defined for each triad the attributes accelerate and decelerate that must contain real values between 0 and 1 (the default value is 0 for both). These attributes arithmetically compute the acceleration/deceleration ratio of the temporal element and the sum of both values must be less or equal than one to avoid inconsistencies. RUX does not adopt the rest of temporal manipulation properties defined in SMIL (e.g., speed and autoReverse) because they can be specified with other elements of RUX or they can lead to inconsistencies with the temporal behavior and the simplified graphic representation adopted by RUX. Acceleration/deceleration temporal behavior is commonly exposed by many RIAs. Indeed, many JavaScript RIA animation frameworks include easing functions to make animations look more natural. RUX also includes a flexible set of easing shorthands such as expIn, elasticOut or quadInOut among others (being e.g., the latter a quadratic function, accelerating until halfway and then decelerating the component until the end). The full set of easing functions is not especially relevant, being only an initial set of shorthands (provided by RUX-Tool according to RUX-Method acceleration/deceleration system). While it is easier to use these shorthands e.g., for novel users, expert designers may customize this set for their UI projects.

3.5 Timeline-based representation and XML-based specification

According to the experience with real users, Deltour et al. [12] confirm that direct manipulation of SMIL is too complex for most users because it requires a deep knowledge of the semantic of the language and its constraints among others. Thus, the authors proposed using visual manipulation as an alternative.

RUX provides a basic timeline representation with different elements and restrictions in order to allow Web designers without engineering background to specify consistent temporal relationships. The different elements of the Temporal Presentation graphical representation are as follows.

Definition 20. **Temporal axes.** Two axes graphic that represents temporal relationships between elements. The horizontal axis indicates the predefined using time and time units. The vertical axis indicates the elements implied in the temporal relationship.

Definition 21. **Temporal Line.** Temporal line of a Temporal Presentation element.

Definition 22. **Black Dot.** Indicates that the temporal behavior starts or finishes (depending on its location at the beginning or the end of the temporal line) inside the visualized period of the temporal graphic representation.

Definition 23. **White Dot.** Indicates that the temporal behavior starts or finishes (depending on its location at the beginning or the end of the temporal line) outside the visualized period of the temporal graphic representation.

Definition 24. Triangle. Indicates the moment in which a handler will be triggered.

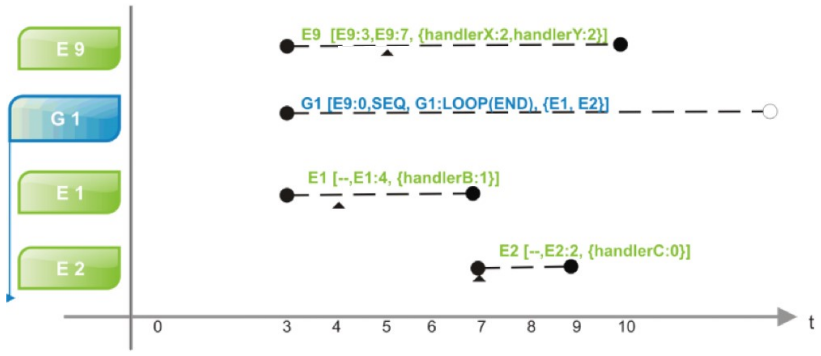

Figure 4 Example of the Temporal graphical representation.

As an example, in Figure 4 a graphical representation of temporal relationships logic is shown. It includes a grouping (i.e., G1). In this example, E9 begins running at the moment three of the predefined using time and finishes seven time units later. E9 also triggers two handlers in the instant two. This temporal diagram also specifies that G1 begins running when E9 begins and defines a temporal sequence relation between E1 and E2, looping until the end of the real using time. E1 and E2 triads have as starting synchronization value the symbol '–' due to them not having to specify their starting point of synchronization because it is defined by the grouping quartet G1.

The following code specifies in a textual way the temporal behavior of the example in Figure 4.

```
<temporalPresentation>
 <temporalbehaviour>
   <triad id="E9" beginSync="E9" beginDelay="3" endSync="E9" endDelay="7" partRef="CTMI1"
modifiedProperty="font-size" endValue="20">
     <handlerTemporal id="ht1" handlerRef="handler1" beginDelay="2"></handlerTemporal>
     <handlerTemporal id="ht2" handlerRef="handler2" beginDelay="2"></handlerTemporal>
   </triad>
   <quartet id="G1" beginSync="E9" beginDelay="0" endSync="G1" endDelay="LOOP(END)">
     <seq id="G1mode">
       <triad id="E1" endSync="E1" endDelay="4" partRef="CSV1" modifiedProperty="height"
endValue="80%" accelerate="0.2"
decelerate="0.8"></ triad>
       <triad id="E2" endSync="E2" endDelay="2" partRef="CSV1" modifiedProperty="width"
endValue="80%" accelerate="0.8" decelerate="0.2">
         <handlerTemporal id="ht2" handlerRef="handler3" beginDelay="0">
         </handlerTemporal>
       </triad>
     </seq>
   </quartet>
  </temporalbehaviour>
</temporalPresentation>
```

The previous specification example defines a temporal behavior trying to cope with different situations. It can be observed the reference to the concrete component, to the property that is wanted to be changed, to the limit values for the property and the acceleration and deceleration values.

The example code defines a temporal behavior E9 that affects the component identified as CTMI1. In this case, the size of CTMI1 text is changed (i.e., font-size property) modifying its current value by 20. The G1 quartet that specifies a sequential relationship of synchronization between temporal

behavior E1 and E2, affects the visual representation of the CSV1 component. The height property will change until obtaining a value of the 80% with an acceleration of 0,2 and a deceleration of 0,8. G1 also establishes that CSV1 width will change until getting a value of the 60%.

While inspired by SMIL, temporal XML specification was not used as it was originally specified, because SMIL mixes inside the structure of the UI the spatial structure and temporal behavior. This mixture is shown in the research by Bulterman [8] and it is not a desirable feature in order to maintain a clean separation of concerns. Thus, several modifications have been performed and some elements have been added in RUX.

In RUX it is possible to establish the desired values for the attributes of the elements implied in a behavior of the Temporal Presentation. This can be done for both, the initial and the final moments of the period for a temporal behavior (from-to as stated in Section 3.4). For this purpose, the triads have properties that do not affect the temporal relationships but affect the temporal behavior of the element. These properties allow the specification of which Component (partRef) and concrete property of the Component (modifiedProperty) change over time and what are the values (initialValue and endValue). If no value is indicated for any of them, it is assumed that it is the property value in the current moment.

The schema for the formal description of the Temporal Presentation specification has been omitted here for space reasons, but readers can access the schema online at [22].

4 Related Work

According to the available bibliography, four engineering fields have tried to support RIA modeling: Web, HCI, hypermedia and multimedia. However, temporal behavior specifications have not been widely integrated with Web and/or HCI proposals [42], [52]. Research contributions coming from the Web Engineering Model Driven community have appeared to extend its originals schemes to cover RIA capabilities; the main proposals are WebML-RIA [7], [13], OOH4RIA [29][30], and UWE for RIA [18]. UWE specifies patterns by means of UML state machines modeling the interaction, functionality, and presentation of typical RIA widgets (e.g., auto-completion of fields, periodic/dynamic refresh). Partial page refresh behaviors: this aspect is explicitly treated in WebML and OOH4RIA, where a dynamic model is introduced to address the computation of the Web page, where the parts to be refreshed/reloaded are specified.

To our knowledge, the hypermedia ones are not able to cover all the temporal relationships required by RIAs. As an example, HMT [49] is a widely cited temporal model for hypermedia applications, where the set of potential temporal relationships between two elements reaches seven connections, while inside the multimedia field it has been established that the full spectrum of simple temporal relationships between two elements is thirteen [3]. Other example is Ariadne [33] that allows a designer to model a hypermedia application and to generate dynamically XML + SMIL implementation templates. AriadneTool is an authoring tool supporting this model that incorporates a timeline-based temporal representation but is a very simple approach to cope with the dynamism of RIA UIs.

Regarding proposals from the Multimedia field that use SMIL, many research works have used it for different purposes. Limsee3 multimedia authoring model [12] provides a specification for the

creation of SMIL documents. It provides an ad-hoc notation for temporal behaviors and a code generation engine (using XSL) that target SMIL Timesheets (but in an older version when it was part of the SMIL 3 working draft) [28]. NCL [4] specifies synchronization relationships using links, avoiding one of the most usual criticisms about SMIL that obliges the document logical organization to match its temporal behavior [48]. A graphical non timeline-based representation is also presented for this model in [43]. A more recent approach is the Scalable MTSI Model [36] that splits the UI definition into three dimensions i.e., spatial, temporal and interactive axis, the same as the RUX Concrete UI level, being able to reuse any of these dimensions to cover some context (e.g., according to the target device capabilities). In [5], Advene application (based on SMIL) is detailed for the specification of hypervideo documents using timeline manipulation. However, none of the previous approaches support the full development process required to model a RIA, including the required connection with the business logic. The main reasons to adopt an SMIL-based approach instead of NCL or other similar approaches are that it is quite complex for the objective of our proposal, redundant elements were added to it (in order to facilitate authoring) according to [5], it is not a standard-based approach (so it could be discontinued and there are less related implementation works e.g., editors and libraries).

Many authors have proposed Timed Petri Nets (TPNs) as a model to capture the timing and synchronization information of multimedia objects specified in SMIL2.0 [11]. The authors incorporated a new type of transition; a couple of different types of state for places (using two tokens of three different types) and quite complicated firing rules that constrain the model to be used mainly by experts in TPNs. Indeed, other authors [57] have expressed their doubts regarding the feasibility of using a graph-based modeling mechanism such as TPN or E-RTSM (an extension of regular state machines) to completely capture the complex temporal relationships among media objects in SMIL2.0

Some of the closest works to RUX coming from the Multimedia field and trying to solve the latter limitation are XIMPF, OMMA and MML. XIMPF [16] claims to give support for multimedia content in different rendering platforms, but it does not explicitly specify related concepts such as adaptation (e.g., adjusts the logic or navigation level) or context (e.g., device features). Santachè et al. [44] defined a building block model based on the concept of digital content component but, as far as we know, is not based on standards and it has not been fully detailed, so it seems impossible to reuse it. OMMA-L [45] gives support to the modeling of structural, functional and dynamic aspects of a system and the corresponding UI. However, it is difficult to be used by non-expert users (e.g., graphic designers) due to the use of UML and its required extensions. MML [34] tries to strengthen the development of interactive multimedia applications and the model-driven development of UIs in HCI. However, this proposal does not take into account the last HCI advances while modeling UIs (e.g., it considers certain events at the Abstract Interface level making it not suitable for multi-device and multi-platform target environments). MML is used in [41] to integrate authoring tools into the model-based development of interactive multimedia applications. The main drawback is the complexity required to maintain the synchronization between the model and the changing and uncontrolled external authoring tool (e.g., Flash Builder). More recently, Pleuss et al. [40] have addressed some specific RIA issues through the introduction of Media Components with different abstraction levels. A survey of many different multimedia authoring systems is performed by Sung et al. [50] that also propose a graphical representation for SMIL supporting seven temporal relations (before, meets, overlaps, during, starts, finishes and equals).

The use of timelines is commonly used by Web designers for their mock-ups and for the informal specification of temporal behavior in some research papers [1], [9]). Many researchers have used or proposed formal and informal graphical representation of temporal behaviors. Pihkala et al. [37] provide an informal timeline-based representation of SMIL. This representation is able to mix passive and active behaviors with the UI structure into a single diagram. While this representation is clear for a simple example, it is not usable due to the number of arrows needed for a real project where the complexity is higher. Thompson et al. [51] provide an original visual representation based on Media Construction Abstractions (connected boxes).

Bertolotti et al. [6] presented both formal automation-based and informal SMIL timeline-based representations. Willrich et al. [57] detailed a hierarchical PetriNet-based formal representation. Indeed, many of the formal temporal representations use Petri Nets. Smilauthor [60] is a multimedia authoring tool being able to import and export from/to SMIL but using RTSM for the textual temporal behavior description. Yang et al. [58] follow a similar approach to the latter but introducing the concept of Dividable Dynamic Timeline. However, these visual representations while being very complete are too complex to be used by non-technical users.

Focusing on not completely standard approaches (e.g., XHTML + SMIL or SMIL 3.0 plus states), they are compared by Jansen et al. [17], but it is a market reality that the technology selected here is the widest supported one by current web browsers. HTML5+CSS3+JavaScript+SMIL-Timesheets combination was used by Cazenave et al. [10], but according to our experience, this approach does not cover the three goals presented here. Firstly, not being a model driven approach and not including transformations, it does not cover timing and synchronization in non-standard RIA technologies (e.g., Flash) and browser support is inconsistent according to our experience. Secondly, a temporal behavior visual specification is not provided. To specify those temporal behaviors programming knowledge is required.

Finally, many well-known RIA development tools such as Flash Builder [2] or Microsoft Expression Blend [31] include timeline specifications that have been designed for users without engineering background. However, these tools do not provide visual temporal relationships representation for SMIL, being based on their own proprietary languages. Also, according to our review at least neither of them covers the set of temporal relationship specified by Allen et al. [3]. Many other authoring tools for SMIL are listed at the Synchronized Multimedia homepage [56].

3.5 Timeline-based representation and XML-based specification

In order to collect the capabilities of the approaches identified in this related work section for covering our three goals to design temporal behaviors developing RIAs, we have fixed a simple comparison process based on a three level degrees process. Each degree represents the particular approach capability to design the objectives shown in Table 2. Suitability of each comparison goals is represented by one of the following Coverage Degrees:

- High Coverage Degree (HIG), which represents that the goal is covered by the methodology and it is ideal to design temporal behaviors in RIAs.
- Partial Coverage Degree (PAR), which is used when the parameter is moderately covered.
- None Coverage Degree (NON) is used when the goal is not suitably covered.

Each Coverage Degree has a specific weight. In this evaluation the proposed weights are: HIG = 3, PAR = 2 and NON = 0. Goal 1: the model must be usable by non-experts in engineering specifications; Goal 2: the model must be able to be implemented by an authoring model driven development tool: Goal 3: the applications specified using the temporal behaviors model must run in current Web browsers.

	Goal1	Goal 2	Goal 3
RUX [25]	HIG	HIG	HIG
OOH4RIA [29, 30]	HIG	HIG	LIM
WebML4RIA [7, 13]	LIM	LIM	LIM
UWE4RIA [18]	LIM	NON	NON
Rich-IDM []	LIM	NON	NON
HMT [49]	LIM	LIM	NON
Ariadne [33]	NON	LIM	NON
Limsee3 [12]	NON	LIM	NON
NCL [4]	NON	NON	NON
Scalable MTSI [36]	NON	NON	NON
Advene [5]	NON	LIM	NON
Bertolloti [6]	NON	NON	NON
Willrich [56]	NON	NON	NON
XIMPF [16]	LIM	LIM	NON
OMMA [45]	LIM	LIM	NON
MML [39]	LIM	LIM	NON

Table 2 Coverage degree reached to design temporal behaviors developing RIAs

Due to the results shown in Table 2, the most suited approaches to design temporal behaviors developing RIAs are represented by the Web approaches when multimedia capabilities are incorporated. Figure 5 depicts the comparison process results graphically.

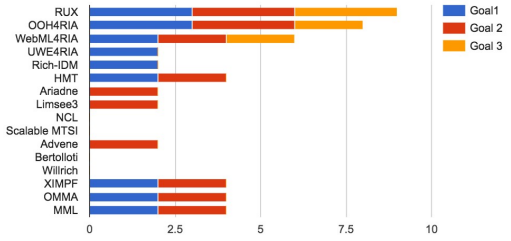

Figure 5 Results about the comparison process for covering the three goals to design temporal behaviors developing RIAs.

5 Implementation

RUX-Tool implementation can be divided into two parts: design-time and run-time. The full process is depicted in Figure 1. Left part of the Figure 1 shows the underlying Web model (e.g., UWE or WebML) at design-time and its corresponding Web application (full Web application or Web services) generated from this model and deployed at run-time. The rest of the Figure shows the RUX-Tool process that relies on RUX-Method until the RIA UI is obtained. From that moment, the RIA UI can be deployed and used at run-time. At design-time and according to RUX-Method, RUX-Tool extracts and analyses existing data and business logic offered by an underlying third party Web model. This

information provides a first UI abstraction which is enriched (adding events, temporal behaviors, and so on) and transformed using the mechanism specied by RUX-Method until the desired RIA UI is reached. Finally, at run time, RUX-Tool communicates with the Web application or services obtained using the Web model via Web communication standards and through a variety of languages. The RIA UI can be obtained for diferent RIA rendering technologies and adapted to diferent devices. This process is performed automatically because TR2 establishes the way the matching takes place among Concrete Interface Components and Final Interface Components.

Regarding the temporal behavior implementation, currently in RUX-Tool two transformations are internally performed. The first one transforms the Concrete temporal model to RUX adapted SMIL Timesheets by means of an XSL file. The other one transforms these Timesheets to JavaScript (and CSS to exploit the CSS hardware acceleration when possible), using the template-based transformations. The latter transformation is based on existing libraries such as Timesheets JavaScript Engine [53] and Timesheets Scheduler [35]. Even when two transformations are not an ideal situation for the code generation phase due to the dependencies generated and the time required to perform this generation, it was selected because of its speed and ease of development for the evaluation process explained latter. We are still working on an optimized code generator for the temporal behaviors able to minimize the time required by the generation itself while maximizing multi-device compatibility exploiting hardware acceleration available in modern browsers.

Our implementation experience has shown that CSS3 animations and transitions alone have serious limitations to specify the temporal behaviors required. Maybe the most important missed feature is nesting time containers, making impossible to synchronize multiple elements such as SMIL Timesheets is able to do. It is also not possible to implement an event (e.g., mouse over) on one DOM element and make the animation happen on a different DOM element. During our study, we also found that some authors controlled synchronization in JavaScript at run-time [10], [32], but according to our tests this solution based on JavaScript SMIL interpretation at run-time performed bad specially on mobile devices due to their limited computational capabilities.

One of the implementation advantages from the development approach proposed in this article is that when the model is able to describe the system, code generation to HTML5 and CSS3 ongoing standards can be achieved progressively. Currently, RUX-Tool provides compatibility with older browser using Modernizr[b] JavaScript library. So, regarding UIs created with RUX, e.g., two years ago, it is only needed to modify our code generation engine once, not the models, and re-generate the models to support the new HTML5 tags or CSS3 properties.

6 Evaluation

We performed two practical experiences regarding the temporal model developed. The first one was devoted to the evaluation of the model by real users. The second one was devoted to the evaluation of the proposal in a larger case study.

During the first experience, a group of companies was involved in a beta program. The companies were selected according to their human resources and RUX-Tool expertise and all of them obtained a

[b] https://modernizr.com/

free year license as consideration. Homeria Open Solutions knows well the users of RUX-Tool since Homeria provides personal support through different channels. So, according to our advice list, Homeria prepared a list of potential users for the evaluation process, selecting partner companies with designer profiles working on them and specifying preferred human resources when possible. The experiment was performed with eight participants. The reasons for these selection criteria are explained as follows.

On the one hand, focusing on designers is important. Even when RUX-Tool is mainly used by designers, in some companies, developers cover both roles (i.e., developer and designer) and it is the objective of this research to deal with non-engineering profiles, so we should filter them. On the other hand, preferred human resources were proposed to the companies because we want to evaluate the proposal with RUX-Tool experienced users, because our aim was measuring some indicators (e.g., learnability) only for the temporal model proposed excluding full RUX-Tool evaluation as much as possible. We want to separate as much as possible two issues that are interrelated: (1) the temporal behavior model proposed and (2) the temporal model implementation that involves the user's interaction with the temporal model through RUX-Tool as the selected authoring tool.

The temporal behavior model evaluation was focused on different indicators related to the objectives of this proposal. These indicators were selected according to the usability evaluation proposed by [34] [21]. They are **learnability** (how easy learning to use the model is), **efficiency** (how quickly a trained user can establish the desired temporal behavior), **memorability** (when users return to use the model after an inactivity period, how easily they can restore efficiency) and **satisfaction** (how pleasant using the temporal behavior model is). Error handling was also measured but only for implementation issues.

To measure these indicators and collect designer's feedback, we built a form with ten questions which answers are based on the Likert-type scale [20]. Users indicated their degree of agreement for each statement on a 1-5 scale (1 = strongly disagree, 2 = partly disagree, 3 = neither agree nor disagree, 4 = partly agree, 5 = strongly agree). Each indicator was measured through two questions in the form, e.g., "It was easy to learn how to use this system" and "Understanding the system to develop the example was easy". All the questions were randomly sorted in each form for each user.

The eight users were randomly split into two groups i.e., A and B with four users each. Group *A* attended to a tutorial workshop including a demo of the temporal modelling capabilities and a practical work to be individually completed. The users' screens were recorded during this session for error handling. Group *A* filled the form in after this one-hour session (without memorability questions).

Two weeks later both groups were asked to complete a simple web UI. During these weeks, the implementation bugs detected by group *A* in the tutorial session were also fixed. Participants of group *B* were provided with a video demo of 6 minutes explaining the temporal model and showing how to use it in RUXTool. Two different web applications, with two pages each, were sent to the designers. Two designers from group *A* and two from *B* received the application 1 and the other half of the designers received the application 2. The structure and look and feel of both applications were already developed to focus on the temporal behavior design. After specifying the temporal behavior of the application, all the designers filled the form in. Figure 5 summarizes the measures of the indicators selected for this evaluation attending to the two groups that were performed.

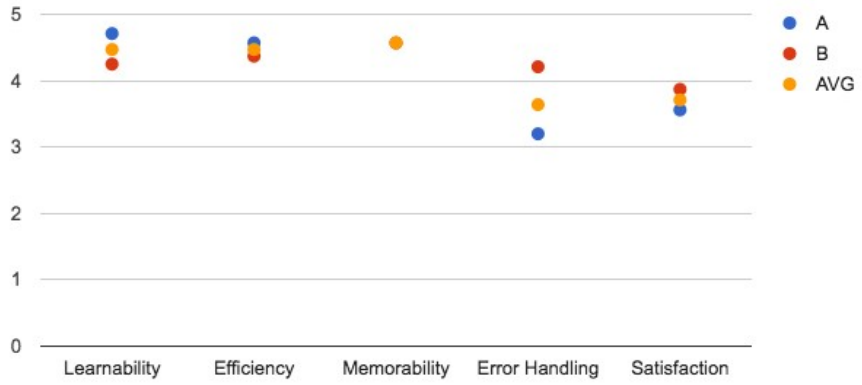

Figure 6 Evaluation questionnaire results.

Learnability. Group *A* designers used the temporal proposal with a tutorial session while group *B* ones used the proposal only with a short orientation. Using two learning methods and two different applications we have tried to avoid the dependency regarding the method used to show the temporal model specification and the dependency regarding the applications selected. The average rating of the questions related to learnability is 4.71 for group *A* and 4.25 for group *B*. From this result, we consider that non-experts can use the temporal behavior model in their practical environments without experts' help, even when a face-to-face session seems to improve learnability value 10% approximately.

Efficiency. The designers could specify the required temporal behaviors within the time that they estimate necessary to do it. We preferred not to fix the time for the operations because we wanted to recreate an evaluation environment as much similar to their real environments as possible. One of the two questions in the form regarding this indicator related the approach proposed in comparison with the method that they are currently using (i.e., hand-coded in all the cases using Javascript or whatever). However, it was important for us to know their impressions regarding the time they spent with the model and the time they estimated to complete the task. The average rating of the questions related to efficiency are 4.57 for group *A* and 4.37 for both groups during the second evaluation round. According to these results, we concluded non-experts consider the temporal behavior model quite efficient in relation to their perception and their current way to develop temporal behaviors.

Memorability. The second evaluation round shows an average of 4.57 in the group *A*, the only one that was evaluated. According to this result, we can consider the temporal behavior model a highly memorable model. Probably, its simplicity makes it not only easy to learn but also easy to remember. Designer's previous knowledge using other timeline-based tools (e.g., Flash Builder) could also influence this result.

Satisfaction. The average ratings of the questions regarding satisfaction show an average of 3.56 after the first evaluation round and 3.87 after the second round. Even the worst value 3.56 is known as a better estimate of neutral user satisfaction.

Designers were able to provide free comments in the form and we observed many positive comments such as "easy to use" or "interesting". Negative comments arose due to minor

implementation bugs and feedback for error handling. Two designers suggested that the model could be complex when many temporal behaviors were required.

For the second experience, an open source RIA task sequencer for people with special needs working in special employment centers has been recently released. In this application, the temporal behavior model plays a pivotal role because many issues such as data-driven presentations, slideshows, and avatars among others must be data-driven timed and/or synchronized. The application target platform is based on HTML+CSS+JavaScript W3C standards but it mixes other RIA technologies for specific issues e.g., flash to capture video from the webcam. The application was developed as part of the Assistive Technologies Chair that our research group leads at the University of Extremadura.

7 Final conclusions and future work

RUX-Method is currently being used by several companies through its authoring tool, RUX-Tool. Based on the experience developing some complex real projects it can be affirmed that the approach here presented is a solid base for the specification of temporal relationships of synchronization in RIAs. Due to the system being based on components, it is quite straightforward to add a new event to the temporal relationships in order to give support to existing or forthcoming RIA technologies. Moreover, we think that the implementation of these ideas into an existing professional authoring tool gives an extra value to the research work.

Although some of the ideas and adaptations that are specified here are ad-hoc (i.e., RUX-Method dependent), we encourage that many of them have a general validity. So, other HCI models (e.g. UsiXML, UIML) could take advantage of them in order to include temporal behaviors as part of their UI models. Beyond functionality, nowadays dynamic/rich UIs play a pivotal role in the success of (e.g., smartphone) applications and the inclusion of passive and active behaviors in a model-driven approach should benefit multi-device UI development. Due to the lack of a standard model-driven UI description language and the broad spectrum of approaches to this issue, that inclusion must be carried out individually and it is the researcher's option to select SMIL, SMIL Timesheets or any other approach. In our case, the selection was SMIL Timesheets due to the "simplicity" of the temporal behaviors required and the facilities it offers to be connected with other languages. We encourage using a standard approach due to the advantages that it implies especially about support (there are larger communities of users) and related resources such as libraries, engines, and frameworks. Our knowledge of other HCI proposals [19] allows us to ensure that a similar approach could be applied to them.

Acknowledgements

The authors gratefully acknowledge the support of TIN2015-69957-R (MINECO/FEDER, UE) project, Consejería de Economía e Infraestructuras/Junta de Extremadura (Spain)- European Regional Development Fund (ERDF)- GR15098 project and IB16055 project and Homeria Open Solutions, S.L. to the work here presented.

References

1. Abdelli, A. and Badache, N., Context-aware adaptation of multimedia documents for consistent presentations. Multimedia systems 17(6), (2011), 465–486
2. Adobe: Flash Builder. Available at: http://www.adobe.com/devnet/flash-builder.
3. Allen, J., Maintaining knowledge about temporal intervals. Communications of the ACM 26(11), (1983), 832–843
4. Antonacci, M., Muchaluat-Saade, D., Rodrigues and R., Soares, L., Improving the expressiveness of XML-based hypermedia authoring languages. In: Multimedia Modeling Conference. Nagano, Japan (2000). DOI 10.1142/9789812791993 0006
5. Aubert, O., Champin, P., Prié, Y. and Richard, B., Canonical processes in active reading and hypervideo production. Multimedia Systems 14(6), (2008), 427–433
6. Bertolotti, P. and Gaggi, O., A study on multimedia documents behavior: a notion of equivalence. Multimedia Tools and Applications 33, (2007), 301–324, DOI 10.1007/s11042007-0102-2
7. Bozzon A. Comai S., Fraternali P. and Toffetti G., Conceptual Modeling and Code Generation for Rich Internet Applications. In: International Conference on Web Engineering (ICWE), ACM 263, Palo Alto, USA, (2006), 353-360.
8. Bulterman, D., SMIL 2.0 part 1: overview, concepts, and structure. IEEE Multimedia 8, (2002), 82–88, DOI 10.1109/93.959106
9. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L. and Vanderdonckt, J., A unifying reference framework for multi-target user interfaces. Interacting with Computers 15(3), (2003), 289–308
10. Cazenave, F., Quint, V. and Roisin, C., Timesheets.js: when SMIL meets HTML5 and CSS3. In: Proceedings of the 11th ACM symposium on Document engineering, DocEng '11, ACM, NY, USA, (2011), 43–52, DOI 10.1145/2034691.2034700
11. Chung, S. M., and Pereira, A. L., Timed Petri net representation of SMIL. *IEEE MultiMedia*, *12*(1), (2005), 64-72
12. Deltour, R. and Roisin, C., The limsee3 multimedia authoring model. In: Proceedings of the 2006 ACM symposium on Document engineering, DocEng '06, ACM, NY, USA, (2006), pp. 173–175, DOI 10.1145/1166160.1166203
13. Fraternali P., Comai S., Bozzon A. and Toffetti G., Engineering rich internet applications with a model-driven approach. ACM Trans. Web 4, 2, Article 7, (2010)
14. Gaggi, O. and Bossi, A., Analysis and verification of SMIL documents. Multimedia systems 17(6), (2011), 487–506
15. Helms, J. and Abrams, M., Retrospective on UI description languages, based on eight years experience with the User Interface Markup Language UIML. Int. J. Web Eng. Technol. 4, (2008), 138–162, DOI 10.1504/IJWET.2008.018095
16. Hendrickx, F., Beckers, T., Oorts, N. and de Walle, R.V., An integrated approach for device independent publication of complex multimedia documents. In: Internet and Multimedia Systems and Applications, (2005), 347–352

17. Jansen, J. and Bulterman, D.C., SMIL State: an architecture and implementation for adaptive time-based web applications. Multimedia Tools and Applications 43, (2009), 203–224, DOI 10.1007/s11042-009-0270-3
18. Koch N., Pigerl M., Zhang G. and Morozova T., Patterns for the Model-Based Development of RIAs. In: International Conference on Web Engineering (ICWE), Springer LNCS 5648, San Sebastián, Spain, (2009), 283-291.
19. Kuijk, F., Guimara˜es, R.L., Cesar, P. and Bulterman, D.C., Adding dynamic visual manipulations to declarative multimedia documents. In: 9th ACM Symposium on Document engineering, DocEng '09, ACM, NY, USA, (2009), 149–152, DOI 10.1145/1600193.1600227
20. LaLomia, M. and Sidowski, J., Measurements of computer satisfaction, literacy, and aptitudes: A review. International Journal of Human-Computer Interaction 2(3), (1990), 231–253
21. Lalomia, M. and Spitzberg, B., Preliminary Development of a Model and Measure of Computer-Mediated Communication (CMC) Competence. Journal of Computer-Mediated Communication, 11, (2006), 629–666, doi:10.1111/j.1083-6101.2006.00030.
22. Linaje M., Sanchez-Figueroa, F. and Preciado, J., RUX-Method temporal schema specification. Available at: https://s3-eu-west-1.amazonaws.com/homeria/RUX/RUXTemporalPresent.png
23. Linaje, M., Preciado, J.C., Morales-Chaparro, R., Rodriguez-Echeverria, R. and Sanchez-Figueroa, F., Automatic Generation of RIAs Using RUX-Tool and Webratio. In: 9th International Conference on Web Engineering, ICWE '9, Springer-Verlag, Berlin, Heidelberg, (2009), 501–504, DOI 10.1007/978-3-642-02818-2 48
24. Linaje, M., Preciado, J.C. and Sanchez-Figueroa, F., Domain-specific model for designing rich internet application user interfaces. In: V. Lopez Jaquero, F. Montero Simarro, J.P. Molina Masso, J. Vanderdonckt (eds.) Computer-Aided Design of User Interfaces VI, Springer London, (2009), 295–306, DOI 10.1007/978-1-84882-206-1 27
25. Linaje, M., Preciado, J.C. and Sanchez-Figueroa, F., Engineering rich internet application user interfaces over legacy web models. IEEE Internet Computing 11, (2007), 53–59 DOI 10.1109/MIC.2007.123
26. Linaje, M. and Preciado, J.C., Multi-device context-aware RIAs using a model-driven approach. Journal of Universal Computer Science 16(15), (2010), 2038–2059
27. Linaje, M., Rux-method: modelado de interfaces de usuario web multidispositivo, multimedia, interactivas y accesibles. Ph.D. thesis, Universidad de Extremadura, (2009)
28. Ma, H., Shin, K., Checking consistency in multimedia synchronization constraints. IEEE Transactions on Multimedia, 6(4), (2004), 565–574
29. Meliá S., Gómez J., Pérez S. and Díaz O., A Model-Driven Development for GWT-Based Rich Internet Applications with OOH4RIA. In: International Conference on Web Engineering (ICWE), New York, USA, (2008), 13-23.
30. Meliá S., Gómez J., Pérez S. and Díaz O., Architectural and Technological Variability in Rich Internet Applications. IEEE Internet Computing 14, 3 (2010), 24-32.
31. Microsoft: Expression Blend. Available at: http://www.microsoft.com/expression
32. Mikác, J., Roisin, C. and Le Duc, B., An export architecture for a multimedia authoring environment. In: Proceeding of the eighth ACM symposium on Document engineering, DocEng '08, ACM, NY, USA (2008), 28–31 DOI 10.1145/1410140.1410147
33. Montero, S., Fernandez, C., Dodero, J., Aedo, I. and Diaz, P., A design toolkit for hypermedia applications based on Ariadne development method. In: R. Jacob, Q. Limbourg, J. Vanderdonckt (eds.) Computer-Aided Design of User Interfaces IV, Springer Netherlands, (2005), 43–54, DOI 10.1007/1-4020-3304-4 4

34. Nielsen, J. and Hackos, J., Usability engineering, vol. 125184069. Academic press San Diego, (1993)
35. Nouguier, E., Timesheets Scheduler. Available at: http://limsee3.gforge.inria.fr/public-site/timesheets/
36. Pellan, B. and Concolato, C., Authoring of scalable multimedia documents. MultimediaTools and Applications 43, (2009), 225–252 DOI 10.1007/s11042-009-0268-x
37. Pihkala, K. and Vuorimaa, P., Nine methods to extend SMIL for multimedia applications. Multimedia Tools and Applications 28, 51/67, (2006), DOI 10.1007/s11042-006-5120-y
38. Pimentel, M.d.G., Cattelan, R.G., Melo, E.L., Prado, A.F. and Teixeira, C.A.C., Ubiquitous end-user live editing of interactive multimedia programs. In: 14th Brazilian Symposium on Multimedia and the Web, WebMedia '08, ACM, NY, USA, (2008), 123–129, DOI 10.1145/1666091.1666113
39. Pleuß, A., Botterweck, G. and Hußmann, H., Modeling advanced concepts of interactive multimedia applications. In: Visual Languages and Human-Centric Computing, (2009), 31–38 DOI 10.1109/VLHCC.2009.5295305
40. Pleuß, A. and Hußmann, H., Integrating authoring tools into model-driven development of interactive multimedia applications. In: 12th Int. Conf. on Human-computer interaction: interaction design and usability, HCI'07, Springer-Verlag, Berlin, Heidelberg, (2007), 1168–1177
41. Pleuß, A., MML: a language for modeling interactive multimedia applications. In: 7th IEEE International Symposium on Multimedia, (2005), 9–17 DOI 10.1109/ISM.2005.80
42. Preciado, J.C., Linaje, M., Sanchez, F. and Comai, S., Necessity of methodologies to model rich internet applications. In: 7th IEEE International Symposium on Web Site Evolution, IEEE Computer Society, Washington, DC, USA, (2005), 7–13, DOI 10.1109/WSE.2005.10
43. Rodrigues, L., Antonacci, M., Rodrigues, R., Muchaluat-Saade, D. and Soares, L., Improving SMIL with NCM Facilities. Multimedia Tools and Applications 16, (2002), 29–54 DOI 10.1023/A:1013289601682
44. Santanchè, A., Medeiros, C. and Pastorello, G., User-author centered multimedia building blocks. Multimedia systems 12(4), (2007), 403–421
45. Sauer, S. and Engels, G., Extending UML for modeling of multimedia applications. In: IEEE Symposium on Visual Languages, IEEE Computer Society, Washington, DC, USA, (1999), 80–87,
46. Scherp, A., Canonical processes for creating personalized semantically rich multimedia presentations. Multimedia Systems 14(6), (2008), 415–425
47. Schwinger, W., Retschitzegger, W., Schauerhuber, A., Kappel, G., Wimmer, M., Pr¨oll,B., Castro, C.C., Casteleyn, S., Troyer, O.D., Fraternali, P., et al., A survey on web modeling approaches for ubiquitous web applications. International Journal of Web Information Systems 4(3), (2008), 234–305
48. Silva, H.V.O., Rodrigues, R.F., Soares, L.F.G. and Muchaluat Saade, D.C., NCL 2.0: integrating new concepts to XML modular languages. In: Proceedings of the 2004 ACM symposium on Document engineering, ACM, NY, USA (2004), 188–197, DOI 10.1145/1030397.1030433
49. Specht, G. and Zoller, P., HMT: Modeling Temporal Aspects in Hypermedia Applications. In: 1st Int. Conf. on Web-Age Information Management, WAIM '00, Springer-Verlag, London, UK, (2000), 259–270
50. Sung, M. and Lee, D., A collaborative multimedia authoring system. In: M. Li, X.H. Sun,Q. Deng, J. Ni (eds.) Grid and Cooperative Computing, *Lecture Notes in Computer Science*, vol. 3033, Springer Berlin / Heidelberg, (2004), 311–318 DOI 10.1007/9783-540-24680-0 54

51. Thompson, S., King, P. and Cameron, H., Modelling reactive multimedia: Design and authoring. Multimedia Tools and Applications 27, (2005), 23–52 DOI 10.1007/s11042005-2713-9
52. Toffetti, G., Comai, S., Preciado, J.C. and Trigueros, M.L., State-of-the-art and trends in the systematic development of rich internet applications. Journal of Web Engineering 10(1), (2011), 70–86
53. Vuorimaa, P., Timesheets JavaScript Engine. Available http://www.tml.tkk.fi/~pv/timesheets/
54. W3C: SMIL 3 (2008). Available at: http://www.w3.org/TR/SMIL3/.
55. W3C: SMIL Timesheets 1.0 (2012). Available at: https://www.w3.org/TR/timesheets/.
56. W3C: Synchronized Multimedia Authoring Tools. Available at: http://www.w3.org/AudioVideo/#Authoring.
57. Willrich, R., De Saqui-Sannes, P., Sénac, P. and Diaz, M., Multimedia authoring with hierarchical timed stream Petri nets and java. Multimedia Tools and Applications 16, (2002), 7–27 DOI 10.1023/A:1013233517612
58. Yang, C. C., Wang, Y. C. and Tien, C. W., Synchronization modeling and its application for SMIL2. 0 presentations. Journal of Systems and Software, 80(7), (2007), 1142-1155
59. Yang, C.C., Chu, C.K. and WANG, Y.C., Extension of Timeline-based Editing for Nondeterministic Temporal Behavior in SMIL2.0 Authoring. Journal of information science and engineering 24(5), (2008), 1377–1395
60. Yang, C.C. and Yang, Y.Z., SMILAuthor: an authoring system for SMIL-based multimedia presentations. Multimedia Tools and Applications 21, (2003), 243–260 DOI 10.1023/A:1025770817293