Semantics of States and Transitions in statecharts-based markup languages: a comparative study between SWC and SCXML

Marco Winckler, Charly Carrère, Eric Barboni

Semantics of States and Transitions in statecharts-based markup languages: a comparative study between SWC and SCXML

Marco Winckler, Charly Carrére, Eric Barboni
ICS-Team, Institute of Research in Informatics of Toulouse (IRIT), Univerity Paul Sabatier (UPS)
118 route de Narbonne, 31062 Toulouse Cedex, France
{winckler, carrere, barboni}@irit.fr

ABSTRACT
Statecharts has been demonstrated as a suitable solution for specifying the navigational behavior of hypermedia systems. However, in order to cope with the idiosyncrasies of the Web development (such as representation of client and server stages) we have been proposed an extension to the original Harel's statecharts called StateWebCharts (SWC). In this paper we discuss the rationale for extending statecharts notations for specific application domains such as the Web. Moreover, we illustrate how the domain-specific constructs provided by SWC might help to solve problems that would require specific semantics for states and transitions. Then, we compare the constructs proposed by the SWC notation with Harel's statecharts and SCXML. We argue that it would be possible to convert SWC specification into SCXML by losing some semantic on transitions and states. Conversely, extensions for adding domain-specific semantics on SCXML would benefit not only its inner utility to specifying Web application but it could also useful in other application domains.

Author Keywords
Statecharts, Web navigation, Web applications, SCXML, StateWebCharts, SWC.

ACM Classification Keywords
H.5.m. Information interfaces and presentation (e.g., HCI).

INTRODUCTION
Research on navigation modelling has a long history in hypertext and hypermedia domain and it has strongly influenced the technology for the Web. State-based notations such as Petri nets [7] and StateCharts [2][4][6][7][9] have been explored to model navigation for hypertext systems. However, such proposals are not able to represent some aspects of Web applications such as dynamic content generation, support to link-types (toward external states, for instance), client and server-side execution. However, some of them [2, 7, 12] do not make explicit the separation between interaction and navigation aspects in the models while this is a critical aspect for the usability of Web application. Connallen [1] proposed an efficient solution for modelling Web applications using UML stereotypes. Such as an approach mainly target data-intensive applications and even propose prototyping environments to increase productivity. However, the limitation is that navigation is described at a very coarse grain (for instance navigation between classes of documents) and it is almost impossible to represent detailed navigation on instances of these classes or documents. The same problem appears in Kock [5] which may reduce creativity at design time as they impose the underlying technology and as they do not provide efficient abstraction views of the application under development. In order to cope with the idiosyncrasies of Web navigation, we have proposed in previous work [10] an extension to Harel's statecharts called StateWebCharts notation (SWC). Such as a notation dedicated constructs for modelling specificities of states and transitions in Web applications. Most elements included in SWC notation aim at providing explicit feedback about the interaction between users and the system.

In this paper we discuss the importance of representing domain-specific semantics that can be associated to states and transitions whilst using statechart-based markup languages such as SWC and SCXML [13]. We illustrate how domain-specific constructs can help to solve problems that would require specific semantics for states and transitions for the Web. We assume that the semantics of states and transitions of SWC might be specific to the Web domain and not easily generalizable. However, other application domains have their idiosyncrasies thus require different semantics for transitions and states. Nonetheless, we argue that by adding semantics to states and transitions of SCXML, it would be possible to employ SCXML as a markup language for copying with the same challenges addressed by SWC. In the next section we present the SWC notation and we illustrate its uses. Then we compare the syntax of construct present in the original Harel's statecharts, those proposed by SWC and SCXML markup languages.

EICS 2014 Workshop, Engineering Interactive Systems with SCXML, June 17, 2014, Rome, Italy

THE STATEWEBCHART NOTATION (SWC)

SWC is rooted on Harel's StateCharts [3] but it adds semantics to it to address Web domain issues. SWC' states are abstractions of containers for objects (graphic or executable objects). For Web applications such containers are usually HTML pages. States in SWC are represented according to their function in the modelling. In a similar way, a SWC transition explicitly represents the agent activating it. Each individual Web page is considered a container for objects and each container is associated to a state. Links and interactive objects causing transitions are triggered by events. The semantic for a SWC state is: current states and their containers are active while non-currents are hidden. Figure 1 show all SWC elements.

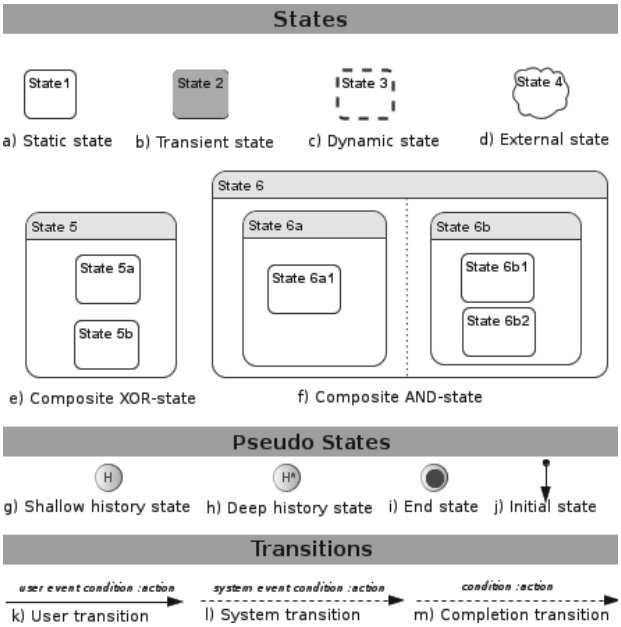

Figure 1. Graphical representation of StateWebCharts.

Static states (Figure 1.a) are the most basic structures to represent information in SWC. They refer to a container with a static set of objects; in a static state all objects are present in the browser. However, those objects are not necessarily static by themselves; they could have dynamic behaviour as we usually find, for example, in applets, JavaScript or animated images. Static is the default type.

Transient states (Figure 1.b) describe a non-deterministic behaviour in the state machine. Transient states are needed when a single transition cannot determine the next state for the state machine. Only completion or system events are accepted as outgoing transitions of transient states. Transient states only process instructions and they do not have a visual representation towards users. They refer to server-side parts of Web applications, such as PHP scripts.

Dynamic states (Figure 1.c) represent content that is dynamically generated at runtime. They are usually the result of a transient state processing. The associated container of a dynamic state is empty. The semantics for this state is that in the modelling phase designers are not able to determine which content (transitions and objects) will be made available at run time. However, designers can include static objects and transitions inside dynamic states; in such case transitions are represented, but the designer must keep in mind that missing transitions might appear at run time and change the navigation behaviour.

External states (Figure 1.d) represent information that is accessible through relationships (transitions) but are not part of the current design. For example, consider two states A and B. While creating a transition from A to B, the content of B is not accessible and cannot be modified. Thus, B is considered external to the current design, which is often the case of external sites. External states avoid representing transitions without a target state, however all activities (i.e. entry, do, and exit) in external states are null.

SWC's events indicate the agent triggering them: user (e.g. a mouse click), system (e.g. a method invocation that affects the activity in a state) or completion (e.g. execution of the next activity). A completion event is a fictional event that is associated to transitions, e.g. change the system state after a timestamp. This classification of event sources is propagated to the representation of transitions. Transitions whose event is triggered by a user are graphically drawn as continuous arrows (Figure 1.k.) while transitions triggered by system or completion events are drawn as dashed arrows (Figure 1.l and Figure 1.m, respectively).

In order to represent behaviour such as those found in StateCharts, SWC provides the following pseudo-states (g) shallow history, (h) deep history, (i) end state and (j) initial state. These pseudo-states do not have any container associated to them. Pseudo-states and composite state in SWC are very close of the definition given by StateCharts (see [10] for details). Both states and transitions can have associated actions. When associated to transitions, actions represent what is executed by the system while traversing a transition. When associated to state, actions represent the activity performed by the state. All SWC constructs are stored in a XML format as illustrated at the Figure 2.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with SWCEditor -->
<swc>
<CompositeState id="root" label="root" file="null"
initial="S1" concurrent="false">
<BasicState id="S1" label="main intro" type="BasicState"
file="spider_intro.html" >
</BasicState>
<BasicState id="S2" label="schedule" type="BasicState"
file="spider_schedule.html">
</BasicState>
...
</CompositeState>
<Transition id="t1" type="user" label="" source="S1"
target="S2" trigger="mouseClick" guard="true" action="">
</Transition>
...
</swc>
```

Figure 2. XML file describing a SWC model.

SWC IN PRACTICE

SWC models can be built using the tool SWCEditor [11] which supports the creation, edition, visualisation, simulation and analysis of SWC models. Hereafter we illustrate how the some elements of the SWC notation have been operationalized using the SWCEditor to solve problems associated to navigation modelling of Web applications.

Separation between client/server states

One of the main features of SWC is the possibility to associate specific semantics for states and transitions in the navigation diagrams. Figure 3 illustrates these semantics by a simple SWC statemachine diagram which models the navigation behaviour for client/dynamic/transient states and user/system driven transitions.

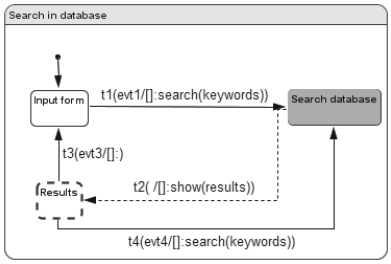

Figure 3. Navigation modelling client/dynamic/transient states and user/system transitions.

As we shall see at Figure 3, states are depicted accordingly to the semantic given to states. For example, the state "input form" is a Web page that contains a web form whilst the page "results" is automatically generated at the client-side (i.e. the browser) as a response to an execution of a state that can only be processed on the server side (i.e. "search database"). User driven transitions, depicted with continuous lines, are interpreted as users' clicks whilst transitions automated by the system (ex. "t2") are depicted as dashed lines. Such as inner semantic for states and transitions can properly mapped to the proper constructs used to build the Web sites.

Setting boundaries between local and external models

During early evaluation phases of development designers have to check if abstract modelling will behave as expected. Simulations of models can be useful for that purpose. Thanks to the special constructs of SWC it is possible to associate navigation model with advanced Web prototypes. Figure 4 presents how SWCEditor allows simulation and co-execution of SWC models. First of all, let us to focus on the left part of the figure 4. There are two windows: the simulator window (at top-left) and the visualization window (at bottom-left). The window simulator is composed of two panels showing: the set of active state (grey panel at left) and the set of enabled transitions at a time (white panel at right). The visualization window is the main graphic editor of SWC models (the SWCEditor module).

When an enabled transition is selected the system fires it immediately causing the changing of the system, which displays the next stable configuration. The current statemachine configuration is shown in red. If a container is associated to a state, it is possible to concurrently display the corresponding container (typically a Web page) in a browser during the simulation. The concurrent simulation of model and implementation is suitable during the prototyping activity. Thus, designers can follow the changes in the abstract specification at the SWCEditor as well as its concrete implementation at the Web browser. Figure 4 shows in a browser window (at right part) the corresponding Web page for the current state in that simulation. Notice that external states are used to represent external links attached to the current web site design.

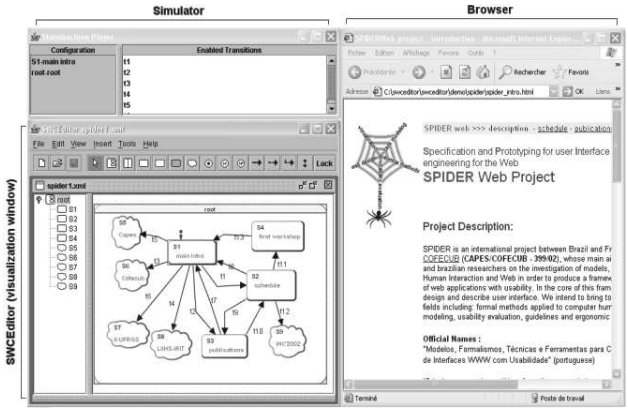

Figure 4. Co-execution of navigation models and Web prototypes.

Automated usability inspection of SWC models

One of the advantages of the semantic added to constructs is to support the reason about models in a certain way. In previous work [14] we have investigated how to use SWC models to support guidelines verification in early phases of development. The basic idea was to map concepts present in ergonomic guidelines (ex. "page") to SWC constructs as show in Figure 5. After that, we have implemented automated parsers for guidelines such as "*Each **page** must have a **link** to it*" that inspect SWC models as follows "*Each **state** must have a **transition** pointing to it*". Those tools thus exploit the semantic of models for automatically inspecting models in

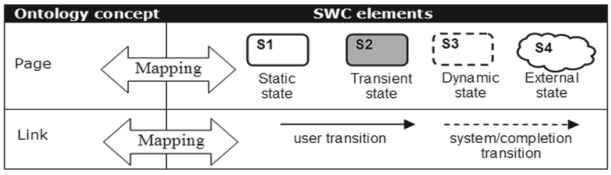

Figure 5. Mapping SWC constructs and Ontological concepts.

COMPARING NOTATIONS

In order to assess the expressiveness power of SWC, we compare in Table 1 its constructs with those defined by the original Harel's statecharts and duly supported by SCXML.

Table 1. Comparing constructs in Harel's statecharts, SCXML and SWC.

Harel's	SCXML	SWC
Statemachine	The language start by an <scxml> tag, example: **<scxml>** <state> … </state> **</scxml>**	The language start by an <swc> tag, ex : **<swc>** <state type="BasicState"> … </state> **</swc>**
States	Basic state reference, example: <state> … </state>	<**BasicState**> … </BasicState> Possible types: Basic/Static/TransientState/External
Composite State	Composition defined by inner hierarchy, example: <state id="S" **initial="s1"** > <state id="s1"> </state> </state> --- **AND states**: Classic state hierarchy, ex : <state id="S" initial="s1" > <state id="s1"> </state> </state> --- **OR states**: The <parallel> element encapsulates a set of child states which are simultaneously, ex : **<parallel id="Test5P">** <state id="Test5PSub1" initial="Test5PSub1Final"> <final id="Test5PSub1Final"/> </state> <state id="Test5PSub2" initial="Test5PSub2Final"> <final id="Test5PSub2Final"/> </state> <onexit> <log expr="'all parallel states done'"/> </onexit> **</parallel>**	Dedicate state type, ex : <**CompositeState** id="root" label="root" file="null" initial="S1" concurrent="false"> </**CompositeStat**e> --- **AND states**: <CompositeState id="root" label="root" file="null" initial="S1" **concurrent="false"** /> --- **OR states**: <CompositeState id="root" label="root" file="null" initial="S1" **concurrent="true"**/>
History	Determined by a pseudo-state, ex : <history type="deep" id="history-actions"> </history>	Determined by a pseudo-state, ex : <CompositeState id="root" …> <**DeepHistory** id="S1" /> <**ShallowHistory** id="S2" /> </CompositeState>
Final states	Determined by a pseudo-element, ex : **<final id="Test5PSub1Final"/>**	Determined by a pseudo-element, ex : <**EndState** id="S1" />
Variables	<datamodel> is a wrapper element which encapsulates any number of <data> elements, ex : **<datamodel>** <**data** id="door_closed" expr="true"/> **</datamodel>** --- **<script>** time.setHours(_event.data.currentHour + (_event.isAm ? 0 : 12) - 1); **</script>**	The name and value are in the parameter declaration, ex : **<parameters>** <parameter name="param1" value="0" /> **</parameters>**
Conditions	The conditions are defined using multiple tags, ex : **<if cond="true">** **<foreach array="cart.books" item="book">** <log expr="'Cart contains book with ISBN ' + book.isbn"/> **</foreach>** **<elseif cond="false"/>** <log expr="You can't use it"/> **</else>** <log expr="Error boolean"/> **</if>**	Condition in transition definition, ex : <Transition id="t1" type="user" label="" source="S1" target="S2" **trigger="mouseClick"** guard="true" action="" />
Action	<state id="s1" initial="s11"> **<onexit>** <log expr="'leaving s1'"/> **</onexit>** **<onentry>** <log expr="'entering S'"/> **<onentry>** </state>	Action defined in the Transition definition, ex : <Transition id="t1" type="user" label="" **source="S1" target="S2"** trigger="mouseClick" guard="true" **action="methodCall()"** />
Transition	<**transition** event="ping" target="takeOrder"/>	**<Transition** id="t2" type="user" label="" source="S1" target="S3" trigger="mouseClick" guard="true" action=""> **</Transition>**
External communication	**<invoke** id="timer" type="x-clock" src="clock.pl"> **<finalize>** <script> time.setHours(_event.data.currentHour + (_event.isAm ? 0 : 12) - 1); </script> **</finalize>** **</invoke>** --- **<send target="csta://csta-server.example.com/" type="x-csta">** <content> <csta:MakeCall> <csta:callingDevice>22343</callingDevice> <csta:calledDirectoryNumber>18005551212</csta:calledDirectoryNumber> </csta:MakeCall> </content> **</send>**	

As we shall see in Table 1, SWC and SCXML cover most of the original elements proposed by Harel's statecharts. Nonetheless, a few elements differ with respect to the inner Document Type Definition (DTD) they implement. Indeed, whilst SWC features a specific tag, SCXML implicitly represent for composite states by adding sub-states inside the tags. In addition, actions in SCXLM are represented by dedicated tags whilst SWC embedded them as expressions associated to attributes elements in the tag transition. Some tags have different names for addressing the same element, ex. *final* and *endstate* for indicating end pseudostates. Most of these differences are syntactic and can be easily overcome by a few transformation rules ensuring the compatibility between notations.

However, SWC does not take into account complex external communication mechanisms. Further investigate would be required to determine in which extension *communication mechanisms* could correspond to *dynamic* states in SWC. In all cases, all these difference worth to be carefully discussed, and would require extension in both notations if compatibility should be assured.

Lastly but not less important, a significant difference between SWC and SXCML is that the latter one provides a generic representation of states and transitions without any domain-specific semantics, whilst the former clearly features a semantics for navigation of Web application. Indeed, SWC offers four alternative types which specific semantics for basic states, whilst SCXML only provides one type of state. This is observable by the attribute *type* that can be associated to *states* and transitions. Moreover, SWC also provides another attribute to states that allows the mapping to contents, namely *file*.

CONCLUSION

SWC and SCXML are both based on Harel's statecharts and therefore share many similarities. In some extensions, models built in one notation could be translated to another, however, the compatibility is not 100% accurate and we would lose semantic and functionality in this operation. Further studies are required to determine the compatibility level and the side-effect implications of converting models. But still, we estimate that some level of compatibility ensured by model-transformation is possible.

However, if we consider the Web as a suitable application domain for SCXML we might argue that this notation lacks of some attributes to express the rich semantic of navigation. This lack of semantics of states and transitions would prevent the reasoning about the application and the development of dedicated tools as illustrated by the research around SWC. Moreover, we assume that this lack of semantics might not be specific to Web navigation models and other researchers would be interested in proposing other elements.

The proliferation of DLS might not be a definite solution for similar problems in different application domains. For that purpose, as standard language such as SCXML would be ideal as a lingua franca between statechart-based DSL like SWC. We argue that the level of semantic expected for state and transitions in SCXML could be easily solved by a couple of attributes that could be added to markup language. If so, we could pursue the research about navigation modeling using SCXML as a replacement to SWC and still achieve similar results as those previous illustrated in this paper.

REFERENCES

1. Connallen, J. Building Web Applications with UML. Addison-Wesley, 1999.
2. Dimuro, G. P.; Costa, A. C. R. Towards an automata-based navigation model for the specification of Web sites. In: 5th Workshop on Formal Methods, Gramado, 2002. Electronic Notes in Theoretical Computer Science.
3. Harel, D. StateCharts: a visual formalism for computer system. Science of Computer Programming, 8, N. 3:231-271 p., 1987.
4. Horrocks, I. Constructing the User Interface with Statecharts. Addison-Wesley, 1999.
5. Koch, N.; Kraus, A. The expressive Power of UML-based Web Engineering. In 2nd Int. Workshop on Web-oriented Software Technology (IWWOST02). June 2002.
6. Leung, K., Hui, L., Yiu, S., Tang, R. Modelling Web Navigation by StateCharts. In proc. 24th Inter. C.S.A., 2000, Electronic Edition (IEEE Computer Society).
7. Oliveira, M.C.F. de; Turine, M. A. S.; Masiero, P.C. A Statechart-Based Model for Modeling Hypermedia Applications. ACM TOIS. April 2001.
8. Stotts, P. D.; Furuta, R. Petri-net-based hypertext: document structure with browsing semantics. ACM Trans. on Inf. Syst. 7, 1 (Jan. 1989), Pages 3 - 29.
9. Turine, M. A. S.; Oliveira, M. C. F.; Masieiro, P. C. A navigation-oriented hypertext model based on statecharts. In Proc. 8th ACM Hypertext. 1997, Southampton, UK.
10. Winckler, M.; Palanque, P. StateWebCharts: a Formal Description Technique Dedicated to Navigation Modelling of Web Applications. DSVIS'2003, Portugal, June 2003.
11. Winckler, M.; Barboni, E.; Farenc, C.; Palanque, P. SWCEditor: a Model-Based Tool for Interactive Modelling of Web Navigation. International Conference on Computer-Aided Design of User Interface - CADUI'2004, Funchal, Portugal, 13-16 January 2004.
12. Zheng, Y.; Pong, M. C. 1992. Using statecharts to model hypertext. In Proc.of the ACM Conference ECHT'92, Milan, Italy. ACM Press, New York, NY, 242-250.
13. State Chart XML (SCXML): State Machine Notation for Control Abstraction. W3C Candidate Recommendation 13 March 2014. At: http://www.w3.org/TR/scxml/
14. Xiong, J., Farenc, C., Winckler, M. Towards an Ontology-based Approach for Dealing with Web Guidelines. In Proc. Int. Workshop on Web Usability and Accessibility. Auckland, New Zealand, September 1-4, 2008. Springer LNCS 5176, pages 132-141.