SCXML for Building Conversational Agents in the Dialog Web Lab

David Junger, Torbjörn Lager and Johan Roxendal

Department of Philosophy, Linguistics and
Theory of Science, University of Gothenburg.
Department of Swedish, University of Gothenburg.
tffy@free.fr, lager@ling.gu.se, johan.roxendal@gu.se

1. Introduction

The W3C has selected Harel Statecharts, under the name of State Chart XML (SCXML), as the basis for future standards in the area of (multimodal) dialog systems (Barnett et al. 2012). In an effort to educate people about SCXML we are building a web-based development environment where the dialogs of embodied, spoken conversational agents can be managed and controlled using SCXML, in a playful and interesting manner.

2. SCXML = STATE CHART XML

SCXML can be described as an attempt to render Harel statecharts (Harel 1987) in XML. Harel developed his statecharts as a tool for specifying reactive systems in great detail. In its simplest form, a statechart is just a finite state machine (FSM), where state transitions are triggered by events appearing in a global event queue.

Just like ordinary FSMs, statecharts have a graphical notation. Figure 1 depicts a very simple example:

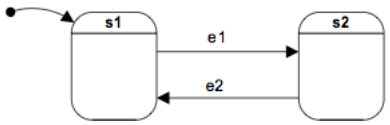

Figure 1: A simple statechart.

Any statechart can be translated into a document written in the linear XML-based syntax of SCXML. Here, for example, is the SCXML document capturing the statechart in Figure 1:

```
<scxml>
  <state id="s1">
    <transition event="e1" target="s2"/>
  </state>
  <state id="s2">
    <transition event="e2" target="s1"/>
  </state>
</scxml>
```

The document can be executed by an SCXML conforming processor, providing for just a small step from a specification into a running application.

Harel (1987) also introduced a number of (at the time) novel extensions to FSMs, which are also present in SCXML. Most importantly, the ability to specify hierarchical state machines as well as machines running in parallel takes care of some of the problems of ordinary FSMs – in particular the notorious state explosion problem. Furthermore, a complex state may contain a *history state*, serving as a memory of which substate S the complex state was in, the last time it was left for another state. Transition to the history state implies a transition to S. In addition, the SCXML standard also provides authors with means for accessing external web-API:s to for example databases.

The expressivity of SCXML makes it possible not only to specify large and very complex dialog managers in the style of FSMs, but also to implement more sophisticated dialog management schemes such as the Information-State Update approach (Kronlid & Lager 2007).

The work on SCXML is almost completed and several implementations exist. At the time of writing, the most conforming implementation, written by the third author, is called PySCXML and is implemented in Python. We are also aiming at an implementation in JavaScript, and a first version exists, written by the first author in collaboration with the third. The latter implementation is used in the Dialog Web Lab.

3. Speech recognition in Google Chrome

Recent versions of the Google Chrome browser offer web developers additional markup for creating <input> fields into which users can either write something in the usual way, or, by clicking a small microphone icon that is part of the widget, speak something that is transcribed into text. We have styled the widget by means of CSS and use it in the Dialog Web Lab in the way shown in Figure 2.

4. Speech synthesis and animated faces

SitePal (www.sitepal.com) is a commercial cloud platform for building speaking avatars, developed by Oddcast, that allows users to deploy “virtual employees" on websites that can welcome visitors, guide them around the site and answer questions. The avatars move their lips in synch with the speech, are capable of some non-verbal behavior, and can express emotions through facial expressions. The speech and bodily behavior of avatars can be controlled from JavaScript and thus also from our SCXML implementation.

5. The Dialog Web Lab

The idea behind the Dialog Web Lab is simply to allow developers of conversational agents access to an application

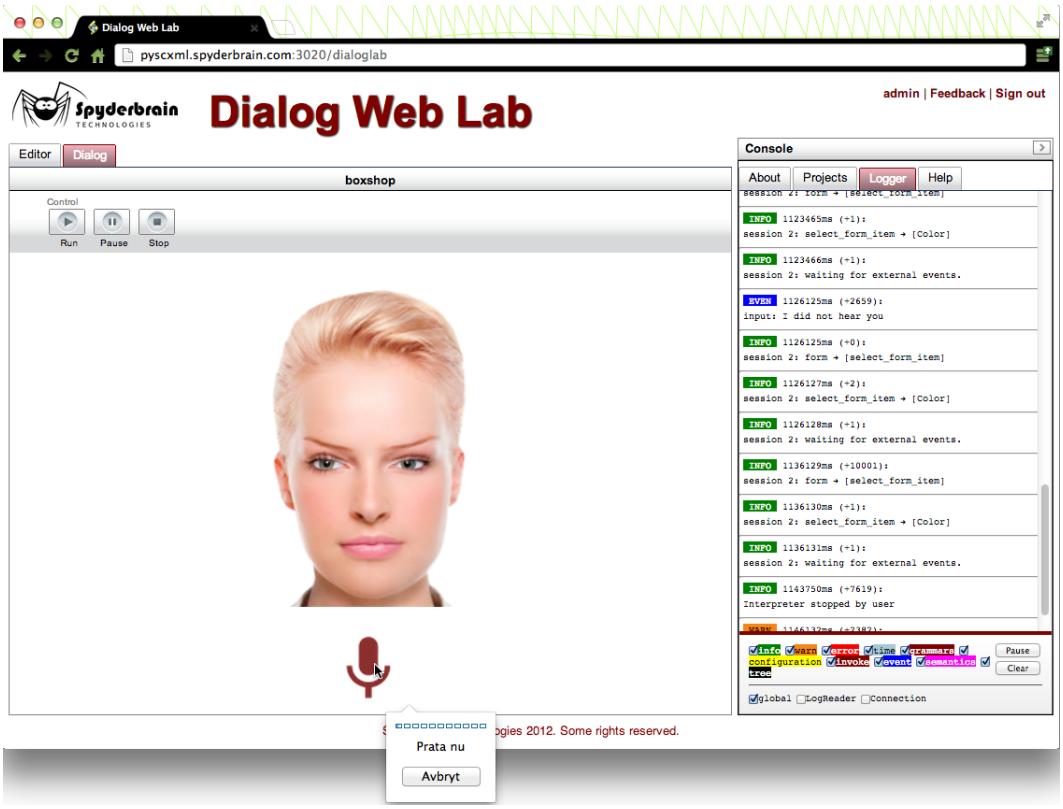

Figure 2: The current version of the Dialog Web Lab.

where they can explore the use of SCXML as a tool for dialog management.

In the current version of the Dialog Web Lab, depicted in Figure 2, the user has authored an SCXML document, is now executing it, and has just clicked the microphone. The user may now speak to the agent which, controlled by the SCXML process, will respond using speech and possibly non-verbal behavior. The Logger is activated, allowing the user to trace the execution in great detail.

6. Future work

We see several ways in which the Dialog Web Lab could be improved and extended:

- The present version of the Dialog Web Lab only works in recent versions of Google Chrome. However, The HTML Speech Incubator Group, in collaboration with the Voice Browser Working Group, has begun to specify a standard that integrates speech technology in HTML5, in the form of both markup and a JavaScript API. The speech recognition implemented in Google Chrome should, we believe, be seen as a preview of what to expect. At this point in time, the work on the standard has only just started, but once it is finished, and widely implemented, this will likely mean that the Dialog Web Lab can take advantage of it.

- In a future version of the Dialog Web Lab we plan to allow a user to control more than one conversational agent at the same time. This might be an interesting way to exercise the parallelism available in SCXML, and an interesting way to experiment with multi-party dialog involving conversational agents.

- As it turns out, a large fragment of VoiceXML can very easily be compiled into SCXML. We plan to allow authors to mix SCXML and VoiceXML in one document, compile it into SCXML and then run it. We believe that this will make possible very succinct expressions of dialog management strategies.

- The statechart formalism is a graphical language. So one idea would be to allow authors to draw statecharts on a canvas, compile them into SCXML, and then run them. Unfortunately, actually building such a graphical editor that works in a browser is not easy.

7. References

Barnett, Jim (Ed.) (2012) State Chart XML (SCXML): State Machine Notation for Control Abstraction, W3C Working Draft 16 February 2012. <http://www.w3.org/TR/scxml/>

Harel, David (1987) Statecharts: A Visual Formalism for Complex Systems, In: *Science of Computer Programming* 8, North-Holland.

Kronlid, Fredrik and Lager, Torbjörn (2007). Implementing the Information-State Update Approach to Dialogue Management in a Slightly Extended SCXML. In Ron Artstein and Laure Vieu (Eds.) *Proceedings of the 11th International Workshop on the Semantics and Pragmatics of Dialogue* (DECALOG), Trento, Italy, s. 99-106.