

SCXML AND VOICE INTERFACES

Graham Wilcock

University of Helsinki, Finland

Abstract. The paper shows how State Chart XML (SCXML) can support the design of voice interfaces. After describing the W3C Data Flow Presentation (DFP) framework, SCXML is illustrated by a simple stopwatch example. When the presentation layer's initial graphical interface is extended by adding voice components, the SCXML-based flow layer can be reused unchanged, thanks to the clean separation between the flow layer and the presentation layer.

1 Introduction

State Chart XML (SCXML) (W3C 2007a) is a general-purpose event-based state machine language. It is expected to play a role in more than one area where W3C is currently developing technical architectures. One area is the Voice Browser Activity, another is the Multimodal Interaction (MMI) Activity. In Section 2 we review the W3C's recent discussions of the technical architectures in these areas.

Section 3 presents a simple example SCXML application. Section 4 describes how the application can be extended by adding new components in the presentation layer, while reusing the same SCXML-based flow layer.

2 W3C architectures

The Voice Browser Activity has developed the Data - Flow - Presentation (DFP) architecture (W3C 2006), outlined in simplified form in Figure 1 (Barnett et al. 2006). The data layer manages data for the application. The flow layer controls the logic or application flow. The presentation layer handles interaction with the user.

The separation of concerns on which the DFP architecture is based is an instance of the successful Model - View - Controller (MVC) design pattern that is widely used in software engineering. Mapping to MVC terms, the DFP data layer is the MVC model, the flow layer is the MVC controller, and the presentation layer is the MVC view.

In Figure 1, the presentation layer shows two distinct presentation components: XHTML for web browsers and VoiceXML for voice browsers. This is therefore an example of the Multimodal Architecture which is being developed in parallel by the W3C Multimodal Interaction Activity. According to the W3C DFP document, the Voice Browser DFP framework is intended as a "voice-centric instance" of the wider multimodal architecture. "The data layers are identical. The MMI's runtime frame-

work corresponds to the flow layer. And MMI's modality components correspond to DFP's presentation components" (W3C 2006).

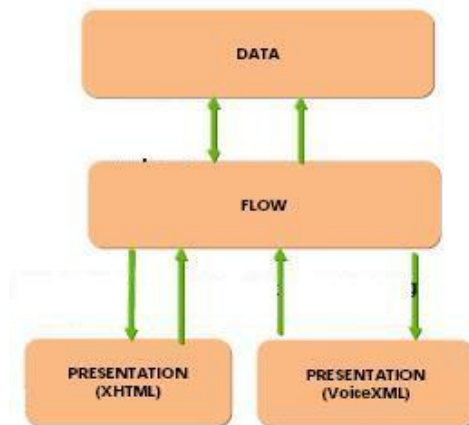


Figure 1. DFP Architecture

When the DFP architecture serves as a framework for multimodal applications, the presentation layer handles multiple interactions with users. In this case, the flow layer is responsible not only for controlling the application flow but also for coordinating the presentation components.

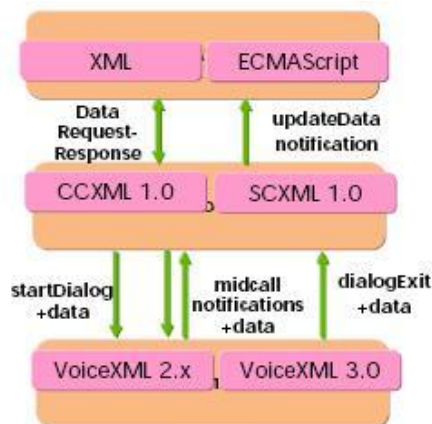


Figure 2. DFP for Voice Applications

Figure 2 (Barnett et al. 2006) shows the DFP architecture for voice applications. The existing CCXML standard (W3C 2005) is shown in the flow layer. Although it is

possible to use it to control application flow, CCXML was not intended as a general purpose flow control language but was designed to handle telephony-specific details such as DTMF. By contrast, SCXML is intended to be a general purpose flow control language suitable for use in the flow layer.

In the presentation layer, Figure 2 shows both the current and the future versions of VoiceXML. Although the current VoiceXML 2.1 (W3C 2007b) can be used to handle voice interaction with the user, it includes `<goto>` statements whose purpose is to manage the flow of control. So VoiceXML 2.1 does not adhere to the clear separation of flow and presentation which is one of the main goals of the DFP architecture. By contrast, VoiceXML 3.0 (Barnett et al. 2006) will be based on the DFP architecture.

The clear separation of the flow layer and presentation layer should bring several advantages. It should simplify code reuse, as the presentation code is not tangled up with `<goto>` logic. It should improve intelligibility of the flow description, which is free of presentation details. It allows a natural extension to multiple modes, when the same flow layer is used with multiple presentation components.

3 SCXML

SCXML is State Chart XML (W3C 2007a). It is a general-purpose event-based state machine language which combines concepts from CCXML (W3C 2005) and Harel Statecharts (Harel 1987).

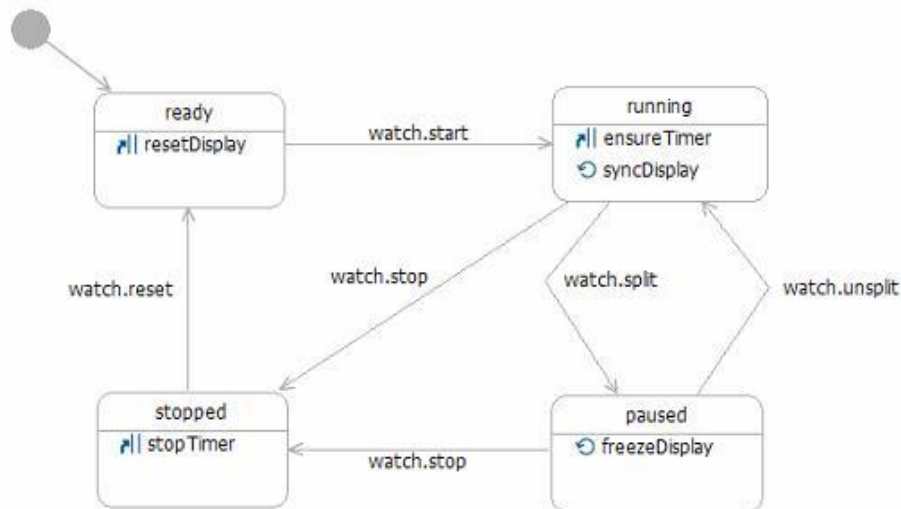


Figure 3. Stopwatch example statechart

An open source Java implementation of SCXML is available from Apache Jakarta (ASF 2006), including a simple stopwatch demo application. We use both the Java implementation and the stopwatch example in the work described here. The statechart for the stopwatch example is shown in Figure 3 (ASF 2006).

SCXML describes states, events, and transitions. States represent the status of the system. Events represent what happens. Transitions move between states. Transitions are triggered by events.

The statechart (Figure 3) is a graphical representation. It shows the four states for the stopwatch: ready/reset, running, paused and stopped. There are five events: start, stop, reset, split and unsplit. The same information is represented using SCXML in Figure 4 (ASF 2006). This is the flow layer of the stopwatch application.

```
<scxml xmlns="http://www.w3.org/2005/07/scxml"
  version="1.0" initialstate="reset">
  <state id="reset">
    <transition event="watch.start" target="running" />
  </state>
  <state id="running">
    <transition event="watch.split" target="paused" />
    <transition event="watch.stop" target="stopped" />
  </state>
  <state id="paused">
    <transition event="watch.unsplit" target="running" />
    <transition event="watch.stop" target="stopped" />
  </state>
  <state id="stopped">
    <transition event="watch.reset" target="reset" />
  </state>
</scxml>
```

Figure 4. Stopwatch example SCXML

4 SCXML and voice interfaces

The stopwatch example application from the Apache Jakarta implementation of SCXML has a graphical user interface (GUI). When the stopwatch is started, the GUI appears as shown in Figure 5 (ASF 2006).



Figure 5. Stopwatch demo in running state

This GUI version of the application displays the time and enables the stopwatch to be started, paused, stopped and reset by mouse clicks. If the user clicks on the Split button with the mouse, the application moves to the paused state shown in Figure 6.



Figure 6. Stopwatch demo in paused state

This graphical user interface is a presentation component, in the presentation layer of the DFP architecture. The stopwatch GUI is provided by Apache Jakarta SCXML (ASF 2006). We have added a voice user interface (VUI) for the stopwatch, using the Sphinx-4 open source Java speech recognizer (CMU 2004) and the FreeTTS open source Java speech synthesizer (Sun 2005).

In a "speaking stopwatch" version of the demo, a brief prompt is spoken when the user starts, unpauses or resets the stopwatch. When it is started (Figure 5), the speech synthesizer says "Running". When the user stops or pauses the stopwatch by a mouse click on the appropriate button, the time is read out aloud by the speech synthesizer. When the demo is paused (Figure 6), it says "Paused at 48 point 3 seconds".

In a "listening stopwatch" version, the user can start, stop, pause, unpause and reset the stopwatch either by voice commands or by mouse clicks. The speech recognizer uses a small JSGF grammar for the voice commands. The stopwatch changes state when an event triggers a transition. An event can be caused either by a mouse click or by speech input. Both graphical output and speech output are part of the stopwatch behaviour on state transitions.

In this version, the presentation layer has three distinct presentation components: the graphical user interface, the speech synthesis interface and the speech recognition interface. There is still only one component in the flow layer: the same SCXML shown in Figure 4 is used unchanged. All three versions of the demo (basic, speaking, listening) follow the same state transitions which are defined in the SCXML file. The use of SCXML supports a clean separation of data, logic and user interface, based on the DFP architecture.

5 Conclusion

The paper showed how SCXML can be used to support the design of voice interfaces. The key idea is to separate the flow layer and the presentation layer. More generally, the same approach applies to the design of multimodal interfaces.

After describing the W3C DFP framework, SCXML was illustrated by a simple stopwatch demo example. The flow layer of the demo was implemented as SCXML. The initial presentation component was a graphical user interface (GUI). The presentation layer was then extended, adding a voice user interface (VUI) with speech input and speech output components. The same SCXML flow layer was reused with no change. This shows the benefits of a clean separation of concerns between the flow layer and presentation layer in the DFP architecture.

References

- ASF (Apache Software Foundation) 2006. Apache Jakarta Project, Commons SCXML. <http://jakarta.apache.org/commons/scxml/>.
- Barnett, Jim; Candell, Emily; Carter, Jerry; Hosn, Rafah; McGlashan, Scott 2006. "Sneak Preview: VoiceXML 3.0". <http://www.w3.org/Voice/2006/voicexml3.pdf>.
- CMU (Carnegie Mellon University) 2004. Sphinx-4: A speech recognizer written entirely in the Java programming language. <http://cmusphinx.sourceforge.net/sphinx4/>.
- Harel, David. 1987. Statecharts: A visual formalism for complex systems. In: Science of Computer Programming 8 (3). 231-274.
- Sun Microsystems 2005. FreeTTS 1.2: A speech synthesizer written entirely in the Java programming language. <http://freetts.sourceforge.net/>.
- W3C 2005. Voice Browser Call Control (CCXML). <http://www.w3.org/TR/2005/WD-ccxml-20050111/>.
- W3C 2006. The Voice Browser DFP Framework. <http://www.w3.org/Voice/2006/DFP>.
- W3C 2007a. State Chart XML (SCXML): State Machine Notation for Control Abstraction. <http://www.w3.org/TR/2007/WD-scxml-20070221/>.
- W3C 2007b. Voice Extensible Markup Language (VoiceXML) 2.1. <http://www.w3.org/TR/voicexml21/>.

GRAHAM WILCOCK is Adjunct Professor of Language Technology at University of Helsinki. He worked in industry for many years with International Computers Limited (UK) and Sharp Corporation (Japan). He has a Ph.D. in computational linguistics from University of Manchester (UMIST). His research interests include machine translation, natural language generation, spoken dialogue systems, HPSG, linguistic ontologies, and XML annotation tools. He has been co-organizer of several international workshops, including 10th European Workshop on Natural Language Generation (2005), Multidimensional Markup in Natural Language Processing (2006), and The Linguistic Annotation Workshop (2007).