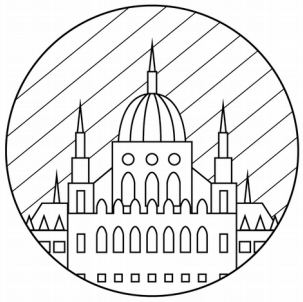
Using Apache Commons SCXML 2.0

a general purpose and standards based state machine engine

Ate Douma
ASF member and committer
R&D Platform Architect at Hippo B.V., Amsterdam
ate@apache.org / a.douma@onehippo.com

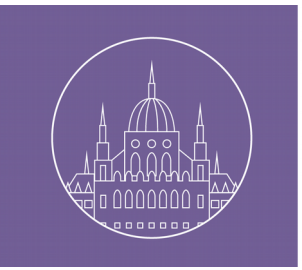


APACHECON EUROPE

CORINTHIA HOTEL
BUDAPEST, HUNGARY
NOVEMBER 17-21, 2014

Outline

- SCXML a quick introduction
- Why SCXML?
- A brief history of Commons SCXML
- Commons SCXML 2.0
  - Why Commons SCXML?
  - High level architecture
  - A simple stopwatch demo
  - Real use-case: Hippo CMS document workflow
  - Status of trunk
  - Roadmap



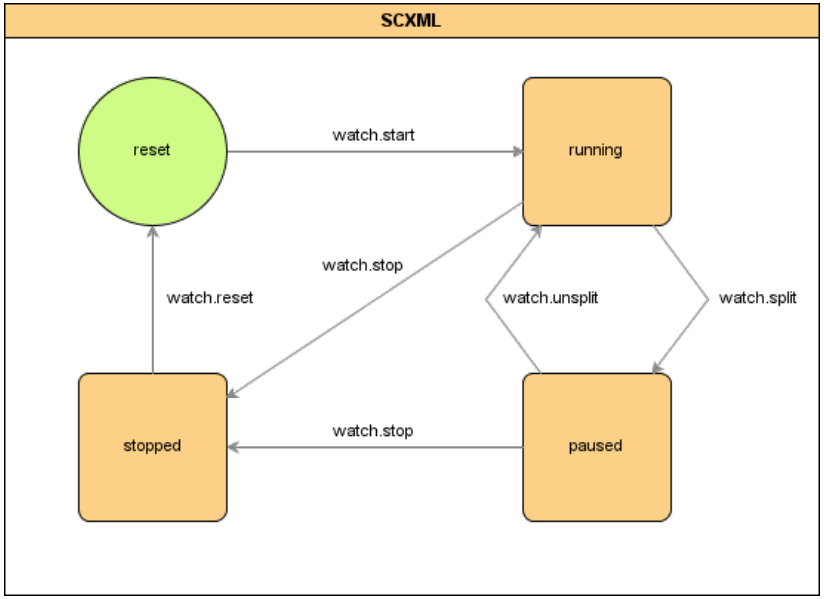
SCXML

State **C**hart **XML**: State Machine Notation for Control Abstraction

- Developed by the W3C: http://www.w3.org/TR/scxml/
- First Draft: July 5, 2005. Now: Last Call Working Draft 29 May 2014
- uses XML format (duh)
- based on CCXML (Call Control XML) and Harel statecharts (UML).
- defines a *generic* state-machine based ***execution environment***
- explicit support for ECMAScript and XPath, and other languages
- defines *both* the XML *and* a normative algorithm for execution

A typical example: a stopwatch

```
<?xml version="1.0" encoding="UTF-8"?>
<scxml xmlns="http://www.w3.org/2005/07/scxml"
       version="1.0" initial="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>
```



A bit more complex: a microwave

```
<datamodel>
  <data id="cook_time"     expr="5"/>
  <data id="door_closed"   expr="true"/>
  <data id="timer"         expr="0"/>
</datamodel>

<parallel id="oven">

  <state id="engine">
    <initial>
      <transition target="off"/>
    </initial>
    <state id="off">
      <transition event="turn.on" target="on"/>
    </state>
    <state id="on">
      <initial>
        <transition target="idle"/>
      </initial>
      <transition event="turn.off" target="off"/>
      <transition cond="timer ge cook_time" target="off"/>
      <state id="idle">
        <transition cond="In('closed')" target="cooking"/>
      </state>
      <state id="cooking">
        <transition cond="In('open')" target="idle"/>
        <transition event="time">
          <assign location="timer" expr="timer + 1"/>
        </transition>
      </state>
    </state>
  </state>

</parallel>
```

```
<state id="door">
  <initial>
     <transition target="closed"/>
  </initial>
  <state id="closed">
     <transition event="door.open" target="open"/>
  </state>
  <state id="open">
     <transition event="door.close" target="closed"/>
   </state>
</state>
```

Why use SCXML?

- Everyone uses some type of state machine implicitly already
- Simple home-brew solutions often mutate into something ugly
- Realization **now** a 'real' solution is needed, typically comes 'too little, too late'

- Use a generalized state machine like SCXML for:
  - improved validation & testing
  - standardized state processing rules
  - simplified usages
  - easier and more controlled extendability and customizations
  - hooking up non-intrusive listeners into your process

- The overhead of a 'real' generalized state machine is mostly neglectable

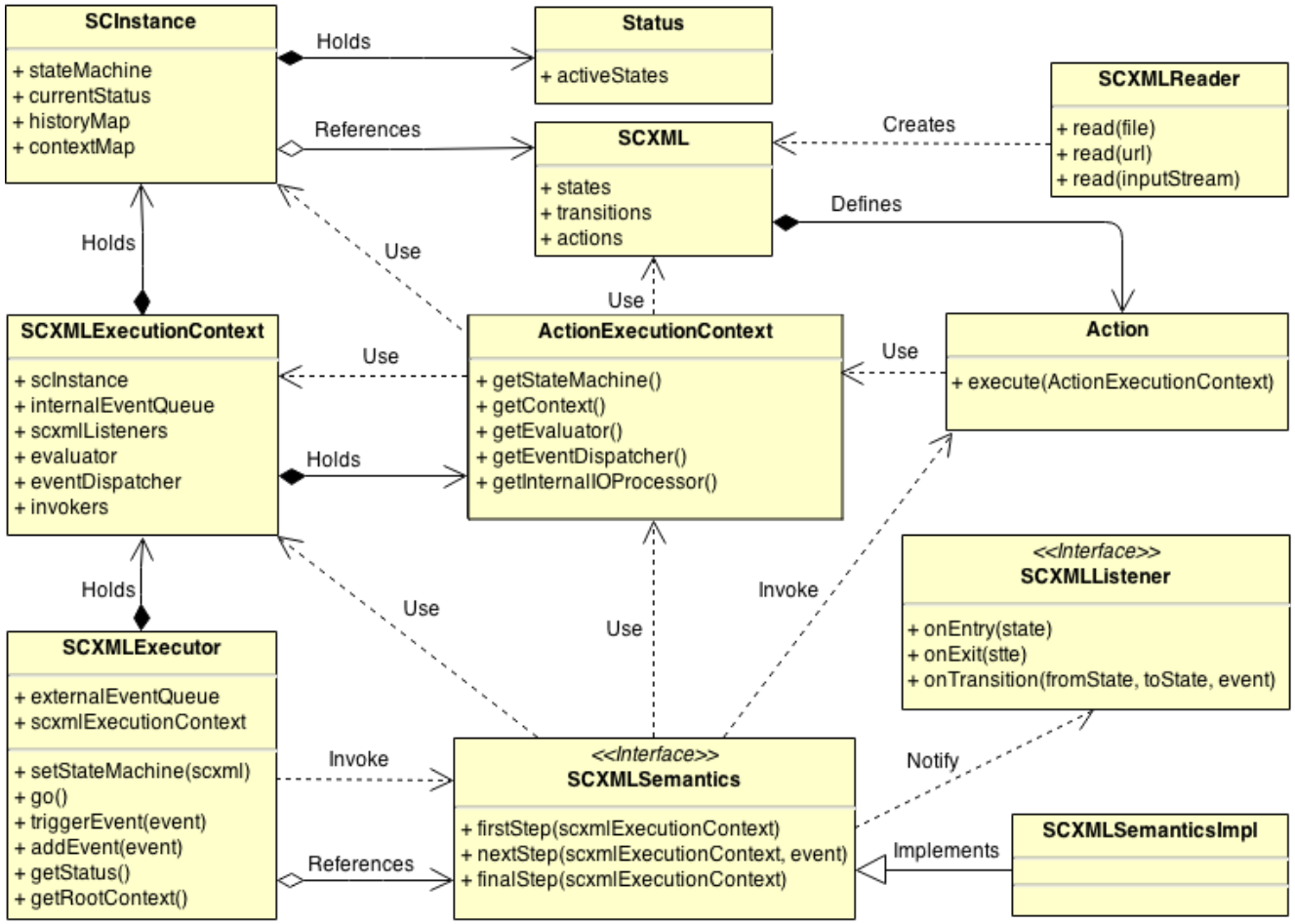
A brief history of Commons SCXML

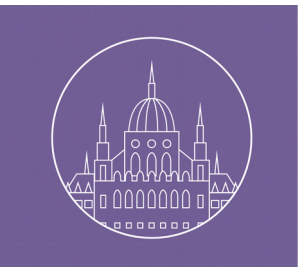
- Started in 2005 ... first release 0.5 in 2007 ... last release 0.9 in 2010 ...
- First and only open source Java implementation of the SCXML specification (other implementations available in Python, C++, Javascript, Lua, etc.)
- Version 0.9 still used a lot, including commercially, for scientific research, etc.
- After 2010 the project stalled while the SCXML specification moved ahead...
- End of 2013 new developers (including me) joined Commons SCXML to revive it
- Specification alignment however was running badly behind
- Catching up requires radical, breaking changes, and is still ongoing
- The next release (sometime 2015) will be Commons SCXML version 2.0

Why Commons SCXML?

- Highly customizable and embeddable engine, clean component model
- Easy to extend and plugin your own custom actions, listeners, etc.
- Can be bootstrapped through code only (no XML needed)
- Bring you own (external) data to drive the state machine
- Runtime state can be serialized and persisted
- Supports custom expression/datamodel languages with an SPI to add your own
  - Apache Commons JEXL & Commons JXPath, Groovy, (still incomplete) Javascript

High level architecture

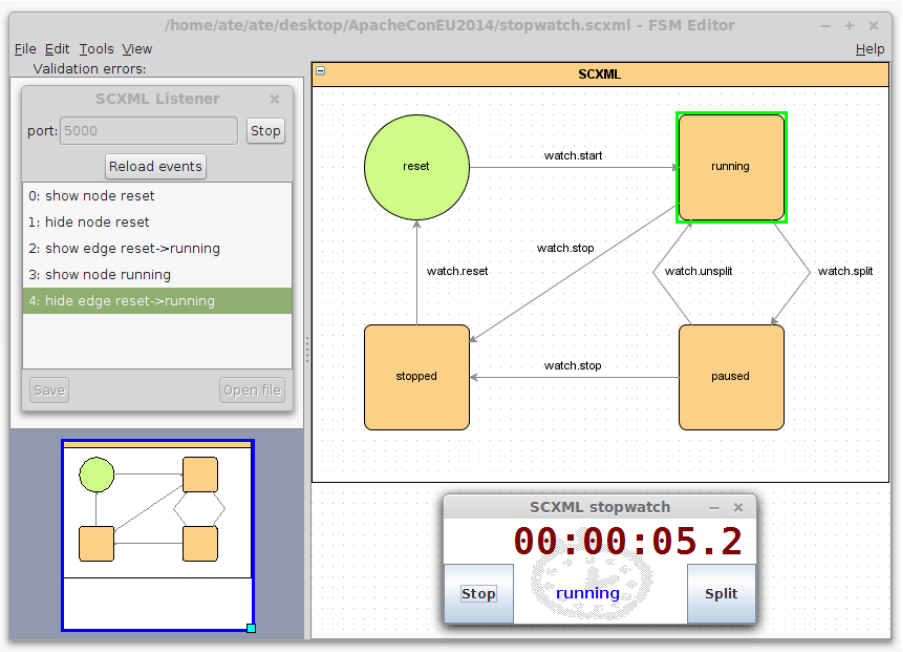




A simple stopwatch demo

Demonstrating a simple stopwatch application:

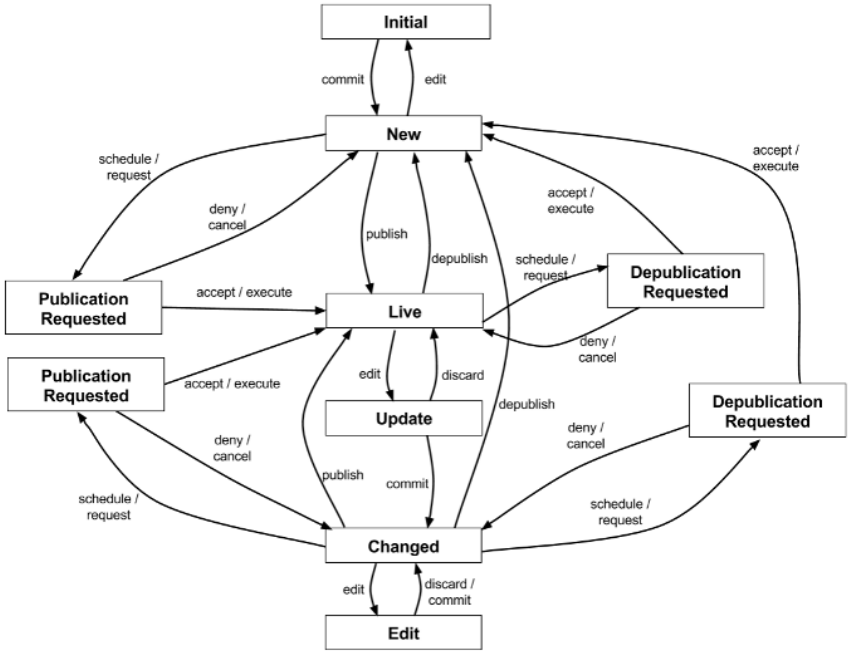
- using an embedded SCXML engine in a Java Swing application
- wired with a custom SCXMLLister through a socket connection to:
- scxmlgui
  A graphical editor for SCXML
  finite state machines (ASL 2.0)
  https://code.google.com/p/scxmlgui



Real use-case: Hippo CMS document workflow

Hippo CMS is an open source Content Management System* using Apache Commons SCXML for its document workflow:

- Used to be 'hand coded' which was rather difficult to extend and customize
- Content and workflow state is stored in a JCR (Apache Jackrabbit based) repository
- Workflow process configuration (SCXML) is now also stored in the repository
- Many workflow processes can be executing concurrently
- In production using Apache Commons SCXML 2.0 Milestone 1



* *http://www.onehippo.org Apache License 2.0*

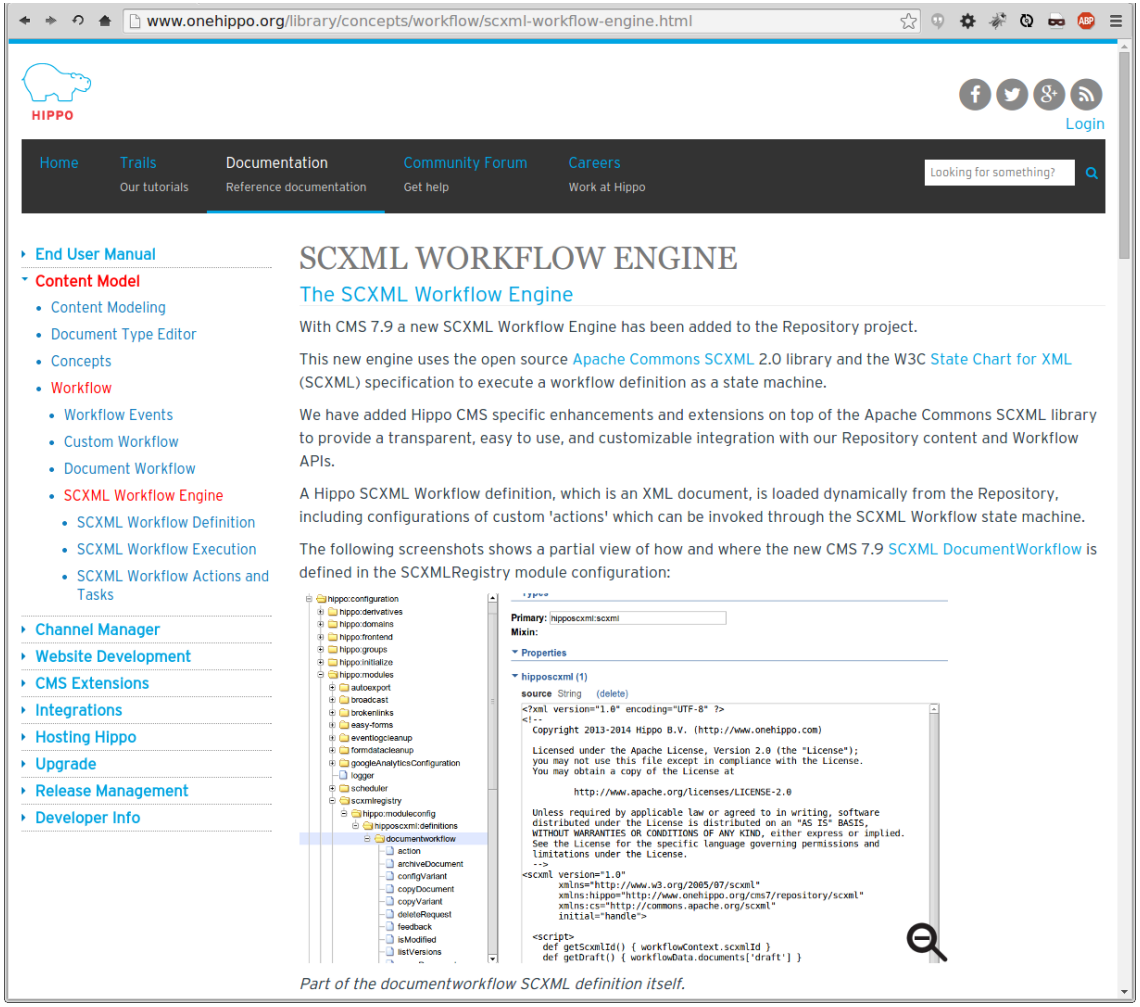
Real use-case: Hippo CMS document workflow

Implementation details:

- http://svn.onehippo.org/repos/hippo/hippo-cms7/repository/trunk/workflow/

  (open source, Apache License 2.0)
- Uses custom SCXMLWorkflowContext and SCXMLWorkflowData objects as 'bridge' between the JCR Repository 'external' context and the SCXML 'internal' context
- All SCXML data is injected dynamically, based on JCR Repository content
- Uses Groovy as expression language
- Custom SCXML Actions implement and execute workflow operation 'commands'
- Workflow definitions can now be customized and extended at runtime
- Online documentation: http://www.onehippo.org/library/concepts/workflow/scxml-workflow-engine.html

Real use-case: Hippo CMS document workflow



Status of trunk

- SCXML processing algorithm rewritten and now aligned with the specification
- Full support for SCXML core model elements and executable content.
- ~ 90%+ completion on data model and data manipulation features
  Full support planned for the upcoming 2.0-m2 test release, soon
- Still running behind on external communications support
  Scheduled for the next (and last) milestone 3 test release
- Committed this week: full rewrite of the XPath language support (big impact)
- Also committed this week: SCXML IRP* tests support
  currently about 35% tests pass (80/231); 2 weeks ago this was still 10%
- Online documentation hopelessly outdated – will get highest priority to fix next

* *SCXML 1.0 Implementation Report Plan: http://www.w3.org/Voice/2013/scxml-irp/*

Roadmap

http://commons.apache.org/proper/commons-scxml/roadmap.html

- Milestone 0: Cleanup (done, 2014-03-11)
  - Dropped support for outdated/broken features
- Milestone 1: SCXML processing Algorithm (done, 2014-04-03)
  - Complete redesign and reimplementation of SCXMLSemantics and architecture
- Milestone 2: Data Model and expression language(s) (~90% done, 2014-11-17)
  - Complete the XPath support, ECMAScript pending, SCXML IRP tests validation
- Milestone 3: External communication
  - Complete support for <invoke> and <send> (data handling almost done)
- Release 2.0 (tentative: 2015) fully compliant with the SCXML specification

That's all folks

Please check out the project!

We are very open to contributions and participation and will welcome any help.

So if you are interested: join the community!

The project: http://commons.apache.org/proper/commons-scxml

The community: http://commons.apache.org/proper/commons-scxml/mail-list.html

The specification: http://www.w3.org/TR/scxml