



XSD as a DSL for Video Generation

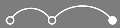
Vit Janota, June 10, 2022

@videate_inc

Motivation

Producing software videos is time-consuming, people-intensive, and cost-prohibitive.

Videate automates the creation of software videos, making it fast and easy to produce up-to-date content any time it is needed.



Challenges

- Producing 1 minute of video takes on average 4 hours of work
 - per Adobe and validated by our polling
- Technical issues
 - mistakes in actions and speech, background noises etc.
- Post processing
 - editing bumper videos, subtitles, effects

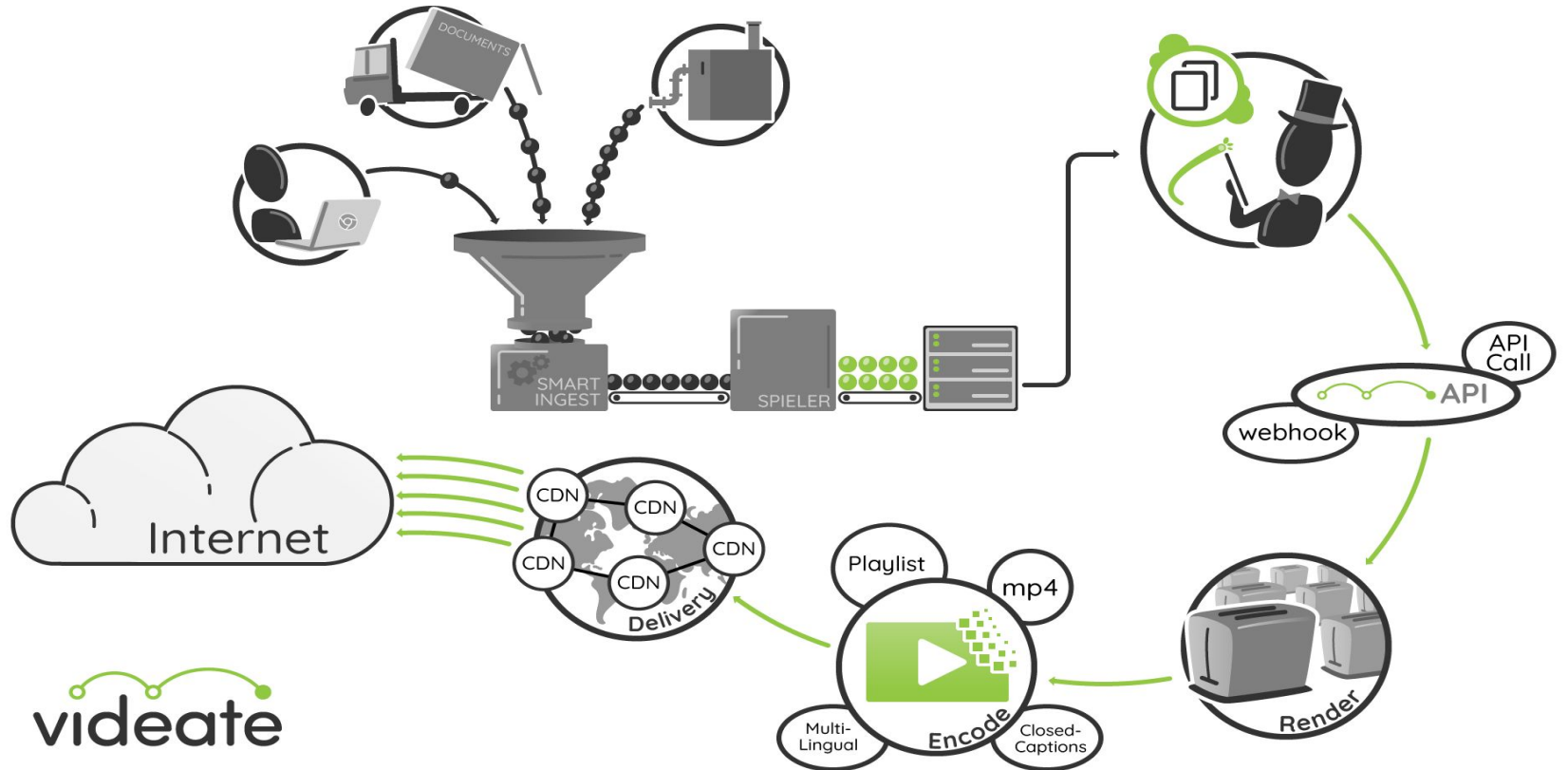


Challenges

- Software updates, changes in environment, changes in wording, UI changes
- Localized versions

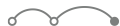


Ideal Workflow



Spiel.xml as a Data Model

- XML (input formats mostly XML, technical precision)
- Follows the linear storyboard/video logic
- Sequence of spoken blocks (paragraphs) and actions to be performed (“say this and do this”)



Spiel.xml as a Data Model

Spiel XML Example

```
<spiel>
  <title>Main menu functionality</title>
  <notify xml:lang="en-US">Main menu functionality</notify>
  <p>
    <ssml:speak xml:lang="en-US" version="1.1">
      As a demonstration of our capabilities we will show an item
      from <highlight selector="Main menu"/> main menu.
    </ssml:speak>
  </p>
  <break time="1000"/>
  <p>
    <ssml:speak xml:lang="en-US" version="1.1">
      For example to see videos generate from
      <ssml:phoneme alphabet="ipa" ph="[pil]">spiel</ssml:phoneme>
      data click <click selector="xpath://span[.='Videos']"/> Videos link.
    </ssml:speak>
  </p>
</spiel>
```



Spiel.xml as a Data Model

- Schema
 - Imports synthesis (SSML) schema, to be able to use the full power of SSML (paragraphs are in fact only wrappers for ssml:speak elements)
 - Extends synthesis schema to allow videate actions within ssml:speak elements




```
<xs:element name="click">
  <xs:annotation>
    <xs:documentation>
      Given the css selector, or xpath:(xpath), clicks the element.
    </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:attribute name="selector" type="xs:string" use="required" />
    <xs:attribute name="modifier" type="xs:string" use="optional" />
    <xs:attribute name="x" type="xs:positiveInteger"/>
    <xs:attribute name="y" type="xs:positiveInteger"/>
  </xs:complexType>
</xs:element>
```



Spiel.xml as a Data Model

- In fact **schema is the central thing**
- No action can be used until it is properly defined in schema (ingest process validates against schema during every document upload)
- Rendering engine functionality is strongly related to schema definitions too



Input Formats

- No document editor is required
- We are not forcing our users to change the editor or format they use
- We work on workflow plugins for Google Docs and Oxygen so that people can “take Videate to where they work”



Input Formats

Dita Example

```
<task>
  <title>Main menu functionality</title>
  <taskbody>
    <steps>
      <step>
        <cmd>As a demonstration of our capabilities
        we will show an item from the main menu.</cmd>
      </step>
      <step>
        <cmd>For example to see videos generated
        from spiel data click Videos link.</cmd>
      </step>
    </steps>
  </taskbody>
</task>
```



Input Formats

- Unobtrusive annotation of existing clients docs (dita, (x)html, proprietary XML) via XML comments, processing-instructions or Word comment
- In other words, minimal document decorations which are “out of band” that don’t wind up in the printed version



Input Formats

Annotated Dita Example

```
<task>
  <title>Main menu functionality</title>
  <taskbody>
    <steps>
      <step>
        <cmd>As a demonstration of our capabilities we will show
          an item <?videate <highlight selector="Main menu"/> ?>
          from the main menu.</cmd>
      </step>
      <?videate <break time="1000"/> ?>
      <step>
        <cmd> For example to see videos generated from spiel data
          <?videate <click selector="xpath://span[.='Videos']"/> ?> click
          Videos link.</cmd>
      </step>
    </steps>
  </taskbody>
</task>
```



Input Formats

- Simplified storyboard format usable within docx, and Google Docs



Input Formats

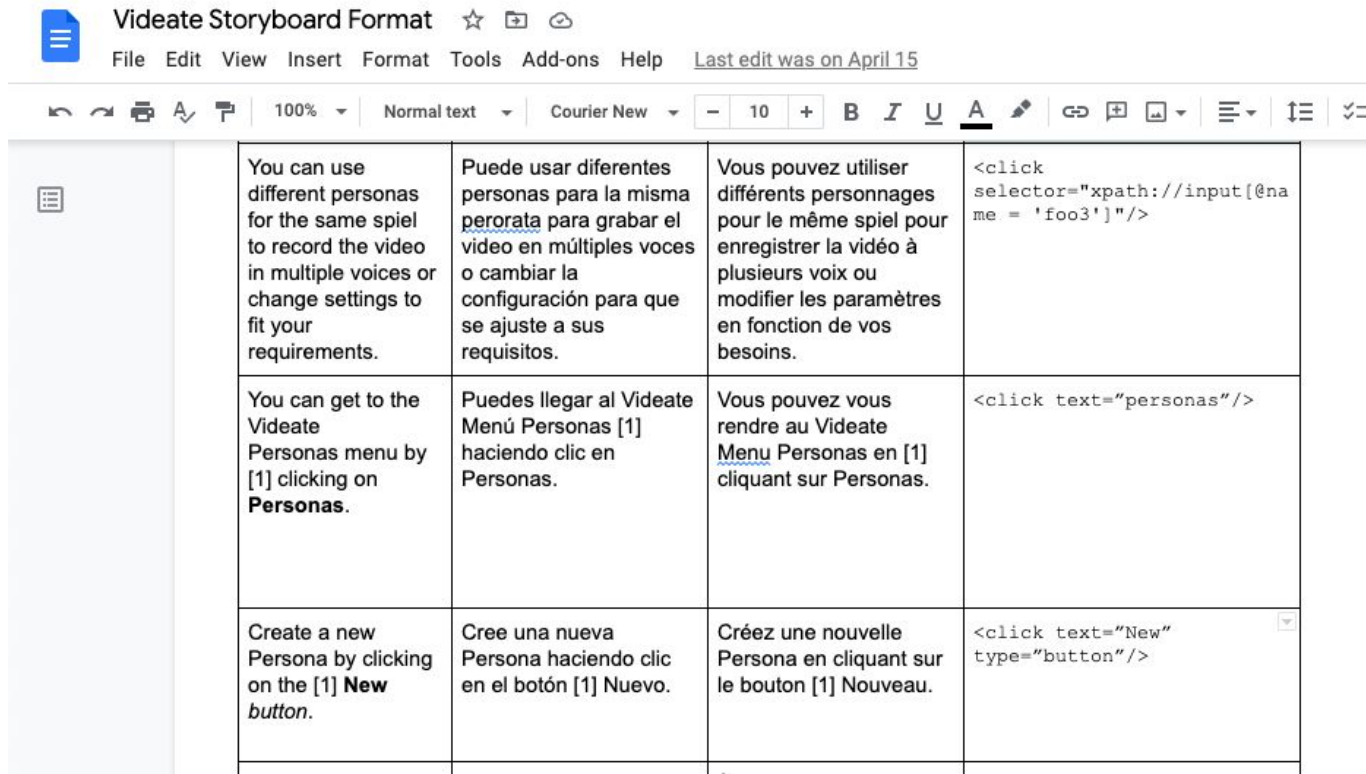
Simplified Storyboard Example

Narrative	Actions
[1]	[1] <title>Main menu functionality</title>
As a demonstration of our capabilities we want to show an item from [1] the main menu. [2]	[1] <highlight selector="Main menu"/> [2] <break time="1000"/>
For example to see videos generate from spiel data click [1] Videos link.	[1] <click selector="xpath://span[.='Videos']"/>



Input Formats

Simplified Storyboard Example (Localized)



The screenshot shows a software interface titled "Videate Storyboard Format". The interface includes a menu bar (File, Edit, View, Insert, Format, Tools, Add-ons, Help) and a toolbar with various editing tools. The main content area displays a storyboard with three rows of localized text and code snippets.

<p>You can use different personas for the same spiel to record the video in multiple voices or change settings to fit your requirements.</p>	<p>Puede usar diferentes personas para la misma <u>perorata</u> para grabar el video en múltiples voces o cambiar la configuración para que se ajuste a sus requisitos.</p>	<p>Vous pouvez utiliser différents personnages pour le même spiel pour enregistrer la vidéo à plusieurs voix ou modifier les paramètres en fonction de vos besoins.</p>	<pre><click selector="xpath://input[@name = 'foo3']"/></pre>
<p>You can get to the Videate Personas menu by [1] clicking on Personas.</p>	<p>Puedes llegar al Videate Menú Personas [1] haciendo clic en Personas.</p>	<p>Vous pouvez vous rendre au Videate <u>Menu</u> Personas en [1] cliquant sur Personas.</p>	<pre><click text="personas"/></pre>
<p>Create a new Persona by clicking on the [1] New button.</p>	<p>Cree una nueva Persona haciendo clic en el botón [1] Nuevo.</p>	<p>Créez une nouvelle Persona en cliquant sur le bouton [1] Nouveau.</p>	<pre><click text="New" type="button"/></pre>



Spiel Interpretation by Rendering Engine

- Browser automation which is similarly related to browser-based testing, but with proprietary features specifically for video production
- Human behavior generation: gestures, typing, scrolling, callouts, interaction, etc...



Spiel Interpretation by Rendering Engine

- Each action element acts like a command in our engine, and when nested in SSML, actions occur concurrently with speaking
- This “command pattern” then means each spiel element definition includes the shape and parameters for function calls by our video engine



Spiel Interpretation by Rendering Engine

- When this plays out in real time, and the commands run as they wish, we get our videos, with clicks, types, callouts, effects, slides, transitions etc...



Spiel Interpretation by Rendering Engine

- The engine manages all of the timeline of speaking and actions intrinsically and there is **no post-production required**, as it is all done in real-time



DEMOS!



VIDEATE
SAMPLE REEL
2022



Thank you