Film Markup Language

Automating Cinemas Using XML

Ari Nordström

Today's Topics

- A Hundred Years of Cinema Technology—the case for automation
- Film Markup Language—rationale and DTD
- FML at work—processing

A Hundred Years Of Cinema Technology

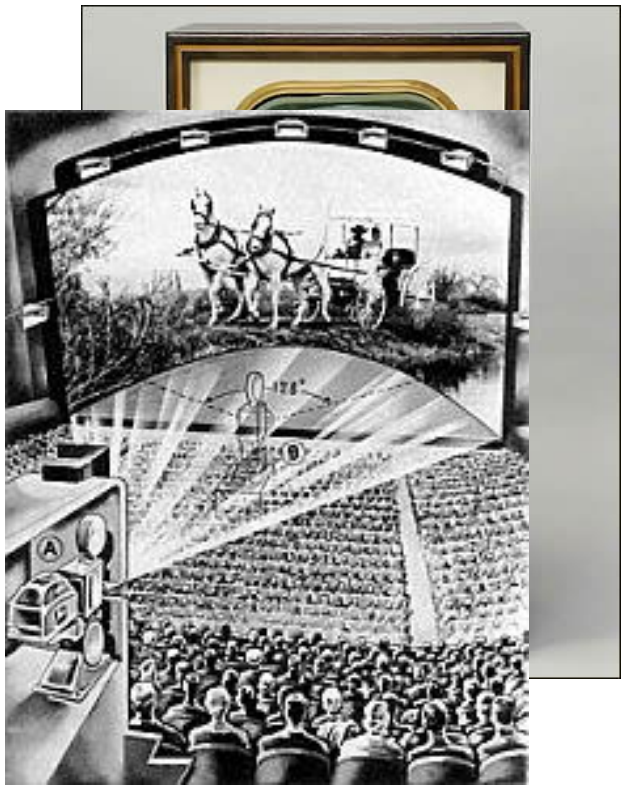
- Until fairly recently, this is how it was done:
  - A film print consists of more than one reel, because of technical limitations (light source)—two projectors were required.
  - The first reel was threaded in one projector and the second in the other.
  - The projectionist did a “change-over” to switch over picture and sound from reel one to reel two.
  - The procedure was then repeated until the end of the show.

A Hundred Years Of Cinema Technology (cont'd)

- Some things have improved...
  - Image technology (Xenon bulbs!)
  - Sound
- But in many ways, these changes are cosmetic:
  - Basic projection technology is still the same (images are projected on a big screen in an auditorium)
  - 35mm film still rules

The War Against Television

- Cinemas were all single-screen but...
  - The 4-5 ushers, 4 candy girls, 5 box-office ladies, and projectionist still had to be paid.
- A number of new technologies were tried:
  - CinemaScope, stereo sound
  - Multi-projector systems such as CINERAMA
  - 70mm presentations and six-channel magnetic sound
- Some were less successful:
  - Smell-O-Vision (no kidding!)

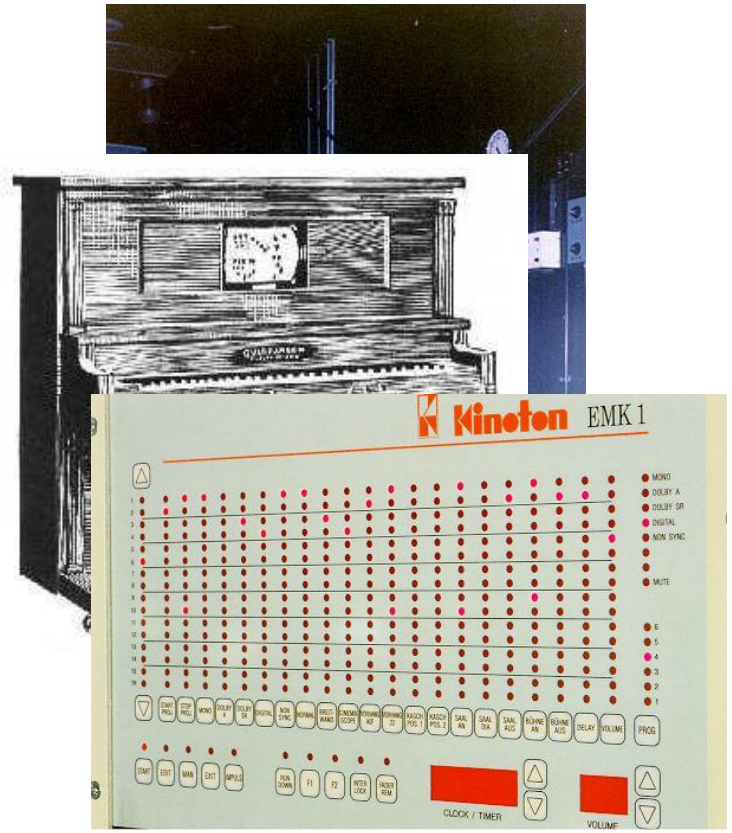


The War Against Television (cont'd)

- But what happened was...
  - Cinemas died in the thousands, all over the world
  - The cinema staff shrank from about 15 people to 4 or fewer
- Multiplex cinemas were born
  - Automating projection booth and auditorium functions was required
  - Larger reels were needed: Enter *the platter*

The Platter and the Mechanical Piano Analogy

- Reels spliced together on a "platter"—no need to rewind
- Event-driven automation (compare with a mechanical piano)
- Actions during a show (dim lights, start projector, change picture format...) are mapped to a pulse- or time code-driven step-by-step function matrix



What Is Automated?

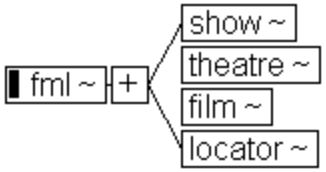
- Actions in the projection booth include...
  - Projector start/stop
  - Igniting Xenon bulb
  - Changing picture aspect ratios (CinemaScope, 16:9, 4:3...)
  - Switching sound formats (mono, optical, stereo, Dolby Digital...)
  - ...
- Actions in the auditorium include...
  - Changing screen aspect ratios
  - Pulling curtains
  - Dimming auditorium lights
  - Triggering light shows before screenings
  - Opening and closing auditorium doors
  - ...

I Sometimes Work At The Draken Cinema...

- I wrote a modest little XML DTD for writing screening instructions.
- But the DTD quicly grew when I realised that it’s actually a structured description for...
  - ...the film's meta-data (title, picture and sound formats, distributor, print number, etc.)
  - ...the auditorium’s meta-data (lights, dim levels, curtains...)
  - ...and the screening as a step-by-step procedure, from the projector start to the final curtain call.

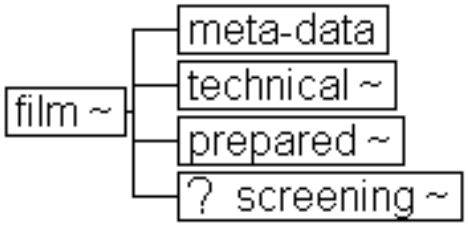
The FML DTD's Three Main Structures

- One `film` structure for each film
- One `theatre` structure for each auditorium
- One `show` structure for each screening with a given auditorium and film
- The `locator` element is for linking



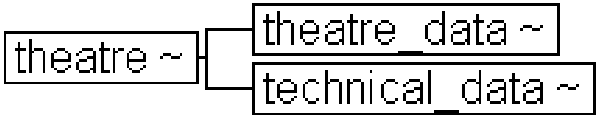
The film Structure

- The `film` structure describes the film—meta-data, sound and picture formats, number of reels, their lengths etc
- Ideally, it should be prepared by the film distributor and accompany every print.
- It's a ***this is what should be done*** statement.



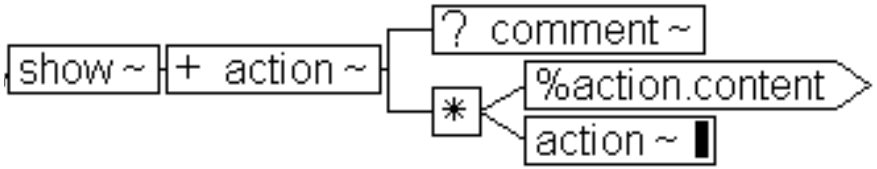
The `theatre` Structure

- The `theatre` structure describes the auditorium, including the projection booth.
- The `theatre` structure contains the technical specifications of the cinema, from picture and sound formats to curtains, auditorium lighting, etc
- It's a ***this is what can be done*** statement.



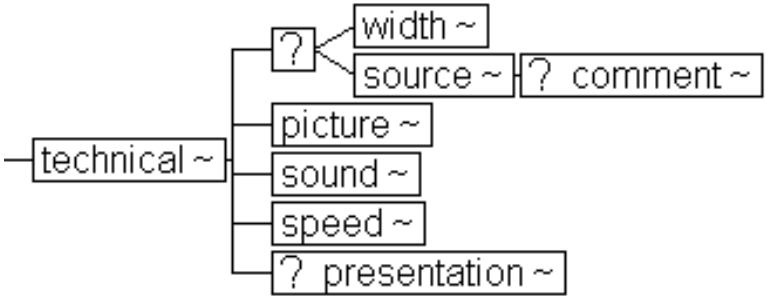
The `show` Structure

- The `show` structure describes a specific screening in a specific auditorium, listing every action required for showing a film.
- It's a ***this is what will be done*** statement.



More on the `film` Structure

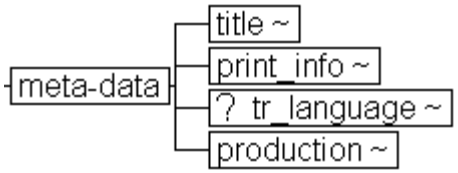
- `technical` contains all necessary technical information about the film:
  - Film widths or source (35, 70, 16, 8, various digital formats...)
  - Picture (aspect ratios, screen requirements)
  - Sound (formats, volume, house equalizer info)
  - Speed (24, 25, 30, 60, custom fps...)
  - Presentation (colour, 3D, black & white...)

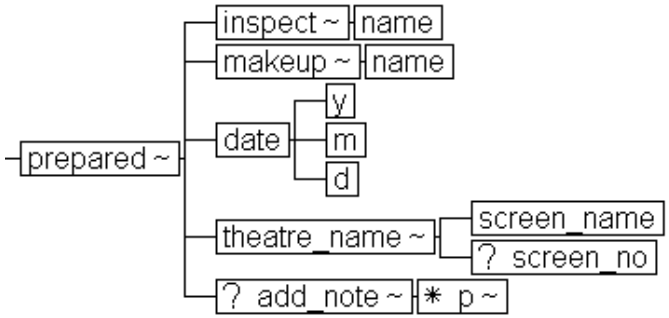


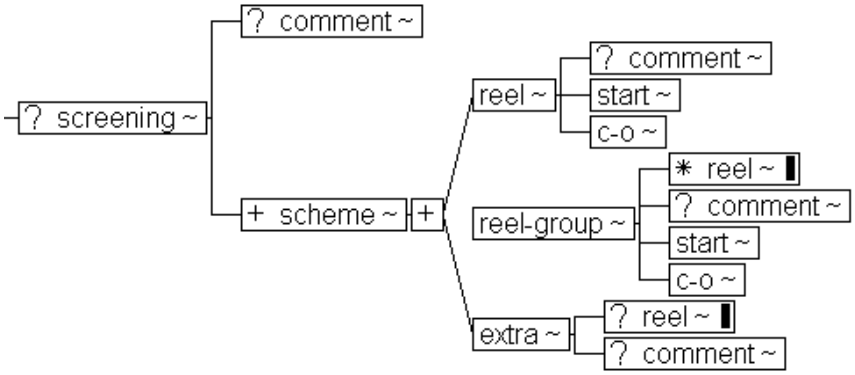

```
</meta-data>
<technical>
    <picture>
        <aspect_ratio value="2.35"/>
    </picture>
    <sound>
        <format format="10"/>
        <vol level="7.0"/>
    </sound>
    <speed fps="24"/>
</technical>
</film>
```

Other `film` Structures

- `meta-data`
  - Non-technical meta-data (the film's title(s), subtitling/dubbing info, production info, etc)
- `prepared`
  - On-site use only
  - Print assembly info
- `screening`
  - Timestamps and offsets
  - Changeover descriptions
  - Reel number info
  - Special events (`extra`)

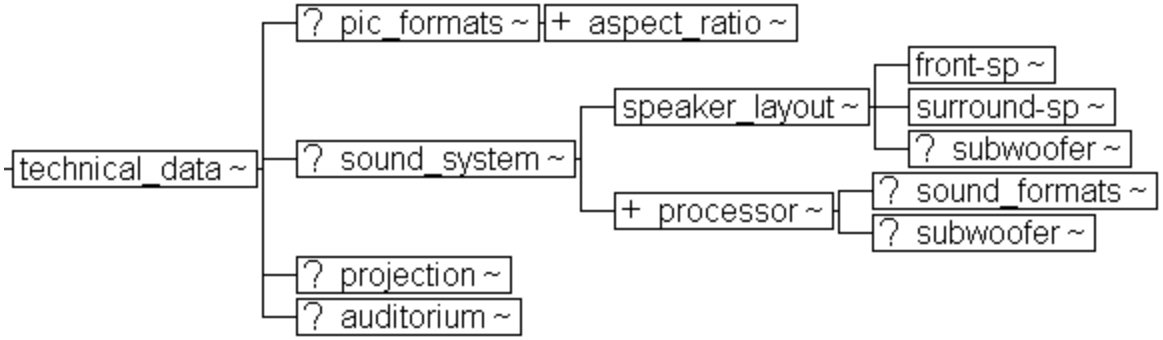






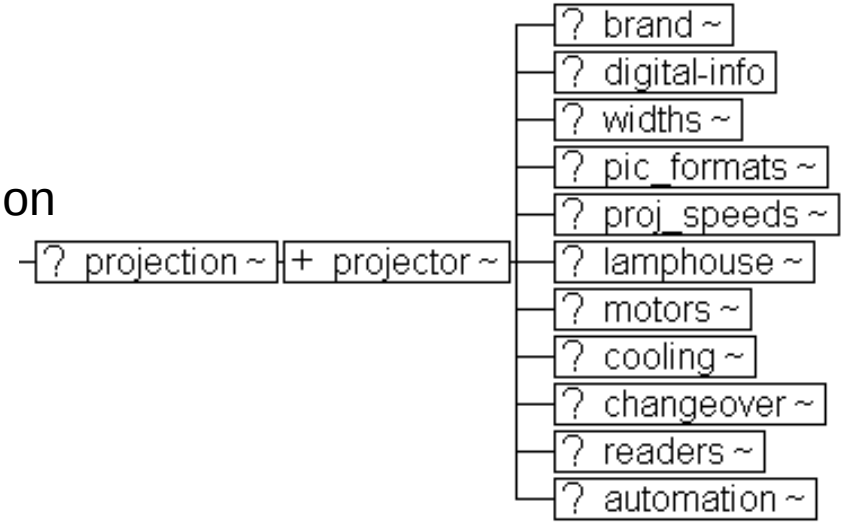
More on the `theatre` Structure

- `theatre` contains all technical data about the *auditorium*
  - Every picture and sound format the auditorium can handle
  - Sound system(s): speakers, sound processors, etc.
  - Every function identified with a function ID



The `projection` Structure (`theatre`)

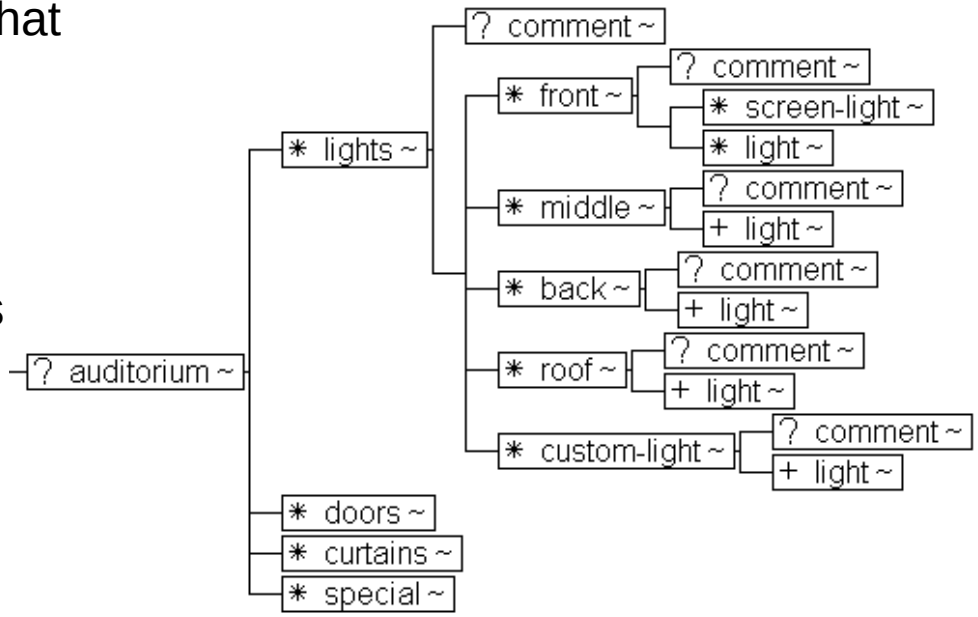
- `projection` contains information about every *projector* that is automated:
  - Film widths
  - Digital formats
  - Picture formats
  - Sound readers
  - Projection speeds
  - Cooling facilities
  - Change-over mechanism(s)
  - Type of automation (sensors, time code feed type, etc)
  - ...




```
</widths>
<pic_formats>
    <aspect_ratio value="1.33" function-id="133-oden"/>
    <aspect_ratio value="1.66" function-id="166-oden"/>
    <aspect_ratio value="1.75" function-id="175-oden-large"/>
    <!-- 1.66 lens -->
    <aspect_ratio value="1.85" function-id="185-oden-small"/>
    <aspect_ratio value="1.85" function-id="185-oden-large"/>
    <aspect_ratio value="2.21" function-id="todd-ao-oden"/>
    <aspect_ratio value="2.35" function-id="cs-oden"/>
</pic_formats>
<proj_speeds>
    <speed fps="18" function-id="18fps-oden"/>
    <speed fps="24" function-id="24fps-oden"/>
    <speed fps="24.7" function-id="247fps-oden"/>
    <speed fps="30" function-id="30fps-oden"/>
</proj_speeds>
<lamphouse>
    <brand>
```

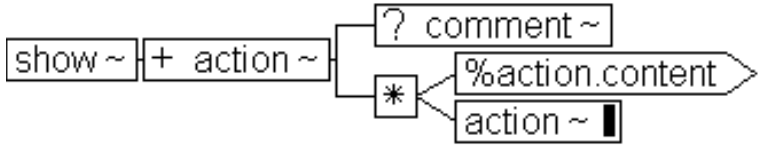
The `auditorium` Structure (`theatre`)

- `auditorium` identifies functions in the auditorium that need to be controlled, from lighting to doors.
- All `light` structures use attributes for dimming levels and on/off functions.



More on the `show` Structure

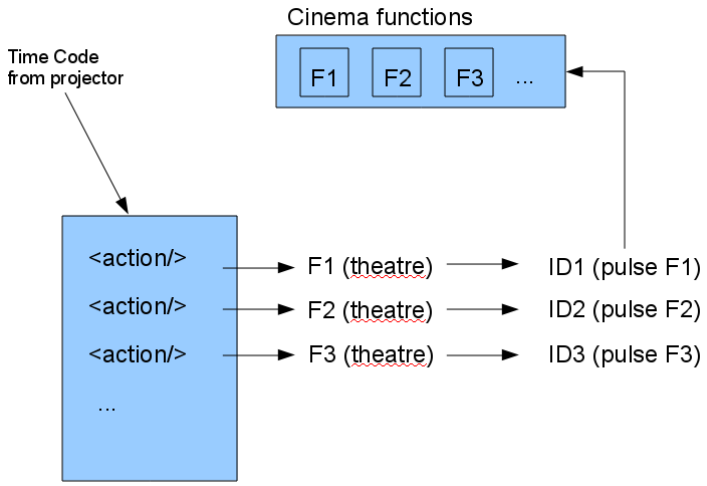
- A show is a sequence of *actions* (remember the mechanical piano?)
- `action` structures can be nested in the DTD
- `action` elements use instructions from `theatre` (but based on `film` contents)
- Or, `technical_data` structures (in `theatre`) may be used for overrides
- Timing by attributes (timestamp or offset) or by pulses from projector




```
<show>
    <action xmlns:xlink="http://www.w3.org/1999/xlink" xlink:typ
        <comment><p>Start projector</p></comment>
        <action function-id="proj-l" timestamp="0"><!-- Power on
            <projector use="yes" function-id="proj-l"/>
        </action>
        <action function-id="proj-l-motor" timestamp="0"><!-- St
            <motor function-id="proj-l-motor" power="on"/>
        </action>
        <action function-id="proj-l-lamphouse" timestamp="24"><!
            <lamphouse function-id="proj-l-lamphouse" power="on"
        </action>
        <action function-id="proj-l-rect" timestamp="72"><!-- St
            <rectifier function-id="proj-l-rect" power="on"/>
        </action>
        <action function-id="rect-l-bulb" timestamp="112"><!-- S
            <bulb function-id="rect-l-bulb"/>
        </action>
    </action>
    <action event-type="picture">
        <comment><p>Change screen/projector aspect ratio</p></co
        <action function-id="custom" timestamp="custom"> <!-- Sc
            <aspect_ratio value="custom" function-id="pic-format
        </action>
        <action function-id="proj-l-format-cs" timestamp="custom
            <aspect_ratio value="custom" function-id="proj-l-for
```

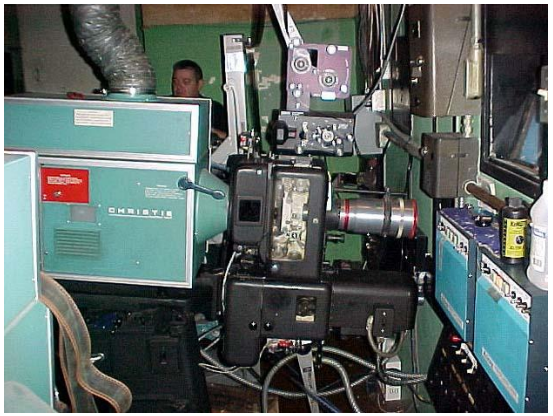
FML Implementation

- A tachometer feeds time code ("heartbeats", fps, etc) from the projector to the FML processor
  - An event-based system (aluminum tape on the film strip) can also be used
- When a time code value matches an action's timestamp, the instructions contained in that action trigger pulses that are sent to an automation box



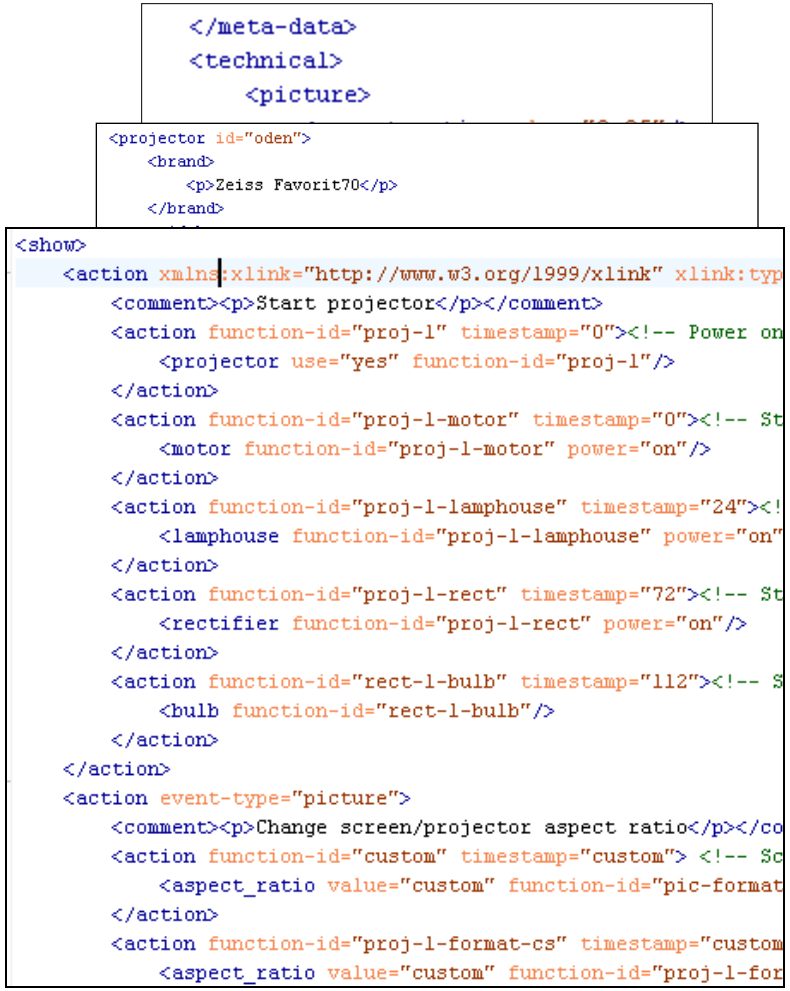
Modifying A Cinema For FML

- Most modern cinemas already have the necessary electronics in place to control the booth and the auditorium.
  - Adding an FML processor and a central automation box is easy.
- Older cinemas might require more significant changes.



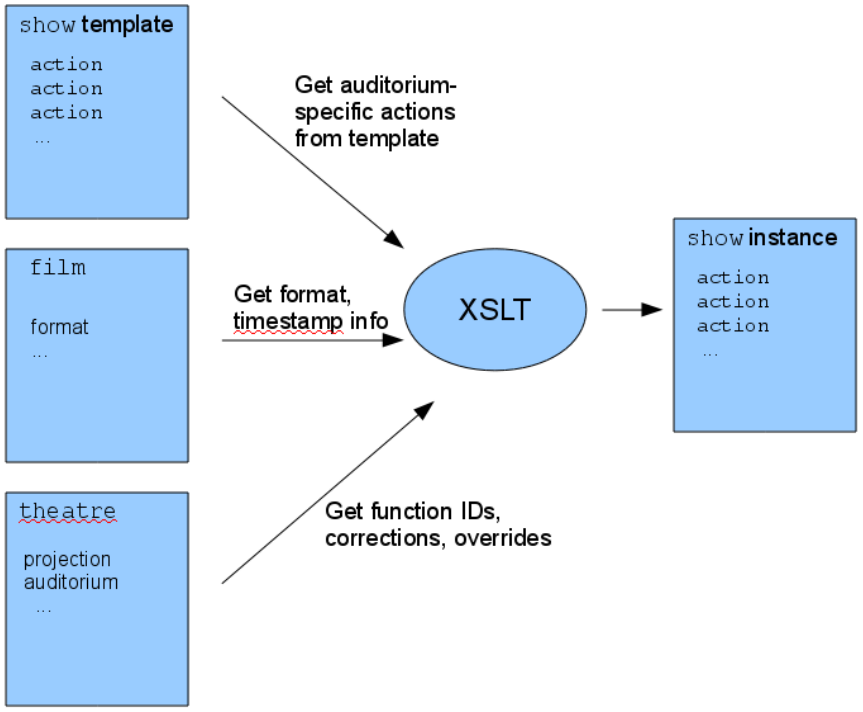
Required FML Instances

- A `film` instance describing the film
- A `theatre` instance describing the auditorium, including all function states
- A `show` instance template describing a basic show in the auditorium
  - No timestamps, just fixed offsets—it's a template!



Producing A Show

- The actual `show` instance (”this is what will be done”) results from processing the `film` and `theatre` instances with the relevant `show` template(s)
- This (most of the time) is all you need



Reprogramming Shows

- Automation systems today require reprogramming when moving films from larger to smaller auditoriums.
- FML eliminates this problem:
  - `show` templates and `theatre` instances already exist for each auditorium.
  - A new `show` instance for the smaller auditorium is just one XSLT conversion away.

```
<action event-type="curtain">
    <comment><p>Close curtain from CS position (28s),
        light up screen lighting to 100% (15s),
        light up middle and roof to 100%</p></comment>
    <action function-id="curtain-close-cs"
            timestamp="136800">
        <curtain closerange="28"
                 function-id="curtain-close-cs"/>
    </action>
    <action event-type="light"
            function-id="screen-dim-up-100"
            timestamp="136848">
        <screen-light uprange="15"
                      function-id="screen-dim-up-100"/>
    </action>
    <action function-id="roof-dim-up-100"
            timestamp="136872">
        <roof function-id="roof-dim-up-100">
            <light level="100" uprange="30"/>
        </roof>
    </action>
    <action function-id="middle-dim-up-100"
            timestamp="136872">
        <middle function-id="middle-dim-up-100">
            <light level="100" uprange="30"/>
        </middle>
    </action>
</action>
```

Error Handling

- The time code from the projector is recorded, telling us *when* the problem occurred.
- If frames are lost, their number and location give us a revised `show` instance (with a simple XSLT conversion).
- If *SMPTE or DTS timecode* is used, that code can automatically provide the offsets resulting from the missing frames.

```
<action function-id="curtain-close-cs"
        timestamp="136800">
    <curtain closerange="28"
        function-id="curtain-close-cs"/>
</action>
```

42 frames lost

```
<action function-id="curtain-close-cs"
        timestamp="136758">
    <curtain closerange="28"
        function-id="curtain-close-cs"/>
</action>
```

Why Use FML?

- Film distributors can accurately describe how their film should be screened.
- FML can completely standardise the programming of a show.
- FML can greatly simplify the task of reprogramming when moving a film to a smaller auditorium.
- FML will, in the long run, be far cheaper than any conventional type of automation.

Questions?