declarative techniques in Distributed Media Center

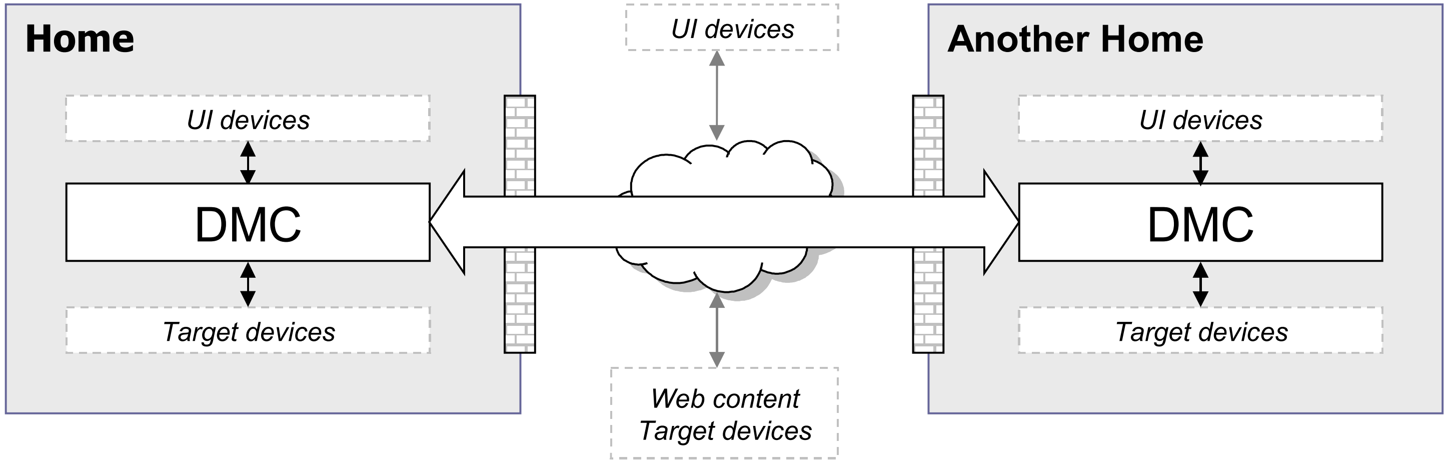
Jari Kleimola and Petri Vuorimaa

Telecommunications Software and Multimedia Lab
Helsinki University of Technology

June 5, 2007

Outline

Application Scenario



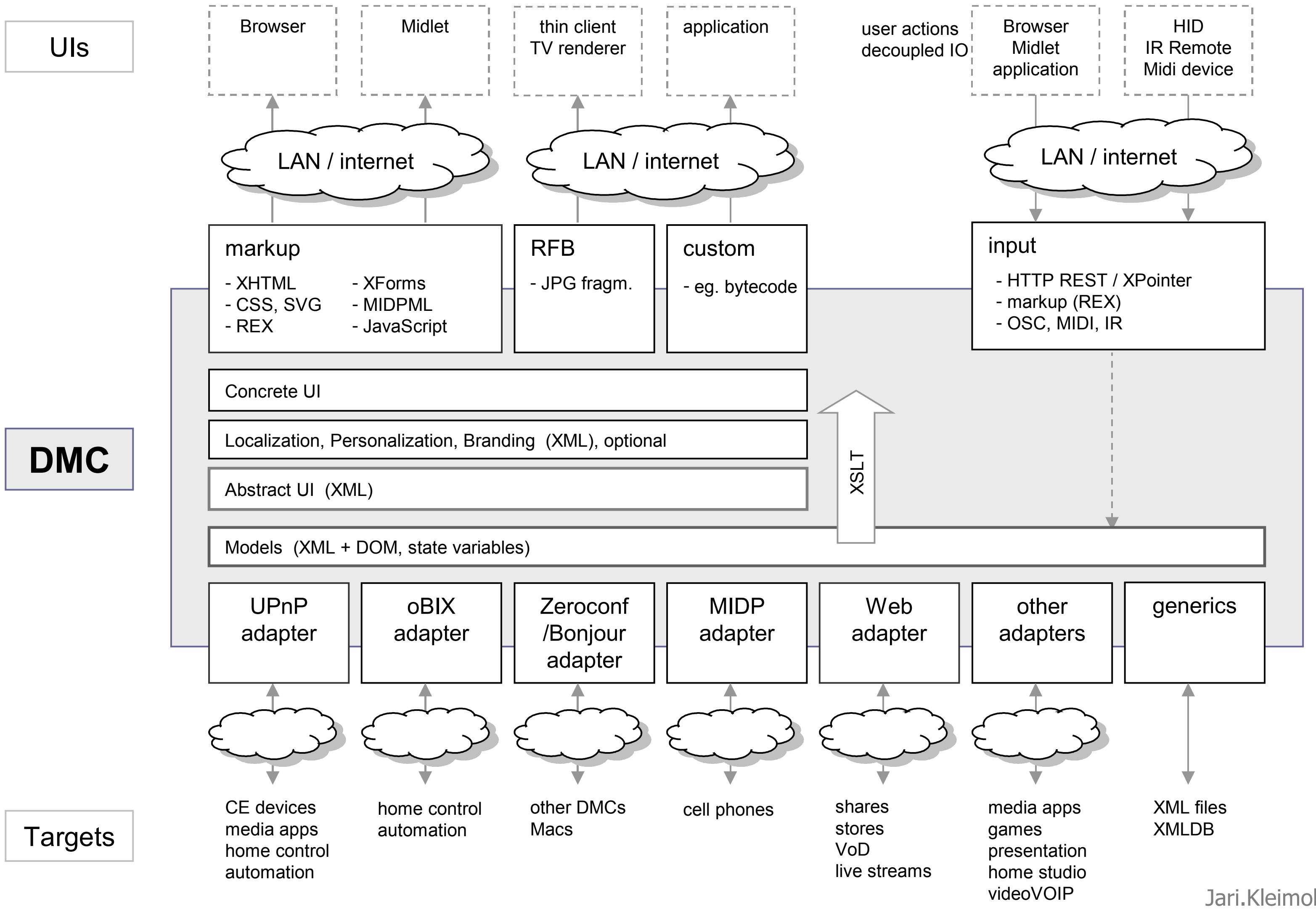
Problem: Unified control and monitoring of networked

- Targets
  - media devices + apps
  - media services + content
  - home automation
- UI devices
  - PCs + laptops
  - phones, PDAs
  - TVs, legacy remotes

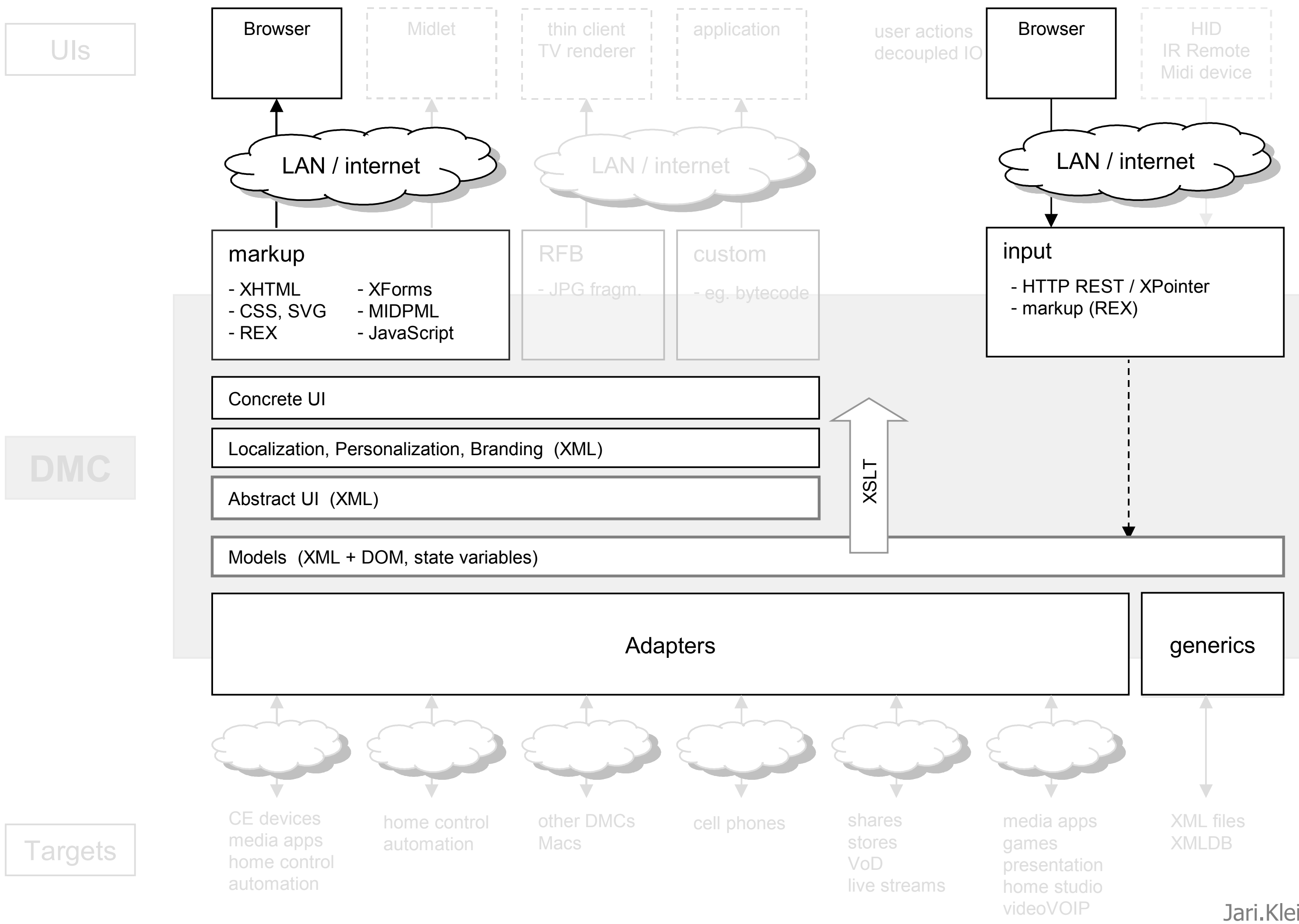
Possible solution:

- DMC = Distributed Media Center
  - remote control hub
  - runs in a PC / STB / gateway

DMC Architecture



DMC Architecture



5.6.2007
Jari.Kleimola@tml.hut.fi

Compound Model

Distributed into

- Dynamic infrastructure DOM (zones, devices, services, people, content)
  - updated by adapters and components
  - only topmost levels
- Virtual models that extend infrastructure DOM
  - handled by adapters
  - state variables
- Generic XML files hosted by DMC
  - eg. for RAD + testing

Addressing

- Dispatcher parses URI to model and resource reference (http://host:port/model#resource)
  - eg. /dmc/$\underbrace{\text{zones\#bedroom/TV}}_{\text{infra}}$/$\underbrace{\text{channel}}_{\text{adapter}}$

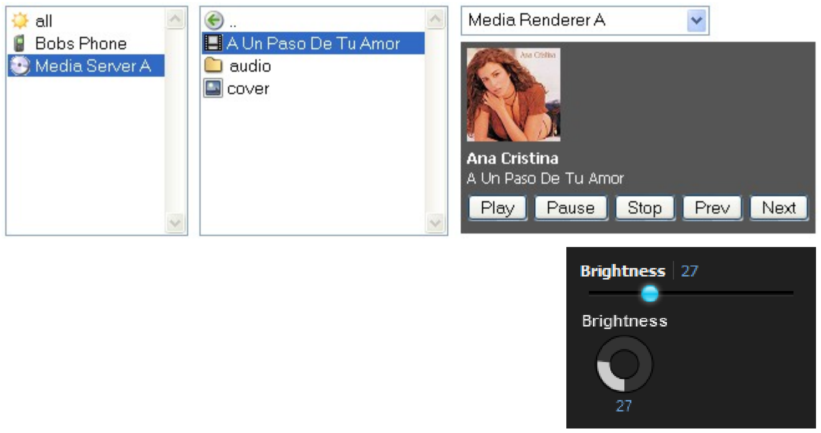
Mutation and Access

- REST and/or REX
  - ui: GET + PUT, adapter: POST + DELETE + PUT
  - POST /dmc/devices body: <device name=”TV” id=”udn”>...</device>
  - PUT /dmc/zones#bedroom/tv/channel=5

Remote UIs

UI Fragments

- extend XHTML controls (currently scripted)
  - eg. bound list, slider, popup
  - patterns, microformats / RDF description ?
- can be updated without loading entire page
  - XHR
  - binding to other controls
- bound into micromodel using XPath
  - eg. bind="/dmps/device[@id=$current]"



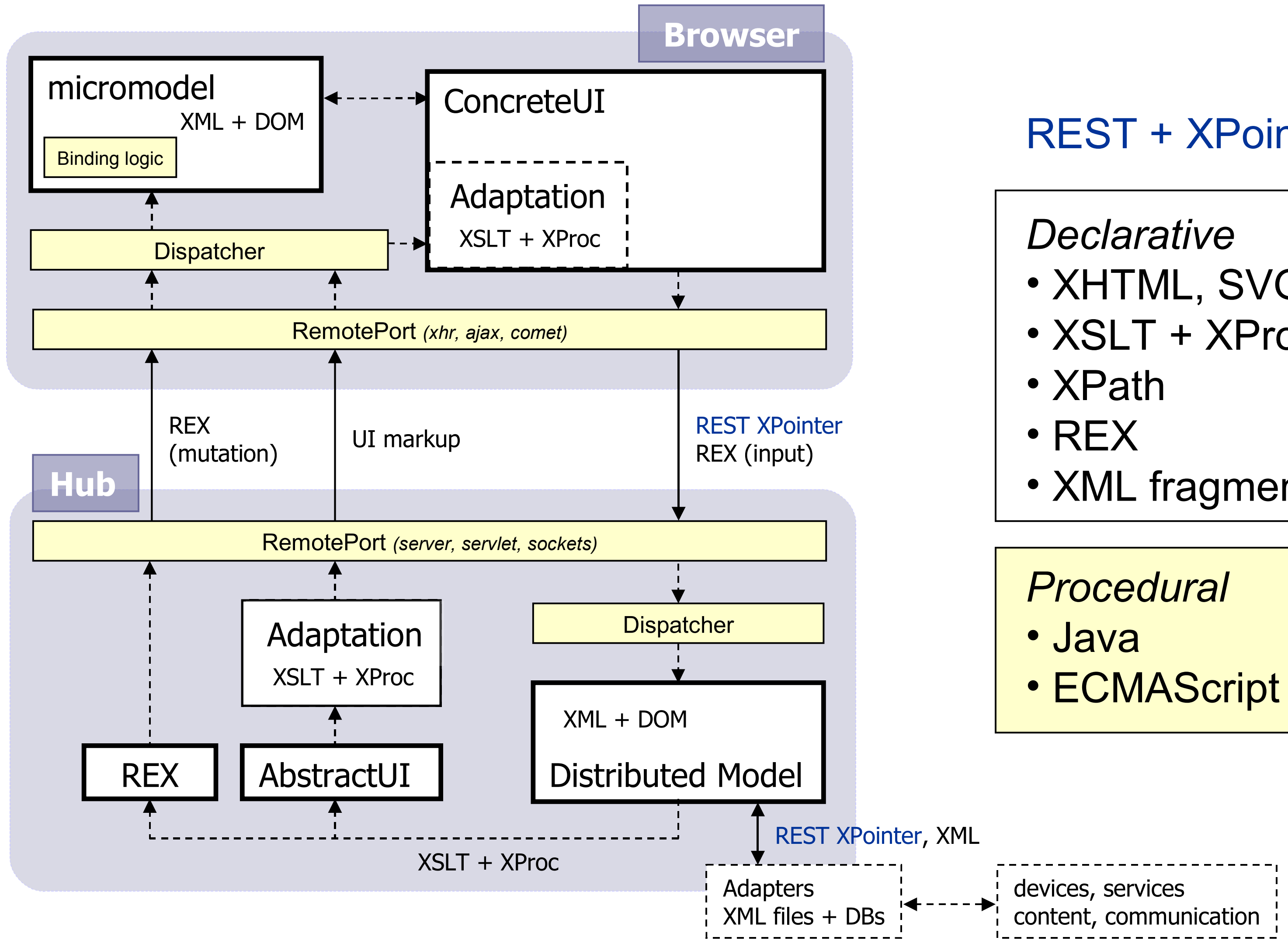
micromodel (uDOM compliant)

- state, maybe some frequently used items as well
- mutated using REX

Other things

- On-the-fly interaction style change
  - eg. forms -> direct manipulation
- IO decoupling
  - attach display from mobile phone to TV
  - continue using mobile keys as input device

Interaction Loop

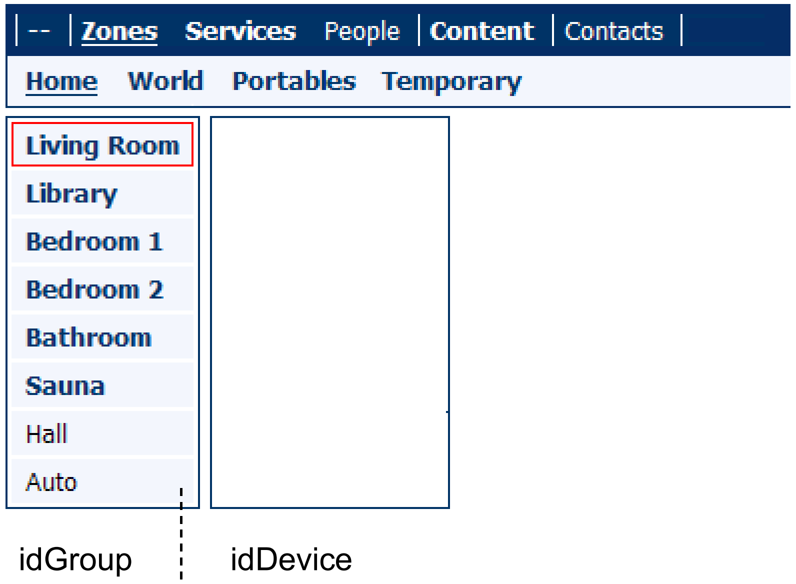


REST + XPointer

Declarative
- XHTML, SVG
- XSLT + XProc
- XPath
- REX
- XML fragments

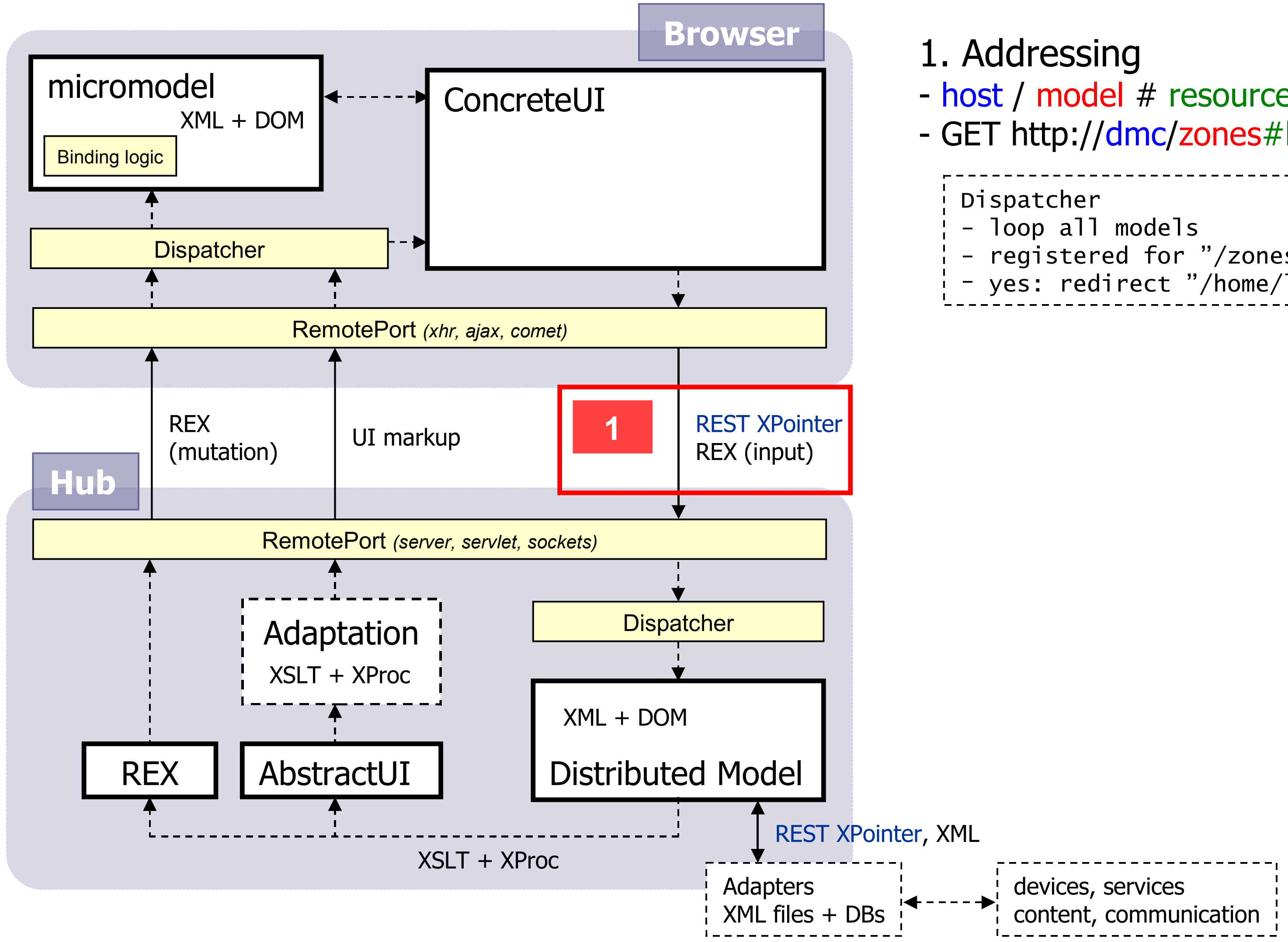
Procedural
- Java
- ECMAScript

Sample Loop


-- | Zones Services People | Content | Contacts
Home World Portables Temporary
Living Room
Library
Bedroom 1
Bedroom 2
Bathroom
Sauna
Hall
Auto
idGroup
idDevice

```
<ul id="idGroup">
  <li href="/zones/home/livingroom" onclick="httpGET(this)" xhrTarget="idDevice">Living Room</li>
  <li href="/zones/home/library" onclick="httpGET(this)" xhrTarget="idDevice">Library</li>
  ...
</ul>
```

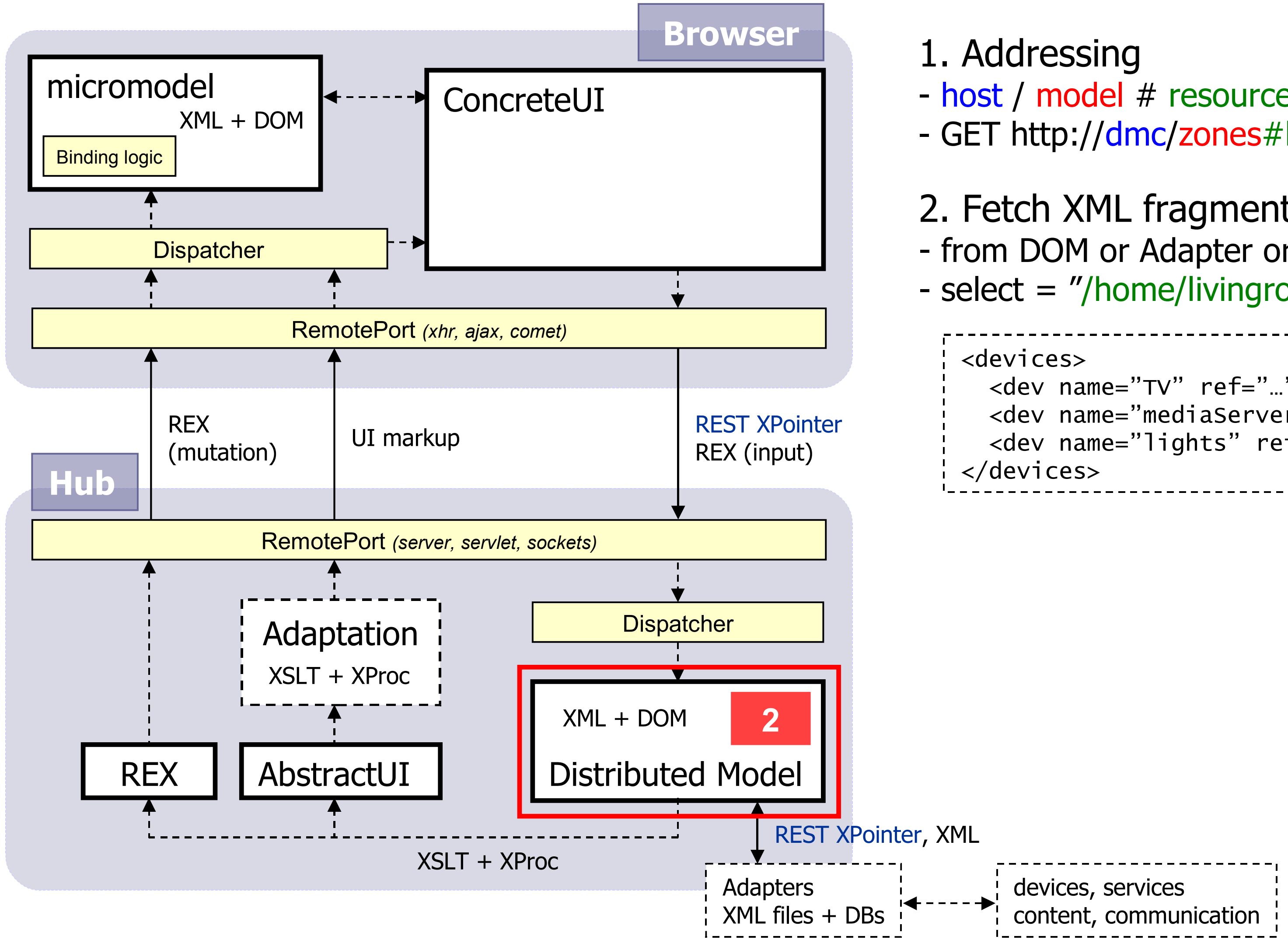
Sample Loop



1. Addressing
- host / model # resource
- GET http://dmc/zones#home/livingroom/

```
Dispatcher
- loop all models
- registered for "/zones" ?
- yes: redirect "/home/livingroom/"
```

Sample Loop

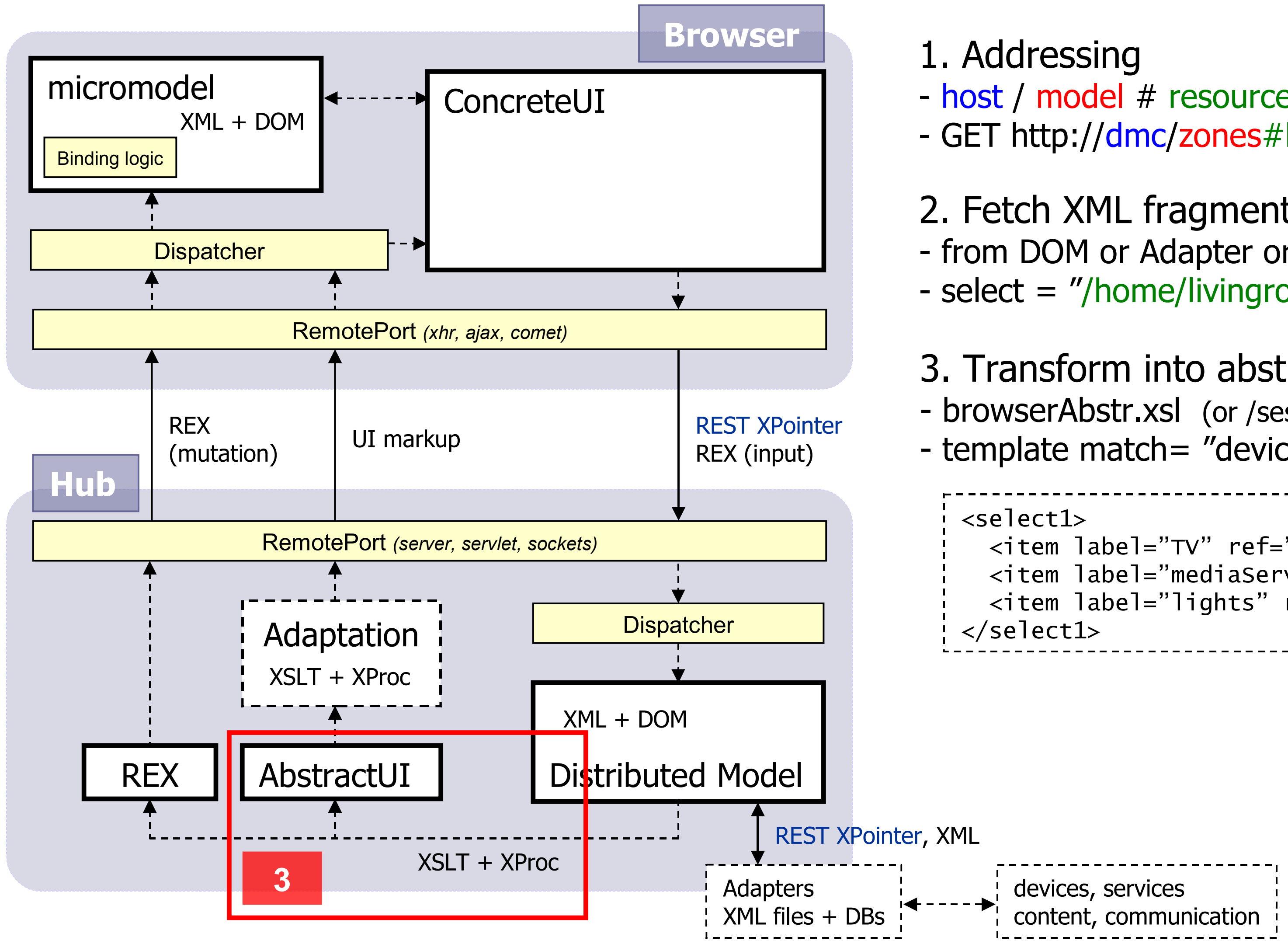


1. Addressing
- host / model # resource
- GET http://dmc/zones#home/livingroom/

2. Fetch XML fragment
- from DOM or Adapter or file
- select = "/home/livingroom/"

```
<devices>
  <dev name="TV" ref="…"/>
  <dev name="mediaServer" ref="…"/>
  <dev name="lights" ref="…"/>
</devices>
```

Sample Loop



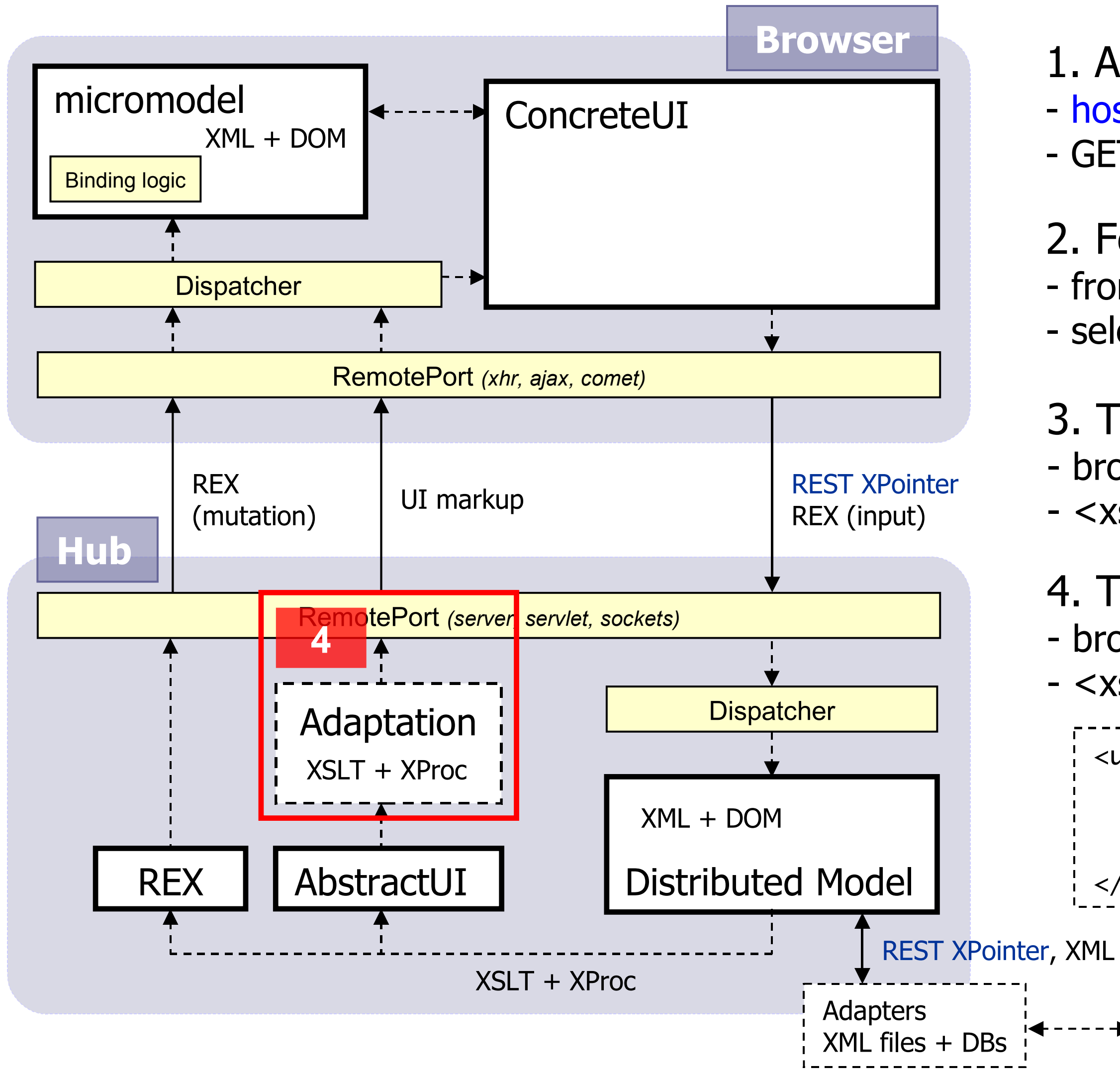
1. Addressing
- host / model # resource
- GET http://dmc/zones#home/livingroom/

2. Fetch XML fragment
- from DOM or Adapter or file
- select = "/home/livingroom/"

3. Transform into abstract UI
- browserAbstr.xsl (or /session[@id]/abstract)
- template match= "devices"

```
<select1>
  <item label="TV" ref="…" … />
  <item label="mediaServer" ref="…" … />
  <item label="lights" ref="…" … />
</select1>
```

Sample Loop



1. Addressing
- host / model # resource
- GET http://dmc/zones#home/livingroom/

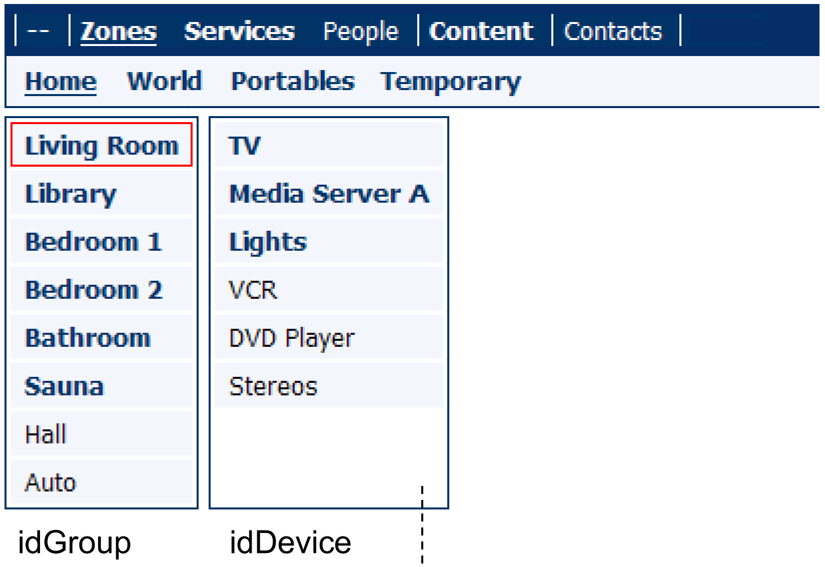
2. Fetch XML fragment
- from DOM or Adapter or file
- select = "/home/livingroom/"

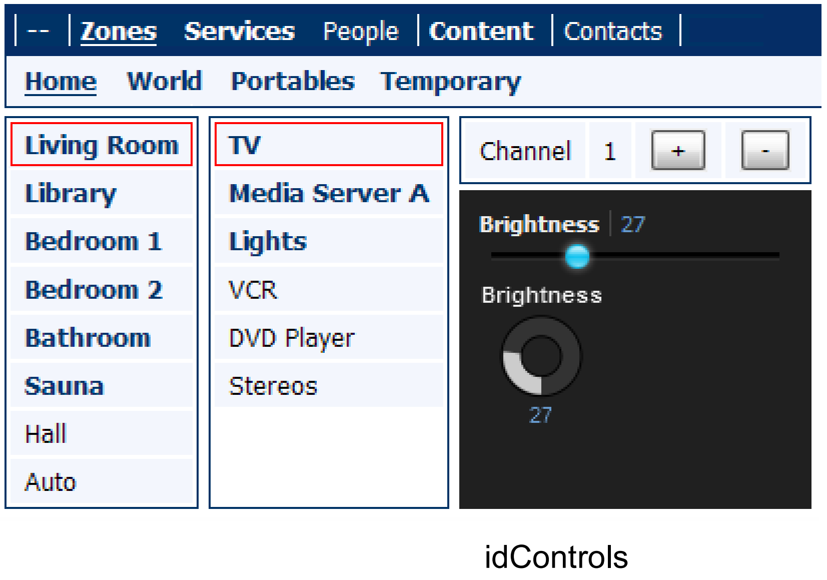
3. Transform into abstract UI
- browserAbstr.xsl (or /session[@id]/abstract)
- <xsl:template match="devices"/>

4. Transform into concrete UI
- browserConcr.xsl (or /session[@id]/concrete)
- <xsl:template match="select1"/>

```
<ul>
  <li href="…" … >TV</li>
  <li href="…" … >Media Server</li>
  <li href="…" … >Lights</li>
</ul>
```

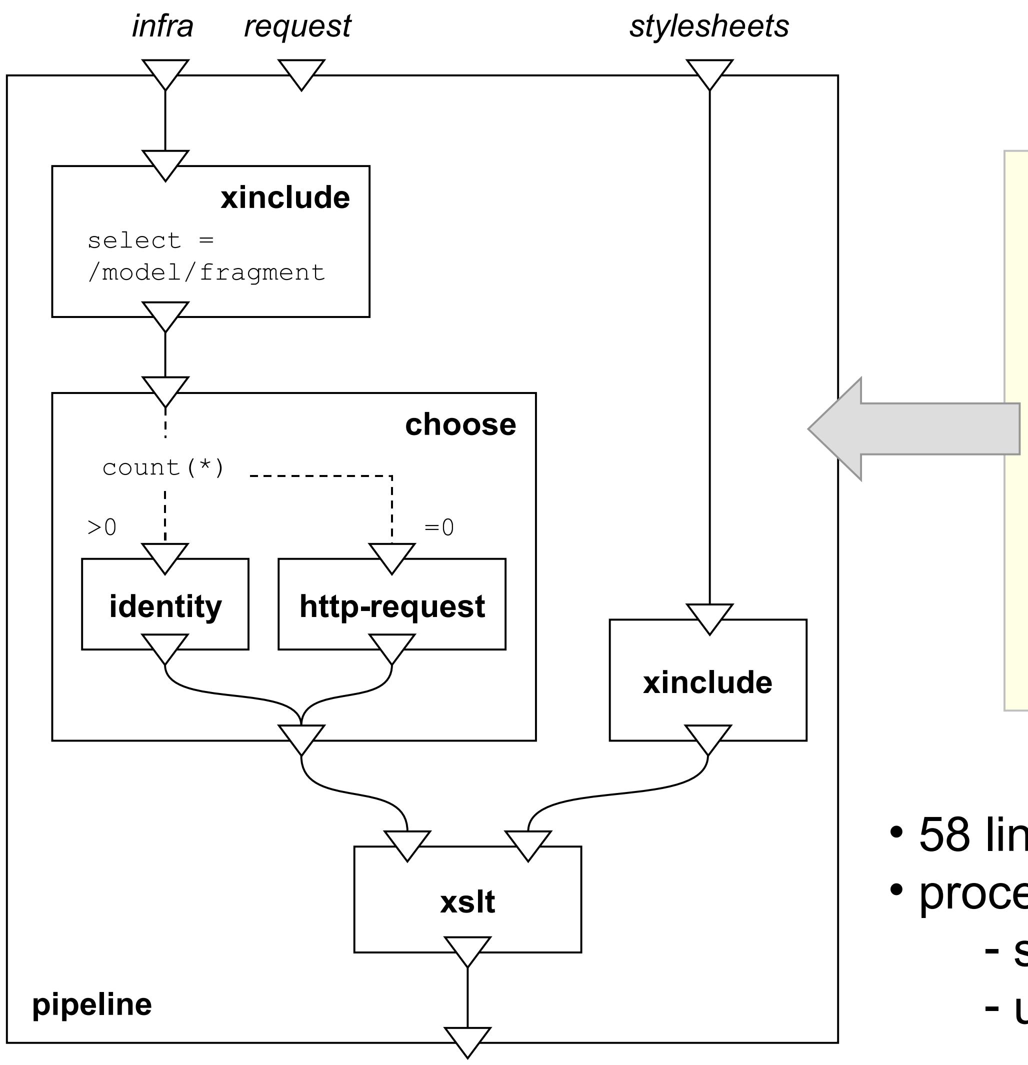
Result





```
<rex>
  <event target=”idDevice” name=”DOMNodeRemoved”>
    <ul id=”idDevice”>
      <li href=”…” onclick=”httpGET(this)” xhrTarget=”idControls”>TV</li>
      <li href=”…” onclick=”httpGET(this)” xhrTarget=“idControls”>Media Server A</li>
      ...
    </ul>
  </event>
</rex>
```

XProc



1. Addressing
- host / model # resource
- GET http://dmc/zones#home/livingroom/

2. Fetch XML fragment
- from DOM or Adapter or file
- select = "/home/livingroom/"

3. Transform into abstract UI
- browserAbstr.xsl (or /session[@id]/abstract)
- template match= "devices"

4. Transform into concrete UI
- browserConcr.xsl (or /session[@id]/concrete)
- template match= "choice"

- 58 lines of markup !
- procedural:
  - server
  - uri + header parser

XProc

W3C working draft (April 5 2007)

- pipelines of (compound) steps
  - input and output ports
  - xinclude, xslt, validate, http-request
  - load, store, serialize, parse, identity, join, subsequence
  - xslt2, xquery, formatter
- micro-operations
  - delete, insert, rename, replace, set, (un)wrap, label
- conditionals, loops, exceptions
- parameters, options
- pipeline libraries

for example

- get UI components from web (eg. icons from Tango)
- mashups: get metadata from IMDB or AMG

Techniques Summary

XProc	declarative glue
XSLT	to create ui pages / ui fragments / rex mutation, adapters also
REX	DOM mutation, user input, model state change notifications
REST	sync ui – model – target, access external devices/services
XPointer	XPath endian URIs to address resources
XPath	used in many places

Conclusion

Observations

- XProc seems to fit well into DMC scenario
- usable techniques are available, but might be difficult to find
- REST scales well, cleaner code than RPC, modeless, passes firewalls

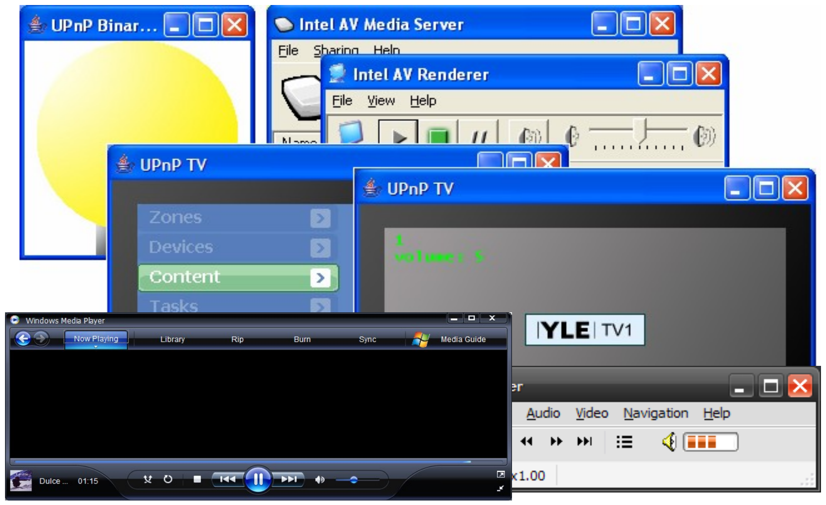
Some concerns

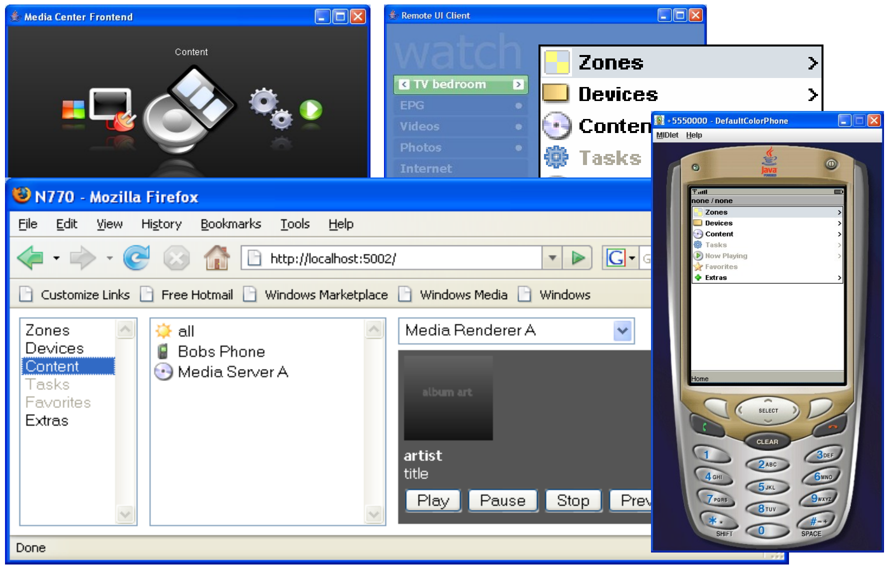
- tools for design, debug etc.
- security issues
- performance issues

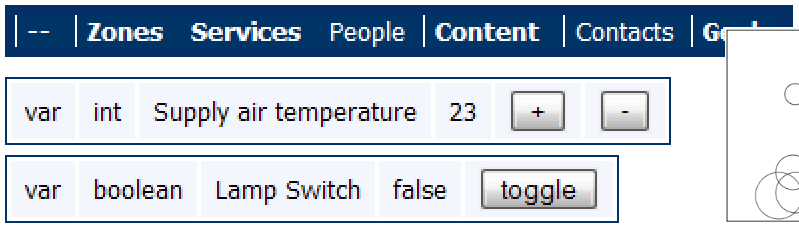
DMC Examples

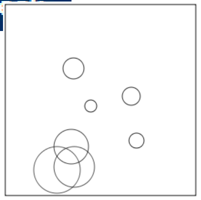
Targets

Concretized UIs






-- Zones Services People Content Contacts
var int Supply air temperature 23 + -
var boolean Lamp Switch false toggle



5.6.2007
Jari.Kleimola@tml.hut.fi

Thank You !

Questions ?