MOST: A 3D WEB ARCHITECTURE STYLE FOR HETEROGENIeous MODEL DATA

MoST Architecture

Max Limper
Timo Sturm
Johannes Behr

Fraunhofer IGD
Fraunhofer-Institut für Graphische Datenverarbeitung IGD
Fraunhoferstraße 5
64283 Darmstadt

V03
Overview

- **Motivation and Introduction**
- **Related Work**
  - ReST Architecture style
  - Web Standard Evolution
  - RFC 2077 Declaration
- **MoST Architecture Principles**
  - Model Representation as Application State
  - Model Domain Links
  - Model Resource Operation
  - Model Fragment URI
  - Server Model Operations
- **Examples**
- **Conclusions and Future Work**
Past: 3D Model Data and Tools is a Growing Market Since 25 years
Trend: Data and Distribution Complexity Exceed Capacity of Existing Solutions

Digital born
Digital scan

Open SaaS
Desk, mobile
VR/AR

3D Link Standards

Live Data
Challenges

- **Great: 3D Is Everywhere 😊**
  - **HW:** 3D Printer (e.g., Makerbot/ultimaker), 3D Scanner (e.g., FaceID), VR/AR (e.g., HoloLens)
  - **OS:** Windows 10 “Creator’s Update”, Paint 3D, MacOS 10.11 (Preview, SceneKit)
  - **Online 3D Service** with APIs to import and/or export 3D model data:
    - **Printing** (e.g., Shapeways)
    - **Scanning** (e.g., Autodesk)
    - **Modeling** (e.g., Onshape, Vectary)
    - **Simulation** (e.g., Simscale)
    - **Viewing and Sharing** (e.g., Sketchfab, Remix3D/Microsoft, Poly/Google)

- **Challenging: 3D Data Platform Boundaries 😞**
  - Data needs to be imported/exported and managed in the system and silo boundaries
  - No common interface pattern to access live data
  - No common pattern to compose data from different services
Overview

- Motivation and Introduction
- Related Work
  - ReST Architecture style
  - Web Standard Evolution
  - RFC 2077 Declaration
- MoST Architecture Principles
  - Model Representation as Application State
  - Model Domain Links
  - Model Resource Operation
  - Model Fragment URI
  - Server Model Operations
- Examples
- Conclusions and Future Work
ReST Architecture Style

- **2000**: Open architecture style for distributed data and service points
  - Web of documents -> Web of resource states
  - Read/Consume interfaces -> Create/Read/Update/Delete (CRUD) interfaces
  - Single representation (e.g. RDF) -> Dynamic representation and late bindings
- Used by all modern service platform interfaces (Google, Amazon, Microsoft, RAMI (Industry 4.0))
ReST Architecture Style

- REST is the major interface pattern for Microservice Architectures (MSA)

Introduction to Microservices, Chris Richardson, 2015
## Web Standard Evolution

<table>
<thead>
<tr>
<th>Web 3D Generation</th>
<th>Server Structure</th>
<th>Transport &amp; Interfaces</th>
<th>Client</th>
<th>3D Standards</th>
<th>Interop. Challenge</th>
</tr>
</thead>
<tbody>
<tr>
<td>1990: 0.0</td>
<td>File-Server</td>
<td>Static File</td>
<td>Fat-Client</td>
<td>obj, ply, iv</td>
<td>Client/Server</td>
</tr>
<tr>
<td>1994: 1.0</td>
<td>Web-Server</td>
<td>Static Document</td>
<td>Web-Browser + Plugin</td>
<td>VRML 1.0</td>
<td>Client/Service</td>
</tr>
<tr>
<td>1998: 2.0</td>
<td>Multi-Tier AS RFC//SOAP</td>
<td>Partial Data Updates</td>
<td>Web-Browser + Plugin</td>
<td>VRML 2.0 X3D, 3D-PDF</td>
<td>Client/Service</td>
</tr>
<tr>
<td>2005: 3.0</td>
<td>Multi-Service RDF Arch</td>
<td>Partial Data Updates</td>
<td>Web-Browser / Apps</td>
<td>Individual Vocabulary</td>
<td>Client/Service Service/Service</td>
</tr>
<tr>
<td>2015: 4.0</td>
<td>Micro-Service REST Arch</td>
<td>Continuous Delivery</td>
<td>Progressive Web-Apps</td>
<td>RFC2077 WebGL</td>
<td>Service/Service</td>
</tr>
</tbody>
</table>
### Web Standard Evolution: From Client/Server to Service/Service Interfaces

<table>
<thead>
<tr>
<th>Year</th>
<th>Generation</th>
<th>Server Structure</th>
<th>Transport &amp; Interfaces</th>
<th>Client</th>
<th>3D Standards</th>
<th>Interop. Challenge</th>
</tr>
</thead>
<tbody>
<tr>
<td>1990</td>
<td>0.0</td>
<td>File-Server</td>
<td>Static File</td>
<td>Fat-Client</td>
<td>obj, ply, iv</td>
<td>Client/Server</td>
</tr>
<tr>
<td>1994</td>
<td>1.0</td>
<td>Web-Server</td>
<td>Static Document</td>
<td>Web-Browser + Plugin</td>
<td>VRML 1.0</td>
<td>Client/Service</td>
</tr>
<tr>
<td>1998</td>
<td>2.0</td>
<td>Multi-Tier AS RFC//SOAP</td>
<td>Partial Data Updates</td>
<td>Web-Browser + Plugin</td>
<td>VRML 2.0, X3D, 3D-PDF</td>
<td>Client/Service</td>
</tr>
<tr>
<td>2005</td>
<td>3.0</td>
<td>Multi-Service RDF Arch</td>
<td>Partial Data Updates</td>
<td>Web-Browser / Apps</td>
<td>Individual Vocabulary</td>
<td>Client/Service</td>
</tr>
<tr>
<td>2015</td>
<td>4.0</td>
<td>Micro-Service REST Arch</td>
<td>Continuous Delivery</td>
<td>Progressive Web-Apps</td>
<td>RFC2077 WebGL</td>
<td>Service/Service</td>
</tr>
</tbody>
</table>
**URI, Resource and Representation**

- **ReST: URI, Resources and Representation** are the core concepts
  - Any form of information
  - Late data/format bindings
  - Allows concept referencing

- **Representation types and encoding**
  - Content type ([IANA](https://www.iana.org/assignments/media-types))
    - “text/html”, “application/json”
  - JSON has no link semantic!
  - Contains type-specific data
    - local payload (e.g., name)
    - links to related resources
  - => Linked Resource Network ([HATEOAS](https://www.w3.org/TR/rdf-hemp/))

```json
{
  "customer" : {
    "name" : "John Smith"
  }
  "storage" : [
    "http://cloud.example.com/SID/34343"
  ]
}
```

e.g.: “application/json” encoded representation
Representation encoding domains

- Text: html, json
- Image: jpeg, png
- Application: application
- Video: mp4, h264
- Audio: audio
- Messages: messages
- Model: model
1997: RFC2077 model/* domain

- **Space**
  - Orthogonal system
  - Spatial (3 or more) and temporal dimensions

- **Model**
  - Models are composed of Objects
  - Objects are instantiated/transformed elements
  - Elements are local or extern linked data
    - Data aggregation semantic
      - E.g. referencing a VRML and IGES rep

- **Structured Data**
  - Multidimensional structures
  - Conversion should be structure preserving
## Resource specification – Content types

### Content types: “model/*” ([IANA Registration](https://tools.ietf.org/html/rfc7231))

<table>
<thead>
<tr>
<th>Date</th>
<th>Name</th>
<th>Template</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>01/01/1997</td>
<td>iges</td>
<td>model/iges</td>
<td><a href="mailto:curtis.parks@cs.man.ac.uk">Curtis_Parks</a></td>
</tr>
<tr>
<td>01/01/1997</td>
<td>mesh</td>
<td>model/mesh</td>
<td><a href="https://tools.ietf.org/html/rfc2077">RFC2077</a></td>
</tr>
<tr>
<td>01/01/1997</td>
<td>vrml</td>
<td>model/vrml</td>
<td><a href="https://tools.ietf.org/html/rfc2077">RFC2077</a></td>
</tr>
<tr>
<td>01/01/1997</td>
<td>vnd.dwf</td>
<td>model/vnd.dwf</td>
<td><a href="mailto:jp@igr.isc.fu-berlin.de">Jason_Pratt</a></td>
</tr>
<tr>
<td>01/01/1997</td>
<td>vnd.gtw</td>
<td>model/vnd.gtw</td>
<td><a href="mailto:yutaka.ozaki@ics.isc.fu-berlin.de">Yutaka_Ozaki</a></td>
</tr>
<tr>
<td>01/01/1997</td>
<td>vnd.flatland.3dml</td>
<td>model/vnd.flatland.3dml</td>
<td><a href="mailto:michael.powers@ics.isc.fu-berlin.de">Michael_Powers</a></td>
</tr>
<tr>
<td>20/08/2002</td>
<td>vnd.parasolid.transmit.binary</td>
<td>model/vnd.parasolid.transmit-binary</td>
<td><a href="http://parasolid.igs.com">Parasolid</a></td>
</tr>
<tr>
<td>20/08/2002</td>
<td>vnd.parasolid.transmit.text</td>
<td>model/vnd.parasolid.transmit-text</td>
<td><a href="http://parasolid.igs.com">Parasolid</a></td>
</tr>
<tr>
<td>20/08/2002</td>
<td>vnd.gs-gdl</td>
<td>model/vnd.gs-gdl</td>
<td><a href="mailto:attila.babits@ics.isc.fu-berlin.de">Attila_Babits</a></td>
</tr>
<tr>
<td>20/08/2002</td>
<td>vnd.mts</td>
<td>model/vnd.mts</td>
<td><a href="mailto:boris.rabinovitch@ics.isc.fu-berlin.de">Boris_Rabinovitch</a></td>
</tr>
<tr>
<td>20/08/2002</td>
<td>vnd.vtu</td>
<td>model/vnd.vtu</td>
<td><a href="mailto:boris.rabinovitch@ics.isc.fu-berlin.de">Boris_Rabinovitch</a></td>
</tr>
<tr>
<td>16/08/2006</td>
<td>vnd.moml+xml</td>
<td>model/vnd.moml+xml</td>
<td><a href="mailto:christopher.brooks@ics.isc.fu-berlin.de">Christopher_Brooks</a></td>
</tr>
<tr>
<td>01/02/2011</td>
<td>vnd.gdl</td>
<td>model/vnd.gdl</td>
<td><a href="mailto:attila.babits@ics.isc.fu-berlin.de">Attila_Babits</a></td>
</tr>
<tr>
<td>01/02/2011</td>
<td>vnd.collada+xml</td>
<td>model/vnd.collada+xml</td>
<td><a href="mailto:riordon@web3d.org">James_Riordon</a></td>
</tr>
<tr>
<td>01/02/2011</td>
<td>x3d+xml</td>
<td>model/x3d+xml</td>
<td><a href="http://www.web3d.org">Web3D</a></td>
</tr>
<tr>
<td>01/02/2011</td>
<td>x3d-vrml</td>
<td>model/x3d-vrml</td>
<td><a href="http://www.web3d.org">Web3D_X3D</a></td>
</tr>
<tr>
<td>01/02/2011</td>
<td>x3d+fastinfoset</td>
<td>model/x3d+fastinfoset</td>
<td><a href="http://www.web3d.org">Web3D_X3D</a></td>
</tr>
<tr>
<td>01/02/2011</td>
<td>vnd.opengex</td>
<td>model/vnd.opengex</td>
<td><a href="mailto:eric.lengyel@ics.isc.fu-berlin.de">Eric_Lengyel</a></td>
</tr>
<tr>
<td>01/02/2011</td>
<td>vnd.valve.source.compiled-map</td>
<td>model/vnd.valve.source.compiled-map</td>
<td><a href="mailto:henrik.andersson@ics.isc.fu-berlin.de">Henrik_Andersson</a></td>
</tr>
<tr>
<td>01/02/2011</td>
<td>vnd.rosette.annotated-data-model</td>
<td>model/vnd.rosette.annotated-data-model</td>
<td><a href="mailto:bmargulies@bnl.gov">Benson_Margulies</a></td>
</tr>
<tr>
<td>12/05/2017</td>
<td>gltf+json</td>
<td>model/gltf+json</td>
<td><a href="http://www.khronos.org">Khronos</a></td>
</tr>
<tr>
<td>01/07/2017</td>
<td>3mf</td>
<td>model/3mf</td>
<td><a href="http://www.3mf.org">3MF</a></td>
</tr>
<tr>
<td>01/07/2017</td>
<td>gltf-binary</td>
<td>model/gltf-binary</td>
<td><a href="http://www.khronos.org">Khronos</a></td>
</tr>
</tbody>
</table>
### Resource specification – Content types – Classification Surface/Link Types

<table>
<thead>
<tr>
<th>Surface Types (e.g. Mesh, Nurbs, BRep)</th>
<th>Multi-Class</th>
<th>Single-Class</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi-Class</td>
<td>e.g. X3D</td>
<td>e.g. gltf</td>
<td>e.g. step242</td>
</tr>
<tr>
<td>Single-Class</td>
<td>e.g. collada</td>
<td></td>
<td></td>
</tr>
<tr>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Link Types</th>
<th>Target Formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Single-Type</td>
</tr>
<tr>
<td></td>
<td>Multi-Type</td>
</tr>
<tr>
<td></td>
<td>Full domain</td>
</tr>
</tbody>
</table>
Overview

- Motivation and Introduction
- Related Work
  - ReST Architecture style
  - Web Standard Evolution
  - RFC 2077 Declaration
- MoST Architecture Principles
  - Model Representation as Application State
  - Model Domain Links
  - Model Resource Operation
  - Model Fragment URI
  - Server Model Operations
- Examples
- Conclusions and Future Work
MoST Architecture Principles

- Format-agnostic 3D model graph for massive data
  - Generalization and concrete, real-world 3D interpretation of the abstract RFC 2077 concept
  - A generalized "compose" semantic for 3D model data
  - Hybrid format data using standard content negotiation
  - Transferring the "Media Fragments URI" to a new domain
  - Generalized Objects accessed via novel Fragment URI
Model Representation as Application State

- **RW/CRUD on Model Data**
  - **Live Data** and not import/export APIs
  - From viewing to service interfaces

- **Model Representation**
  - Any model/* format
    - e.g. VRML, X3D, GLTF/GLB, STEP242
  - **Local payload** and linked references
    - Local 3D element instances
    - Any model/* format
      - Again any model/* data

- **Classification per standard**

```latex
DEFL CarBody Shape {
  geometry IndexFaceSet {} 
}

DEFL CarTire Inline {
  url [ "http://example.org/model/tire" ]
}

Transform { 
  translation -2 -1 -1 
  children [ USE CarTire ]
}
e.g.: “model/vrml” encoded representation
```
Model Domain Links

- Links should be representation-independent
- Service should use content negotiation for late bindings
  - E.g., STL for printing and glTF for viewing
- Standards should support generic content types
  - "model/*"
- Standards should support any number of URIs for single references
  - urn:part:23432423
  - http://model/most/23434
  - Youdata://model/2433
# Model Resource Operation

<table>
<thead>
<tr>
<th>HTTP Operation</th>
<th>Request Data (GET)</th>
<th>Request Property (HEAD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>★★★★★</td>
<td>HTTP2 push updates</td>
<td>HTTP2 push updates</td>
</tr>
<tr>
<td>★★★★★☆</td>
<td>HTTP pipelinig</td>
<td>HTTP pipelinig</td>
</tr>
<tr>
<td>★★★☆☆☆</td>
<td>Provide etag, Cache-Control</td>
<td>Validate etag</td>
</tr>
<tr>
<td>★★☆☆☆☆</td>
<td>Provide WWW-Authenticate</td>
<td>Validate SSO, 401 on fail</td>
</tr>
<tr>
<td>★☆☆☆☆☆</td>
<td>200 (OK), 401 (Unauthorized), 404</td>
<td>No Hander, No WWW-Authenticate data</td>
</tr>
</tbody>
</table>
Model Fragment URI

- URI Fragment standards allow to address sub-data in representation
  - E.g.: [RFC7111](https://www.rfc-editor.org/rfc/rfc7111) for “text/csv”: “example.com/data.csv#row=3-5”

- **2012 W3C Recommendation: Media Fragment URI**
  - Spatially-, temporally- and structure-based addressing schema
    - E.g. example.com/media/movie.mpg#xywh=100,100,10,10&t=10,20
    - E.g example.com/media/movie.mpg#id=cap

- **New Proposal: “Model Fragment URI”**
  - Spatially- and temporally-based addressing schema
    - E.g., example.com/model/434/#xyzwhz=100,100,100,20,20,20&t=10,20
  - Object- and Object-set-based addressing schema
    - example.com/model/434/#objects=“/tire”
Server Model Operation

- URI Fragments are built to be processed in the consuming service
  - Fragment is encoding independent
  - Objects as example.com/model/434/#objects="/tire"
  - Service can use local data for different fragments
  - Service might need specific formats for given fragments
  - Service might need alternative representation

- Alternative service request structure
  - Objects as example.com/model/434/#objects="/tire"
  - Translates to standard server request
  - example.com/model/434/?objects="/tire"
Overview

- Motivation and Introduction
- Related Work
  - ReST Architecture style
  - Web Standard Evolution
  - RFC 2077 Declaration
- MoST Architecture Principles
  - Model Representation as Application State
  - Model Domain Links
  - Model Resource Operation
  - Model Fragment URI
  - Server Model Operations
- Examples
- Conclusions and Future Work
Use-Case Examples: Select Data from given Room-Volumes

- Multi-Resource Data
  - Structure => Parts (X3D => OBJ)
- Declarative 3D-Room selection with Fragment selection
  - E.g datagateway.local/most/3423423#xyzwhz=100,100,100,20,20,20
Use-Case Example: VIN Data Selector / Digital Twin

- Multi-Resource Data
  - Body => Structure => Segment selection
- Digrial-Twin Use-case
  - Service provides declarative setup for scanned VIN (Vehicle identification number)
    - example.com/kbl/434/?objects="2343,3424,2224"
Use-Case Example: Compose 3D Data for printing

- Multi-Resource Data
  - Print-Jobs => Structure => Data-Elements
- Print-Jobs uses group of different Data-Sets
  - example.com/data/214/?objects="34"
  - example.com/data/434/?objects="24"
Conclusions and Future Work

- **Linked 3D Data** = Powerful concept, flexible and scalable (massive 3D data)
- Related past developments: ReST, RFC2077, Web/3D standards
- **MoST Architecture Principles:**
  - Model Representation as Application State
  - Model Domain Links
  - Model Resource Operation
  - Model Fragment URI
  - Server Model Operations

- Future Work: Proof-of-Concept implementation
  - Web Portal (Open Source standard + validation Tools )
  - Multiple supporting organizations / Standards
  - “Model Fragment URI” standardization process inside W3C