Addressing On-Demand Assembly and Adaptation
Using a Runtime Intentional Versioning Engine

Jaroslav Gergič

DISTRIBUTED SYSTEMS RESEARCH GROUP
http://nenya.ms.mff.cuni.cz

CHARLES UNIVERSITY PRAGUE
Faculty of Mathematics and Physics
• Multimodal Web Applications
  – the client capabilities only known at runtime

• Multimodal Web Applications
  – allowing for different modes of interaction depending on a client device and its capabilities
  – providing several alternative user interface with possibly very different interaction styles

• Current situation: Multi-authoring
  – separate applications for PCs, mobile, voice, etc.
  – duplication of efforts, inconsistent user experience
  – device-driven adaptation resulting in poor coverage

• Target: Single Authoring
  – enable parallel development in a single integrated and coherent application framework
  – device independent capability-driven adaptation
  – expand coverage while managing the cost
W3C/OMA have been trying to address device independence:

- **RDF** – Resource Description Framework (W3C)
  - the foundation of the Semantic Web
- **CC / PP** – Composite Capabilities / Preference Profiles (W3C)
  - structure and transport framework
- **UAPerf** – User Agent Profile (OMA)
  - a concrete CC/PP vocabulary
- **JSR-188** - CC/PP Processing Specification (Java Community Process)
  - standard Java extension API
Problem Statement

• The W3C/OMA standards stack provides the necessary technological basis
  - does not sufficiently address all aspects
  - introduces some issues on its own

• Three major issues still affecting multimodal applications development

  (1) Metadata Consolidation
  (2) Metadata Canonicalization
  (3) Level of Abstraction Gap
The same metadata attribute (*locale*) needs to be consolidated from multiple overlapping sources:

1) web browser (HTTP header)
2) web application (session cookie)
3) user profile (persistent storage)

- CC/PP is not the exclusive source of relevant metadata
- Not all devices support CC/PP:
  - CC/PP broadly supported by cell phone manufacturers
  - abandoned by traditional web browser providers
- There is a need to support heterogeneous environments with many metadata sources
- We need to be able to define
  - resolution policy (order of precedence)
  - fall-back strategy (if preferred source can not provide metadata)
- above: UAProf 2.0 **JavaPlatform** attribute values*)
- fact: 64 out of 109 UAProf 2.0 metadata attributes are of type *Literal* (an arbitrary string)

*) References (used for data analysis):
[UAREP1]  *UAProf Repository*, WWW Repository, W3Development.de, 2005, [http://w3development.de/rdf/uaprof_repository/] (234 profiles, all major vendors represented)
(3) Level of Abstraction

- UAProf 2.0 RDF Schema:
  - `ScreenSize` of type `Dimension`
    - a pair of positive integers
      (e.g. 640x480 or 1024x768)

- UAProf 2.0 does not mention the fact that screen resolutions are actually well standardized
  - QVGA, VGA, SVGA, etc.

- CC/PP does not directly support any kind of classification/clustering schema
  - impossible to customize an application for each and every potential or even actual screen resolution

- possible solution: `ScreenSizeClass` taxonomy:
  - hierarchical classification of screen resolutions
  - can be derived from the UAProf `ScreenSize` or other sources
  - raises the level of abstraction
  - allows for incremental generalization (subsumption) and clustering (resource variant reuse)
Goals

- Support Metadata Consolidation
  - resolution rules and fall-back strategies
- Support Metadata Canonicalization
  - prevent proliferation to the main application logic
- Raise the Level of Abstraction
  - hierarchical classification, device clustering, generalization

Secondary Goals

- Avoid the Domain Expertise Issue
  - leverage common knowledge (no need to master Semantic Web in order to develop web applications)
- Best Practices Enforcement
  - separation of concerns, modularity
Versatile Framework: Key Concepts

• Semantically rich metadata **Properties**
  - *taxonomy*, controlled vocabulary, binary relation, (partial) order

• Chained **Value Providers** and **Property Mappings**
  - resolution rules and fall-back strategies for metadata acquisition
  - metadata canonicalization and transformation services

• **Delivery Context**
  - centralized metadata provider integrating all external metadata sources into a single coherent view

• **Query Templates**
  - high-level configuration rules used to express **constraints** and **preferences**
  - reusable – one query template for each category of resources

• **Resource Providers**
  - resource/class factories for versioned entities
  - implement multicritical constraining and subsumption-based approximate matching with result **scoring** (fidelity score)
Layout Manager Example

- Task: instantiate a layout manager for a particular device
  - a layout manager is a component which controls how individual user interface widgets are positioned on the screen.

- The device provides its capabilities as a list of concrete attribute/value pairs.

- The application abstracts away from concrete values where appropriate, by clustering devices by similarity
  - in order to reuse layout managers for similarly capable devices

ScreenSize = 480x640  FramesCapable = false
Part I – Delivery Context

- **query template**: a set of reusable constraints and preferences
- **query**: specifies concrete resource and property values

```
FRAMES_CAPABLE = false
ScreenSize = 480x640
ScreenSizeClass = PDA
ScreenOrientation = Portrait
```

```
qt : QueryTemplate
NBest (default = 1)
ScoringFactor (default = 0.99)
P[0] { assert, FRAMES_CAPABLE, false}
P[1] { bestMatch, SCREEN_SIZE_CLASS, "PDA"}
```

```
deliveryContext : DeliveryContext
```

```
quarry : Query
ResourceName = "LayoutManager"
NBest (default = 1)
ScoringFactor (default = 0.99)
P[0] { assert, FRAMES_CAPABLE, false}
P[1] { bestMatch, SCREEN_SIZE_CLASS, "PDA"}
```

```
uiFactory : ResourceProvider
```

```
query = qt.newQuery("LayoutManager");
ResultSet rSet = uiFactory.get(query);
```

- Metadata acquisition/interpretation managed and automated by the framework.
- consolidation, canonicalization/Transformation
- The developer works in terms of high-level constraints/preferences.
Part II – Resource Provider

- Finding the *most appropriate* implementation of the layout manager in the user interface components repository.
- Key feature is the ability to support both constraints and preferences applied to multiple metadata attributes – multicriterial constraining and matching

---

Table 1: A resource repository example

<table>
<thead>
<tr>
<th>UID</th>
<th>ResourceName</th>
<th>FramesCapable</th>
<th>ScreenSizeClass</th>
<th>ScreenOrientation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MainMenu</td>
<td>TRUE</td>
<td>Tablet</td>
<td>Landscape</td>
</tr>
<tr>
<td>2</td>
<td>ToolBar</td>
<td>TRUE</td>
<td>PersonalComputer</td>
<td>Landscape</td>
</tr>
<tr>
<td>3</td>
<td>ContextMenu</td>
<td>FALSE</td>
<td>CellPhone</td>
<td>Portrait</td>
</tr>
<tr>
<td>4</td>
<td>LayoutManager</td>
<td>TRUE</td>
<td>PDA</td>
<td>/</td>
</tr>
<tr>
<td>5</td>
<td>LayoutManager</td>
<td>FALSE</td>
<td>PDA</td>
<td>/</td>
</tr>
<tr>
<td>6</td>
<td>LayoutManager</td>
<td>FALSE</td>
<td>CompactDevice</td>
<td>Portrait</td>
</tr>
<tr>
<td>7</td>
<td>LayoutManager</td>
<td>FALSE</td>
<td>PDA</td>
<td>Landscape</td>
</tr>
</tbody>
</table>
Part II – Resource Provider

- Finding the *most appropriate* implementation of the layout manager in the user interface components repository.
- Key feature is the ability to support both *constraints* and *preferences* applied to multiple metadata attributes
  - multicriterial constraining and matching

Table 1: A resource repository example

<table>
<thead>
<tr>
<th>UID</th>
<th>ResourceName</th>
<th>FramesCapable</th>
<th>ScreenSizeClass</th>
<th>ScreenOrientation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MainMenu</td>
<td>TRUE</td>
<td>Tablet</td>
<td>Landscape</td>
</tr>
<tr>
<td>2</td>
<td>ToolBar</td>
<td>TRUE</td>
<td>PersonalComputer</td>
<td>Landscape</td>
</tr>
<tr>
<td>3</td>
<td>ContextMenu</td>
<td>FALSE</td>
<td>CellPhone</td>
<td>Portrait</td>
</tr>
<tr>
<td>4</td>
<td>LayoutManager</td>
<td>TRUE</td>
<td>PDA</td>
<td>/</td>
</tr>
<tr>
<td>5</td>
<td>LayoutManager</td>
<td>FALSE</td>
<td>PDA</td>
<td>/</td>
</tr>
<tr>
<td>6</td>
<td>LayoutManager</td>
<td>FALSE</td>
<td>CompactDevice</td>
<td>Portrait</td>
</tr>
<tr>
<td>7</td>
<td>LayoutManager</td>
<td>FALSE</td>
<td>PDA</td>
<td>Landscape</td>
</tr>
</tbody>
</table>
Part II – Resource Provider

- Finding the *most appropriate* implementation of the layout manager in the user interface components repository.
- Key feature is the ability to support both *constraints* and *preferences* applied to multiple metadata attributes
  - multicriterial constraining and matching

Table 1: A resource repository example

<table>
<thead>
<tr>
<th>UID</th>
<th>ResourceName</th>
<th>FramesCapable</th>
<th>ScreenSizeClass</th>
<th>ScreenOrientation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MainMenu</td>
<td>TRUE</td>
<td>Tablet</td>
<td>Landscape</td>
</tr>
<tr>
<td>2</td>
<td>ToolBar</td>
<td>TRUE</td>
<td>PersonalComputer</td>
<td>Landscape</td>
</tr>
<tr>
<td>3</td>
<td>ContextMenu</td>
<td>FALSE</td>
<td>CellPhone</td>
<td>Portrait</td>
</tr>
<tr>
<td>4</td>
<td>LayoutManager</td>
<td>TRUE</td>
<td>PDA</td>
<td>/</td>
</tr>
<tr>
<td>5</td>
<td>LayoutManager</td>
<td>FALSE</td>
<td>PDA</td>
<td>/</td>
</tr>
<tr>
<td>6</td>
<td>LayoutManager</td>
<td>FALSE</td>
<td>CompactDevice</td>
<td>Portrait</td>
</tr>
<tr>
<td>7</td>
<td>LayoutManager</td>
<td>FALSE</td>
<td>PDA</td>
<td>Landscape</td>
</tr>
</tbody>
</table>

Finding the most *appropriate* implementation of the layout manager in the user interface components repository.
Part II – Resource Provider

- Finding the *most appropriate* implementation of the layout manager in the user interface components repository.
- Key feature is the ability to support both *constraints* and *preferences* applied to multiple metadata attributes
  - multicriterial constraining and matching

Table 1: A resource repository example
Part II – Resource Provider

- Finding the *most appropriate* implementation of the layout manager in the user interface components repository.
- Key feature is the ability to support both *constraints* and *preferences* applied to multiple metadata attributes
  - multicriterial constraining and matching

Table 1: A resource repository example
Part II – Resource Provider

- Finding the *most appropriate* implementation of the layout manager in the user interface components repository.
- Key feature is the ability to support both *constraints* and *preferences* applied to multiple metadata attributes
  - multicriteria constraining and matching

Step 1: Fetch Resources by Name

---

<table>
<thead>
<tr>
<th>UID</th>
<th>ResourceName</th>
<th>FramesCapable</th>
<th>ScreenSizeClass</th>
<th>ScreenOrientation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MainMenu</td>
<td>TRUE</td>
<td>Tablet</td>
<td>Landscape</td>
</tr>
<tr>
<td>2</td>
<td>ToolBar</td>
<td>TRUE</td>
<td>PersonalComputer</td>
<td>Landscape</td>
</tr>
<tr>
<td>3</td>
<td>ContextMenu</td>
<td>FALSE</td>
<td>CellPhone</td>
<td>Portrait</td>
</tr>
<tr>
<td>4</td>
<td>LayoutManager</td>
<td>TRUE</td>
<td>PDA</td>
<td>/</td>
</tr>
<tr>
<td>5</td>
<td>LayoutManager</td>
<td>FALSE</td>
<td>PDA</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>LayoutManager</td>
<td>FALSE</td>
<td>CompactDevice</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>LayoutManager</td>
<td>FALSE</td>
<td>PDA</td>
<td>Landscape</td>
</tr>
</tbody>
</table>

*Table 1: A resource repository example*
# Evaluation / Related Work

<table>
<thead>
<tr>
<th>Framework/Project</th>
<th>Metadata Consolidation</th>
<th>Metadata Canonicalization</th>
<th>Level of Abstraction</th>
<th>Domain Expertise</th>
<th>Best Practices</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Versatile Framework</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>Volantis Mobile Content Framework (commercial)</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>++*</td>
<td>n/a</td>
</tr>
<tr>
<td>MobileAware Interaction Server (commercial)</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>WURFL / WALL (Luca Passani, an open source project)</td>
<td>+</td>
<td>++</td>
<td>+</td>
<td>++*</td>
<td>n/a</td>
</tr>
<tr>
<td>DELI with Capability Classes (Mark H Butler, HP Labs, UK)</td>
<td>–</td>
<td>+</td>
<td>++</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Project PACE (Indulska J., et al.; University of Queensland, Australia)</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>+</td>
<td>–</td>
</tr>
<tr>
<td>&quot;Simple CC/PP&quot; (Korolev V., et al.; University of Maryland Baltimore County, USA)</td>
<td>–</td>
<td>–</td>
<td>+</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>OPERA Project, WAM Project (Lemlouma T., Layaida N.; INRIA Rhône Alpes, France)</td>
<td>–</td>
<td>–</td>
<td>+</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>CC/PP and annotation (Hwe-Mo Kim, Kyong-Ho Lee; Yonsei University, South Korea)</td>
<td>–</td>
<td>+</td>
<td>++</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>Rule-based adaptation (Stephen J.H. Yang, Norman W.Y. Shao; National Central University, Taiwan)</td>
<td>–</td>
<td>–</td>
<td>++</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Graceful Degradation (Florins M., et al.; Université catholique de Louvain, Belgium)</td>
<td>–</td>
<td>–</td>
<td>+</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>Project MONA (Telecommunications Research Center - ftw., Vienna, Austria)</td>
<td>–</td>
<td>–</td>
<td>+</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>The W3C / OMA Standards Stack (CC/PP, UAProf)</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>+</td>
<td>–</td>
</tr>
</tbody>
</table>

++ good, + moderate, – out of scope, *) vendor lock-in, n/a – not enough information
Current Status and Future Work

- **versatile.jar** – a Java API library consisting of 45 interfaces and abstract classes
- **versatile-api.pdf** – an API Reference Manual, over 100 pages
  - detailed specifications, implementation guidelines and examples
- The library can be loaded into a Java IDE (e.g. Eclipse) and used to develop examples
  - sufficient for dry run testing and code-impact analysis
  - as there are not concrete implementations of the framework entities, the examples compile, but can not be actually executed
- Given the extent of the framework, not possible to proceed with a full-scale implementation as a single person
  - possibility: an open source community effort
- Possible applications outside of the chosen domain
  - web services, business rules
Publications

Master's Thesis

Refereed Papers

International Standards

Patent Applications

Projects: SOFA/DCUP CATCH 2004 WSVA/MMOD
Thank you ...

http://dsrg.mff.cuni.cz

For more information please refer to http://dsrg.mff.cuni.cz/~gergic/versatile/