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

October 31 2007

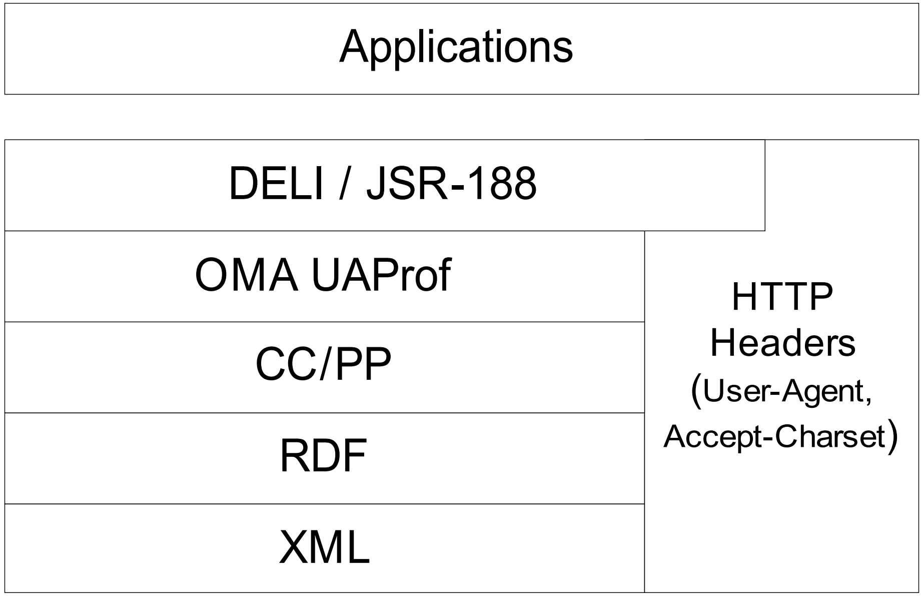
Context: Multimodal Web Applications

- Multimodal <u>Web</u> Applications
  - the client capabilities only known at runtime
- <u>Multimodal</u> Web Applications
  - allowing for different <u>modes of interaction</u> 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

Background: Existing Standards

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
- **UAProf** – User Agent Profile (OMA)
  - a concrete CC/PP vocabulary
- JSR-188 - CC/PP Processing Specification (Java Community Process)
  - standard Java extension API



Applications

DELI / JSR-188

OMA UAProf

CC/PP

RDF

XML

HTTP Headers (User-Agent, Accept-Charset)

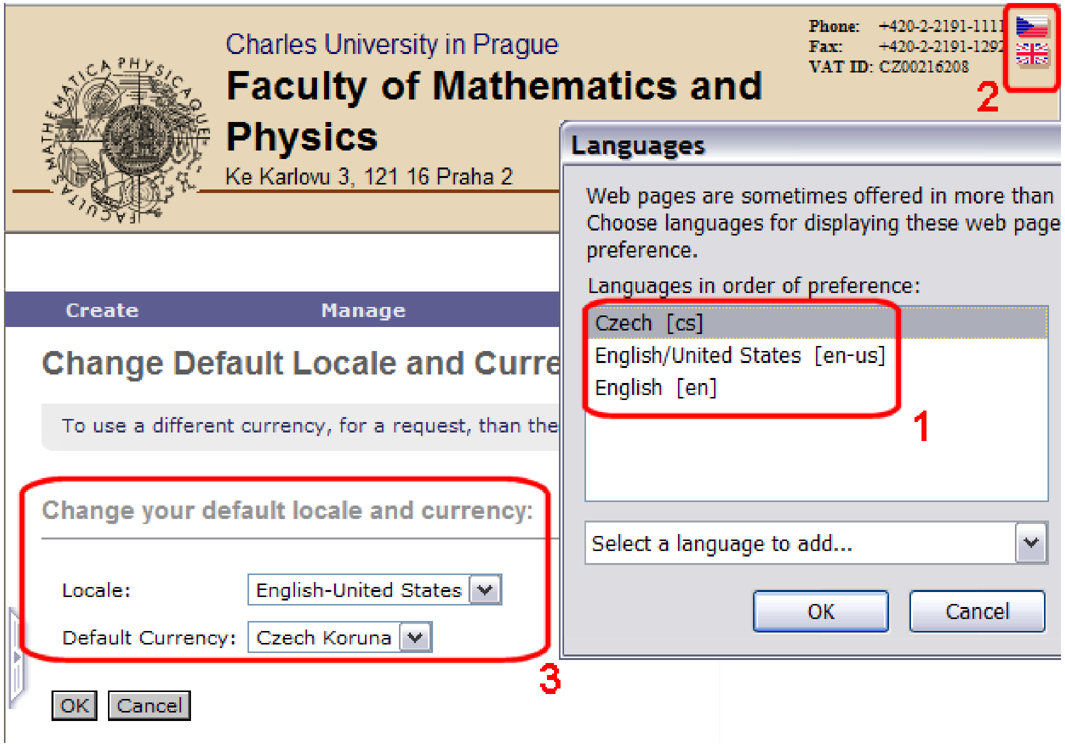
W3C: World Wide Web Consortium
OMA: Open Mobile Alliance
JSR: Java Specification Request
XML: eXtensible Markup Language

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

(1) Metadata Consolidation



Charles University in Prague
Faculty of Mathematics and Physics
Ke Karlovu 3, 121 16 Praha 2

Phone: +420-2-2191-1111
Fax: +420-2-2191-1292
VAT ID: CZ00216208

Change Default Locale and Curre

To use a different currency, for a request, than the

Change your default locale and currency:

Locale: English-United States

Default Currency: Czech Koruna

Languages

Web pages are sometimes offered in more than
Choose languages for displaying these web page
preference.

Languages in order of preference:

Czech [cs]
English/United States [en-us]
English [en]

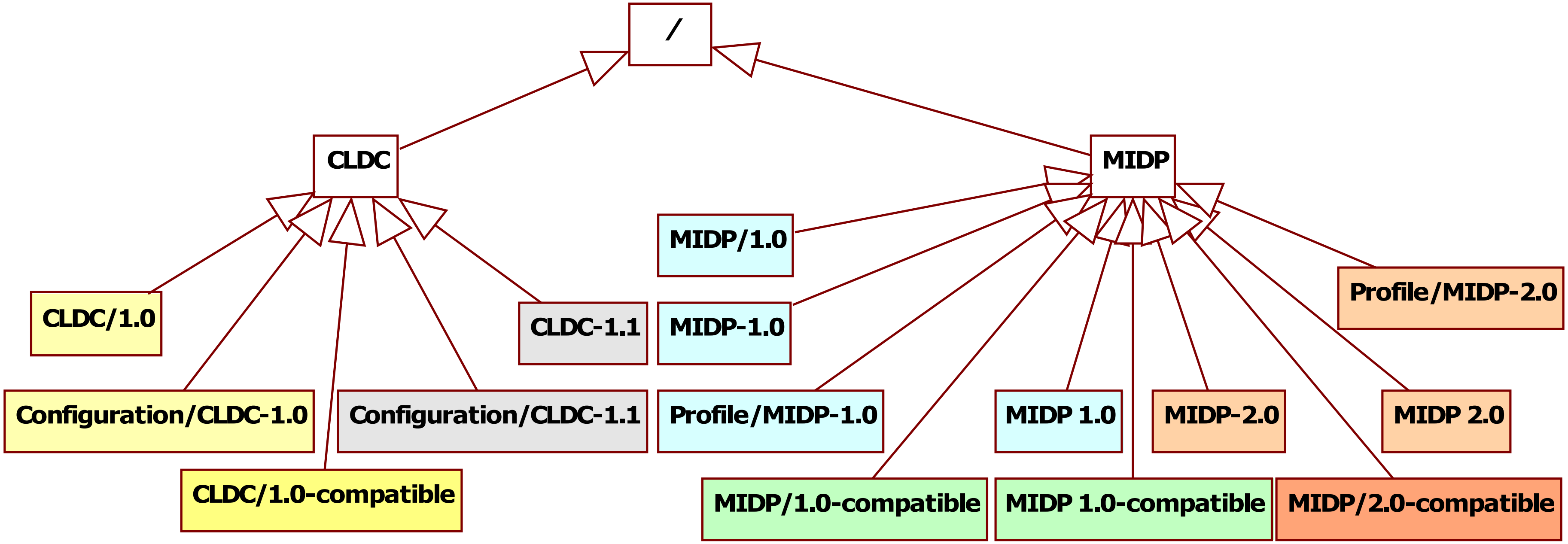
Select a language to add...

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)

(2) Metadata Canonicalization

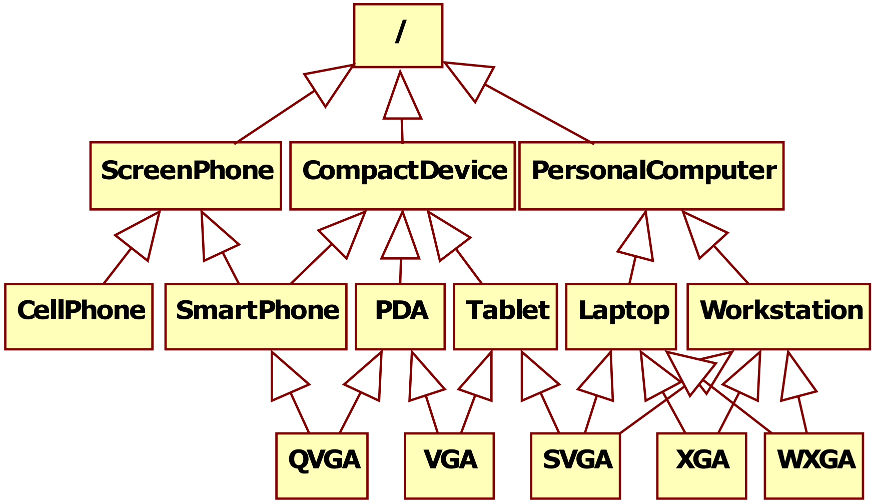


- above: UAProf 2.0 **JavaPlatform** attribute values*)
- fact: **64** out of 109 UAProf 2.0 metadata attributes are of type *Literal* (an arbitrary string)

*) 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)
[UAREP2] *UAProf Profiles*, WWW Repository, Nokia Corporation, 2007,
http://nds.nokia.com/uaprof/ (**436** profiles of Nokia and Vertu devices)

(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

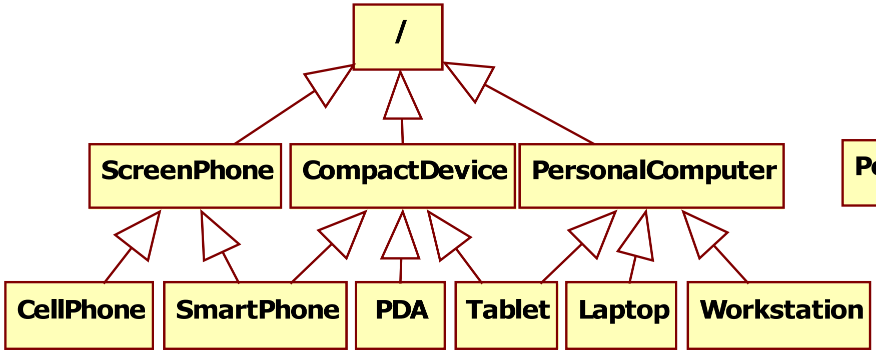
Versatile Framework: Key Concepts

- 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 multicriterial constraining and subsumption-based approximate matching with result scoring (fidelity score)

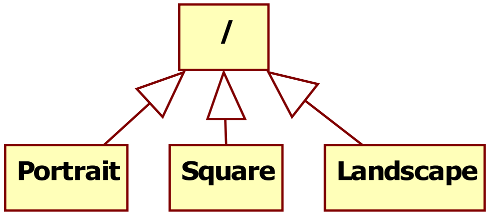
Layout Manager Example

- 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



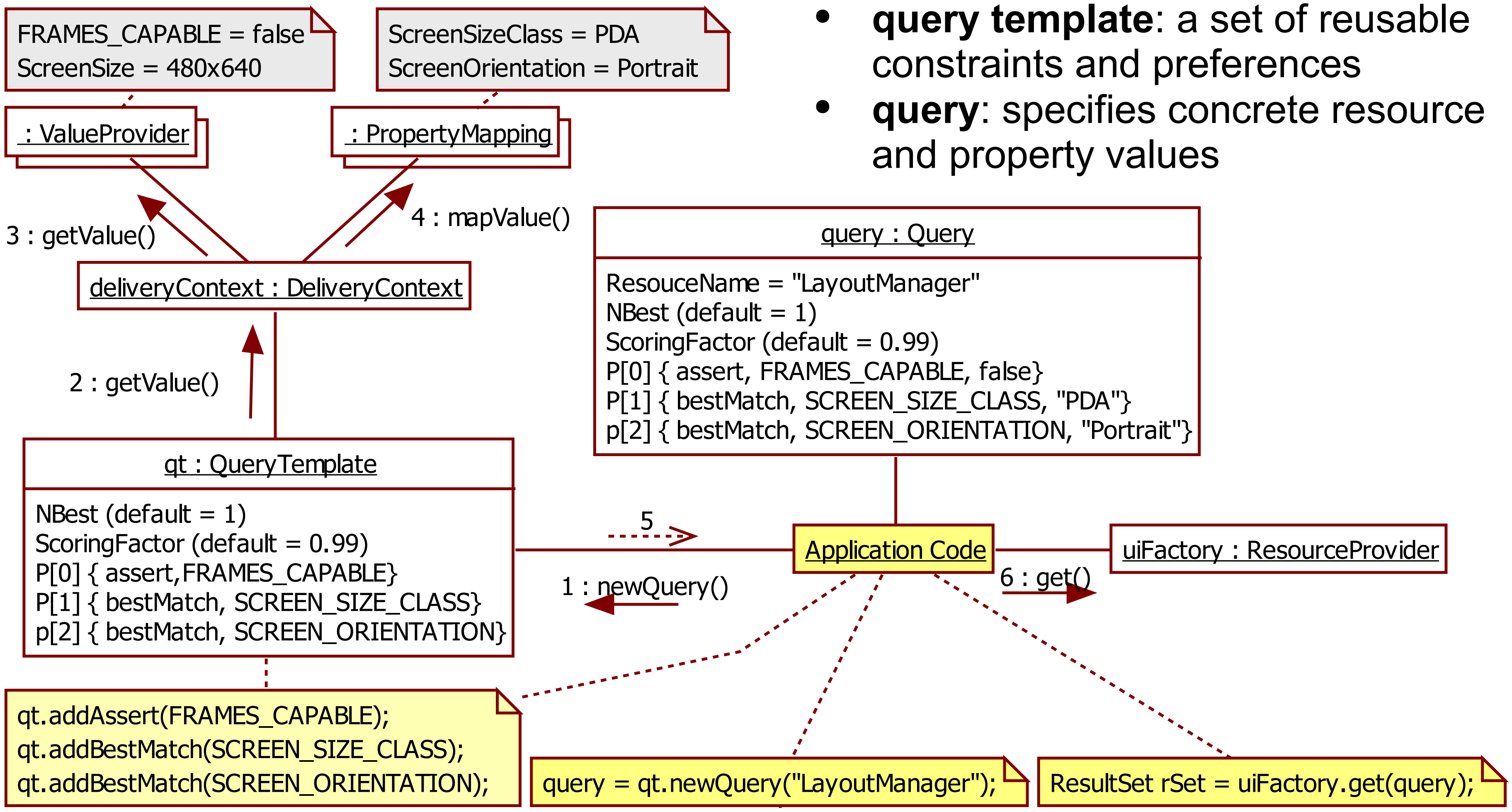
SCREEN_SIZE_CLASS



SCREEN_ORIENTATION

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

: ValueProvider

: PropertyMapping

3 : getValue()

4 : mapValue()

deliveryContext : DeliveryContext

2 : getValue()

query : Query

ResouceName = "LayoutManager"
NBest (default = 1)
ScoringFactor (default = 0.99)
P[0] { assert, FRAMES_CAPABLE, false}
P[1] { bestMatch, SCREEN_SIZE_CLASS, "PDA"}
p[2] { bestMatch, SCREEN_ORIENTATION, "Portrait"}

qt : QueryTemplate

NBest (default = 1)
ScoringFactor (default = 0.99)
P[0] { assert,FRAMES_CAPABLE}
P[1] { bestMatch, SCREEN_SIZE_CLASS}
p[2] { bestMatch, SCREEN_ORIENTATION}

5

1 : newQuery()

Application Code

6 : get()

uiFactory : ResourceProvider

```
qt.addAssert(FRAMES_CAPABLE);
qt.addBestMatch(SCREEN_SIZE_CLASS);
qt.addBestMatch(SCREEN_ORIENTATION);
```

```
query = qt.newQuery("LayoutManager");
```

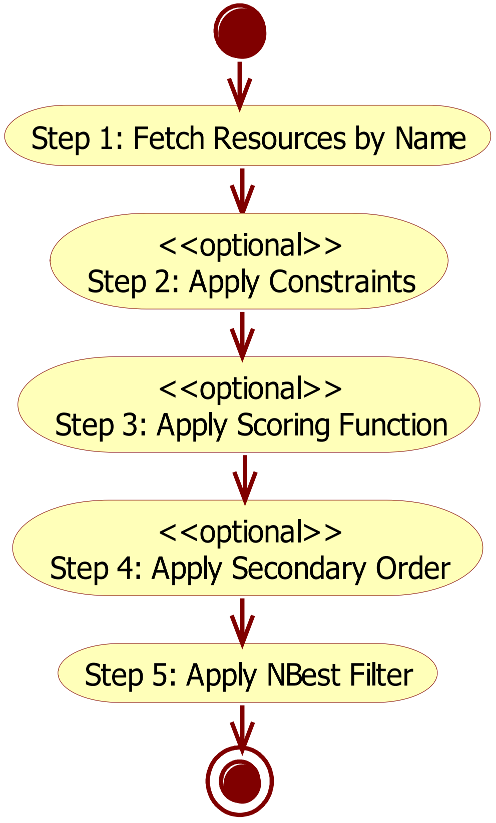
```
ResultSet rSet = uiFactory.get(query);
```

- 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

query : Query

ResouceName = "LayoutManager"
NBest (default = 1)
ScoringFactor (default = 0.99)
P[0] { assert, FRAMES_CAPABLE, false}
P[1] { bestMatch, SCREEN_SIZE_CLASS, "PDA"}
p[2] { bestMatch, SCREEN_ORIENTATION, "Portrait"}



Step 1: Fetch Resources by Name
<<optional>> Step 2: Apply Constraints
<<optional>> Step 3: Apply Scoring Function
<<optional>> Step 4: Apply Secondary Order
Step 5: Apply NBest Filter

- 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

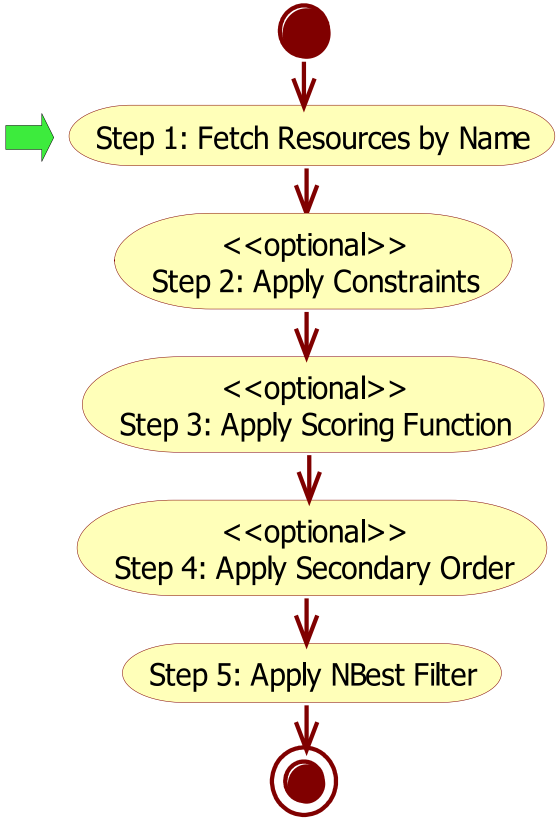
UID	ResourceName	FramesCapable	ScreenSizeClass	ScreenOrientation
1	MainMenu	TRUE	Tablet	Landscape
2	ToolBar	TRUE	PersonalComputer	Landscape
3	ContextMenu	FALSE	CellPhone	Portrait
4	LayoutManager	TRUE	PDA	/
5	LayoutManager	FALSE	PDA	/
6	LayoutManager	FALSE	CompactDevice	Portrait
7	LayoutManager	FALSE	PDA	Landscape

Table 1: A resource repository example

Part II – Resource Provider

query : Query

ResouceName = "LayoutManager"
NBest (default = 1)
ScoringFactor (default = 0.99)
P[0] { assert, FRAMES_CAPABLE, false}
P[1] { bestMatch, SCREEN_SIZE_CLASS, "PDA"}
p[2] { bestMatch, SCREEN_ORIENTATION, "Portrait"}



Step 1: Fetch Resources by Name
<<optional>>
Step 2: Apply Constraints
<<optional>>
Step 3: Apply Scoring Function
<<optional>>
Step 4: Apply Secondary Order
Step 5: Apply NBest Filter

- 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

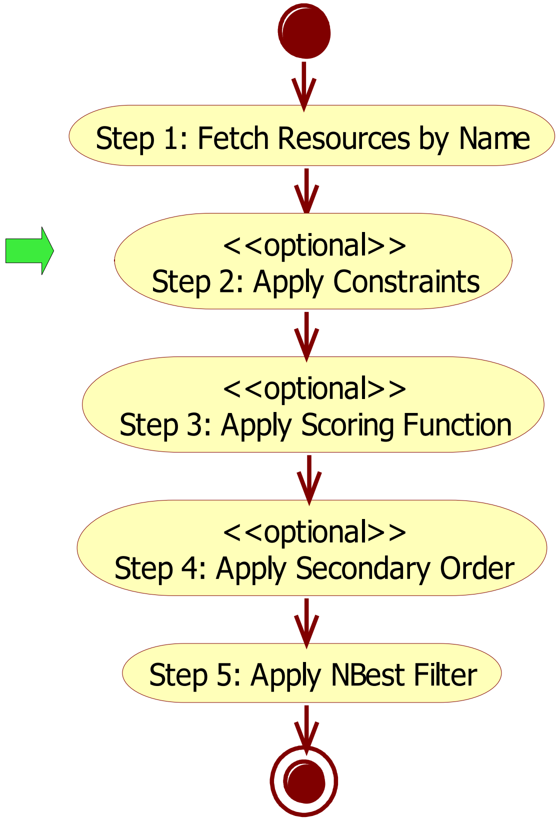
UID	ResourceName	FramesCapable	ScreenSizeClass	ScreenOrientation
~~1~~	~~MainMenu~~ 🚫	~~TRUE~~	~~Tablet~~	~~Landscape~~
~~2~~	~~ToolBar~~ 🚫	~~TRUE~~	~~PersonalComputer~~	~~Landscape~~
~~3~~	~~ContextMenu~~ 🚫	~~FALSE~~	~~CellPhone~~	~~Portrait~~
4	LayoutManager	TRUE	PDA	/
5	LayoutManager	FALSE	PDA	/
6	LayoutManager	FALSE	CompactDevice	Portrait
7	LayoutManager	FALSE	PDA	Landscape

Table 1: A resource repository example

Part II – Resource Provider

query : Query

ResouceName = "LayoutManager"
NBest (default = 1)
ScoringFactor (default = 0.99)
P[0] { assert, FRAMES_CAPABLE, false}
P[1] { bestMatch, SCREEN_SIZE_CLASS, "PDA"}
p[2] { bestMatch, SCREEN_ORIENTATION, "Portrait"}



Step 1: Fetch Resources by Name

<<optional>>
Step 2: Apply Constraints

<<optional>>
Step 3: Apply Scoring Function

<<optional>>
Step 4: Apply Secondary Order

Step 5: Apply NBest Filter

- 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

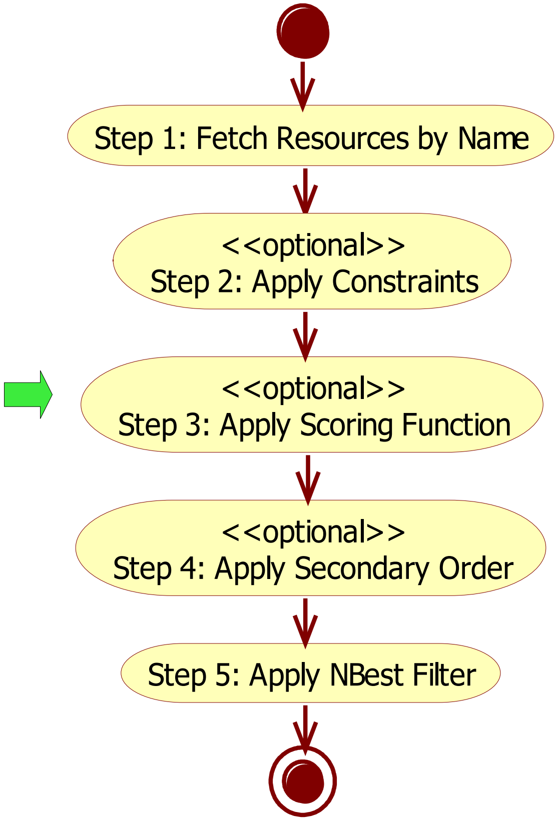
UID	ResourceName	FramesCapable	ScreenSizeClass	ScreenOrientation
~~1~~	~~MainMenu~~ 🚫	~~TRUE~~	~~Tablet~~	~~Landscape~~
~~2~~	~~ToolBar~~ 🚫	~~TRUE~~	~~PersonalComputer~~	~~Landscape~~
~~3~~	~~ContextMenu~~ 🚫	~~FALSE~~	~~CellPhone~~	~~Portrait~~
~~4~~	~~LayoutManager~~	~~TRUE~~ 🚫	~~PDA~~	~~/~~
5	LayoutManager	FALSE	PDA	/
6	LayoutManager	FALSE	CompactDevice	Portrait
~~7~~	~~LayoutManager~~	~~FALSE~~	~~PDA~~	~~Landscape~~ 🚫

Table 1: A resource repository example

Part II – Resource Provider

query : Query

ResouceName = "LayoutManager"
NBest (default = 1)
ScoringFactor (default = 0.99) ⬅
P[0] { assert, FRAMES_CAPABLE, false}
P[1] { bestMatch, SCREEN_SIZE_CLASS, "PDA"}
p[2] { bestMatch, SCREEN_ORIENTATION, "Portrait"}


Step 1: Fetch Resources by Name
<<optional>>
Step 2: Apply Constraints
<<optional>>
Step 3: Apply Scoring Function
<<optional>>
Step 4: Apply Secondary Order
Step 5: Apply NBest Filter

- 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

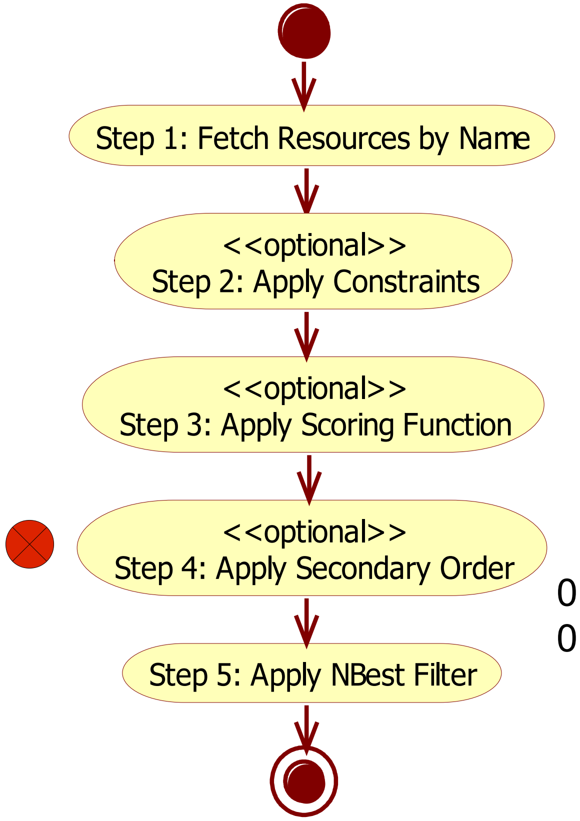
	UID	ResourceName	FramesCapable	ScreenSizeClass	ScreenOrientation
	~~1~~	~~MainMenu~~ 🚫	~~TRUE~~	~~Tablet~~	~~Landscape~~
	~~2~~	~~ToolBar~~ 🚫	~~TRUE~~	~~PersonalComputer~~	~~Landscape~~
	~~3~~	~~ContextMenu~~ 🚫	~~FALSE~~	~~CellPhone~~	~~Portrait~~
	~~4~~	~~LayoutManager~~	~~TRUE~~ 🚫	~~PDA~~	~~/~~
0.505	5	LayoutManager	FALSE 0	PDA 0	/ 1
0.502	6	LayoutManager	FALSE 0	CompactDevice 1	Portrait 0
	~~7~~	~~LayoutManager~~	~~FALSE~~	~~PDA~~	~~Landscape~~ 🚫

Table 1: A resource repository example

Part II – Resource Provider

query : Query

ResouceName = "LayoutManager"
NBest (default = 1)
ScoringFactor (default = 0.99)
P[0] { assert, FRAMES_CAPABLE, false}
P[1] { bestMatch, SCREEN_SIZE_CLASS, "PDA"}
p[2] { bestMatch, SCREEN_ORIENTATION, "Portrait"}


Step 1: Fetch Resources by Name
<<optional>>
Step 2: Apply Constraints
<<optional>>
Step 3: Apply Scoring Function
<<optional>>
Step 4: Apply Secondary Order
Step 5: Apply NBest Filter

- 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

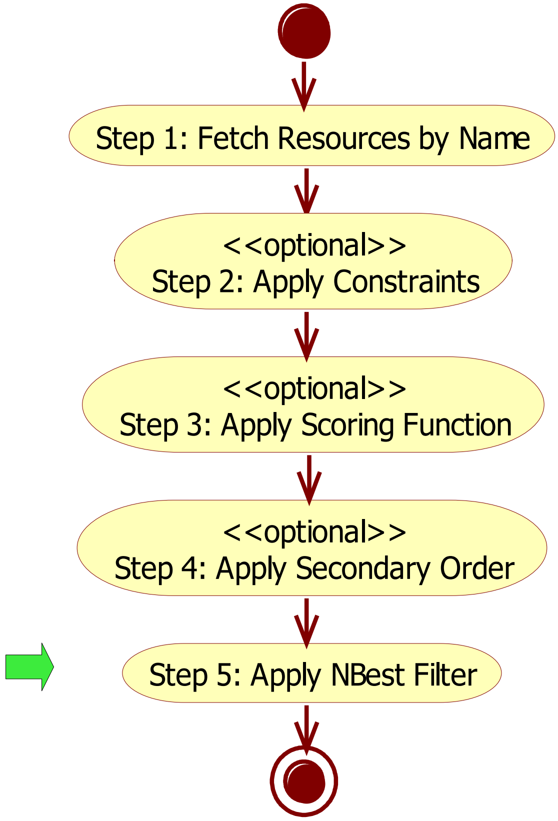
	UID	ResourceName	FramesCapable	ScreenSizeClass	ScreenOrientation
	~~1~~	~~MainMenu~~ 🚫	~~TRUE~~	~~Tablet~~	~~Landscape~~
	~~2~~	~~ToolBar~~ 🚫	~~TRUE~~	~~PersonalComputer~~	~~Landscape~~
	~~3~~	~~ContextMenu~~ 🚫	~~FALSE~~	~~CellPhone~~	~~Portrait~~
	~~4~~	~~LayoutManager~~	~~TRUE~~ 🚫	~~PDA~~	~~/~~
0.505	5	LayoutManager	FALSE 0	PDA 0	/ 1
0.502	6	LayoutManager	FALSE 0	CompactDevice 1	Portrait 0
	~~7~~	~~LayoutManager~~	~~FALSE~~	~~PDA~~	~~Landscape~~ 🚫

Table 1: A resource repository example

Part II – Resource Provider

query : Query

ResouceName = "LayoutManager"
NBest (default = 1)
ScoringFactor (default = 0.99)
P[0] { assert, FRAMES_CAPABLE, false}
P[1] { bestMatch, SCREEN_SIZE_CLASS, "PDA"}
p[2] { bestMatch, SCREEN_ORIENTATION, "Portrait"}


Step 1: Fetch Resources by Name
<<optional>>
Step 2: Apply Constraints
<<optional>>
Step 3: Apply Scoring Function
<<optional>>
Step 4: Apply Secondary Order
Step 5: Apply NBest Filter

- 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

	UID	ResourceName	FramesCapable	ScreenSizeClass	ScreenOrientation
	~~1~~	~~MainMenu~~ 🚫	~~TRUE~~	~~Tablet~~	~~Landscape~~
	~~2~~	~~ToolBar~~ 🚫	~~TRUE~~	~~PersonalComputer~~	~~Landscape~~
	~~3~~	~~ContextMenu~~ 🚫	~~FALSE~~	~~CellPhone~~	~~Portrait~~
	~~4~~	~~LayoutManager~~	~~TRUE~~ 🚫	~~PDA~~	~~/~~
0.505	5	LayoutManager	FALSE 0	PDA 0	/ 1
0.502 🚫	~~6~~	~~LayoutManager~~	~~FALSE~~ 0	~~CompactDevice~~ 1	~~Portrait~~ 0
	~~7~~	~~LayoutManager~~	~~FALSE~~	~~PDA~~	~~Landscape~~ 🚫

Table 1: A resource repository example

Evaluation / Related Work

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

++ 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

🟪 [1] J Gergic: *A Versioning Model for SOFA/DCUP Architecture*, Master's Thesis, Dept. of SW Engineering, Charles University, Prague,1999

Refereed Papers

🟪 [2] J Gergic: *Towards a Versioning Model for Component-based Software Assembly*, In proceedings of 19th International Conference on Software Maintenance (ICSM 2003), Amsterdam, Netherlands, 22-26 September 2003, pages 138 - 147, IEEE Computer Society, 2003

🟦 [3] V. Demesticha, J.Gergic, J.Kleindienst, M. Mast, L.Polymenakos,H.Schulz, L.Seredi: *Aspects of design and implementation of multi-channel and multi-modal information system*, In proceedings of 2001 International Conference on Software Maintenance (ICSM 2001), Florence, Italy, pages 312 - 319, IEEE Computer Society, 2001

🟦 [4] Yannis Despotopoulos, George Patikis, John Soldatos, LazarosPolymenakos, Jan Kleidienst, Jaroslav Gergic: *Accessing and Transforming Dynamic Content based on XML: Alternative Techniques anda Practical Implementation*, In proceedings of 3rd International Conference on Information Integration and Web-based Applications and Services (IIWAS 2001), pages 162 - 164, Linz, Austria, Sep 2001

🟦 [5] J. Gergic, J. Kleindienst, Y. Despotopoulos, J. Soldatos, G. Patikis, A. Anagnostou, L. Polymenakos: *An Approach to Lightweight Deployment of Web Services*, In proceedings of 14th International Conference on Software Engineering and Knowledge Engineering (SEKE '02), pages 635 - 640, Ischia (Italy), ACM Press New York, NY, USA, 2002

International Standards

🟥 [6] J. Engelsma, C. Cross, J. Gergic, R. Hosn, T. Ling, C. Wiecha, M., Pearce, R. Chaudhri, J. Ferrans, P. Baggia, A. Wahbe: *Distributed Multimodal Synchronization Protocol*, Internet Engineering Task Force (IETF), Internet-Draft, 2005-2007, http://www.ietf.org/internet-drafts/draft-engelsma-dmsp-04.txt (4th revision)

Patent Applications

🟥 [7] Hosn Rafah A., Gergic Jaroslav, Ling Nai Keung Thomas, Wiecha Charles: *System for factoring synchronization strategies from multimodal programming model runtimes*, United States Patent Application #20060036770, U.S. Patent & Trademark Office, 2006

🟦 [8] Gergic Jaroslav, Kleindienst Jan, Maes Stephane H., Raman Thiruvilwamalai V., Sedivy Jan: *Systems and methods for providing conversational computing via javaserver pages and javabeans*, United States Patent Application #20030046316, U.S. Patent & Trademark Office, 2003

🟦 [9] Gergic Jaroslav, Hosn Rafah A., Kleindienst Jan, Maes Stephane H., Raman Thiruvilwamalai V., Sedivy Jan, Seredi Ladislav: *Reusable voiceXML dialog components, subdialogs and beans*, United States Patent Application #20020198719, U.S. Patent & Trademark Office, 2002

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/