User Profile Integration Made Easy – Model-Driven Extraction and Transformation of Social Network Schemas


MultiA-Pro Workshop, WWW Conference, Lyon, France, April 16th, 2012
Motivation

Social Networks & User Profile Data

Private Networks
- Facebook
- Google+
- StudiVZ

Professional Networks
- LinkedIn
- XING

Music
- Last.fm
- Myspace.com

Messaging & Sharing
- Twitter
- Del.icio.us

Martin Wischenbart, Vienna University of Technology, April 16th 2012, MultiA-Pro Workshop 2012
Motivation

Social Networks & User Profile Data

**Private Networks**
- identity
- group memberships
- location
- applications
- ...

**Professional Networks**
- education/work history
- job interests
- skills & languages
- ...

**Music**
- genres
- artists
- songs
- ...

**Messaging & Sharing**
- communication behavior
- hobbies
- interests
- beliefs
- ...

---

Martin Wischenbart, Vienna University of Technology, April 16th 2012, MultiA-Pro Workshop 2012
Needed for **various data processing tasks:**

- Search
- Manipulation
- Optimization
- Translation
- Evolution
- Integration

→ integrate user **data from multiple social networks** to achieve comprehensive profiles for **recommender** applications
Social networks often use NoSQL-DBs

- **No traditional schema** (e.g., HBase in Hadoop)
- **Schema-less** multidimensional maps (e.g., Apache Cassandra)

→ **Explicit schemas not available**

RESTful social network APIs:

- Leading data format: **JSON** (supported by all surveyed social networks)
- Authentication: OAuth 1.0a/2.0

→ **Documentation (schema description) is often exemplary or outdated**

Evolution of social networks and APIs

- Requires **adaptation of existing** schemas and applications

→ **Repetitive manual creation of up-to-date schemas is not an option**
• Semi-automatically **derive schemas** from instance data

• **Strategies** to handle **specifics** of social networks and **JSON** (JSON Schema)

• **Transformation** to different technical spaces (ECORE, XML Schema, OWL)

→ **Application** of **existing integration tools**

• Evaluate approach with **Facebook, Google+, LinkedIn**
Outline

- Related Work
- JSON Data & Schema
- Approach
  - Schema Extraction (3 steps)
  - Transformation
- Results & Evaluation
  - Test User Profile Setup
  - Extracted Schemas
  - Comparison to Documentation
  - Outlook on Integration
- Conclusions
**Related Work**

Approaches for Automatic Schema Extraction

- **No focus** on JSON Schema so far

- **Generation of DTDs or XML Schemas**  
  *(Bex et al. [4], Eki et al. [7], Hegewald et al. [10], Mylnkova [18] (survey))*
  - Must use XML APIs or transform instances before
  - **Specificity** of extracted schemas
  - No **configurability** to social network APIs, does **not consider peculiarities of social networks**

- **Ontology learning**  
  *(Drumond et al. [6] (survey), Hazman et al. [8] (survey))*
  - Focus on concepts and taxonomic relationships → **disregard non-taxonomic relationships** (i.e., references between classes)

- **Meta-models from models** *(Javed et al. [11])*  
  - Evolution requires high number of transformations and grammars
  - **Not flexible and reusable** enough

→ **Existing approaches not applicable in social network integration scenario**
**JSON Data & Schema**

**Example of Extracted Data**

```
{
    "id": "100002345678964",
    "name": "Jane Doe",
    "birthday": "04/18/1978",
    "gender": "female",
    "type": "user",
    "work": [
        {
            "employer": {
                "id": "106119876543210",
                "name": "Doe Inc."
            },
            "start_date": "2007-08"
        },
        {
            "start_date": "2004",
            "end_date": "2007"
        }
    ]
}
```

```
{
    "id": "106119876543210",
    "name": "Doe Inc.",
    "picture": "http://www.doe.net/logo.jpg",
    "link": "http://www.facebook.com/doeinc",
    "likes": 25946937,
    "category": "Food/beverages",
    "username": "doeinc",
    "founded": "April 1st. Seriously.",
    "company_overview": "Doe Power Drink is a functional beverage. Thanks to a unique combination of high quality ingredients Doe Power Drink vitalizes body and mind. Doe Power Drink has been developed for people who want to have a clear and focused mind, perform physically, are dynamic and performance-oriented whilst also balancing this with a fun and active lifestyle. In short, Doe Power Drink gives wings to people who want to be mentally and physically active and have a zest for life."
}
```
JSON Data & Schema

JSON Schema Example

conforms to

```
{
    "id": "100002345678964",
    "name": "Jane Doe",
    "birthday": "04/18/1978",
    "gender": "female",
    "type": "user",
    "work": [
        {
            "employer": {
                "id": "106119876543210",
                "name": "Doe Inc."
            },
            "start_date": "2007-08"
        },
        {
            "start_date": "2004",
            "end_date": "2007"
        }
    ]
}
```

```
{
    "type": "object",
    "id": "user",
    "properties": {
        "id": { "type": "string" },
        "name": { "type": "string" },
        "birthday": { "type": "string", "pattern": 
            "[0-9]{2}/[0-9]{2}/[0-9]{4}" },
        "gender": { "type": "string", "enum": ["male", "female"] },
        "type": { "type": "string" },
        "work": {
            "type": "array",
            "items": [
                {
                    "type": "object",
                    "properties": {
                        "employer": {
                            "type": "object",
                            "id": "employer",
                            "properties": {
                                "id": { "type": "string" },
                                "name": { "type": "string" }
                            }
                        },
                        "start_date": { "type": "string" }
                    }
                },
                {
                    "start_date": { "type": "string" },
                    "end_date": { "type": "string" }
                }
            ]
        }
    }
}
```
**Approach**

**Four Phases to Integration – Overview**

1. **Data Extraction:** extract instance data from social networks via APIs (JSON)
2. **Schema Extraction:** derive separate schemas for each social network, corresponding to a schema language (JSON Schema)
3. **Transformation:** transform to different technical space (XML Schema/XML)
4. **Integration:** integrate (integrated XML Schema/XML)

---

**Data Extraction:**
- Graph API
- Facebook API
- LinkedIn API
- Google+ API

**Schema Extraction:**
- Ls1: FB Schema
- Ls2: LinkedIn Schema
- Ls3: Google+ Schema

**Transformation:**
- Lx1: JSON
- Lx2: LinkedIn
- Lx3: Google+

**Integration:**
- Ls*: Integrated Schema
- Ls*: Schema Language X
  - e.g., XML Schema

---

**Existing Integration Tools**
- MapForce
- Enterprise Architect
- Comma
Approach
Four Phases to Integration – Overview

1. **Data Extraction**: extract instance data from social networks via APIs (JSON)

2. **Schema Extraction**: derive separate schemas for each social network, corresponding to a schema language (JSON Schema)

3. **Transformation**: transform to different technical space (XML Schema/XML)

4. **Integration**: integrate (integrated XML Schema/XML)

### Phase 2: Schema Extraction

1. Generalization Strategies
2. Merging and Clearance
3. Refactoring

### Phase 3: Transformation

- $L_{s1}$: FB Schema
- $L_{s2}$: LinkedIn Schema
- $L_{s3}$: Google+ Schema

$L_{s1}$, $L_{s2}$, $L_{s3}$ conform to $L_{s}':$ Integrated Schema

$L_{x1}$, $L_{x2}$, $L_{x3}$ conform to $L_{s}':$ JSON Schema

$L_{x1}$, $L_{x2}$, $L_{x3}$ conform to $L_{s}':$ e.g., XML Schema

$L_{s}':$ Schema Language X (e.g., XML Schema)

$L_{s1}':$ FB Schema
$L_{s2}':$ LinkedIn Schema
$L_{s3}':$ Google+ Schema

$L_{x1}':$ e.g., XML
$L_{x2}':$ e.g., XML
$L_{x3}':$ e.g., XML
### Approach

#### Phase 2: Schema Extraction – 3 Steps

1. **Generalization Strategies**
   - Create **schema parts** from instances
   - Introduce **links** between schema parts

2. **Merging and Clearance**
   - Merge linked schema **parts** into single schema
   - Clear duplicate schema parts (merge into coherent schema)

3. **Refactoring**
   - Build **class hierarchy**
   - Homogenize array types
   - Lookup class names in ontology
### Approach

**Phase 2-1: Generalization Strategies**

- **Different alternative strategies available**
  - **Configure once per social network**

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Configuration Options</th>
<th>Priority</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TypeFromObject</td>
<td></td>
<td></td>
<td></td>
<td>derives a type for each object</td>
</tr>
<tr>
<td>IdFromValue</td>
<td>names of keys</td>
<td>1</td>
<td></td>
<td>derives the name of a type from the value of a property</td>
</tr>
<tr>
<td>IdFromReferenceName</td>
<td></td>
<td>2</td>
<td></td>
<td>derives the name of a nested type from the reference name</td>
</tr>
<tr>
<td>IdFromNameConcat</td>
<td></td>
<td>3</td>
<td></td>
<td>derives the name of the type by concatenating the names of the contained properties</td>
</tr>
<tr>
<td>PropertyFromKeyValuePair</td>
<td></td>
<td></td>
<td></td>
<td>derives a property for each key/value pair</td>
</tr>
<tr>
<td>NameFromProperty</td>
<td></td>
<td></td>
<td></td>
<td>derives the name of the property from the key of the key/value pair</td>
</tr>
<tr>
<td>TypeFromPropertyValue</td>
<td></td>
<td></td>
<td></td>
<td>derives the type of the property from the type of the value (String, Boolean, Number, Array, Object)</td>
</tr>
<tr>
<td>EnumFromValue</td>
<td>names of keys</td>
<td>✓</td>
<td></td>
<td>derives an enumeration for the key/value pair</td>
</tr>
<tr>
<td>IntervalFromValue</td>
<td>names of keys</td>
<td>✓</td>
<td></td>
<td>derives an interval for the key/value pair</td>
</tr>
<tr>
<td>LinkFromProperty</td>
<td></td>
<td></td>
<td></td>
<td>derives links between types</td>
</tr>
<tr>
<td>LinkRoleFromName</td>
<td>names of keys</td>
<td></td>
<td></td>
<td>derives the role name of the link from the key of a key/value pair</td>
</tr>
<tr>
<td>LinkPatternFromValue</td>
<td></td>
<td></td>
<td></td>
<td>derives the href of the link from values that are valid URLs</td>
</tr>
</tbody>
</table>

**Example:**

```json
{
  "name": "Jane Doe",
  "type": "user"
}
```

**JSON instance**

```json
{
  "type": "object",
  "id": "user"
}
```

**JSON schema**

---

**Martin Wischenbart, Vienna University of Technology, April 16th 2012, MultiA-Pro Workshop 2012**
• Merge properties of equal types
• Clear duplicates

Example:
• Merge multiple employers to single type

JSON schema
• Merge properties of equal types
• Clear duplicates

Example:
• Merge *multiple employers* to single type
• Clear duplicate employer (replace with reference)
• **Build class hierarchy:**
  – introduce **superclass** for user and employer

• **Homogenize array types**
  – homogenize **types of work** array
Motivation

- Related Work

JSON Data & Schema

- Approach

Results & Evaluation

Conclusions
Transformation to different technical space

- **Mapping** of meta-models
- E.g., from **JSON Schema** to **ECORE**

<table>
<thead>
<tr>
<th>Source concept (JSON)</th>
<th>Target concept (ECORE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>EClass</td>
</tr>
<tr>
<td>Primitive property</td>
<td>EAttribute (with corresponding datatype EDataType)</td>
</tr>
<tr>
<td>Nested type (without link)</td>
<td>EReference (composition with multiplicity 1)</td>
</tr>
<tr>
<td>Nested array (without link)</td>
<td>EReference (composition with unbounded multiplicity)</td>
</tr>
<tr>
<td>Link</td>
<td>EReference (reference with maximum multiplicity 1)</td>
</tr>
<tr>
<td>Array of links</td>
<td>EReference (reference with unbounded multiplicity)</td>
</tr>
</tbody>
</table>

→ from **ECORE** draw **class diagram**
Approach

Phase 3: Transformation Example

```json
{
  "type": "object",
  "id": "user_employer",
  "properties": {
    "id": { "type": "string" },
    "name": { "type": "string" }
  },
},

{
  "type": "object",
  "id": "user",
  "extends": "user_employer"
  "properties": {
    ...
    "work": { "type": "array", "items": [ { "type": "object", "properties": { "employer": { "type": "object", "id": "employer", "extends": "user_employer" }, "start_date": { "type": "string" }, "end_date": { "type": "string" } }, ] } ]
}
```
Manually created test users

- 1 user + connected friend
- For Facebook, Google+, LinkedIn
- Equal properties and activities

Properties
- Name
- Email
- City
- Birthday
- Status update
- Education and work
  - High school & university
  - Current & previous jobs

Activities
- Connect to friend
- Direct communication
- Group conversation
  - Comments
  - Likes
  - Pictures
Extracted schema for Facebook test user profile

Facebook API provides optional meta-information

<table>
<thead>
<tr>
<th>Metric</th>
<th>Facebook</th>
<th>Google+</th>
<th>LinkedIn</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Classes</td>
<td>58</td>
<td>25</td>
<td>34</td>
</tr>
<tr>
<td>Number of Properties</td>
<td>269</td>
<td>71</td>
<td>75</td>
</tr>
<tr>
<td>Number of References</td>
<td>93</td>
<td>23</td>
<td>58</td>
</tr>
</tbody>
</table>
• Carefully analyzed **documentation** of User/Person (properties and references)
• Compared to **extracted data** from API (instances & created schema)

<table>
<thead>
<tr>
<th>Source</th>
<th>Facebook</th>
<th>Google+</th>
<th>LinkedIn</th>
</tr>
</thead>
<tbody>
<tr>
<td>API documentation of user</td>
<td>71</td>
<td>45</td>
<td>60</td>
</tr>
<tr>
<td>Subset expected for test user</td>
<td>24</td>
<td>27</td>
<td>24</td>
</tr>
<tr>
<td>Successfully extracted</td>
<td>30</td>
<td>20</td>
<td>29</td>
</tr>
<tr>
<td>Intersection of expected &amp; extracted</td>
<td>19</td>
<td>18</td>
<td>23</td>
</tr>
<tr>
<td><strong>Documented but missing</strong></td>
<td>5</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td><strong>Not documented</strong></td>
<td>11</td>
<td>2</td>
<td>6</td>
</tr>
</tbody>
</table>
Schema excerpt for Facebook

Multiple requests to get all results

>190 types in 1 class
Results & Evaluation
Extracted Schemas: LinkedIn

Schema excerpt for LinkedIn

5 references to same data structure

No dashes in ECORE
Results & Evaluation
Extracted Schemas: Google+

Schema excerpt for **Google+**

- **urls**
  - placeLived
    - 0..1
  - 0..*
- **image**
  - 0..1
- **name**
  - 0..1

**person**
- kind : EString
- etag : EString
- id : EString
- displayName : EString
- gender : EString
- url : EString

**Birthday?**

**APIs do not comply to documentation**
Compared schemas of Google+, Facebook, LinkedIn (user and address only)

<table>
<thead>
<tr>
<th>User</th>
<th>Google+</th>
<th>Facebook</th>
<th>LinkedIn</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>Person.displayName</td>
<td>User.name</td>
<td>(not available)</td>
</tr>
<tr>
<td>firstname</td>
<td>Name.givenName</td>
<td>User.first_name</td>
<td>Person.firstName</td>
</tr>
<tr>
<td>lastname</td>
<td>Name.familyName</td>
<td>User.last_name</td>
<td>Person.lastName</td>
</tr>
<tr>
<td>gender</td>
<td>Person.gender</td>
<td>User.gender</td>
<td>(not available)</td>
</tr>
<tr>
<td>date of birth</td>
<td>(not available)</td>
<td>User.birthday</td>
<td>Person.dateOfBirth</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>PlacesLived.value</th>
<th>Location.zip</th>
<th>(not available)</th>
</tr>
</thead>
<tbody>
<tr>
<td>zip</td>
<td></td>
<td>Location.city</td>
<td>Location.name</td>
</tr>
<tr>
<td>city</td>
<td></td>
<td>Location.country</td>
<td>Location.country</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User</th>
<th>Google+ vs. Facebook</th>
<th>Google+ vs. LinkedIn</th>
<th>Facebook vs. LinkedIn</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>0.55</td>
<td>(not applicable)</td>
<td>(not applicable)</td>
</tr>
<tr>
<td>firstname</td>
<td>0.48</td>
<td>0.41</td>
<td>0.73</td>
</tr>
<tr>
<td>lastname</td>
<td>0.46</td>
<td>0.39</td>
<td>0.73</td>
</tr>
<tr>
<td>gender</td>
<td>0.74</td>
<td>(not applicable)</td>
<td>(not applicable)</td>
</tr>
<tr>
<td>date of birth</td>
<td>(not applicable)</td>
<td>(not applicable)</td>
<td>0.34</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>Google+ vs. LinkedIn</th>
<th>Facebook vs. LinkedIn</th>
</tr>
</thead>
<tbody>
<tr>
<td>zip</td>
<td>(not applicable)</td>
<td>(not applicable)</td>
</tr>
<tr>
<td>city</td>
<td>0.2</td>
<td>0.33</td>
</tr>
<tr>
<td>country</td>
<td>0.2</td>
<td>0.17</td>
</tr>
</tbody>
</table>

Average: 0.33 0.16 0.37

Structural and semantic heterogeneities
- Missing information
- Naming differences
- Fine- and coarse-grained cardinality differences

Schema matching tool COMA++
- First indicator for similarity of social network schemas
Approach depends on availability of user data
- Complete schemas require complete real/pseudo profiles

Schema extraction requires manual intervention
- Facebook: meta-information to automate process
- Google+ & LinkedIn: links from documentation

Different views on same objects require schema merging
- Merging of potentially many view classes

Differences in support for query restriction
- LinkedIn: requested information only (i.e., no "SELECT *" possible)

Nested objects must have ID to be reusable
- Facebook: anonymous work elements, pages for years

Heterogeneous arrays in JSON Schema
- Not representable in every technical space (e.g., UML)
• **Transformation of instances**
  – Prerequisite for **integration**
  – Automatically **derive instance transformation rules** from **schema transformations**

• **Deal with incomplete and unstable interfaces**
  – Create **request code** for data extraction
  – Build **dynamic self-evolving social network adaptors**

• **Co-evolution of extraction and integration applications**
  – Derive **integration rules** from schema mappings
  – **Meta models for change** of schemas and dependent artifacts
    → **automatic co-evolution** of request code and integration rules
Thank you!

Questions & Comments?
References


