Evolution and Adaptability of Complex XML Applications

Talk for the KEG Seminar, Faculty of Informatics and Statistics, University of Economics, Prague (8.11.2012)

Irena Holubova (Mlynkova)

XML and Web Engineering **R**esearch **G**roup
Department of Software Engineering
Faculty of Mathematics and Physics
Charles University in Prague
Czech Republic

holubova@ksi.mff.cuni.cz
http://www.ksi.mff.cuni.cz/~holubova/index.html

Department of Software Engineering

- Staff:
  - 9 researchers
  - 4 lecturers
  - 20 PhD students
- Average results per year:
  - 70 papers in refereed proceedings/journals
  - 6 SW prototypes
- Research groups:
  - Similarity Retrieval Research Group
  - Web Semantization Research Group
  - XML and Web Engineering Research Group

XML and Web Engineering Research Group (XRG)

- Head: Prof. RNDr. Jaroslav Pokorny, CSc.
- Researchers:
  - RNDr. Irena Mlynkova, Ph.D.
  - Mgr. Martin Necasky, Ph.D.
  - doc. Ing. Karel Richta, CSc
- PhD students:
  - Mgr. Sobeslav Benda
  - RNDr. Jakub Klimek
  - RNDr. Tomas Knap
  - Mgr. Jakub Maly
  - Mgr. Marek Polak
  - Mgr. Jakub Starka
  - Mgr. Martin Svoboda

Overview of XRG Research Topics

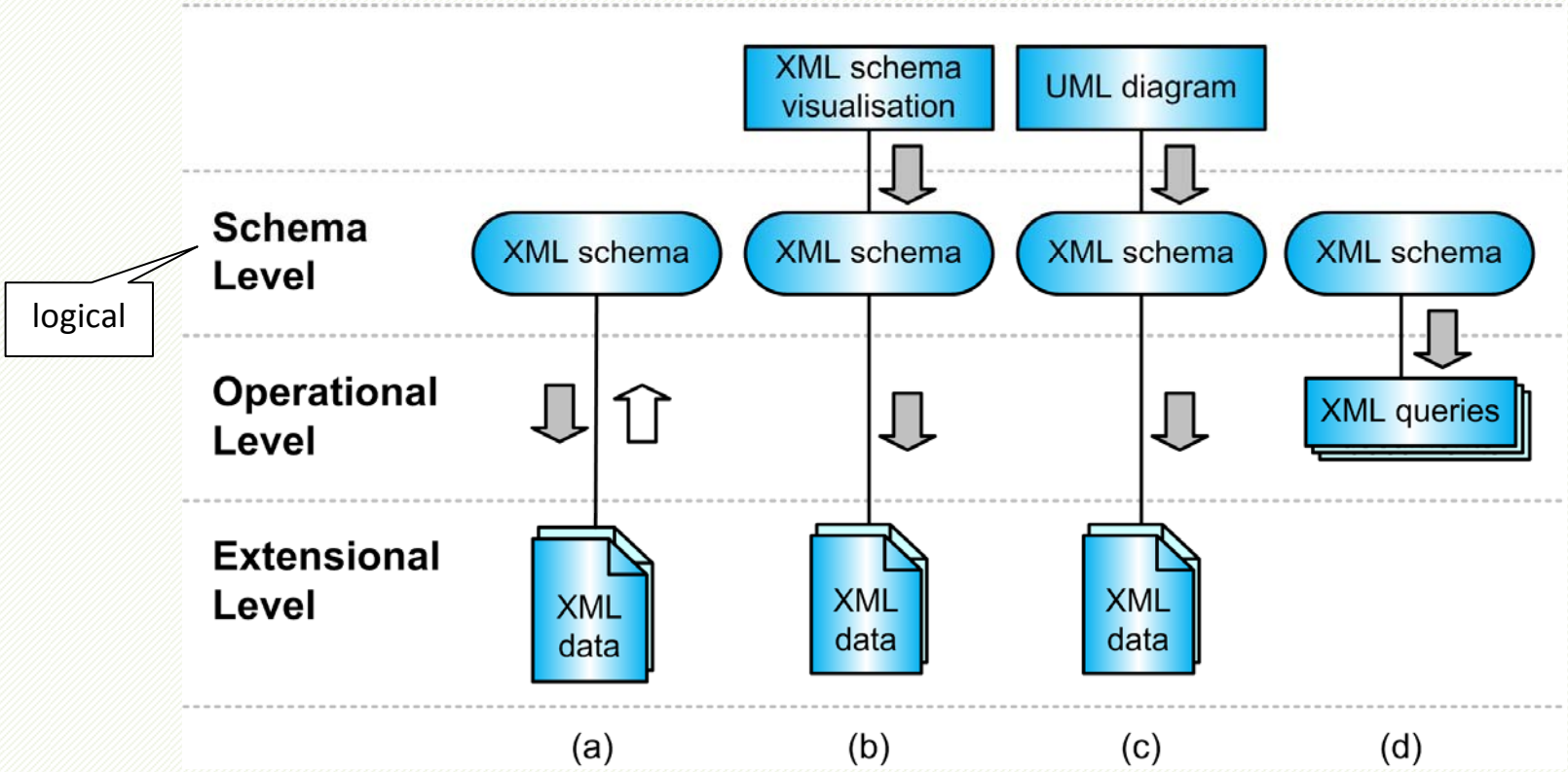
- In general: various aspects of Web technologies
  - Originally (and primarily) XML technologies
  - Gradually extended with regard to current trends
- Current areas:
  - Modelling of XML applications and systems
  - Evolution and change management of applications
  - Inference of XML schemas
  - Similarity of XML data
  - Statistical analysis of real-world data
  - Linked Data, graph databases
  - Data provenance
- Older (but not obsolete) topics:
  - XML benchmarking
  - Storage strategies for XML data

Evolution of XML Applications

- XML application = an application that utilizes XML technologies
  - XML data, XML schemas, XML queries, XSLT scripts, …
- Evolution of application
  - User requirements or surrounding environment changes => the application needs to adapt accordingly
  - Key problem: changes in data structures
  - Many related issues to be solved

Let us see them…

Preliminary Approaches



- Separate issues, simple use cases
  - An XML application can be much complex

XML System of Applications

- An XML application usually involves a family of XML schemas
  - Each involved in particular execution part
  - Example: a set of web services

  => XML system of applications

- Schemas are related, overlap, exploit each other, …
- Problems:
  - The amount of schemas can be high
    - Tens, hundreds, …
  - A change in a single schema may need to be propagated to numerous other schemas

  => Difficult, error-prone, …

Simple Example

- Common problem domain: purchasing goods

```
<custList version="1.3">
 <cust>
  <name>Martin Necasky</name>
  <address>Vaclavske nam. 123, Prague</
address>
  <phone>123 456 789</phone>
 </cust>
 <cust>
  <name>Department of Software Engineering,
   Charles University</name>
  <hq>Malostranske nam. 25, Prague</hq>
  <storage>Ke Karlovu 3, Prague</storage>
  <secretary>Ke Karlovu 5, Prague</secretary>
  <phone>111 222 333</phone>
 </cust>
</custList>
```

List of customers

```
<purchaseRQ version="1.0">
 <cust>
  <name>Department of Software Engineering,
   Charles University</name>
  <code>ksi@mff.cuni.cz</code>
  <bill-to>Malostranske 25, Prague</bill-to>
  <ship-to>Ke Karlovu 3, Prague</ship-to>
 </cust>
 <items>
  <item>
   <code>P045</code><price>17</price>
  </item>
  <item>
   <code>P332</code><price>34</price>
  </item>
 </items>
</purchaseRQ>
```

Purchase requests

- And other schemas
  - Customer details, purchase responses, purchase transport details, …

Example

- The same part of the domain may be represented in different XML schemas in different ways
- Example: Customer
  - In purchase request:
    - name + code, bill-to + ship-to address
  - In list of customers:
    - Private customers: name, address, phone
    - Corporate customers: name, different addresses (headquarters, storage, secretary, …), phone
- New user requirement: Address is not a simple string, but should be divided into street, city, zip code, …

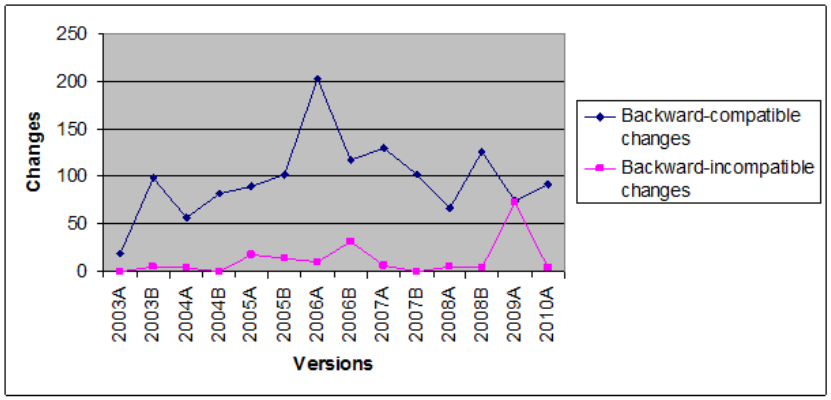
Example

- We need to:
  - Find all the addresses in all the schemas
  - Correctly modify their structure
  - Correctly propagate the modification to all respective documents
    - (and operations, storage strategies, source codes, …)
- The situation can be even more complicated…
  - E.g. We may want to change only addresses that represent a place to ship the goods, not all addresses
    - i.e. not, e.g., the address of the headquarters

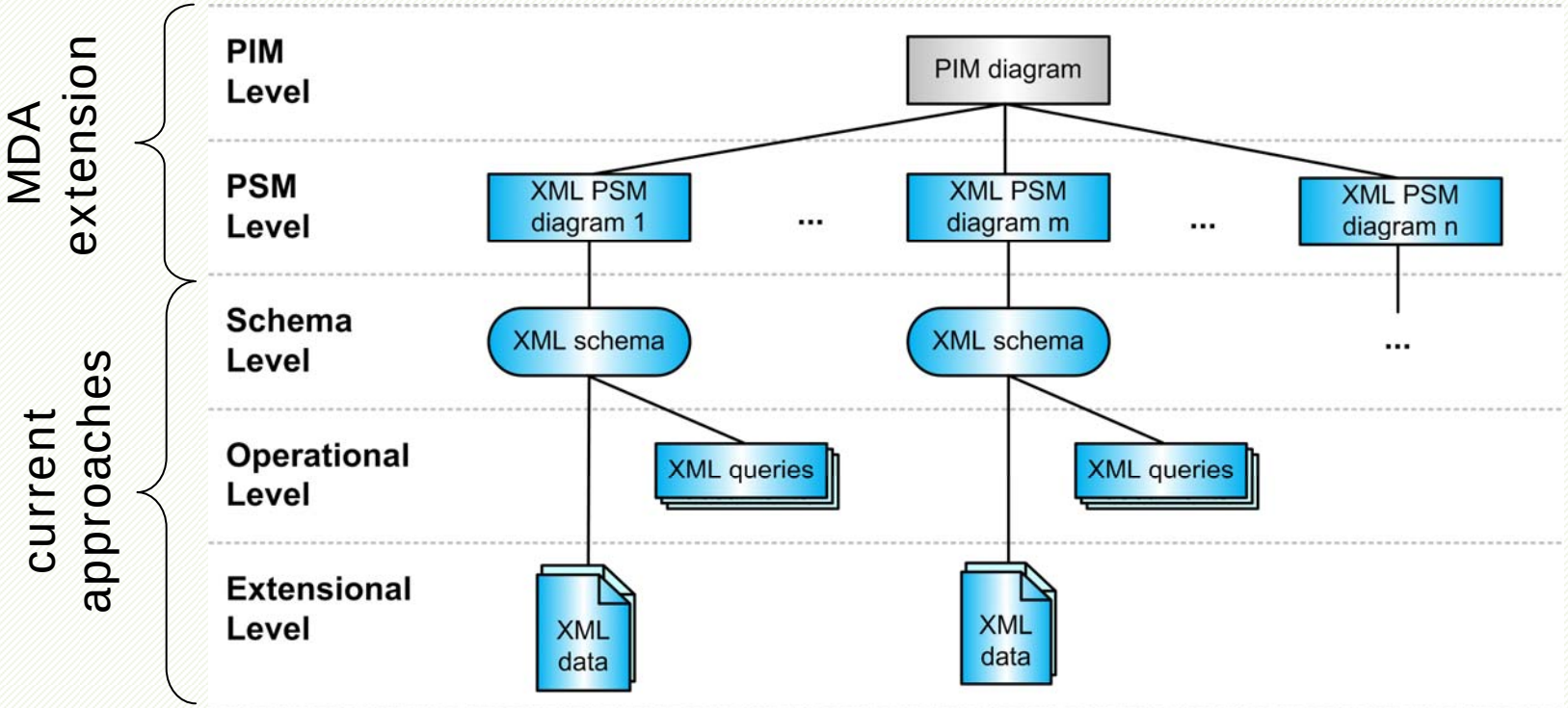
And what if we have hundreds of XML schemas?

More Complex Example: OpenTravel.org

- Standard for communication in travel industry
- Currently: 319 XML schemas
- Evolution:
  - Changed twice a year
  - Changes are published in the form of a new version of XML schemas + documentation of the changes in a form of human-readable document
- Solution: Preserving backward compatibility as much as possible
  - 7,5% on average within years 2003 – 2010
  - Consequence: artificial data structures, plenty of optional items, obsolete data structures, …



Our Solution



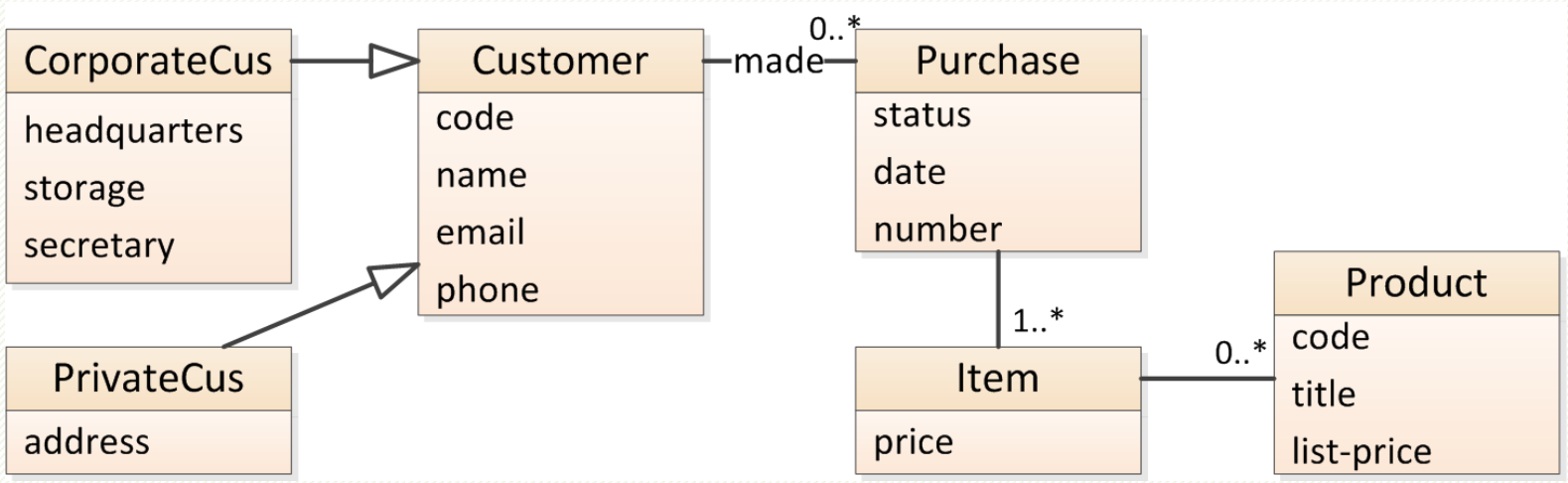
Necasky, M. - Mlynkova, I.: On Different Perspectives of XML Schema Evolution. FlexDBIST '09 workshop of DEXA '09, pages 422 - 426, Linz, Austria, September 2009. IEEE Computer Society Press, 2009.

Five-Level XML Evolution Framework

- Based on the MDA principle
  - Idea: modelling of problem domain at different levels of abstraction
- Levels:
  - Extensional level – XML documents
  - Operational level – XML queries
  - Schema level – XML schemas
  - Platform-independent level – conceptual model of the whole problem domain
    - e.g. purchasing goods
  - Platform-specific level – mapping of the problem domain to particular XML formats
    - e.g. list of customers, purchase requests, …

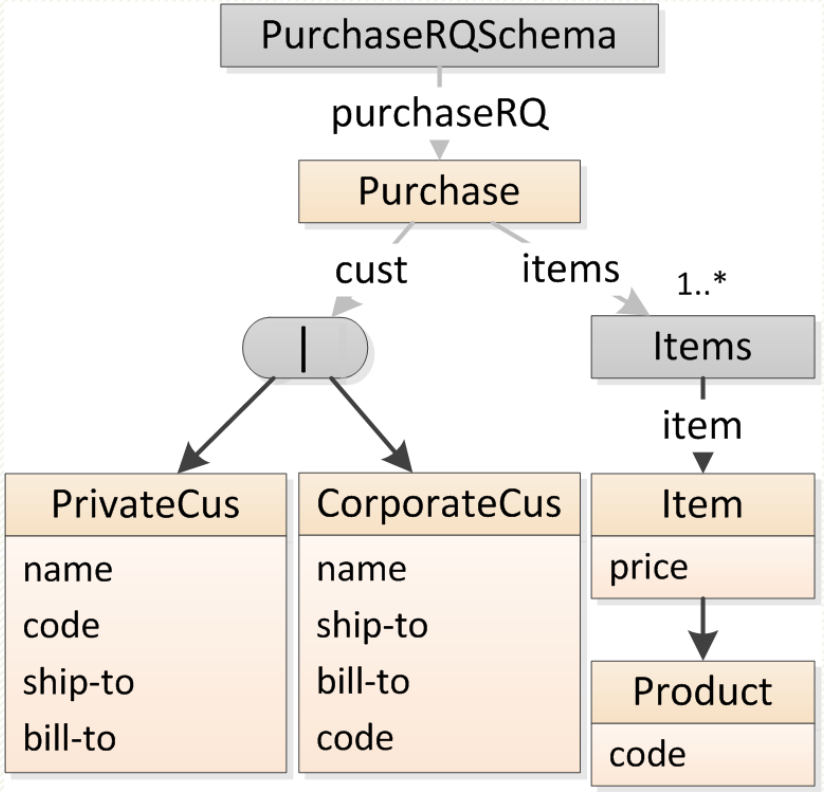
Platform-Independent Model (PIM)

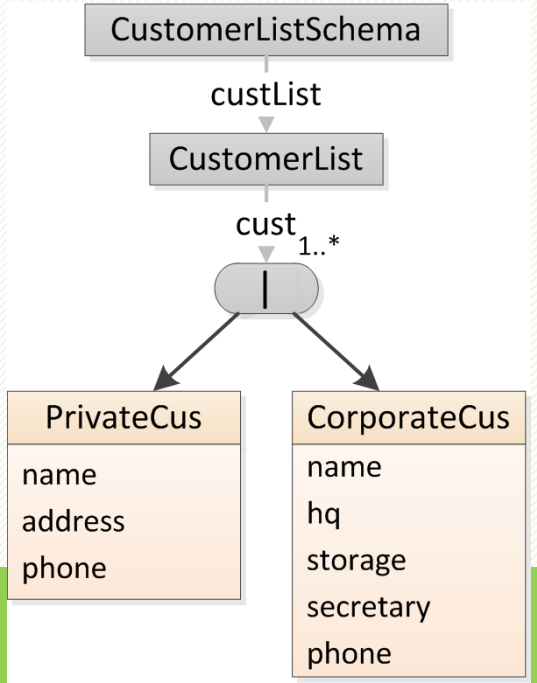
- Models real-world concepts + relationships among them
- Simplified UML class diagram
  - Classes, attributes + data types, binary associations + cardinalities
    - (+ inheritance + …)



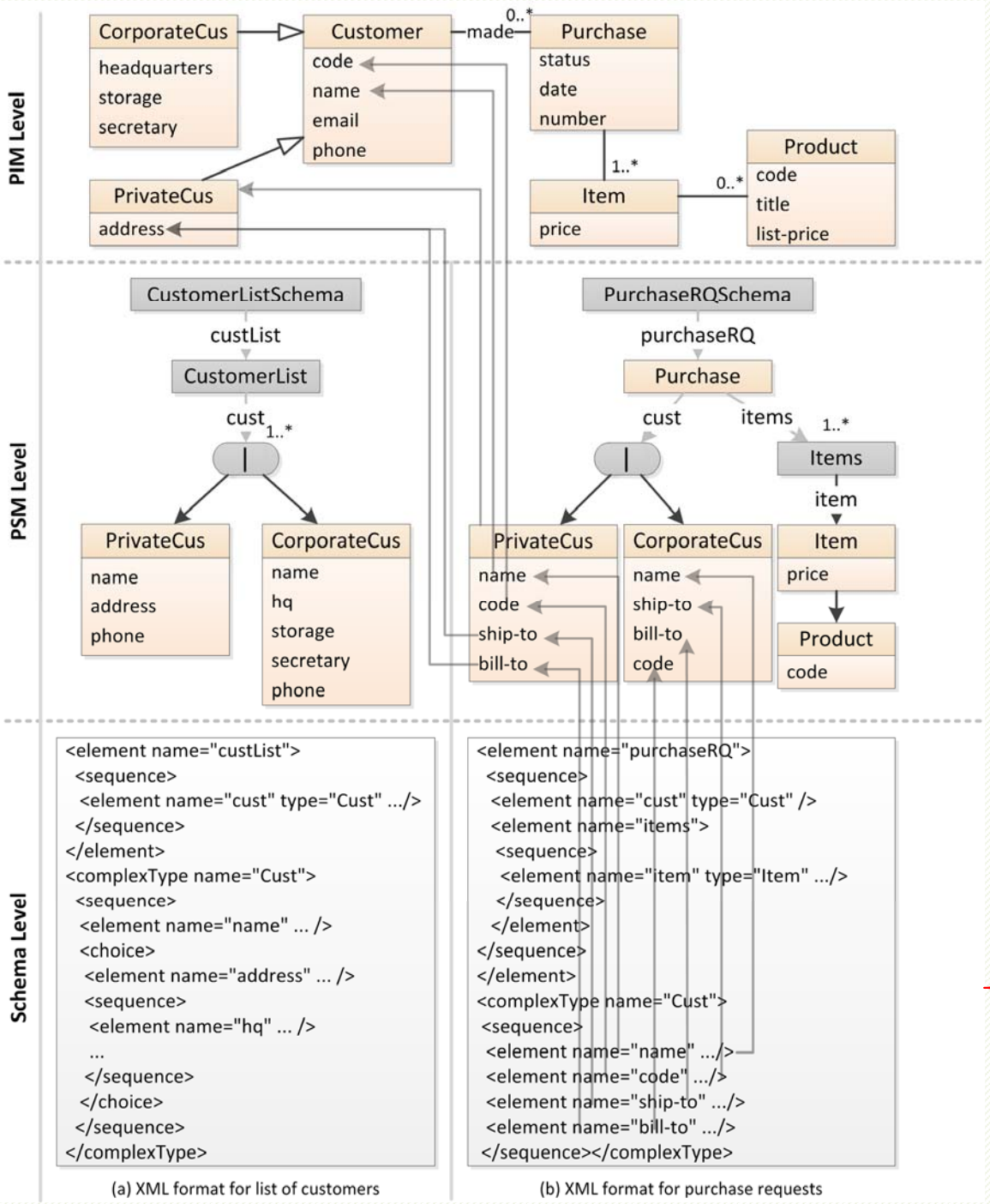
Platform-Specific Model (PSM)

- How part of reality modelled by PIM is represented in an XML schema
  - "Glue" between PIM and XML schemas
- UML class diagrams extended for the purposes of XML modelling
  - Hierarchical structure, order, XML elements vs. attributes, …
  - XSEM model



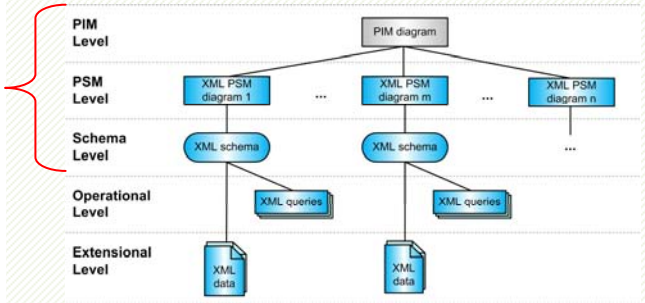


Necasky, M.: Conceptual Modeling for XML. In Dissertations in Database and Information Systems Series, Volume 99. IOS Press/AKA Verlag. January 2009, 164pp. ISBN: 978-58603-956-1.



(a) XML format for list of customers

(b) XML format for purchase requests



Two Perspectives of PSM

- Conceptual:
  - Classes, attributes and associations of PSM are mapped to components of PIM
  - The mapping specifies the semantics of PSM schema in terms of PIM schema
- Grammatical:
  - A PSM schema models an XML format = an XML schema = a regular tree grammar
  - A regular tree grammar can be translated to a PSM schema and vice versa
    - Multiple possible translations from PSM to XML schema

Logical Level

Op. Level

Extensional Level

```
<element name="custList">
 <sequence>
  <element name="cust" type="Customer" .../>
 </sequence>
</element>
<complexType name="Customer">
 <sequence>
  <element name="name" type="string"... />
  <choice>
   <element name="address" type="string" />
   <sequence>
    <element name="hq" type="string" />
    ...
   </sequence>
  </choice>
 </sequence>
</complexType>
```

```
for $c in //cust
where $c/hq
return
 <corporate>{$c/name}
  </corporate>
```

```
<custList version="1.3">
 <cust>
  <name>Martin Necasky</name>
  <address>Vaclavske nam. 123, Prague</
address>
  <phone>123 456 789</phone>
 </cust>
 <cust>
  <name>Department of Software Engineering,
   Charles University</name>
  <hq>Malostranske nam. 25, Prague</hq>
  <storage>Ke Karlovu 3, Prague</storage>
  <secretary>Ke Karlovu 5, Prague</secretary>
  <phone>111 222 333</phone>
 </cust>
</custList>
```

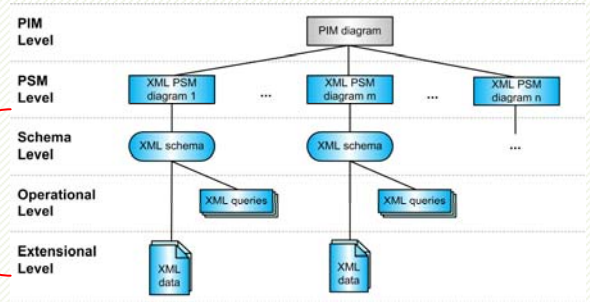
(a) XML format for list of customers

```
<element name="purchaseRQ">
 <sequence>
  <element name="cust" type="Cust" />
  <element name="items">
   <sequence>
    <element name="item" type="Item" .../>
   </sequence>
  </element>
</sequence>
</element>
<complexType name="Cust">
 <sequence>
  <element name="name" .../>
  <element name="code" .../>
  <element name="ship-to" .../>
  <element name="bill-to" .../>
 </sequence></complexType>
```

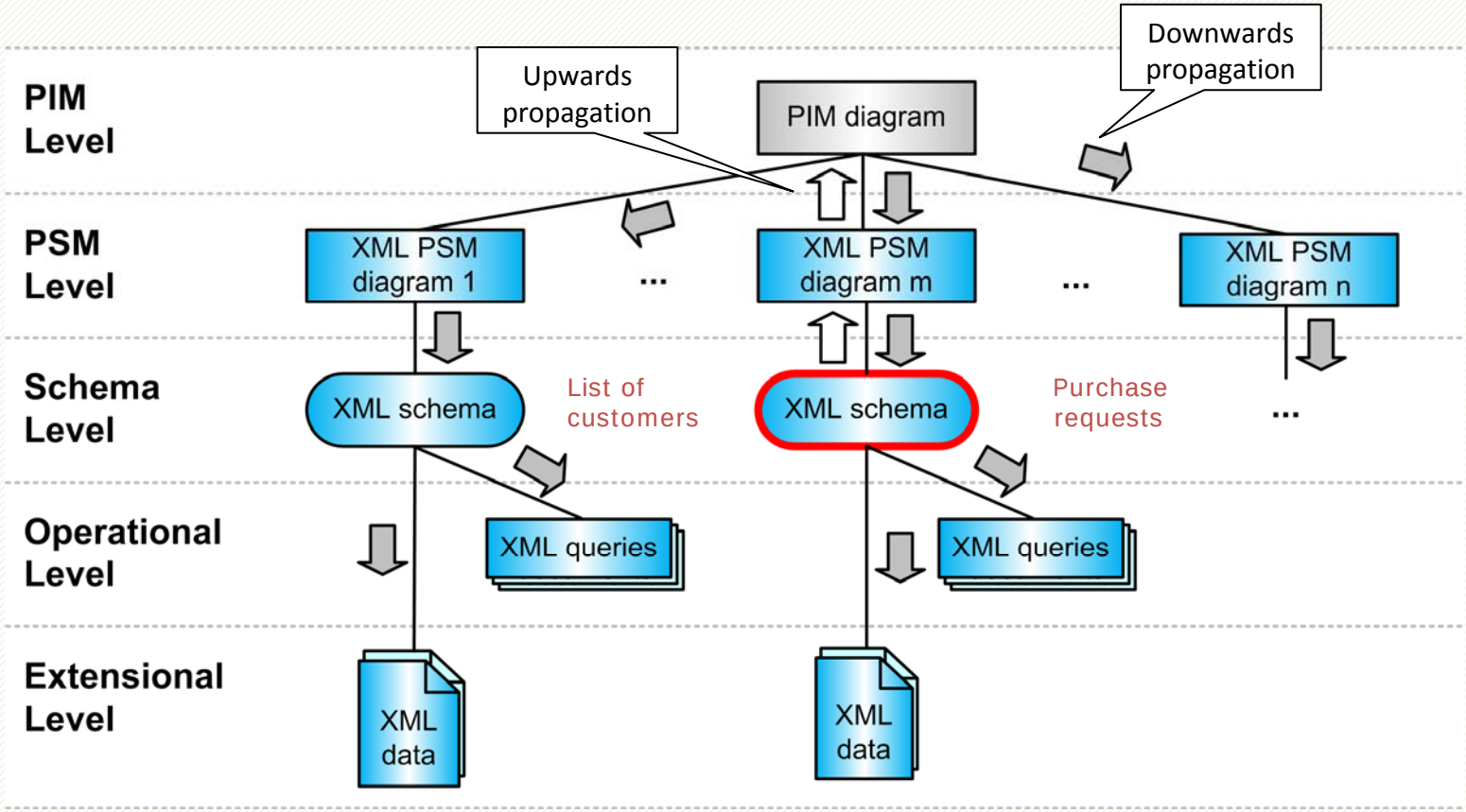
```
for $p in /purchaseRQ
return fn:sum(
 for $it in $p//item
 where $it/price > 20
 return $price)
```

```
<purchaseRQ version="1.0">
 <cust>
  <name>Department of Software Engineering,
   Charles University</name>
  <code>ksi@mff.cuni.cz</code>
  <bill-to>Malostranske 25, Prague</bill-to>
  <ship-to>Ke Karlovu 3, Prague</ship-to>
 </cust>
 <items>
  <item>
   <code>P045</code><price>17</price>
  </item>
  <item>
   <code>P332</code><price>34</price>
  </item>
 </items>
</purchaseRQ>
```

(b) XML format for purchase requests



How the Evolution Process Works?



But there are many related problems to be solved…

“Background” Issues

- Correct definition of PIM and PSM models
- Correct definition of mapping between the levels
- Definition of edit operations at each level
  - Atomic vs. composite
- Correct definition of propagation of each operation to neighbouring levels

In related literature mostly omitted

or solved partially,

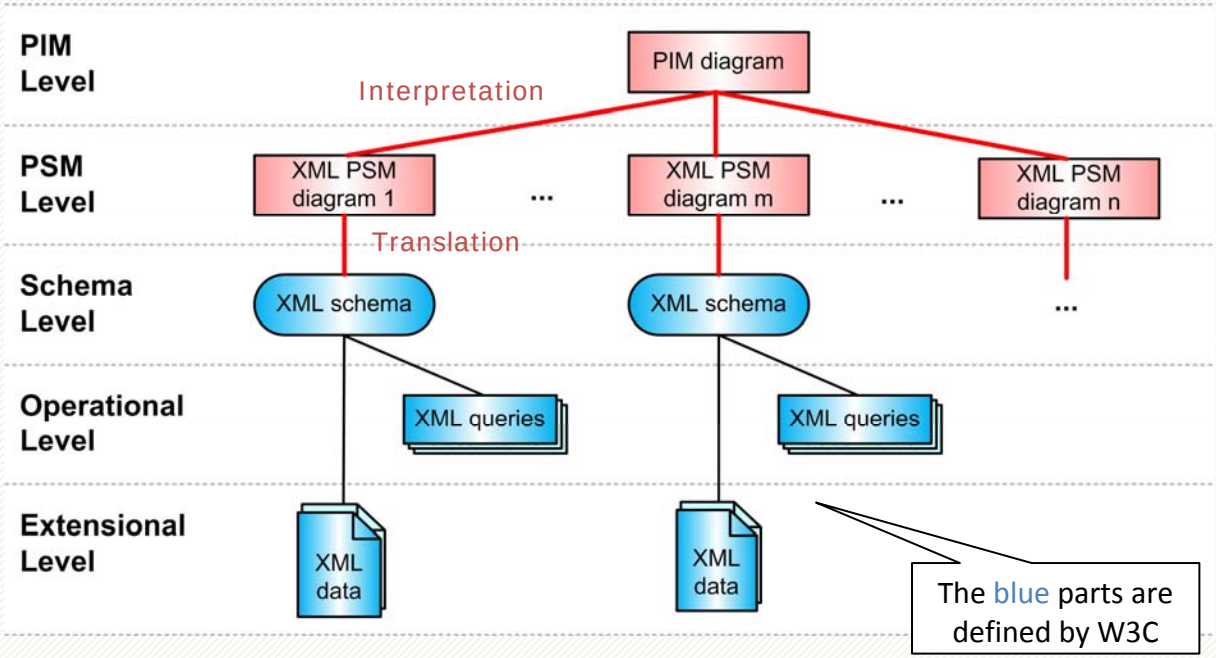
or without the general context…

Even More “Background” Issues

- Forward vs. reverse engineering
  - Inference of XML schemas
  - Translation of XML schemas to PSM
  - Mapping of PSM to PIM

Formal Definition of Models and Mapping

- Algorithm for translating PSM to regular tree grammars and vice versa
  - Proof of correctness and expressive power

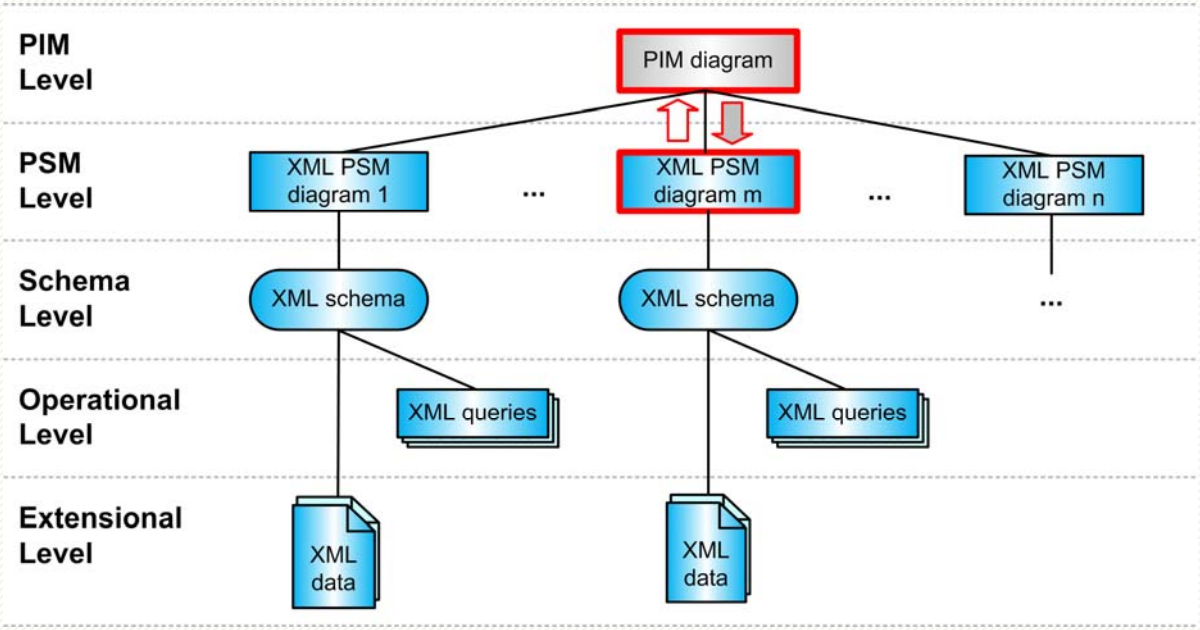


- Definition of interpretation = formal definition of mapping between conceptual levels
  - Proof of correctness

Necasky, M. - Mlynkova I. - Klimek, J. - Maly, J.: When Conceptual Model Meets Grammar: A Dual Approach to XML Data Modeling. International Journal on Data & Knowledge Engineering, volume 72, pages 1 - 30. Elsevier, February 2012.

Edit Operations and Their Propagation

- Definition of a set of atomic operations at PSM and PIM levels
  - Pre- and post-conditions that must be fulfilled
  - Proof of minimality, correctness



- Specification of propagation mechanism
  - Proof of correctness
- Definition of composition of edit operations
  - More user-friendly operations

Necasky, M. - Klimek, J. - Maly, J. - Mlynkova, I.: Evolution and Change Management of XML-based Systems. Journal of Systems and Software, volume 85, issue 3, pages 683–707. Elsevier, February 2012.

Our Edit Operations

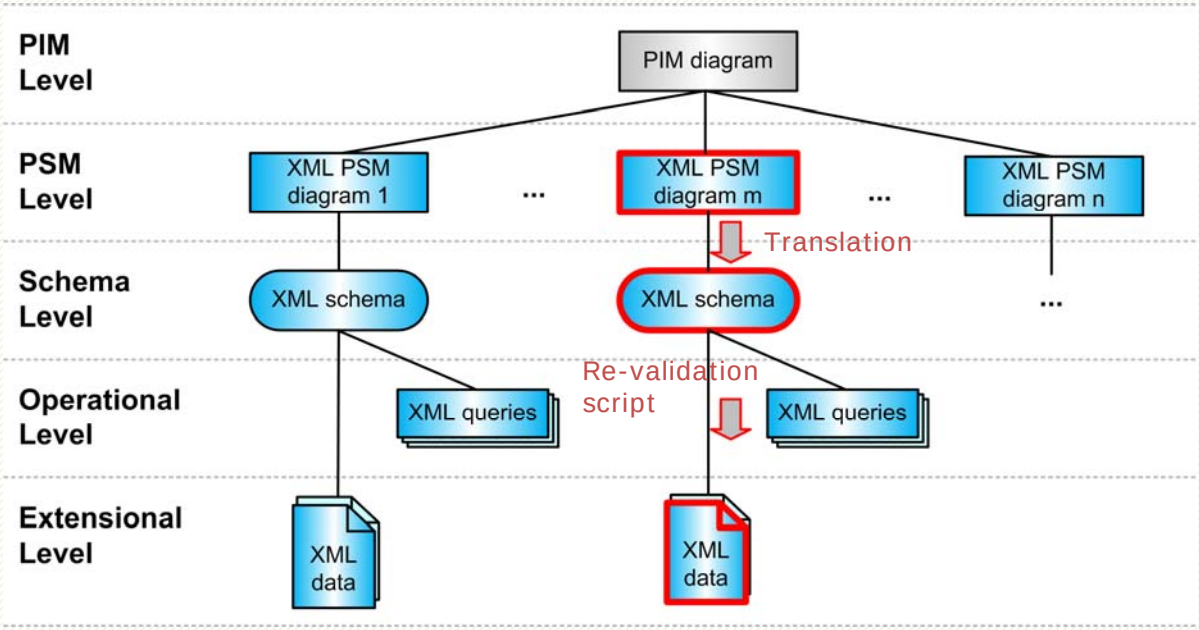
- Creation, update, removal – classical operations
- New atomic operation: synchronization
  - Two sets of schema components are semantically equivalent
- Idea:
  - Requirement: Address is not a simple string, but should be divided into street, city, zip code, …
  - Existing approaches: creating new attributes + removing the old one

    => loss of information (data)
  - Our approach: synchronization

    => the propagation mechanism knows where to “get” data

Re-Validation of XML Documents

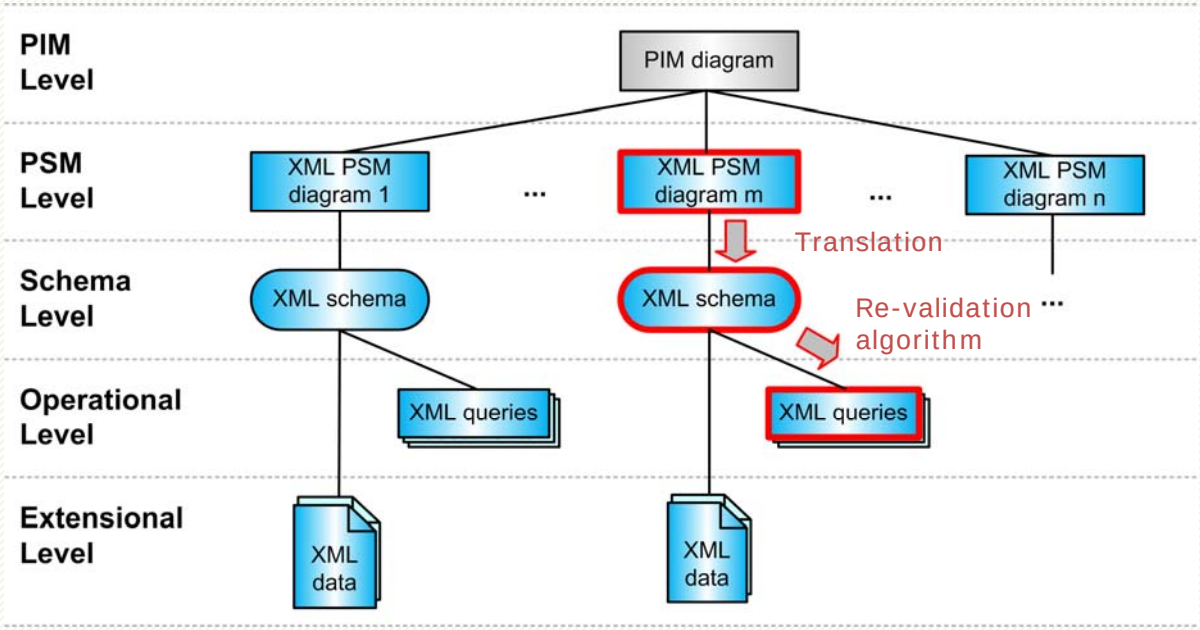
- PSM diagram = XML schema changes => XML documents need to be adapted
  - Re-validation (XSLT) scripts
- Approaches:
  - Recording changes
  - Schema comparison
- New concept: version links



Maly, J. - Necasky, M. - Mlynkova, I.: Efficient adaptation of XML data using a conceptual model. Information Systems Frontiers. Springer Science+Business Media, 2012. ISSN 1387-3326. (In Press.)

Propagation to Operational Level

- Schema evolves => query inconsistency



- We need: a model of queries, a mapping between data and queries, edit operations, propagation mechanism, …

Propagation to Operational Level

- So far we have considered only data levels
- Query inconsistency
  - Returning error (e.g. non-existing element), more/less results, completely different data
- Current approaches: a single paper describing rules for backward compatible queries
  - Artificial
  - Cannot be used in every case
  - Imprecise for various applications
- Our current work: extension of the evolution framework with propagation to operational level
  - First step: subset of XPath queries, study of impact of particular atomic operations

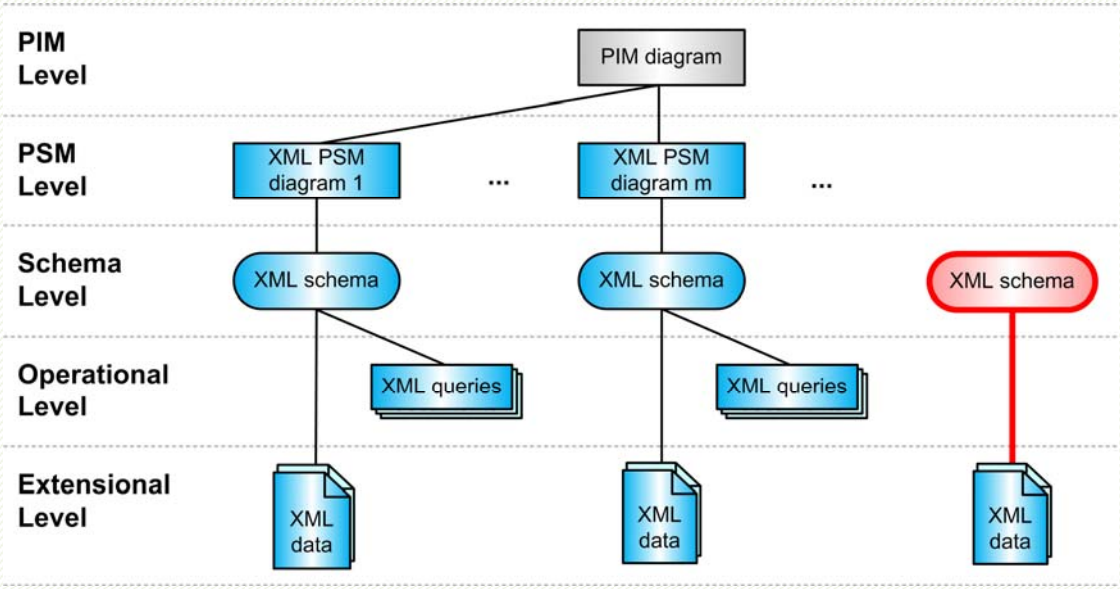
Polak, M. - Mlynkova, I. - Pardede, E.: XML Query Adaptation as Schema Evolves. ISD '12. Springer Science+Business Media, Inc., 2013.

Modelling Strategies

- Top-Down (forward engineering) – modelling a new system (or its modification)
  1. Creating PIM schema
  2. Creating a PSM schema from PIM
     - Or its modification
  3. Translating PSM schema to XML schema
- Bottom-Up (reverse engineering) – an existing system + integration of an existing XML schema
  1. If we do not have a schema, we need an inference method
  2. Translating XML schema to PSM
     - Straightforward thanks to the previous results
  3. Mapping of PSM schema to and PIM schema
     - We need a suitable similarity measure

Reverse Engineering: Inference of XML Schemas

- Real-world XML data:
  - 52% of randomly crawled / 7.4% of semi-automatically collected have no schema at all
- But we need a schema to be integrated into the system

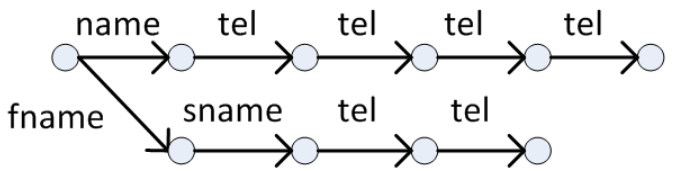


A General Schema of the Inference Process

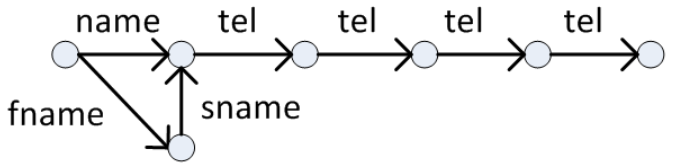
```
<custList>
 <cust>
  <name>Department</name>
  <tel>111-222-444</tel>
  <tel>222-333-444</tel>
  <tel>111-444-333</tel>
  <tel>111-222-555</tel>
 </cust>
 ...
</custList>
```

```
<custList>
 <cust>
  <fname>Martin</sname>
  <sname>Necasky</sname>
  <tel>666-444-333</tel>
  <tel>666-444-222</tel>
 </cust>
 ...
</custList>
```

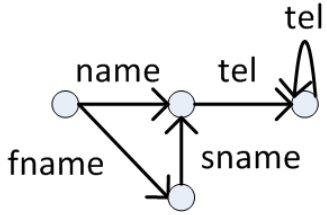
step 1:



step 2:



step 3:



step 4: ((name|(fname,sname)),tel+)

Inference of XML Schema from XML Documents

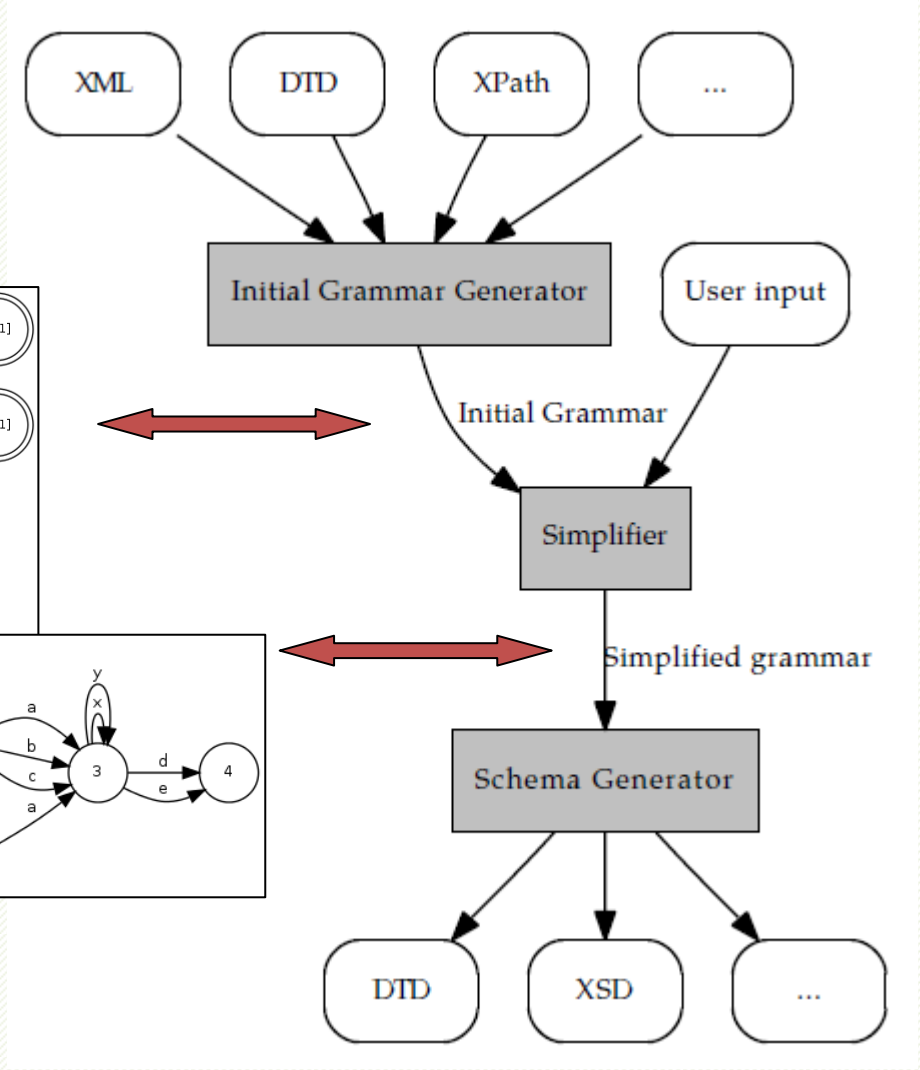
- Input: a set of XML documents
- Output: an XML schema against which the documents are valid
- Gold's theorem: regular languages are not identifiable from positive examples
  - regular language = XML schema
  - positive example = XML documents which should conform to the resulting schema
- Currently:
  - Heuristic approaches vs. grammar-based approaches
  - Exploit purely XML documents
- Our proposals:
  - Other input data
    - Obsolete schema, queries, user interaction, …
  - Other output information
    - XSD constructs, integrity constraints, …
  - Optimization of the inference process

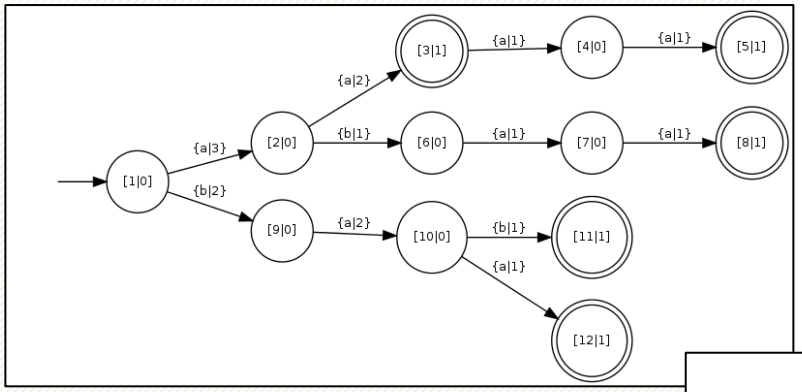
Mlynkova, I. - Necasky, M.: Heuristic Methods for Inference of XML Schemas: Lessons Learned and Open Issues. Informatica. IOS Press, 2012. ISSN 0868-4952. (In Press.)

jInfer

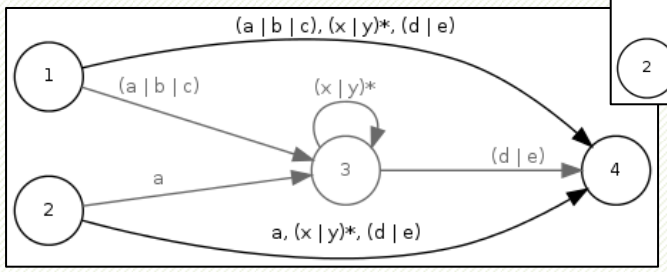
- A framework for XML schema inference
- Based on the idea of plug-ins
  - Modules
  - Can be added, removed, replaced, compared, …
- "Playground" for proposing new inference methods
  - "Not interesting" features are implemented
    - Transforming XML data to grammar rules
    - Transforming automata to XML schemas
    - …
  - Current know approaches are implemented
    - New ones can be compared

jInfer


XML
DTD
XPath
...
Initial Grammar Generator
User input
Initial Grammar
Simplifier
Simplified grammar
Schema Generator
DTD
XSD
...

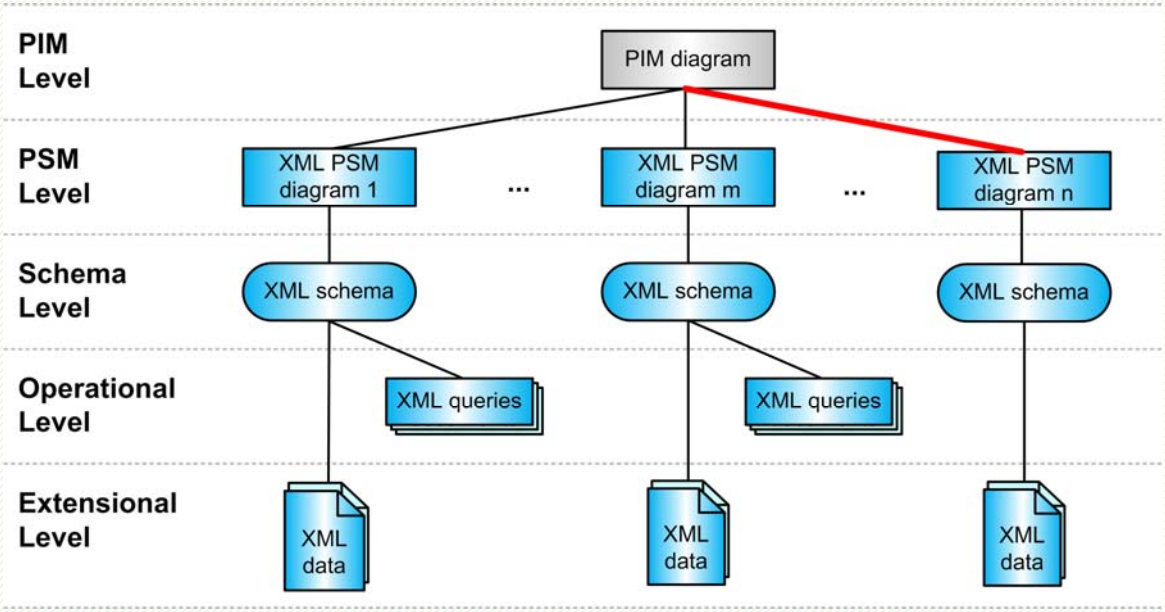

[1|0]
[2|0]
[3|1]
[4|0]
[5|1]
[6|0]
[7|0]
[8|1]
[9|0]
[10|0]
[11|1]
[12|1]
{a|3}
{b|2}
{a|2}
{b|1}
{a|1}

1
2
3
4
a
b
c
x
y
d
e


(a | b | c), (x | y)*, (d | e)
1
2
3
4
(a | b | c)
(x | y)*
(d | e)
a
a, (x | y)*, (d | e)

Reverse Engineering: Mapping of PSM to PIM

- Top-down:
  - User directly provides the mapping denoting parts of PIM to be interpreted in PSM
- Bottom-up:
  - More complex
  - We need to "find" the mapping



Starka, J. - Mlynkova, I. - Klimek, J. - Necasky, M.: Integration of Web Service Interfaces via Decision Trees. IIT '11, pages 47-52, Abu Dhabi, United Arab Emirates, April 2011. IEEE Computer Society, 2011.

Wojnar, A. - Mlynkova, I. - Dokulil, J.: Structural and Semantic Aspects of Similarity of Document Type Definitions and XML Schemas. Journal on Information Sciences, volume 180, issue 10, pages 1817 - 1836. Elsevier, May 2010.

Mapping of PSM Schema to PIM

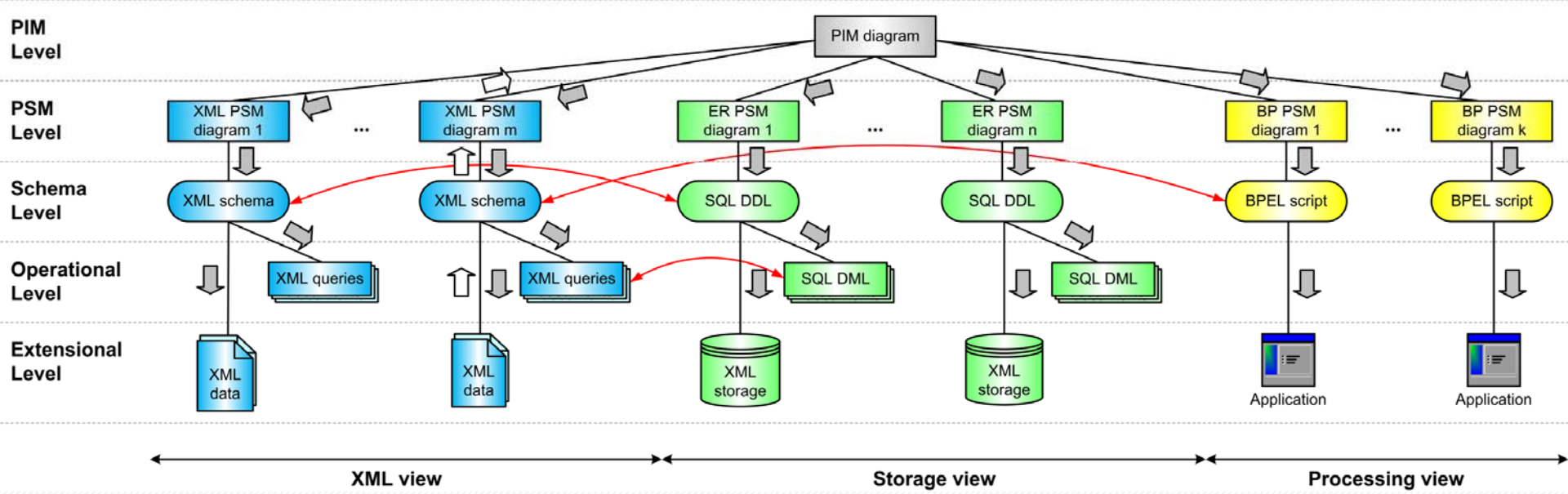
- Input: a PIM schema and a PSM schema
- Output: correct mapping between PIM ad PSM
  - Correct interpretation
- Currently there exists a huge amount of mapping strategies
  - Analyze structure, context, semantics, operations, …
- Our case: semi-automatic
  - But we can help the user with automatic parts
- Aims:
  - To adapt verified techniques to given application
    - Decision trees
  - To minimize user interaction
    - Exploitation previous results

Summary

- We have solved the problem of evolution of complex XML applications - we have defined:
  - Five-level framework
  - All the levels + mutual mapping
  - Operations at levels
    - Atomic + composite
  - Propagation of operations to other levels
  - Building the system from both directions
    - Top-down + bottom-up

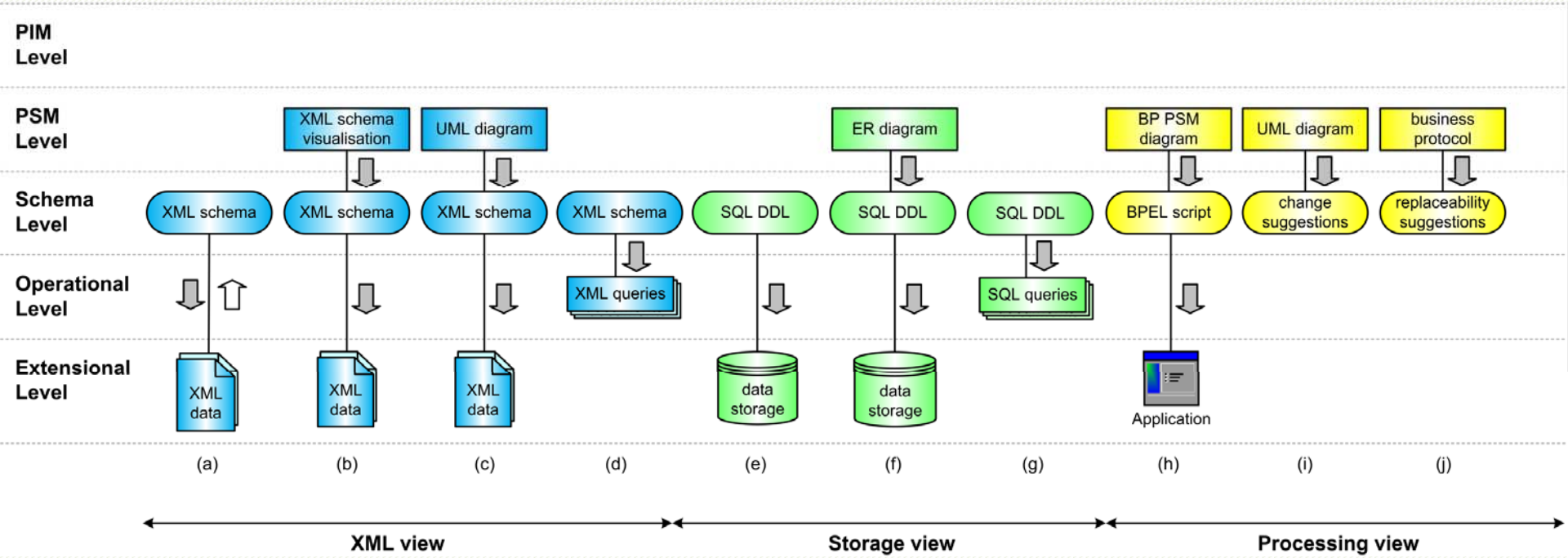
The work is done…?

We Can Go Further…



- An XML system can contain other than XML data => the ideas can be generalized
  - Relational models, UML models, …
  - Business process models

Related Work



- Particular cases and views of the problem have previously only been solved separately
  - Sometimes just superficial, imprecise description
  - No theoretical or formal basis

Implementations

- XCase (http://xcase.codeplex.com/)
  - Implementation of first ideas
  - Desktop application
  - XML data levels and propagation among them
- eXolutio (http://exolutio.com/)
  - Web application of XCase
  - Automatic generation of mappings between PIM and PSM
- DaemonX (http://daemonx.codeplex.com/)
  - A general framework for any kind of data (not only XML)
    - New data format => new plug-in
  - Currently: XML, relational, UML

XCASE

Future Work

- Adding more complex conceptual-modelling constructs
  - E.g. inheritance
- Queries,
  - Modelling, propagation, …
- Integrity constraints (at all the levels)
  - Modelling, propagation, …
- Extension to other data formats
  - Formal specification, proofs of correctness, …
- Storage strategies
  - New dimension: schema evolves => storage strategy should be optimized
- Extension to business process models
  - New dimension: not data structures, but “what happens with them”

Thank you very much
for your attention