Business Informatics Group

Vienna University of Technology

XMLText: From XML Schema to Xtext*

Patrick Neubauer, Alexander Bergmayr, Tanja Mayerhofer,
Javier Troya, and Manuel Wimmer

8th ACM SIGPLAN International Conference on
Software Language Engineering (SLE), 2015, Pittsburgh, USA

Business Informatics Group
Institute of Software Technology and Interactive Systems
Vienna University of Technology
Favoritenstraße 9-11/188-3, 1040 Vienna, Austria
phone: +43 (1) 58801-18804 (secretary), fax: +43 (1) 58801-18896
office@big.tuwien.ac.at, www.big.tuwien.ac.at

*This work is co-funded by the European Commission under the ICT Policy Support Programme, grant no. 317859

Introduction

Problem and Motivation

Problem:

- Immutable and verbose XML syntax
- Complex, error-prone, and time-consuming engineering of DSLs
- Human-comprehensibility and maintainability

Motivation / Objectives:

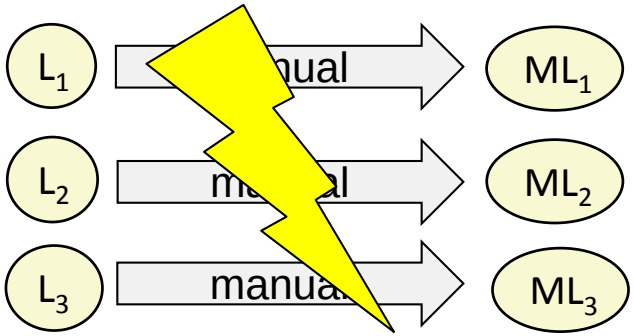
- Modernization of existing languages for domain experts
- Create interoperability between XMLware, Grammarware, and Modelware
- Minimize required domain knowledge and language engineering skills

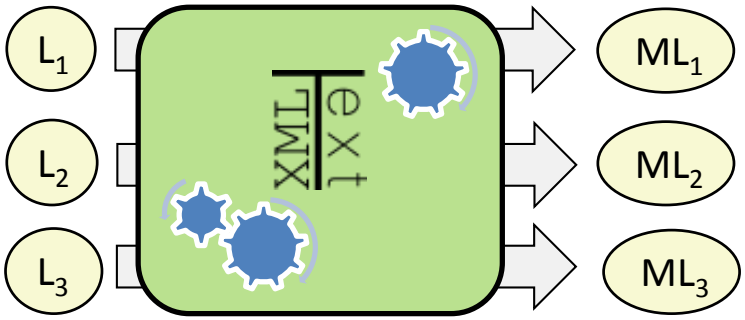
G. J. Badros. JavaML: A Markup Language for Java Source Code. Computer Networks, 33(1):159–177, 2000.
Kurtev, I., Bézivin, J., Akşit, M.: Technological Spaces: An Initial Appraisal. In: Proc. CoopIS, 1–6 (2002).
M. Mernik, J. Heering, and A. M. Sloane. When and how to develop domain-specific languages. ACM Computing Surveys (CSUR), 37(4):316–344, 2005.

Contribution

- Bridging *n* languages of technical space A with *m* languages of technical space B via one generic bridge instead of *n* * *m* individual bridges

Legend:
L_x … Language x
ML_x … Modernized language x





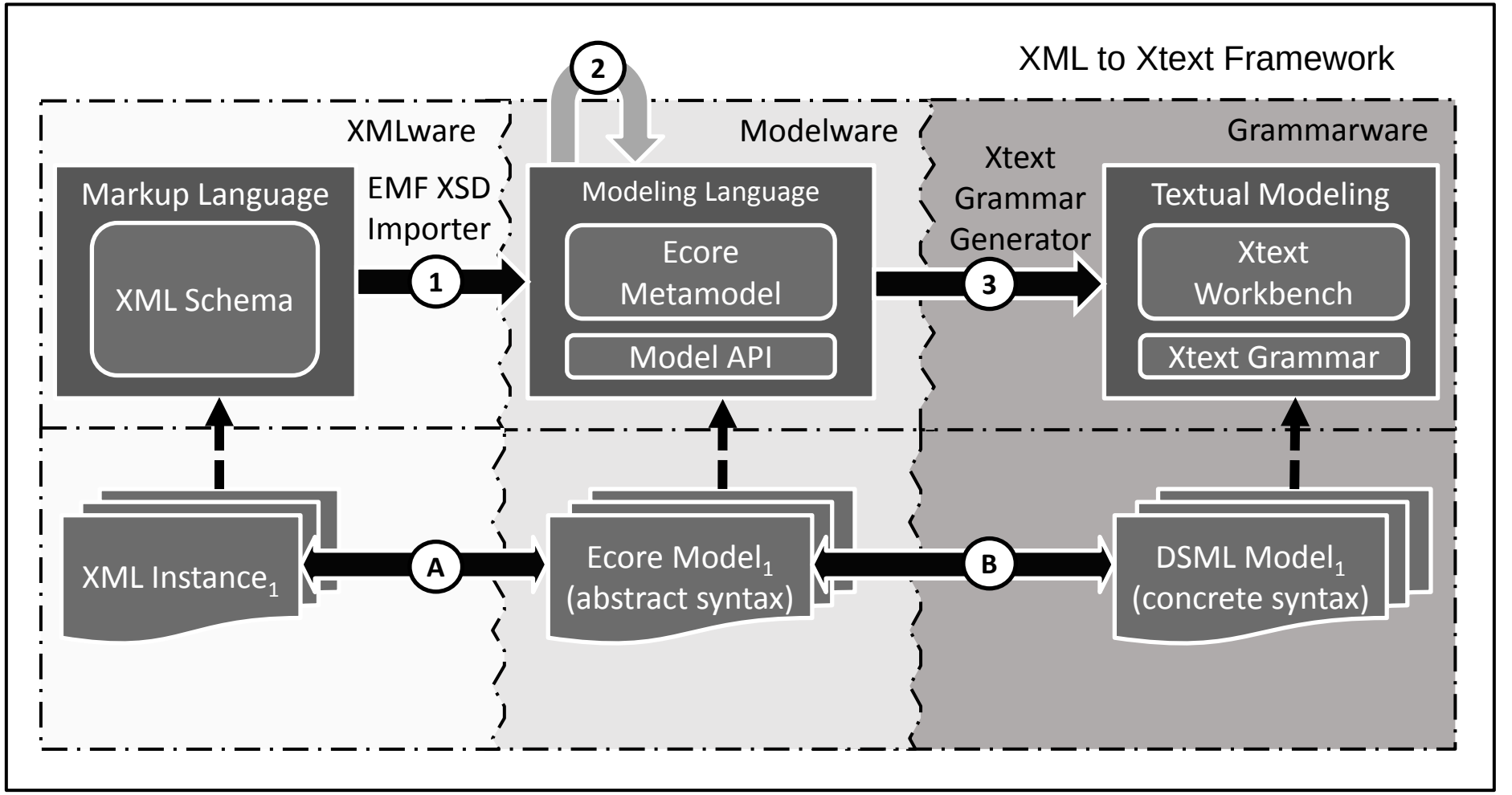
- Automation of language modernization in consolidated framework
- Exploitation of existing tools: EMF XSD Importer and Xtext Grammar Generator
- Bridging individual gaps between involved technical spaces

Czarnecki, K., Favre, J. M., Gogolla, M., & Mens, T. (2006, January). Essentials of the 4th UML/MoDELS Workshop in Software Model Engineering (WiSME'2005). In *Satellite Events at the MoDELS 2005 Conference* (pp. 151-158). Springer Berlin Heidelberg.

Muller, P. A., Fondement, F., Fleurey, F., Hassenforder, M., Schnekenburger, R., Gérard, S., & Jézéquel, J. M. (2008). Model-driven analysis and synthesis of textual concrete syntax. *Software & Systems Modeling*, *7*(4), 423-441.

Approach and Demo

Overview



Motivating Example

Library3 XML Schema

```
<xs:simpleType name="pagesType">
  <xs:restriction base="xs:int"></xs:restriction>
</xs:simpleType>
```

Data types

```
    <xs:sequence>
        <xs:element name="book" type="bookType" maxOccurs="unbounded" minOccurs="0"/>
        <xs:element name="customer" type="customerType" maxOccurs="unbounded" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
<xs:complexType name="bookType">
  <xs:sequence>
    <xs:element name="name" type="xs:ID"/>
  </xs:sequence>
  <xs:attribute name="isbn" type="isbnType" use="required"/>
</xs:complexType>
```

Identifiers

```
<xs:complexType name="customerType">
  <xs:sequence>
      <xs:element name="firstName" type="xs:string"/>
      <xs:element name="lastName" type="xs:string"/>
      <xs:element name="borrowedBookId" type="xs:IDREF"/>
  <xs:sequence>
     <xs:any namespace="##any" processContents="lax" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>
```

Identifier References

Wildcards / Semi-Structured data

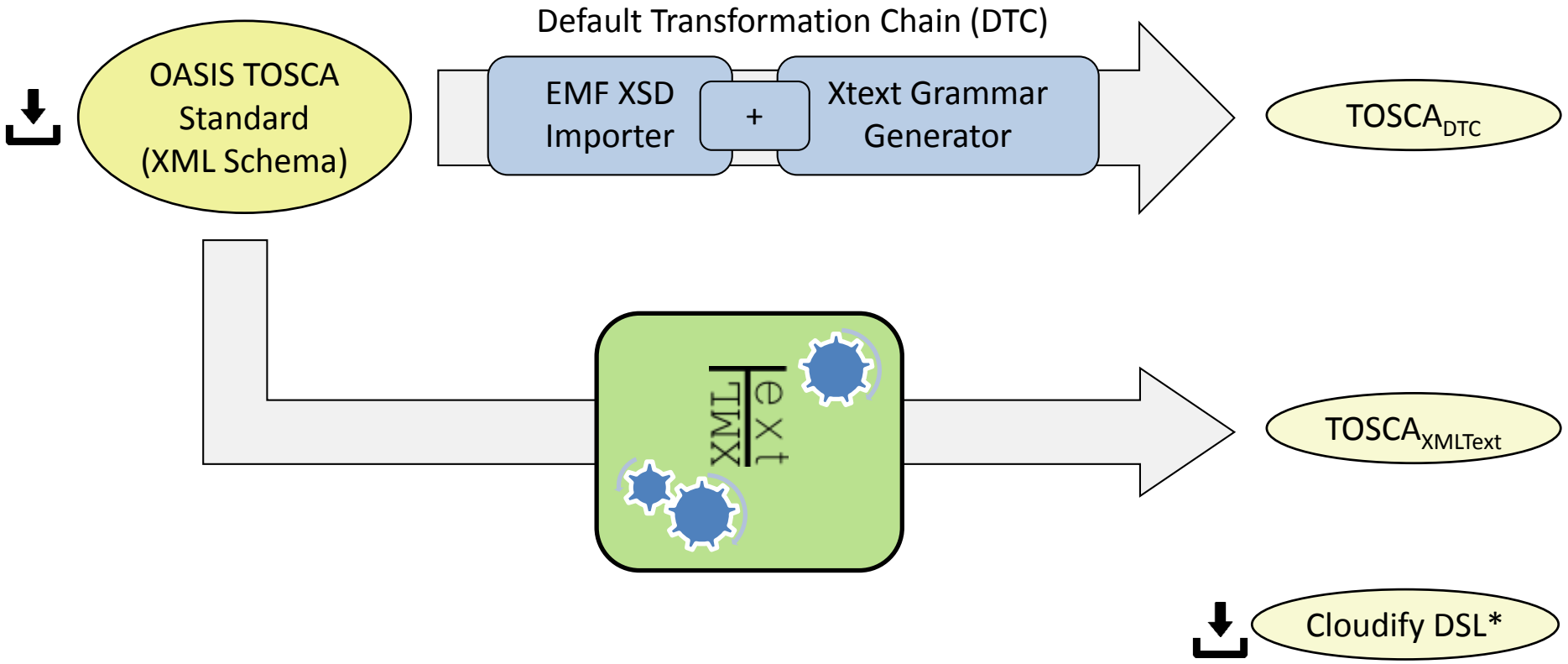
Excerpt of library3.xsd

- Library books and customers
- Customers and books own attributes of different types
- Books are uniquely identified
- Customers can borrow a book

Vlist, Eric Van Der. *XML Schema*. Sebastopol, CA: O'Reilly, 2002. Print.

Case Study

Setup



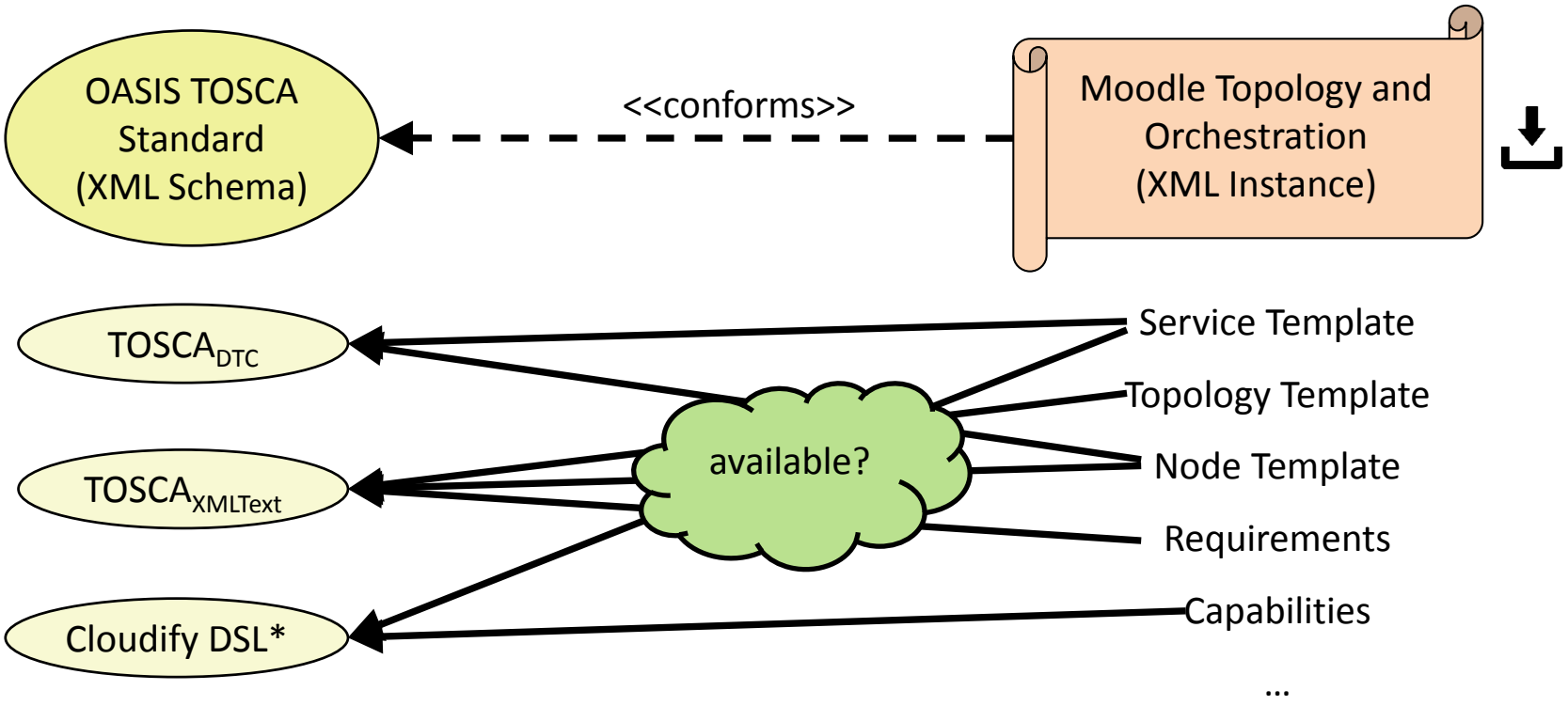
Derek Palma, Thomas Spatzier. Topology and Orchestration Specification for Cloud Applications Version 1.0, 2013.

T. Binz, U. Breitenbücher, F. Haupt, O. Kopp, F. Leymann, A. Nowak, and S. Wagner. OpenTOSCA – a runtime for TOSCA-based cloud applications. Springer, 2013.

(*) GigaSpaces Technologies Ltd. Retrieved September 30, 2015, from https://goo.gl/JzPL7U.

Case Study

Setup



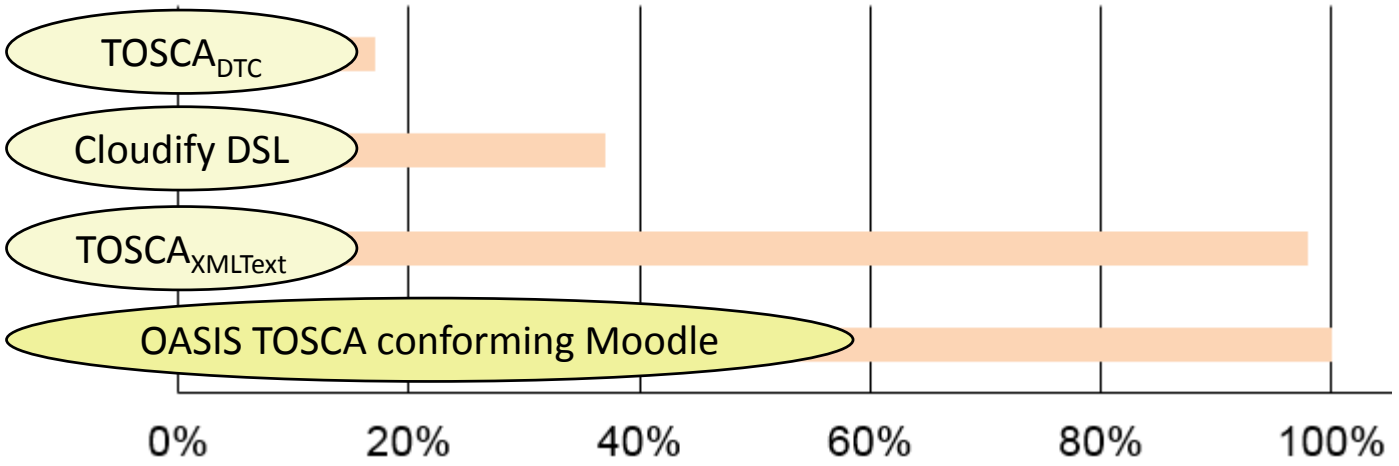
Derek Palma, Thomas Spatzier. Topology and Orchestration Specification for Cloud Applications Version 1.0, 2013.

T. Binz, U. Breitenbücher, F. Haupt, O. Kopp, F. Leymann, A. Nowak, and S. Wagner. OpenTOSCA – a runtime for TOSCA-based cloud applications. Springer, 2013.

(*) GigaSpaces Technologies Ltd. Retrieved September 30, 2015, from https://goo.gl/JzPL7U.

Case Study

Results



Coverage of OASIS TOSCA Moodle concepts and features in respective languages

	OASIS TOSCA conforming Moodle	TOSCA$_{DTC}$	Cloudify DSL	TOSCA$_{XMLText}$
TOSCA Concepts	19	2 (~11%)	11 (~58%)	19 (100%)
TOSCA Features	35	7 (20%)	9 (~26%)	34 (~97%)
TOSCA Combined	54	**9 (~17%)**	**20 (~37%)**	**53 (~98%)**

Case Study

Discussion

$TOSCA_{DTC}$ is not sufficient to represent the Moodle example

- Missing nodes, relationships, …

Cloudify DSL is more sophisticated

- However, missing requirements, capabilities, …
- “Our goal is to have full compliance with the standard in one of the near future versions” – Cloudify DSL Specification Overview

$TOSCA_{XMLText}$ allows to specify almost all concepts and features

- Missing (only) “xmlns” feature

Results can be extended

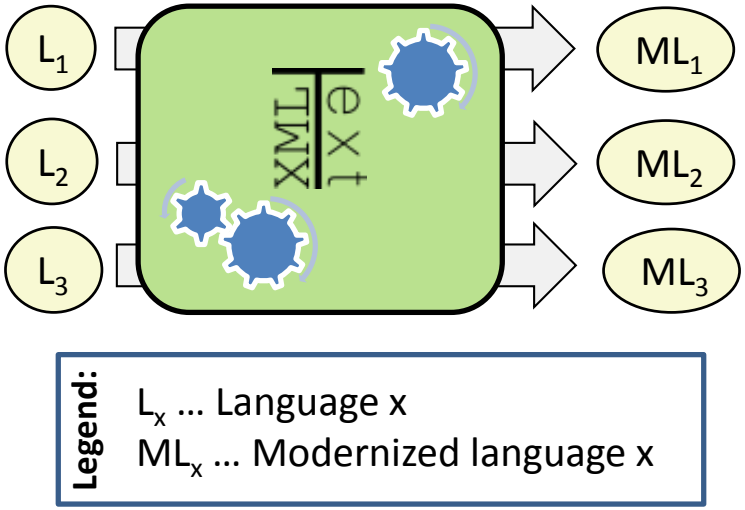
- … to establish overall coverage of TOSCA
- … to establish applicability to different XSD-based languages

Cloudify DSL Specification Overview. Retrieved October 19, 2015, from http://getcloudify.org/guide/3.1/dsl-spec-general.html.

Summary, Ongoing Work, and Future Work

Summary

- Approach to modernize XML Schema languages with Xtext-based languages
- Evaluated by comparison of different implementations of TOSCA language based on Moodle example
- AutomationML evaluation (ongoing)



Future Work

- Evaluation by different languages and examples
- Automation of several steps
- Support for more XML Schema data types
- Extension of syntax customization capabilities
- Impact evaluation of syntactical changes through user study conduction

Business Informatics Group

Vienna University of Technology

Thank You !

For more please visit the **poster session** on Wednesday, 6-9pm (room: Admiral and Reflections, 1st floor).

http://xmltext.big.tuwien.ac.at

Business Informatics Group
Institute of Software Technology and Interactive Systems
Vienna University of Technology
Favoritenstraße 9-11/188-3, 1040 Vienna, Austria
phone: +43 (1) 58801-18804 (secretary), fax: +43 (1) 58801-18896
office@big.tuwien.ac.at, www.big.tuwien.ac.at

*This work is co-funded by the European Commission under the ICT Policy Support Programme, grant no. 317859

Related Work

- Forward engineering: UML to XML Schema or DTD
- Reverse engineering: XML Schema or DTD to UML
  - Wimmer et al.: semi-automatic approach to generate MOF metamodel from DTD with the purpose to enhance language understanding (among others)

- Eysholdt et al.: practitioners report on migration of modeling environment from XML/UML to Xtext/GMF
  - List manual steps and focus on a specific XSD-based language
  - XML is inefficient due to verbose syntax and lack of good tool support

- Grammar-based approaches: generate metamodel out of existing grammar definition
  - J Canovas et al. Gra2Mol: similar to ATL, however source transformation element is grammar element instead of metamodel element

- Metamodel-based approach: generate grammar out of existing metamodel definition
  - F. Jouault et al. defines extension for ATLAS Model Management Architecture to specify textual concrete syntax in form of keywords; transformation rules are manually defined.

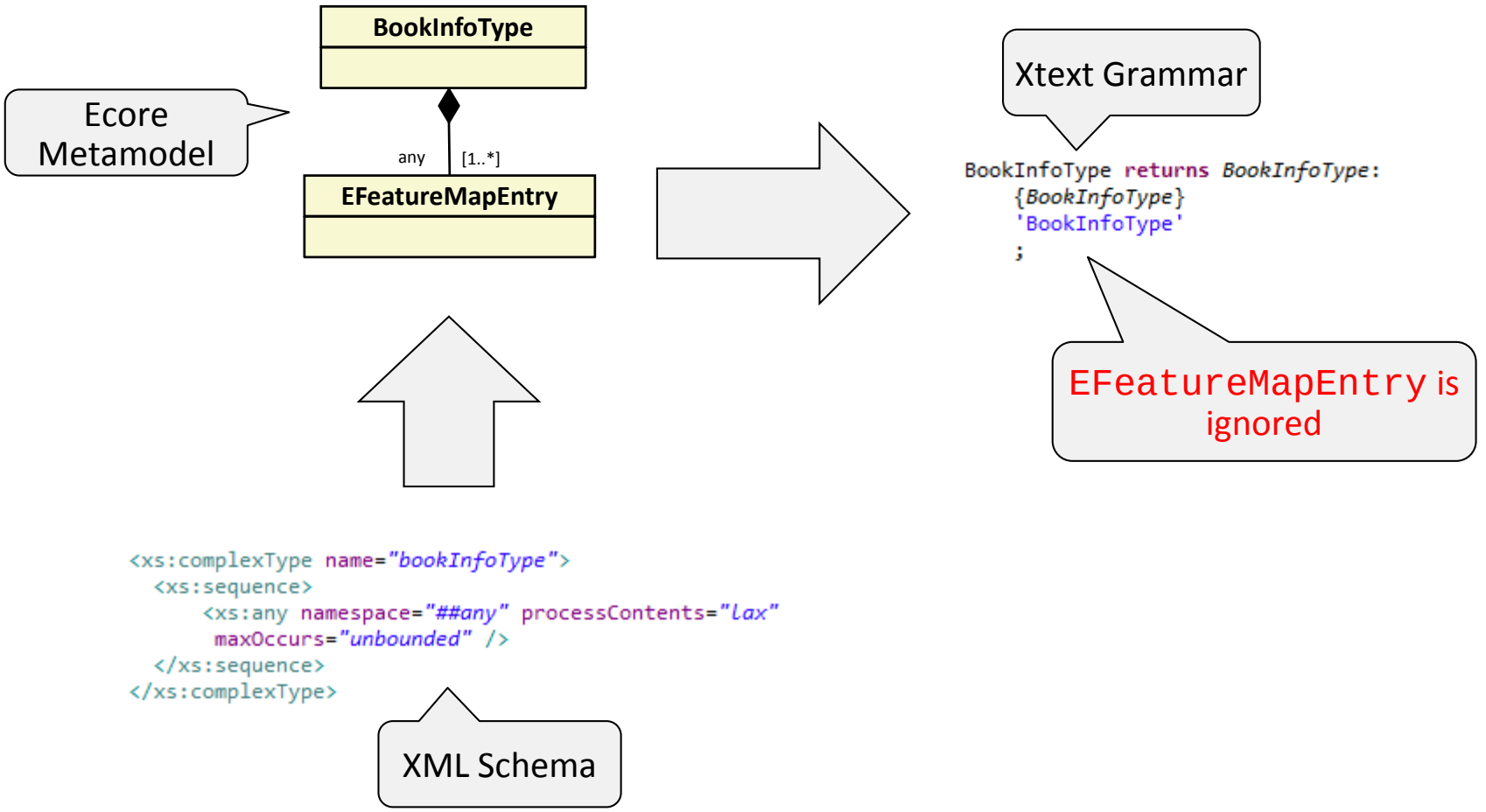
- Bernstein et al. defines tool to translate schemas from source metamodel to target metamodel in an object-to-relational mapping

M. Wimmer and G. Kramler. Bridging Grammarware and Modelware. In Proc. of Satellite Events at MoDELS, pages 159–168. Springer, 2006.
M. Eysholdt and H. Behrens. Xtext: Implement your Language Faster than the Quick and Dirty Way. In Companion Proc. of OOPSLA, pages 307–309. ACM, 2010.
J. L. Canovas, Izquierdo, and J. G. Molina. Extracting models from source code in software modernization. Software and System Modeling, 13(2):713–734, 2014.
J. Jézéquel, O. Barais, and F. Fleurey. Model driven language engineering with kermeta. In Proc. of GTTSE, pages 201–221, 2009.
P. Atzeni, P. Cappellari, and P. A. Bernstein. Modelgen: Model independent schema translation. In Proc. of ICDE, pages 1111–1112, 2005.

Default Transformation Chain

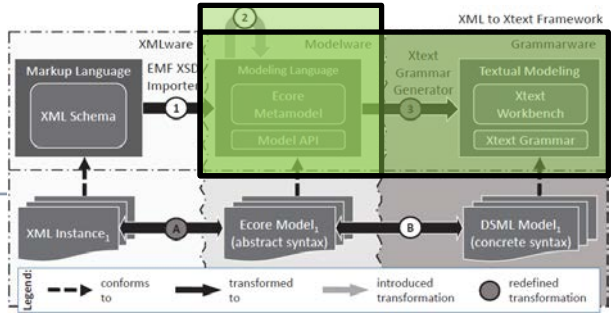
Wildcards / Semi-Structured Data

EMF XSD Importer + Xtext Grammar Generator

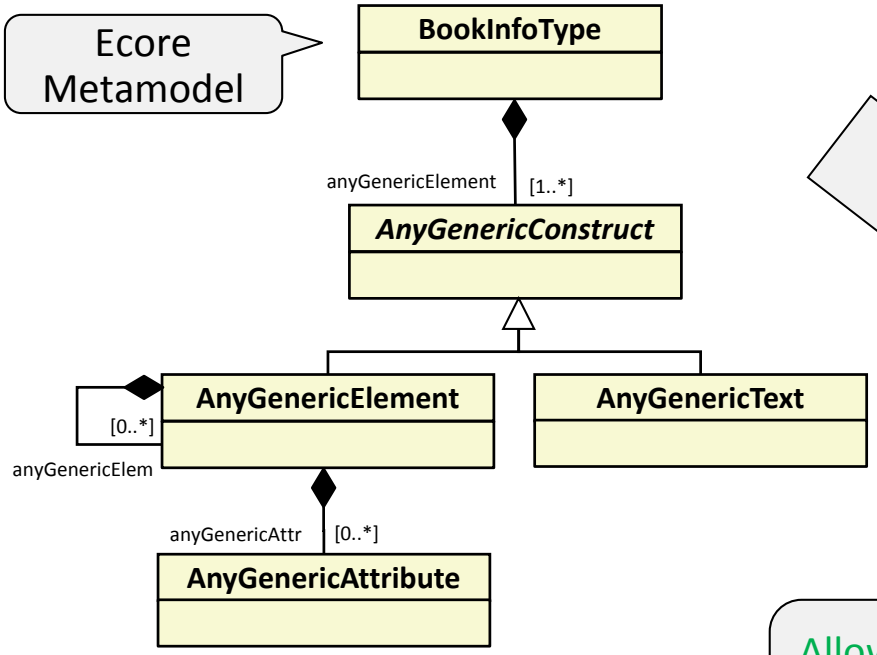


XMLText

Wildcards / Semi-Structured Data



Ecore Metamodel



Xtext Grammar

```
BookInfoType returns BookInfoType:
    'BookInfo'
    any+=AnyGenericConstruct ( "," 
        any+=AnyGenericConstruct )*
;

AnyGenericElement returns AnyGenericElement:
    {AnyGenericElement}
    '{'
        ( elemName=STRING)?
        (':' elemValue=STRING)?
        ( anyGenericAttr+=AnyGenericAttribute ( "," 
            anyGenericAttr+=AnyGenericAttribute )* )?
        ( anyGenericElement+=AnyGenericElement ( "," 
            anyGenericElement+=AnyGenericElement )* )?
    '}';

AnyGenericAttribute returns AnyGenericAttribute:
    attrName=STRING '='
    attrValue=STRING
;

AnyGenericText returns AnyGenericText:
    {AnyGenericText}
    ( textValue=STRING)?
;
```

Allows modeling semi-structured data

```
<xs:complexType name="bookInfoType">
  <xs:sequence>
      <xs:any namespace="##any" processContents="lax"
       maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>
```

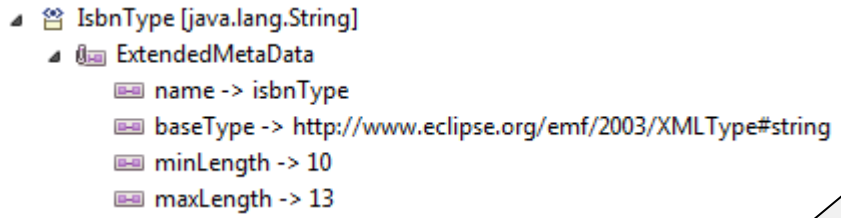
XML Schema

Default Transformation Chain

Data Types

EMF XSD Importer + Xtext Grammar Generator

Ecore Metamodel



Xtext Grammar

```
IsbnType returns IsbnType:
    'IsbnType' /* TODO: implement this rule and an appropriate IValueConverter */;
```

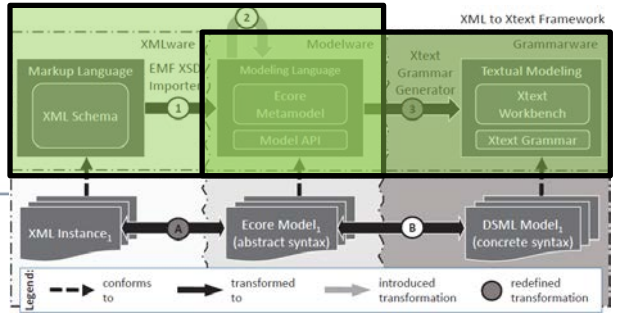
```
<xs:simpleType name="isbnType">
  <xs:restriction base="xs:string">
    <xs:minLength value="10"/>
    <xs:maxLength value="13"/>
  </xs:restriction>
</xs:simpleType>
```

XML Schema

Forces manual implementation of data type rules

XMLText

Data Types



Ecore Metamodel

- IsbnType [java.lang.String]
  - ExtendedMetaData
    - name -> isbnType
    - baseType -> http://www.eclipse.org/emf/2003/XMLType#string
    - minLength -> 10
    - maxLength -> 13

```
datatype IsbnType : 'java.lang.String' { serializable }
{
    annotation _'http:///org/eclipse/emf/ecore/util/ExtendedMetaData'
    (
        name = 'isbnType',
        baseType = 'http://www.eclipse.org/emf/2003/XMLType#string',
        minLength = '10',
        maxLength = '13'
    );
}
```

```
<xs:simpleType name="isbnType">
  <xs:restriction base="xs:string">
    <xs:minLength value="10"/>
    <xs:maxLength value="13"/>
  </xs:restriction>
</xs:simpleType>
```

XML Schema

```
IsbnType returns IsbnType:
    STRING;
```

Xtext Grammar

Allows definition of (only) appropriate values for data type instances

Default Transformation Chain

Identifiers and References

EMF XSD Importer + Xtext Grammar Generator

- BookType
  - ExtendedMetaData
  - name : ID
  - title : String
  - author : NameType
  - pages : Int
  - bookInfo : BookInfoType
  - isbn : IsbnType
- CustomerType
  - ExtendedMetaData
  - firstName : String
  - lastName : String
  - borrowedBookId : IDREF

Property	Value
Changeable	true
Default Value Literal	
Derived	false
EAttribute Type	IDREF [java.lang.String]
EType	IDREF [java.lang.String]
ID	false
Lower Bound	1
Name	borrowedBookId

Ecore Metamodel

```
<xs:complexType name="libraryType">
  <xs:sequence>
      <xs:element name="book" type="bookType" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element name="customer" type="customerType" maxOccurs="unbounded" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="bookType">
  <xs:sequence>
    <xs:element name="name" type="xs:ID"/>
    <xs:element name="title" type="xs:string"/>
    <xs:element name="author" type="nameType"/>
    <xs:element name="pages" type="xs:int"/>
    <xs:element name="bookInfo" type="bookInfoType" min
  </xs:sequence>
  <xs:attribute name="isbn" type="isbnType" use="required"/>
</xs:complexType>
<xs:complexType name="customerType">
  <xs:sequence>
      <xs:element name="firstName" type="xs:string"/>
      <xs:element name="lastName" type="xs:string"/>
      <xs:element name="borrowedBookId" type="xs:IDREF"/>
  </xs:sequence>
</xs:complexType>
```

EAttribute instead of EReference

XML Schema

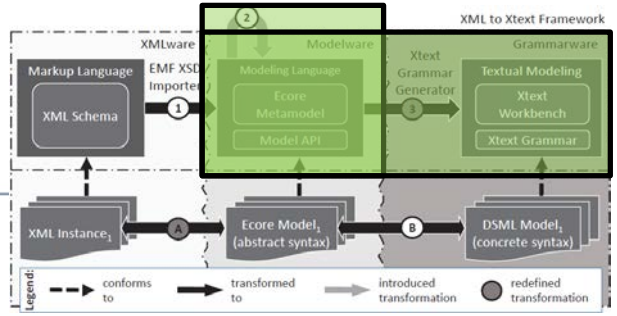
Xtext Grammar

```
BookType returns BookType:
    'BookType'
    name=ID0
    '{'
        'title' title=String0
        'author' author=NameType
        'pages' pages=Int0
        'isbn' isbn=IsbnType
        ('bookInfo' bookInfo=BookInfoType)?
    '}';

CustomerType returns CustomerType:
    'CustomerType'
    '{'
        'firstName' firstName=String0
        'lastName' lastName=String0
        'borrowedBookId' borrowedBookId=IDREF
    '}';

ID0 returns type::ID:
    'ID' /* TODO: implement this rule and an appropriate IValueCon

IDREF returns type::IDREF:
    'IDREF' /* TODO: implement th e and an appropriate IValue
```

Does not allow references

XMLText

Identifiers and References



- BookType
  - ExtendedMetaData
  - name : ID
  - title : String
  - author : NameType
  - pages : Int
  - bookInfo : BookInfoType
  - isbn : IsbnType
- CustomerType
  - ExtendedMetaData
  - firstName : String
  - lastName : String
  - borrowedBookId : EObject

Property	Value
Changeable	true
Container	false
Containment	false
Default Value Literal	
Derived	false
EKeys	
EOpposite	
EType	EObject [org.eclipse.emf.ecore.EObject]

Ecore Metamodel

```
<xs:complexType name="libraryType">
  <xs:sequence>
      <xs:element name="book" type="bookType" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element name="customer" type="customerType" maxOccurs="unbounded" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="bookType">
  <xs:sequence>
    <xs:element name="name" type="xs:ID"/>
    <xs:element name="title" type="xs:string"/>
    <xs:element name="author" type="nameType"/>
    <xs:element name="pages" type="xs:int"/>
    <xs:element name="bookInfo" type="bookInfoType"/>
  </xs:sequence>
  <xs:attribute name="isbn" type="isbnType" use="required"/>
</xs:complexType>
<xs:complexType name="customerType">
  <xs:sequence>
      <xs:element name="firstName" type="xs:string"/>
      <xs:element name="lastName" type="xs:string"/>
      <xs:element name="borrowedBookId" type="xs:IDREF"/>
  </xs:sequence>
</xs:complexType>
```

EReference to EObject

XML Schema

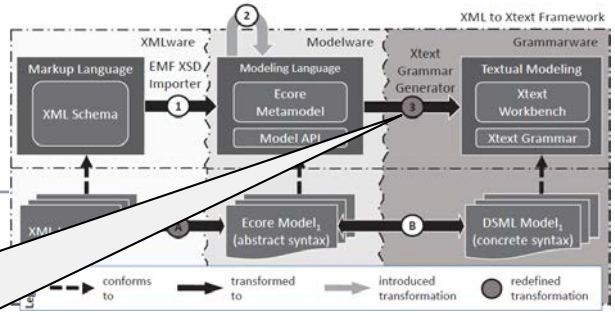
```
BookType returns BookType:
    'Book'
    name=ID0
    '{'
        'title' title=STRING
        'author' author=NameType
        'pages' pages=Int0
        'isbn' isbn=IsbnType
        ('bookInfo' bookInfo=BookInfoType)?
    '}';

CustomerType returns CustomerType:
    'Customer'
    '{'
        'firstName' firstName=STRING
        'lastName' lastName=STRING
        ('borrowedBookId' borrowedBookId=[ecore::EObject|IDREF])?
    '}';

ID0 returns type::ID:
    ID;

IDREF returns ecore::EString:
    ID;
```

Xtext Grammar

Does allow references and enables corresponding Editor Content Assist

XMLText

Customizing Concrete Syntax



Syntax customization on generic level.

```
FEATURE_MEMBER_OPEN_TS = ‘{‘
FEATURE_MEMBER_CLOSE_TS = ‘}’
SIMPLE_TERMINAL_FOR_MULTI_FEATURES = true
DROP_TYPE_IN_TERMINAL = true
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<library xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="libra
  <book isbn="0123456789">
    <name>book1</name>
    <title>Hamlet</title>
    <author>William Shakespeare</author>
    <pages>400</pages>
    <bookInfo>
      <ANY-ELEMENT/>
    </bookInfo>
  </book>
  <book isbn="0061122416">
    <name>book2</name>
    <title>The Alchemist</title>
    <author>Paolo Coelho</author>
    <pages>197</pages>
    <bookInfo>
      <bookReview rating="5" reviewText="I really liked this book">Book Review By Peter
        <helpfulRating>17%</helpfulRating>
      </bookReview>
    </bookInfo>
  </book>
  <customer>
    <firstName>John</firstName>
    <lastName>Denver</lastName>
    <borrowedBookId>book2</borrowedBookId>
  </customer>
</library>
```

XML Instance

```
Library {
    Book book1 {
        title "Hamlet"
        author "William Shakespeare"
        pages 400
        isbn "0123456789"
    },
    Book book2 {
        title "The Alchemist"
        author "Paolo Coelho"
        pages 197
        isbn "0061122416"
        BookInfo {
            "bookReview": "Book Review By Peter"
            "rating"= "5",
            "reviewText"= "I really liked this book"
            {
                "helpfulRating": "17%"
            }
        }
    }
    Customer {
        firstName "John"
        lastName "Denver"
        borrowedBookId book2
    }

}
```

DSML model