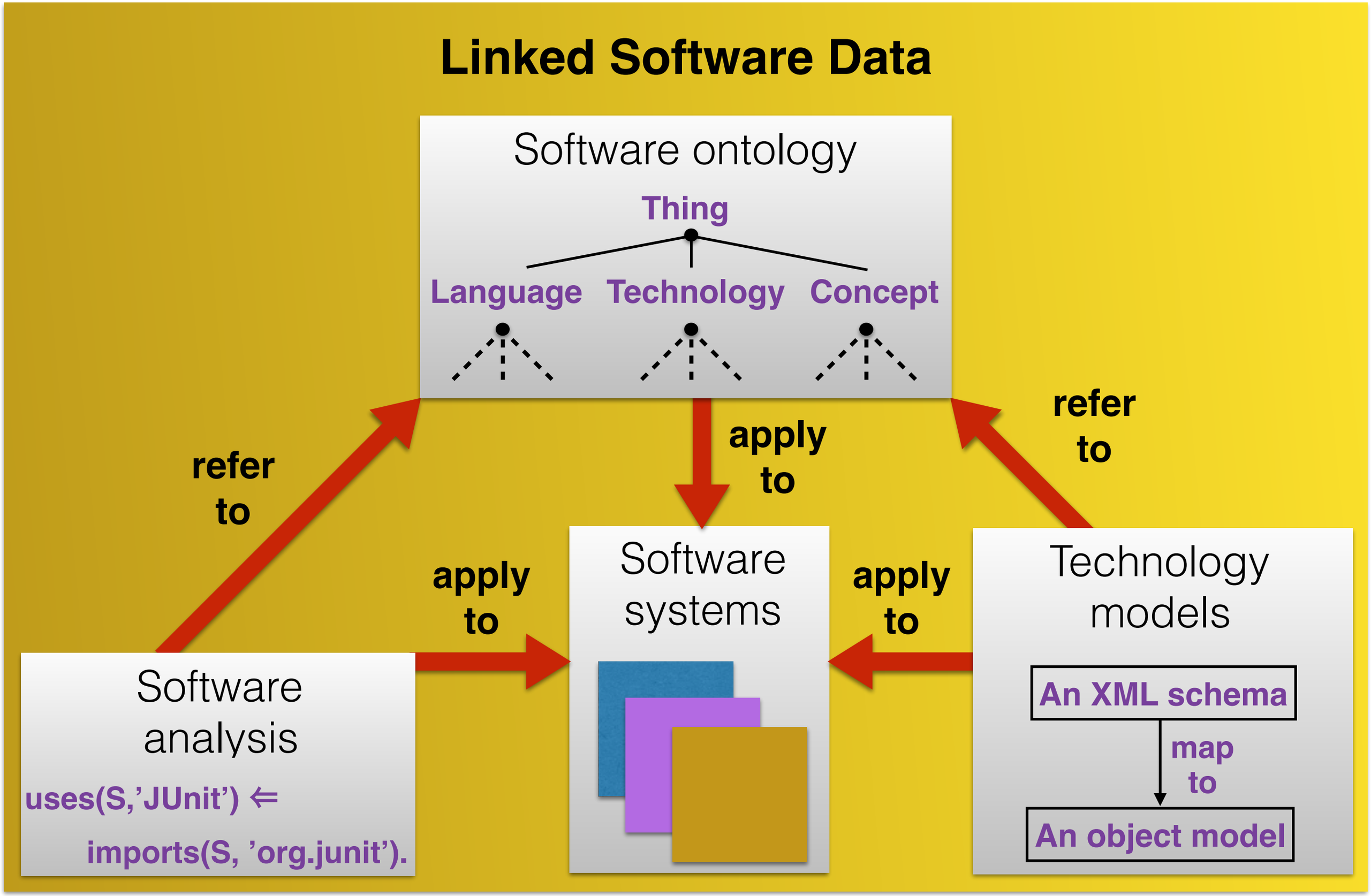
The SoLaSoTe ontology for software languages, technologies, and concepts

Ralf Lämmel

Software Languages Team, University of Koblenz Landau

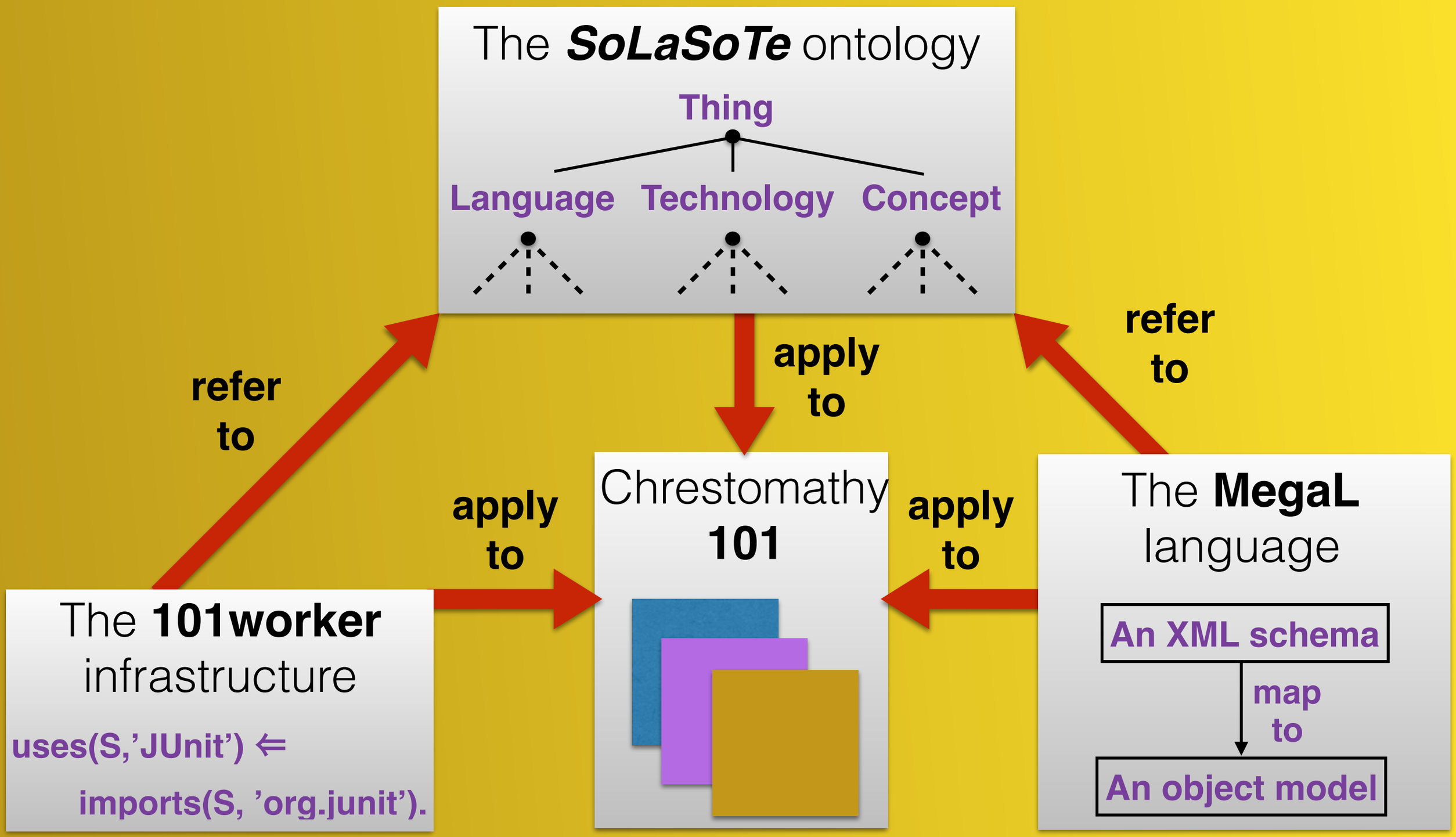
Joint work with Andrei Varanovich and Martin Leinberger

Software ontologies in context



SoLaSoTe in context

Linked Software Data with 101explorer and 101triples



Why is this important?

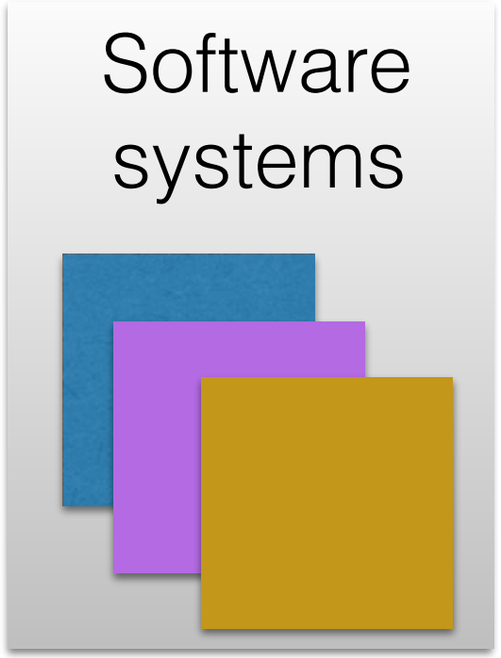
XPATH TXL Sesame JPA
Jena Rose JDBC EMF.gen XText
jDOM JAXB Jersey RDF(S) UTF8 Jena ODM
XSD MOF JavaBeans Teneo
UML BNF
Stratego VLDB
xerces GWT Json saxon OWL
OCL sax
RDFS OWL Ecore
Rest RDF ArgoUML OMG
MySQL XMI ORACLE JMI
JMI EMF XSD
xalan ODBC SparQL XMLSpy Yacc RDFa LALR Prolog
XSLT JAXP Protégé
CFG SBVR DOM Java ATOM ER
SQL DDL Dragan SQL Antlr QVT
TENEO XLST Awk DTD TCS
sed Saxon
grep ASCII XSD
Hibernate XQuery

language & technology plethora

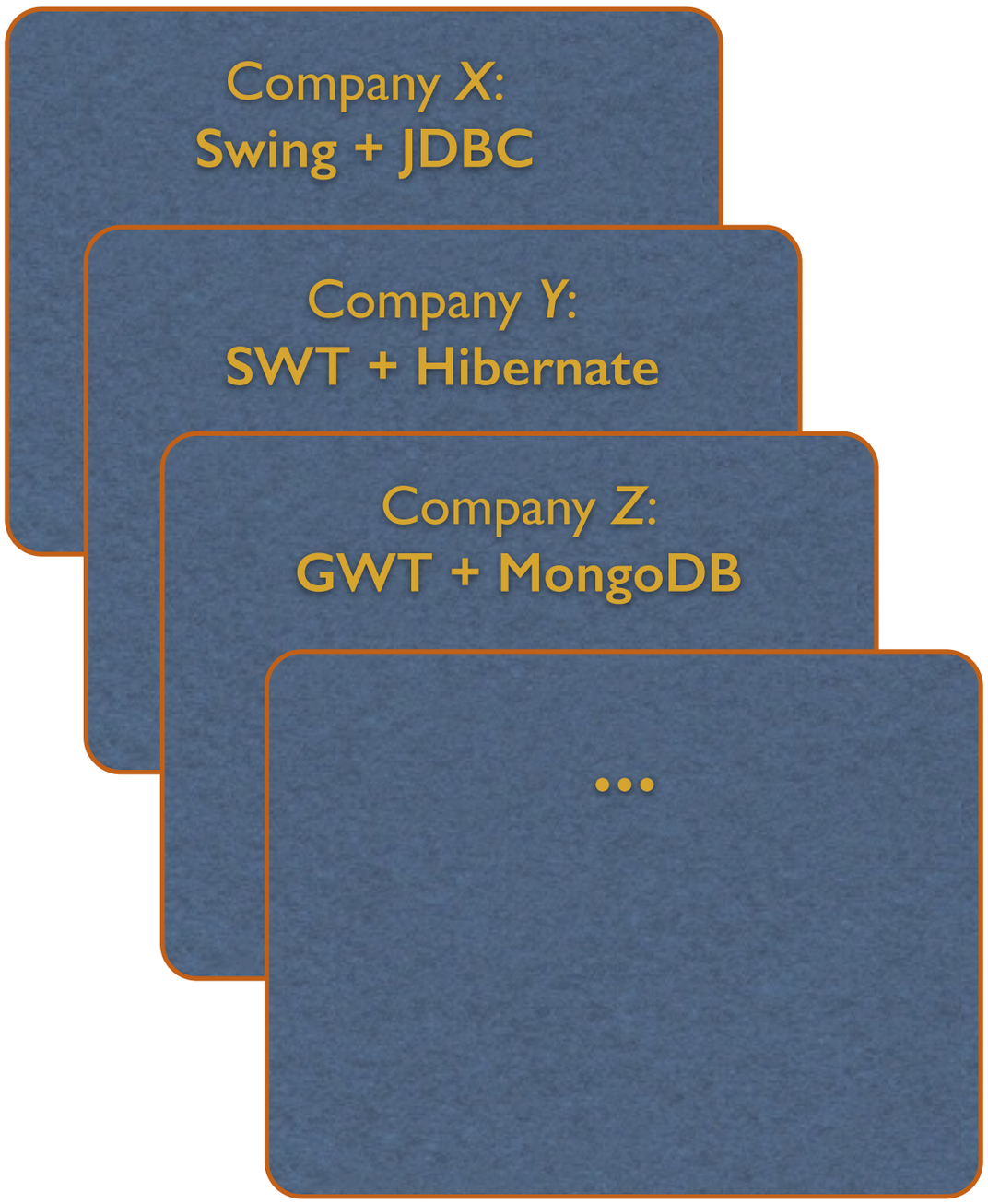
...because of the

The software chrestomathy
101*companies*

A quick intro



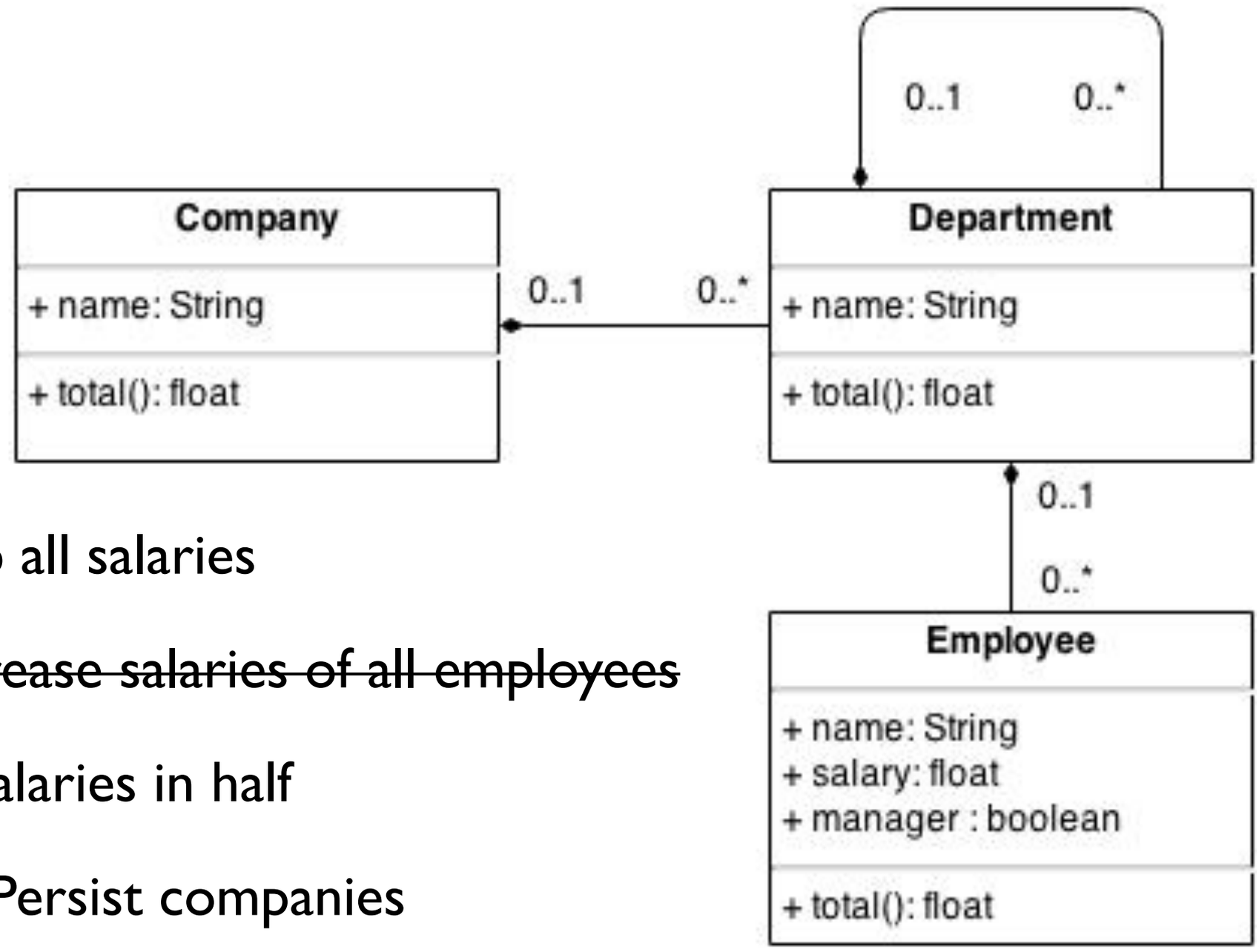
The 101companies project



Different implementations of the same system **varying languages, technologies, and concepts**.

For what it matters, it's an **HRMS**: a human resources management system.

Data & functionality of 101's system



- **Total**: Sum up all salaries
- ~~**Increase**: Increase salaries of all employees~~
- **Cut**: Cut all salaries in half
- **Persistence**: Persist companies
- **Editing**: GUI support for editing companies
- …

A Prolog-based implementation

```
% Total all salaries in a company
total(X,R) :- collect(getSalary,X,L), sum(L,R).

% Helper for salary extraction
getSalary(employee(_,_,S),S).

% Higher-order traversal for accumulation
collect(P,X,L) :-
   apply(P,[X,Y]) ->
      L = [Y];
      X =.. [_|Xs],
      maplist(collect(P),Xs,Yss),
      append(Yss,L).
```

Scrap Your Boilerplate—Prologically!
PPDP 2009

101's features

feature	headline
Total	Sum up the salaries of all employees.
Cut	Cut the salaries of all employees in half.
Hierarchical_company	A data model for companies with nested departments
Parsing	Parse an external format for company data
Closed_serialization	Closed serialization for company data
Unparsing	Unparse company data to an external format
Browsing	UI support for browsing company data
Depth	Compute the nesting depth of departments
Editing	UI support for editing company data
Web_UI	A web-based user interface
Distribution	Distribution of company data and operations
Company	Basic company structure
Flat_company	A data model for flat companies
Persistence	Persistence for company data
Mapping	Mapping company data across technological spaces
Open_serialization	Open serialization for company data

101's implementations

contribution	headline
haskellEngineer	Basic software engineering for Haskell
haskellComposition	Data composition in Haskell with algebraic data types
mySqlMany	A MySQL database with SQL scripts
javaComposition	Object composition in Java
antlrAcceptor	An ANTLR-based acceptor for textual syntax
antlrLexer	Lexer-based text processing with ANTLR
javaInheritance	Class inheritance in Java
antlrParser	An ANTLR-based parser with semantic actions
antlrObjects	ANTLR-based object-text mapping for Java
antlrTrees	Parsing and tree walking with ANTLR
jdom	XML processing with Java's JDOM API
jaxbComposition	Object-XML mapping with JAXB of the Java platform

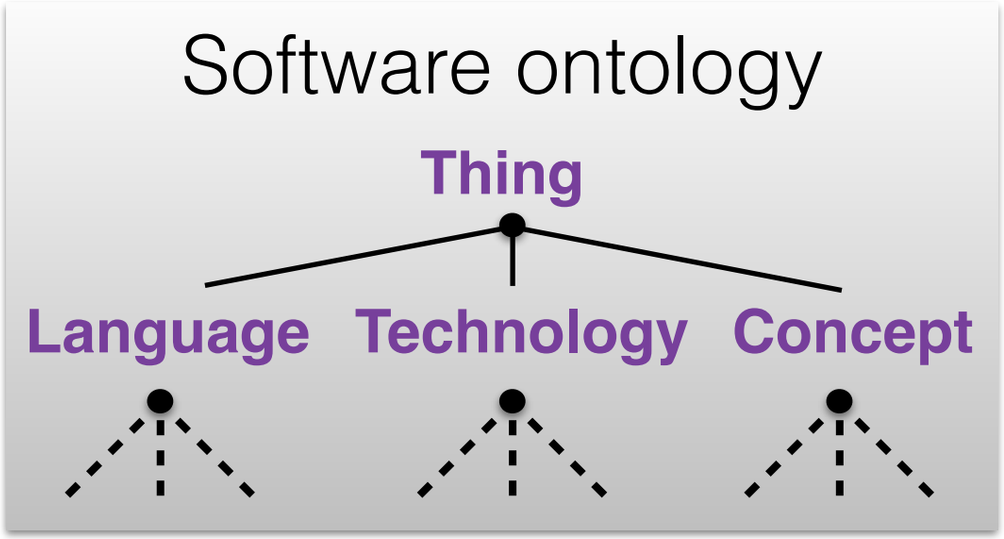
...

The 101companies project

- **101repo** — **GitHub-based source code repo**
- **101wiki** — **Semantic wiki-based documentation**
- **101worker** — **Infrastructure for automated analysis**
- **101explorer** — **Linked Data explorer**
- **101triples** — **Linked Data SPARQL endpoint**
- **SoLaSoTe** — **Underlying ontology**
- **MegaL** — **Language for technology models**

The software ontology
SoLaSoTe

A quick intro



SoLaSoTe

What kind of ontology is it?

- It's ***not*** a domain ontology.
  - A close example of a domain ontology:
    - An ontology for software testing
- It's ***not*** a foundational ontology.
  - We may use foaf and others, though.
- It's a ***core ontology***.
  - Applies to the field of software engineering & programming.

Types of SoLaSoTe's individuals

type	comment
onto:Concept	Software concepts
onto:Contribution	Contributions to the 101 project
onto:Contributor	Contributors to the 101 project
onto:Course	Courses on programming and software engineering
onto:Document	Documents in a broad sense
onto:Feature	Software features
onto:Language	Software languages
onto:Script	Scripts as units of a course
onto:Technology	Software technologies
onto:Theme	Containers of contributions
onto:Vocabulary	Containers of terms

For instance: software **languages**

language	headline
Java	An OO programming language
Haskell	A purely-functional programming language
XML	The extensible markup language
JavaScript	A multi-paradigm programming language for the web et al.
JSON	The JavaScript Object Notation for data exchange
SQL	Data definition and manipulation for relational databases
Python	A multi-paradigm programming language
...	

For instance: software **technologies**

technology	headline
Gradle	A build tool inspired by Ant and Maven
JUnit	A framework for unit testing for Java
Eclipse	An IDE for Java with a plug-in system
.NET	A library and runtime for programming languages on Windows
ANTLR	A parser generator with various language processing capabilities
GHC	A Haskell compiler
MySQL	A relational database management system
...	

For instance: software **concepts**

concept	headline
Web_programming	The domain of web application development
Algebraic_data_type	A type for the construction of terms
OO_programming	The object-oriented programming paradigm
Functional_programming	The functional programming paradigm
API	An interface for reusable functionality
Type_class	An abstraction mechanism for polymorphism
Software_system	A system of intercommunicating software components
...	

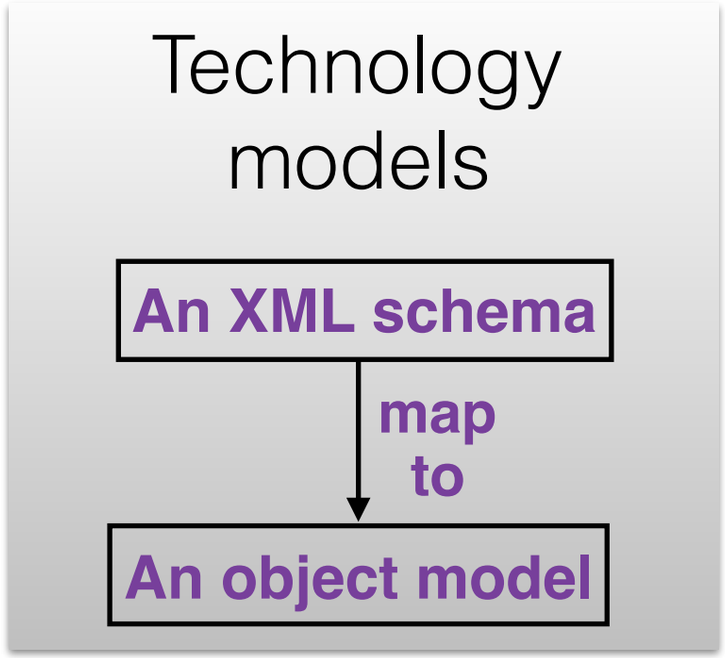
A SPARQL query sorting software concepts by popularity

```
SELECT ?concept ?headline (COUNT(?subject) AS ?count)
WHERE {
  ?concept a onto:Concept .
  ?concept onto:hasHeadline ?headline .
  ?subject ?predicate ?concept .
}
GROUP BY ?concept ?headline
ORDER BY DESC(?count)
```

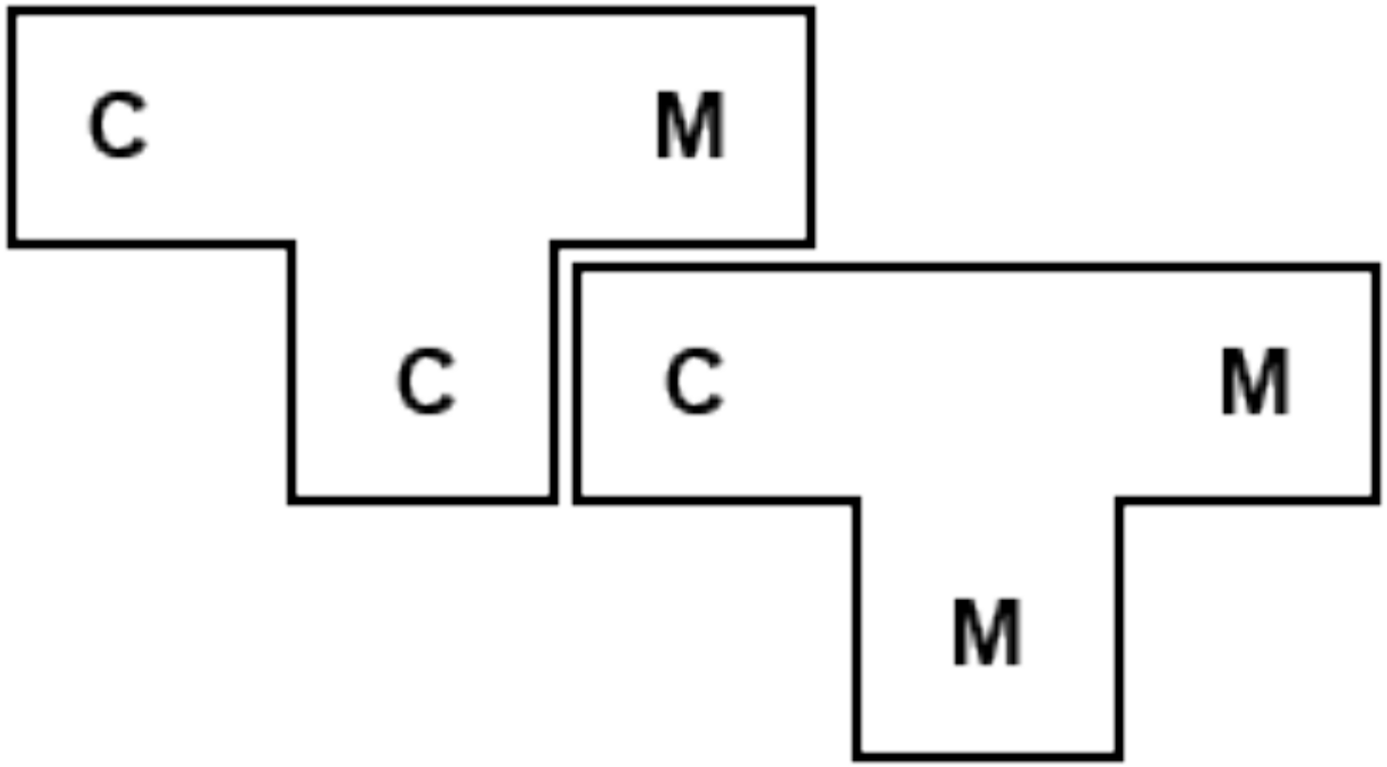
Thus, the realization of SoLaSoTe depends on RDF, RDFS (OWL), and SPARQL.

Technology modeling with ***MegaL***

A quick intro



A classic „technology model“ for bootstrapping a compiler

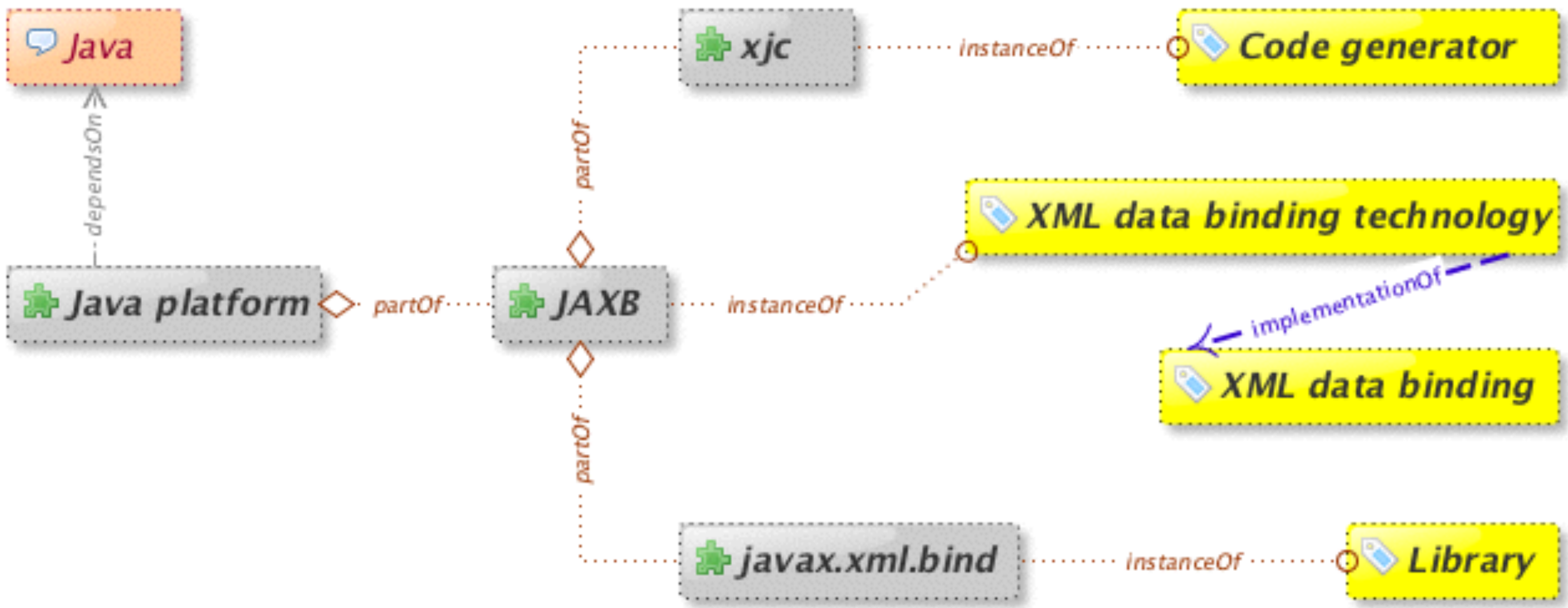


http://en.wikipedia.org/wiki/Tombstone_diagram

A general notion of **technology** model

- Technology models are „ER models“.
- Entities of interest
  - Software technologies and parts thereof, e.g., *Hibernate*
  - Software languages, e.g., *SQL*
  - Software artifacts, e.g., *O/R mapping file*
  - Software concepts, e.g., *persistence*
  - ...
- Relationships of interest
  - Conformance
  - Transformation
  - ...

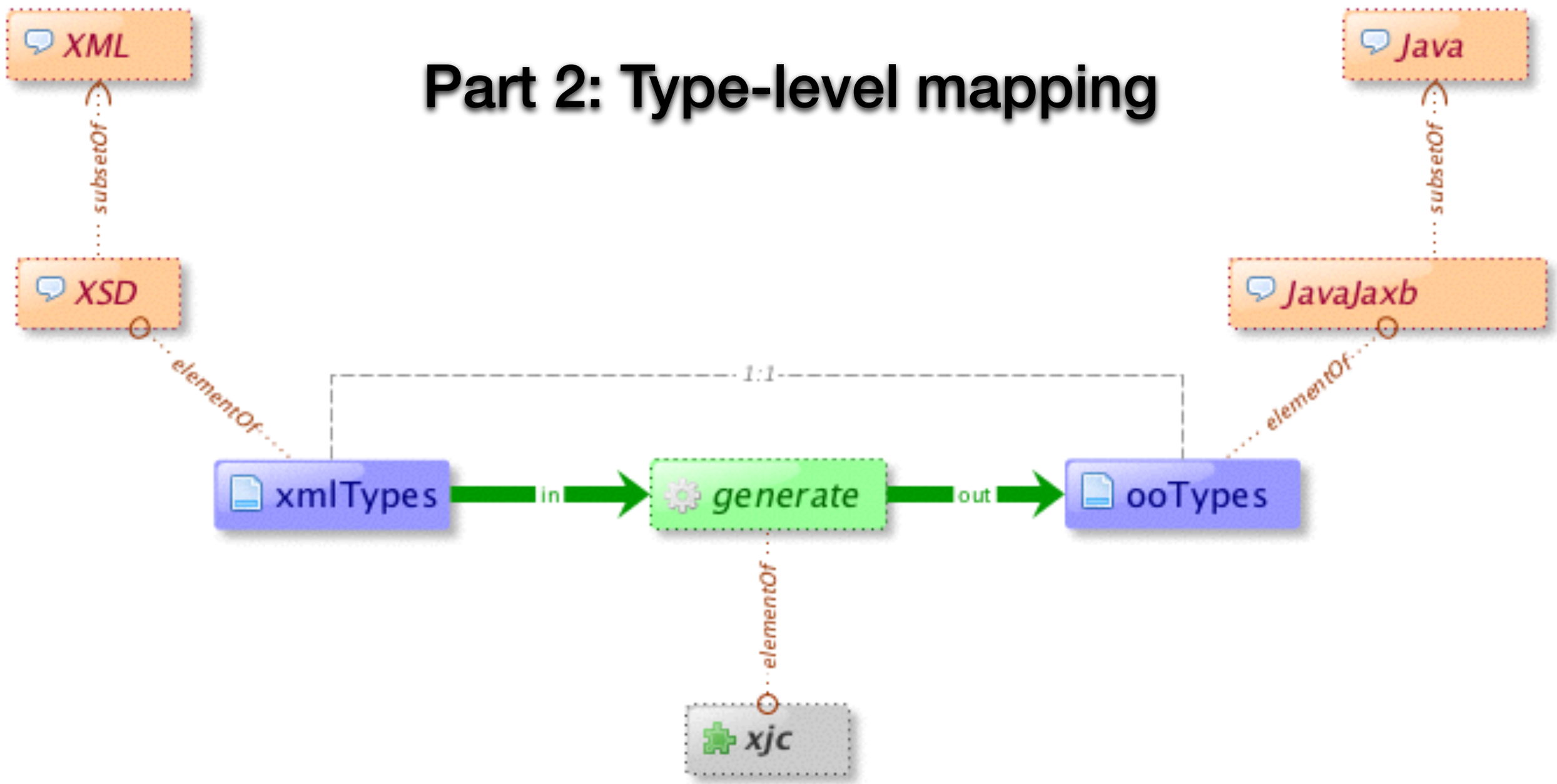
A technology model for JAXB (XML-data binding of the Java platform)



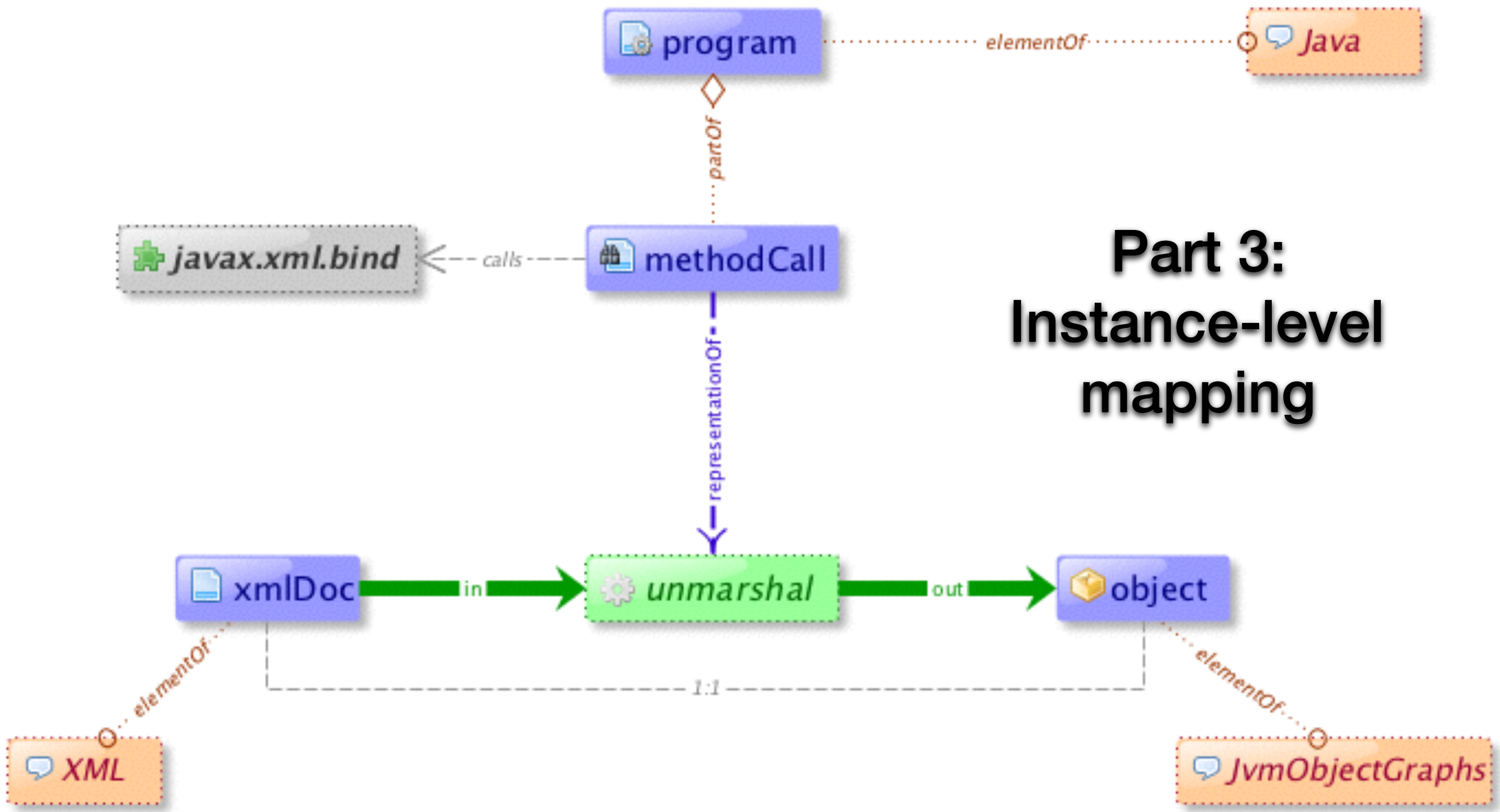
Part 1: Technology break-down and concepts

A technology model for JAXB (XML-data binding of the Java platform)

Part 2: Type-level mapping

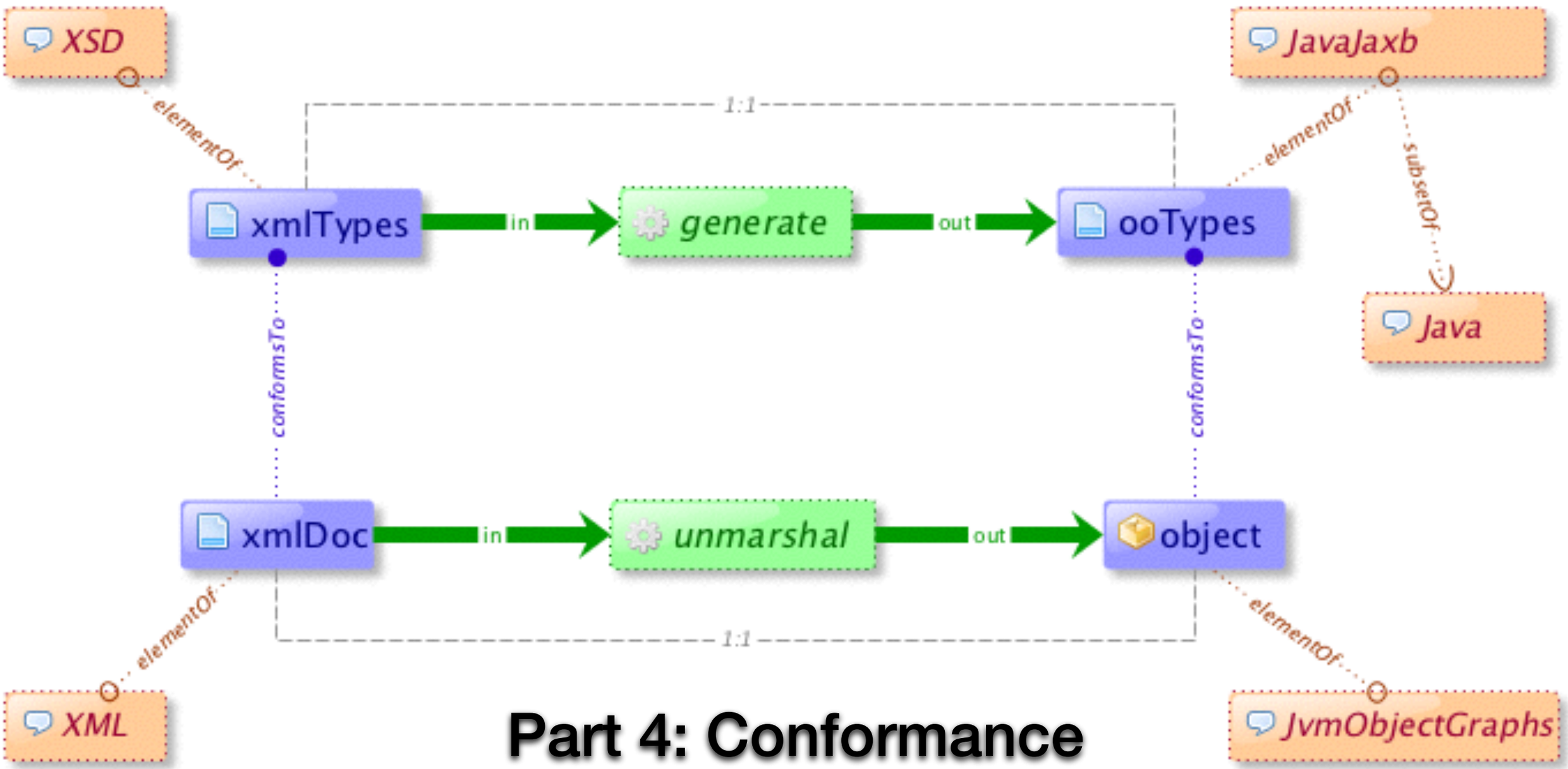


A technology model for JAXB (XML-data binding of the Java platform)



Part 3: Instance-level mapping

A technology model for JAXB (XML-data binding of the Java platform)



Part 4: Conformance

Software analysis with ***101worker*** *(incl. 101meta)*

A quick intro

Software analysis

imports(S,'org.junit')
⇐ uses(S, 'JUnit')

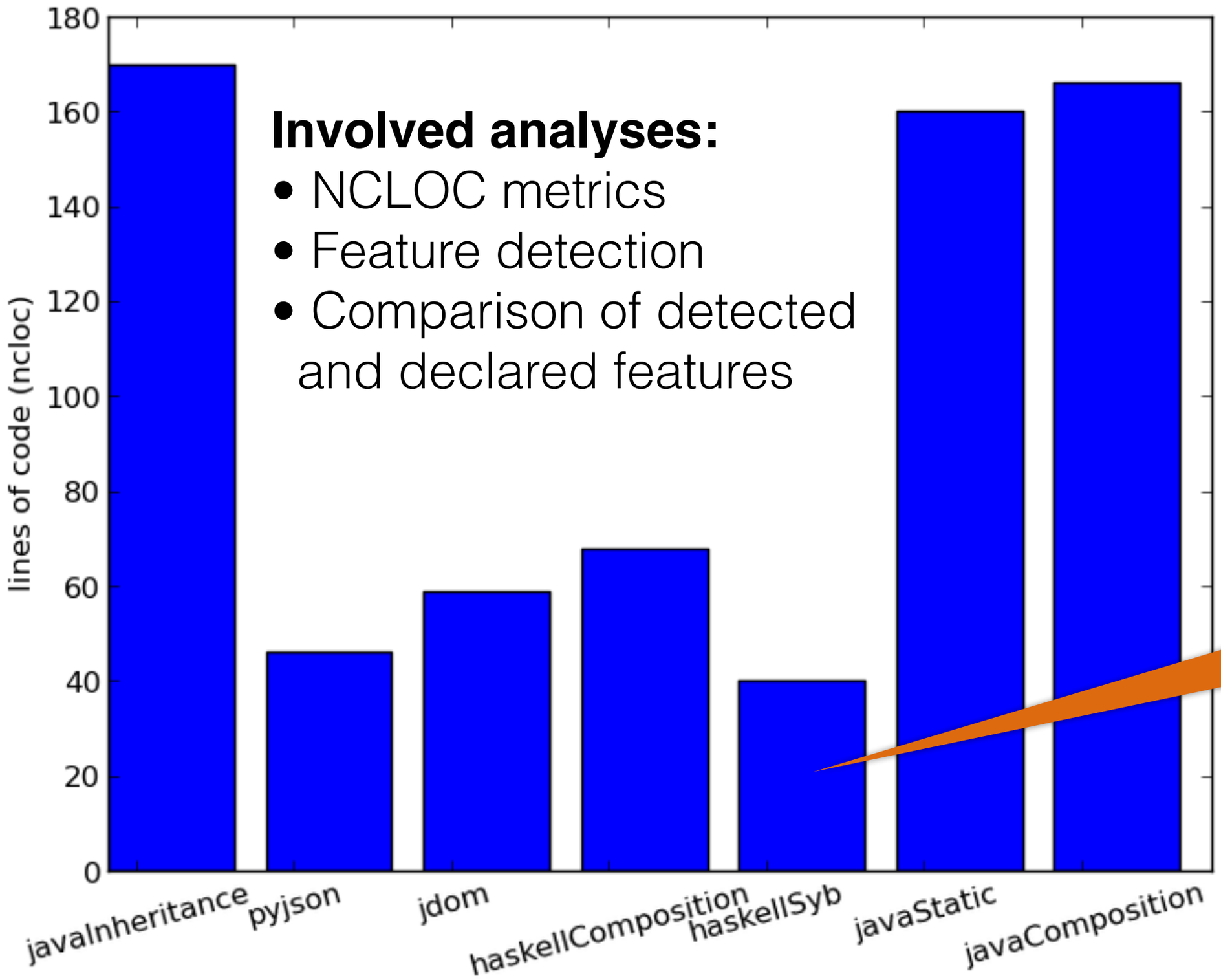
101meta — a rule-based language for metadata inference

```
uses(C, lang:Haskell) ⇐
    contrib(C),
    filePartOf(F, C),
    suffix(F, '.hs').
```

```
uses(C, tech:JUnit) ⇐
    contrib(C),
    filePartOf(F, C),
    imports(F, 'org.junit').
```

```
implements(C, feature:Total) ⇐
    contrib(C),
    filePartOf(F, C),
    tokenPartOf('total', F).
```

Comparison of feature implementations across languages, technologies, and styles



Involved analyses:

- NCLOC metrics
- Feature detection
- Comparison of detected and declared features

Haskell „wins“

Linked Software Data with ***101explorer and 101triples***

A quick intro

Linked software data

Use and enrich *Linked Data* principles:

- **Use URIs as names for things, e.g., languages.**
- **Make URIs HTTP resolvable.**
- **Provide useful info, e.g., schemas or source code.**
- **Provide data for humans & software: HTML, JSON, RDF.**
- **Include links to other resources, e.g., GitHub or 101wiki.**

Exploration of software data

A **video** going through these stages:

- 101's contribution *haskellStarter* on *101**wiki***
- Ditto's contribution source code on *101**repo***
- Various related resources on *101**explorer***
- The Haskell entity on *101**triples***
- An illustrative query on SPARQL endpoint

https://www.youtube.com/watch?v=0MkzWMPp1II

The software ontology
SoLaSoTe

Deep(er) dive

SoLaToSe aspects

- **Classification (instanceOf, isA)**
- **Relationships (uses, supports, …)**
- Containers (themes, vocabularies, …)
- Systems (101)
- Other resources (sameAs, …)
- **Metamodeling with validation**

Classification

```
ASK {
  lang:Java a onto:Language
}
```

Is ‚Java' a language?

true

```
SELECT ?type
WHERE {
  lang:Java rdf:type ?type .
  ?type a onto:Classifier
}
ORDER BY ?type
```

What are the supertypes of ‚Java'?

```
onto:OO_programming_language
onto:Programming_language
```

```
SELECT ?concept ?headline (COUNT(?subject) AS ?count)
WHERE {
  ?concept a onto:Concept .
  ?concept onto:hasHeadline ?headline .
  ?classifier a onto:Classifier .
  ?classifier onto:classifies ?concept .
  ?subject ?predicate ?concept
}
GROUP BY ?concept ?headline
ORDER BY DESC(?count)
```

What are popular classifiers?

concept
Algebraic_data_type
OO_programming
Functional_programming
API
Software_system
Client
Web_browser
...

Predicates for relationships

predicate	comment
basedOn	Reuse of systems
carries	Tagging of entities
dependsOn	Dependence
designedBy	Designer of a
developedBy	Developer of
illustrates	Chrestomath
implements	Systems im ementing descriptions
linksTo	Non-spe ic link to external resource
memberOf	Mem ership
mentions	Nonspecific r
moreComplexThan	Comparison
partOf	Whole-part
profile	Web page wi
reviewedBy	Reviewer of an entity
sameAs	Equivalence relative to external resource
similarTo	Similarity rel
supports	Instruments
uses	Use of instru
varies	Similarity of

For example:
feature:**Hierarchical**_company
moreComplexThan
feature:**Flat**_company

For example:
tech:**ghc**
partOf
tech:**Haskell_platform**

For example:
tech:**Ruby_on_Rails**
supports
concept:**REST**

Predicates for relationships

predicate	domain	range
onto:basedOn	onto:System	onto:System
onto:carries	onto:Entity	onto:Tag
onto:dependsOn	onto:Entity	onto:Entity
onto:designedBy	onto:Entity	foaf:Person
onto:developedBy	onto:Entity	foaf:Person
onto:illustrates	onto:Description	onto:Instrument
onto:implements	onto:System	onto:Description
onto:linksTo	onto:Entity	rdfs:Literal
onto:memberOf	onto:Entity	onto:Container
onto:mentions	onto:Entity	onto:Entity
onto:moreComplexThan	onto:Entity	onto:Entity
onto:partOf	onto:Entity	onto:Entity
onto:profile	onto:Contributor	rdfs:Literal
onto:reviewedBy	onto:Entity	foaf:Person
onto:sameAs	onto:Entity	rdfs:Literal
onto:similarTo	onto:Entity	rdfs:Literal
onto:supports	onto:Instrument	onto:Instrument
onto:uses	onto:System	onto:Instrument
onto:varies	onto:System	onto:System

Query: *Arrange lecture in a course*

```
SELECT DISTINCT
    ?course
    (COUNT(?prerequisites) AS ?count)
WHERE {
    ?course onto:memberOf course:Lambdas_in_Koblenz .
    OPTIONAL { ?course onto:dependsOn+ ?prerequisites }
}
GROUP BY ?course
ORDER BY ?count
```

course	count
First_steps_in_Haskell	0
Basic_software_engineering_for_Haskell	1
Searching_and_sorting_in_Haskell	2
Basic_data_modeling_in_Haskell	3
Higher-order_functions_in_Haskell	4
Type-class_polymorphism	5
Functional_data_structures	5
Functors_and_friends	6
Generic_functions	7
Unparsing_and_parsing_in_Haskell	7
Monads	8

Arrange scripts (lectures) in an order that respect the lecture dependencies.

Metamodeling

- Semi-structured data (semantic properties) on wiki
- WIKI to JSON
- JSON-based schema DSL (close to RDFS)
- JSON to TTL
- Template-based SPARQL queries for validation
  - Uniqueness of entity types
  - CWA for properties

Entity type *concept*

```
{
    "@id": "Concept",
    "@type": [
        "Entity",
        "Instrument"
    ],
    "comment": "Software concepts",
    "wikialias": [
        "Software_concept"
    ],
    "properties": [
       {
            "property": "conceptMemberOf",
            "super": "memberOf",
            "range": "Vocabulary",
            "minCardinality": "0",
            "comment": "Concepts collected in vocabularies"
       }
    ]
}
```

Entity type *contributor*

```
{
    "@id": "Contributor",
    "@type": [
        "foaf:Person",
        "Entity"
    ],
    "comment": "Contributors to the 101 project",
    "properties": [
        {
            "property": "profile",
            "range": "rdfs:Literal",
            "minCardinality": "1",
            "comment": "Web page with info about contributor"
        }
    ]
}
```

M. d'Aquin and A. Gangemi. Is there beauty in ontologies? *Applied Ontology*, 6(3):165–175, 2011.

Towards a beautiful ontology

- Structure
- Conceptual coverage
- Conceptual task
- Pragmatic sustainability

M. d'Aquin and A. Gangemi. Is there beauty in ontologies? *Applied Ontology*, 6(3):165–175, 2011.

Structure: Reusing foundational ontologies; being designed in a principled way; being formally rigorous; implementing also non-taxonomic relations; following strictly an evaluation methodology; being modular, or embedded in a modular framework.

M. d'Aquin and A. Gangemi. Is there beauty in ontologies? *Applied Ontology*, 6(3):165–175, 2011.

Conceptual coverage: Providing important reusable distinctions; having a good domain coverage; implementing an international standard (e.g., ISO or WHO); providing an organization to unstructured or poorly structured domains.

M. d'Aquin and A. Gangemi. Is there beauty in ontologies? *Applied Ontology*, 6(3):165–175, 2011.

Conceptual task: Being oriented at an explicit task; having spelled out requirements from scenarios; being based on competency questions. Social sustainability: Being the result of an evolution (many revisions); having wide usage or acceptance; having commercial impact; being recommended by industry; implementing scientific knowledge.

M. d'Aquin and A. Gangemi. Is there beauty in ontologies? *Applied Ontology*, 6(3):165–175, 2011.

Pragmatic sustainability: Having applications built on top of it; having successful personal experience in building apps with it; designed for efficient query answering; maintaining original expressivity of data, and improving or enriching it; able to get rid of clunky constructs or to overcome expressivity limitations; being well documented; solving other technical aspects.

RDFS' inference vs. *SPARQL's* validation

RDFS

```
onto:contribUsesLang
  rdfs:type rdfs:Property ;
  rdfs:subPropertyOf onto:uses ;
  rdfs:comment "Use of languages by contributions" ;
  rdfs:domain onto:Contribution;
  rdfs:range onto:Language .
```

Semantics: if a resource is the subject of ‚contribUsesLang', then it is of type `Contribution'.

SPARQL

```
SELECT ?x {
  ?x onto:contribUsesLang ?y .
  FILTER NOT EXISTS { ?x sesame:directType onto:Contribution }
}
```

Semantics: find resources that are subjects of ‚contribUsesLang' without being of declared type `Contribution'.

OWL's consistency vs. *SPARQL's* validation

OWL

```
<owl:AllDisjointClasses>
<owl:members rdf:parseType="Collection">
  <owl:class rdf:about="http://101companies.org/ontology#Language"/>
  <owl:class rdf:about="http://101companies.org/ontology#Technology"/>
  <owl:class rdf:about="http://101companies.org/ontology#Concept"/>
  ...
</owl:members>
</owl:AllDisjointClasses>
```

Merely a declaration of a consistency requirement without standardized reporting semantics

SPARQL

```
SELECT ?entity ?t1 ?t2 {
  ?entity a ?t1 .
  ?entity a ?t2 .
  FILTER (?t1 != ?t2 && ?t1 != onto:Entity && ?t2 != onto:Entity) .
  ?t1 rdfs:subClassOf onto:Entity .
  ?t2 rdfs:subClassOf onto:Entity .
  FILTER NOT EXISTS { ?t1 a onto:Classifier } .
  FILTER NOT EXISTS { ?t2 a onto:Classifier }
}
```

An operational query for entities with more than one entity type

Summary of contributions

- The first core ontology for software languages, technologies, and concepts at an abstraction level for software engineers and programmers useful for understanding, comparing, or learning about such entities.
- An advanced case study of ontology development in the context of advanced ecosystem (wiki, worker, explorer, …) with coverage of authoring, exploring, and validation capabilities for authors and users.
- The use of a software chrestomathy as the driving force in developing the ontology (such as inclusion of entities and new kinds of relationships).

Future research

- **Community process for ontology evolution**
- **Ontology integration with Wikipedia et al.**
- **Help developers (akin to StackOverflow)**
- **Programming models for Linked Data**
- **Validation / mapping for ontologies**
- **More advanced reasoning than in SoLaSoTe**

Further reading

End of talk

- R. Lämmel. Software chrestomathies. Sci. Comput. Program., 2013
- J.-M. Favre, R. Lämmel, T. Schmorleiz, and A. Varanovich. 101companies: A Community Project on Software Technologies and Software Languages. TOOLS 2012.
- J.-M. Favre, R. Lämmel, and A. Varanovich. Modeling the Linguistic Architecture of Software Products. MODELS 2012.
- R. Lämmel and A. Varanovich. Interpretation of Linguistic Architecture. ECMFA 2014.
- J.-M. Favre, R. Lämmel, M. Leinberger, T. Schmorleiz, and A. Varanovich. Linking Documentation and Source Code in a Software Chrestomathy. WCRE 2012.
- R. Lämmel, M. Leinberger, T. Schmorleiz, and A. Varanovich. Comparison of feature implementations across languages, technologies, and styles. CSMR-WCRE 2014.
- M. Leinberger. Enhancement of a software chrestomathy for open linked data. MSc Thesis. Universität Koblenz-Landau. 2013.

See the PPDP 2014 paper.