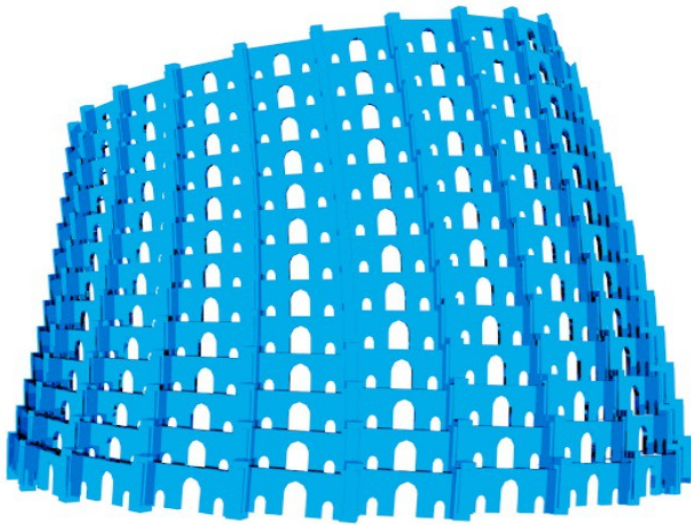


MegaL-Text

A natural language description

Marcel Heinz
Software Languages Team
University of Koblenz-Landau



SOFTLANG

© 2017 Software Languages Team

We have a problem!

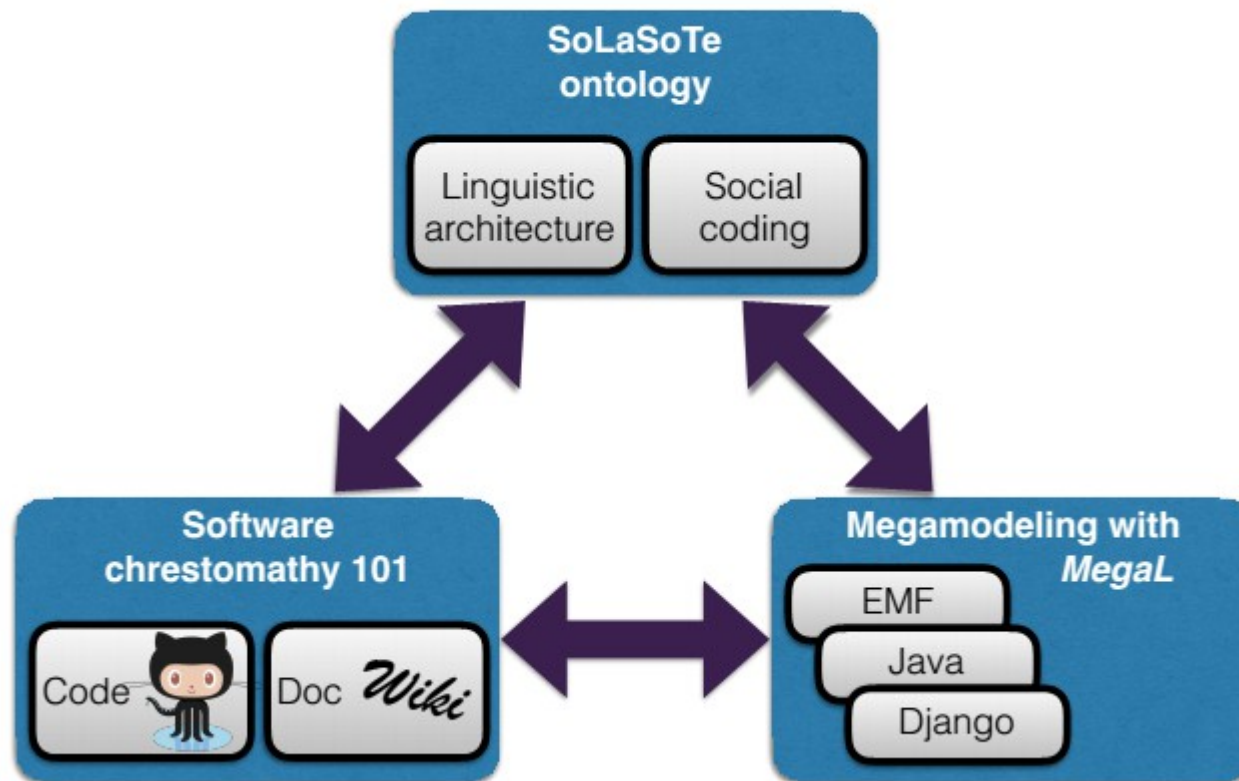
- Too many technologies, not enough time to master them all...



Consequences

- Vendor lock in (dependency on a software vendor)
- Missing Expertise
- Exhaustion
- Job-Security?
- High costs for introducing a new technology
-

SoLaSoTe Process



MegaL

- MegaL is short for 'Megamodeling Language', where a model describes entities in the context of software development and their relationships from a conceptual perspective.

Megal-Text

- Textual syntax.
- Stable, but minor evolution might happen.
- Newest vocabulary diverges from the vocabulary in papers.

Modularization

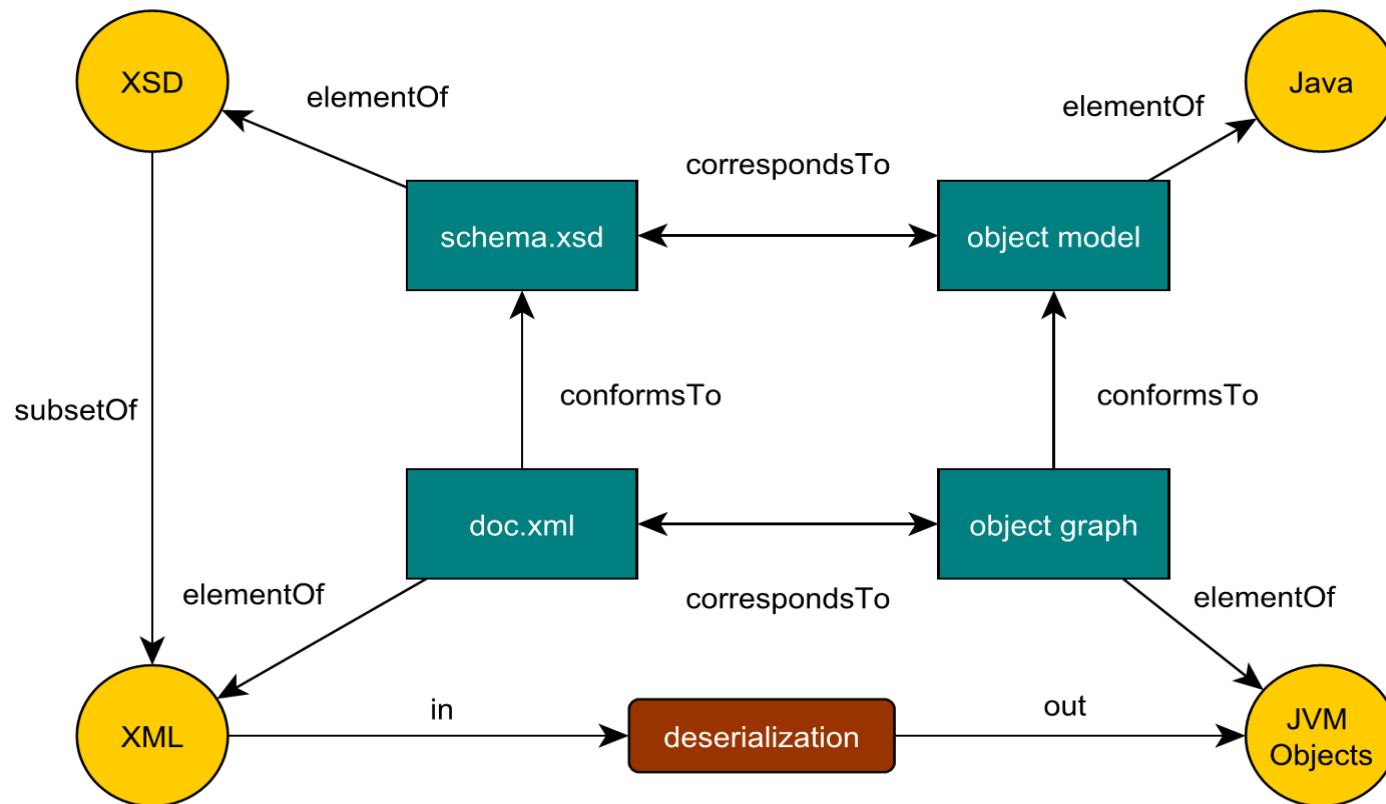
- To enable the reuse of facts every technology model is composed of various modules.
- Every module starts with a module name.
 - e.g., module java.JavaPlatform
- To reuse facts, you can import a module in another module.
 - e.g., import java.JavaPlatform

Path Resolution

- The name of the module can be resolved to a path.
 - E.g., the name `java.xml.JAXB` can be resolved to the File '`JAXB.megal`' in the folder '`xml`' in the folder '`java`'. Here, the resolution process starts at the parent of '`java`'.

An Abstract Technology Model

- Imagine a conceptual model for XML Binding technology in Java.



Prelude

- The Prelude module contains all subtypes and possible relationships.
- It represents the ground truth for the vocabulary.
- It is imported automatically, when processing a new model.
- The following slides shall make you acquainted with the prelude vocabulary.

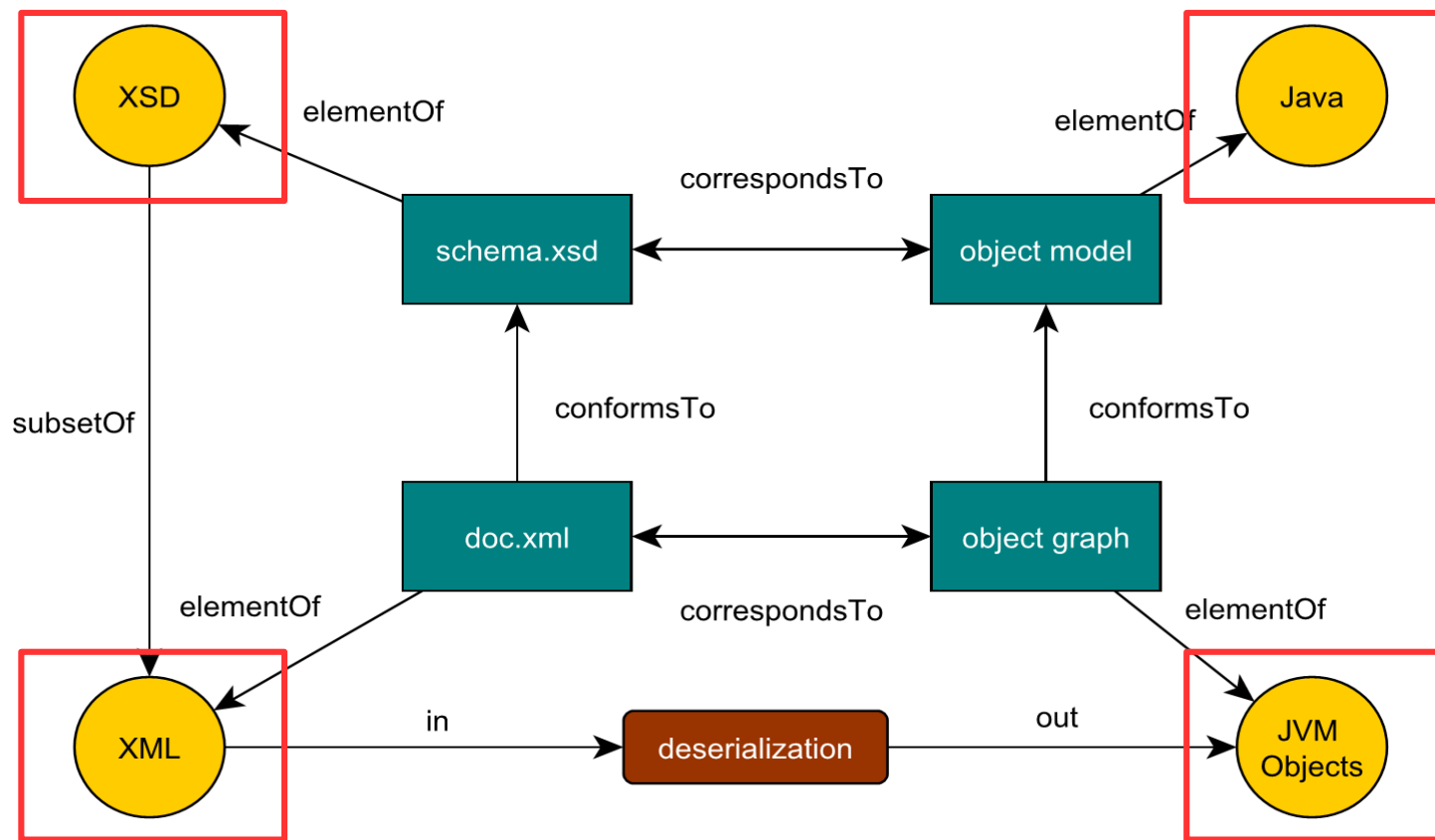
Language

- A language is a set of syntactic entities.
 - Language < Entity
- A language has one specific purpose.
 - Java : ProgrammingLanguage
 - XML : MarkupLanguage
 - XSD : SchemaLanguage
 - JVMObjects : ObjectGraph

Subsets and Embedding

- A language can be a subset of another language.
 - `subsetOf < Language # Language`
 - XSD `subsetOf` XML
 - SQLDDL `subsetOf` SQL
- A language can be embedded into another.
 - `embeddedInto < Language # Language`
 - EmbeddedSQL `embeddedInto` Java
 - EmbeddedJavaScript `embeddedInto` HTML5

An Abstract Technology Model



Artifact

- An artifact is a digital entity.
 - $\text{Artifact} < \text{Entity}$
- An artifact is element of a language.
 - $\text{elementOf} < \text{Artifact} \# \text{Language}$
 - `schema.xsd` elementOf XSD
 - `doc.xml` elementOf XML
 - `objectmodel` elementOf Java
 - `objectgraph` elementOf JVMObjects

Manifestation

- A manifestation describes the shape of an artifact at runtime.
 - `Manifestation < Entity`
 - `File < Manifestation`
 - `Transient < Manifestation`
- An artifact has a manifestation.
 - `manifestsAs < Artifact # Manifestation`
 - `doc.xml manifestsAs File`
 - `objectgraph manifestsAs Transient`

Definition and Conformance

- An artifact can define a language.
 - defines `< Artifact # Entity`
 - Java8Spec defines Java
 - *FSMLGrammar* defines *FSML*
- An artifact may be conform to another.
 - conformsTo `< Artifact # Artifact`
 - doc.xml conformsTo schema.xsd
 - objectgraph conformsTo objectmodel

Pattern

- A design pattern describes a reusable structure that addresses maintainability on the level of code.
 - DesignPattern < Entity
 - Subject-Observer : DesignPattern
- An architectural style describes a reusable structure that addresses maintainability on the level of components.
 - ArchitecturalStyle < Entity
 - Client-Server : ArchitecturalStyle

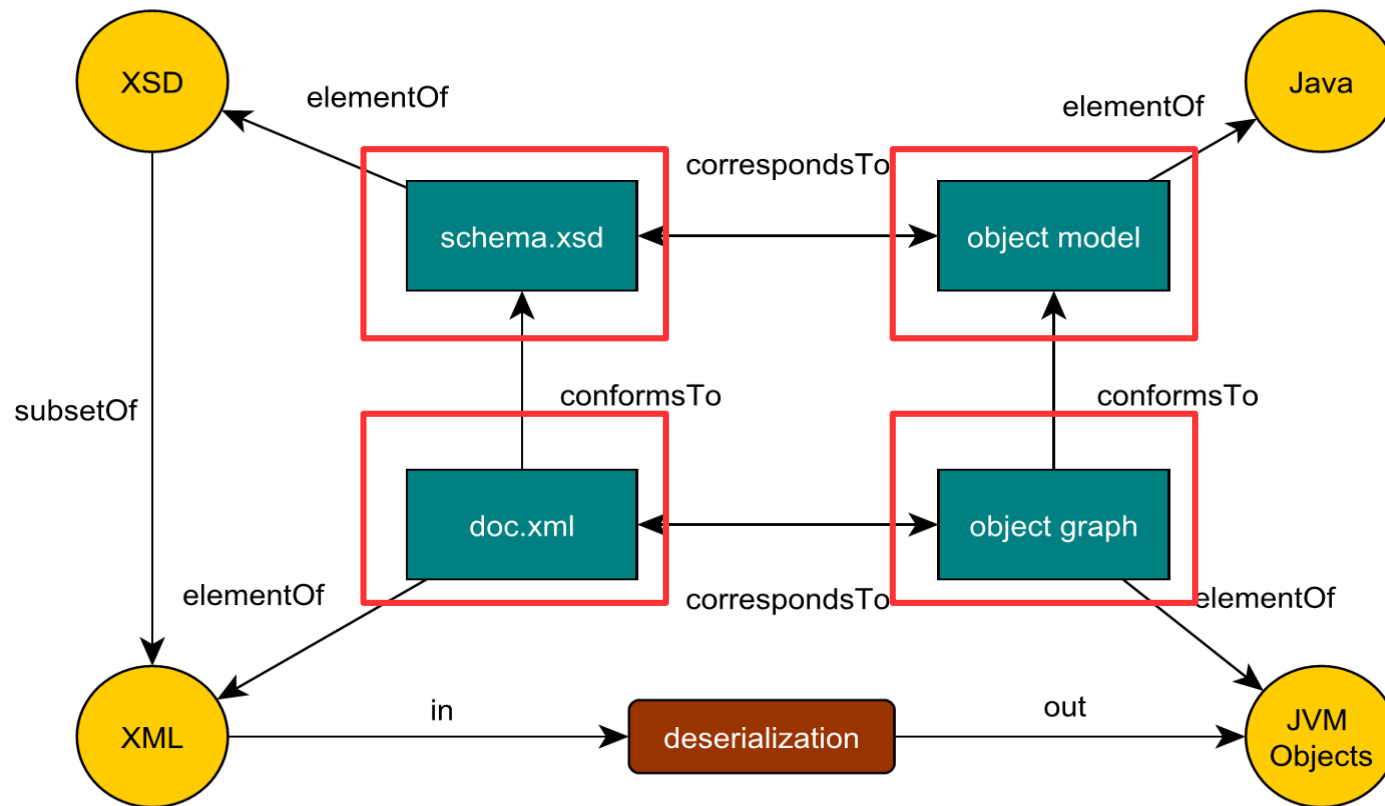
Role

- A design pattern or an architectural style may describe a set of participants, namely Roles.
 - Role < Entity
 - participantOf < Role # DesignPattern
 - participantOf < Role # ArchitecturalStyle
- An artifact plays a defined role.
 - hasRole < Artifact # Role
 - ?models.py hasRole MvcModel

Correspondence

- An artifact can correspond to another in the sense that it is semantically but not syntactically equal.
 - `correspondsTo < Artifact # Artifact`
 - `objectgraph correspondsTo doc.xml`
 - `objectmodel correspondsTo schema.xsd`

An Abstract Technology Model



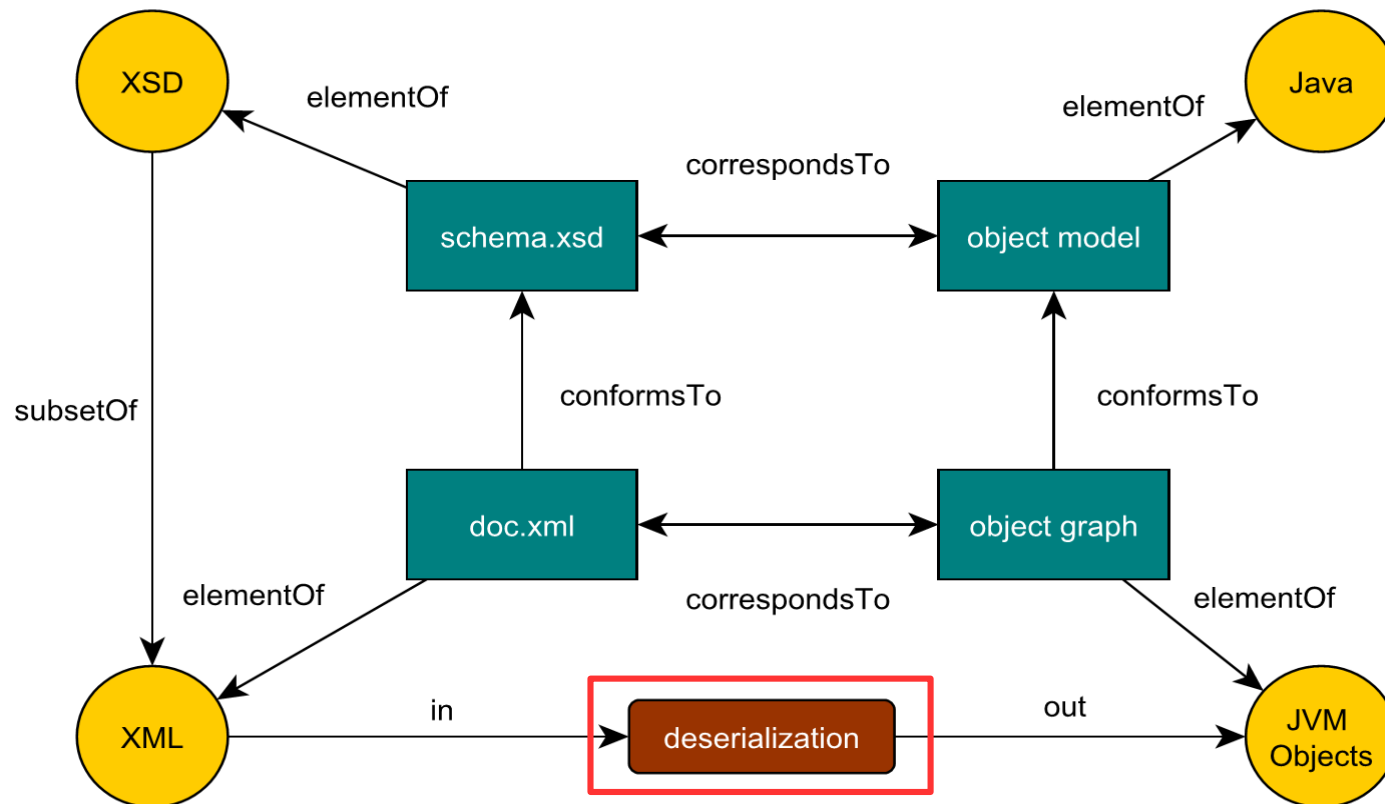
Function

- A function defines a mapping between an input and an output, which are elements of some language.
 - $\text{Function} < \text{Entity}$
- A function has a specific syntax.
 - `serialize : JsonObject -> XML`
 - `cutBy : XML # Int -> XML`
 - `totalAndCount : XML -> Int # Int`

Function Application

- A function application maps input to output.
 - `deserialization(doc.xml)|-> objectgraph`
 - `serialize(?aJavaObject)|->?anXMLFile`
 - `cutBy(?company1, 3)|-> ?company2`
 - `totalAndCountEmpl(?company)|-> (12000, 5)`

An Abstract Technology Model



Technology

- A technology is a reusable piece of software that has various use cases.
 - Technology < Entity

Technology Classification

- Various technology subtypes.
 - FacebookAPI : API
 - Django : Framework
 - JavaSwing : Library
 - Netbeans.RubyPlugin : Plugin
 - EMF.Core : Component
 - MicrosoftOffice2010 : SoftwareSuite
 - Eclipse : IDE
 - GHCi : Platform

Implementation

- A technology implements functions or languages and an artifact can only implement functions.
 - implements < Technology # Language
 - JAXB implements XML
 - implements < Technology # Function
 - JAXB implements serialize
 - implements < Artifact # Function
 - CutClass implements cut

Usage Scenario

- A system represents a set of artifacts realizing some use cases.
 - System < Entity
- Various kinds of systems.
 - Application < System
 - WebApplication < System
 - FileSystem < System

Let us raise the
level of abstraction
more!

Paradigm

- A programming paradigm is a way of thinking to have while programming in a language that facilitates it.
 - Paradigm < Entity

Paradigm Facilitation

- A language is classified by the paradigms that it facilitates.
 - facilitates < Language # Paradigm
 - Java facilitates ObjectOrientation
- Besides being a way of thinking it has implications on the kinds of:
 - Semantics
 - Type System
 - Syntax

Domain

- A programming domain is a field of study that may be covered by conferences and communities.
 - ProgrammingDomain < Entity
- A programming domain defines ...
 - ... common requirements and problems.
 - ... terminology.
 - ... ways for technologies and languages to support it.

Domain Support

- A language or a technology may be suited to support a programming domain.
 - supports < Language # ProgrammingDomain
 - Cobol supports DatabaseProgramming
 - Java supports GeneralPurposeProgramming
 - supports < Technology # ProgrammingDomain
 - ANTLR supports MetaProgramming
 - Eclipse supports GeneralPurposeProgramming

Technology Space

- A technological space is a conceptual entity that describes a set of:
 - application scenarios.
 - software languages.
 - programming tools such as IDEs
 - technologies
 - knowledge corpora
 - conferences and communities

Technology Space

- A technological space is a conceptual entity.
 - TechnologySpace < Entity
 - GrammarWare : TechnologySpace
 - JavaWare : TechnologySpace
- A technology can belong to a technological space.
 - belongsTo < Technology # TechnologySpace
 - JAXB belongsTo JavaWare
 - ANTLR belongsTo GrammarWare

Be careful here! It gets difficult to explain such relationships.

Abstract Process

- Commonly known processes where the realization depends on the used technologies and involved languages.
 - AbstractProcess < Entity
 - Serialization : AbstractProcess
 - Compilation : AbstractProcess
 - Transformation : AbstractProcess

Construct

- A construct is an idealized constellation of artifacts where the realization depends on the program's context, and involved languages and technologies.
 - Construct < Entity
 - Semaphore : Construct

Aspects

- Abstract solutions such as constructs may be an aspect of a way of thinking or field of study.
 - aspectOf < Construct # Paradigm
 - Semaphore aspectOf ConcurrentProgramming
 - aspectOf < Construct # ProgrammingDomain
 - QuasiQuotation aspectOf MetaProgramming
 - aspectOf < AbstractProcess # ProgrammingDomain
 - Compilation aspectOf MetaProgramming
 - aspectOf < Role # ProgrammingDomain
 - Grammar aspectOf MetaProgramming

Parthood

- There exist various types of parthood.
 - $\text{partOf} < \text{Artifact} \# \text{Artifact}$
 - $\text{partOf} < \text{Artifact} \# \text{Technology}$
 - $\text{partOf} < \text{Artifact} \# \text{System}$
 - $\text{partOf} < \text{Technology} \# \text{Technology}$
 - $\text{partOf} < \text{System} \# \text{System}$

Software Reuse

- Systems, technologies and artifacts can be reused. The using software depends on the used software.
 - $\text{uses} < \text{System} \# \text{System}$
 - $\text{uses} < \text{System} \# \text{Technology}$
 - $\text{uses} < \text{Technology} \# \text{Technology}$
 - $\text{uses} < \text{Artifact} \# \text{System}$
 - $\text{uses} < \text{System} \# \text{Technology}$

Used Language

- Since only artifacts can be real members of a language, one may still be interested in which languages are used in a composed piece of software.
 - uses < Artifact # Language
 - DatabaseManager uses EmbeddedSQL
 - uses < System # Language
 - MyApp uses ANT
 - uses < Technology # Language
 - JDBC uses Java

Usable Ideal Solutions

- Re-usable forms of solutions can be used in the sense of realization.
 - uses < System # DesignPattern
 - MyWebApp uses MVC
 - uses < System # ArchitecturalPattern
 - MyWebApp uses LayerArchitecture
 - uses < System # AbstractProcess
 - MyWebApp uses Serialization
 - uses < System # Construct
 - MyWebApp uses MessageQueue

Usable Ideal Solutions

- For technologies such facts are of interest to developers who want to improve a technology.
 - uses < Technology # DesignPattern
 - EMF uses FactoryPattern
 - uses < Technology # ArchitecturalPattern
 - Owncloud uses ClientServer
 - uses < Technology # AbstractProcess
 - Owncloud uses Synchronization
 - uses < Technology # Construct
 - Owncloud uses SynchronizationQueue

Facilitation

- A technology facilitates the use of a design pattern or architectural style or abstract process, in the sense of a deferred usage.
 - facilitates < Technology # DesignPattern
 - Django facilitates Model-View-Controller
 - facilitates < Technology # ArchitecturalPattern
 - Chef facilitates ClientServer
 - facilitates < Technology # AbstractProcess
 - ANTLR facilitates Parsing
 - facilitates < Technology # Construct
 - JMS facilitates MessageQueue

Syntactic sugar

- Based on RDF Turtle syntax:

```
models.py : Artifact  
  elementOf Python  
  hasRole MvcModel  
  manifestsAs File  
  partOf MyWebApp
```


Abstraction

- Instances concerned with general facts need to be linked to describing resources.
 - Django = „<https://www.djangoproject.com/>“
- Artifacts that should exist in any usage scenario do not need to be linked.
 - ?models.py : Artifact

Abstraction

- When describing a non-abstract usage scenario, artifacts need to be linked as well.
 - ContributionsController =
`"https://github.com/101companies/101rails/blob/326a894e38b164c1f1508a73b1954ff807e27cf3/app/controllers/contributions_controller.rb"`

Prescriptive vs Descriptive

- One should begin with stating facts in a prescriptive way without relating to a concrete use case.
- Several abstract entities are introduced first.
- A system can then be modeled in a separate module.
- The system module should make use of substitution.

Substitution

- When importing a module it is possible to substitute abstract entities by concrete ones.

```
import XMLBinding where {  
    MyClass substitutes ?objectModel  
    MyXML substitutes ?doc.xml  
}  
MyClass = "..."  
MyXML = "..."
```

Grouping

- As a modeling rule of thumb a human can perceive 7-11 model elements at once and not lose track.
- All facts in one module are split into groups.
- Every group has to start with a block comment.
- When creating groups, imagine creating a single diagram that only states an aspect.

Grouping Example

```
/* The microsoft office compatibility plugin enables a user to edit files
written in the new XML format with Office 2003. */
MicrosoftOfficeOpenXML : StylesheetLanguage
    = "https://en.wikipedia.org/wiki/Office\_Open\_XML"
    subsetOf XML
CFBF : FileFormat
    = "https://en.wikipedia.org/wiki/Compound\_File\_Binary\_Format"
MicrosoftOffice2003 implements CFBF
```

Constraints

- Are implemented in the Checker and are stated in natural language in the checker's readme.