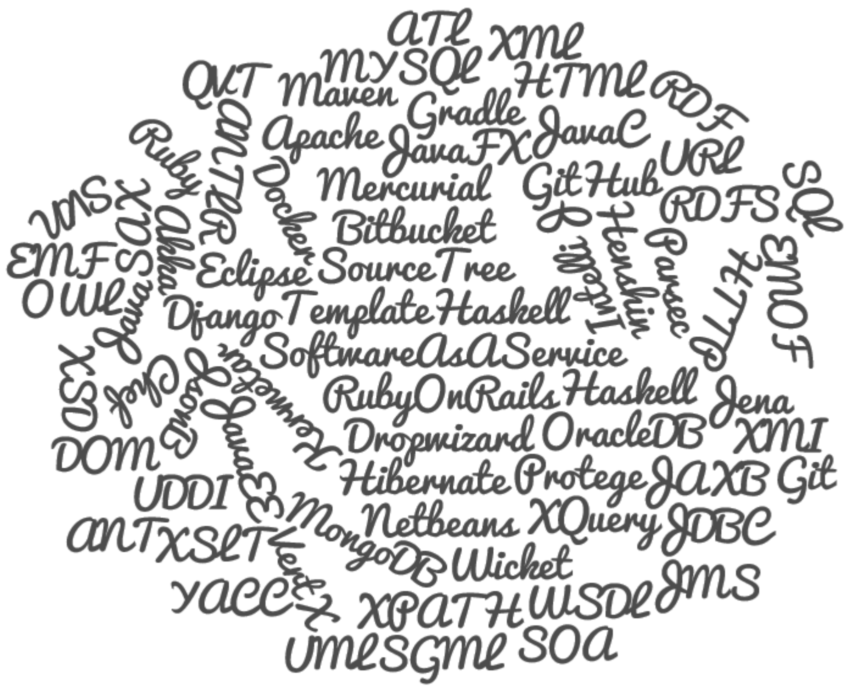
# MegaL-Text
A natural language description

Marcel Heinz
Software Languages Team
University of Koblenz-Landau

SOFTLANG

## We have a problem!

- A rather small set of technologies that come to mind. Who knows what each technology is about?

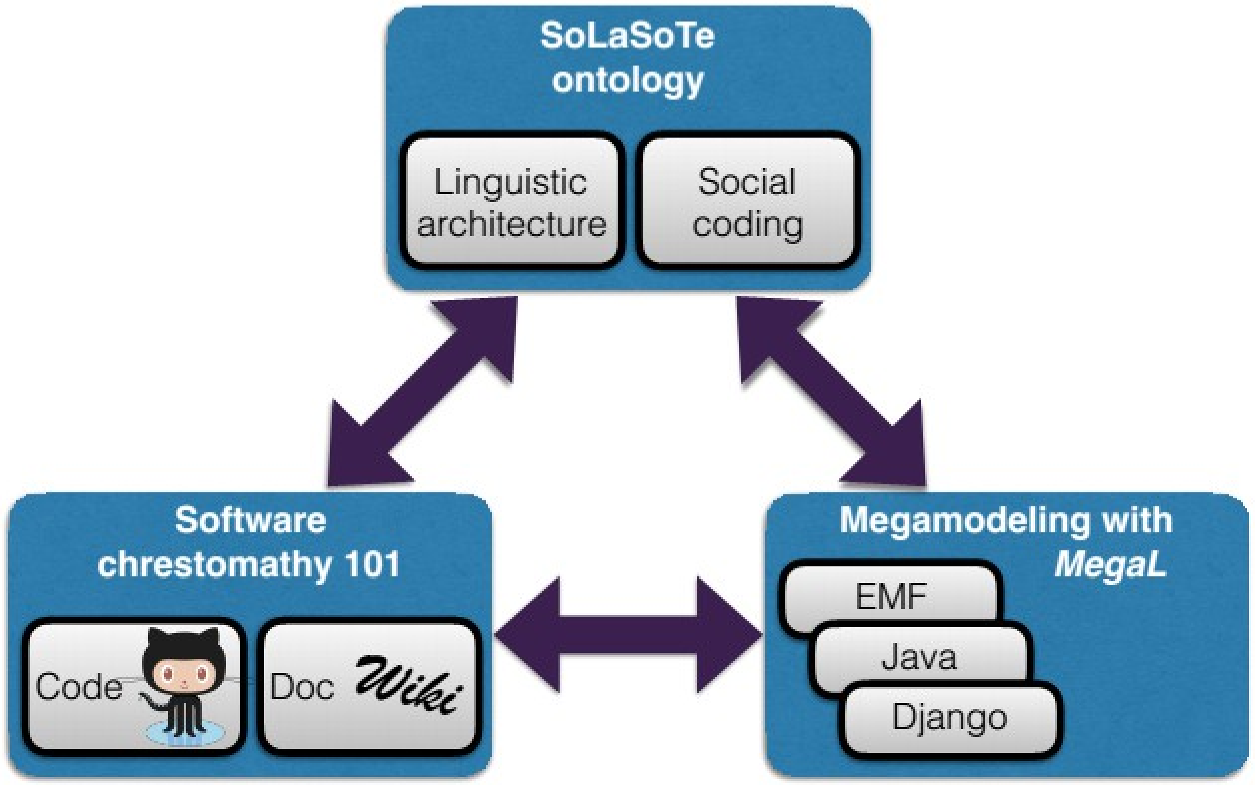


© 2016 Software Languages Team

## Consequences

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

# SoLaSoTe Process



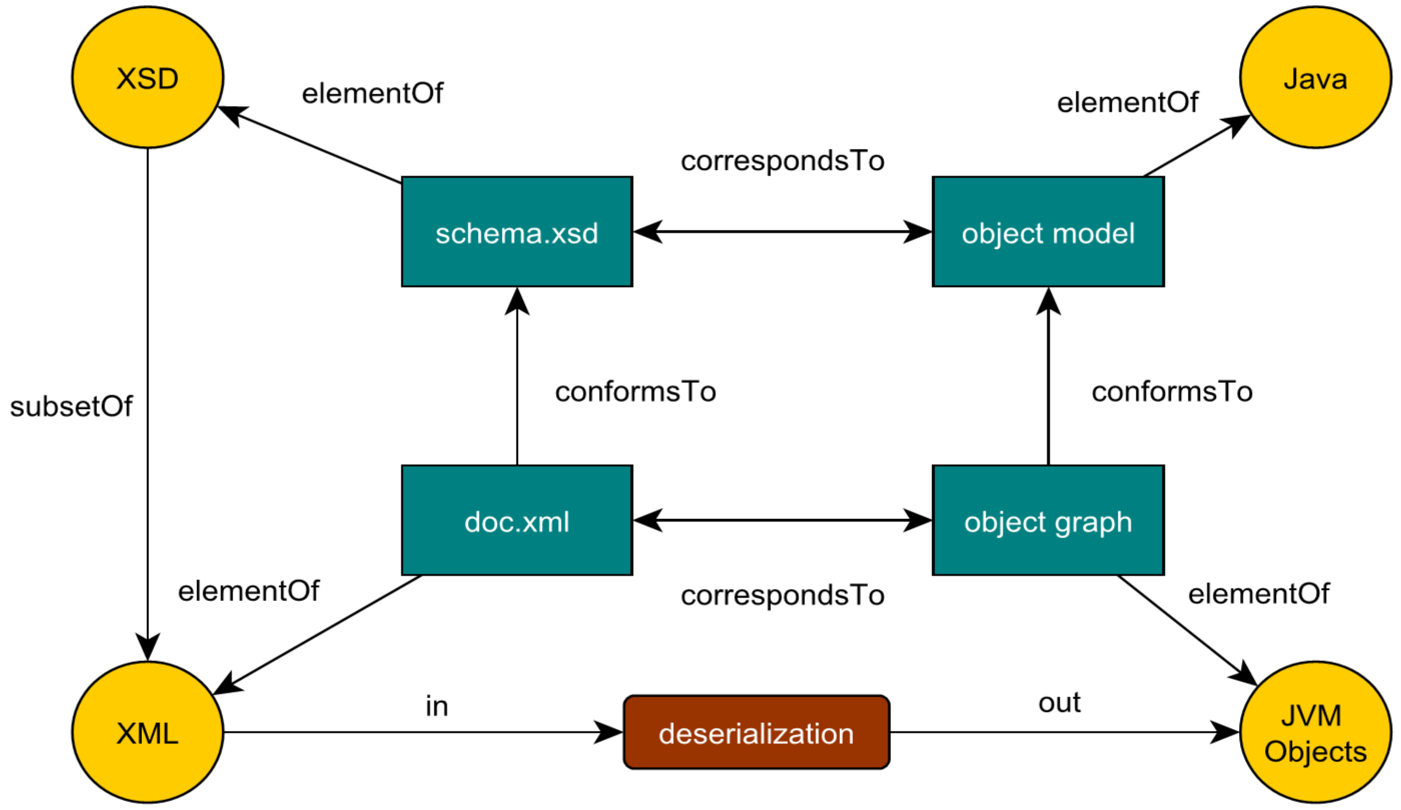
## MegaL

- MegaL is short for 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.

# An Abstract Technology Model



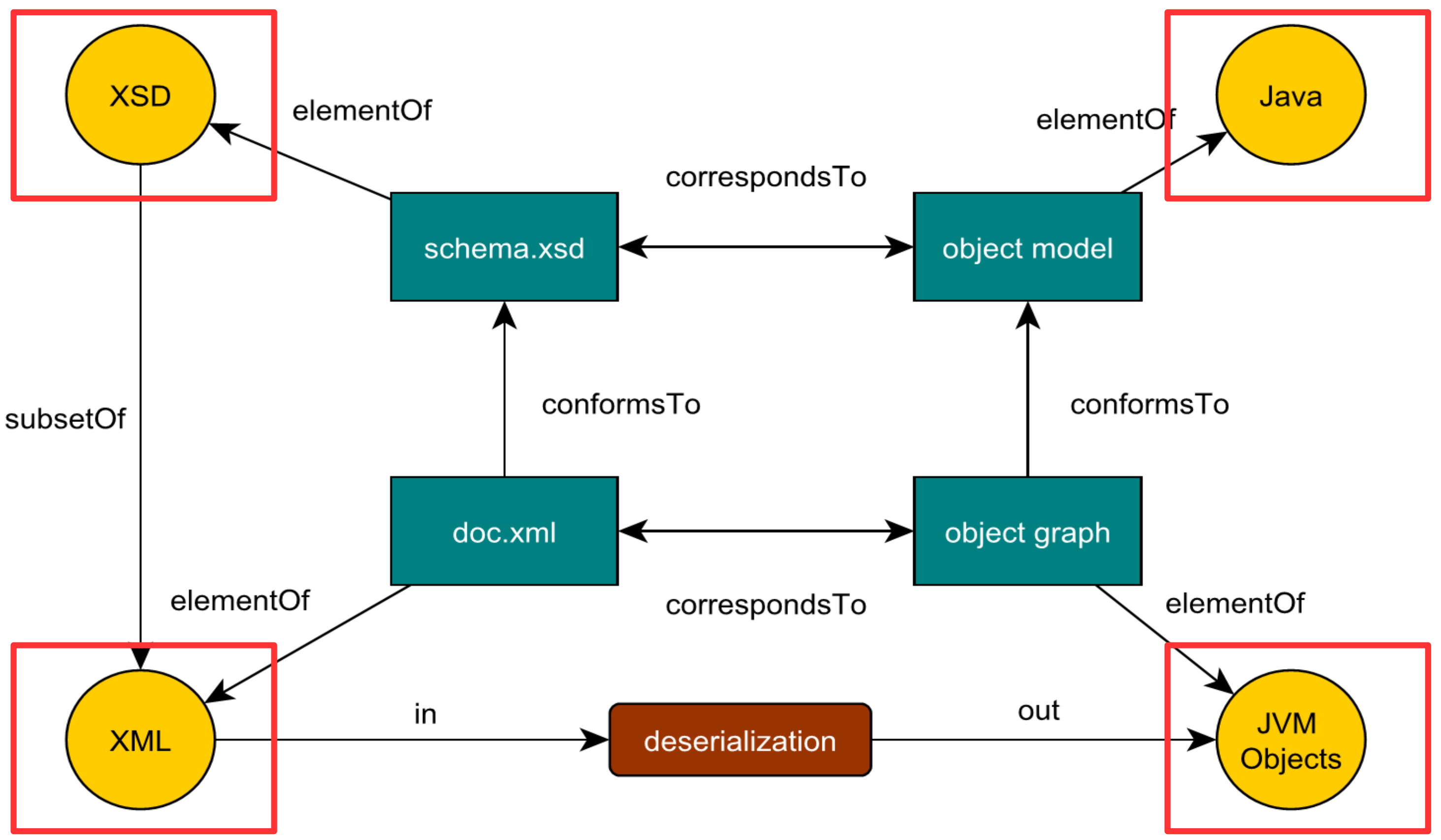
## MegaL/Checker - Prelude

- 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.

## MegaL/Checker - 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
- A language can be a subset of another language.
  - XSD subsetOf XML
  - SQLDDL subsetOf SQL

# An Abstract Technology Model



## MegaL/Checker - Artifact

- An artifact is a digital entity.
  - Artifact < Entity
- An artifact is element of a language.
  - schema.xsd elementOf XSD
  - doc.xml elementOf XML
  - objectmodel elementOf Java
  - objectgraph elementOf JVMObjects

## MegaL/Checker - 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

# MegaL/Checker - 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

## MegaL/Checker - 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

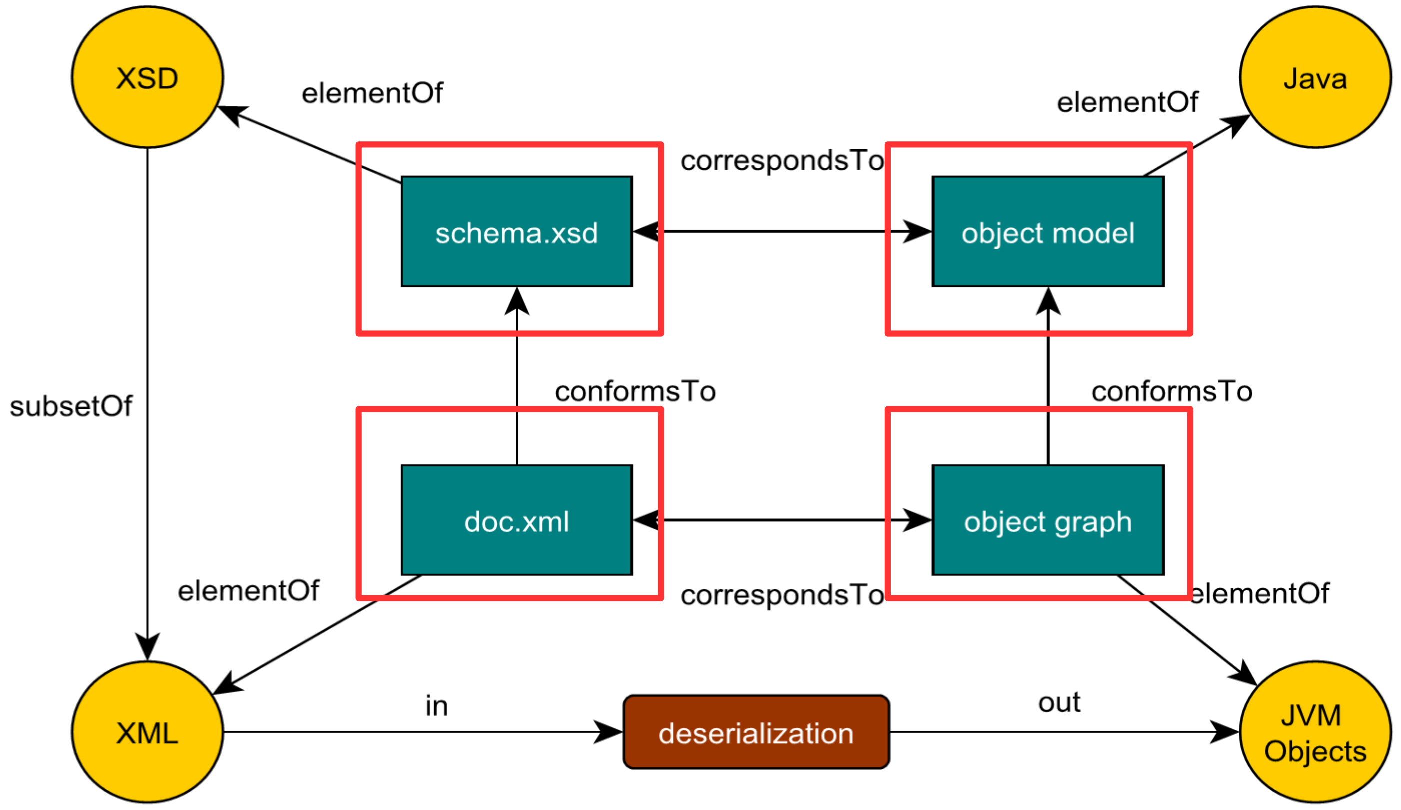
# MegaL/Checker - Role

- 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

# MegaL/Checker - Artifact

- An artifact can correspond to another in the sense that it is semantically equal.
  - correspondsTo < Artifact # Artifact
  - objectgraph correspondsTo doc.xml
  - objectmodel correspondsTo schema.xsd

# An Abstract Technology Model



## MegaL/Checker - Function

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

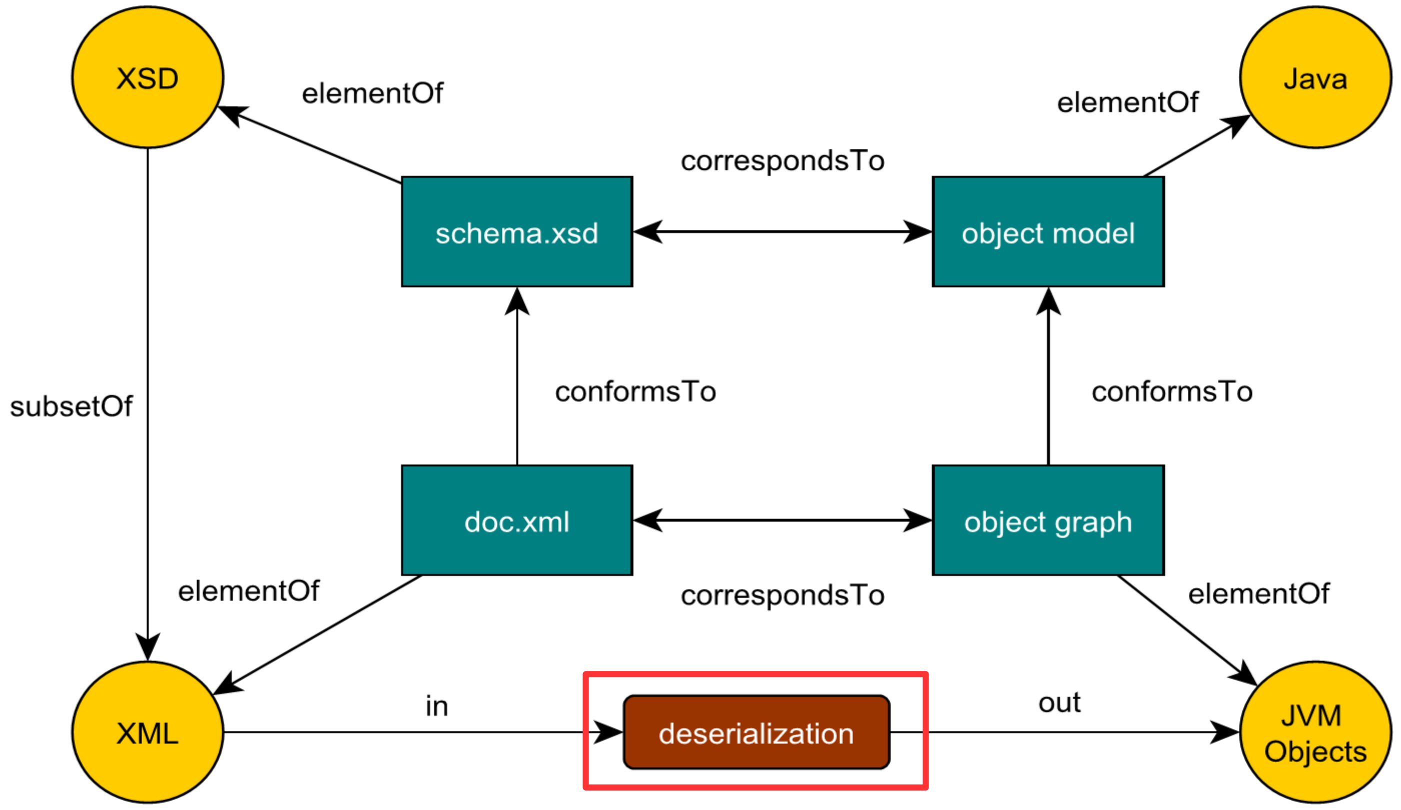
# MegaL/Checker - Function Application

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

## MegaL/Checker - Function

- An artifact may implement a function
  - implements < Artifact # Function
  - ?CutClass implements cut

# An Abstract Technology Model



## MegaL-Text - Technology

- A technology is classified by the kind of usage.
  - FacebookAPI : API
  - Django : Framework
  - JavaSwing : Library
  - Netbeans.RubyPlugin : Plugin
  - EMF.Core : Component
  - MicrosoftOffice2010 : SoftwareSuite
  - Eclipse : IDE
  - GHCI : Platform

## Megal-Text - Technology

- A technology implements functions or languages.
  - implements < Technology # Language
  - JAXB implements XML
  - implements < Technology # Function
  - JAXB implemets serialize

## MegaL-Text - System

- Represents a set of artifacts in an actual technology usage scenario.
  - System < Entity
- Further classification by degree of coordination.
  - Application < System
  - WebApplication < System
  - FileSystem < System

## Let us raise the level of abstraction more!

## MegaL/Checker - Paradigm

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

## Megal/Checker - 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.

## MegaL/Checker - TechnologySpace

- 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

## MegaL/Checker - TechnologySpace

- 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*

## Megal-Text - 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

# MegaL/Checker - Technology

- 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 # ArchitecturalStyle
    - Chef facilitates ClientServer
  - facilitates < Technology # AbstractProcess
    - ANTLR facilitates Parsing

## MegaL/Checker - Domain

- A domain is a conceptual entity.
  - ProgrammingDomain < Entity
  - BusinessProgramming : ProgrammingDomain
  - ProgrammingEducation : ProgrammingDomain
- A technology supports a programming domain.
  - supports < Technology # ProgrammingDomain
  - SAPNetWeaver supports BusinessProgramming

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

## MegaL/Checker - System

- Represents a set of artifacts in an actual technology usage scenario.
  - System < Entity

# MegaL/Checker - Usage

- A system or artifact can <u>use</u> a system, technology, design pattern, architectural style, abstract process or language.

## MegaL/Checker - Language

- 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

## MegaL/Checker - Parthood

- There exist various partOf relations
    - partOf < Artifact # Artifact
    - partOf < Artifact # Technology
    - partOf < Artifact # System
    - partOf < Technology # Technology
    - partOf < System # System

## MegaL/Checker - Syntactic sugar

- Based on RDF Turtle syntax:

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

## MegaL/Checker - 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

## MegaL/Checker - 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"

## Constraints

- Are implemented in the Checker and are stated in natural language here :

  https://github.com/softlang/megalib/blob/master/checker/Constraints.txt