gesis
Leibniz Institute for the Social Sciences

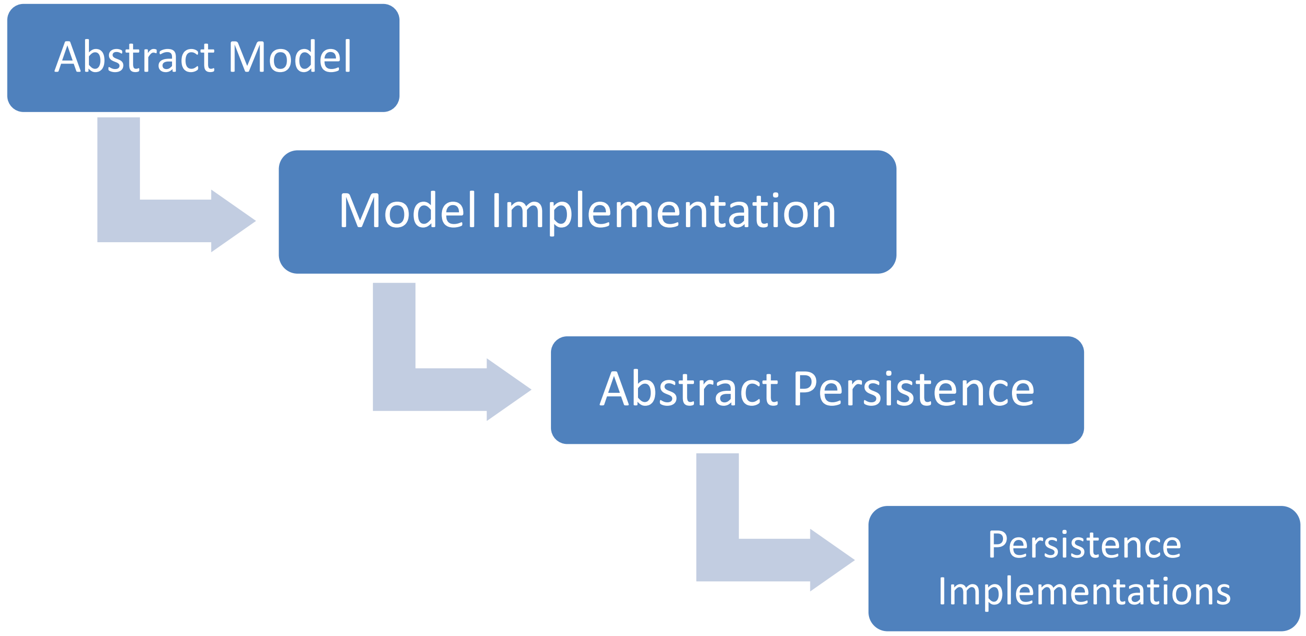
Lessions-learned from Using DDI-RDF Discovery Vocabulary as Backend Model

EDDI14 – 6th Annual European DDI User Conference

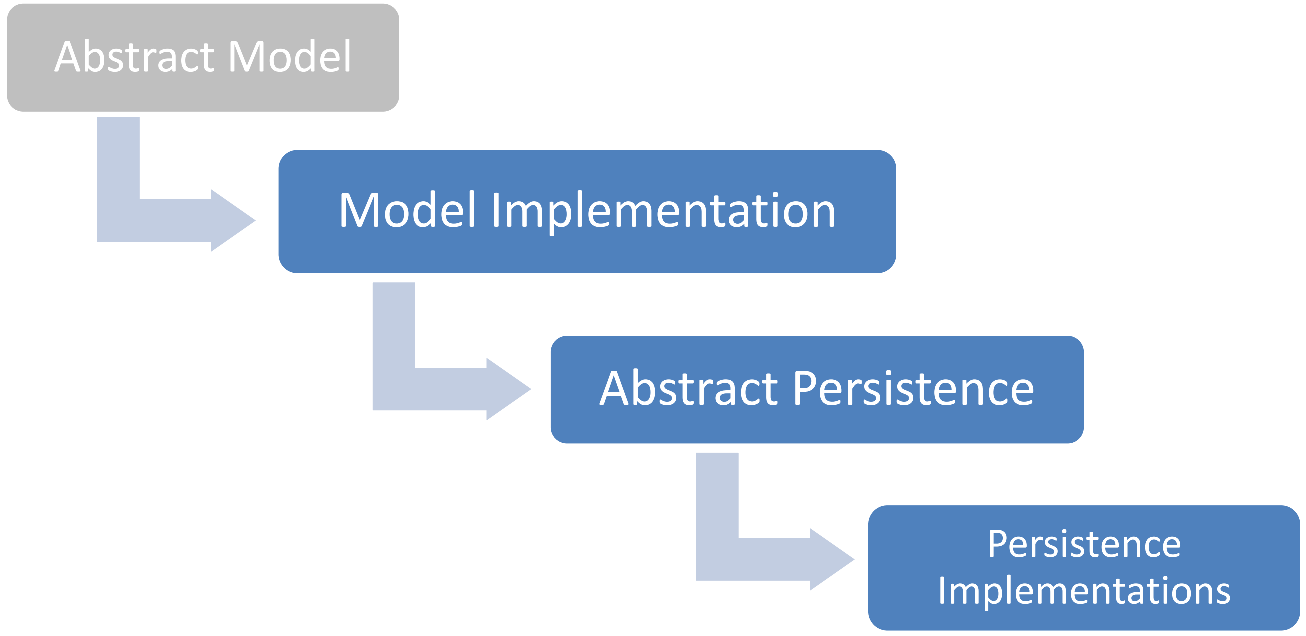
Matthäus Zloch
matthaeus.zloch@gesis.org
PhD Student, M.Sc. CS
GESIS - Leibniz Institute for the Social Sciences

Member of the Leibniz Association

Levels of Lessions-learned



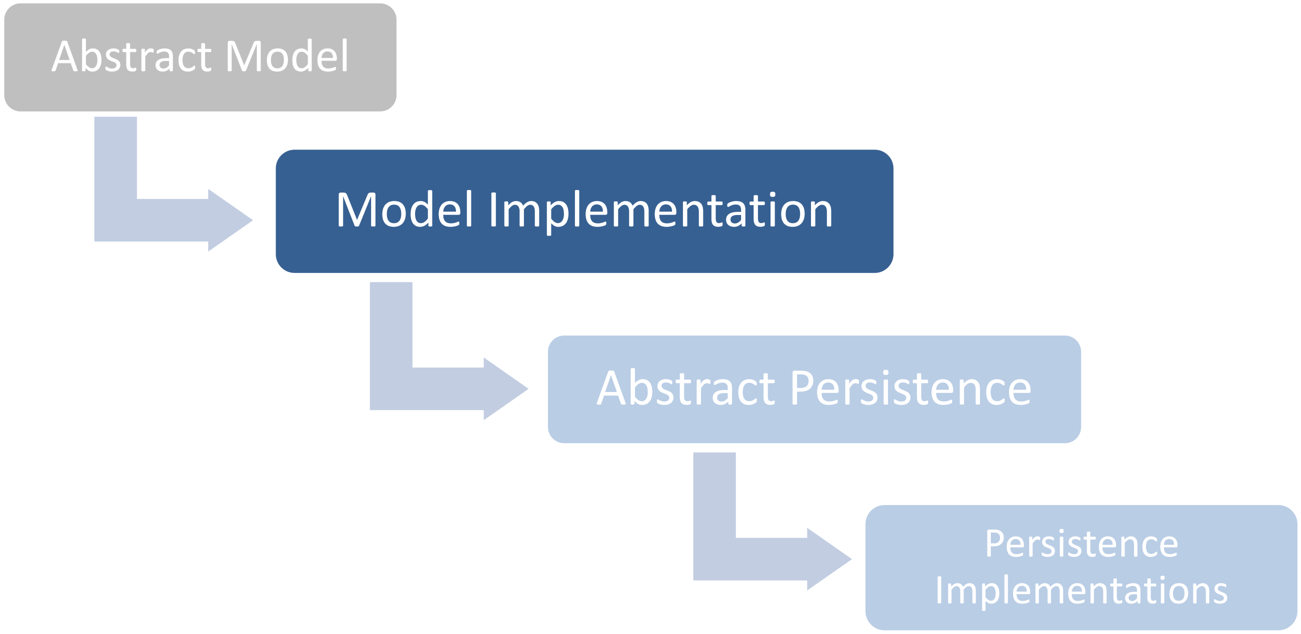
Levels of Lessions-learned



Levels of Lessions-learned

- Levels
- Views on data
- Statistics

Levels of Lessions-learned



About Modeling in General

- Conceptual data model is developed according to a requirements document
- Good practice: use abstract model and extend it to own needs 👍

About Persisting the Model

- The model shouldn’t be restricted to a physical persistence type
- Persistence types exchangeable by configuration
- Data model must not be driven by the views of the application on your data

Leibniz Institute for the Social Sciences

Why we have chosen DDI-RDF

Requirements

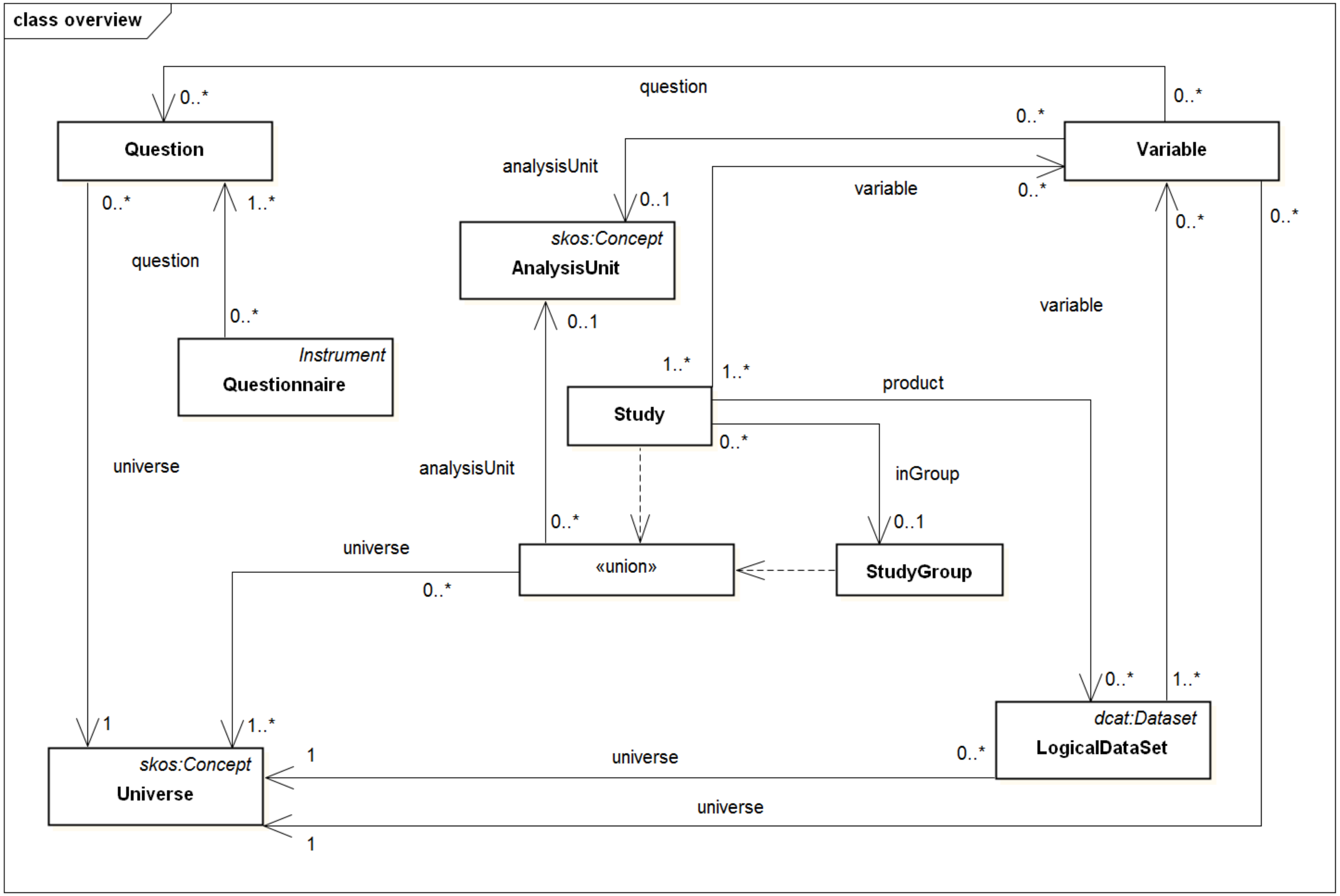
- General enough as “native” application model
- Separate model from application
  - In order to be used in other projects
- Export easy
  - No mapping to the standard schema
  - Iterate through object-structure

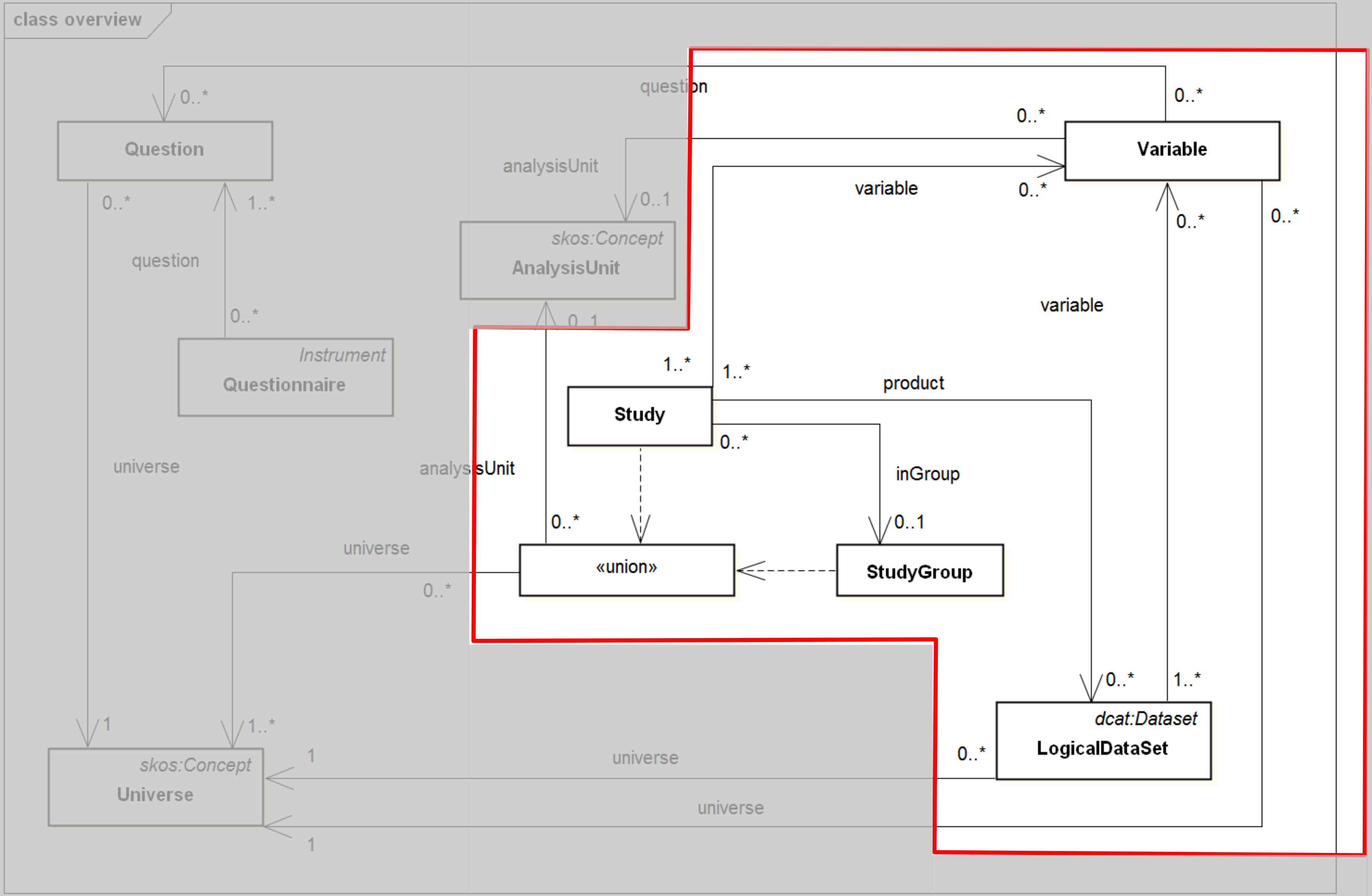
Member of the Leibniz Association

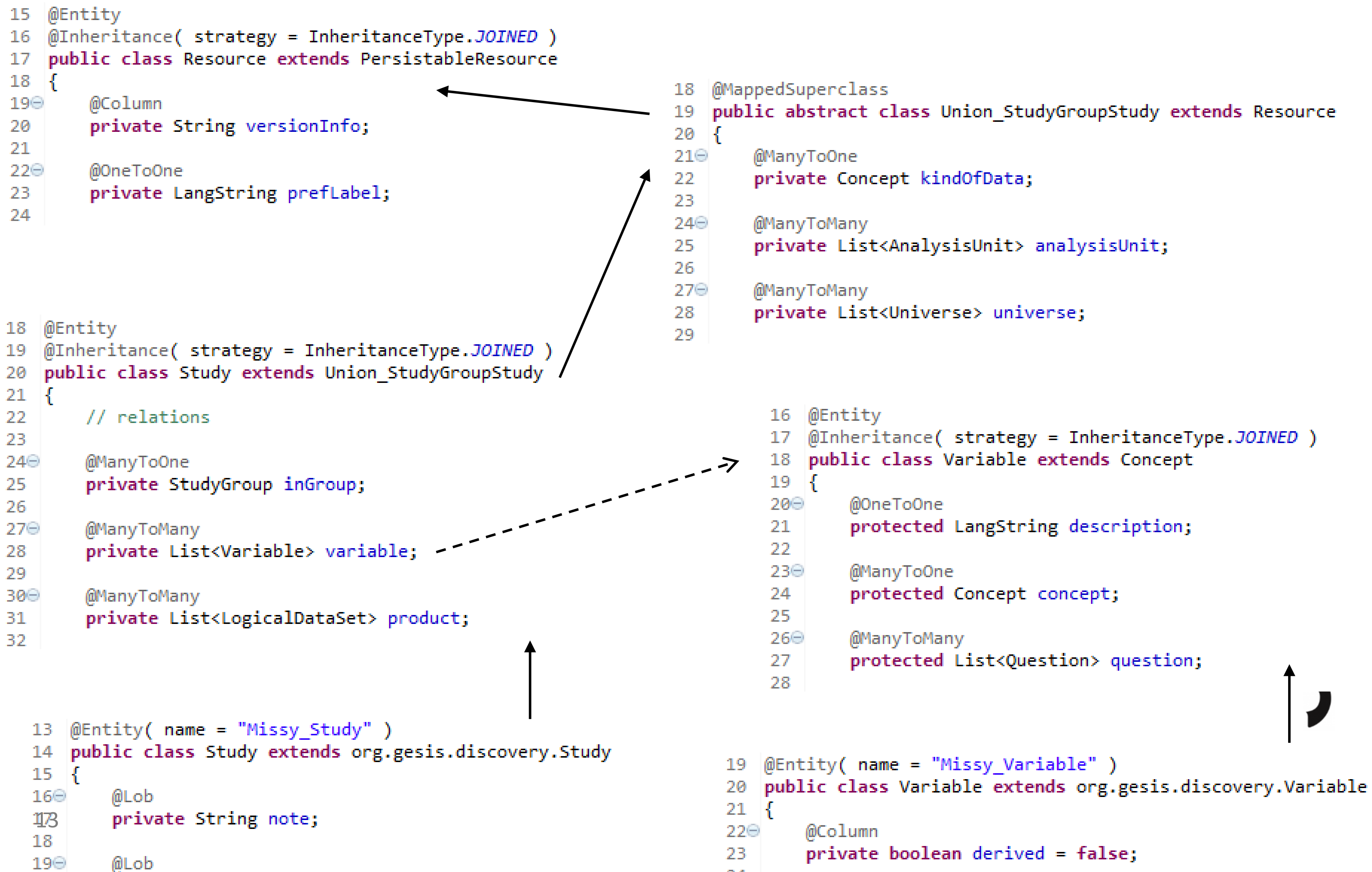
DDI-RDF and JPA

- Implemented abstract model as Java classes
- Annotated with JPA
  - Persistence model for object-relational mappings
  - Can be used with any implementation of JPA
  - Creates entity types on physical layer
  - Should be a matter of configuration
- Focus on API design and code reuse

JPA Specification http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html


class overview
Question
Variable
question
0..*
0..*
0..*
analysisUnit
0..1
variable
0..*
skos:Concept
AnalysisUnit
0..*
1..*
question
0..*
0..*
variable
Instrument
Questionnaire
0..1
1..*
1..*
product
Study
0..*
inGroup
universe
analysisUnit
0..*
0..1
«union»
StudyGroup
universe
0..*
0..*
1..*
1
1..*
dcat:Dataset
LogicalDataSet
skos:Concept
Universe
1
universe
0..*
universe
1


class overview
Question
skos:Concept
AnalysisUnit
Variable
Instrument
Questionnaire
Study
«union»
StudyGroup
dcat:Dataset
LogicalDataSet
skos:Concept
Universe
question
analysisUnit
variable
variable
product
inGroup
universe
universe
universe
universe
question
analysisUnit
0..*
0..1
1..*
1



```java
@Entity
@Inheritance( strategy = InheritanceType.JOINED )
public class Resource extends PersistableResource
{
    @Column
    private String versionInfo;

    @OneToOne
    private LangString prefLabel;

```

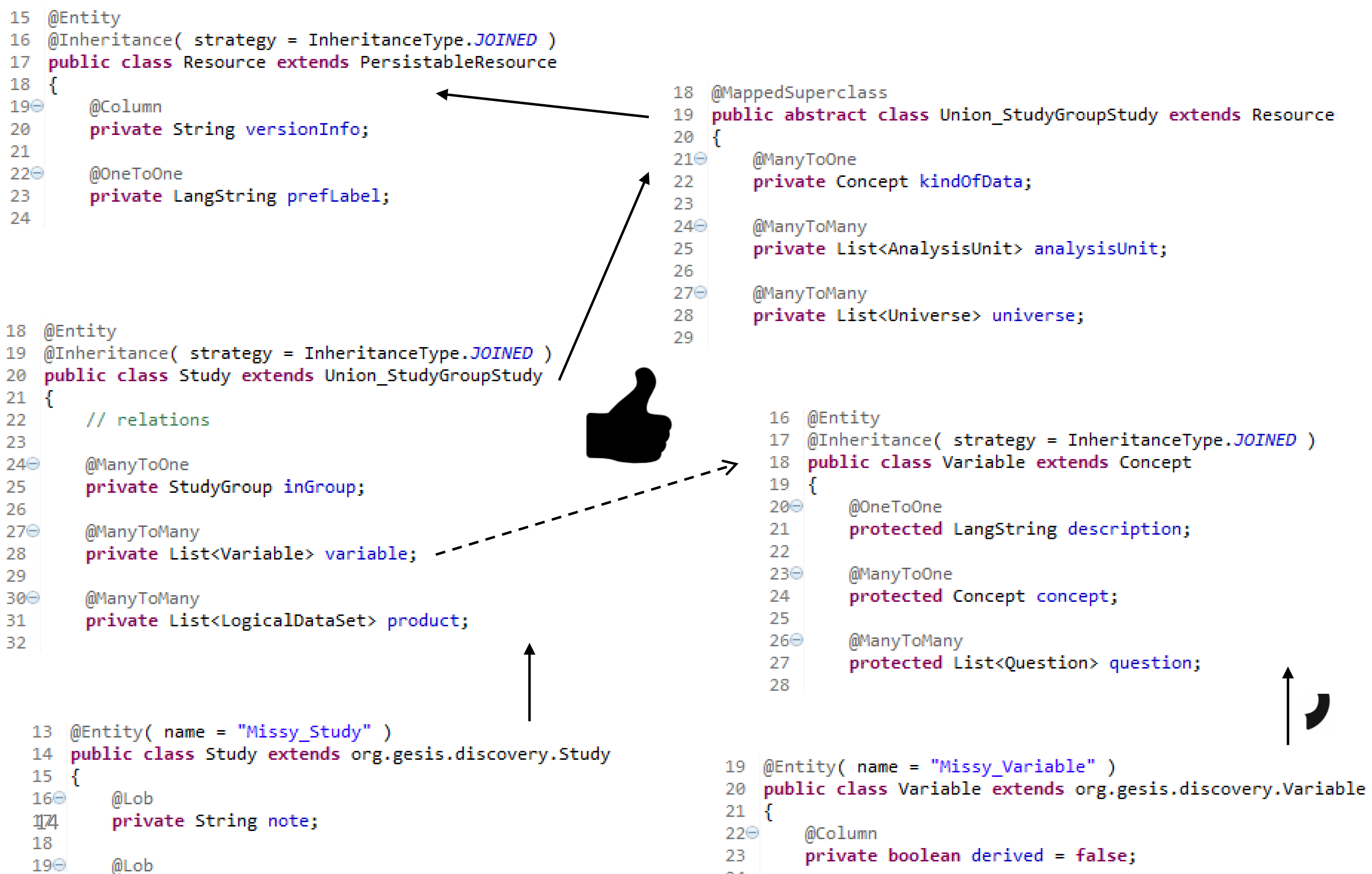
```java
@MappedSuperclass
public abstract class Union_StudyGroupStudy extends Resource
{
    @ManyToOne
    private Concept kindOfData;

    @ManyToMany
    private List<AnalysisUnit> analysisUnit;

    @ManyToMany
    private List<Universe> universe;

```

```java
@Entity
@Inheritance( strategy = InheritanceType.JOINED )
public class Study extends Union_StudyGroupStudy
{
    // relations

    @ManyToOne
    private StudyGroup inGroup;

    @ManyToMany
    private List<Variable> variable;

    @ManyToMany
    private List<LogicalDataSet> product;

```

```java
@Entity
@Inheritance( strategy = InheritanceType.JOINED )
public class Variable extends Concept
{
    @OneToOne
    protected LangString description;

    @ManyToOne
    protected Concept concept;

    @ManyToMany
    protected List<Question> question;

```

```java
@Entity( name = "Missy_Study" )
public class Study extends org.gesis.discovery.Study
{
    @Lob
    private String note;

    @Lob
```

```java
@Entity( name = "Missy_Variable" )
public class Variable extends org.gesis.discovery.Variable
{
    @Column
    private boolean derived = false;
```



```java
@Entity
@Inheritance( strategy = InheritanceType.JOINED )
public class Resource extends PersistableResource
{
    @Column
    private String versionInfo;

    @OneToOne
    private LangString prefLabel;

```

```java
@MappedSuperclass
public abstract class Union_StudyGroupStudy extends Resource
{
    @ManyToOne
    private Concept kindOfData;

    @ManyToMany
    private List<AnalysisUnit> analysisUnit;

    @ManyToMany
    private List<Universe> universe;

```

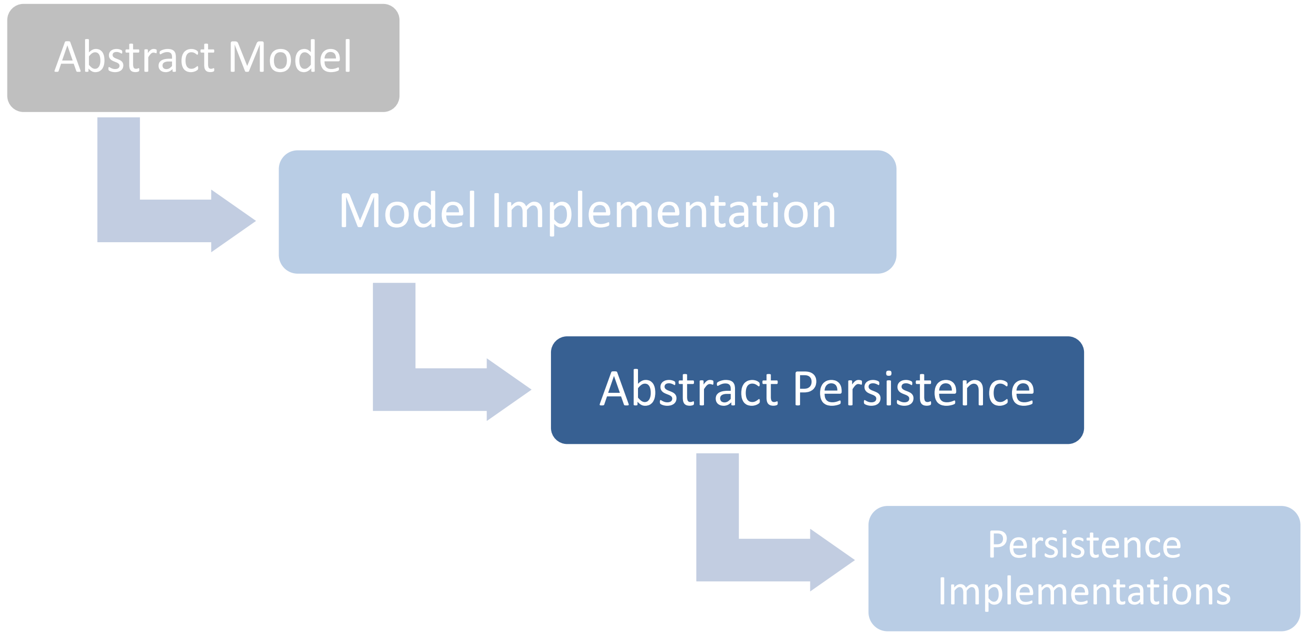
```java
@Entity
@Inheritance( strategy = InheritanceType.JOINED )
public class Study extends Union_StudyGroupStudy
{
    // relations

    @ManyToOne
    private StudyGroup inGroup;

    @ManyToMany
    private List<Variable> variable;

    @ManyToMany
    private List<LogicalDataSet> product;

```

```java
@Entity
@Inheritance( strategy = InheritanceType.JOINED )
public class Variable extends Concept
{
    @OneToOne
    protected LangString description;

    @ManyToOne
    protected Concept concept;

    @ManyToMany
    protected List<Question> question;

```

```java
@Entity( name = "Missy_Study" )
public class Study extends org.gesis.discovery.Study
{
    @Lob
    private String note;

    @Lob
```

```java
@Entity( name = "Missy_Variable" )
public class Variable extends org.gesis.discovery.Variable
{
    @Column
    private boolean derived = false;
```

Levels of Lessions-learned



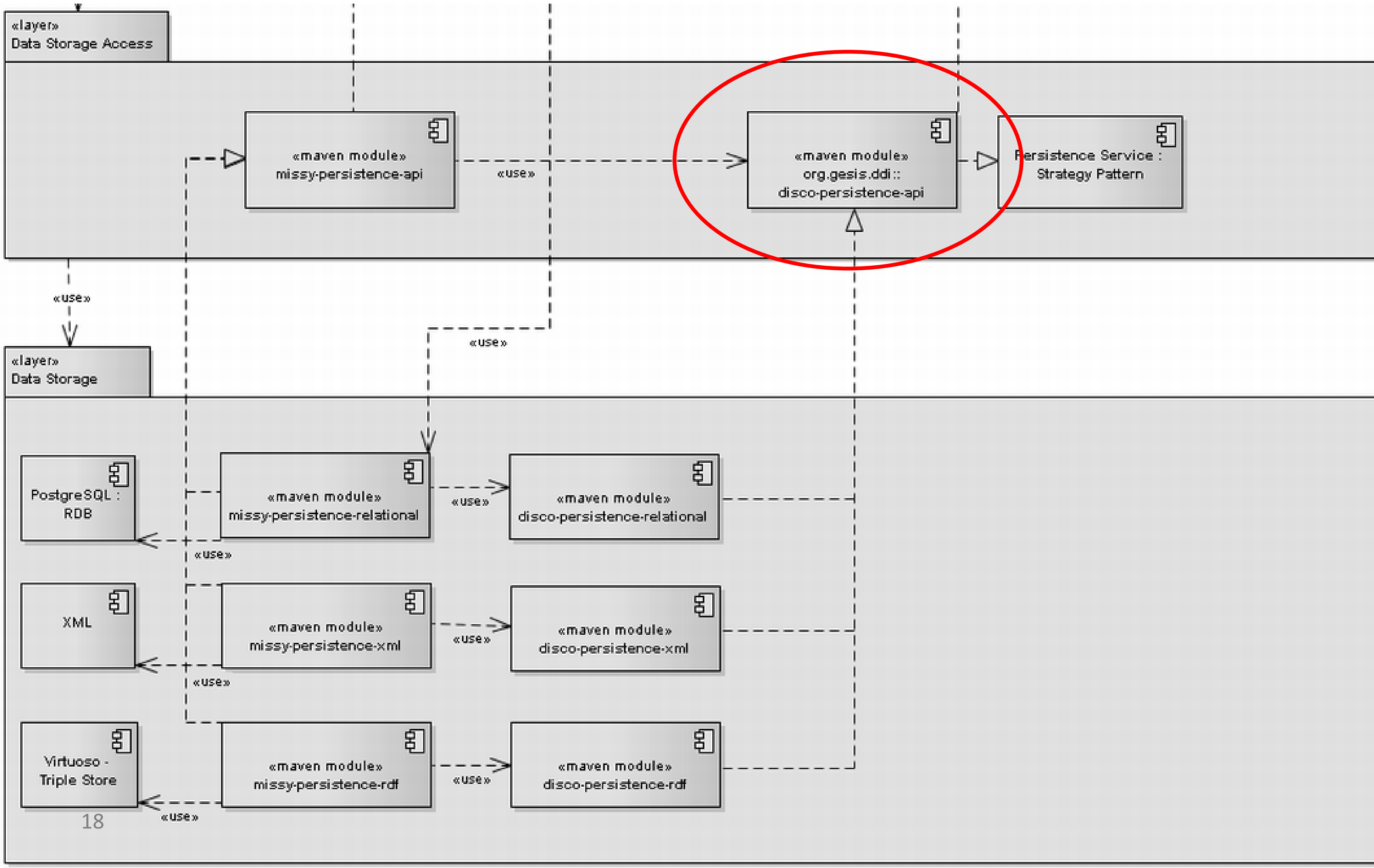
Leibniz Institute for the Social Sciences

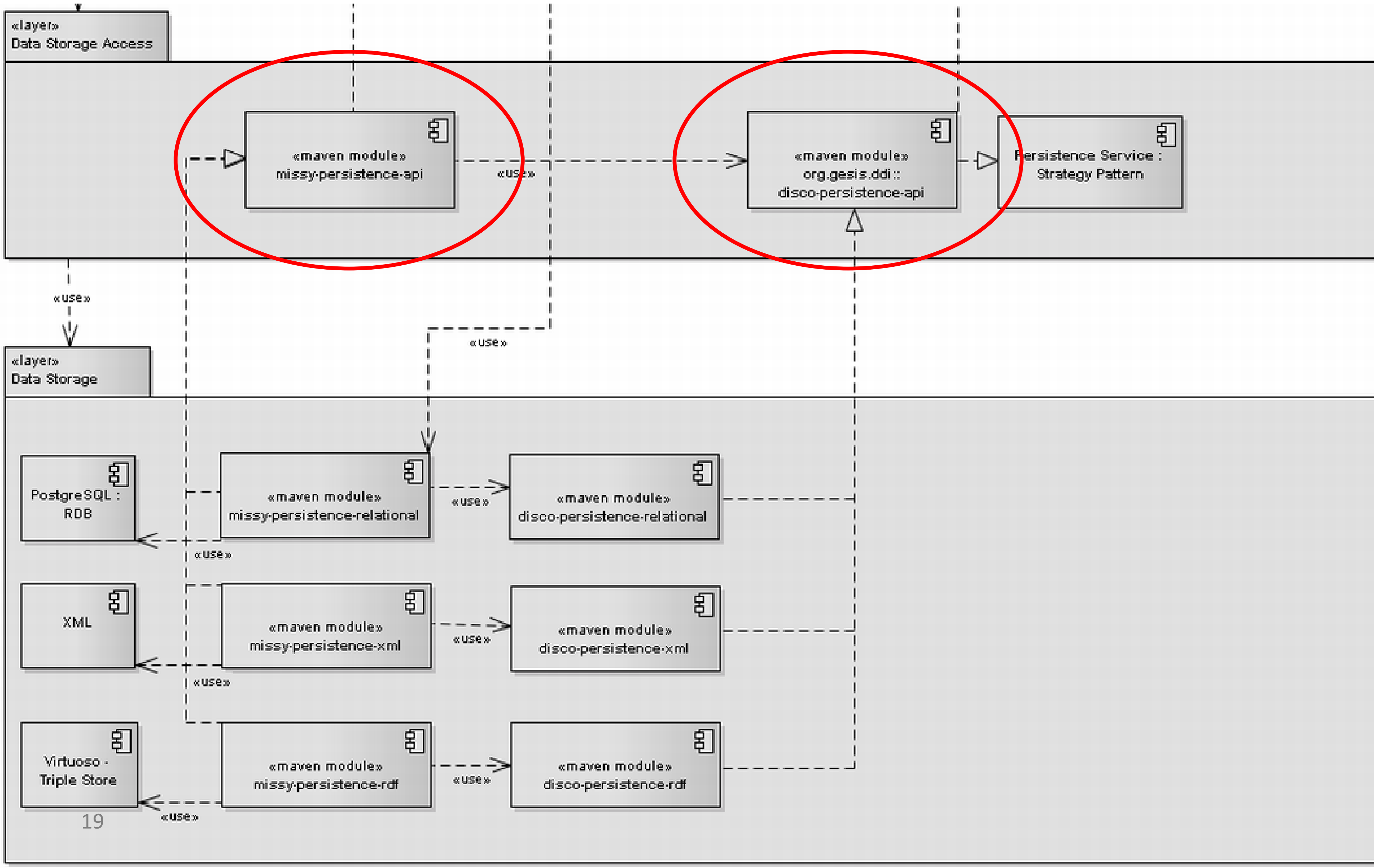
Persistence Level

- Architecture you want to access your data through

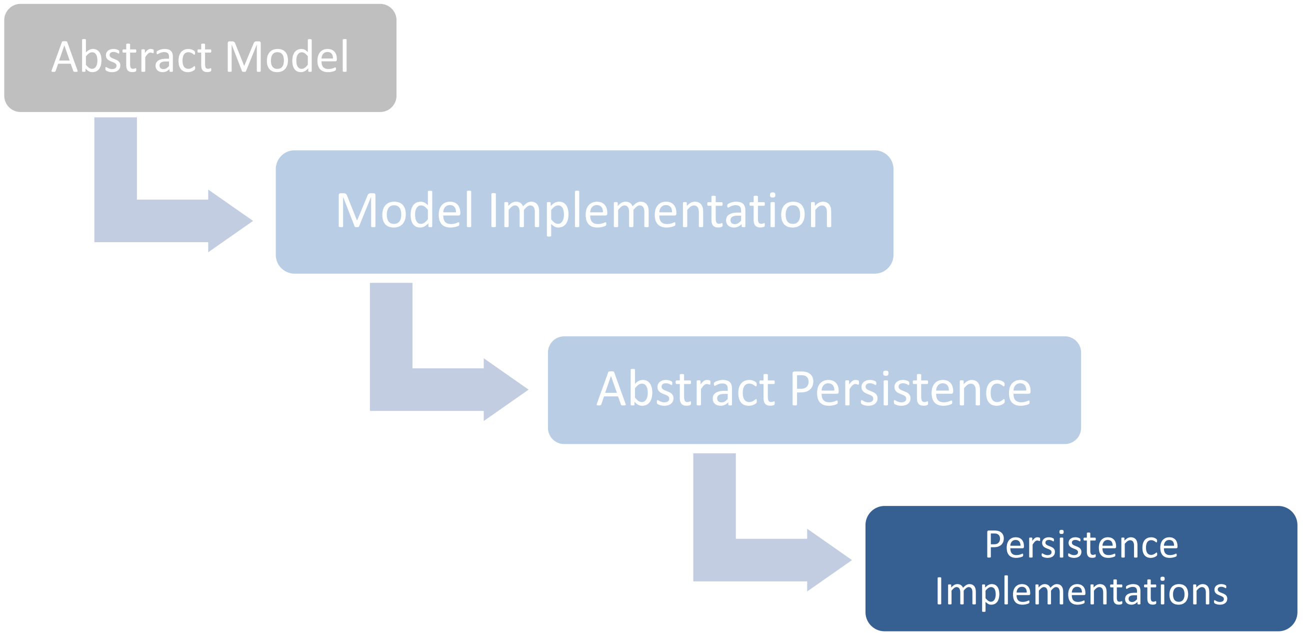
- Good practice to abstract *method access* away from *how you access data*
- Business Level does not need to know *how* the data is stored

Member of the Leibniz Association


«layer»
Data Storage Access
«maven module»
missy-persistence-api
«use»
«maven module»
org.gesis.ddi ::
disco-persistence-api
Persistence Service :
Strategy Pattern
«use»
«use»
«layer»
Data Storage
PostgreSQL :
RDB
«maven module»
missy-persistence-relational
«use»
«maven module»
disco-persistence-relational
«use»
XML
«maven module»
missy-persistence-xml
«use»
«maven module»
disco-persistence-xml
«use»
Virtuoso -
Triple Store
«maven module»
missy-persistence-rdf
«use»
«maven module»
disco-persistence-rdf
«use»


«layer»
Data Storage Access
«maven module»
missy-persistence-api
«use»
«maven module»
org.gesis.ddi::
disco-persistence-api
Persistence Service :
Strategy Pattern
«use»
«use»
«layer»
Data Storage
PostgreSQL :
RDB
«maven module»
missy-persistence-relational
«use»
«maven module»
disco-persistence-relational
«use»
XML
«maven module»
missy-persistence-xml
«use»
«maven module»
disco-persistence-xml
«use»
Virtuoso -
Triple Store
«maven module»
missy-persistence-rdf
«use»
«maven module»
disco-persistence-rdf
«use»

Levels of Lessions-learned



Member of the Leibniz Association

DDI-RDF and JPA

- JPA annotations
  - Can be used with any implementation of JPA to materialize the model

HIBERNATE

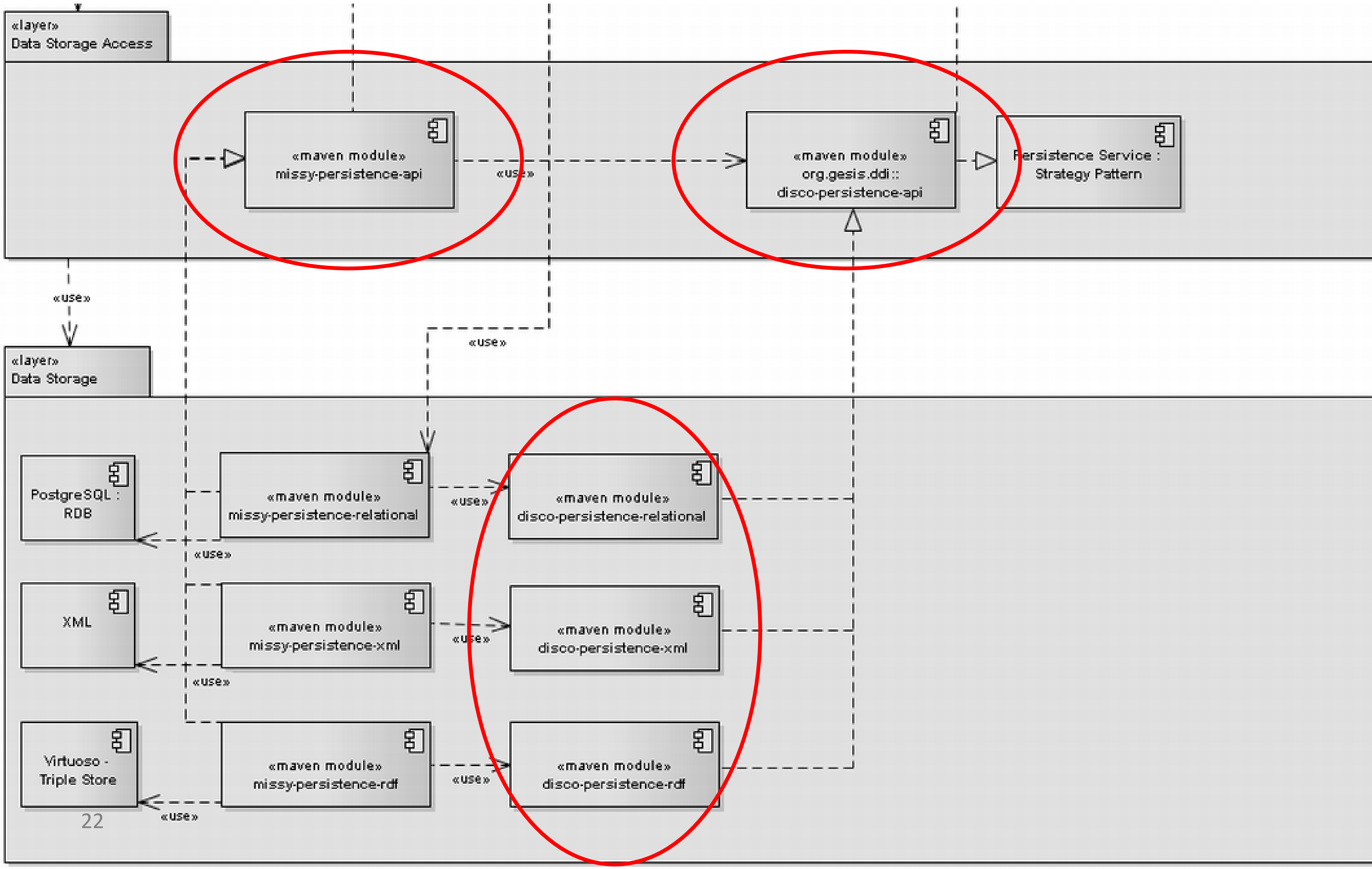
mhgrove / Empire

Spring Data Neo4J

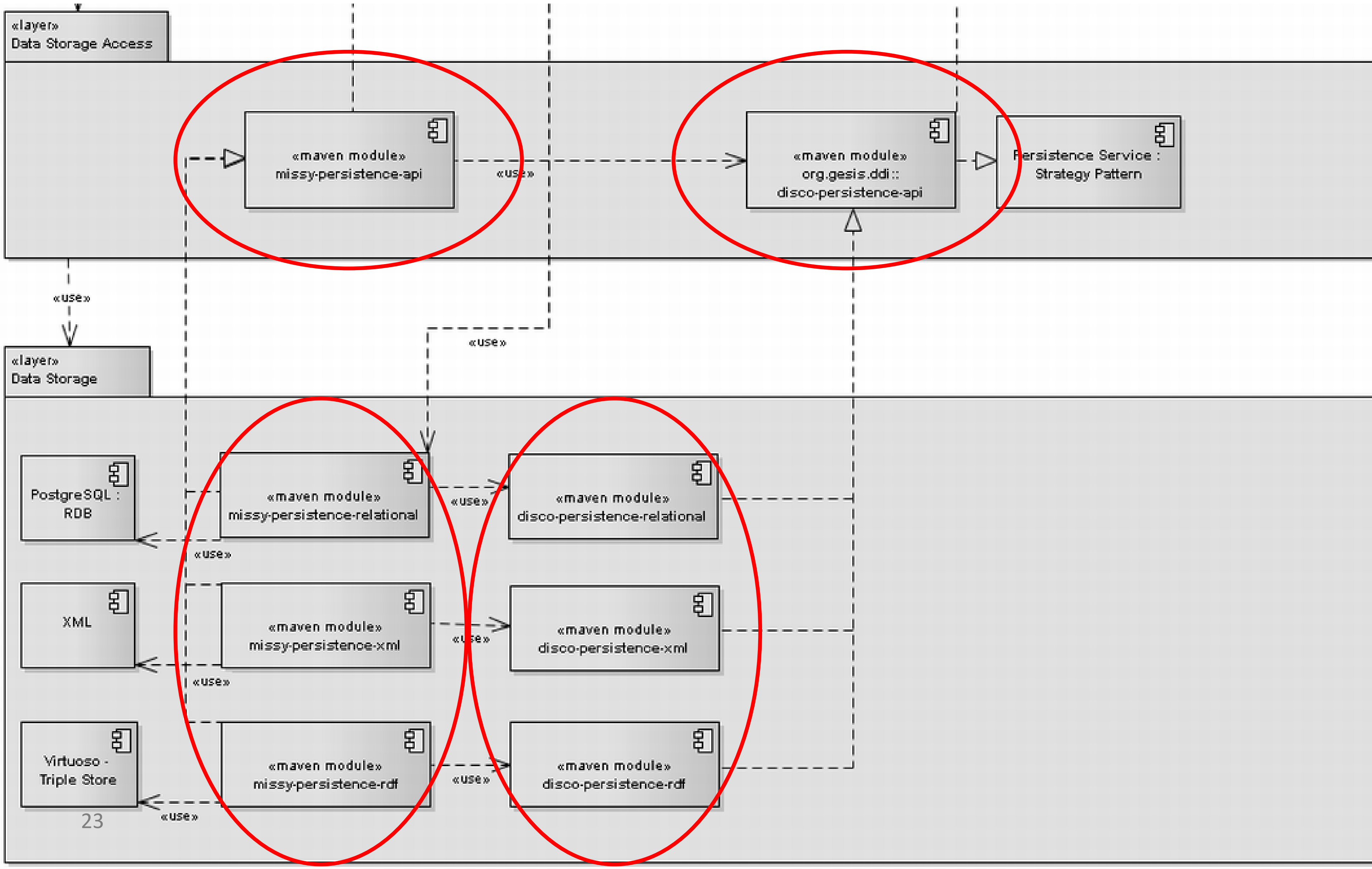
MySQL …

Sesame …

Neo4j …


«layer»
Data Storage Access
«maven module»
missy-persistence-api
«use»
«maven module»
org.gesis.ddi::
disco-persistence-api
Persistence Service :
Strategy Pattern
«use»
«use»
«layer»
Data Storage
PostgreSQL :
RDB
«maven module»
missy-persistence-relational
«use»
«maven module»
disco-persistence-relational
«use»
XML
«maven module»
missy-persistence-xml
«use»
«maven module»
disco-persistence-xml
«use»
Virtuoso -
Triple Store
«maven module»
missy-persistence-rdf
«use»
«maven module»
disco-persistence-rdf
«use»

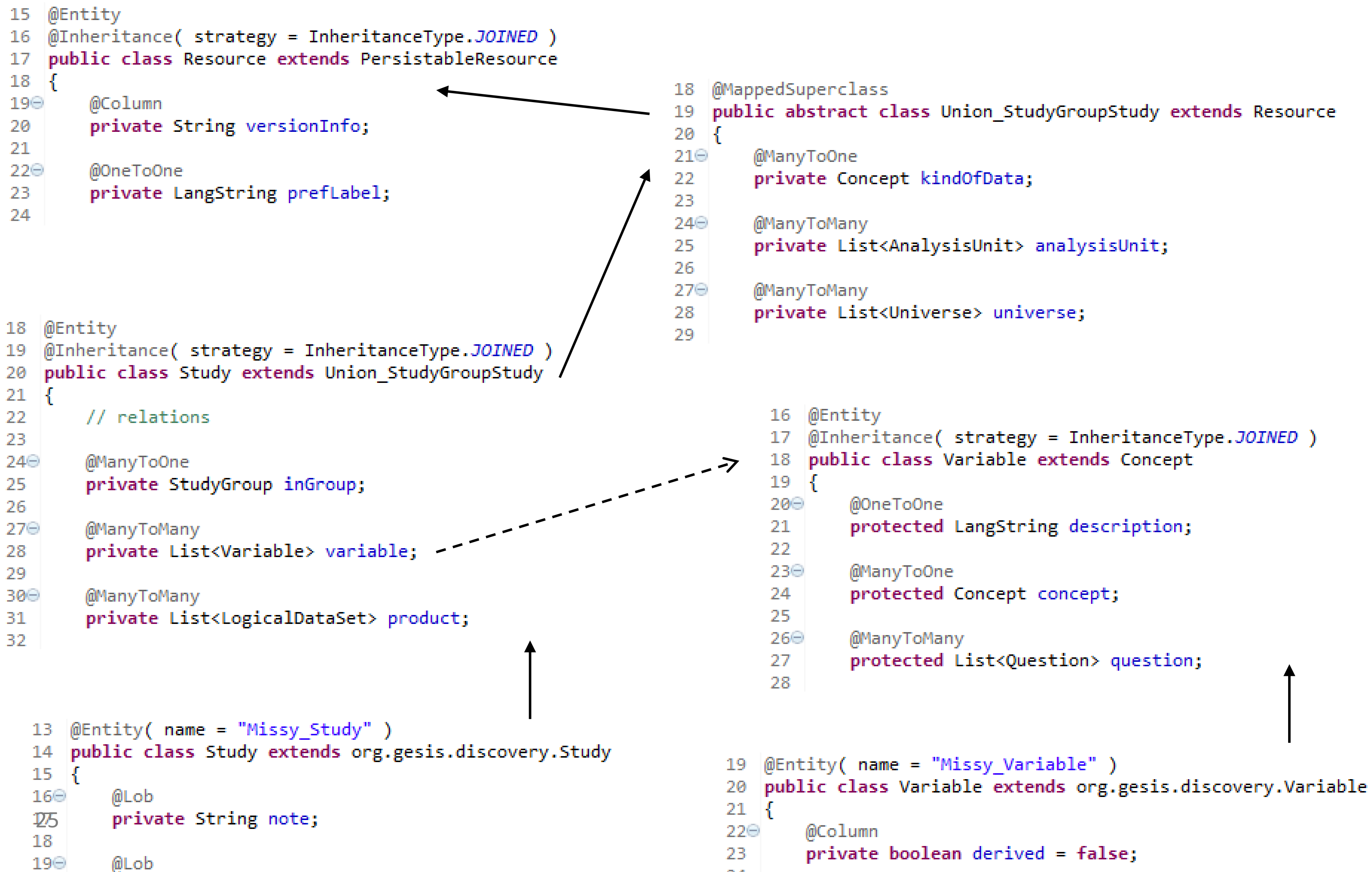
gesis
Leibniz Institute for the Social Sciences



Storage Types

- Implementation of model classes is highly hierarchical

- How does the storage type save the data?



```
@Entity
@Inheritance( strategy = InheritanceType.JOINED )
public class Resource extends PersistableResource
{
    @Column
    private String versionInfo;

    @OneToOne
    private LangString prefLabel;

```

```
@MappedSuperclass
public abstract class Union_StudyGroupStudy extends Resource
{
    @ManyToOne
    private Concept kindOfData;

    @ManyToMany
    private List<AnalysisUnit> analysisUnit;

    @ManyToMany
    private List<Universe> universe;

```

```
@Entity
@Inheritance( strategy = InheritanceType.JOINED )
public class Study extends Union_StudyGroupStudy
{
    // relations

    @ManyToOne
    private StudyGroup inGroup;

    @ManyToMany
    private List<Variable> variable;

    @ManyToMany
    private List<LogicalDataSet> product;

```

```
@Entity
@Inheritance( strategy = InheritanceType.JOINED )
public class Variable extends Concept
{
    @OneToOne
    protected LangString description;

    @ManyToOne
    protected Concept concept;

    @ManyToMany
    protected List<Question> question;

```

```
@Entity( name = "Missy_Study" )
public class Study extends org.gesis.discovery.Study
{
    @Lob
    private String note;

    @Lob
```

```
@Entity( name = "Missy_Variable" )
public class Variable extends org.gesis.discovery.Variable
{
    @Column
    private boolean derived = false;
```

Storage Types

- Relational Database – Tables
  - Use InheritanceType.JOINED
  - Use InheritanceType.SINGLE_TABLE
- Graph Database – Nodes
- RDFStore – Triples

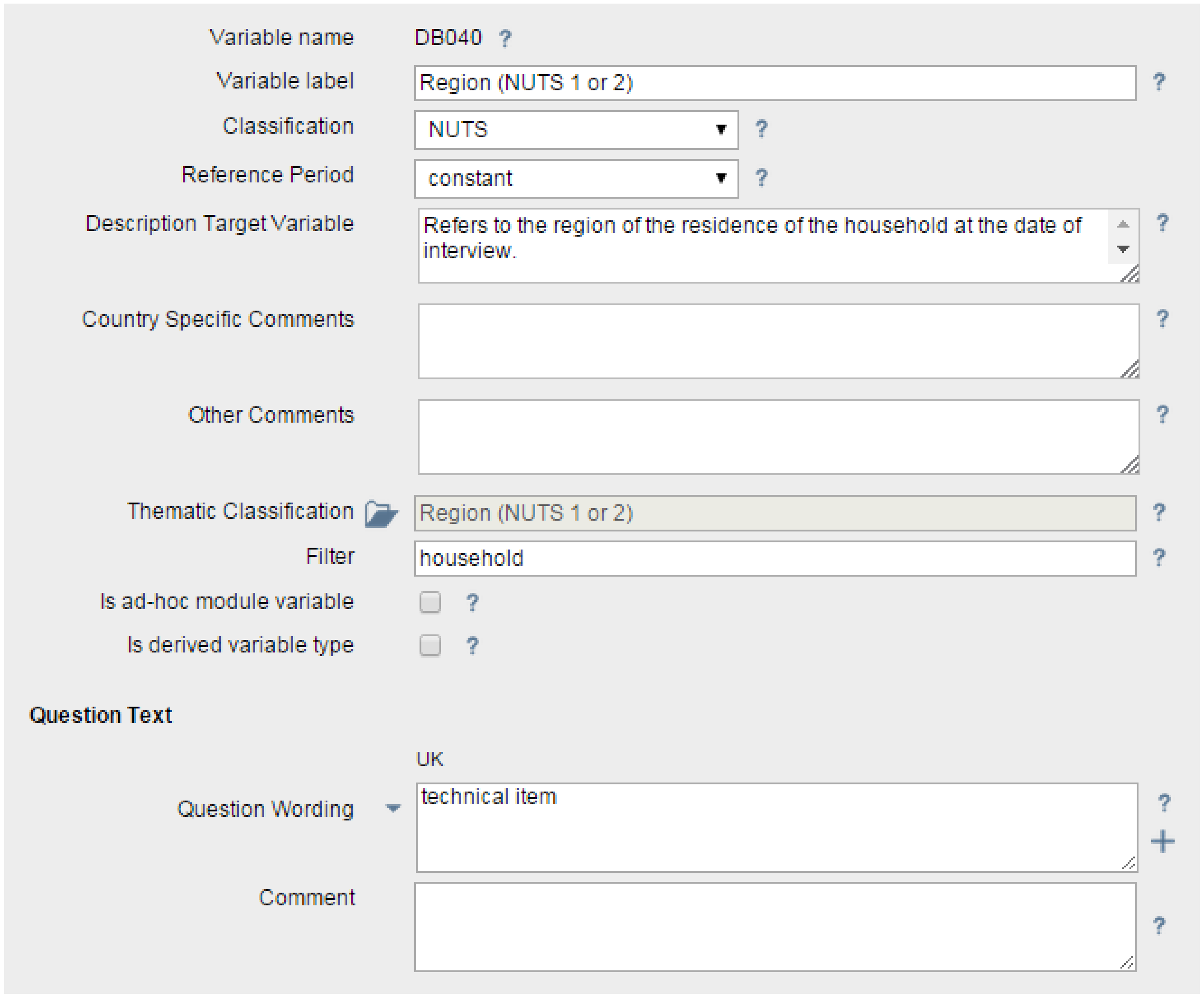
Example: Relational Database

- One table for each @Entity
  - Clean database
  - Many (unnecessary) tables involved in query
  - Updates affect several tables

```
FROM
            missy_variable, variable, concept, resource,
            missy_logicalDataSet, logicalDataSet, resource, logicalDataSet_variable
            missy_study, study, resource, study_logicalDataSet
```

Levels of Lessions-learned

- Levels
- Views on data
- Statistics



Variable name	DB040
Variable label	Region (NUTS 1 or 2)
Classification	NUTS
Reference Period	constant
Description Target Variable	Refers to the region of the residence of the household at the date of interview.
Country Specific Comments	
Other Comments	
Thematic Classification	Region (NUTS 1 or 2)
Filter	household
Is ad-hoc module variable	☐
Is derived variable type	☐

Question Text

	UK
Question Wording	technical item
Comment	

Views on data

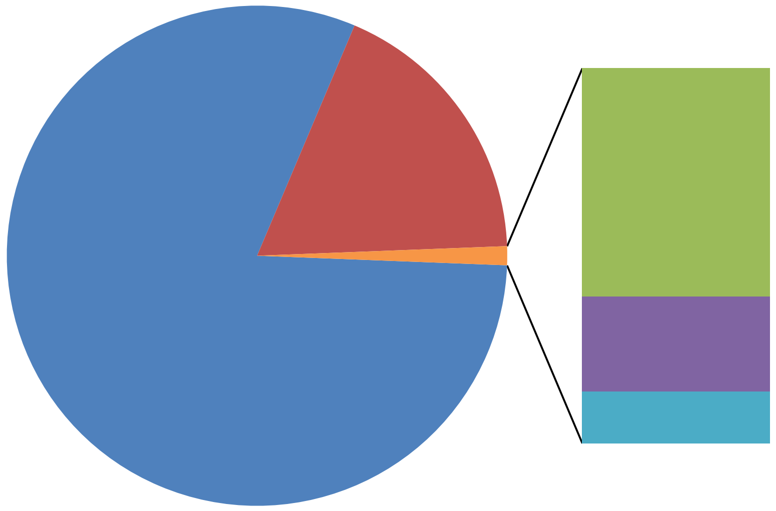
- Specific parts of a model
  - Project specific
  - Use case specific

- Good practice: create views on the physical level with the introduction of new entities

Levels of Lessions-learned

- Levels
- Views on data
- Statistics

Functional Statistics



- StudyGroup 5
- Study 50
- LogicalDataSet 90
- Countries 33

🟦 CategoryStatistics 1,3M
🟥 SummaryStatistics 290 TSD
🟩 Variable 12 TSD
🟪 Document 5 TSD
🟦 Question 2,7 TSD

Technical Statistics

- Intel XEON 2,6Mhz, 2GB RAM, 40GB HDD
- MySQL default installation on Debian 6
- 1GB HDD space usage by MySQL
- 150 Tables

Conclusions DDI-RDF

- DDI-RDF after standardisation, is ready to be implemented
  - As back-end model in different projects
  - With different persistence types
- Open Source frameworks provide many ways to get your data persisted
- It is possible to generate a framework for disco that may be extended

Conclusions Code-Reusage

- Do not create isolated, project specific software
- Create (software) pieces that are reusable
- Reuse other software pieces and/or customize it to your own needs

Contribute and Share

- Go to GitHub
- Download missy-project / disco-model-impl
- Discuss and Contribute

Matthäus Zloch
matthaeus.zloch@gesis.org
PhD Student, M.Sc. CS

GESIS - Leibniz Institute for the Social Sciences

Member of the Leibniz Association