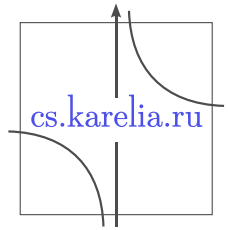
Petrozavodsk State University
Department of Computer Science



Kirill Kulakov, Sergei Marchenkov, Sergey Tishkov

An Approach to Generating OntologyBased Object Model for Smart-M3 platform

The research was financially supported by the Ministry of Education and Science of Russia within project # 2.5124.2017/8.9 of the basic part of state research assignment for 20172019.
The reported study was funded from Russian Fund for Basic Research according to research project # 19-07-01027.
The results were implemented by the Government Program of Flagship University Development for Petrozavodsk State University in 20172021.

24th FRUCT Conference

April 11, 2019, Moscow, Russia

Motivation

- Smart-M3 based service development:
  - Semantic information broker (SIB)
  - Set of knowledge processors (KP)
- SIB provides low-level API (insert, remove, query, subscribe)
- Low-Level support libraries:
  - kpi low, java KPI, C KPI
  - works with “object–predicate–subject” triplets
  - implements SSAP protocol
- High-Level support libraries:
  - users ontology model
  - provides “object-to-triplets” translation

- Java development (desktop, mobile platforms)
  - most implementation in object oriented style
  - interaction with SIB implements by Java KPI or JNI + C KPI

Idea

- Basic idea: convert ontology model to the Java objects
- Expectations:
  - Developer uses objects as presented in the model (objects, methods, properties)
  - Knowledge of SIB work is not required
  - Asynchronous programming
  - Platform-independent API
  - “Easy to use”
- Implementation option: code generator based ontology

Object model

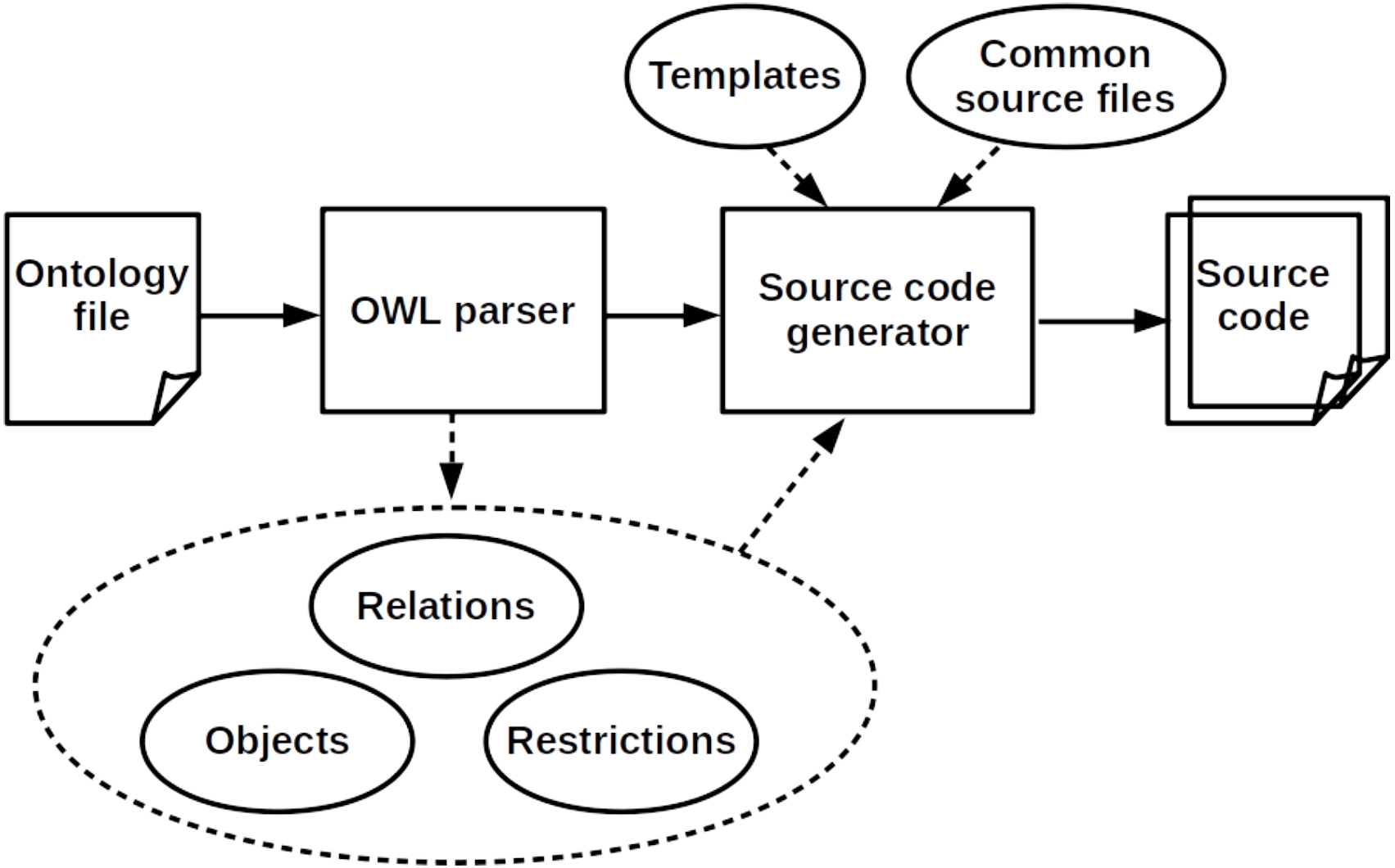
- Ontology classes $\rightarrow$ Object classes
- Instances $\rightarrow$ Objects
- Data type properties or slots $\rightarrow$ Data attribute variables & get/set methods
- Object type properties or slots $\rightarrow$ Object attribute variables & get/set methods
- Value-type/space facets $\rightarrow$ Attribute variables types & if-then-else statements in set methods
- Cardinality facets $\rightarrow$ Additional attributes & if-then-else statements in set methods
- Multiple inheritance $\rightarrow$ Single inheritance & multiple interface inheritance

KP's interaction methods

- Two types of interaction between KP and SIB
  - "Query—Answer" interaction
  - "Subscription—Notification" interaction

- High-level interaction methods
  - Insert object with properties
  - Update object properties
  - Remove object with properties
  - Search one or more objects (triple template or SPARQL query)
  - Object inserting notification
  - Object updating notification
  - Data updating notification

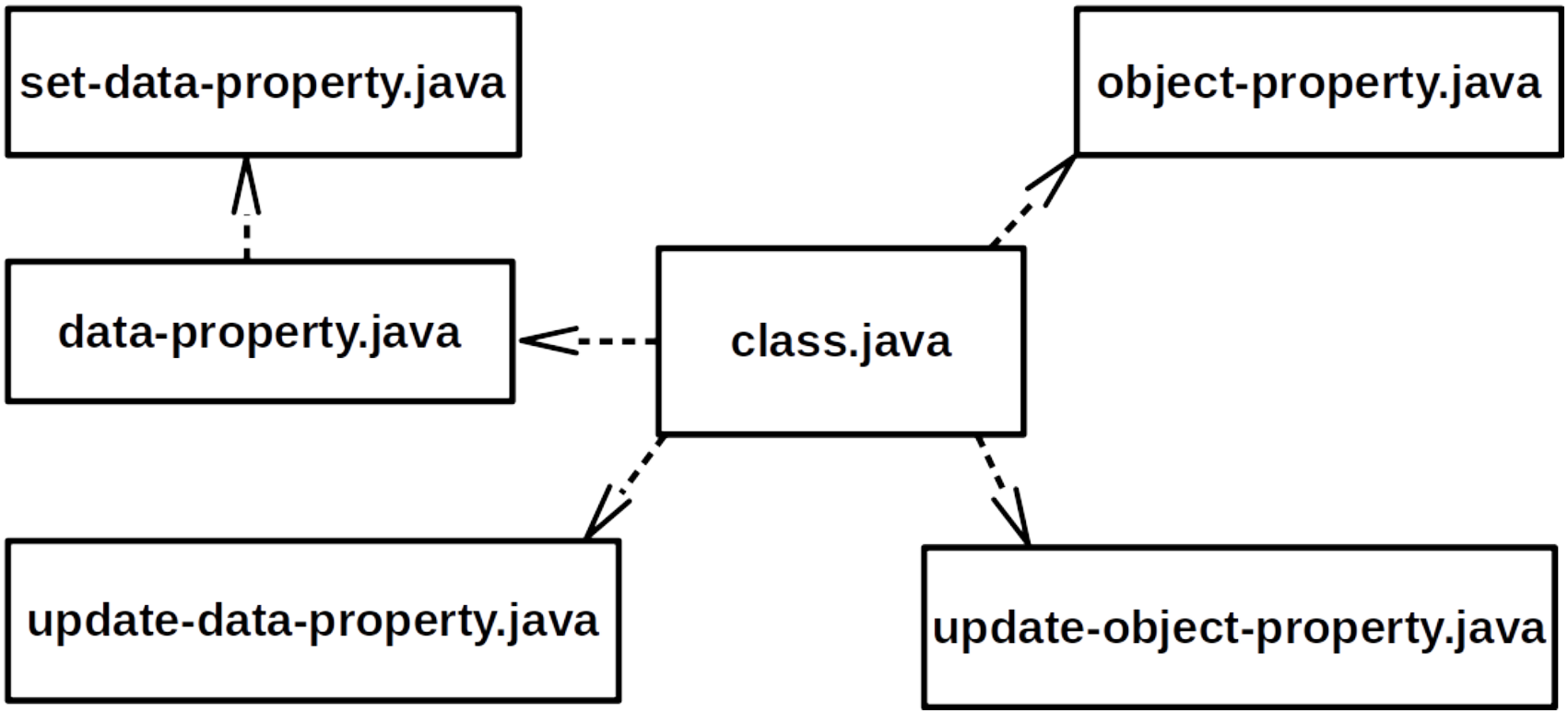
Source code generation process



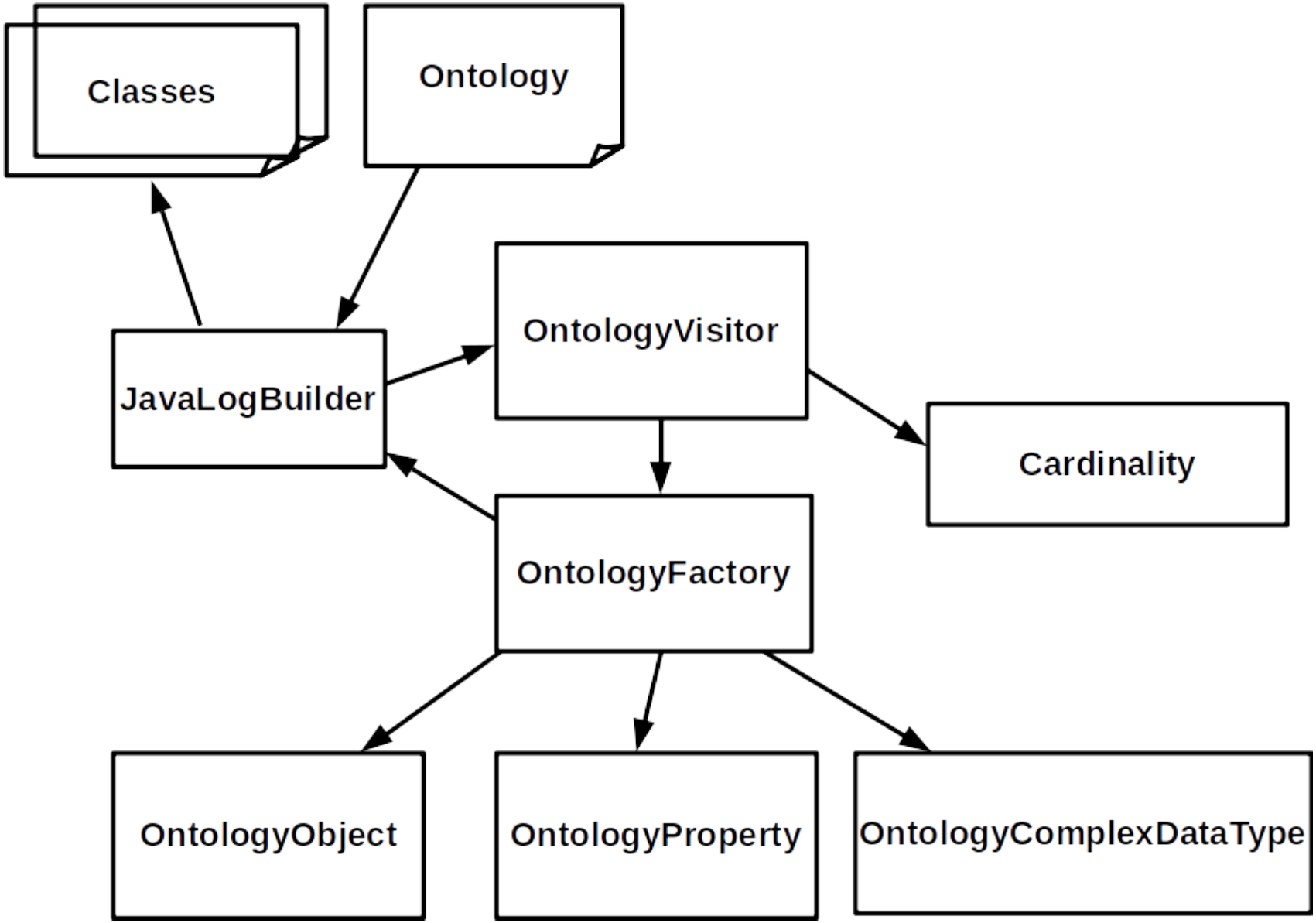
Common source files

- class **BaseRDF** — parent class for all ontology objects;
- class **KPIproxy** — JavaKPI library wrapper;
- interface **QueryListener** — notification interface;
- class **SIBFactory** — main point to work with one or more SIBs, uses “Factory” template;
- class **SIBQueryTask** — parent class for asynchronous access to SIB;
- class **SIBSubscribeTask** — parent class for subscription processes;
- class **SubscribeQuery** — main point of subscriptions, implements wrapper for JavaKPI subscriptions;
- interface **SubscribeListener** — subscription notification interface;
- class **TaskListener** — parent class for all tasks;
- interface **UpdateListener** — object changes notification interface.

Templates

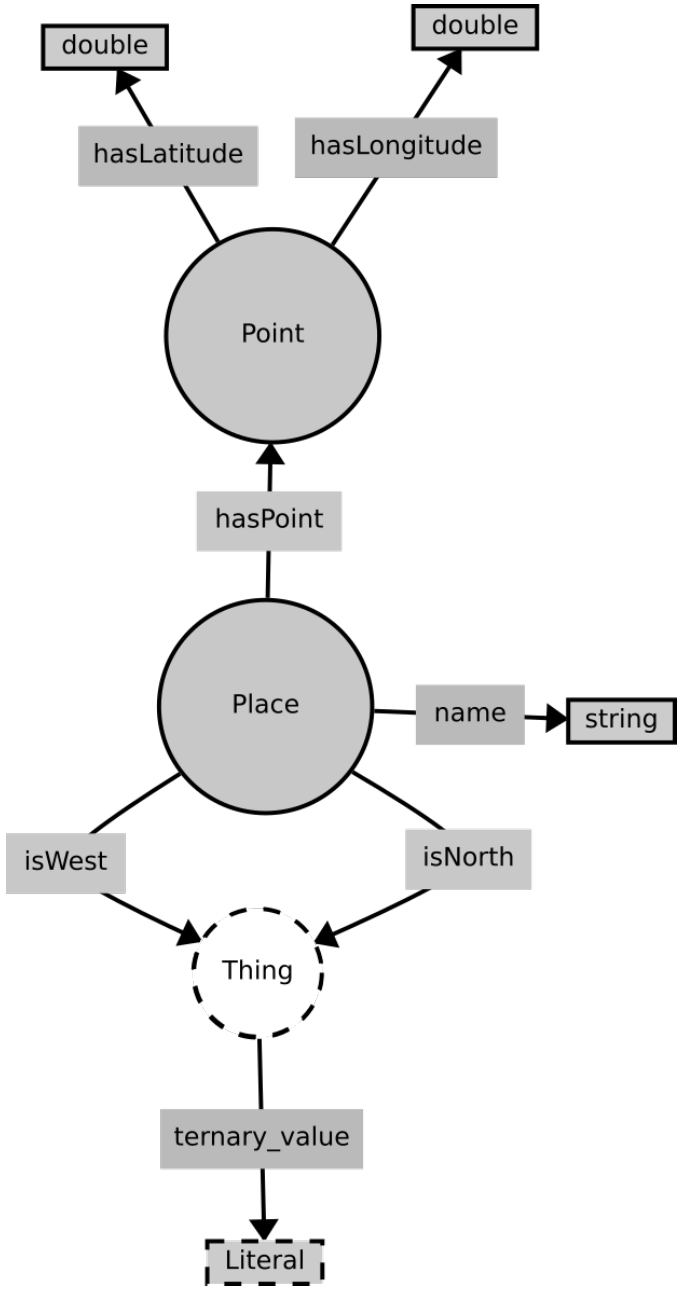

set-data-property.java
object-property.java
data-property.java
class.java
update-data-property.java
update-object-property.java

SmartJavaLog architecture



Example of usage

- Smart service “GeoCode”
  - GeoCode test — geo point generation KP
  - GeoCode teacher — KP of adding geo-dependent information to geo point
  - GeoCode Android — KP for showing result to user
- Two objects: Place and Point
  - GeoCode test generates pair Place—Point with random coordinates
  - GeoCode teacher determines a direction of the world
  - GeoCode Android shows result



Example: Fragment of source code generation

```
public ArrayList<Point> getHasPoint() {
  if (_HasPoint_new != null)
        return _HasPoint_new;

  ArrayList<Point> ret = new ArrayList();
  //search IDs in triples
  ArrayList<String> HasPointIDs = getInTriples(
     HasPoint_URI);
  for (String locID: HasPointIDs) {
    Point value = Point.getInstance(locID,
        _accessPointName);
        ret.add(value);
  }

  return ret;
}
```

Example: Fragment of code to connect to SIB

```
SIBFactory.getInstance().getAccessPoint().setAddr("
    localhost", 10101);

// register used classes before connection
Point.getClassUri();

// connect to SIB
SIBFactory.getInstance().getAccessPoint().connect().
    addListener(new TaskListener() {
  @Override
  public void onSuccess(SIBResponse response) {
    // interaction with SIB was here
  }
  @Override
  public void onError(Exception ex) {
    // do something when connection was not established
  }
}
```

Example: Point creation

```
Place gp = Place.getInstance();
Point pt = Point.getInstance();
gp.setHasPoint(pt);
gp.setName("Generated point");
pt.setHasLatitude(Math.random() * 180 - 90);
pt.setHasLongitude(Math.random() * 180 - 90);

// update point
pt.update().addListener(new TaskListener() {
  @Override
  public void onSuccess(SIBResponse response) {
    // update place
    gp.update().addListener(new TaskListener() {
      @Override
      public void onSuccess(SIBResponse response) {
        // it's OK
          }
  ....
}
```

Example: Subscription source code

```
SubscribeQuery.getInstance().addSubscription(Place.
    getClassUri(), new SubscribeListener<Place>() {
  @Override
  public void addItem(Place item) {
    // item was added to SIB
  }

  @Override
  public void removeItem(Place item) {
    // item was removed from SIB
  }

  @Override
  public void onError(Exception ex) {
    // something happen
  }
});
```

Conclusion

- Approach to generating ontology–based object model for the Smart-M3 platform
- Implemented as a source code generator (SmartJavaLog) for the Java language
- SmartJavaLog will be useful for Java developers

Open source code (MIT license)

`https://github.com/seekerk/smartjavalog`

Thank to you attention

Email: marchenk@cs.petrsu.ru