Formal Ontologies in Model-based Software Development

Hele-Mai Haav, Andres Ojamaa, Vahur Kotkas, Pavel Grigorenko, Jaan Penjam
Institute of Cybernetics at TUT

About

In general, ontologies as formal models are used in MBSD for formal representation of

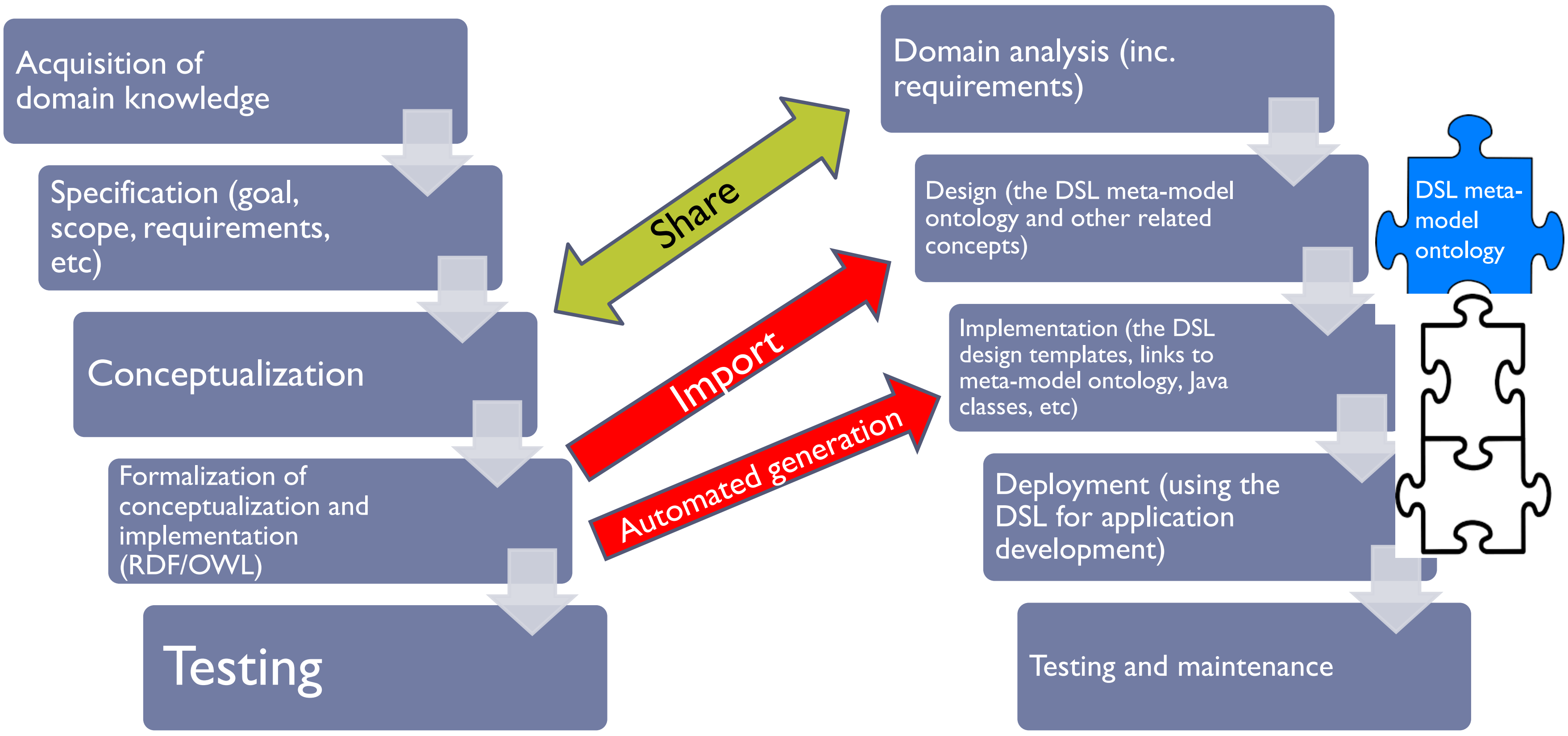
- domain knowledge (i.e. domain models) and
- requirements of a system.

In our approach, in addition to the above

- the system ontology of a MBSD tool (i.e. CoCoViLa)
- and links to external software artefacts available on the web or made available by other MBSD tools are used.

Introducing ontologies into software development process supported by CoCoViLa

Domain ontology development phases | **DSL development phases**

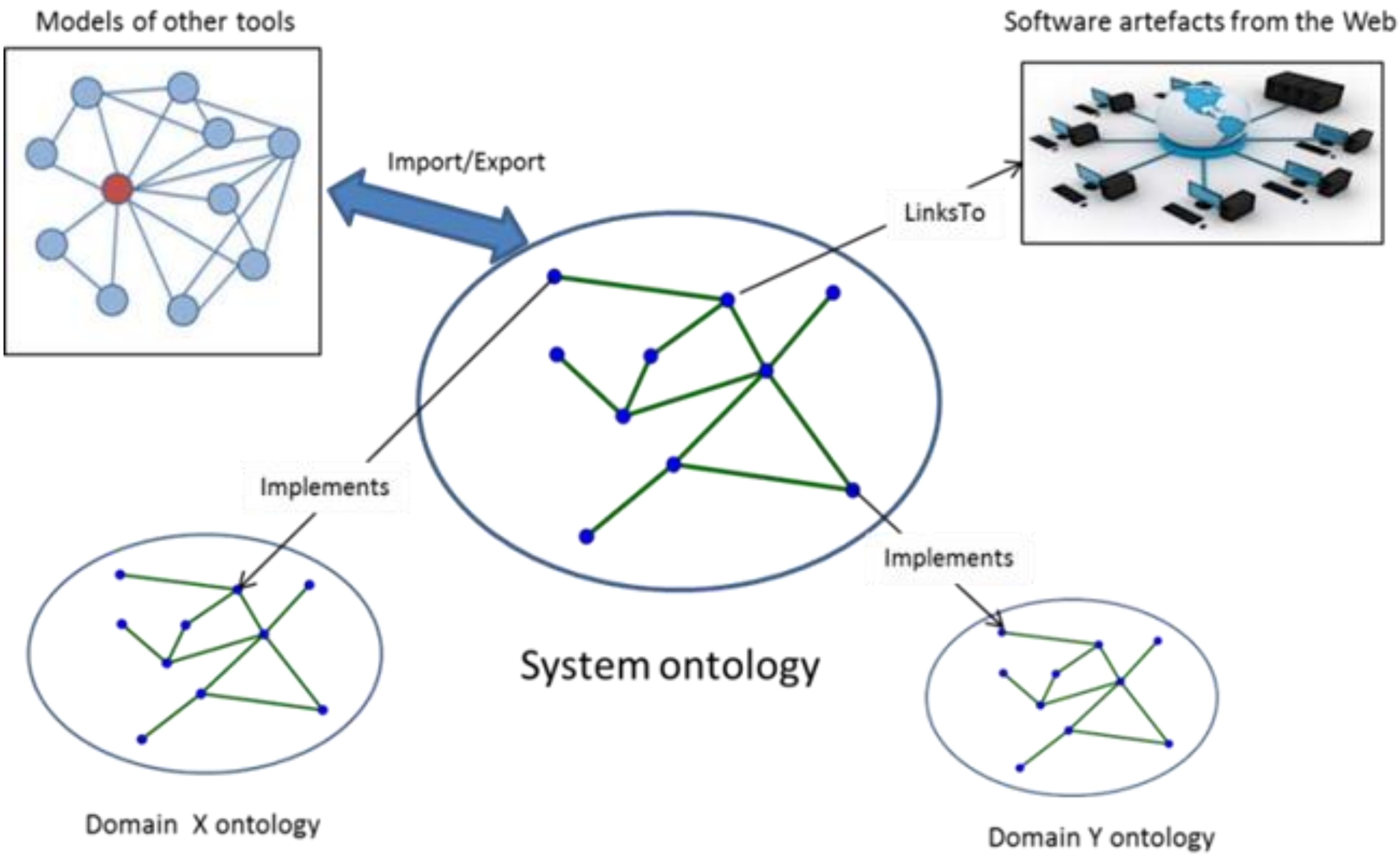


The DSL meta-model ontology

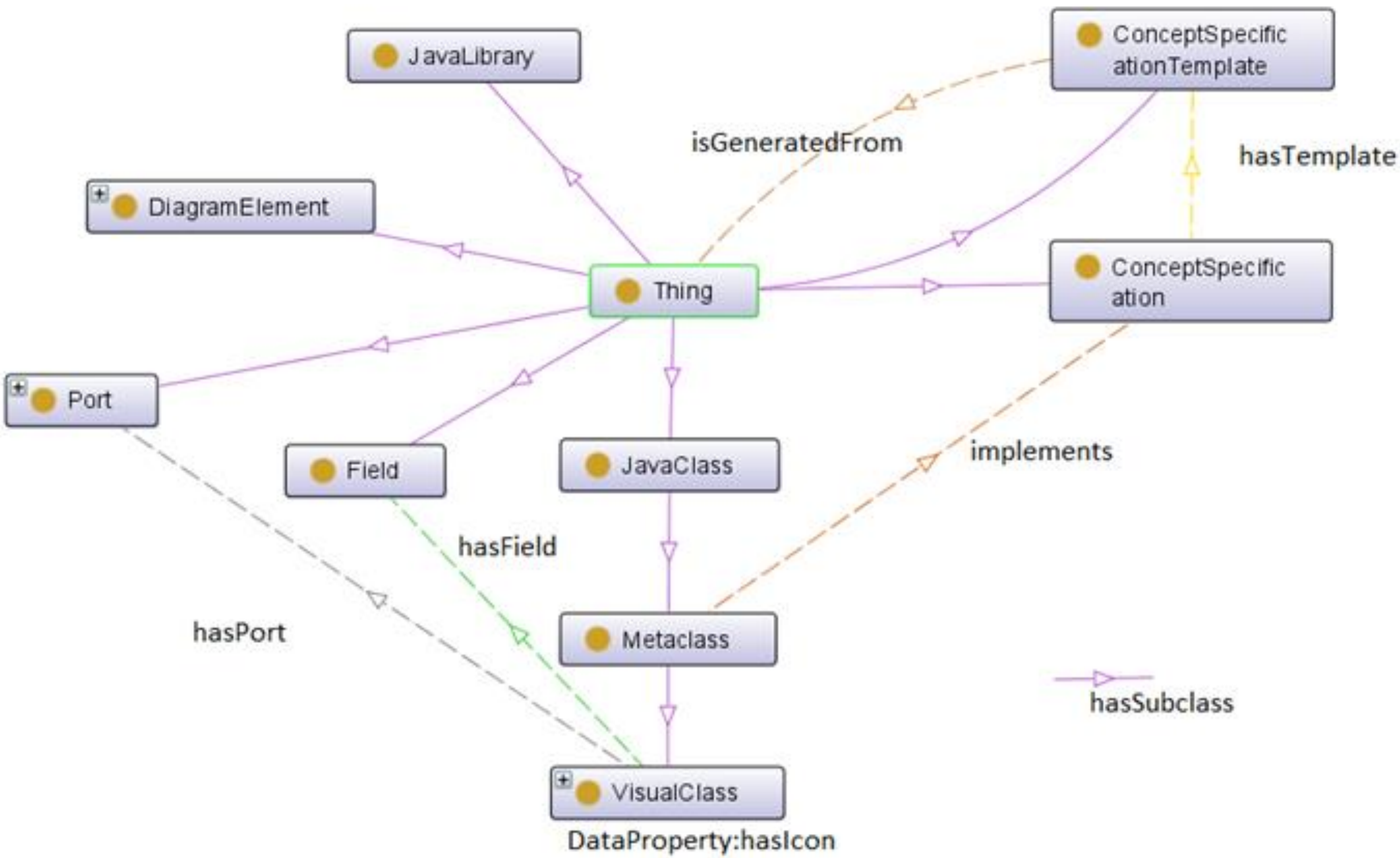
- The *DSL meta-model ontology* is a formal ontology that links together the system ontology and one or more domain ontologies as well as may include links to external software artefacts on the Web.

- The *system ontology* defines a set of system-specific concepts of the MBSD system (or tool) and relationships among them.
- The *domain ontology* provides a specification of a conceptualization of a domain

The concept of DSL meta-model ontology

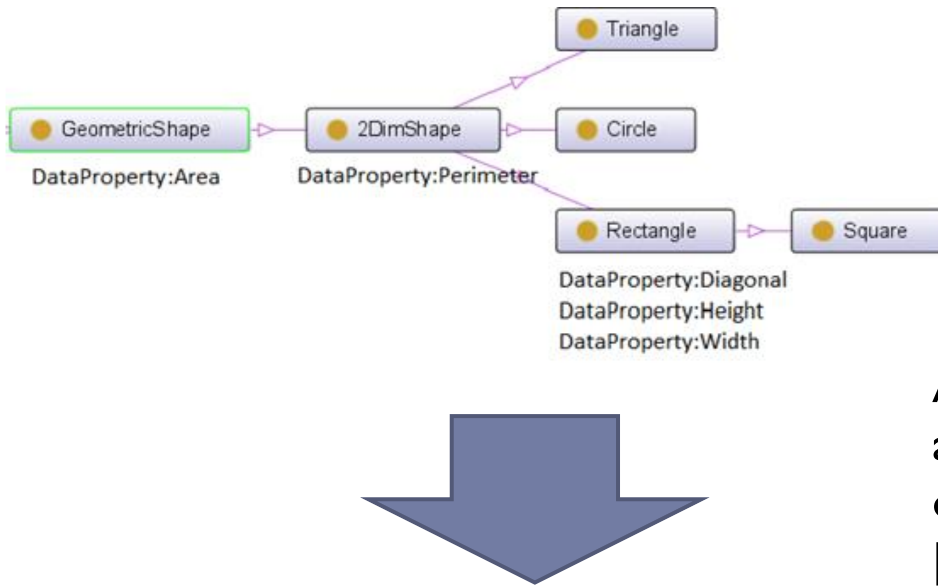


A fragment of the CoCoViLa system ontology



The current version of this ontology includes OWL descriptions of 40 classes, 21 object properties and 16 data properties.

Automated generation of concept specification templates from domain ontology



Automatically generated according to the given set of transformation rules [2]

Templates for the OWL class Rectangle and its super-classes.

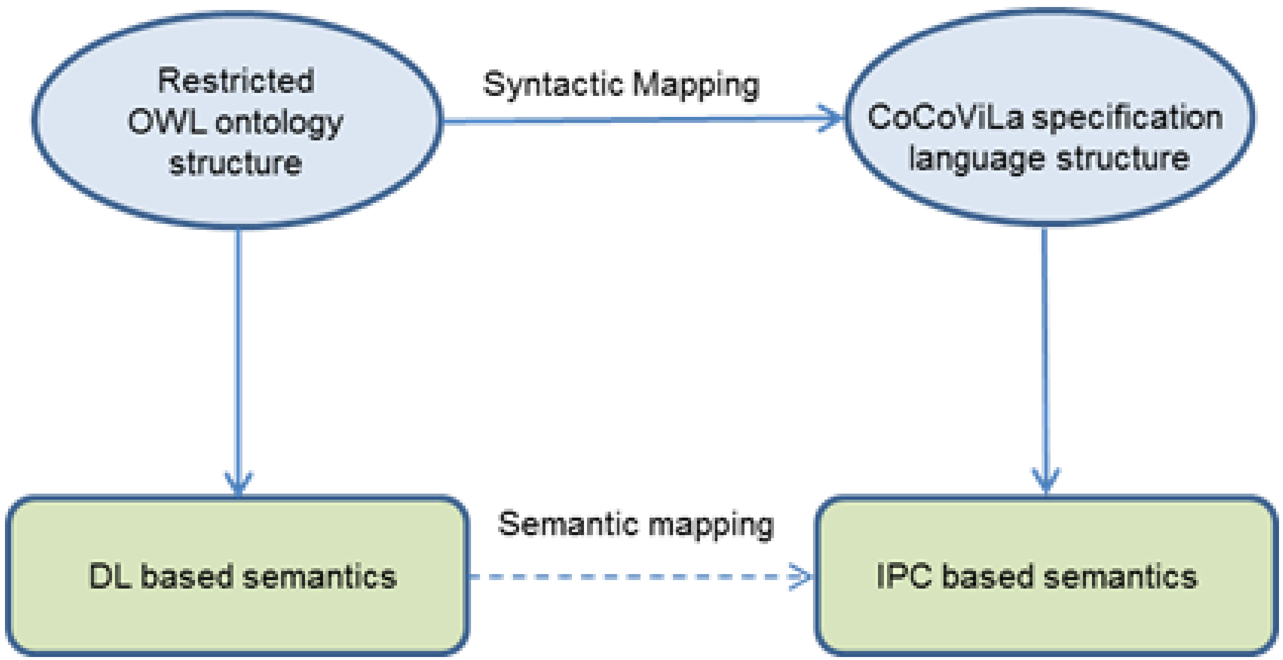
```
specification GeometricShape {
            double Area; }

specification _2DimShape super GeometricShape {
            double Perimeter; }

specification Rectangle super _2DimShape {
            double Height, Width, Diagonal; }
```

Mapping domain ontology to the CoCoViLa language

OWL constructs for object property characteristics, property restrictions and complex classes cannot be mapped to the CoCoViLa modelling language [2].



Mapping …

- For each (or selected) domain ontology class in the DSL meta-model ontology a corresponding instance of the **ConceptSpecificationTemplate** class (from the system ontology) is created and linked to it via the **isGeneratedFrom** object property value.

- This object property value indicates from what domain ontology class the template is automatically generated.

- All this can be automated.

An Example:the Geometry DSL Meta-model Ontology (a fragment)

For DSL designer:

1. Import system ontology („sys“) and domain ontology („geo“) to the DSL meta-model ontology („meta“).
2. Some individuals and links are created automatically.
3. Create additional individuals and their relationships

SubClassOf(sys:MetaClass sys:JavaClass)
SubClassOf(sys:VisualClass sys:MetaClass)
DataPropertyDomain(sys:hasIcon sys:VisualClass)
DataPropertyRange(sys:hasIcon xsd:anyURI)
SubClassOf(geo:Rectangle geo:2DimShape)

Imported

Individuals and their relationships

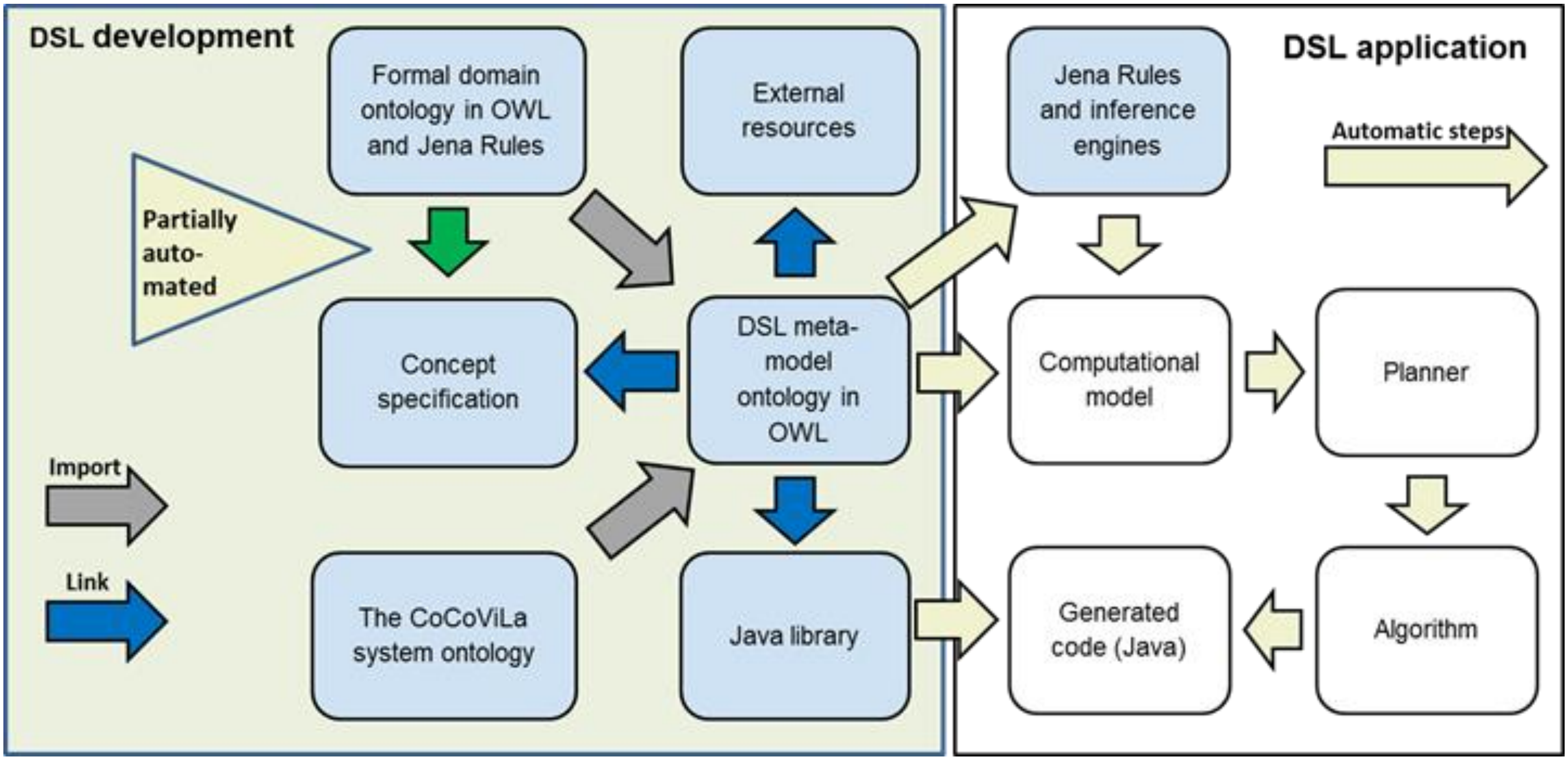
Autom. created

ClassAssertion(sys:ConceptSpecificationTemplate meta:CST1)
ClassAssertion(sys:ConceptSpecification meta:CS1) ?
ClassAssertion(sys:VisualClass meta:VC1)
ClassAssertion(geo:Rectangle meta:Rectangle)

ObjectPropertyAssertion(sys:isGeneratedFrom meta:CST1 meta:Rectangle)
ObjectPropertyAssertion(sys:hasTemplate meta:CS1 meta:CST1) ?
ObjectPropertyAssertion(sys:implements meta:VC1 meta:CS1)

DataPropertyAssertion(sys:hasIcon meta:VC1
"http://www.cs.ioc.ee/cocovila/icons/rectangle.png "^^xsd:anyURI)

The architecture of the CoCoViLa extension



Implementation details

- Contact Andres!

Benefits

- It is easy to incorporate the domain terminology into the DSL at the early development stages due to formalization of domain ontology.
- Separation of different kinds of knowledge about the system, domain and a DSL into modular OWL ontologies makes the knowledge more reusable.
- Automatic generation of design templates of the DSL meta-model.
- Linking the DSL meta-model ontology components with external resources over the Web.
- Capturing the evolution of a domain in the DSL via automated transformations [2].
- Using the DSL meta-model ontology makes it possible to automatically check its consistency using DL reasoning facilities used for debugging DSL meta-models.
- The system ontology is always related to the running version of the system. Creation of the system documentation on the basis of the system ontology guarantees that documentation is up-to-date and consistent with the system.

Limitations

- Problems of creation of formal domain ontologies as ontology engineering techniques do not constitute a part of existing traditional software development methodologies. This may create initial complexity
- For semantic integration of artefacts from external tools and models, the approach requires the commitment to a common system ontology or availability of system ontologies of these tools

Conclusion

- Representing domain models and the system model as OWL ontologies and linking them together to form a unified DSL meta-model ontology makes it possible to effectively integrate software artefacts that constitute a DSL meta-model as well as link it with external resources over the Web.
- This facilitates dynamic loading (instantiation) of software artefacts of DSL meta-models to a DSL development tool.
- We have prototypically implemented our approach as an extension to the CoCoViLa DSL modelling tool.

Acknowledgements.

This research was supported by Estonian Research Council institutional research grant no. IUT33-13, and by the ERDF through the ICT project MBJSDT and Estonian national CoE project EXCS.

References

1. H.-M. Haav, A. Ojamaa, P. Grigorenko, V. Kotkas. Ontology-based integration of software artefacts for DSL development. In R. Meersman, H. Panetto, I. Ciuciu et al., (eds) Proceedings of OTM Workshops 2015. (INBAST 2015), LNCS, Springer (2015), (to appear)
2. Ojamaa, A., Haav H-M., Penjam J.: Semi-automated Generation of DSL Meta Models from Formal Domain Ontologies, In: Manolopoulos, Y., Bellatreche, L. (eds) Proceedings of the 5th International Conference on Model & Data Engineering (MEDI 2015), LNCS, vol. 9344, pp. 1-12, Springer (2015) (to appear)

Thank you!