Ontology and DSL co-evolution using graph transformations methods

Ulitin Boris

Scientific director: prof., PhD Babkin Eduard

MOSCOW, 2017

Structure of presentation

0. Introduction&Main questions.

1. Context:

1.1 Graph-representation of the Ontology;

1.2 Structure&Graph-representation of Domain-specific languages (DSLs);

1.3 Definition&Classification of the model natural evolution.

2. Solution design:

2.1 Scheme of created approach;

2.2 Using technologies.

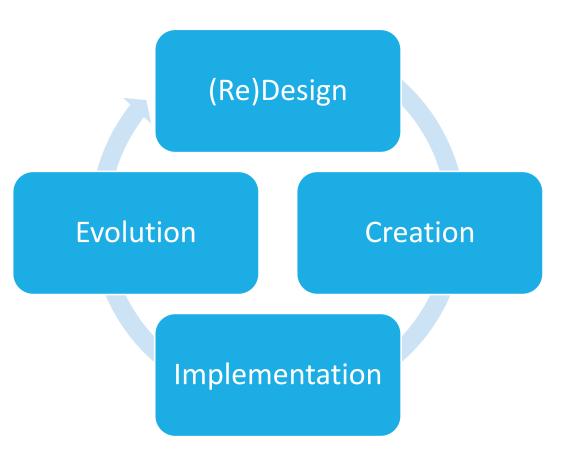
3. Use case.

4. Further steps and conclusion.

Definition and lifecycle of the DSL

 Domain-specific language (DSL) – a computer language specialized to a particular application domain (M. Fowler "Domain Specific Languages").

It means, that DSL contains only terms, which are needed for working with a specific domain and for solving tasks of this domain.



Classification of DSL evolution

Natural

Bell, Fowler, Sprinkle, Karsai, Gómez-Abajo, Mengerink,...

Caused by changes in the target domain **Behavioral**

Sprinkle, Karsai

Caused by changes in users needs, skills and experience

Definition of the Ontology

•Ontology* – a representational artifact, comprising a taxonomy as proper part, whose representations are intended to designate some combination of universals, defined classes, and certain relations between them.

Application ontology* describes concepts that depend both on a particular domain and a task (and often combine specializations of both the corresponding domain and task ontologies).

*G. Guizzardi "Ontological foundations for structural conceptual models"

Main questions

- 1. Why and how can ontology be used as a model of the subject area for DSL?
- 2. How to organize the bridge between the ontology and DSL?
- 3. How to provide the co-evolution (=corresponding changes without recreation) of the domain (=ontology) and DSL?



Why and how can ontology be used as a model of the subject area for DSL?

How to organize the bridge between the ontology and DSL?

1.1 Graph-representation of the Ontology

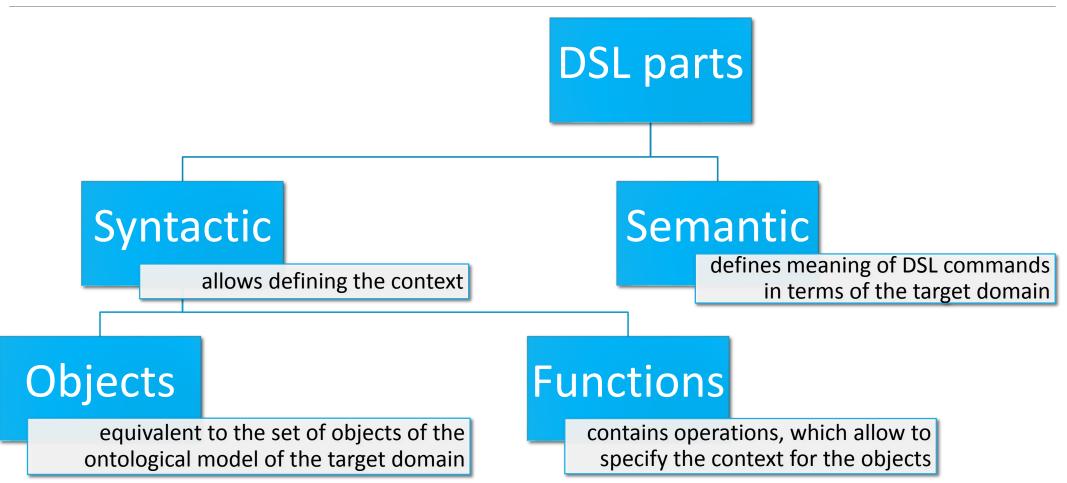
Ontology can be represented as a triple (*O*, *R*, *F*):

- $O = U \cup C$ is a set of objects;
- *R* is a set of relations between elements of *O*;
- *F* is an interpretation function.

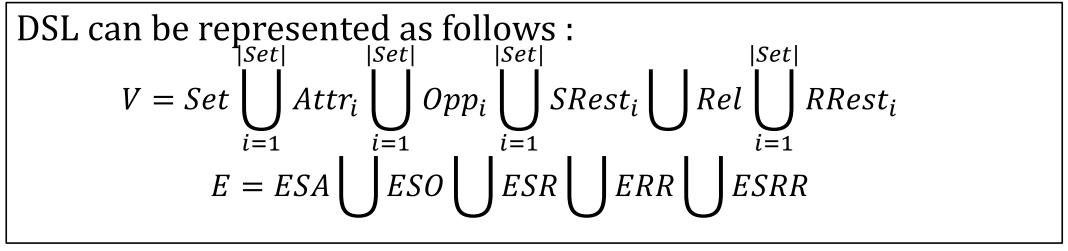


There is a graph-representation of the Ontology

1.2 Structure of the DSL



1.2 Graph-representation of the DSL





There is a similar to the Ontology graph-model scheme:

(*Obj*, *Rel*, {*Rest*, *Opp*})

Results

- Both Ontology and DSL have similar triples in graphrepresentation;
- •We can organize the transferring elements from one of them into another and vise versa;
- •To achieve this goal graph-transformation rules can be used.



How to provide the co-evolution of the domain (=ontology) and DSL?

1.3 Definition of the model evolution

From the formal point of view every model is (*E*, *R*) where *E* is a set of entities:

$$e_i = \{attr_{i_1}, attr_{i_2}, \dots, attr_{i_M}\}, M \in \mathbb{N}, i = 1, N$$

and *R* is a set of relations between them.



Evolution of the model is a process of changes in the structure of this model

1.3 Classification of the model evolution

Vertical:

Means the change in level of conceptualization (perspective) of the model:

 (E^1, R^1) is a result of vertical evolution of the model (E, R) if $|E| \neq |E^1|$ and $|R| \neq |R^1|$

Horizontal:

Means preserving the level of conceptualization, but changing the sets of attributes for some entities, or changing the set of relations:

 (E^1, R^1) is a result of horizontal evolution of the model (E, R) if $|E| = |E^1|$ and $\exists r_i \in R, r_j \in R^1$: $r_i \notin R^1 \lor r_j \notin R$ and $\exists e_i \in E, e_i^1 \in E^1$: $Attr_i \cap Attr_i^1 = Attr_i \land |Attr_i| \neq |Attr_i^1|$

Results¹

In case of Vertical evolution:

We have to **<u>align</u>** the **<u>entities</u>** and <u>**relations**</u> between them.

It means, that 4 general graph-transformation rules have to be defined: <u>to transfer</u> <u>entities</u> between two graph-models (added/deleted) and <u>to transfer relationships</u> (added/deleted).

In case of Horizontal evolution:

We have to <u>align</u> lists of entities' <u>attributes</u> and/or set of <u>relation</u> between entities.

It means, that 4 general graph-transformation rules have to be defined: <u>to transfer</u> <u>new attributes</u> between two graph-models and <u>to drop deleted</u>; <u>to transfer new</u> <u>relationships</u> and <u>to drop deleted</u>.

Results²

•All transformation rules can be separated into 3 categories:

synchronizing entities:

- for added;
- for deleted;

synchronizing attributes of entities:

- for added;
- for deleted;

synchronizing relationships:

- for added;
- for deleted.

Solution design

2.1 Scheme of created approach

<u>Stage I:</u> Identification of the ontology (OWL description) <u>Stage II:</u> Identification of the DSL metamodel (KM3 description)

<u>Stage III:</u> Definition of transformation rules <u>Stage IV:</u> Applying the transformation rules

2.2 Using technologies

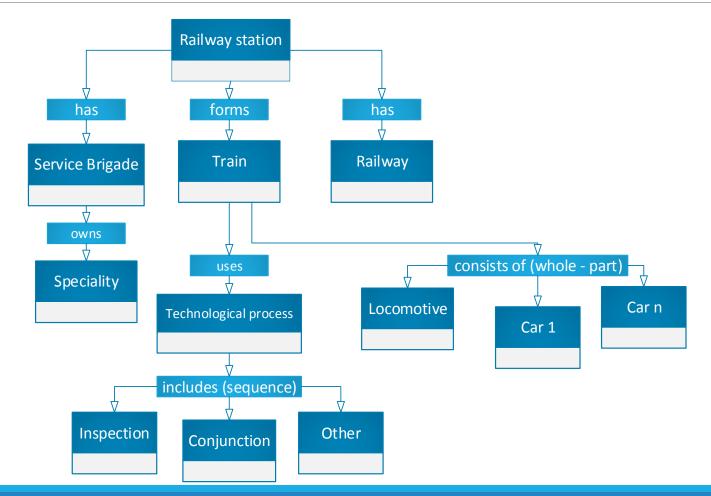
- Eclipse IDE platform;
- •*Eclipse plugins*:
 - TCS (Textual Concrete Syntax) for definition of concrete DSL syntax undo the metamodel ontological terms;
 - *ATL Transformations* for automated connections between the OWL and KM3.

- 🔡 🐚 💷 🔅 - O - 🍋 - 🗁 🖋 - 🔛 💷 🔳 🖢 😓 - 🖓 - 🏷 - → -

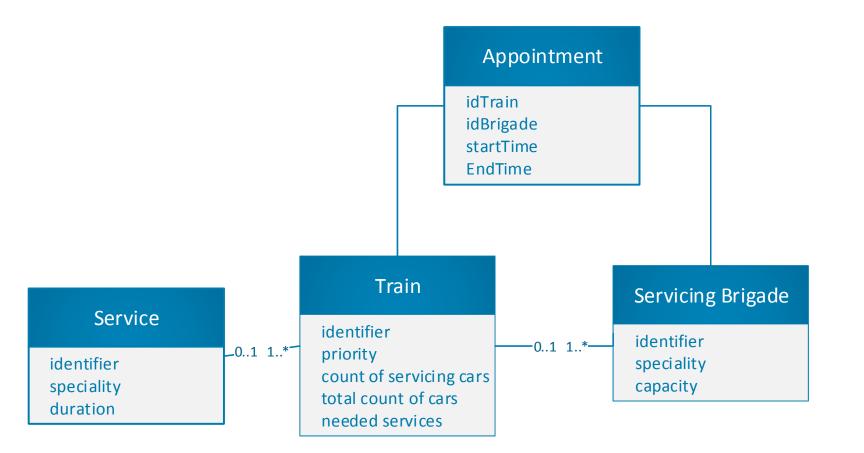
Taking/Ontbioggound 32 D DLkm3 32 O TONLEDGLaf 32 1 krdf:RDF> 1 package DSL { 1 module OWLtoDSL; * 2 cowl:Ontology rdf:about=""> 2 class Train { 3 attribute identifier : int; 3 attribute countOfService; 5 from a: OWL:OBLass 2DSLClass { * 6 cowl:Class rdf:ID="ServiceBri 6 attribute countOfService; 5 from a: OWL:OBLass CDULClass (not exists (see 6 to b: DSLIDSLClass, * * 7 cowl:Class rdf:ID="Train" /> 9 * * * * * 9 cowl:Class rdf:ID="Train" /> 9 * * * * * * * 11 cowl:Class rdf:ID="Conjunctio 10 class ServicingBrigade { * <th></th> <th></th> <th></th> <th></th> <th></th> <th></th>										
2 cowl:Ontology rdf:about=""> 3 <u>crdfs:labelSRailway</u> alloc 4 5 6 cowl:Class rdf:ID="ServiceBri 6 cowl:Class rdf:ID="ServiceBri 7 cowl:Class rdf:ID="Railway" / 9 cowl:Class rdf:ID="Railway" / 9 cowl:Class rdf:ID="Train" /> 9 cowl:Class rdf:ID="Train" /> 9 cowl:Class rdf:ID="Train" /> 9 cowl:Class rdf:ID="Train" /> 10 cowl:Class rdf:ID="Inspection 11 attribute identifier : int 12 crdfs:subClassOf rdf:reso 13 c/owl:Class rdf:ID="Conjunctio 14 crdfs:subClassOf rdf:reso 15 crdfs:subClassOf rdf:reso 16 c/owl:Class rdf:ID="Conjunctio 17 cowl:Class rdf:ID="Conjunctio 18 crdfs:subClassOf rdf:reso 19 class Appointment { 10 class Appointment { 11 crdfs:subClassOf rdf:reso 12 create OUT : DSL refining IN : OWL; 3 attribute totalCountOfCare 10 class ServicingBrigade { 11 cowl:Class rdf:ID="Conjunctio 12 crdfs:subClassOf rdf:reso 13 crdfs:subClassOf rdf:reso 14 crdfs:subClassOf rdf:reso 15 crdfs:subClassOf rdf:reso 16 class Appointment { 17 cowl:Class rdf:ID="Carl"> 20 cowl:Class rdf:ID="Carl"> 20 class Appointment { 21 crdfs:subClassOf rdf:reso 22 creference [1] Train; 23 cowl:Class rdf:ID="Carl"> 20 cowl:Class rd	RailwayOntology.owl		DSL.km3 🔀		ē	NWLtoDSL.atl 🔀				
3 <pre>crdfs:label>Railway alloc3attribute identifier : int34attribute priority : int;4attribute priority : int;55attribute countOfServicing6to b: DSLIDSLClass {6attribute neededServices :7c: DSL!Constructor8attribute neededServices :7c: DSL!Constructor9attribute neededServices :7c: DSL!Constructor9attribute application1010class rdf:ID="Train" />9910attribute application1111attribute application11attribute application1112attribute application1114drop1315attribute application : int;13do {1615}161717class Applintment {171818attribute idServicingBriga19from a: OWLClassWithParents2DSLClassWitl19attribute idServicingBriga161714drop c: DSL!Constructor (c.i151617attribute idServicingBriga1617attribute idServicingBriga1618<!--</th--><th colspan="2">1krdf:RDF></th><th colspan="2">1package DSL {</th><th>Ш</th><th>1 module OWLtoDSL; ^</th></pre>	1krdf:RDF>		1package DSL {		Ш	1 module OWLtoDSL; ^				
44attribute priority : int; attribute countOfServicing attribute totalCountOfCars oul:Class rdf:ID="ServiceBri cowl:Class rdf:ID="Speciality cowl:Class rdf:ID="Railway" / 84attribute priority : int; attribute countOfServicing attribute totalCountOfCars attribute neededServices : 85forma: OWLClass2DSLClass { 556cowl:Class rdf:ID="Speciality cowl:Class rdf:ID="Train" /> 97attribute identifier : int attribute identifier : int attribute speciality : int 1110class ServicingBrigade { 10101011cowl:Class rdf:ID="Inspection 11attribute speciality : int attribute speciality : int 1211from b: DSL! DSL Class (not exists (second 812crdfs:subClassOf rdf:reso 1212attribute speciality : int; attribute capacity : int; 1313do for attribute identifier : int 1414cowl:Class rdf:ID="Comotive 1513attribute idfrain : int; 1413do for attribute idfrain : int; 151617cowl:Class rdf:ID="Comotive 1617class Appointment { 161718cowl:Class rdf:ID="Carl">10class Appointment { 1619cowl:Class rdf:ID="Carl">20attribute idfrain : int; 181819cowl:Class rdf:ID="Carl">20attribute idfrain : int; 1310cowl:Class rdf:ID="Carl">20attribute idfrain : int; 1310cowl:Class rdf:ID="Carl">20attribute idfrain : int; 1310c	2	<owl:ontology rdf:about=""></owl:ontology>	2	class Train {	Ш	2 create OUT : DSL refining IN : OWL;				
55attribute countOfServicing attribute totalCountOfCars attribute neededServices:5froma: OWL!OWLClass (not exists (se6cowl:Class rdf:ID="Speciality soul:Class rdf:ID="Railway" /7attribute neededServices:7c: DSL!DSLClass, c: DSL!Constructor8cowl:Class rdf:ID="Railway" /8}9910cowl:Class rdf:ID="Treinn" />910class ServicingBrigade { attribute identifier : int attribute speciality : int attribute speciality : int int10route DropDeletedOWLClassInDSLClass { of p11cowl:Class rdf:ID="Conjunctio soul:Class11attribute speciality : int attribute capacity : int; int;13do q do q13c/owl:Class rdf:ID="Conjunctio soul:Class14reference [1*] Specialit attribute idTrain : int; int;13do q do q14cowl:Class rdf:ID="Conjunctio soul:Class17class Appointment { attribute idTrain : int; int18rule owlLClassWithParents2DSLClassWith int15cowl:Class rdf:ID="Cor1">20attribute idTrain : int; int18rule owlLlassWithParents2DSLClassWith int14cowl:Class rdf:ID="Car1">20attribute idTrain : int; int18rule owlLlass20cowl:Class rdf:ID="Car1">20attribute endTime : Date; int20tob: DSL!DSLClass (b.name = a.nam21c/owl:Classreference [11] Train; int22cowl:Class (b.name = a.parent.nai23b.parent.name = a.parent.nai22 </th <th>3</th> <th><rdfs:label>Railway alloc</rdfs:label></th> <th>3</th> <th>attribute identifier : int</th> <th>Ш</th> <th>3</th>	3	<rdfs:label>Railway alloc</rdfs:label>	3	attribute identifier : int	Ш	3				
6cowl:Class rdf:ID="ServiceBri (cowl:Class rdf:ID="Railway") (cowl:Class rdf:ID="Train" /> (cowl:Class rdf:ID="Train" /> 	4		4	attribute priority : int;	Ш	4⊖rule OWLClass2DSLClass {				
7cowl:Class rdf:ID="Speciality7attribute neededServices :7c: DSLIConstructor8 </th <th>5</th> <th></th> <th>5</th> <th>attribute countOfServicing</th> <th>Ш</th> <th>5 from a: OWL!OWLClass (not exists (se</th>	5		5	attribute countOfServicing	Ш	5 from a: OWL!OWLClass (not exists (se				
8 cowl:Class rdf:ID="Railway" / 8 } 9 cowl:Class rdf:ID="Train" /> 9 10 cowl:Class rdf:ID="Technologi 10 class ServicingBrigade { 10 10 rule DropDeletedOWLClassInDSLClass { 11 cowl:Class rdf:ID="Technologi 10 class ServicingBrigade { 11 from b: DSL! DSL Class (not exists (: 12 cdfs:subClassOf rdf:reso 12 attribute speciality : int; 13 do forp 13 c/owl:Class 13 attribute capacity : int; 13 do { 14 cowl:Class rdf:ID="Conjunctio 14 reference [1*] Specialit 14 drop c: DSL!Constructor (c.i 15 16 17 class Appointment { 17 16 17 class Appointment { 17 18 rule OWLClassWithParents2DSLClassWitl 19 owl:class rdf:ID="Carl"> 20 attribute idTrain : int; 18 rule OWLClassWithParents2DSLClassWitl 19 attribute startTime : Date; 20 to b: DSLIDSLClass (b.name = a.name; <tr< th=""><th>6</th><th><owl:class rdf:id="ServiceBri</th><th>6</th><th>attribute totalCountOfCars</th><th>Ш</th><th>6 to b: DSL!DSLClass,</th></tr><tr><th>9(owl:Class rdf:ID=" train"=""></owl:class> 99910(owl:Class rdf:ID="Technologi 1110class ServicingBrigade { attribute identifier : int attribute identifier : int attribute apeciality : int; 1310class ServicingBrigade { 111112(rdfs:subclassOf rdf:reso (rdfs:subclassOf rdf:reso 512attribute speciality : int; attribute capacity : int; 1312to drop13(rdfs:subclassOf rdf:reso (rdfs:subclassOf rdf:reso 614reference [1*] Specialit 513drop c: DSL!Constructor (c.l 414(rdfs:subclassOf rdf:reso 61617class Appointment { 1917181814(rdfs:subclassOf rdf:reso 16161718attribute idTrain : int; 1919181519attribute idServicingBriga attribute idServicingBriga 2019attribute endTime : Date; 201010b.parent.name = a.parent.name 2120(rdfs:subclassOf rdf:reso 2224242424242424(rdfs:subclassOf rdf:reso 242424242424</th><th>7</th><th><owl:class rdf:id="Speciality</th><th>7</th><th>attribute neededServices :</th><th>Ш</th><th>7 c: DSL!Constructor</th></tr><tr><th>10<owl:Class rdf:ID=" technologi<="" th="">10class ServicingBrigade { attribute identifier : int attribute identifier : int attribute speciality : int 310ruleDropDeletedOWLClassInDSLClass { b: DSL! DSL Class (not exists (: attribute speciality : int attribute capacity : int; id13attribute speciality : int attribute capacity : int; id13attribute speciality : int attribute capacity : int; id14from b: DSL! DSL Class (not exists (: drop14drop c: DSL!Constructor (c.) id14reference [1*] Specialit15161617class Appointment { attribute idTrain : int; id1619attribute idServicingBrigs attribute idServicingBrigs attribute startTime : Date; id19attribute endTime : Date; id1020b.parent.name = a.parent.name id23<!--</th--><th>8</th><th><owl:class <="" rdf:id="Railway" th=""><th>8</th><th>}</th><th>Ш</th><th>8 }</th></owl:class></th></owl:class></th></tr<>	6	<owl:class rdf:id="ServiceBri</th><th>6</th><th>attribute totalCountOfCars</th><th>Ш</th><th>6 to b: DSL!DSLClass,</th></tr><tr><th>9(owl:Class rdf:ID=" train"=""></owl:class> 99910(owl:Class rdf:ID="Technologi 1110class ServicingBrigade { attribute identifier : int attribute identifier : int attribute apeciality : int; 1310class ServicingBrigade { 111112(rdfs:subclassOf rdf:reso (rdfs:subclassOf rdf:reso 512attribute speciality : int; attribute capacity : int; 1312to drop13(rdfs:subclassOf rdf:reso (rdfs:subclassOf rdf:reso 614reference [1*] Specialit 513drop c: DSL!Constructor (c.l 414(rdfs:subclassOf rdf:reso 61617class Appointment { 1917181814(rdfs:subclassOf rdf:reso 16161718attribute idTrain : int; 1919181519attribute idServicingBriga attribute idServicingBriga 2019attribute endTime : Date; 201010b.parent.name = a.parent.name 2120(rdfs:subclassOf rdf:reso 2224242424242424(rdfs:subclassOf rdf:reso 242424242424	7	<owl:class rdf:id="Speciality</th><th>7</th><th>attribute neededServices :</th><th>Ш</th><th>7 c: DSL!Constructor</th></tr><tr><th>10<owl:Class rdf:ID=" technologi<="" th="">10class ServicingBrigade { attribute identifier : int attribute identifier : int attribute speciality : int 310ruleDropDeletedOWLClassInDSLClass { b: DSL! DSL Class (not exists (: attribute speciality : int attribute capacity : int; id13attribute speciality : int attribute capacity : int; id13attribute speciality : int attribute capacity : int; id14from b: DSL! DSL Class (not exists (: drop14drop c: DSL!Constructor (c.) id14reference [1*] Specialit15161617class Appointment { attribute idTrain : int; id1619attribute idServicingBrigs attribute idServicingBrigs attribute startTime : Date; id19attribute endTime : Date; id1020b.parent.name = a.parent.name id23<!--</th--><th>8</th><th><owl:class <="" rdf:id="Railway" th=""><th>8</th><th>}</th><th>Ш</th><th>8 }</th></owl:class></th></owl:class>	8	<owl:class <="" rdf:id="Railway" th=""><th>8</th><th>}</th><th>Ш</th><th>8 }</th></owl:class>	8	}	Ш	8 }
11cowl:Class rdf:ID="Inspection11attribute identifier: int11fromb: DSL! DSL Class (not exists (:12crdfs:subClassOf rdf:reso12attribute speciality: int12to drop13c/owl:Class>13attribute capacity: int;13do {14cowl:Class rdf:ID="Conjunctio14reference [1*] Specialit14drop c: DSL!Constructor (c.)15crdfs:subClassOf rdf:reso15}161516c/owl:Class rdf:ID="Locomotive17class Appointment {18attribute idTrain : int;19c/owl:Class rdf:ID="Car1">20attribute startTime: Date;20to b: DSL!DSLClass (b.name = a.name21crdfs:subClassOf rdf:reso22reference [11] Train;22b.parent.name = a.parent.name23cowl:Class24}2524242424crdfs:subClassOf rdf:reso2424242425c/owl:Class2525252526	9	<owl:class rdf:id="Train"></owl:class>	9		Ш	9				
12 <rdfs:subclassof rdf:reso<="" td="">12attribute speciality : int attribute capacity : int; 1312 todrop13attribute capacity : int; 1413attribute capacity : int; 1414drop c: DSL!Constructor (c.1515}1615161717class Appointment { 1916161718<!--</th--><th>10</th><th><owl:class conjunctio<="" pre="" rdf:id="Technologi</th><th></th><th>class ServicingBrigade {</th><th>Ш</th><th>10 rule DropDeletedOWLClassInDSLClass {</th></tr><tr><th>13</owl:Class>13attribute capacity : int;13do {14reference [1*] Specialit14reference [1*] Specialit1415</</th><th>11</th><th></th><th></th><th>attribute identifier : <u>int</u></th><th>Ш</th><th>11 from b: DSL! DSL Class (not exists (:</th></tr><tr><th>14<pre>cowl:Class rdf:ID=">14reference [1*] Specialit14drop c: DSL!Constructor (c.115<!--</th--><th>12</th><th><rdfs:subclassof rdf:reso<="" th=""><th></th><th>attribute <u>speciality</u> : <u>int</u></th><th>Ш</th><th>12 to drop</th></rdfs:subclassof></th></owl:class></th></rdfs:subclassof>	10	<owl:class conjunctio<="" pre="" rdf:id="Technologi</th><th></th><th>class ServicingBrigade {</th><th>Ш</th><th>10 rule DropDeletedOWLClassInDSLClass {</th></tr><tr><th>13</owl:Class>13attribute capacity : int;13do {14reference [1*] Specialit14reference [1*] Specialit1415</</th><th>11</th><th></th><th></th><th>attribute identifier : <u>int</u></th><th>Ш</th><th>11 from b: DSL! DSL Class (not exists (:</th></tr><tr><th>14<pre>cowl:Class rdf:ID=">14reference [1*] Specialit14drop c: DSL!Constructor (c.115<!--</th--><th>12</th><th><rdfs:subclassof rdf:reso<="" th=""><th></th><th>attribute <u>speciality</u> : <u>int</u></th><th>Ш</th><th>12 to drop</th></rdfs:subclassof></th></owl:class>	12	<rdfs:subclassof rdf:reso<="" th=""><th></th><th>attribute <u>speciality</u> : <u>int</u></th><th>Ш</th><th>12 to drop</th></rdfs:subclassof>		attribute <u>speciality</u> : <u>int</u>	Ш	12 to drop		
15 </th <th>13</th> <th></th> <th></th> <th></th> <th>Ш</th> <th>L L</th>	13				Ш	L L				
16161617 <owl:class car1"="" rdf:id="Locomotive</td>17class Appointment {1718<rdfs:subClassOf rdf:reso</td>18attribute idTrain : int;181719</owl:Class>19attribute idServicingBriga19from a: OWLClassWithParents2DSLClassWith20<owl:Class rdf:ID=">20attribute startTime : Date20to b: DSL!DSLClass (a.parent->exis:21<rdfs:subclassof rdf:reso<="" td="">21attribute endTime : Date;20to b: DSL!DSLClass (b.name = a.name)21<rdfs:subclass rdf:id="Car2">22reference [11] Train;23}23<owl:class rdf:id="Car2">24}25524<rdfs:subclassof rdf:reso<="" td="">24>24>25<!--</th--><th>14</th><th>3</th><th></th><th>reference [1*] Specialit</th><th>Ш</th><th>14 drop c: DSL!Constructor (c.)</th></rdfs:subclassof></owl:class></rdfs:subclass></rdfs:subclassof></owl:class>	14	3		reference [1*] Specialit	Ш	14 drop c: DSL!Constructor (c.)				
17 <owl:class car1"="" rdf:id="Locomotive
(rdfs:subClassOf rdf:reso)17class Appointment {
attribute idTrain : int;
19attribute idTrain : int;
191718<rdfs:subClassOf rdf:reso
(owl:Class rdf:ID="> 20attribute idServicingBriga attribute startTime : Date 211721<rdfs:subclassof rdf:reso<br=""></rdfs:subclassof>2121attribute endTime : Date; 2220to b: DSL!DSLClass (b.name = a.name 212222reference [11] Train; 2323reference [11] Servicing 232324<!--</th--><td>15</td><td><rdfs:subclassof rdf:reso<="" td=""><th>15</th><td>}</td><th>Ш</th><td>15 }</td></rdfs:subclassof></td></owl:class>	15	<rdfs:subclassof rdf:reso<="" td=""><th>15</th><td>}</td><th>Ш</th><td>15 }</td></rdfs:subclassof>	15	}	Ш	15 }				
18 <rdfs:subclassof rdf:reso<="" th="">18attribute idTrain : int;18attribute idTrain : int;1919attribute idServicingBriga19attribute idServicingBriga20<owl:class rdf:id="Car1">20attribute startTime : Data20to b: DSL!OWLClass (a.parent->exis21<rdfs:subclassof rdf:reso<="" td="">21attribute endTime : Data;20to b: DSL!DSLClass (b.name = a.name2221attribute endTime : Data;21b.parent.name = a.parent.name2322reference [11] Servicing23}24242425<!--</th--><td></td><td></td><th></th><td></td><th>Ш</th><td>16 }</td></rdfs:subclassof></owl:class></rdfs:subclassof>					Ш	16 }				
1919attribute idServicingBriga attribute startTime : Date 2019froma: OWL!OWLClass (a.parent->exis 2020 <owl:class rdf:id="Car1"> 20attribute startTime : Date 21attribute endTime : Date; 2120tob: DSL!DSLClass (b.name = a.name 2121<rdfs:subclassof rdf:reso<br=""></rdfs:subclassof>2222reference [11] Train; 2323b.parent.name = a.parent.name 2323<owl:class rdf:id="Car2"> 2424}24}24<rdfs:subclassof rdf:reso<br=""></rdfs:subclassof>2524>24}25<!--</th--><td></td><td></td><th></th><td></td><th>Ш</th><td>17</td></owl:class></owl:class>					Ш	17				
20 <owl:class rdf:id="Car1">20attribute startTime : Date20tob: DSL!DSLClass (b.name = a.name21<rdfs:subclassof rdf:reso<="" td="">21attribute endTime : Date;21dt2122</rdfs:subclassof></owl:class> 22reference [11] Train;22b.parent.name = a.parent.name23 <owl:class rdf:id="Car2">242424<rdfs:subclassof rdf:reso<="" td="">242425<!--</th--><td></td><td></td><th></th><td></td><th>Ш</th><td></td></rdfs:subclassof></owl:class>					Ш					
21 <rdfs:subclassof rdf:reso<="" td="">21attribute endTime : Date;21do {2222reference [11] Train;22b.parent.name = a.parent.name2323reference [11] Servicing2324<!--</th--><td></td><td></td><th></th><td></td><th>Ш</th><td>· · ·</td></rdfs:subclassof>					Ш	· · ·				
22 22 22 reference [11] Train; 22 b.parent.name = a.parent.name 23 cowl:Class rdf:ID="Car2"> 23 reference [11] Servicing 22 3 24 crdfs:subClassOf rdf:reso 24 3 24 3 25 c/owl:Class> 25 25					Ш					
23 <owl:class rdf:id="Car2"> 23 reference [11] Servicing 23 } 24 <rdfs:subclassof rdf:reso<="" td=""> 24 } 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 25</rdfs:subclassof></owl:class>				-		L. L				
24 <rdfs:subclassof 24="" 25="" <="" br="" rdf:reso="" }=""></rdfs:subclassof>										
25 25 25 26 25 25 25 25 25 25 25 25 25 25 25 25 25				reference [11] Servicing		5				
			24	}		24 }				
1 st stage: OWL description of the ontology 2 nd stage: KM3 description of DSL 3 rd stage: Definition of ATL transformation rules	25		25	× .	┫┣	25 *				
2 stage. This description of DSE stage. Deminion of ATE transformation rates	1 st sto	ae : OWL description of the ontology	2	nd stage: KM3 description of DSI		3 rd stage: Definition of ATL transformation rules				
	1 510	get of the description of the ontology				Juge. Definition of ATE transformation rates				

3. Use case: co-evolution of the ontology and DSL in the railway allocation domain

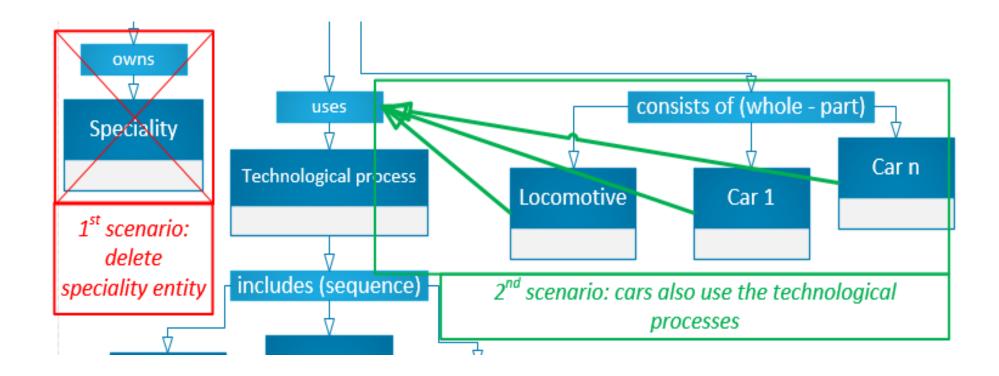
A part of the railway transportation ontology



A part of DSL objects scheme



Sceneries of evolution



Results

In case, when the ontology and DSL has no whole coherence:

we need to complete the system of universal transformation rules with a set of specific rules, which aims to resolve differences between the ontology and DSL; this system of rules extends every time, when the ontology changes.

In order to avoid the above problems, we have to build a DSL model as a complete reflection of the ontology when the latter is designed for the first time, using the universal system of graph-transformation rules.

4. Conclusion

- DSL is an effective tool for working with the subject area; DSL contains inside the model of the target domain; the ontology can be used as a model for DSL;
- •In order to solve the problem of coherence between the subject domain (=ontology) and DSL the graph-transformation approach can be used;
- •Proposed approach allows to simplify the needed procedures for achieving the correspondence between the ontology and DSL by creation of system of universal graph=transformation rules;
- •In comparison to existing solutions (ex.: Cleenewerck et al. "Component-Based DSL Development"), created approach is not based on some specific DSL, but can be applied for any of them and allows to edit existing ontologies and DSL without recreating them.

Further steps:

- •Realization of the opportunity to change the functional level of DSL model in accordance with the ontology;
- •Implementation of additional ontological concepts as Roles (according to the results of Laird and Barrett from "Towards Dynamic Evolution of Domain Specific Languages").

Thank you for attention!

Questions?