A Projection-Based Approach for Development of Domain-Specific Languages

Boris Ulitin(✉), Eduard Babkin, and Tatiana Babkina

National Research University Higher School of Economics,
Nizhny Novgorod, Russia
{bulitin, eababkin, tbabkina}@hse.ru

Abstract. The article is related to the problem of sustainable flexibility of a domain-specific language (DSL) in response to evolution of the application domain and users' capabilities. We offer a solution of that problem based on a particular model-driven approach. We propose to create a DSL structure from the domain-semantic-model (DSM) through the so-called semantic projection mechanism. The semantic projection is an operation, which is conducted over DSM. The result of the projection is a fragment of DSM, which describes the semantic model of a particular DSL dialect. We suggest to apply a group of model-to-model (M2M) transformations for practical implementation of semantic projections and producing corresponding DSL artefacts. We demonstrate the application of the proposed approach by the example in railway allocation domain.

Keywords: Domain-specific language · Domain-semantic model
Projection · Model-driven development · Model-to-model transformations
Evolution · Railway transportation

1 Introduction

Currently, domain-specific languages (DSL) become more and more widespread. Such popularity can be explained by the fact, that DSL is a fairly simple and convenient way of organizing work in a certain target domain. DSLs contain only the required set of terms of the domain, representing some kind of its conceptualization, because every DSL uses as its basis some model of the application domain, which is named as Domain Semantic Model (DSM) [1]. Using the concept of DSM allows the users to achieve addressing resilience of businesses information systems because of integration of DSM and general model-oriented approach in development of DSL and its consequent evolution process.

The DSL life cycle consists of five development phases: decision, analysis, design, implementation and deployment [2]. In our work we concentrate on the first three phases, leaving behind a framework for considering the specific application of the created DSLs. The first two phases are connected with the target domain, for which DSL is developing, and result in the definition of some domain model and gathering domain knowledge. Only after that the DSL design is initiated. Traditionally, the DSL design is organized in two orthogonal ways: to develop DSL from scratch or to base it on the existing one. However, both cases do not take into account any formal results,

J. Zdravkovic et al. (Eds.): BIR 2018, LNBIP 330, pp. 219–234, 2018.
https://doi.org/10.1007/978-3-319-99951-7_15

which can be obtained during the domain analysis phase. Although DSM contains all terms and relationships between them, needed to be included into DSL created. As a result, the designer actually repeats the provided analysis phase while designing DSL and, actually, creates one more DSM for DSL created instead of transferring once defined DSM into DSL structure.

This is especially useful, considering that the DSL structure contains semantic and syntactic levels. The syntactic level is responsible for the opportunity to define some context, while the semantic level reflects this context on the concepts of the target domain. As a result, the semantic level absolutely depends on the model of the target domain, used during DSL creation. Taking into account, that DSL is based on a specific DSM, we can argue that the semantic model of the DSL can be obtained by projecting the corresponding DSM.

At the same time, the syntactic level of DSL is a result of combination of the general domain concepts with the problem-specific tasks of the users. This means, the syntactic part of DSL is a result of adaptation of the DSL semantic model to information and operational needs, experience and skills of every specific user. As a result, in parallel to the development of the skills and knowledge of the user, the set of DSL terms, that he/she operates with, can also change. The syntactic model of DSL is also the result of projection of the DSL semantic model on the user's needs and can be based on the principles of projecting DSM into the DSL semantic model. This co-directional development of both levels of DSL leads to the necessity of sustainable maintaining a growing arbitrary tree of syntactic and semantic dialects of the original DSL. Some dialects may be based on the one semantic DSL model, while other dialects use different DSL semantic models on the one DSM.

Certain attempts to facilitate such sustainable maintenance of DSL dialects were done by several researchers. For example, Cleenewerck in [3] tries to identify the tree of DSL dialects using graph-transformations for transition between different DSL dialects. However, every DSL dialect in this approach is a combination of the syntactic and semantic parts. Such a mechanism, when the environment stores not just different components of DSL, but the complete structure of different DSL dialects and its handlers, is called versification [2]. Obviously, this approach does not allow us to organize a flexible and rapid transition from one version of DSL to another, because it causes the semantic transition between different DSLs.

A similar model-oriented approach is described by Haav et al. in [12], but they concentrate only on the opportunity to transform the DSL semantic level into the syntactic level, without reference to DSM. Furthermore, the structure of DSL in this case is considered stable over the time. In order to support co-directional development of both DSL levels, we propose to use the notion of semantic projections. Semantic projection is a procedure which produces a subset of the original DSM. This subset fully describes the DSL semantic model of a particular DSL dialect. In our case the group of semantic projections formally specifies transitions between all known DSL models.

For practical implementation of our approach we suggest to consider the object-oriented paradigm, because in that case DSL becomes a combination of objects and operations on these objects. Consequently, the semantic part of DSL can be represented as an object-oriented model, which is a result of DSM semantic projection.

Furthermore, the syntactic part can also be formalized as an object-oriented model, which components are projections of the semantic model. From such point of view, we can organize the DSL development as a unified process of sequential projection of some models into others. In order to provide such projections, the mechanism of model-to-model (M2M) transformations [10] can be used. In order to achieve these goals, we adopt the graph-oriented transformations, described in our previous article [15], for projection of DSM into DSL semantic model and expand it for organizing projections of the DSL semantic model to the syntactic level. From the point of implementation, we propose to use such model-driven instruments as Eclipse GMF platform for models' definition and QVT language for realization of M2M transformations among them.

Our projection-based principle of sequential creation of DSL from DSM to a specific semantic model, differs substantially from existing approaches [2–4], because the proposed projection-based procedure of DSL development allows not only to unify all stages of DSL creation, but also modify different parts of DSL independently. Also, our approach eliminates the need to recreate the DSL in the case of making any changes to it without the opportunity to modify different DSL parts separately.

As a result, the proposed approach allows not only to construct one specific DSL, but also to modify it in accordance with changes in DSM. Thus, it is possible to determine the different syntactic dialects of DSL over one DSL semantic model as well as to define several semantic models of the DSL over one DSM model. All these models (DSM, the DSL semantic and syntactic models) are dynamic and can be updated consistently over the time in accordance with the changes that are taking place in the domain.

The proposed approach can be the most effective in the domains with a high need for adaptation to end users. For example, in a case of definition of DSLs for elderly people, which are limited not only because of the domain's restrictions, but and physical and psychological limitations of the actual users. Another case of application are rapidly evolving domains, where new terms are appeared and have to be included in DSL developed. We show the effectiveness of the proposed approach by the example of DSL development for the railway allocation domain: starting from the DSM development and the further definition of two different consistent DSL dialects, indicating cyber-resilience of the established system.

The article describes our proposed approach as follows. In Sect. 2 we give some facts from the theory of design of domain-specific languages, define the DSL structure in model-oriented manner and analyze various tools and notations, appropriate for DSL model-oriented development. Section 3 describes a high-level design and technologies of the proposed approach to DSL development. Section 4 demonstrates an example of application of the proposed approach to the railway allocation domain. We conclude the article with analysis of the results and specification of the future researches.

2 Background

2.1 Definition of Domain Semantic Model (DSM)

DSM offers a flexible and agile representation of domain knowledge. DSM can be constituted by either just small pieces of a domain knowledge (e.g. small taxonomies equipped with few rules) or rich and complex ontologies [13] (obtained, for example, by translating existing ontologies). That gives respectively weak or rich and detailed representation of a domain [14]. More formally DSM is a seven-tuple of the form:

$$DSM = (\mathcal{H}_C, \mathcal{H}_R, O, R, A, M, D)$$

where

- $\mathcal{H}_C$ and $\mathcal{H}_R$ are sets of classes and relations schemas. Each schema is constituted by a set of attributes, the type of each attribute is a class. In both $\mathcal{H}_C$ and $\mathcal{H}_R$ are defined partial orders allowing the representation of concepts and relation taxonomies;
- O and R are sets of class and relation instances also called objects and tuples;
- A is a set of axioms represented by special rules expressing constraints about the represented knowledge;
- M is a set of reasoning modules that are logic programs constituted by a set of (disjunctive) rules that allows to reason about the represented and stored knowledge, so new knowledge not explicitly declared can be inferred;
- D is a set of descriptors (i.e. production rules in a two-dimensional object-oriented attribute grammar) enabling the recognition of class (concept) instances contained in O, so their annotation, extraction and storing is possible.

In our research, we focus only on the sets O and R. Consideration of other parts of DSM is beyond the scope of our study, since it determines more meta-characteristics of DSM itself, rather than the objects and connections between them, which are the most interesting for the further development of the DSL semantic model.

It is also important to note, that DSM usually has a dynamic structure, demonstrates a tendency to changes over the time (evolution, in other words). In accordance with the evolution of the domain, the evolution of its DSM also occurs [2]. As a result, any DSL, based on the corresponding DSM, should be adopted according to the changes. Consequently, the structure of the DSL metamodel should be as close as possible to the structure of DSM in order to guarantee the coherence between DSL and the target domain. So, that is reasonable to select a common meta-meta model which will be used both for definition of a DSL metamodel and DSM. We believe that a widely accepted object-oriented meta-meta model can be suitable for our purposes. The following manifestation of DSL as a special kind of the object-oriented model proves that believe.

2.2 Manifestation of DSL in Terms of Object-Oriented Models

A domain-specific language (DSL) is a computer language specialized to a particular application domain. This is in contrast to a general-purpose language, which is broadly applicable across domains, and lacks specialized features for a particular domain [1]. In [5] two parts of the DSL are identified: (1) a syntactic part, which defines the constructions of DSL; and (2) a semantic part, which manifests itself in the semantic model. The first part allows defining the context for working with the second one, which defines meaning of DSL commands in terms of the target domain.

From the formal point of view, a semantic part of DSL is derived from the set of objects of the target domain and operations on them. The structure of every model is a combination (E, R) of some entities (each entity is a set of its attributes $e_i = \{attr_{i_1}, attr_{i_2}, \ldots, attr_{i_M}\}, M \in \mathbb{N},\ i = 1, N$) and relations between them. Therefore, we can formalize the semantic DSL level in a model-oriented manner as a combination (O, R) of some objects of the target domain and relations between them, where each object is a set of its attributes and operations $o_i = (Attr_i, Opp_i) = (\{attr_{i_1}, attr_{i_2}, \ldots, attr_{i_M}\}, \{opp_{i_1}, opp_{i_2}, \ldots, opp_{i_K}\}), M, K \in \mathbb{N},\ i = 1, N$).

In these terms, the syntactic part of DSL can be represented as a subset of the semantic level, needed for representation of a certain problem situation. The very one difference is, that the syntactic part may not absolutely reflect the semantic constructions but identify its own definitions (pseudonyms) for the semantic constructions, according to the user's needs.

In addition, a syntactic part of DSL can also be separated into two levels: the level of objects and the level of functions. The object-level is equivalent to the set of objects of the semantic level. The functional level contains operations, which allow to specify the operational context for the objects.

As follows, the structure of the syntactic level can be formalized as a triple $\left(O_{syntax}, R_{syntax}, Alias_{syntax}\right)$, where $O_{syntax} \subseteq O$ and $R_{syntax} \subseteq R$ are the subsets of objects and relations between them of the semantic DSL level respectively, and $Alias_{syntax}$ is a set of pseudonyms for objects' components (attributes and operations).

Under these circumstances, we can tell about the complete model-oriented representation of the DSL structure. The structure allows us not only to describe both levels of DSL in structured and unified manner but optimize the process of DSL development and evolution by introducing several syntactic DSL dialects on one fixed semantic level. Furthermore, the versification of DSL can be provided in a similar way on the semantic level as well as on the syntactic level, without need to re-create the whole DSL structure every time, when the changes are required.

3 Proposed Approach

3.1 A Semantic Hierarchy of Model-Oriented DSL Development

Combining the object-oriented model of the DSL structure with the formal definition of DSM on the basis of a single meta-meta model, we can specialize a well-known semantic hierarchy of meta-models for our approach to model-oriented development and evolution of DSL (Fig. 1).

In our case this hierarchy is separated into four layers, according to the stages of the DSL development. Each lower level is based on the model artefacts of the upper level.

A single M3 meta-meta-model determines common grounds for all meta- and models of the lower levels. This meta-level defines also notations in which concrete models will be defined and what rules for their transformations will be used.

The structure of the semantic hierarchy determines the corresponding process of DSL creation. It starts with the definition of DSM, containing all important entities of the target domain and relationships between them. The process of DSM creation is beyond the scope of our current research, we propose its consistency for DSL development. For more details on DSM definition and checking its correctness, see [13].

When DSM is created, we can build the DSL semantic model by the operation of semantic projection. Any semantic projection performs a certain M2M transformation of DSM to some its fragment. Thus, semantic projection fully determines the semantic model of a particular dialect of DSL. In this case the semantic model becomes an object-temporal structure, because it should be adopted according changes in DSM over the time, thereby defining a new object filling of the DSM.

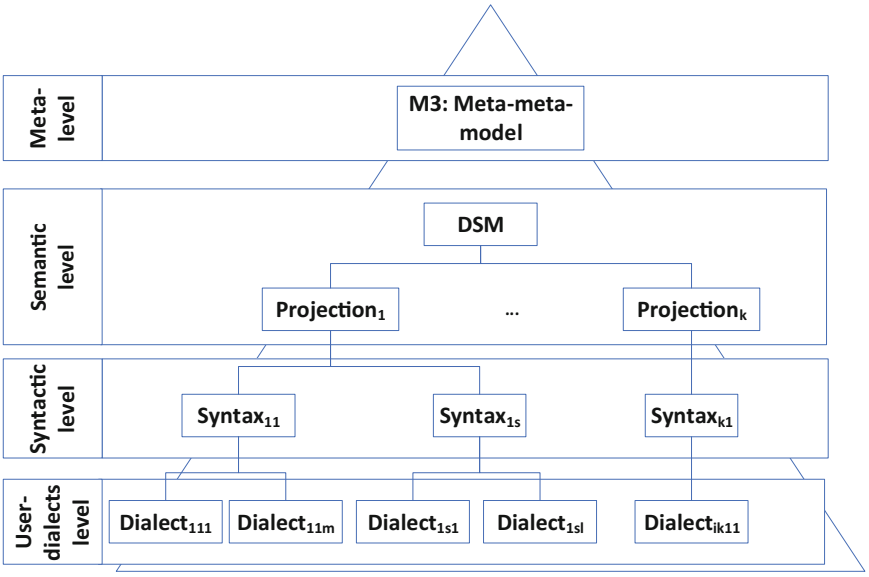


Fig. 1. The semantic hierarchy of projection-based DSL development

After the semantic projection was performed, the syntactic level of DSL can be developed by a M2M transformation of the result of the corresponding projection. What is important, these DSL syntactic models are independent of each other and are determined by end-users in accordance with the adaptation of the semantic projection to their own tasks.

Finally, created syntaxes are used by the end-users of DSL, who determine the set of DSL dialects within the single specific syntactic model.

Figure 2 shows differences between traditional approaches and our proposals. Traditional approaches start with the manual definition of the DSL concrete syntax which is followed by the translation of the syntax in terms of grammars. Consequently, every change in the target domain leads to the need to redefine the DSL concrete syntax and re-create the corresponding grammar. A similar process repeats in a case, when changes in DSL are caused by the end-users. As a result, outcomes of traditional approaches contain inconsistent dialects of DSL, which cannot be mapped among themselves due to differences in all levels of the DSL structure.

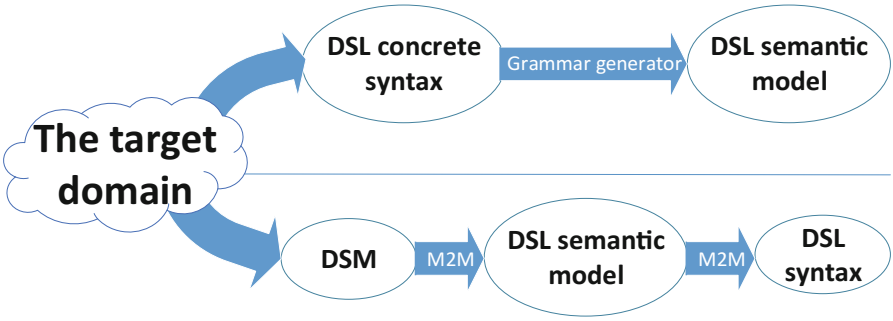


Fig. 2. The scheme of differences between traditional (top) and projection-based (bottom) DSL development approaches

In comparison to traditional approaches, the proposed projection-based scheme (Fig. 2top) of DSL development is organized in the strong correspondence of the target domain. Such correspondence is provided by the consequent projections among different models in a semi-automatic way through M2M transformations: from DSM into a DSL semantic model and then into a syntactic model of the specific DSL dialect. As a result, we can define several DSL syntactic dialects over one specific DSL semantic model, which will be consistent and can be transformed between themselves without the redefinition of the DSL semantic models.

3.2 Model-Driven Tools for DSL Development

After the definition of the projection-based approach for DSL development, we need to analyze available tools, which allow us to practically develop DSL according to the projection-based principles.

This seems to be important, because each tool supports a limited set of notations for the model-oriented DSL representation. That, in turn, any tool imposes restrictions on the feasibility of practical implementation of M2M transformations.

Our analysis shows that there are several tools, which allow us to define or transform DSL in a model-oriented manner (see Table 1).

Table 1. Available tools for DSL model-oriented development

Tool	Supported notations	Transformation support	Supported transformation formats
MetaEdit+	GOPRR	None	None
Eclipse GMP + XText	MetaGMP, ECORE	Complete	QVT, ATL
MPS	Own textual notation	None	None
MetaLanguage	Own textual notation	Partial (whole DSL structure versions, without transformation on separate levels)	ATL

As we can see, only two of four tools provide an opportunity not only to create DSL, but also to make changes in it: Eclipse GMP [7] and MetaLanguage (http://pespmc1.vub.ac.be/METALARE.html). Both tools support organization of transformations of the once created DSL structure, using the ATL language. However, MetaLanguage is based on the following inappropriate principle: any DSL transformation is equivalent to the DSL re-creation, that means the need to redefine the whole DSL structure instead of definition of the transformation of its separated components of the semantic and/or syntactic levels. Such a limited toolkit paired with the need to use MetaLanguage specific notation to determine the DSL model, makes it impossible to conduct flexible transformations of the DSL in this environment.

```
modeltype modelTypeName;
transformation ruleName (in i : originalModel,
                         out o : targetModel);
mapping originalModelType::originalModelElement() : targetMod-
elElement
{
  actions
}
```

Fig. 3. The scheme for definition the transformation rules in QVT

The only possible option for carrying out flexible DSL modifications is the Eclipse GMP. This platform provides unified and standardized notations to define the model-oriented structure of DSL, and also allows users to organize the transformations of various DSL components without redefining the previously created structure. In order to define the formal specification of transformations a set of specialized graph-transformation languages can be used such as ATL Transformation Language (http://www.eclipse.org/atl/), QVT (Query/View/Transformation) [6], etc.

In our research, we propose to use QVT, because this language allows us to describe the transformation rules from any original model into any target model, conducting a transformation at the level of meta-models. In QVT the rule definition includes five components: the name of the model type and the rule, the type of the element of the original model, the type of the equivalent element of the target model and the block for definition of the additional actions with transforming and corresponding components of the original and the target model (Fig. 3). In addition, QVT supports not only a textual form for rule definition, but also a model-driven rule architecture, which permits automatic organization of the transformation according to the definition of the original and target models.

In our research we have studied appropriateness of the Eclipse GMP platform for design and further modification of DSL in the particular domain of resources allocation in the railways. However, this does not limit the possibility of using this tool to create DSLs in other domains. The only thing that will be required for this is to determine the structure of DSL in terms of the ECORE notation and the rules for its further modification.

ECORE is an EMF's UML-based object-oriented modelling language. Inheriting from the object-oriented UML notation, the ECORE notation also represents a model in the form of a collection of entities and relationships between them. Main elements of the ECORE notation include: *EClass* in order to define a specific class/entity, *EAttribute* and *EDataType* in order to specify the entity's attributes and their types, *EReference* in order to define relationships between different entities.

ECORE notation provides the same universal expressive abilities to represent both levels of DSL, that greatly simplifies the process of its development and allows to organize the creation of the DSL syntactic level as a transformation of the DSL semantic level, defined in the ECORE notation.

In addition, the ECORE model is not strongly connected with one fixed meta-model, thereby allowing the user to enter own components into the language and reuse previously created DSL models and the possibility of their flexible and coherent integration, that is the main goal of our research.

4 Application of the Proposed Approach to the Railway Domain

4.1 DSM of the Railway Allocation Domain

The railway allocation problem identifies the case, where the effectiveness and performance of the user strongly depends on the convenience and adaptability of DSL. The more flexible and native DSL syntax leads to the faster process of real time resource allocation. DSL allows, in particular, to allocate trains among railways, specify what services have to be provided to trains, what brigades perform these services, etc.

However, the concrete syntax can be defined only on the general semantic model of DSL. As a result, in order to increase the flexibility and create opportunities for customization of the DSL syntax for each specific user, we need to define a semantic model of the language as such accurate as possible. Consequently, it's vital to identify all the types of resources and operations on them in this domain.

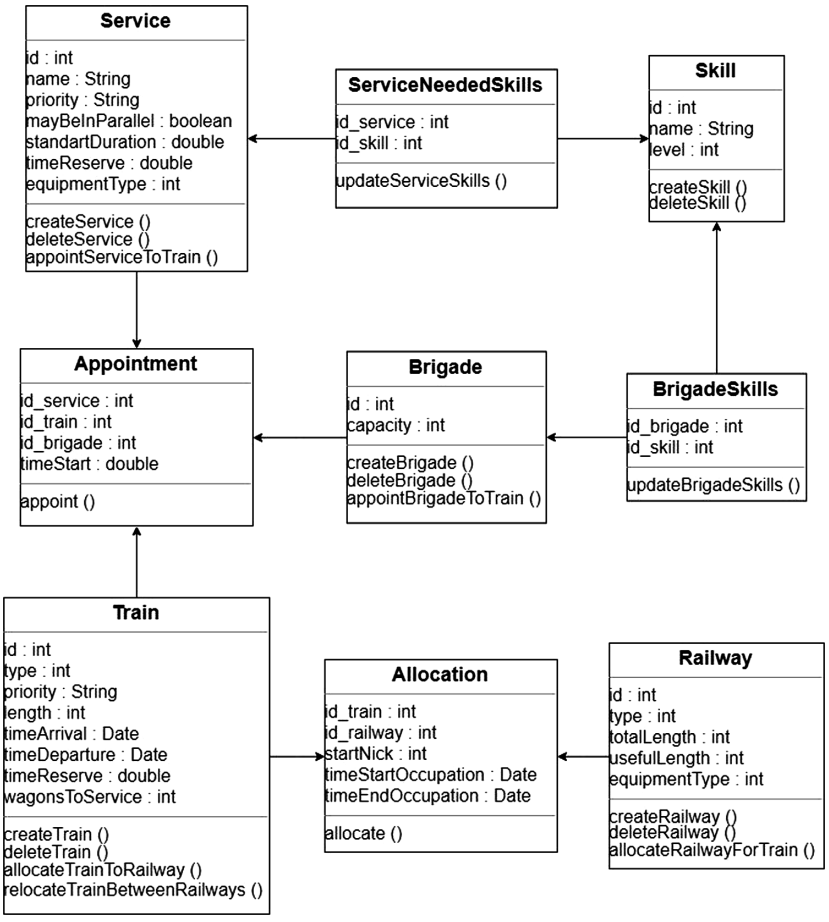


Fig. 4. The UML representation of DSM for the railway allocation domain

Developed DSM of the railway allocation domain is more complete in comparison with that considered in our previous work [15], since it contains the specification of the requirements (Skills) both for the Services and for the Brigades providing them. There are three general resources with own specific attributes for any railway station (Fig. 4): ***Trains***, ***Railways*** and ***Servicing brigades***. The more fundamental description and analysis of these resources is represented in our previous work [15], here we are interested only in the object-oriented structure of them.

In order to represent the relations between classes, two additional entities are defined: ***Allocation*** (to store the relations between the Train and the Railway) and ***Appointment*** (to store the relations between the Trains, the Service, needed for this Train, and the Brigade, which provide needed Service for the Train).

The behavior of all classes is defined using the methods of the corresponding classes. For example, the user can create a new Train, Brigade, Railway; appoint the Brigade for the created Train and allocate it on the created Railway. Other sceneries of the behavior can be realized through the corresponding methods.

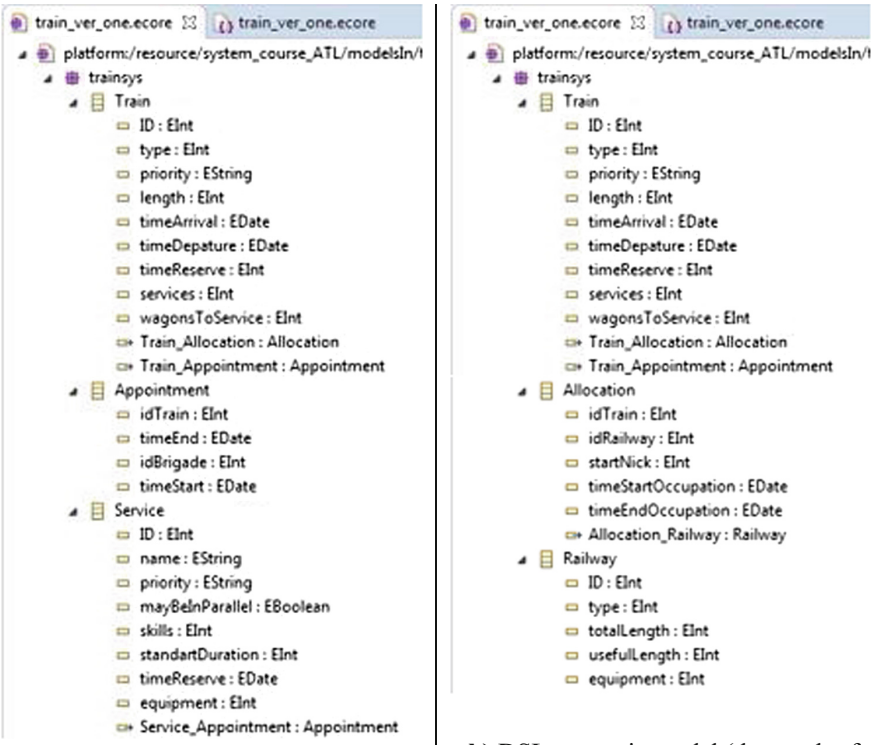
4.2 The DSL Semantic Level Definition

After the definition of DSM of the target domain, we can identify the semantic level of DSL. For this we need to reformulate the above created scheme in the UML notation on terms of the ECORE notation. In order to simplify this process, we use the graphical tool of Eclipse GMP, which translates the UML-description of DSM, into the ECORE model. As a result, the following ECORE representation of DSM is generated (the fragment is represented in Fig. 5a).

After this stage, our approach allows the users maintaining of DSL dialects based on the projections of DSM into DSL semantic models. In our particular case we need to create two DSLs dialects in response to the users' demands: the first one has to provide the opportunity to allocate Trains among Railways, while the second one - to appoint Brigades for Service delivery among Trains. For producing both DSL dialects, we should project only required set of DSM elements in order to achieve strong correspondence between the DSL structure and its purpose: Train, Railway and Allocation for the first DSL, and all DSM entities for the second DSL.

In order to implement corresponding semantic projections and realize needed DSL semantic models, the system of QVT transformation rules have to be defined. In more details the set of QVT and ATL rules, needed for different types of model transformations, is described in [15]. For the first DSL the rules (Fig. 6) remove all unnecessary entities from DSM, leaving only those necessary for the first DSL. As an output we have an ECORE-representation of the DSL semantic model (Fig. 5b). As we can see, this model contains a subset of entities from DSM necessary for specifying the corresponding DSL dialect: Train, Railway and Allocation.

In order to realize the projection for the second DSL semantic model, we have to use only the final mapping rule from the previously created (Fig. 6), because the DSL semantic model completely coincides with the DSM in this case (Fig. 5a).



a) DSM

b) DSL semantic model (the result of projection)

Fig. 5. The fragments of the key modeling artefacts (ECORE-representation)

```
modeltype ECORE;
transformation SemanticToSintactic (
   in i : "semanticDSLlvl.ecore", out o : "syntacticDSLlvl.ecore" );
rule killRedundEntities {
   from
      s : Ecore!ENamedElement (
        s.name <> 'Train' and s.name <> 'Railway' and
        s.name <> 'Allocation')
   to drop }
rule killRedundMethods {
   from
      s : Ecore!ENamedElement (
        s.rootObject().name <> 'Train' and
        s.rootObject()name <> 'Railway' and
        s.rootObject()name <> 'Allocation')
   to drop }
mapping EPackage :: copyPackage() : EPackage {
   name :=self.name; eClassifiers += self.eClassifiers[EClass]; }
```

Fig. 6. QVT M2M transformation rules

4.3 The DSL Syntactic Level Definition

After we have got the ready semantic model of DSL, we can define its syntactic level. In fact, this means that we need to describe a number of aliases for accessing specific functions on the semantic level of DSL. In our case end-user DSLs have a graphical form. Therefore, the aliases have a form of graphical pictograms. Each pictogram represents the corresponding entity of the DSL semantic model. The line connectors between the pair of pictograms represent the semantic connection between corresponding domain entities.

In order to achieve our goal, the Eclipse XTend plugin (http://www.eclipse.org/xtend/) can be used. This plugin allows us to identify the pictograms above ECORE-model entities and provides the opportunity to automatically create the form for the definition of the corresponding entity attributes. Such automated form creation is available only for the attributes, which type is internal for the ECORE-metamodel. Custom types are not supported and require manual implementation of form fields. In our case all attributes are standard and are implemented in ECORE. That means that automatic generation of the DSL syntax is available for both DSL dialects. The examples of the created interface for both DSLs are represented in Figs. 7 and 8 respectively.

Created interfaces are absolutely coherent with the corresponding DSL semantic models and allow to organize the corresponding DSL dialects scenarios: allocate Trains among Railways in the first dialect and appoint Brigades for providing Services for Trains in the second dialect.

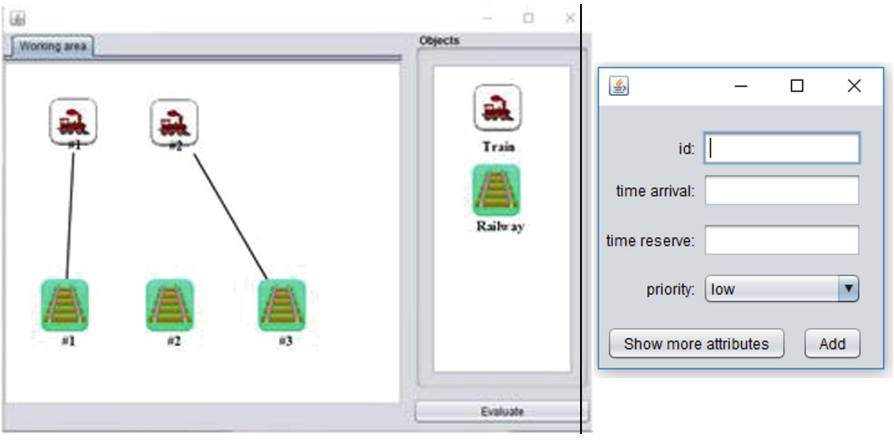


Fig. 7. The interface and forms prototypes for the first DSL dialect

What is more important, both interfaces are based on the same more general DSM. That results in the opportunity to update and extend them without the need to re-create the whole DSL scheme. Only a modification of the QVT rule and the interface concretization are needed for this operation and can be provided by the end-users.

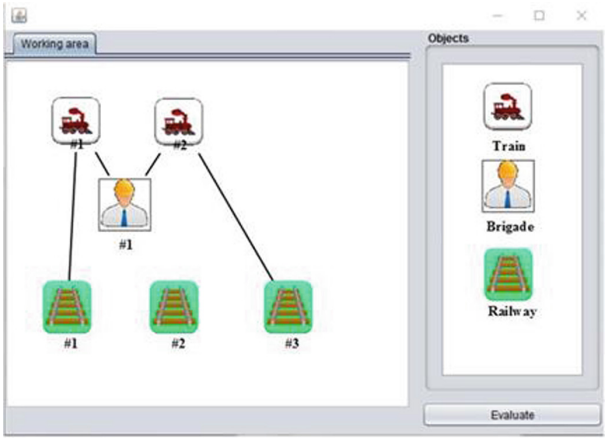


Fig. 8. The interface prototype for the second DSL dialect

Comparing the proposed approach of DSL model-oriented development with other approaches in terms of time-consuming, we can state that our approach significantly accelerates the process of DSL development. For example, in our previous work [15] a full cycle of DSL development took about two months, while using the proposed approach this time was reduced to one week. In fact, during this time frame, two different DSLs were developed against one in the previous case. The main part of time was spent on building the original full DSM, while all subsequent operations took several hours, because they were organized in (semi)automatic manner. Such a significant reduction in the time to develop DSL shows the effectiveness of the proposed approach and the possibility of its use in the industrial development of customizable DSLs.

5 Conclusion

In our research we explored a model-oriented and projection-based approach for DSL development. Proposed approach is based on the idea, that every DSL contains two parts: semantic and the syntactic. Both can be represented as a set of interconnected objects, characterized by specific attributes. Such object-oriented description of DSL levels results in the opportunity to define the DSL as a model-oriented structure, where a certain entity is assigned to each element.

What is the most important, the DSL structure is created automatically from the specific DSM using the so-called semantic projection mechanism. The semantic projection is an operation, which is conducted over the DSM and the result of which is also a semantic model that is obtained by transforming the original DSM. The result of projection describes the DSL semantic model.

In contrast to the superset modelling language [8], DSM in this case is not only the formal representation for the domain's terms and relationships between them, but also reflects users' knowledge and semantical scheme of the domain, allowing to take it into account during the DSL creation.

In comparison with the existing approaches to DSL development (like [3, 9, 10]), which use a traditional cycle of DSL development, starting from the definition of the DSL concrete syntax, our approach starts with the generation of DSM, which is a dynamic, time-varying structure. Under these circumstances, the DSL semantic model can be obtained as a projection of such DSM through M2M transformations. Furthermore, the DSL syntactic model is also the result of the projection of the DSL semantic model onto users' requirements and needs.

As a result, the proposed DSL development process is conducted in full accordance with the conceptual scheme of the target domain, thereby ensuring the participation of end-users in the process of its creation. In addition, the projection-based principle of the DSL development allows the users to achieve the resilience of the DSL created both with the target domain (represented by DSM) and users' requirements. Created DSL can be transformed on the semantic and syntactic levels separately, using M2M transformations for projections realizations. At the same time, the consistency of the created DSL dialects is preserved.

Among advantages of the approach proposed its reusability and end-user orientation should be mentioned. The approach can be transferred to any domain for which the DSM is defined. The model-oriented structure of DSL is also an understandable and convenient for end-user. This format of DSL representation results in the opportunity to make changes to DSL without special programming skills. The defining a specific syntax for different versions of DSL is maximally automated and is carried out at the level of models, rather than in the program code. Such automation significantly reduces the time required to complete the DSL development cycle, that was demonstrated by the examples.

Planning further research, we consider to organize the whole model-driven DSL life-cycle with opportunity to define the more complex behavior for the user through the combination of different syntactic DSL elements. One more possible improvement can be the implementation of versification mechanism not only at the level of syntax, but also at the level of semantics. For achieving this goal an approach, described by Bell, can be used. In [11] Bell tries to implement the coherent changes in several DSL meta-models, using the automated transformations, that can allow to organize the consistent changes in the semantic and syntactic DSL levels models within our approach.

References

1. Fowler, M.: Domain Specific Languages. Addison Wesley, Boston (2010)
2. Laird, P., Barrett, S.: Towards dynamic evolution of domain specific languages. In: van den Brand, M., Gašević, D., Gray, J. (eds.) SLE 2009. LNCS, vol. 5969, pp. 144–153. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12107-4_11

3. Cleenewerck, T.: Component-based DSL development. In: Pfenning, F., Smaragdakis, Y. (eds.) GPCE 2003. LNCS, vol. 2830, pp. 245–264. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-39815-8_15
4. Challenger, M., Demirkol, S., Getir, S., Mernik, M., Kardas, G., Kosar, T.: On the use of a domain-specific modeling language in the development of multiagent systems. In: Engineering Applications of Artificial Intelligence, pp. 111–141 (2014)
5. Parr, T.: Language Implementation Patterns: Create Your Own Domain-Specific and General Programming Languages. Pragmatic Bookshelf (2012)
6. QVT (Query/View/Transformation). https://projects.eclipse.org/projects/modeling.mmt.qvt-oml
7. Elipse Graphical Modeling Project (GMP). http://www.eclipse.org/modeling/gmp/
8. Lucassen, G., Robeer, M., Dalpiaz, F., Werf, G.M., Brinkkemper, S.: Extracting conceptual models from user stories with Visual Narrator. Requir. Eng. **22**(3), 339–358 (2017)
9. Sprinkle, J.: A domain-specific visual language for domain model evolution. J. Vis. Lang. Comput. **15**, 291–307 (2004)
10. Kosar, T., Martınez Lopez, P., Barrientos, P., Mernik, M.: A preliminary study on various implementation approaches of domain-specific language. In: Information and Software Technology, pp. 390–405. Elsevier (2008)
11. Bell, P.: Automated transformation of statements within evolving domain specific languages. In: Computer Science and Information System Reports, pp. 172–177 (2007)
12. Haav, H.-M., Ojamaa, A., Grigorenko, P., Kotkas, V.: Ontology-based integration of software artefacts for DSL development. In: Ciuciu, I., et al. (eds.) OTM 2015. LNCS, vol. 9416, pp. 309–318. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26138-6_34
13. Guizzardi, G.: Ontological foundations for structural conceptual models. Telematica Instituut Fundamental Research Series, The Netherlands, no. 15 (2005). ISBN 90-75176-81-3
14. Ruffolo, M., Sidhu, I., Guadagno, L.: Semantic enterprise technologies. In: Proceedings of the First International Conference on Industrial Results of Semantic Technologies, vol. 293, pp. 70–84 (2007)
15. Ulitin, B., Babkin, E.: Ontology and DSL co-evolution using graph transformations methods. In: Johansson, B., Møller, C., Chaudhuri, A., Sudzina, F. (eds.) BIR 2017. LNBIP, vol. 295, pp. 233–247. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-64930-6_17