

**HP-Graph as a Basis of a DSM Platform Visual Model Editor**

N.M. Suvorov, ORCID: 0000-0003-2871-9757 <SuvorovNM@gmail.com>
L.N. Lyadova, ORCID: 0000-0001-5643-747X <LNLyadova@gmail.com>
National Research University Higher School of Economics, 38 Studecheskaya Ulitsa, Perm, 614070, Russia

**Abstract.** The language-oriented approach is becoming more and more popular in the development of information systems, but the existing DSM platforms that implement this paradigm have significant limitations, including insufficient expressive capabilities of the models used to implement visual model editors for complex subject areas and limited abilities to transform visual models. Visual languages are usually based on graph models, but the types of graphs used have certain limitations, such as insufficient expressiveness, the complexity of representing large-dimensional models and operation executions. For creating a tool that does not have the described constraints, development of a new formal model is needed. HP-graphs can become a solution for this problem. It is not only possible to create new visual languages for diverse domains based on them, but also to develop efficient algorithms to perform different operations on models constructed using these languages. The HP-graph definition is given and the justification of the expressive power of the proposed model is presented, the main operations for HP-graphs are described. The chosen graph formalism combines the capabilities of different types of graphs to represent visual models and allows creating a flexible model editor for the DSM platform, to implement effective algorithms of performing operations, in particular, model transformations.

**Keywords:** Domain-specific language; DSM platform; visual model; graph model; HP-graph; algorithms on graphs.

**For citation:** Suvorov N.M., Lyadova L.N. HP-Graph as a Basis of a DSM Platform Visual Model Editor. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 2, 2020. pp. 149-160. DOI: 10.15514/ISPRAS–2020–32(2)–12

**1. Introduction**

The study of any objects and processes, as well as their design, can barely be done without modeling, that is why software tools that allow non-IT specialists to build various models and formalize descriptions of objects and processes, or use modeling as a method of analysis for the study of objects are becoming more popular. Among the subject areas where modeling is particularly important, the development of information systems stands out. Currently the main approach to creating large information systems is a model-oriented approach [1]. Using it, developers usually deal only with models, which helps to ensure high quality of programs and prevent errors. CASE tools [2], which automate the system development process as much as possible due to the capabilities of visual modeling, model interpretation, and code generation based on the created models, are used especially for these purposes.

However, the traditional model-oriented approach to developing systems has its drawbacks, among which are:

• universality of the languages used for system development as languages operate not in terms of the subject area, but in constructs of the means by which the system is created;
• immutability of modeling languages, which does not allow all the subtleties, pitfalls, and limitations of the subject area to be displayed and taken into account;
• complexity of modification of the created systems as making changes to the system is possible only if there are development tools, source codes and a professional IT specialist;
• the impossibility of transitioning from one modeling language to another, but, creating large systems usually involves building several models describing the system from different points of view, with different granularity, so in such cases, there is a need to harmonize the models created by different professionals at different stages of development, which requires the ability to perform a transition from one modelling language to another.

These problems are solved by a paradigm called language-oriented programming [3]. This paradigm at the initial stage of development implies the creation of a metamodel of a subject area represented by one or more languages for solving various project tasks. These languages are used to build the necessary models for implementing the system. For implementing this approach DSM-platforms [4], language tools, and Meta-CASE systems, such as MetaEdit+ [5], which facilitate the development of domain-specific languages (DSL), are usually used. These languages operate in terms of the subject area and reflect the specifics of the tasks they solve. Moreover, subject-oriented languages can also consider the qualifications of users who will use them [6].

Nevertheless, existing tools only partially solve the problems of the traditional approach to modeling. To solve all the problems described above, a language tool must meet the particular requirements [4], [7], [8]. It should

• have an ability to define modeling languages for most subject areas;
• have an ability to dynamically change the modeling language;
2. Related works

Many different tools have been created that allow people to develop modeling languages and build models based on these languages. These tools are Microsoft DSL Tools [10], Eclipse Sirius [11], MetaEdit+ [5], Microsoft Visio [12], QReal [13], etc. Detailed description and comparison of these platforms is given in [4]. All these platforms have some limitations and do not fully meet the requirements described above. Consideration of main constraints of the platforms is given below.

Microsoft DSL Tools uses templates based on UML diagrams to create a new DSL, which leads to complexity and confusion when building model hierarchies and leads to appearance of limitations and inaccuracies in the resulting modeling language [8]. Moreover, this platform is characterized by the lack of the ability to dynamically change metamodels and transform models, as well as the inability to use DSLs outside of MS Visual Studio.

Eclipse Sirius offers a solution for rapid development of a graphical tool for DSM, but certain complex tasks may require changes to the EMF and GMF code. There is also a need for interpreted expressions which will be evaluated at runtime to provide a behavior specific to domain and representations and which can only be written in Acceleo, OCL or Java [11]. Sirius allows a user to perform horizontal transformations, but the knowledge of special addins is needed.

MetaEdit+ contains only limited possibilities for transforming visual models. Models exported from the platform have their own format, which makes it difficult to use models created in this platform in other software tools.

The main drawbacks of Microsoft Visio are the inability to change the modeling language while the system is running, and the need to purchase MS Visio to use the tools developed on its basis. Also, a language metamodel can only be built using a UML class diagram, which significantly limits the platform's capabilities and complicates the process of creating languages.

The QReal platform does not have the ability to change the metamodel, the ability to transform models, and this platform is characterized by the complexity of modifying the created modeling language.

As it seems from the Table 1, there is no platform that meets all the previously given requirements. Nevertheless, it should be noted that at least some of the requirements for tools are met by each of the platforms listed.

### Table 1. The comparison of the tools

| Requirements | MS DSL Tools | Eclipse Sirius | MetaEdit+ | MS Visio | QReal |
|--------------|--------------|----------------|-----------|----------|-------|
| Ability to define modeling languages for most subject areas | + | + | + | + | + |
| Ability to dynamically change the modeling language | - | - | + | - | - |
| Ability to alienate the created language from the system | - | + | - | - | - |
| Ability to modify the visual model | + | + | + | + | + |
| Ability to perform a horizontal transformation | - | + | - | - | - |

Modeling and designing information systems tend to be done using special methodological approaches which can be divided to structural and object-oriented approaches. Despite the difference in the approaches and the division of all tools into two large groups depending on the approach underlying them (UML and "No-UML"), there are general modeling principles that the model should be aimed to implement.

The essence of the structural approach is to decompose a process into automated functions – the function of the upper level is decomposed and divided into subfunctions, refining properties of the functions at the upper levels of the hierarchy. Each subfunction, in turn, is decomposed into elements of the next level, and this happens until the obtained structure becomes trivial enough. Among the diagrams of this approach are Structural Analysis and Design Technique (SADT), a Data-Flow Diagram (DFD) and an Entity-Relation Diagram (ERD). The structural approach is used in simulation systems [15], as well as for functional and information modeling [16]. DSL can also be developed as part of this approach [17].

The essence of the object-oriented approach is an object decomposition, when the system is represented as a set of objects that exchange messages during the interaction. Moreover, the object itself in this case is an independent entity characterized by its state, behavior, and semantics [18]. Based on this approach, a set of DSLs [19],[20] is developed, but this approach is characterized by certain disadvantages, among which the complexity of building a hierarchy of models is highlighted. Using this approach does not always allow a person to properly express the concepts of the subject area, so the resulting language may have some limitations and inaccuracies. However, using it we can significantly reduce the language development time [21]. With all this in mind, the formalism underlying the visual model editor for a DSM-platform must meet the following requirements:

- to allow multi-level and multi-aspect modeling, which makes the decomposition of models from different points of view possible;
- to unify the description of models at different levels of the hierarchy, which means that the same formalism should be used to describe both models and metamodels;
- to allow development of modeling languages for a wide range of subject areas;
- to allow a user to discard constructions that are not details of the subject area, which will simplify the study of the developed language by end users;
- to perform both horizontal and vertical transformations.

Various types of graph formalisms are used for constructing and visualizing models, including oriented graphs, multigraphs [22], hypergraphs [23], hi-graphs [24], [25], meta-graphs [7], [26], P-graphs [27], [28]. Nevertheless, all these formalisms cannot meet all the mentioned requirements due to their certain limitations, therefore, development of a new graph model is needed.
3. Description of the graph model

Hypergraph with poles (HP-graph) is a graph model which meets the given requirements and can be used as a base for a visual model editor.

HP-graph is an ordered triple $G = (P, V, W)$, where $P = \{p_1,…,p_n\}$ is a set of external poles, $V = \{v_1,…,v_n\}$ is a non-empty set of vertices, $W = \{w_1,…,w_n\}$ is a set of edges [9]. Let $Pol$ be an abstract set of all poles of the graph. Thus, $2^{Pol}$ is a powerset of all poles of the graph. Then:

- Every vertex $v \in V$ is a subset of the set of all subsets of poles $(v \subset 2^{Pol})$ but $\forall v_i \in V, \forall v_j \in V (v_i \neq v_j \rightarrow v_i \cap v_j = \emptyset)$, which means that $V$ is a set of mutually disjoint subsets of $Pol$.
- A set of external poles $P$ is also a subset of the powerset of poles $(P \subset 2^{Pol})$. This set consists of input and output poles of the graph $(P = I(G) \cup O(G))$. Each vertex of the graph $v \in V$ is also represented by a set of input $(I(v))$ and output $(O(v))$ poles $P_i = \{p_{i_1},…,p_{i_n}\} \in \{v \in V | I(v) \cap O(v) = \emptyset\}$. Sets of input and output poles can also intersect.

If no poles are specified for a vertex, it is assumed that the vertex consists of a single pole, which is both input and output $(I(v) = O(v))$.

- Each edge $w \in W$ defines connections between vertices and is represented as a subset of the powerset of poles $(w \subset 2^{Pol})$. An edge cannot be represented as an empty set $(\forall w \in W : |P_i = \emptyset| = 0)$. The edge can allow a vertex to be even linked to itself. Each edge must contain at least one input pole and one output pole, so for $\forall w \in W(G)$, $\forall w \in W(G)$ the following condition must be met: $[\exists p \in I(v) \Rightarrow (I(v) \cap J(v)) \neq \emptyset]$ or $[\exists p \in O(v) \Rightarrow (I(v) \cap J(v)) \neq \emptyset]$.

An example of the hypergraph with poles is demonstrated in fig. 1.

\[
\text{Fig. 1. Example of an HP-graph}
\]

In this figure, external poles are represented by a set $P = \{p_1,…,p_n\}$, edges are represented by a set $W = \{w_1,…,w_n\}$ and vertices are represented by a set $V = \{v_1,v_2,v_3\}$. Every vertex contains a certain number of poles $p_i$ which are connected to other poles by means of hyperedges from the set $W$.

In the HP-graph, edges and vertices are represented as sets of inputs and outputs, while the actual structure of these elements is hidden. Thus, it can be assumed that these elements are represented as a «Black box».

3.1 Main operations

To describe main operations on the HP-graph, let us define $G = (P, V, W)$ as an original HP-graph, $G’ = (P’, V’, W’)$ as a resulting HP-graph, $v$ as a vertex, $p_1$ as an inner pole, $p_2$ as an outer pole and $w$ as an edge.

The following operations add elements to an HP-graph:

- $v \cup p_1$ is the addition of the inner pole to the node. The pole is added to both the vertex itself and the set of all poles of the graph: $Pol(G) = Pol(G) \cup \{p_1\}$, $v’ = v \cup \{p_1\}$.
- $G’ = w + v$ is the addition of the node to the graph. If a cardinality of $v$ is more than 0 ($|v| > 0$), a vertex is added to set of vertices $V(G)$, and all poles of this vertex are added to the set of all poles of the graph:
  \[
  \forall p \in v, Pol(G) = Pol(G) \cup \{p\},
  V(G) = V(G) \cup \{v\},
 \]
- $w + p_1$ is the addition of the outer pole to the graph. An edge is formed from the already existing poles of the graph by combining them into a single set. Let $I(w)$ be the set of input poles of an edge $(I(w))$ and $O(w)$ be the set of output poles of $w$, then the operation is represented as:
  \[
  W(G) = \{W(G) \cup \{w\}, |I(w)| = 0 \land |O(w)| = 0\}.
  \]
- $w + p_2$ is the addition of the inner pole to the edge. When a pole is removed from a vertex, all its occurrences in the edges are cut off and it is removed from the set of all poles of the graph:
  \[
  \forall v \in V(G) \{w = \{w \setminus \{p\} \mid p \in w\}\},
  Pol(G) = Pol(G) \setminus \{p\},
  v’ = v \setminus \{p\} \mid |v| > 1\}.
  \]
- $G’ = w – w$ is the removal of the edge from the graph. Performing this operation only removes an edge from the set of all edges of the graph, leaving the external poles and vertex poles unchanged:
  \[
  W(G) = W(G) \setminus \{w\}.
  \]
- $w – p_1$ is the removal of the inner pole from the node. A pole is removed only from an edge $w$, without changing the set of all poles of the graph and the set of poles of the vertex to which it belongs, but if the resulting edge $w’$ does not contain at least one input and one output pole, then the edge is removed:
  \[
  w’ = w \setminus \{p\},
  W(G) = \{W(G) \setminus \{w\}, |I(w)| = 0 \lor |O(w)| = 0\}.
  \]
- $w – p_2$ is the removal of the external pole from the edge, which is equal to the previous operation.

\[
\text{Suvorov N.M., Lyadova L.N. HP-Graph as a Basis of a DSM Platform Visual Model Editor. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 2, 2020. pp. 149-160.}
\]
3.2 Operations of Decomposition

A hypergraph with poles allows vertices and edges to be decomposed during the decryption operation. This feature makes multilevel representation possible. This possibility is achieved by correctly correlating the poles of the source and received graph which is done by implementing a mapping function.

The mapping \( f: v \rightarrow P \), which is a decoding function for a vertex \( v \), must be concordant with the sets \( I(v) \) and \( O(v) \), so that \( \forall p \in I(v): \{ [p] \} \subseteq I(G) \), \( \forall r \in O(v): [r] \subseteq O(G) \). Thus, the mapping of the pole \( pe \) \( v \) to the next level of the hierarchy is represented as \( f(p) = \pi \), where \( \pi \in P(G) \), which means that a pole \( p \) becomes the external pole \( \pi \) for a resulting graph.

Fig. 2 illustrates decomposition of the vertex \( v_1 \) by a new HP-graph.

Fig. 2. Example of vertex decomposition by a new HP-graph

An edge can be decomposed similarly but with the help of mapping \( f: w \rightarrow P \) which also must be concordant with sets of input \( I(w) \) and output \( O(w) \) poles, so that \( \forall p \in I(w): \{ [p] \} \subseteq I(G) \), \( \forall r \in O(w): [r] \subseteq O(G) \). Thus, the mapping of the pole \( pe \) \( w \) to the next level of the hierarchy is also represented as \( f(p) = \pi \), where \( \pi \in P(G) \). Example of decomposition of the edge \( w_5 \) is demonstrated in Fig. 3.

Fig. 3. Example of edge decomposition by a new HP-graph

3.3 Operations of Transformation

Many different approaches are used to transform visual models, but from the point of view of some scientists and developers [8], [29] the most promising one is the algebraic approach [30], which allows parsing graphs and checking graph models for consistency. This approach and its modifications are implemented in such tools as MetaLanguage [8], AGG [31] and VIATRA [32]. It is worth mentioning that there are also toolskits, such as ATL [14], that implement technologies from other areas of software engineering, but most of them have considerable restrictions.

To determine transformation operations, it is necessary to give a definition to a subgraph of a HP-graph. A subgraph of the HP graph \( G = (P, V, W) \) is an HP-graph \( G' = (P', V', W') \) that is a part of...
the graph $G(P, P, (v,v') \in V; (v,v') \in W)$ and fulfills the condition $Open'$. The subgraph must also meet the condition (1) to make transformation operations possible.

$$\forall v \in V \setminus V_{\text{partial}} \forall v' \in V \setminus V_{\text{partial}} \forall w \in W_{\text{partial}} \exists w' \in W_{\text{partial}}$$

(1)

The set $V_{\text{partial}}$ is a set of the incomplete vertices in the graph, where $V_{\text{partial}} = V'$. A subgraph can contain vertices called incomplete whose sets of poles can only be a part of the sets of poles of the vertices of the original graph.

To perform a transformation, it is needed to select the source and the target graph and set production rules that describe the transformation. A production rule is represented as $p = (G_1, G_2)$, where $G_1$ is a pattern-graph and $G_2$ is a replacement graph. Nevertheless, there is a restriction which is represented in (1) and must be satisfied to perform a transformation. It can be explained by the fact that it is unknown how certain hyperedges should change during the transformation while this restriction obliges to redefine all edges which are incident to poles involved in the transformation.

To display all hyperedges, all the poles that are included in them must be displayed, so it is needed to add such auxiliary (incomplete) vertexes that store only those poles that belong to the displayed hyperedges.

An algorithm for the transformation can be divided into two functions. The first one removes a subgraph isomorphic to the pattern and the second one adds a replacement graph to the original graph.

The first step can be described as follows (listing 2):

**Procedure Function DeleteGraph(HostG, GL):**

- $G' = $ Find Isomorphic Subgraph(HostG, $G_h$);
- partials = {};
- foreach $w' \in W(\text{HostG})$:
  - $W(\text{HostG}) = W(\text{HostG}) \setminus \{w\}$;
- foreach $v' \in V(\text{HostG})$:
  - if $(v', v) \in V(\text{HostG})$:
    - $V(\text{HostG}) = V(\text{HostG}) \setminus \{v\}$;
  - else:
    - partials = partials $\cup \{v\}$;
- foreach $p' \in P'(\text{HostG})$:
  - if $(\exists w \in W(\text{HostG}) \setminus \{v\})$:
    - $P'(\text{HostG}) = P'(\text{HostG}) \setminus p'$;
- return partials;

**Listing 2. Pseudocode of the algorithm that removes a subgraph isomorphic to the pattern**

The second step of the algorithm will be following (listing 3):

**Procedure AddGraph(HostG, GR, partials,):**

- if $p \in P(\text{HostG})$:
  - $P(\text{HostG}) = P(\text{HostG}) \cup \{p\}$;
- foreach $w' \in W(\text{HostG})$:
  - if $(\exists v \in V(\text{HostG})$:
    - $V(\text{HostG}) = V(\text{HostG}) \setminus \{v\}$;
- foreach $w \in W(\text{HostG})$:
  - $W(\text{HostG}) = W(\text{HostG}) \cup \{w\}$;

**Listing 3. Pseudocode of the algorithm that adds a graph to the result graph**

These algorithms can be repeated several times as the set of transformation rules may not be limited to just one rule.

4. Justification of expressive power of formalism

It is possible to justify the transcending expressive power of the HP-graph by proving that the graph formalisms generally used for building and visualizing models can be represented as an HP-graph. Previously, it was mentioned that oriented graphs, hypergraphs, hi-graphs, meta-graphs and $P$-graphs are most frequently used for such purposes. Table 2 describes formulas which represent these graph structures as an HP-graph.

| Graph model | Representation in the HP-graph $G = (P, V', W')$ |
|-------------|-----------------------------------------------|
| Oriented Graph | $V = P \setminus P'$, $W = W'$, where $\forall v \in V$: $[v'] = 1$ |
| Hypergraph | $V = P \setminus P'$, where $\forall v \in V$: $[v'] = 1$ |
| Hi-graph | $V = P \setminus P'$, where $\forall v \in V$: $[v'] = 1$ |
| Metagraph | $V = P \setminus P'$, where $\forall v \in V$: $[v'] = 1$ |
| P-graph | $V = P \setminus P'$, $W = W'$, where $\forall w \in W$: $[w] = 2$ |

From the table it can be concluded that the HP-graph has more expressive power than the previously described graph models. These graph models are special cases of the HP-graph; thus, the HP-graph is a generalization of all of these graph formalisms.

5. Conclusion

The definition of the mathematical apparatus underlying the visual model editor was given above, including a detailed description of the graph structure itself, as well as the operations that can be performed on it. For the selected graph formalism, algorithms for decoding vertices and edges, as well as algorithms for performing transformations, were described.

The HP-graph combines expressive possibilities of various types of graphs, therefore, algorithms that are designed for these types of graphs (particularly model transformation algorithms [33, 34]) can also be implemented for HP-graphs. The time complexity of model transformation algorithms can be reduced. The paper proves that HP-graph allows the creation of a flexible visual model editor based on this graph formalism for a DSM platform. Representing both vertices and links as sets of poles simplifies the object model of DSM editor and visual model editing algorithms.

It is planned to develop a program that will demonstrate the practical significance of the selected graph formalism.

References / Список литературы

[1] Koznov D.V. Basics of visual modeling. Knowledge laboratory, Internet University of information technologies; BINOM, 2008, 280 p. (in Russian) / Koznov D. V. Основы визуального моделирования. Лаборатория знаний, Интернет-университет информационных технологий; БИНОМ, 2008 г., 280 стр.

[2] Fugetta A.A. A classification of CASE technology. Computer, vol. 26, no. 12, 1993, pp. 25-38.

[3] Ward M.P. Language Oriented Programming. Software – Concepts & Tools, vol. 15, no. 4, 1994, pp. 47-161.

[4] Sukhov A.O. Comparison of visual object-oriented language development systems. Mathematics of software systems: Intercollegiate collection of scientific articles, 2012, issue 9, pp. 84-111 (in Russian) / Сухов А.О. Сравнение систем разработки визуальных предметно-ориентированных языков.
Суворов Н.М., Лядова Л.Н. HP-Graph как основа для разработки редактора визуальных моделей DSM-платформы. Труды ИСП РАН, том 32, вып. 2, 2020 г., стр. 149-160

Математика программных систем: межвузовский сборник научных статей, 2012, вып. 9, стр. 84-111.

Келли С., Лытинен К., Росси. MetaEdit+: A Fully Configurable Multi-User and Multi-Tool CASE Environment. Lecture Notes in Computer Science, vol. 1080, 1996, pp. 1-21.

Лядова Л.Н., Сухов А.О., Замятнина Е.Б. An Integration of Modeling Systems Based on DSM-Platform. In Proc. of the 18th International Conference on Computers, 2014, pp. 421-425.

Сухов А.О., Лядова Л.Н. MetaLanguage: a Tool for Creating Visual Domain-Specific Modeling Languages. In Proc. of the 6th Spring/Young Researchers’ Colloquium on Software Engineering, 2012, pp. 42-53.

Сухов А.О. Development of tools for creating visual subject-oriented languages. PhD thesis, Moscow, 2013, 256 p. (in Russian) Сухов А.О. Разработка инструментальных средств создания визуальных предметно-ориентированных языков. Диссертация на соискание ученой степени кандидата физико-математических наук, М., 2013: 256 стр.

Лятова Л.Н., Порязов С.А. Hypergraphs with poles as the basis for creating visual language editors. Mathematics of software systems, no. 15, 2018, pp. 97-104 (in Russian) / Сухов А.О. Лядова Л.Н., Поряжов С.А. Гиперграфы с полюсами как основа для создания редакторов визуальных языков. Математика программных систем, вып. 15, 2018, стр. 97-104.

Microsoft. Visual Studio Docs. Overview of Domain-Specific Language Tools. Available at: https://docs.microsoft.com/en-us/visualstudio/modeling/overview-of-domain-specific-language-tools?view=vs-2019, accessed 10.01.2020.

Вулгович В., Максимов М., Перис Б. Comparative analysis of analysis DSM Graphical Editor frameworks: Graphiti vs. Sirius. In Proc. of the 23rd International Electrotechnical and Computer Science Conference ERK, 2014, pp. 7-10.

Павлинов А.А. A set of tools for developing problem-oriented visual languages. Bulletin of Saint Petersburg University, vol. 10, no. 1, 2007, pp. 86-96 (in Russian) / Павлинов А.А. Система средств разработки проблемно-ориентированных визуальных языков. Вестник Санкт-Петербургского университета, том 10, вып. 1, 2007, стр. 86-96.

Терехов А.Н. Architecture of the visual modeling environment QReal. System programming, no. 4, 2009, pp. 172-197 (in Russian) / Терехов А.Н. Архитектура системы визуального моделирования QReal. Программное обеспечение, вып. 4, 2009, стр. 172-197.

Беззин В., Jouault Ф., Touzet Д. An Introduction to the ATLAS Model Management Architecture, Laboratoire d’Informatique de Nantes-Atlantique, Research Report No. 05.01, 2005, 24 p.

Jeong K.-W., Wu L., Hong J. IDEF method-based simulation model design and development. Journal of Industrial Engineering and Management, vol. 2, no. 2, 2009, pp. 337-359.

Serifi V., Dasic P., Jecmenica R., Labovic D. Functional and Information Modeling of Production Using IDEF Methods. Strojniški vestnik – Journal of Mechanical Engineering, vol. 55, no. 2, 2009, pp. 131-140.

Inran S., Foping F., Feohan J., Dokas I. Domain Specific Modeling Language for Early Warning System: Using IDEFO for Domain Analysis. International Journal of Computer Applications, vol. 10, no. 7, 2010, pp. 10-17.

OMG. Unified Modeling Language Specification. Available at: https://www.omg.org/spec/UML/2.5.1/PDF, accessed 15.02.2020.

James P., Knapp A., Moosakowski T., Rzegocki M. Designing Domain Specific Languages – A Craftsman’s Approach for the Railway Domain Using CASL. Lecture Notes in Computer Science, vol. 7841, 2012, pp. 178-194.

Wise R., Brinhall E. A Systems Engineering Approach to the Development of a Domain-Specific Language for Functional Reference Architectures. In Proc. of the 16th Annual Conference on Systems Engineering Research, 2019, pp. 241-254.

Veltor M. MD*/DSL Best Practices Update March 2011. Version 2.0. Available at: http://www.voelter.de/data/pub/DSLBestPractices-2011Update.pdf, accessed 20.03.2020.

Struchkov I.V. A formalism for describing software systems and computational processes for cyclic parallel processing of real time data. Information and control systems, issue 2, 2006, pp.8-13 (in Russian) / Стрючков И.В. Формализм для описания программных систем и вычислительных процессов циклической параллельной обработки данных реального времени. Информационно-управляющие системы, вып. 2, 2006, стр. 8-13.

Courcelle B. Recognizable Sets of Graphs, Hypergraphs and Relational Structures: A Survey. Lecture Notes in Computer Science, vol. 3340, 2005, pp. 1-11.

Information about authors Информация об авторах

Николай Михайлович СУВОРОВ – студент бакалавриата НИУ ВШЭ (Пермь). Научные интересы включают языково-ориентированное моделирование, моделирование, языковые инструментарии.

Людмила Николаевна ЛЯДОВА – кандидат физико-математических наук, доцент кафедры информационных технологий в бизнесе НИУ ВШЭ (Пермь). Сфера научных интересов: языки моделирования, средства моделирования, предметно-ориентированное моделирование, языковые инструментарии, семантическое моделирование.

Ehrig H., Ehrig K., Prange U., Taentzer G. Fundamentals of Algebraic Graph Transformation, Springer, Berlin, 2006, 388 p.

AGG. The Homebase. A brief Description of AGG. Available at: https://www.user.tu-berlin.de/o.runge/agg/agg-docu.html, accessed 12.03.2020.

Eclipse Modeling Project. Eclipse VIATRA. Available at: https://www.eclipse.org/viatri/, accessed 12.03.2020.

Sukhov A., Lyadova L.N. An Approach to Graph Matching in the Component of Model Transformations. In Proc. of the 7th Spring/Summer Young Researchers’ Colloquium on Software Engineering, 2013, pp. 41-46.

Сухов А., Лядова Л.Н. Horizontal Transformations of Visual Models in MetaLanguage System. In Proc. of the 7th Spring/Summer Young Researchers’ Colloquium on Software Engineering, 2013, pp. 31-40.