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

Nikolai M. Suvorov
Department of Business Informatics
National Research University
Higher School of Economics
Perm, Russian Federation
E-mail: SuvorovNM@gmail.com

Lyudmila N. Lyadova
Department of Business Informatics
National Research University
Higher School of Economics
Perm, Russian Federation
E-mail: LNLyadova@gmail.com

***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 restrictions, such as inefficiency and complexity of operations and insufficient expressiveness of the created models. 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, not only providing the ability to define and implement new visual languages, but also providing a basis for implementing operations on models built using these languages. Definitions of the HP-graph and its elements are given. Justification of expressive power of the HP-graph is presented. Main operations for the HP-graph are described. The chosen graph formalism unites expressive possibilities of various types of graphs and allows the creation of a flexible visual model editor based on it for a DSM-platform.**

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

I. 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 modeling 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;
- have an ability to alienate the created modeling language from the system where it has been developed;
- have an ability to modify the visual model of the system, rather than the source code, when a modeled process or system undergoes changes;
- unify representation and description of both models and metamodels, which allows a person to work with models and metamodels using the same tools, as well as, for example, provides the opportunity to perform vertical and horizontal transformations of visual models.

To create a tool that has all these features, the development of a new formal model is needed. Visual languages are usually based on graph models [1], but the types of graphs used have limitations, such as insufficient expressiveness of the created models, inefficiency, and complexity of operations. However, there is a more powerful formal model that solves these problems, but has not been used by developers yet, which is called a hypergraph with poles (HP-graph) [9], which connects the expressive capabilities of various types of graph models.

II. 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 DSL 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 addons 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 metalanguage, 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 I, 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. 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.

TABLE I. The comparison of the tools

Requirements	MS DSL Tools	Eclipse Sirius	Meta Edit+	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	–	+	–	–	–

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.

III. 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 $P = \{\pi_1,\ldots,\pi_n\}$ is a set of external poles, $V = \{v_1,\ldots,v_m\}$ is a non-empty set of vertices, $W = \{w_1,\ldots,w_l\}$ 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 $\forall v_i \in V, \forall v_j \in V [i \neq j \rightarrow v_i \cap v_j = \varnothing]$, 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 $P_v = \{p_{v_1},\ldots p_{v_n}\}$ ($\forall v \in V\ \exists I(v) \subset P_v$, $\exists O(v) \subset P_v [I(v) \cup O(v) = v]$). 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 ($w = P_w = \{p_{w_1},\ldots,p_{w_k}\} \subset 2^{Pol}$). An edge cannot be represented as an empty set ($\forall w \in W$: $[P_w \neq \varnothing]$). 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 $\forall v \in V(G)$, $\forall w \in W(G)$ the following condition must be met: $[\exists p \in w \cap (I(v) \cup I(G))$ *and* $\exists r \in w \cap (I(v) \cup I(G))]$.

An example of the hypergraph with poles is demonstrated on Fig. 1.

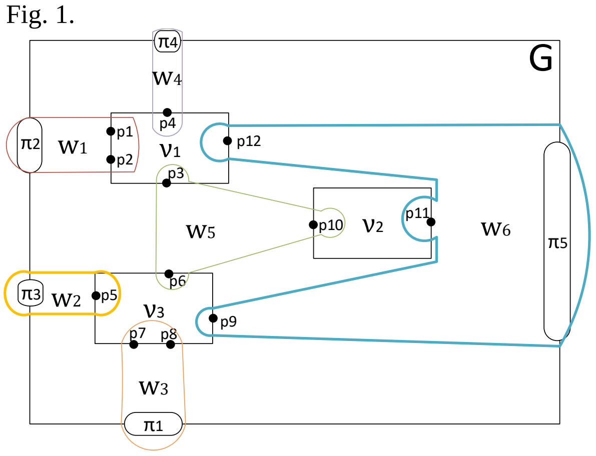

Fig. 1. Example of an HP-graph

On this figure external poles are represented by a set $P = \{\pi_1,\ldots,\pi_5\}$, edges are represented by a set $W = \{w_1,\ldots,w_6\}$ and vertices are represented by a set $V = \{v_1, v_2, v_3\}$. Every vertex contains a certain number of poles p, 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".

A. 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, $p1$ as an inner pole, $p2$ as an outer pole and w as an edge.

The following operations *add elements* to an HP-graph:

- $v + p1$ 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:
$$Pol(G') = Pol(G) \cup \{p1\},$$
$$v' = v \cup \{p1\}.$$
- $G + v$ 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 the set of vertices $V(G)$, and all poles of this vertex are added to the set of all poles of the graph:
$$Pol(G') = Pol(G) \cup v,$$
$$V(G') = \{V(G) \cup \{v\} \mid |v|>0\}.$$
- $G + w$ is the addition of the edge 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 w, and $O(w)$ be the set of output poles of w, then the operation is represented as:
$$W(G') = \{W(G) \cup \{w\} \mid |I(w)|>0 \textit{ and } |O(w)|>0\}.$$
- $w + p1$ is the addition of the inner pole to the edge. An existing pole belonging to one of the vertexes is added to the edge ($\exists v \in V(G)\ [p1 \in v]$), which is represented as:
$$w' = w \cup \{p1\}.$$
- $w + p2$ is the addition of the outer pole to the edge. An existing pole belonging to the set of outer poles ($p2 \in P(G)$) of the graph is added to the edge:
$$w' = w \cup \{p2\}.$$
- $G + p2$ is the addition of the outer pole to the graph. A pole is added to both the set of outer poles of the graph G and the set of all poles:
$$Pol(G') = Pol(G) \cup \{p2\},$$
$$P(G') = P(G) \cup \{p2\}.$$

The following operations *remove elements* from an HP-graph:

- $v - p1$ is the removal of the inner pole from the node. 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:
$$\forall w \in W(G)\ [w = \{w \backslash \{p1\} \mid \{p1\} \in w\}],$$
$$Pol(G') = Pol(G) \backslash \{p1\},$$
$$v' = \{v \backslash \{p1\} \mid |v|>1\}.$$

- $G - v$ is the removal of the node from the graph. In addition to deleting a vertex, all occurrences of the poles of this vertex in the edges are cut off, and all poles of the vertex are removed from the set of poles of the graph:

$$\forall w \in W(G)\ \forall p \in v\ [w = \{w \setminus \{p\} \mid p \in w\}],$$
$$Pol(G') = Pol(G) \setminus \{v\},$$
$$V(G') = V(G) \setminus \{v\}.$$

- $G - w$ 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:

$$W(G') = W(G) \setminus \{w\}.$$

- $w - p1$ is the removal of the inner pole from the edge. 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:

$$w' = w \setminus \{p1\},$$
$$W(G') = \{W(G) \setminus \{w\} \mid |I(w)| = 0 \ or\ |O(w)| = 0\}.$$

- $w - p2$ is the removal of the external pole from the edge, which is equal to the previous operation.

- $G - p2$ is the removal of the outer pole from the graph, which also includes removing this pole from all edges that contain this pole:

$$\forall w \in W(G)\ [w = \{w \setminus \{p2\} \mid \{p2\} \in w\}],$$
$$P(G') = P(G) \setminus \{p2\},$$
$$Pol(G') = Pol(G) \setminus \{p2\}.$$

B. *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 $\forall p \in I(v)$: $[f(p) \in I(G)]$, $\forall r \in O(v)$: $[f(r) \in O(G)]$. Thus, the mapping of the pole $p \in v$ 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 π for a resulting graph.

Fig. 2 illustrates decomposition of the vertex v_3 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 $\forall p \in I(w)$: $[f(p) \in I(G)]$, $\forall r \in O(w)$: $[f(r) \in O(G)]$. Thus, the mapping of the pole $p \in w$ 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_6 is demonstrated on Fig. 3.

As is seen, the decomposition of edges and vertices is almost equal, therefore, it is possible to define a common decryption algorithm for these structures.

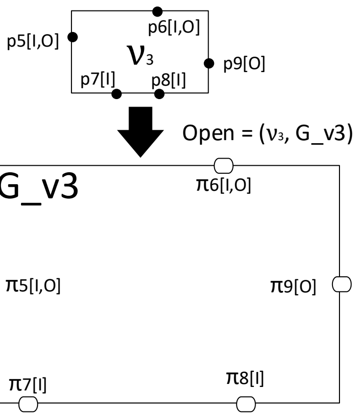

Fig. 2. Example of vertex decomposition by a new HP-graph

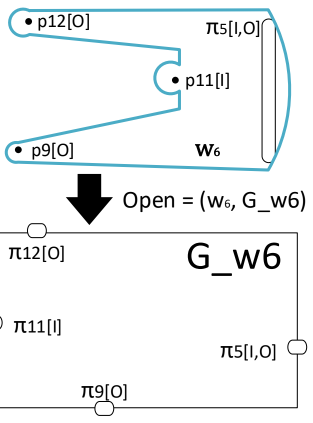

Fig. 3. Example of edge decomposition by a new HP-graph

To do it, let us define a set of structures $Str = V \cup W$. Hence, $str \in Str$ is a structure which can be either a vertex or an edge. The algorithm of the structure decomposition by a new HP-graph can be described as follows:

Algorithm 1. Procedure *DecomposeStructure*

```
G = new HPGraph();
foreach $p \in str$:
  if ($p \in I(str)$):
    $I(G) = I(G) \cup p$;
  if ($p \in O(str)$):
    $O(G) = O(G) \cup p$;
$Open_{str} = Open_{str} \cup (str, G)$
```

As is seen from the algorithm, for every structure str several decoding operations can be defined. Generally, they can be presented as $Open_{str} \subset str \times G_{all}$, where G_{all} is the set of all HP-graphs determined on the set Pol.

An edge of an HP-graph can also be decrypted by ordinary links between the poles. To implement this operation, it is necessary to define the set $E_w = \{e_1,\ldots,e_n\} \subset I(w) \times O(w)$ for each edge $w \in W$, so that every link ($e \in E_w$) is represented by a pair (p, r) provided that $p \in I(w)$, $r \in O(w)$. Thus, the decoding of the edge $w \in W$ can be represented by the mapping function f: $w \rightarrow E_w$, which replaces the hyperedge with normal connections between the input and output poles.

Fig. 4 illustrates the example of hyperedge decoding by ordinary links. As $E_w \subset I(w) \times O(w)$, some input and output poles can be unconnected such as poles $p9[O]$ and $p11[I]$ in Fig. 4.

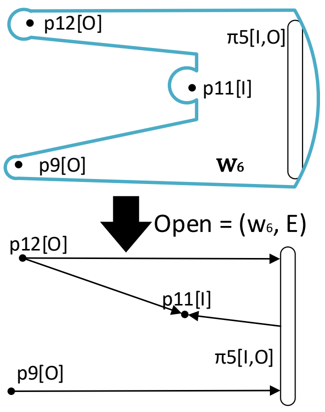

Fig. 4. Example of edge decomposition by a ordinary links

C. *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 toolsets, 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 HP-graph. A *subgraph* of the HP graph $G = (P, V, W)$ is an HP-graph $G' = (P', V', W')$ that is part of the graph G $(P' \subset P$ & $(\forall v' \in V'$ $\exists v \in V$: $[v' \subset v])$ & $W' \subset W)$ and fulfills the condition $Open' \subset Open$. The subgraph must also meet the condition (1) to make transformation operations possible.

$$(\forall v \in V' \setminus V'_{partial}\,[p \in v]\ \&\ \exists w \in W[p \in w]) \rightarrow w \in W' \qquad (1)$$

The set $V'_{partial}$ is a set of the *incomplete vertices* in the graph, where $V'_{partial} \subset V'$.

A subgraph can contain vertices called *incomplete* whose sets of poles can only be 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 $p = (G_L, G_R)$, where G_L is a pattern-graph and G_R 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 replacement graph to the original graph.

The first step can be described as follows:

Algorithm 2. Function *DeleteGraph(HostG, G_L)*

```
$G' = Find\_Isomorphic\_Subgraph(HostG, G_L)$;
$partials = \{\}$
**foreach** $w' \in W(G')$:
  $W(HostG) = W(HostG) \setminus \{w'\}$;
**foreach** $v' \in V(G')$:
  **if** $(v' \in V(HostG))$:
    $V(HostG) = V(HostG) \setminus \{v'\}$;
  **else**:
    $partials = partials \cup \{v'\}$;
**foreach** $p' \in P(G')$:
  **if** $(\neg\exists w \in W(HostG)[p' \in w])$:
    $P(HostG) = P(HostG) \setminus p'$;
**return** $partials$;
```

The second step of the algorithm will be following:

Algorithm 3. Procedure *AddGraph(HostG, G_R, partials)*

```
**foreach** $p \in P(G_R)$:
  **if** $p \notin P(HostG)$:
    $P(HostG) = P(HostG) \cup \{p\}$;
**foreach** $v \in V(G_R)$:
  **if** $(v \notin Partials)$:
    $V(HostG) = V(HostG) \cup \{v\}$;
**foreach** $w \in W(G_R)$:
  $W(HostG) = W(HostG) \cup \{w\}$;
```

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

IV. 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 II describes formulas which represent any of these graph structures as an HP-graph.

TABLE II. Representation of Graphs as an HP-Graph

Graph model	Representation in the HP-graph $G' = (P', V', W')$
Oriented Graph $G = (V, E)$	$V = P' = V'$, where $\forall v' \in V'$: $[\lvert v'\rvert = 1]$ $E = W'$, where $\forall w' \in W'$: $[\lvert w'\rvert = 2])$
Hypergraph $G = (X, E)$	$X = P' = V'$, where $\forall v' \in V'$: $[\lvert v'\rvert = 1]$ $E = W'$
Hi-graph $G = (X, E)$	$\{x \mid x \in X\ \&\ \lvert x\rvert = 1\} = P' = V'$, where $\forall v' \in V'$: $[\lvert v'\rvert = 1]$ $E \cup \{x \mid x \in X\ \&\ \lvert x\rvert > 1\} = W'$
Metagraph $G = (V, MV, E)$	$V = P' = V'$, where $\forall v' \in V'$: $[\lvert v'\rvert = 1]$ $E \cup MV = W'$
P-graph $G = (P, V, W)$	$P = P'$ $V = V'$ $W = W'$, where $\forall w' \in W'$: $[\lvert w'\rvert = 2]$

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.

V. 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 unites 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

[1] Koznov D.V. Osnovy vizual'nogo modelirovanija [Basics of visual modeling]. Uchebnoe posobie [Textbook], 2007 (in Russian).

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

[3] Ward M. P. Language Oriented Programming. *Software – Concepts & Tools*, 1994, vol. 15, no. 4, pp. 147-161.

[4] Sukhov A.O. Sravnenie sistem razrabotki vizual'nyh predmetno-orientirovannyh jazykov [Comparison of visual object-oriented language development systems]. *Matematika programmnyh sistem: mezhvuzovskij sbornik nauchnyh statej* [*Mathematics of software systems: Intercollegiate collection of scientific articles*], 2012, no. 9, pp. 84-111 (in Russian).

[5] Kelly S., Lyytinen K., Rossi. M. MetaEdit+: A Fully Configurable Multi-User and Multi-Tool CASE Environment. In: *Proceedings of CAiSE'96, 8th Intl. Conference on Advanced Information Systems Engineering, Lecture Notes in Computer Science*, 1996, vol. 1080, pp. 1-21.

[6] Lyadova L.N., Sukhov A.O., Zamyatina E.B. An Integration of Modeling Systems Based on DSM-Platform. In: *Advances in Information Science and Applications. Volumes I & II. Proceedings of the 18th International Conference on Computers*, 2014, pp. 421-425.

[7] Sukhov A.O., Lyadova L.N. MetaLanguage: a Tool for Creating Visual Domain-Specific Modeling Languages. In: *Proceedings of the 6th Spring/Summer Young Researchers' Colloquium on Software Engineering*, 2012, pp. 42-53.

[8] Sukhov A.O. Razrabotka instrumental'nyh sredstv sozdanija vizual'nyh predmetno-orientirovannyh jazykov [Development of tools for creating visual subject-oriented languages]. PhD thesis, Moscow, 2013, 256 p. (in Russian).

[9] Sukhov A.O., Lyadova L.N., Poryazov S.A. Gipergrafy s poljusami kak osnova sozdanija redaktorov vizual'nyh jazykov [Hypergraphs with poles as the basis for creating visual language editors]. *Matematika programmnyh sistem* [*Mathematics of software systems*], 2018, no. 15, pp. 97-104 (in Russian).

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

[11] Vujovic V., Maksimovic M., Perisic B. Comparative analysis of DSM Graphical Editor frameworks: Graphiti vs. Sirius. 23nd International Electrotechnical and Computer Science Conference ERK, 2014, pp. 7-10.

[12] Pavlinov A.A. Kompleks sredstv razrabotki problemno-orientirovannyh vizual'nyh jazykov [A set of tools for developing problem-oriented visual languages]. *Vestnik Sankt-Peterburgskogo universiteta* [*Bulletin of Saint Petersburg University*], 2007, vol. 10, no. 1, pp. 86-96 (in Russian).

[13] Terekhov A.N. Arhitektura sredy vizual'nogo modelirovanija QReal [Architecture of the visual modeling environment QReal]. *Sistemnoe programmirovanie* [*System programming*], 2009, no. 4, pp. 172-197 (in Russian).

[14] Bezivin J., Jouault F., Touzet D. An Introduction to the ATLAS Model Management Architecture, 2005, 24 p.

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

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

[17] Imran S., Foping F., Feehan J., Dokas I. Domain Specific Modeling Language for Early Warning System: Using IDEF0 for Domain Analysis. *International Journal of Computer Science Issues*, 2010, vol. 7, no. 5, pp. 10-17.

[18] OMG. Unified Modeling Language Specification [Online]. Available: https://www.omg.org/spec/UML/2.5.1/PDF [Accessed 15.02.2020].

[19] James P., Knapp A., Mossakowski T., Roggenbach M. Designing Domain Specific Languages – A Craftsman's Approach for the Railway Domain Using CASL. In: *International Workshop on Algebraic Development Techniques, Lecture Notes in Computer Science*, 2012, vol. 7841, pp. 178-194.

[20] Wise R., Brimhall E. A Systems Engineering Approach to the Development of a Domain-Specific Language for Functional Reference Architectures. In: *Systems Engineering in Context*, 2019, pp. 241-254.

[21] Velter M. MD*/DSL Best Practices Update March 2011. Version 2.0 [Online]. Available: http://www.voelter.de/data/pub/DSLBestPractices-2011Update.pdf [Accessed: 20.03.2020].

[22] Struchkov I.V. Formalizm dlja opisanija programmnyh sistem i vychislitel'nyh processov ciklicheskoj parallel'noj obrabotki dannyh real'nogo vremeni [A formalism for describing software systems and computational processes for cyclic parallel processing of real time data]. *Informacionno-upravljajushhie sistemy* [*Information and control systems*], 2006, no. 2, pp. 8-13.

[23] Courcelle B. Recognizable Sets of Graphs, Hypergraphs and Relational Structures: A Survey. In: *Developments in Language Theory. International Book Series «Lecture Notes in Computer Science»*, 2005, vol. 3340, pp. 1-11.

[24] Grosu R., Stefanescu Gh., Broy M. Visual Formalisms Revisited. In: *Proceedings 1998 International Conference on Application of Concurrency to System Design*, 1998, pp. 41-51.

[25] Power J., Tourlas K. Abstraction in Reasoning about Higraph-Based Systems. In: *Foundations of Software Science and Computation Structures. International Book Series «Lecture Notes in Computer Science»*, 2003, vol. 2620, pp. 392-408.

[26] Basu A., Blanning R. Graphs, Hypergraphs, and Metagraphs. In: *Metagraphs and Their Applications. Integrated Series in Information Systems*, 2007, vol 15.

[27] Mikov A.I. Performance evaluation: textbook, 2013, 89 p.

[28] Filatov D.Ju., Lyadova L.N. Razrabotka redaktora vizual'nyh modelej, osnovannogo na P-grafah [Development of P-graph based visual model editor]. In: *Tehnologii razrabotki informacionnyh sistem (TRIS-2017): 104 Materialy VIII Mezhdunarodnoj nauchno-tehnicheskoj konferencii* [*Information Systems Development Technologies (TRIS-2017): 104 Materials of the VIII International Scientific and Technical Conference*], 2017, pp. 113-118 (in Russian).

[29] Parra F. Dean T. Survey of Graph Rewriting applied to Model Transformations. In: *Proceedings of the 2nd International Conference on Model-Driven Engineering and Software Development*, 2014, pp. 431-441.

[30] Ehrig H., Ehrig K., Prange U., Taentzer G. Fundamentals of Algebraic Graph Transformation, 2006, 388 p.

[31] AGG The Homebase. A brief Description of AGG [Online]. Available: https://www.user.tu-berlin.de//o.runge/agg/agg-docu.html [Accessed: 12.03.2020].

[32] Eclipse Modeling Project. Eclipse VIATRA [Online]. Available: https://www.eclipse.org/viatra/ [Accessed: 12.03.2020].

[33] Seryi A.P., Lyadova L.N. An Approach to Graph Matching in the Component of Model Transformations. In: *Proceedings of the 7th Spring/Summer Young Researchers' Colloquium on Software Engineering, SYRCoSE 2013*. Kazan : 2013, pp. 41-46.

[34] Sukhov A., Lyadova L. N. Horizontal Transformations of Visual Models in MetaLanguage System. In: *Proceedings of the 7th Spring/Summer Young Researchers' Colloquium on Software Engineering, SYRCoSE 2013*. Kazan, 2013, pp. 31-40.