Designing Data Visualization System Based on Language-Oriented Approach

Anna Dzheiranian
Department of Business Informatics
HSE University
Perm, Russian Federation
ORCID: 0009-0000-8916-2855

Ivan Ermakov
Department of Computing Systems Software
Perm State University
Perm, Russian Federation
ORCID: 0000-0003-2897-7158

Kirill Proskuryakov
Department of Business Informatics
HSE University
Perm, Russian Federation
ORCID: 0009-0001-1678-5653

Lyudmila Lyadova
Department of Business Informatics
HSE University
Perm, Russian Federation
ORCID: 0000-0001-5643-747X

***Abstract* — The data visualization method based on a language-oriented approach is proposed. An analysis of data visualization tools and their customizability for subject areas based on user needs has been conducted. However, these tools require highly qualified users to customize the data visualization form (users must have programming skills). It is proposed to customize visualization tools to the needs of users and the specifics of the user's tasks being solved by creating domain-specific languages (DSL). A system architecture based on the use of multifaceted ontology is proposed. The ontology includes descriptions of languages and domains, as well as rules for generating new languages and transforming constructed models. Languages are designed to describe different classes of diagrams. This system includes tools for automatic new DSL generation via mapping domain ontology to the base language metamodel. Different types of diagrams have been classified and the main components of each have been identified, which provides the basis for creating an ontology for data visualizations languages. A base language is proposed for creating diagrams. The language customizability for specific domains is demonstrated. An example of the created data visualization models is provided.**

Keywords — data visualization, domain-specific modeling, domain-specific languages, metamodeling, multifaceted ontology, model transformation.

I. INTRODUCTION

Visualization tools have gained widespread use in various industries, business functions, and IT disciplines, both in the private and public sectors. They are actively used in such areas as energy industry, cartography, structural health monitoring, discrete mathematics, nutrition, biology, finance, social networks, and many others. In this context, data visualization serves as a method of data analysis.

The needs of end users (data analysts) include the necessity to create custom types of diagrams for specific tasks and domains, as basic types of diagrams with basic geometries can limit information transmission [1] and lead to ineffective visualization, which, in turn, can lead to mistakes in the decision-making [2]. Often such customization requires the use of a programming language [3]. The lack of deep programming knowledge among users leads to the need to create low-code platforms.

The available data visualization tools systems can be categorized into following groups: (1) spreadsheets (e.g., Excel, Google Sheets), (2) analytics platforms (e.g., Microsoft Power BI, Tableau), (3) diagram editors (e.g., Miro, ChartBlocks). The standard tools of the first two groups are limited to the basic diagram types and visual effects customization options. The tools of the third group allow to create custom visualizations, edit the locations of elements, but they do not provide settings for domains.

In contrast, the use of general-purpose programming languages (e.g., Python libraries for data visualization: Matplotlib, Seaborn, Plotly, etc.) contributes to the creation of the expressive visualizations to solve specific tasks, but it requires deep programming knowledge from the diagram developer. Also, the created solution cannot be reused for other visualizations, and it is a “black box” where it is not clear how the visualization is configured [3].

Thus, the end users face two challenges:

1. How to automate visualizations development to reduce level of requirements for user programming knowledge?
2. How to ensure the customization of visualizations for specific domains?

To enhance visualization tools, a language-oriented approach is proposed. Initially, the authors proposed the concept of automating domain-specific languages (DSL) creation based on using multifaceted ontology [4], [5]. To implement the approach, the following tasks must be solved:

1. To determine specific user requirements for customizing data visualizations for the tasks being solved and domains. To identify the problems that users face in order to remove these limitations through the development of DSL.
2. To analyze and classify data visualization diagrams to identify the foundation for developing the data visualization languages and formalize the results for ontology development.
3. To determine the general structure of the data visualization system.
4. To develop the ontology of data visualization languages based on the completed classisication of charts.
5. To describe the base language meta-model for the development of new data visualization languages.
6. To give an example of data visualization model customization.
7. To describe a code generation approach for data visualization based on created models.

II. Requirements and Design Challenges

The specificity of user requirements lies in the need to identify new types of diagrams. This result can be achieved by interactivity, combining different types of diagrams (e.g., a combination of a histogram and a line graph) or different elements/shapes within a single diagram, etc. Users require the creation of visualizations adapted to specialized tasks and domains too. There is also a need for modeling previously created diagrams and implementing visualization specifications into the system. These specifications define how users can specify their requirements for creating visualizations [6]. Statistical visualizations may not be effective for comparing concepts within large data sets. Therefore, there is a demand for the rapid and straightforward creation of interactive visualizations [3]. These methods should be capable of working with various types of data and data sources [6].

Modern researchers [2], [7] highlight the limited degree of flexibility in manipulating diagram elements and the lack of focus on the real needs of the user in existing data visualization tools. Another challenge in information visualization is the loss of data transmission efficiency when an inappropriate visualization method is chosen.

III. Related Works

A. Classification of Visualization Methods

The diversity of visualization methods is quite large and continues to expand, which is confirmed by the constant emergence of new specialized types of diagrams (tree-like, chord, network, etc.).

Determining the most appropriate type of visualization is not an exact science and involves a multitude of approaches, but it is based on the key question: what idea do you want to convey with the diagram [8], [9]. To determine which specific visualization methods are sufficient to implement most visualization ideas, it is necessary to settle on a specific taxonomy for classifying data visualization methods from the existing range.

It was decided to focus on the five-category structure proposed by Andy Kirk [9]: (1) comparing categories, (2) assessing hierarchies and part-to-whole relationships, (3) showing changes over time, (4) plotting connections and relationships, and (5) mapping geospatial data. It is followed by a review of each of these categories, highlighting specific diagrams, their unique characteristics, and elements.

The main classification criteria have been identified. These criteria are listed below:

1. The greatest popularity of visualization methods.
2. The inclusion of methods for visualizing abstract data that is characterized by multiple dimensions and the absence of explicit spatial references in addition to standard visualization methods (line graphs, histograms, pie charts, bar charts, and scatter plots).
3. The ability to present any type of visualization in an interactive form, enhancing their utility for large-scale datasets.

As a result, a classification consisting of sixteen visualization methods depending on the purpose of creating the visualization was developed (Fig. 1). This classification is the basis for the development of the data visualization languages ontology and languages metamodels.

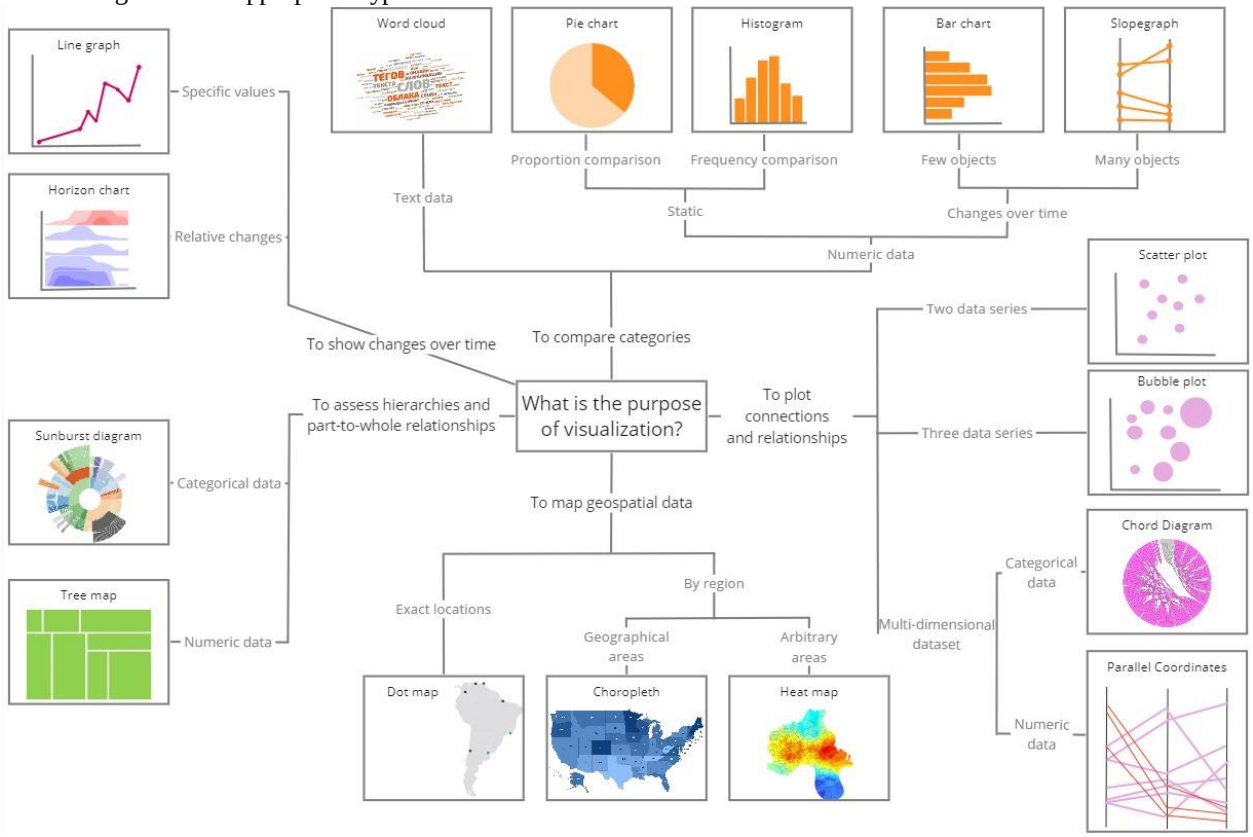

Fig. 1. Classification of visualization methods by purpose

B. Data Visualization Tools and Customization Options

There are few publications on creating DSLs for data visualization. Based on the review, most DSLs focus on a small set of standard charts (pie charts, histograms, etc.) or visualizations of specific data types (e.g., geospatial). They differ in levels of abstraction, contexts of use, and implementation capabilities.

The article [10] describes the process of developing a DSL for constructing and transforming data visualization techniques. The DSL is built into the Haskell programming language. The authors provide several levels of abstraction: at the lowest level, the user can create an element consisting of a specific primitive shape and a set of visual parameters. It is important to note that the basic language constructs are limited to the histogram and pie chart. However, it is allowed to arrange their elements in different ways to create more complex examples.

Article [11] presents a variational visualization model implemented through a DSL built into the PureScript programming language. This DSL allows to create variation visualizations and their combinations, such as overlaying alternative histograms. The article also discusses methods for representing variation and adding variation to visualizations via DSL. This includes creating, manipulating, navigating, and rendering variational visualizations.

The researchers in study [12] introduce a DSL that is focused on data geovisualizations. They utilize a compiler to facilitate the automatic generation of visualizations and the pre-processing of data. Their system leverages the power of multi-core parallelism to expedite the data pre-processing.

The considered tools provide the ability to develop new types of diagrams. But customization for specific domains was not found in them. Thus, a language-oriented approach for implementing a data visualization tool can become the main one for developing a data visualization system.

C. Ontology-Driven Approach to Implementing DSL Toolkits

In papers [13], [14] has been suggested to use ontologies as part of the architecture for the analytical platform. In this case, a multifaceted ontology is used, which allows to avoid data duplication, ensure changes traceability of ontologies, and automatically interpret data and the results of data analysis to provide them to different groups of users according to terminology that they are familiar with.

Use of ontologies is also considered by researchers within the domain-specific modeling (DSM) approach [15]. Domain-specific modeling is the part of model-driven engineering approach. It allows for the reduction of complexity in software system development by using DSL's. Language toolkits (DSM platforms) are used to implement this approach. They enable the generation of all essential components for working with the language (graphic editor, interpreter, etc.) according to the described metamodel.

One of the ideas for implementing the DSM approach is the usage of knowledge, specifically – the ontologies [4], [5], [13]. Due to this, it is possible to automate the process of creating domain-specific languages. This approach is taken as the basis for the development of a data visualization system.

IV. Generalized Structure of the Data Visualization Tools Based on the DSM Approach

The Fig. 2 shows the structure of the data visualization platform based on the DSM approach. The core of the system is a multifaceted ontology. This is an ontology that includes many other ontologies. They can be divided into three groups:

1. *Ontology of languages.* This is an ontology, in which metamodels of visual and textual languages are stored in accordance with a certain classification (by task or methodology). To describe metamodels, the HPGPR [15] metalanguage is used – an extended version of MetaCase's GOPPRR language. Models are also presented in the ontology of languages as instances of a model class.
2. *Information ontology*. This group may include several types of ontologies. First, the ontologies of data sources – they include information about data types, data storage formats, relationships, attributes, etc. Secondly, the domain ontologies. These ontologies contain a description of a specific subject area – the basic concepts (objects) and the relationships between them.
3. *Applied ontologies.* These are ontologies, that are used during metamodel generation (projection rules ontology) and model tranformations (transsformation rules ontology).

The platform is also partitioned into logical blocks. Let's take a closer look. First of all, this is a block of *Functional Modules.* These are modules responsible for the basic functionality of the platform – model creation and editing, model transformations, code generation and others. *The Language Generation Module* is used to automate the creation of metamodels using ontologies. It is based on the idea of reusing previously created languages with their reconfiguration to new subject areas through domain ontologies. *The Models Management Module* is used to manage models in the platform, import and export them from ontologies and convert into a view applicable within the platform. Both other modules use this module to access models and languages.

V. Developing languages for Creating Data Visualization Models

To create a new language the user needs to complete several preliminary steps. First, an ontology of data visualization should be developed. At the same time, metamodels of basic languages for creating diagrams should be developed. The basis of the ontology should consist of diagram classification and a special language that enables the creation of these languages based on it. To customize created languages to the user's domain, it is proposed to use the mentioned approach for automizing language creation via mapping the concepts of the domain onto the metamodel of the base language [13].

A. Data Visualization Ontology

The process of building an ontology of data visualization languages includes the following steps:

1. Formalization of an abstract diagram, which will include properties common to all types of diagrams (title, legend, width, height, etc.).

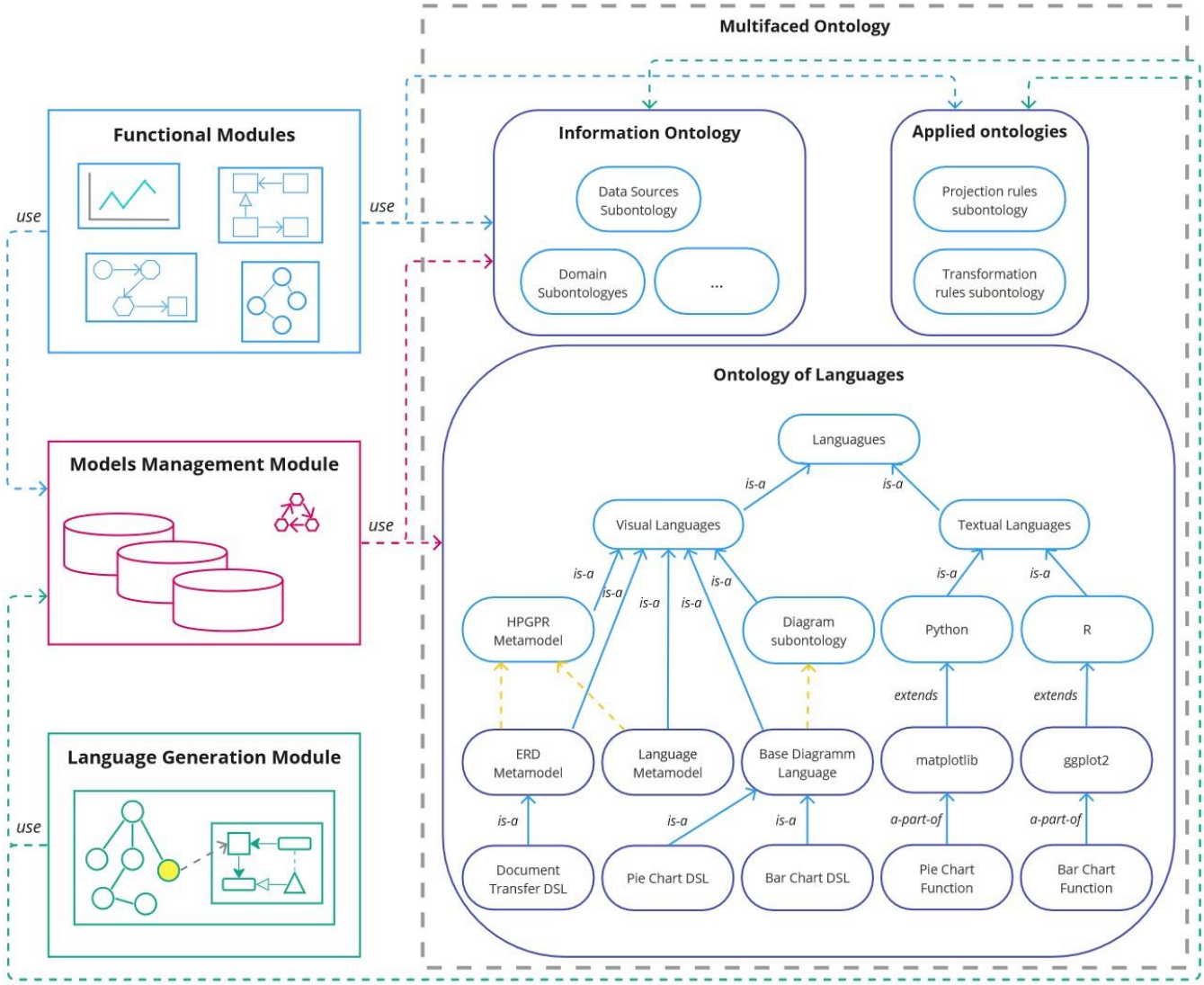

Fig. 2. Generalized structure of the data visualization tools

2. Distinguishing types of diagrams into separate classes based on the developed classification of diagrams (Fig. 1).
3. Adding unique elements of charts to classes.
4. Defining relationships between the classes.

As a result, each type of diagram will have its own model formalized, for which its own DSL will be generated later.

Fig. 3 (*a*) shows an abstract diagram class and its descendants in the ontology. But it is not enough to create only subclasses of diagrams. Each of the charts consists of separate elements that have their own properties. Such elements are separated into subclasses of the "*Diagram_Element*" class (Fig. 3, *b*). After defining the main classes and their unique elements, it is necessary to define the hierarchical and part-to-whole relationships between the classes. Relationships "*as-is*" are automatically created between the parent class and the child class. Then, to display the "*part-of*" relationships between a specific diagram and its elements, special connections were created. The list of relationships is shown in Fig. 3 (*c*).

B. Base Language Metamodel and Example

The classification of diagrams and the identification of its main components allowed the development of a new DSL for creating diagrams. The metamodel of this language is shown in Fig. 4. Created language allows building data flows from sources to a diagram, with the ability to filter data. User can attach various events to components, such as mouse hover or mouse click.

Created metamodel describes an abstract language. This language is the basis for the development of new languages for visualizing data in a specific domain area. It needs to be modified for a specific type of diagrams in order to use. Designed language metamodel can be extended and customized by user with mapping domain model onto components of diagrams.

C. Language Customization Example

For example, let's take the customer evaluation of service. As a basic language, we take a bar chart. Now we need to create an ontology in which we describe emoji's – add vertices for each emotion and set an image for them. During the language generation, we will need to specify the correspondence between the concepts of the domain ontology and the concepts of the diagram language. As a result, we get a language that allows to create diagrams, as in Fig. 5.

Such language uses domain terms, and it is easy to use for end users.

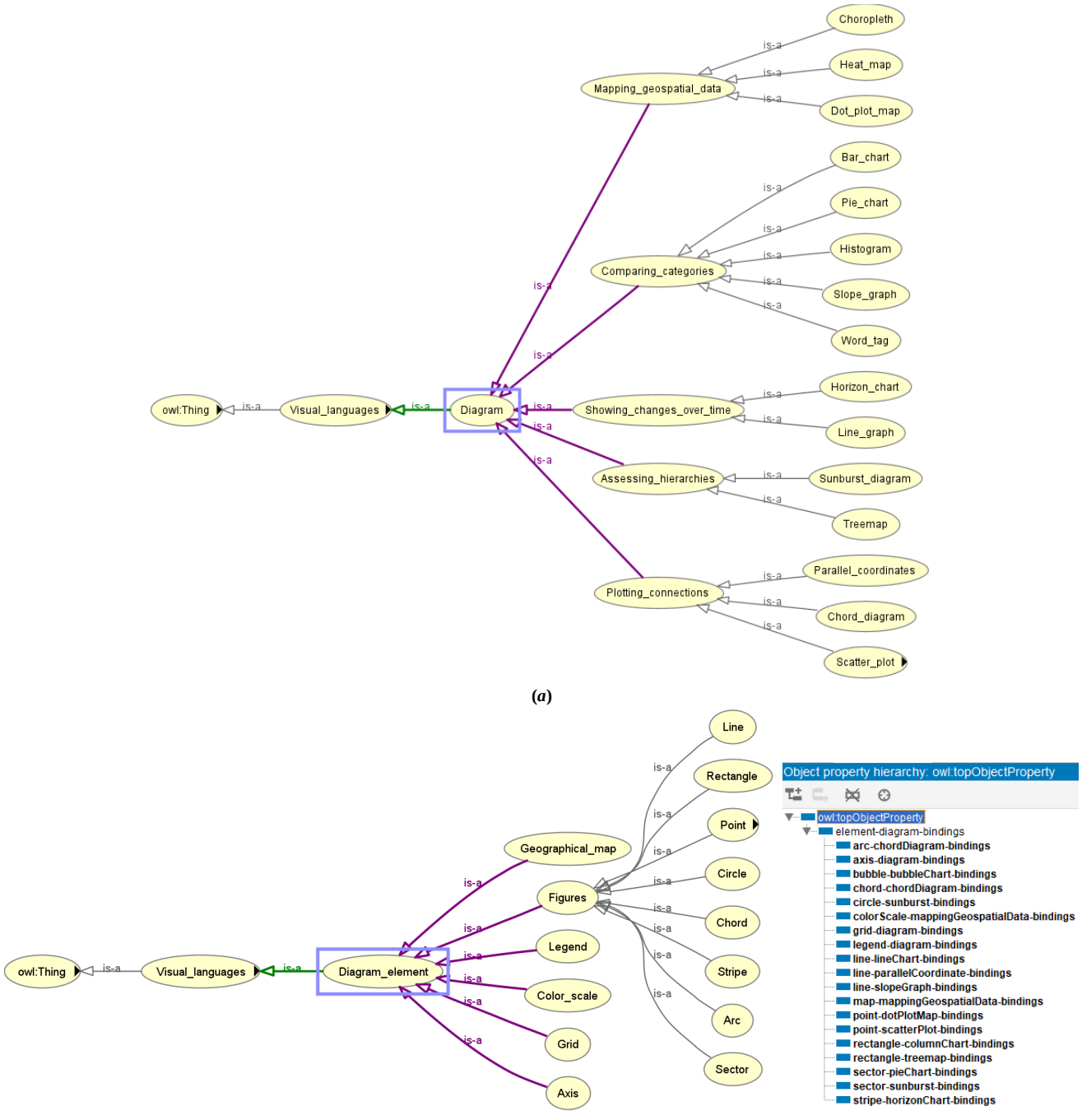

Fig. 3. Fragments of the multifaceted ontology:*a*) diagram classes hierarchy in the ontology; *b*) diagram element class hierarchies; *c*) relation types

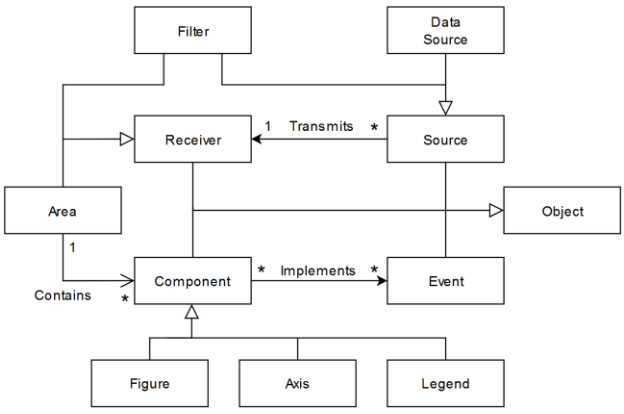

Fig. 4. B. Metamodel of the base language for diagram creation

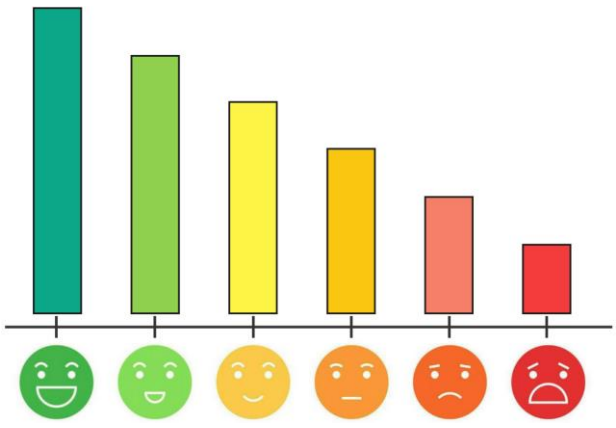

Fig. 5. B. Diagram customization example

VI. Generating Code for Data Visualization Based on Created Models

Using DSL, visualization models are built, but code generation (also called Model-To-Text transformation) is required for data visualization. The vast majority of modern DSM platforms use a template-based approach to code generation, which allows efficient templates reuse [16]. Templates are described not for a specific model, but for a metamodel [17], or in this case a visualization language. Each template consists of two parts – static and dynamic. The static part is the same for all models, and the dynamic part uses information "extracted" from a particular model.

In this paper, we propose to use the approach described in [13], which consists in creating and using transformation rules in the visual environment in the form of a constructor. Each rule consists of the left part – visual language metamodel objects and the corresponding right part – textual language constructs, which are templates. All language constructs, as well as created transformation rules, are stored in a multifaceted ontology.

Python and R are the preferred programming languages for data visualization purposes. Python is widely popular in this area due to its wide range of suitable libraries, including Matplotlib, Seaborn and Plotly, and its simple syntax. Although R is second only to Python in industry, it also has a rich arsenal of visualization tools and continues to be a popular choice in academia. Thus, as textual language constructs, it makes sense to use code fragments in Python and R containing calls to library functions for data visualization.

Once transformation rules are created, they can be applied to specific models to generate data visualization code. The code generation algorithm is based on traversing the internal representation of models, which can be modeled with a graph.

VII. Conclusion

In this paper a structure of a knowledge-driven data visualization tool based on a language-oriented approach is proposed. This approach allows overcoming several limitations of existing visualization tools and providing users an ability to customize the data visualization models for different domain areas via creating special languages and reduces the requirements for end user's programming skills when creating DSL and charts.

The practical applicability of this approach is demonstrated through the example of creating a chart for assessing customer service quality. New base domain-specific language metamodel for data visualization was created with using the proposed approach. Then a diagram model was built with created DSL and data visualization was built according to the specified requirements.

The next stage of the research is implementing of the described ideas and expanding possibilities of interactive visualization via interpretation of the created models.

References

[1] S. R. Midway, "Principles of Effective Data Visualization," Patterns, vol. 1, no. 9, p. 100141, Nov. 2020, doi: 10.1016/j.patter.2020.100141.

[2] E. Oral, R. Chawla, M. Wijkstra, N. Mahyar, and E. Dimara, "From Information to Choice: A Critical Inquiry Into Visualization Tools for Decision Making," in IEEE Transactions on Visualization and Computer Graphics, vol. 30, no. 1, pp. 359–369, Jan. 2024, doi: 10.1109/TVCG.2023.3326593.

[3] R. Morgan, G. Grossmann, M. Schrefl, M. Stumptner, and T. Payne, "VizDSL: A Visual DSL for Interactive Information Visualization," Advanced Information Systems Engineering, pp. 440–455, 2018, doi: 10.1007/978-3-319-91563-0_27.

[4] L. Lyadova, A. Sukhov, and M. Nureev, "An Ontology-Based Approach to the Domain Specific Languages Design," In Proc. IEEE 15th International Conference on Application of Information and Communication Technologies (AICT2021). Baku, Azerbaijan: IEEE, 2021, pp. 1–6, doi: 10.1109/AICT52784.2021.9620493.

[5] G. Kulagin, I. Ermakov, and L. Lyadova, "Ontology-Based Development of Domain-Specific Languages via Customizing Base Language," in Proc. IEEE 16th International Conference on Application of Information and Communication Technologies (AICT2022), Washington, USA: IEEE, 2022, doi: 10.1109/AICT55583.2022.10013619.

[6] X. Qin, Y. Luo, N. Tang, and G. Li, "Making data visualization more efficient and effective: a survey," The VLDB Journal, vol. 29, no. 1, pp. 93–117, Nov. 2019, doi: 10.1007/s00778-019-00588-3.

[7] M. T. Cepero García, L. G. Montané-Jiménez, "Visualization to support decision-making in cities: advances, technology, challenges, and opportunities," in Proc of the 8th International Conference in Software Engineering Research and Innovation (CONISOFT), Chetumal, Mexico, 2020, pp. 198–207, doi: 10.1109/CONISOFT50191.2020.00037.

[8] G. Zelazny, The Say It With Charts Complete Toolkit, 1 ed. McGraw-Hill, 2006.

[9] A. Kirk, Data Visualization: A Successful Design Process. Packt Publishing, Limited, 2012.

[10] K. Smeltzer, M. Erwig, and R. Metoyer, "A transformational approach to data visualization," Sigplan Notices, Sep. 2014, doi: 10.1145/2658761.2658769.

[11] K. Smeltzer, M. Erwig, "A domain-specific language for exploratory data visualization," in GPCE 2018: Proc. of the 17th ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences, Nov. 2018, pp. 1–13, doi: 10.1145/3278122.3278138.

[12] C. Ledur, D. Griebler, I. Manssour, and L. G. Fernandes, "A High-Level DSL for Geospatial Visualizations with Multi-core Parallelism Support," in Proc. IEEE 41st Annual Computer Software and Applications Conference (COMPSAC), Turin, Italy, 2017, pp. 298–304, doi: 10.1109/COMPSAC.2017.18.

[13] V. Zayakin, L. Lyadova, and E. Rabchevskiy, "Design Patterns for a Knowledge-Driven Analytical Platform." Proceedings of the Institute for System Programming of the RAS (Proceedings of ISP RAS). vol. 34, no. 2, pp. 43–56, 2022, doi: 10.15514/ISPRAS-2022-34(2)-4.

[14] V.S. Zayakin, L.N. Lyadova, V.V. Lanin, E.B. Zamyatina, and E. Rabchevskiy, "An Ontology-Driven Approach to the Analytical Platform Development for Data-Intensive Domains," in Knowledge Discovery, Knowledge Engineering and Knowledge Management. IC3K 2021. Communications in Computer and Information Science, vol. 1718, pp. 129–149. Springer, Cham, doi: 10.1007/978-3-031-35924-8_8.

[15] L. Lyadova, I. Ermakov, V. Lanin, and K. Proskuryakov, "Approach to the Development of Ontology-Driven Language Toolkits Based on Metamodeling," in Proc. IEEE 17th International Conference on Application of Information and Communication Technologies (AICT2023). Baku, Azerbaijan: IEEE, 2023, pp. 1–6, doi: 10.1109/AICT59525.2023.10313152.

[16] N. Kahani, M. Bagherzadeh, and J. Cordy, "Survey and classification of model transformation tools," Software & Systems Modeling, vol. 18, pp. 2361–2397, 2019, doi: 10.1007/s10270-018-0665-6.

[17] J. Ding, J. Lu, G. Wang, J. Ma, D. Kiritsis, and Y. Yan, "Code Generation Approach Supporting Complex System Modeling based on Graph Pattern Matching," IFAC-PapersOnLine, vol. 55, Issue 10, 2022, pp. 3004-3009, https://doi.org/10.1016/j.ifacol.2022.10.189.

[18] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer, "Vega-Lite: A Grammar of Interactive Graphics," in IEEE Transactions on Visualization and Computer Graphics, vol. 23, no. 1, pp. 341–350, 2016, doi: 10.1109/TVCG.2016.2599030.