technologies

MDPI

Systematic Review

The Use of Domain-Specific Languages for Visual Analytics: A Systematic Literature Review

Alireza Khakpour [1], Ricardo Colomo-Palacios [1,2], Antonio Martini [3] and Mary Sánchez-Gordón [1,*]

[1] Department of Computer Science and Communication, Faculty of Computer Sciences, Engineering and Economics, Østfold University College, 1757 Halden, Norway
[2] Escuela Técnica Superior de Ingenieros Informáticos, Universidad Politécnica de Madrid, 28660 Madrid, Spain
[3] Department of Informatics, University of Oslo, 0373 Oslo, Norway
* Correspondence: mary.sanchez-gordon@hiof.no

Abstract: Visual Analytics (VA) is a multidisciplinary field that requires various skills including but not limited to data analytics, visualizations, and the corresponding domain knowledge. Recently, many studies proposed creating and using Domain-Specific Languages (DSLs) for VA in order to abstract complexities and assist designers in developing better VAs for different data domains. However, development methods and types of DSLs vary for different applications and objectives. In this study, we conducted a systematic literature review to overview DSL methods and their intended applications for VA systems. Moreover, the review outlines the benefits and limitations of each of these methods. The aim is to provide decision support for both the research and development communities to choose the most compatible approach for their application. We think the communication of this research delivers a broad figure of previous relevant research and assists with the transfer and adaptation of the results to other domains.

Keywords: visual analytics; domain-specific language

check for updates

Citation: Khakpour, A.; Colomo-Palacios, R.; Martini, A.; Sánchez-Gordón, M. The Use of Domain-Specific Languages for Visual Analytics: A Systematic Literature Review. *Technologies* **2023**, *11*, 37. https://doi.org/10.3390/technologies11020037

Academic Editor: Sikha Bagui

Received: 18 January 2023
Revised: 18 February 2023
Accepted: 27 February 2023
Published: 2 March 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

When we started to research Visual Analytics (VA) systems for supply chain decision-making support [1], we found that supply chain actors have enormous and heterogeneous data and different domain analytics problems. Generally, to deploy VA, they need to deal with three aspects [2]: (1) Identifying decisions that VA can support known as VA tasks, (2) data requirements, and (3) VA system development. These aspects formulate the primary constructs of a VA system.

However, the link between business goals and VA activities is not explicit in the domain [3]. In other words, enterprise visualization systems often do not support the entire pipeline of VA-driven decision-making [4], which identifies the business goals and decisions, decides on the visualization and analytical tasks, and chooses the suitable visualization design [5]. The primary issue is that developing this pipeline for various application areas is costly and time-consuming. On the other hand, one-size-fits-all VA systems are inefficient given the variability of users, data sources, and domain requirements.

From a software engineering perspective, the process of analyzing a certain domain is called domain analysis. In turn, domain analysis is important for implementing DSLs. Eventually, we came across several studies that used DSLs for VA applications [6]. DSLs can be used to program the pipeline for different domains, for example, in the supply chain, to promote enterprise systems interoperability [7]. However, not all domain experts and visualization users can code [8]. On the other hand, not all visualization experts who can develop customized visualization using programming understand the domain aspects [9]. DSL is a system engineering technique to raise the level of abstraction in a problem domain and automate the development of system artifacts [10], such as a VA dashboard or a machine learning pipeline. DSLs in the context of VA can specify the visualization tasks

for a given domain and streamline the goal-oriented VA development. Moreover, it can provide the foundation for developing a VA system that satisfies the different needs of different users by incorporating domain-specific aspects into the VA system.

Previously, several studies utilized various types of DSLs for creating VA in different domains for similar motivations to the ones we discussed above. Nevertheless, the types of DSLs used for VA applications are different, and each of them has different applicability, benefits, and limitations. Knowing these aspects can help make the best decision while designing a DSL for a particular domain. Indeed, there is no clear picture of the types of DSLs used for visualizations and their corresponding application domains to the best of our knowledge. Moreover, different choices of modeling methods and programming languages have not been identified. An overview of these aspects provides insight into the existing endeavors and suggests future directions for research and development that other domains can adapt.

Deciding on the type of DSL to be used for a specific application area depends on multiple factors. Developers need to first decide on the significance of these factors for their application to make the best decision. For instance, the level of programming skills of the target users [11] may be more important than providing high-level design abilities to users [12]. However, the choice of significant factors is not trivial. In fact, developers should consider trade-offs among various factors which requires an overview of them.

In this study, we conducted a Systematic Literature Review (SLR) to identify the previous uses of DSLs for VA purposes. In this regard, we extracted the type of DSLs, the application domains, version and the corresponding benefits and limitations. Based on the extracted data, we have discussed the suitability of each of the DSL types and reflected on the approaches taken. The main goal of this work is to find the benefits of using DSL for different VA applications and provide a way to make this technique replicable for other application areas.

In the rest of this article, we first introduce the background of information visualization and VA, followed by a classification of DSLs based on the related works. Next, we present the utilized methodology of our SLR, followed by reporting the findings and interpreting the results. Then, a discussion of the results addresses our research questions. Finally, a conclusion and future research directions wrap up our findings.

2. Background and Related Works

2.1. Information Visualization and Visual Analytics

Generally, Information visualization is to use visual encodings in order to understand data [13]. On the other hand, visual analytics is defined as "The science of analytical reasoning facilitated by interactive interfaces" [14]. As the goal of both fields is to make sense of data visually, we do not distinguish between them in our work and consider any study under any one of them in the same category.

Within the information visualization and VA communities, Sedlmair et al. [15] discussed the suitability of a design study for visualization research. In short, a design study is an attempt to develop design artifacts, such as visualization systems, using domain analysis [16]. Authors argue that when available data is limited and the visualization tasks are fuzzy (not clear), the focus of the study will be problem characterization and data abstraction, which are the initial steps toward the VA design proposal. Given that data availability and task clarity is the challenge of many real-world VA problems [15], DSLs can be used for data abstraction and task characterization.

2.2. Domain-Specific Languages

Previously, DSLs have been used in various domain engineering applications to provide domain problem abstraction as well as automation [17]. Software engineering has also used DSLs to create Software Product Lines (SPL). SPLs are used for the "mass customization" of software solutions [7], where the focus is to create a software product family based on commonalities and variation points.

Creating visualization as a product of SPL is performed to capitalize on shared assets, namely requirements (business decisions and goals) and assets (data). Later, designing variation or decision points are designed as visualization tactics, for instance, comparison and clustering. Vázquez-Ingelmo et al. [5] discussed using SPL concepts for visualization system design.

There are two main taxonomies for different types of DSL applications. Primarily, DSLs vary by description between domain-specific modeling language (DSML), domain-specific programming language (DSPL), or domain-specific specification language (DSSL). DSMLs are used within the Model-Driven Engineering (MDE) community and are created based on metamodels. Instead, DSPLs and DSSLs are used by the language engineering community and are generated based on grammar [18]. Furthermore, DSLPs and DSSLs differ in their execution [19]. DSSLs are based on formal languages and are not executed directly and are typically used for domain analysis and requirement engineering. Moreover, DSLs vary based on the language implementation, either text-based or graphical, and either embedded or external [17]. Previously, another SLR in [17] used the latter taxonomy to classify the use of DSLs. However, we believe the initial decision for choosing a DSL should be made on the type of DSL from the first taxonomy, therefore, we use the first taxonomy to classify the results of our SLR, although discussing other types in the discussion section. Figure 1 illustrates representative elements of the three different DSL types mentioned above. Some of the elements of the figure are adapted from [20].

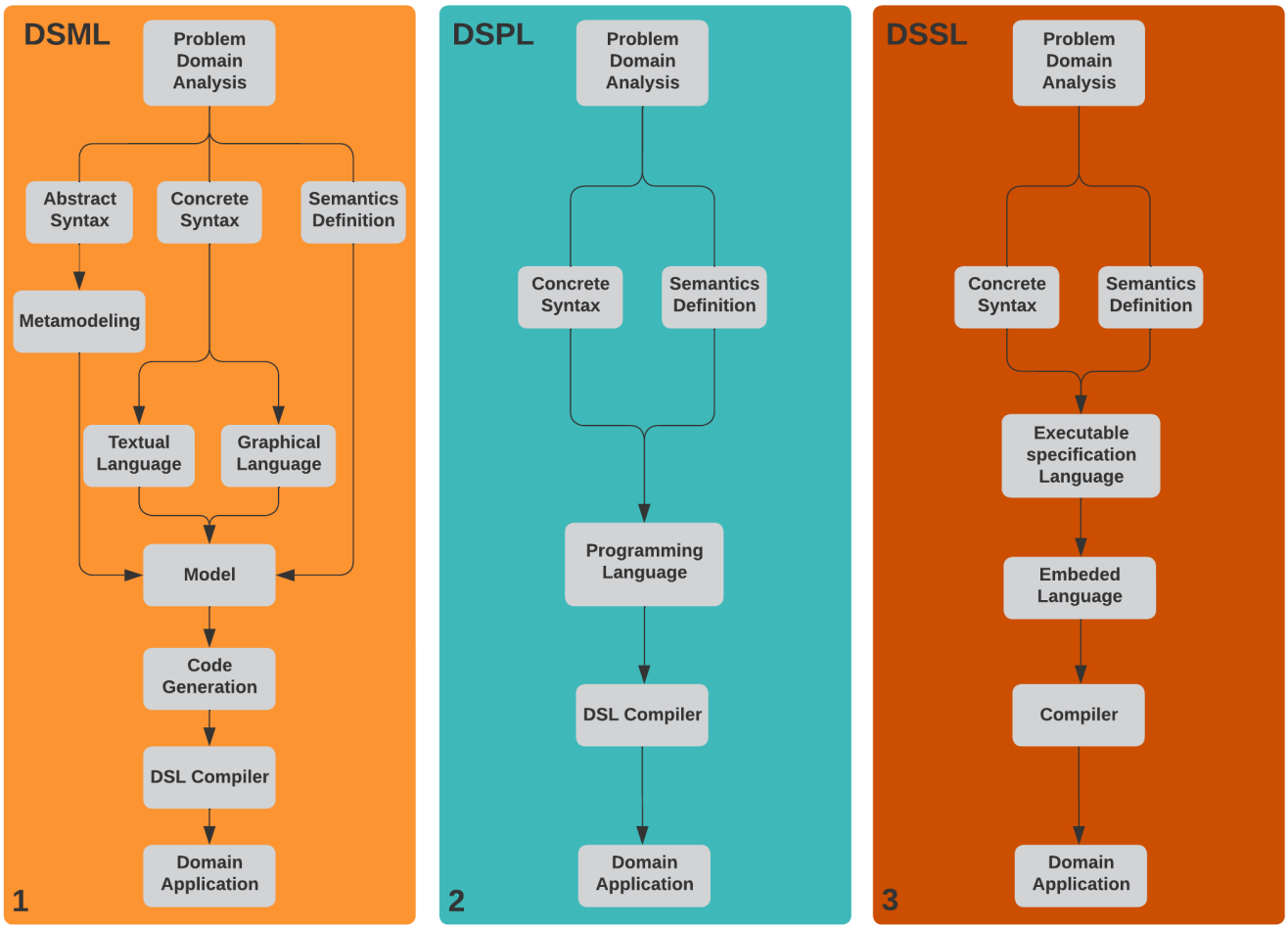


Figure 1. Different DSL types of elements.

Every DSL design starts with an analysis of the problem domain to understand and gather the domain concepts [19,20]. DSMLs ground these concepts to define the abstract syntax of the DSL by metamodeling. On the other hand, the concrete syntax is the mapping between the domain concepts and the DSL notations, a textual or graphical model in the case of DSMLs, a programming language in the case of DSPLs, and an executable specification language in the case of DSSLs. Semantic definition provides the meaning of DSL notations. Furthermore, DSMLs use a code generation module to translate the model into code and later give it to the DSL compiler for interpretation. DSPLs directly use a DSL compiler whereas DSSLs use an embedded language to interpret the specification for the compiler. Finally, the compiler translates the code into the required domain application.

2.3. Domain-Specific Languages and Visual Analytics

In recent years, VA has been used extensively for various applications and VA system developers used different techniques to develop the best suited system for a specific application. Consider the study presented in [21], where a VA system is developed for a software project management use case. The system is designed by the developers based on a designed architecture in java and provides certain capabilities to the end users. However, user requirements normally change over time, and it requires developers to spend a lot of time changing the architecture of the system. Another study presented in [22] proposed the use of MDE to support VA system design. Authors demonstrated that the use of DSLs can facilitate the customization of a developed VA system design using modeling and prototyping.

Another application domain that uses VA systems to exploit the potential of data is smart cities. Numerous studies developed VA systems for smart city applications, however, the visualization systems often come with a predefined set of capabilities developed based on requirements gathered before the system development [23,24]. This results in a lack of applicability of these systems in similar application domains. Authors in [25] proposed a DSL for the automatic creation of smart city dashboards that can be used for any city by abstracting and integrating the domain concepts into the system development process.

Although the study in [25] provided a VA development approach that is generalizable for different cities, it is still not fit for similar applications that use sensors in another context of the Internet of Things (IoT). For instance, industrial sensor analytics can benefit from a VA system, such as the one developed in [26]; however, it is still specific to the application domain. Instead, authors in [27] used DSL to not only provide domain variability but also provide simplicity and reusability.

2.4. Related Systematic Reviews

The topic of DSLs has been the focus of several previous systematic review studies. For instance, [28] conducted a Systematic Mapping Study focused on DSPLs and DSSLs, while [7] explored DSMLs by carrying out another systematic mapping. Both studies considered DSLs broadly to understand the field in general and identify research trends. These mapping studies merely focused on the DSL research fields and omitted the discussion of the benefits and limitations of various DSL types for different applications. Apart from that, authors in [28] specifically excluded DSMLs and visual languages. In [17], the researchers conducted a systematic review that is closest to our research objective, but it categorized the previous visualization studies in the areas of physics, chemistry, biology, and meteorology. It means these authors considered all types of visualization including simulation and 3D modeling. Therefore, their proposed taxonomy of DSLs reflects a broader area of visual computing. We performed an SLR focused on our research questions regarding the use of DSLs for the VA application area based on a different taxonomy of DSLs.

Another highly cited review on DSLs is the work presented in [29]. The authors identified other types of DSLs that are less recognized, such as framework-specific modeling language, architecture description language, and domain-specific aspect languages. We try to look for such specific types of DSLs among VA studies. In particular, we distinguish

between different types of DSLs concerning their use in VA systems. We also aim to describe the applicability of each type for VA systems in order to classify previous works and position new research activities.

3. Materials and Methods

3.1. Overview

The SLR carried out in this study is based on the Kitchenham guidelines [30] for systematic reviews for software engineering. The guideline includes mainly two stages of planning and conducting the review. During the planning stage, we will define the goals of the review and a search strategy containing the research questions, study selection criterion, search string, and choice of databases. Then, conducting the review involves searching databases with the search string, identifying, and selecting relevant studies, extracting data, and synthesizing the data to answer research questions. In what follows, we briefly explain each of the steps.

3.2. Planning the Review

3.2.1. Goals of the SLR

Kitchenham provided several reasons for conducting an SLR. One is to "summarize the empirical evidence concerning a treatment or technology." Similarly, the goal here is to overview the existing literature on different types of DSLs for VA purposes, and as a result, to identify their benefits and limitations. We believe this can act as a starting point to decide whether a DSL can be created and used for a particular VA application and, if so, how it can be done.

3.2.2. Search Strategy

The next step is to plan a search strategy, defining the research questions, formulating a search string, presenting inclusion and exclusion criteria, and choosing the research databases.

Research Questions

Concerning the scope of the study, we have formulated the following research questions:

1. How are DSLs being used for VA system design? By answering this question, one can understand how DSLs can be used to assist with VA system design and decide on the DSL type while designing a similar system.
2. What are the application domains that DSLs have been used for? Answering this question helps identify the goals of using DSLs by discussing their usage in different application areas.
3. What are the benefits and limitations of using DSLs for VA? Answering this question facilitates choosing the best DSL for a particular VA system design.

Search String

The study started with multiple rounds of pilot searches in different databases to examine the search results' relevancy and quantity with different search strings. Many VA studies simply refer to their work interchangeability as interactive, exploratory, or information visualization from the pilot search. However, all VA studies are about visualization and should include the term visualization. Therefore, we used visualization as the primary term of the search string to identify studies regarding VA. Eventually, we formulated the following search string using the Boolean "AND" operator to find studies that include both VA and DSL

"Visualization" AND "Domain-Specific Language".

Inclusion and Exclusion Criteria

- The study must use a DSL for visualization, and the focus should be on information visualization, not simulation, animation, 3D, or scientific visualization. This is to prevent collecting information from those areas that may not be applicable for information visualization. However, we have not excluded any study based on this criterion in the first round of title and abstract reading and it took place only during the full-text relevancy checking.
- The language of the study must be English.
- The paper must be peer-reviewed.
- The study must be accessible in full text.

Choice of Databases

In order to execute an automatic search, we need to decide on a specific number of databases. After consulting with domain experts and performing preliminary searches within various databases, we decided to select following five significant databases previously used by many other high-quality SLRs [31].

1. Association for Computing Machinery Digital Library (ACM DL)
2. Institute of Electrical and Electronics Engineers Xplore (IEEE Xplore)
3. Elsevier ScienceDirect (ScienceDirect)
4. Springer Link (SpringerLink)
5. Wiley Online Library (Wiley OL)

3.3. Conducting the Review

The execution of the study is explained in detail by Kitchenham [30]. The outline is as follows, followed by a brief description:

1. Locating the relevant research
2. Selecting the relevant studies
3. Verifying the quality of studies
4. Extracting data based on research questions
5. Synthesis of the results and reporting

Locating the relevant research was carried out by searching each database using the formulated search string. We then iteratively reviewed the results in two rounds. The first round included reading the title, abstract, and keywords and selecting the relevant studies. Next, we went through the full texts of all these articles and decided on their relevance to the research questions. To increase the depth of the study and include papers that are most relevant either in other publication channels or do not contain our search terms, we performed backward snowballing as well. Backward snowballing is to perform the same iterations over the reference lists of the identified articles. We repeated the previous process over the reference lists until we found no more studies. Finally, we removed short papers and duplicate reports. The remaining studies formed the final set of primary studies for the review. Figure 2 shows the research process and the number of studies in each round.

Moreover, we developed a data extraction form to collect the relevant data systematically and rigorously. The form contains the general information of each article, including title, authors, and publication venue, along with the questions to answer research questions and any relevant data that can assist with understanding the studies more deeply during the synthesis stage, e.g., the programming language that a DSL is embedded in.

After collecting the data, we provide a descriptive synthesis of the extracted data from all selected primary studies. Table 1 presents the relation between research questions and the corresponding types of information required for an SLR along with the numeric identifiers of the selected sources (see Table A1 for more details).

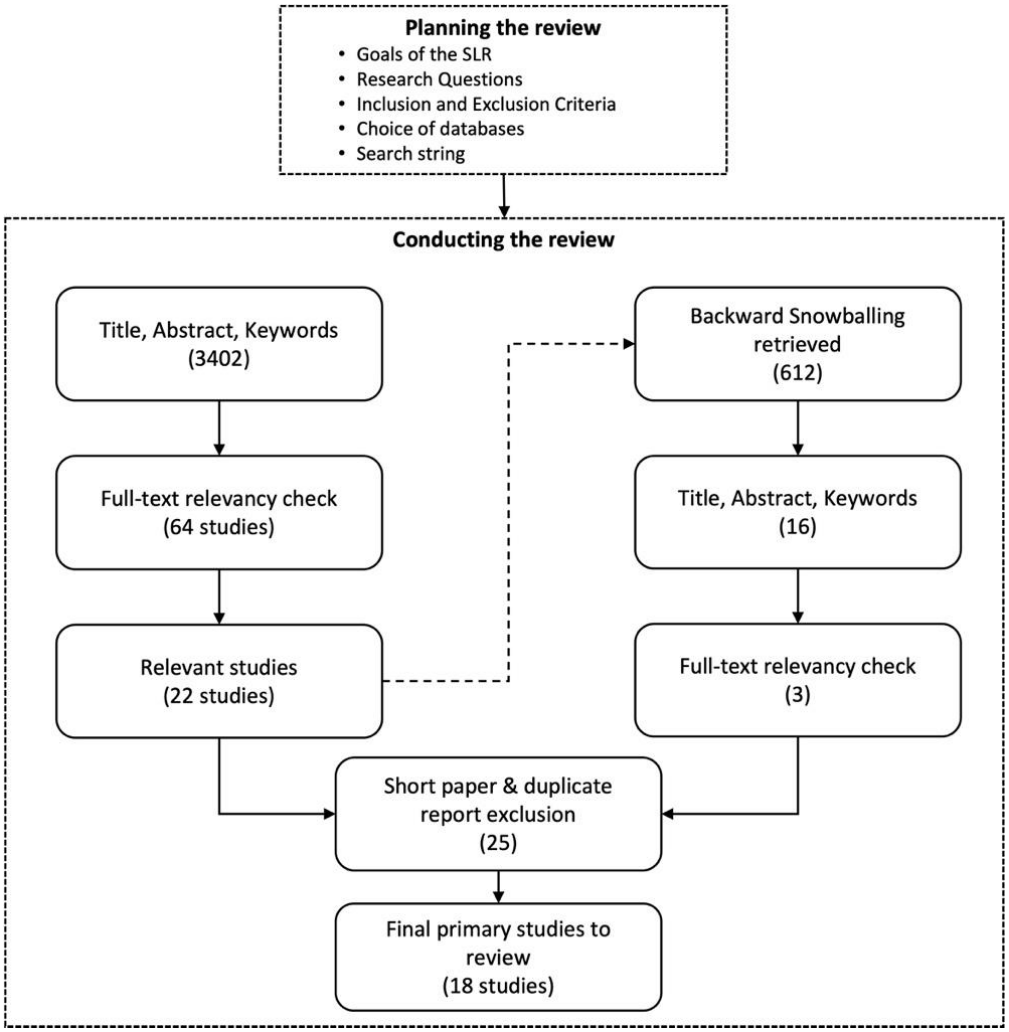


Figure 2. Research Process.

Table 1. Types of information extracted based on SLR Methodology.

Type of Information	Research Questions	Extracted Information	Source (ID#) *
Contribution of the study	RQ1	The type of DSLs	[1]–[18]
Context of the study	RQ2	The application domains DSL used for	[1]–[4], [7]–[10], [12], [13], [15], [16], [18]
The outcome of the study	RQ3	Benefits of the DSL method for VA system	[1]–[3], [6], [8]–[11], [15], [17]
Limitations of the study	RQ3	Limitation of the DSL method for the VA system	[10]

* ID#—(Numeric identifier see Table A1 for more details).

Appendix A presents the complete list of primary studies and a summary of the data extraction form, excluding basic information such as title, authors, and year. Instead, that information is provided using references.

3.4. Study Quality Assurance

Generally, SLRs need to verify the quality of the studies to increase the reliability of the results. However, given that there is no consensus on the definition of quality [30], it is not easy to conduct quality checks. Additionally, the focus of our study is the use of DSLs for visualization purposes, which is mainly to develop and propose a system. Therefore, the quality of these studies highly depends on the development and validation methods, and

it is precarious to validate them without considering their approach. In other words, the suggested quality concepts to consider, namely bias, internal validity, and external validity, are not the best fit for these types of studies. Therefore, to verify the quality of the studies included in the SLR, we first make sure that they are from peer-reviewed journals, and we verify this during the data selection stage of the review.

Moreover, we assessed the quality level of each of the final primary studies based on the approach provided by the Kitchenham guideline [30]. Based on this, a set of questions should be answered regarding three aspects to ensure the quality of the study, including (1) selection and measurement bias, (2) validity, and (3) generalizability. Table 2 presents the set of questions answered for each of the aspects. The primary studies selected for review passed these questions.

Table 2. Study quality assessment questions.

Quality Aspect	Question
Selection and measurement bias	Are the inferences based on the subject of the study?
Validity	Is the methodology of the study scientific and well defined?
Generalizability	Whether the study is validated with a suitable test case?

3.5. Limitations and Threats to Validity

Every SLR may be subject to various threats to its validity. Therefore, this study reports the possible threats based on the proposed four validities in [32], including construct, internal, external, and conclusion validities.

3.5.1. Construct Validity

In SLRs, construct validity is mainly about the extent to which the selection of primary studies reflects the subject under study. Mitigating the threats to construct validity largely depends on the planning of the review, when the research questions, search string, publication channels, and inclusion and exclusion criteria are defined. We formulated the research questions carefully only to report the existence of DSLs regarding the VA design, and we prevented any biased questions regarding the outcome. Additionally, our search string only contained general terms of "visualization" and "domain-specific language" to cover as many related primary studies as possible. In other words, although the main subject of the study is "visual analytics," we have used a broader term to include a bigger sample size, particularly in older studies that do not recognize information visualization as VA.

However, we recognize that there may be studies having DSLs without having this term specifically. This is mainly because some of the programming languages for visualizations are not recognized as domain-specific, which is also the case with older studies. In order to alleviate this threat, we tried to extend our search with a backward snowballing technique to identify other previous relevant studies that our selection of primary studies may cite. We also decided not to consider any publication period for our search to include older papers not having the DSL term but having a methodology that replicates a DSL, for instance proposing a declarative language for visualization which is actually a type of DSPL. Another threat is missing articles based on inclusion and exclusion criteria in the first round by reading only the title, abstract, and keywords. To reduce this possibility, if we were unsure about the scope of a study, we also screened first the conclusion of the study. Finally, we included the paper for later full-text verification if we were still uncertain.

3.5.2. Internal Validity

The internal validity of SLRs is related to the possible systematic errors while extracting data from primary studies. Although literature proposed various classifications of DSLs, we classified our studies based on three broad types of DSLs and only on one factor: the development method (modeling, programming, and specification). Hence, we increased data extraction consistency and reduced the possibility of ambiguous interpretations. We

also included research questions in our data extraction form. Additionally, we collected the relevant data while reading the full text of the studies only to reduce the chance of internal errors. Furthermore, we excluded duplicate reports to decrease the chance of biasing the results by the same authors. Duplicate reports are different studies by the same authors, reporting the same study in different venues or iterations, such as conferences and journals, and most often one is an extended version of another, in which case we chose the most recent publication, for example, removing [33] which is similar to [34].

3.5.3. External Validity

The external validity is related to the generalizability extent of the study. In this study, we discussed various applications and use cases of DSLs for VA. Moreover, we identified the benefits and limitations of different types of DSLs with respect to their context. However, since the VA application is very broad with its own requirements, it is not known exactly to what extent these benefits and limitations are applicable in all other contexts of using DSL for VA. In this regard, in order to mitigate the threat to external validity, we tried to collect as much information as possible regarding an application area that a DSL was used for and present the benefits and limitations accordingly. The focus of this study is the use of DSLs only for information visualization and VA. Consequently, we excluded studies regarding other areas of visualization, namely 3D modeling, simulation, animation, and scientific visualization. Therefore, the results of this study may not be applicable to those areas, and the results are explicitly specific to VA for exploratory analysis. However, given that the classification of reported DSLs is considered broadly, we can assume that the reported benefits and limitations may be useful to other types of visualizations as well.

On the other hand, the exclusion of visualization areas may result in missing some of the related studies to our scope due to missing interpretation of the type of visualization. We tried to prevent this by not excluding any study based on visualization type during the first round of title and abstract reading and only during the full-text relevancy checking.

3.5.4. Conclusion Validity

Conclusion validity deals with the accurate inference of the conclusions based on the study. In order to minimize threats, we discussed the results concerning the research questions and only based on the evidence presented in the result section to prevent any biases. The research questions and the corresponding data used to answer them were discussed between the authors to mitigate any biases by a single author. Additionally, we reported the number of studies accurately in every round of the study to facilitate validation of the results by other researchers. Other threats to conclusion validity are mitigated mainly during the "planning the review" stage of the study, which is already explained regarding the construct validity.

3.5.5. This SLR vs. Others

In this SLR, we focused on using DSLs for VA systems. Previous SLRs, [7,17,28,29] are more general to the application of DSLs and have not recognized their benefits for VA designs. Therefore, this SLR identifies the benefits and challenges of different DSL types for various VA system designs. Compared to other SLRs, we also provided a clear distinction between the three types of DSLs and their benefits and limitations that facilitate the type selection for future research and developments. However, the focus of the SLR may result in losing some other benefits of DSLs that have not yet been explored by previous works and are adaptable to VA system design.

4. Results

4.1. Demographics

Table 3 presents the number of papers retrieved from each of the databases in different rounds of the study. During the initial search, we identified 3402 studies across the five databases. We then identified 64 studies in the first round. Going through the full texts of all these articles and deciding on relevancy, we selected 22 studies in this round. The last 22 articles contain 552 references as the initial studies for snowballing. Next, we identified 16 articles from reviewing titles and abstracts, later reduced to 3. Further iterations in the reference list of these four articles did not find any more articles. Therefore, a total of 25 articles were identified. Next, we excluded short papers and duplicate reports, resulting in a final set of 18 studies.

Table 3. Number of studies retrieved from the selected databases.

Database \ Action	1st Round (Title and Abstract Relevancy)	2nd Round (Full-Text Relevancy)	3rd Round (Backward-Snowballing)	4th Round (Short and Duplicate Report Exclusion)	Final Review
ACM DL	573	7	6	7	5
IEEEXplore	997	34	12	14	11
ScienceDirect	989	6	2	2	1
SpringerLink	632	14	1	1	0
Wiley OL	211	3	1	1	1
Total	3402	64	22	25	18

Figure 3 presents the dispersion of the studies in different years. The figure shows that the first attempt to use DSL for visualization started in 2009. Later, there were only two published studies until 2014. Therefore, this research area is relatively new, and an overview can shed light on previous efforts to be useful for future research.

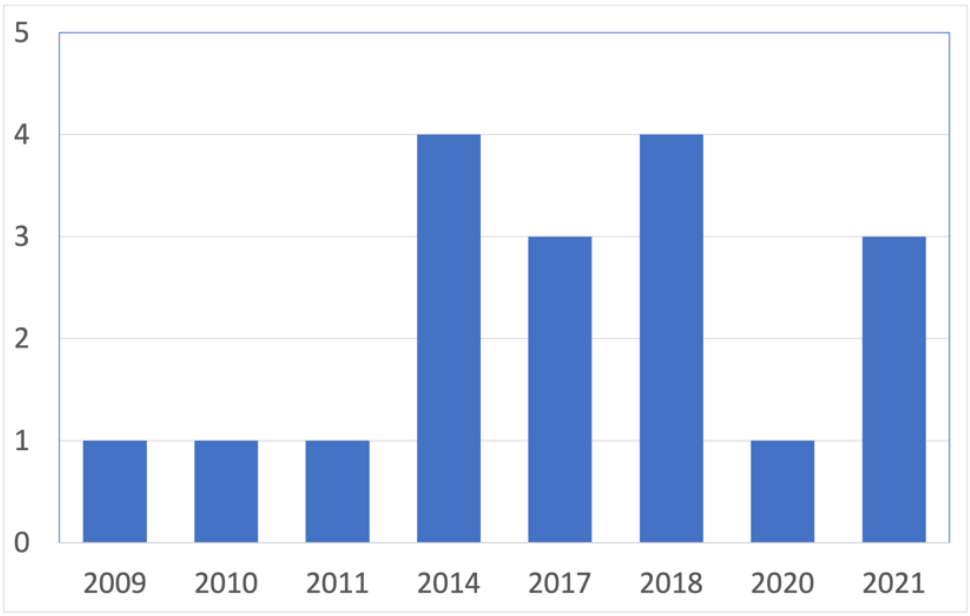


Figure 3. Number of studies in different years.

Figure 4 presents the number of selected studies grouped by the three different types of DSLs that we identified in the literature (i.e., DSML, DSPL, AND DSSL; for details see Section 2.2). There are an equal number of five DSMLs and DSSLs and fewer than eight DSPL studies.

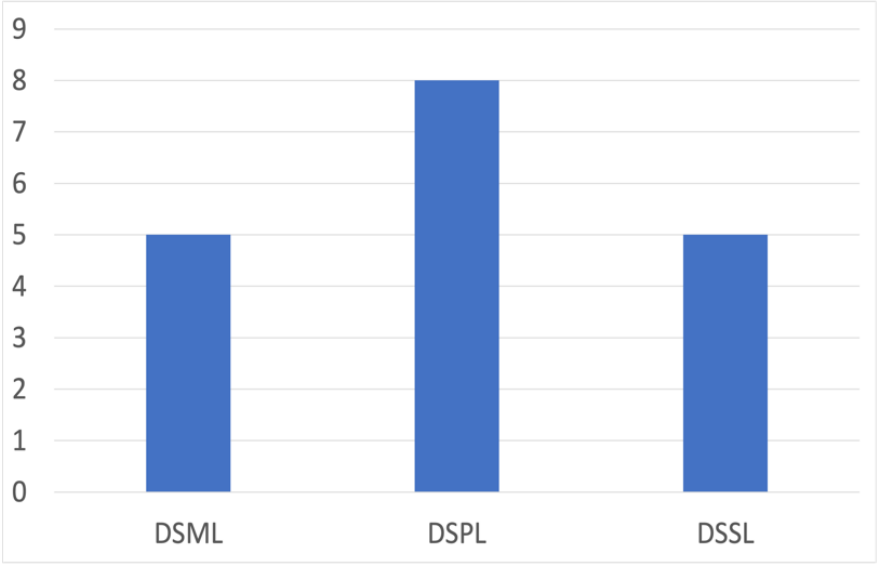


Figure 4. Number of studies in each DSL type.

Furthermore, to provide the practitioners such as project managers and decision-makers in projects, the ability to choose the right DSL for their needs, we also identified the version of the DSL proposed in each of the selected studies. We categorized the version of the DSL based on the software development practices as follows:

- Proof of Concept (PoC): PoC is more of a research prototype that aims at demonstrating the feasibility of an idea and verifying certain concepts or theories.
- Prototype: Prototype is a way of presenting the design of a product to stakeholders and prospective users in order to showcase the functionality.
- Minimum Viable Product (MVP): MVP is a releasable version of a product that contains the most significant function and features to exhibit usability.
- Open-source full-fledged product: Open-source full-fledged product is a freely accessible finished product that is ready to be used.

In the selected study, we did not find any proprietary DSL; therefore, we omit this definition. Figure 5 shows the distribution of the proposed DSLs in each of these categories (see details in Appendix A).

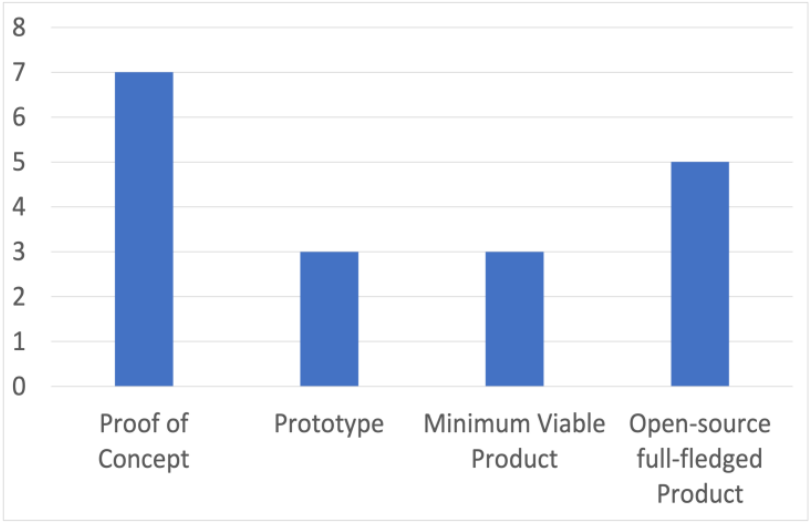


Figure 5. Number of studies in each version.

4.2. Taxonomy of DSL Types

4.2.1. Domain-Specific Modeling Language (DSML)

The first types of DSLs are DSMLs from MDE. Authors in [22] used a modeling approach for visualization system design. They attempted to address the challenges regarding visualization systems: (1) to support the visualization system design and (2) to choose suitable visual encoding from various visualization libraries. To do so, they have used feature modeling from the SPL paradigm to characterize different visualization widgets. Then, they provided a metamodel of a visualization system based on a visualization task taxonomy. They also proposed a prototype DSL using XText over the familiar language framework.

In [35] authors included domain-specific modeling and code generation aspects as domain engineering and application engineering stages in the visualization personalization pipeline. Domain-specific modeling uses domain analysis to study the domain, and application engineering uses requirement engineering to create the final visualization. They used a code template as a code generator. In their PoC, the domain concepts are extracted using feature models and fed into the template using XML configuration files.

Authors in [25] used MDE to generate a MVP version of a graphical DSL to automate dashboard generation for smart cities. The proposal is based on the argument that the similarity of dashboard requirements for different cities justifies the application of a dashboard generation framework. Authors used meta-modeling to abstract the concepts of smart city dashboards for the respective domain experts. The metamodel is then used to create the DSL.

In [27] authors aimed to address two issues of making sensor data visualization dashboards with a PoC. First, MDE is used to provide domain-specific dashboards by metamodeling, and second, SPL is used to provide widget variability by feature models. The metamodel is based on three aspects, data (what am I visualizing?), visualization type (how do I visualize?), and spatial and temporal layout (where and when do I visualize it?). The authors assumed that data specification reveals the visualization task. However, identifying visualization tasks also depends on the goals of the user. This usually is distinguished as data-oriented and goal-oriented visualizations.

Authors in [36] proposed the use of a DSML for what-if analysis by simulation of different scenarios. The simulation is presented through a PoC of both simulation software and information visualization encodings.

4.2.2. Domain-Specific Programming Language (DSPL)

As mentioned earlier, DSPLs are the main types of DSLs used. In this sense, authors in [12] argued that the support for exploratory visualization is lacking in current systems. To address this, they proposed a technique called variational visualizations. Variational visualization is defined as a systematic way of structuring various presentations in visualization design. In this regard, they have modeled variations using choice calculus, a mathematical model, and developed a PoC DSL embedded in Purescript.

Authors in [37] explored creating new visualizations by incrementally transforming the first visualization. They have provided a PoC for a Haskell-embedded DSL and demonstrated the use of functional programming idioms and type classes to create and transform visual encodings.

In [38] authors argued the need for a graphical tool that mitigates the gap between inflexible abstract high-level visualization systems, e.g., Tableau, and complex low-level graphical systems, e.g., processing. As a result, they created Protovis, a MVP version of an embedded DSL that creates visualizations implemented in JavaScript. Protovis provides information visualization domain-specific visual encodings along with the capability of customizing visuals by low-level programming. Later, authors extended Protovis in [39] by implementing it in Java. The goal was to provide portability to visualization specifications within various platforms. It is claimed that a statically typed language, like Java, has the advantage of portability across platforms and "format agnostic processing" over a

dynamically typed language like JavaScript. Format agnostic processing is adapting a visualization specification to fit new data rather than the challenging task of reconfiguring the data for the visualization system.

Authors in [40] provided an open-source full-fledged DSPL to create web-based interactive visualizations. Authors used JSON format to declare building blocks of visualizations known as visual encodings as components. Components are declared along with their interactions and styles.

In [41] authors proposed a PoC to use DSL for a visualization source code data set creation. The authors attempted to create a dataset for sketches of visualization to train a deep learning network for converting human visualization sketches to visualization codes. The DSL specification provides a simple syntax that can be learned easily by the deep learning network. They then suggested using a code generator, such as D3, to create visualization outputs.

Authors in [42] provided a prototype of a model-based embedded DSL on JavaScript for narrative visualizations for storytelling. The model is developed based on the domain requirement constructs. Scenes are narrative structures, parameters to manage visualizations, annotations to highlight information, and triggers to provide interactions. The proposed DSL works with existing visualization languages like D3 and adds a storytelling capability to a visualization system.

D3 from [43] is an open-source full-fledge Java Scripts library that can be seen and used as a DSPL for visualizations. D3 includes transformations that enable adding animation and interaction to visualizations. The visualization community widely uses D3 to create dynamic visualizations.

4.2.3. Domain-Specific Specification Language (DSSL)

Finally, DSSL has also been the approach of some of the studies. In this regard, ref. [44] focused on the analytic factor of the visualization by discussing multiverse analysis. Multiverse is to consider different analytical decisions during the VA pipeline parallelly and interpret the results collectively, resulting in robustness and transparency. Authors argue that different analytic decisions, for instance, choosing control variables for target prediction, do not always produce robust outputs. An open-source full-fledge DSL is proposed to annotate analysis source codes and create specifications for later visualization designs.

In [11] authors developed a graphical DSL prototype based on Interaction Flow Modeling Language (IFML) for enterprise software interoperability. The graphical DSL is considered less challenging to learn for non-expert users. The advantage of using IFML is its recognition as a standard by Object Management Group (OMG), also the possibility to be extended by the specification defined in Unified Modeling Language (UML).

Authors in [34] focused on the geospatial data visualizations and created a DSSL to generate visualizations automatically. The authors created a PoC of an external DSL that uses Google API to generate visualizations based on the DSL specifications.

In [45] authors created an internal DSL based on the Pharo programming language for visualizing software dependencies using graphs. The authors addressed the domain problem using one specific visualization type. The nodes of the graphs are the software elements, namely classes or packages, and edges are dependencies. The developed open-source full-fledge DSL can be seen as a specification language, given that the visualizations are generated using the Roassal visualization engine.

Authors in [46] extended the open-source full-fledge Vega-Lite, a grammar-based visualization specification language, by providing interactive features to the language. Generally, exploratory visualizations require a high level of interaction with the visual encodings, a feature that many previous visualization grammars lack, for example, Protovis and D3 [38,43].

5. Descriptive Synthesis and Discussion

The primary objective of this SLR was to collate and summarize previous attempts at using DSLs for VA system design and to Synthesize the findings. As a result, we assess the robustness and transferability of creating and using DSLs for VA systems by recognizing the benefits and limitations. In what follows, we provide our findings concerning the research questions.

5.1. Types of DSLs (RQ1)

In general, three types of DSLs have been identified in the literature. DSPLs, DSSLs, and DSMLs. DSPLs are programming languages that are either embedded in another language or act as standalone languages known as externals [12,37–43]. DSPLs are either textual by writing the actual syntax of the language [43] or graphical by providing a graphical user interface to the user to create a VA design [38]. DSPLs can help increase the portability of VA design systems by embedding them in languages that are widely used by various platforms [39]. DSSLs provide the same advantages, plus raising the level of abstraction by keeping the users away from visualization design programming [11,34,44–46]. Users generally focus on what to design instead of how to design. This promotes a goal-oriented VA and covers the gap between VA activities and requirements.

Furthermore, this gap is even more addressed with the DSMLs as the last type [22,25,27,35,36]. DSMLs' primary focus is to use modeling techniques, such as metamodeling, to maintain a high level of abstraction and, at the same time, to stay away from coding by using automatic code generation engines. Although the benefits and limitations of each of the types are considerable, this distinction and the reason to choose any one is not usually specified by authors.

5.2. Goals of Using DSLs (RQ2)

In this regard, we classify the primary usage of DSLs for two main goals. First, to support the VA system design, and second, to provide a way to decide on a suitable visualization design for a particular purpose. Within the identified primary studies, nine different DSLs are designed for general-purpose VA systems towards achieving the first goal. These DSLs fulfill various VA domain-specific requirements, such as supporting exploratory visualizations by providing interactivity [43]. Additionally, DSLs have been used to provide what-if analysis capability by different means, for instance, simulation of scenarios or storytelling support [42].

The main benefit of such DSLs is to provide flexibility and simplicity [39,40]. DSLs provide the ability to customize VA designs, given the scalability of DSLs in general. This can be achieved either by adding new features to the language or extending the metamodels. On the other hand, DSLs are more understandable for domain experts who are frequently alien to VA programming and instead familiar with domain concepts [11]. Although, these benefits are often possible from different types of DSLs. For instance, DSPLs [37] and DSSLs [46] provide more flexibility and less simplicity than DSMLs [25]. On the other hand, programming languages may be challenging to learn for domain experts, and graphical DSLs showed to be of considerable assistance [25,38].

The latter classification of DSLs is concerned with using VA systems for specific applications including smart cities [25], geospatial visualization [34], or sensor data [27]. The application domain concepts are used to support the VA design. The common aspect within these domains is the presence of similar requirements for VA designs across each domain. For example, all smart city visualization systems have standard requirements that can be created with a DSL [25].

In this regard, various techniques have been used to assist with modeling and language design. For instance, feature modeling from SPL identifies domain concepts as first-class citizens of the DSL [35]. Furthermore, taxonomies have been used to create catalogs for decision points determining the variations in design, being analytic task taxonomy or visual encodings taxonomy [22]. Apart from that, studies used different domain elements as

underlying concepts to their DSLs. The choice of domain elements is highly dependent on the requirements that the VA system seeks to address. In this context, authors in [25] used dashboard, visualizer, position, social media, map, charts, and data source as their domain concepts. Authors in [27] used data, visualization type, and spatial and temporal layout as the foundation for their DSL. Other domain elements used for various applications are listed in Table A1.

5.3. Benefits and Limitations of Different DSLs (RQ3)

In general, DSLs provide higher abstraction into domain concepts and reduce the complexities. In terms of VA design and information visualization, many complexities can be overcome with the help of DSLs, such as variable user supports [22], domain-specific customization management [35], and handling interactivity [40]. However, different types of DSLs provide different benefits that can be taken into account while deciding on a DSL design.

In this regard, DSMLs are highly useful when the main focus is to conceptualize and integrate the domain aspects into the visualization design [25]. Incorporating domain concepts into visualization designs provides the ability to users to focus on the visualization design for their specific decision-making process and facilitate the diagnosis process [22]. Moreover, it provides the ability to non-programmers who only have the domain knowledge to generate visualizations.

On the other hand, DSPLs provide high-level design abilities for more concise VA design, both in terms of visual design and interactivity [12,38]. Adding to that, having a programming language provides the ability to let the users focus on visual design specific to their application domain while allowing language developers to optimize processing. This separation of specification from execution simplifies development, enables optimization, and retargeting over different platforms [39]. Furthermore, language design can be tailored for different purposes, for example, faster learning for ML model training in the case of an automatic visualization design as presented in [41]. Although a DSPL host language can be a known language by some of the users which can facilitate visualization generation for them, learning a new programming language for others may be cumbersome.

On contrary, DSSLs are easy to learn and use without extensive programming [11], given they are supposedly designed in a small and consistent manner [45]. An important benefit of this simplicity is that the outcome of the VA design can be assessed before making inferences. Moreover, writing design specifications provides the ability to generate and reuse some parts of the design, which in turn reduces future development efforts. Additionally, data pre-processing and visualization implementation can be omitted by integrating them into design specification which reduces development time.

Conclusively, based on the identified benefits of using each of the DSL types we have created a decision diagram to make the best design decision for system developers. The decision diagram presented in Figure 6 uses different decision criteria to provide the best DSL design decision.

- **Criterion A** guides the decision-maker to choose the suitable DSL based on their purpose. Practitioners can decide on the type of DSL based on the main purpose that the DSL aims to serve. For example, if the purpose of creating a DSL is to provide high-level design ability to the VA developers, a DSPL would be the best fit.
- **Criterion B** is when the programming skills level of the VA developers is the main concern of the decision-maker. For instance, if the end-user is a solely a domain expert without any programming knowledge, a DSML where users interact with the language by means of a textual or graphical interface would be the best choice.
- **Criterion C** is when integrating a domain aspect into the VA design is the main motivation behind the creation of a DSL. For example, if one wants to provide the ability to customize the VA system based on the domain needs, the suitable DSL type would be DSML.

- **Criterion D** is when the user support requirement is the main motivation for creating the DSL, such as the case where the user needs design specification reusability. That is, when the VA developer wants to reuse a design specification repeatedly for different VA activities where DSSL is the best choice, whereas DSML supports the users' requirement for handling diagnosis VA tasks.

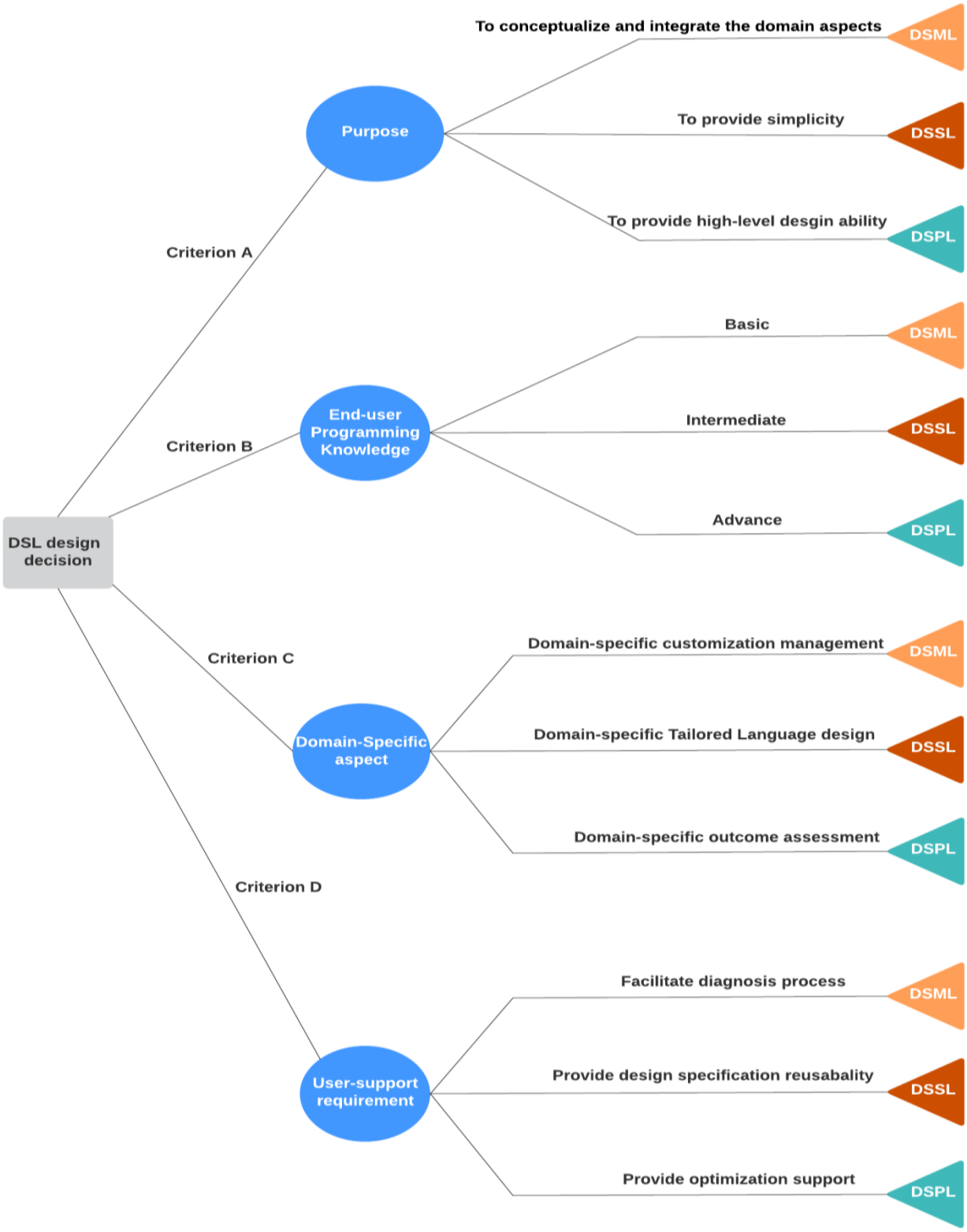


Figure 6. DSL design decision diagram.

6. Conclusions and Future Works

Several DSLs have been created and used for VA designs in recent years and they have shown to be of prominent benefits. Various DSL methods exist, and each of them satisfies different needs with multiple benefits and limitations. However, to the best of our knowledge, there is no clear overview of these DSLs and how they can be transferred and generalized for other applications with respect to those benefits and limitations. Therefore, in this study, we identified the related research and synthesized the findings in order to provide such an overview.

The results include different types of DSLs and their applications for VA design, their benefits, and their limitations. Ultimately, a decision diagram is provided to support choosing the best DSL type based on different criteria. The results of the review are categorized based on three types of DSL methods, namely DSML, DSPL, and DSSL. It is argued that the difference between these methods mainly relies on the purpose they are going to serve. Three main purposes for employing these DSL methods are (1) conceptualizing and integrating the domain concepts into the VA activity best served by DSML, (2) simplifying the design and implementation of VA system, provided by DSSL, and (3) providing high level design abilities for VA, for instance, interactivity, enabled by DSPL. On the other hand, the level of end user programming skills, the most important domain aspect requirement, and the user-support requirement are other aspects that play an important role in deciding which DSL method should be considered.

However, this study focused only on applying DSLs for VA design. For future work, it would be interesting to conduct another SLR, including other application domains trying to identify if there are other benefits not present in these studies but transferable to the VA design. At the same time, the application of DSL for VA in other domains other than those presented in this study can identify more benefits and limitations and check if those discussed here will remain the same. In this regard, for future work, we want to design a DSML for supply chain VA. We believe the identified benefits can be highly advantageous for supply chain VA systems and further advance this field of research.

Author Contributions: Conceptualization, A.K. and R.C.-P.; methodology, A.K. and M.S.-G.; validation, A.K., R.C.-P., M.S.-G. and A.M.; investigation, A.K.; resources, R.C.-P.; data curation, A.K.; writing—original draft preparation, A.K.; writing—review and editing, A.K., M.S.-G., R.C.-P. and A.M.; visualization, A.K.; supervision, M.S.-G., R.C.-P. and A.M.; project administration, R.C.-P.; funding acquisition, R.C.-P. All authors have read and agreed to the published version of the manuscript.

Funding: This work is part of the DigiMat project partially funded by the Research Council of Norway (RCN) with Project Number: 296686.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are openly available as a replication package in Zenodo at [https://doi.org/10.5281/zenodo.7628571] accessed on 17 January 2023.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Table A1. Primary studies.

ID#	Ref. No.	RQ1: DSL Type	RQ2: DSL Domain (Target)	RQ2: Domain Elements (Concepts)	Modeling or Language	Version	Benefits
1	[22]	DSML	Software project management	Widgets: concerns, name, library	Feature model from SPL using Familiar, Metamodel, Xtext	Prototype	Abstraction of domain concepts and variable user support.
2	[35]	DSML	Analyze employability in academia through dashboards	Functions, layouts, data	Feature model from SPL and Python processes XML configuration file on jinja2 code templates	Proof of Concept	Domain-specific customization management made possible.
3	[25]	DSML	Smart city visualization dashboard generation	Dashboard, visualizer, position, social media, map, charts, data source.	EMF (Ecore) implemented metamodel based on OMG MOF and Graphical DSL using Sirius	Minimum Viable Product	Closer language to the end user, abstraction of domain concepts.
4	[27]	DSML	Sensor data visualization dashboard	Data (what am I visualizing?), visualization type (how do I visualize?), spatial and temporal layout (where and when do I visualize it?)	DSML, metamodel, and feature model, Familiar code to model AmChart Library	Proof of Concept	No implementation knowledge is required; no library bound, high level of abstraction, and reusability of visualization.
5	[36]	DSML	What-if analysis for the bus transit system	Scenario-based simulation	Model-Integrated computing, TextX, SUMO Simulator	Proof of Concept	Provides the ability to generate what-if scenarios.
6	[12]	DSPL	Variational visualization	Composition of visualization into larger visualizations Allowing visualization to be transformed between Cartesian and polar coordinate systems	Choice calculus embedded in Purescript	Proof of Concept	Provides high-level design abilities for visualization such as an ability to group functionality into different layers.

Table A1. *Cont.*

ID#	Ref. No.	RQ1: DSL Type	RQ2: DSL Domain (Target)	RQ2: Domain Elements (Concepts)	Modeling or Language	Version	Benefits
7	[37]	DSPL	Incremental visualization using transformations	Layered abstraction: mark, primitive, visual parameter; Chart composition and layout, Functor, Monad	Functional programming idioms and type classes, Haskel-embedded	Proof of Concept	Host language can address some issues such as data handling and rendering, use of a known language by the corresponding user can facilitate visualization generation.
8	[38]	DSPL	Interactive domain-specific visualization	Marks, panels, built-in mark types, data transformations	JavaScript	Minimum Viable Product	Enable concise visualization definition
9	[39]	DSPL	Interactive domain-specific visualization	Visualization specification, interaction, animation	Java embedded	Minimum Viable Product	Separating specification from execution, simplifying development, enabling unobtrusive optimization, and supporting retargeting across platforms.
10	[40]	DSPL	Chart visualization domain-specific	Component, styles, data, and interactions	JSON	Open-source full-fledged product	Allow users to focus on visual design. Ability to design rich built-in interactions.
11	[41]	DSPL	Generate source code for data visualizations randomly.	-	XML-based syntax, D3 code generator for visualization	Proof of Concept	Language design can be tailored for different purposes, such as faster learning for ML models.
12	[42]	DSPL	Narrative visualization, storytelling	State-based scenes, visualization parameters, dynamic graphical and textual annotations, and interaction triggers	embedded in JavaScript syntax	Prototype	Enables storytelling without extensive programming, a decoupled implementation of narrative and visualization, and facilitates visualization generation.

Table A1. *Cont.*

ID#	Ref. No.	RQ1: DSL Type	RQ2: DSL Domain (Target)	RQ2: Domain Elements (Concepts)	Modeling or Language	Version	Benefits
13	[43]	DSPL	Visualization for web	Selection, Operators, Data joins, enter and exit, transitions, event handlers, modules	A JavaScript Library	Open-source full-fledged product	Allow language users to focus on the specifics of their application domain while freeing language developers to optimize processing.
14	[44]	DSSL	Analytic decisions, Multiverse analysis	Source code: Analysis code, placeholder variables: decision points, blocks: multiple snippets of code, constraints: dependencies between decision points, and code graphs: execution order of code blocks by DAG		Open-source full-fledged product	Ability to assess the output of visualization before making inferences, the sharing portion of the analysis code can be specified only once and be reusable.
15	[11]	DSSL	Enterprise system inter-operability	Layout, Interactions, Seman-tic/relationship visualization, hierarchical navigation	IFML implemented in DoME and Agile visualization engine	Prototype	Easy to use, highly abstract, no code writing makes it easier for non-programmers.
16	[34]	DSSL	Geovisualization	Filter, classification, data format, and visualization specification	External DSL using Google API	Proof of Concept	Reduces efforts of data pre-processing and visualization implementation, abstracts complexities, and less development time.
17	[45]	DSSL	Visualizing software de-pendencies using graphs	Nodes as software elements, edges as dependencies, layouts for specific situations of dependencies, global rules to avoid duplications	Pharo and Roassal	Open-source full-fledged product	Small and consistent language, produced visualization may be seen in either the host language framework or a browser.

Table A1. *Cont.*

ID#	Ref. No.	RQ1: DSL Type	RQ2: DSL Domain (Target)	RQ2: Domain Elements (Concepts)	Modeling or Language	Version	Benefits
18	[46]	DSSL	Interactive visualization	Selection, transformation	Vega JSON	Open-source full-fledged product	Enable search and inference over the space of visualizations, facilitate programmatic generation and retargeting of interactive visualizations

References

1. Khakpour, A.; Colomo-Palacios, R.; Martini, A. Visual Analytics for Decision Support: A Supply Chain Perspective. *IEEE Access* **2021**, *9*, 81326–81344. [CrossRef]
2. Khakpour, A. Information Sharing for Customized Dynamic Visual Analytics: A Framework. *CEUR Workshop Proc.* **2021**, *2906*, 89–97.
3. Nalchigar, S.; Yu, E. Business-driven data analytics: A conceptual modeling framework. *Data Knowl. Eng.* **2018**, *117*, 359–372. [CrossRef]
4. Ltifi, H.; Kolski, C.; Ben Ayed, M. Survey on Visualization and Visual Analytics pipeline-based models: Conceptual aspects, comparative studies and challenges. *Comput. Sci. Rev.* **2020**, *36*, 100245. [CrossRef]
5. Vázquez-Ingelmo, A.; García-Peñalvo, F.J.; Therón, R.; Conde, M. Representing Data Visualization Goals and Tasks through Meta-Modeling to Tailor Information Dashboards. *Appl. Sci.* **2020**, *10*, 2306. [CrossRef]
6. Lam, H.; Tory, M.; Munzner, T. Bridging from Goals to Tasks with Design Study Analysis Reports. *IEEE Trans. Vis. Comput. Graph.* **2017**, *24*, 435–445. [CrossRef]
7. Iung, A.; Carbonell, J.; Marchezan, L.; Rodrigues, E.; Bernardino, M.; Basso, F.P.; Medeiros, B. Systematic mapping study on domain-specific language development tools. *Empir. Softw. Eng.* **2020**, *25*, 4205–4249. [CrossRef]
8. Kandel, S.; Paepcke, A.; Hellerstein, J.M.; Heer, J. Enterprise Data Analysis and Visualization: An Interview Study. *IEEE Trans. Vis. Comput. Graph.* **2012**, *18*, 2917–2926. [CrossRef]
9. Wongsuphasawat, K.; Liu, Y.; Heer, J. Goals, Process, and Challenges of Exploratory Data Analysis: An Interview Study. *arXiv* **2019**, arXiv:1911.00568 [cs].
10. Kelly, S.; Tolvanen, J.-P. *Domain-Specific Modeling: Enabling Full Code Generation*; John Wiley & Sons: Hoboken, NJ, USA, 2008.
11. Morgan, R.; Grossmann, G.; Stumptner, M. VizDSL: Towards a Graphical Visualisation Language for Enterprise Systems Interoperability. In Proceedings of the 2017 International Symposium on Big Data Visual Analytics (BDVA), Adelaide, Australia, 7–10 November 2017; pp. 1–8.
12. Smeltzer, K.; Erwig, M. A Domain-Specific Language for Exploratory Data Visualization. In Proceedings of the 17th ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences, New York, NY, USA, 5 November 2018; Association for Computing Machinery: Boston, MA, USA, 2018; pp. 1–13.
13. Heer, J.; Card, S.K.; Landay, J.A. Prefuse: A Toolkit for Interactive Information Visualization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*; Association for Computing Machinery: New York, NY, USA, 2005; pp. 421–430.
14. Cook, K.A.; Thomas, J.J. *Illuminating the Path: The Research and Development Agenda for Visual Analytics*; IEEE Computer Society: Los Alamitos, CA, USA, 2005.
15. Sedlmair, M.; Meyer, M.; Munzner, T. Design Study Methodology: Reflections from the Trenches and the Stacks. *IEEE Trans. Vis. Comput. Graph.* **2012**, *18*, 2431–2440. [CrossRef]
16. Hevner, A.; Chatterjee, S. Design Science Research in Information Systems. In *Design Research in Information Systems: Theory and Practice*; Hevner, A., Chatterjee, S., Eds.; Integrated Series in Information Systems; Springer: Boston, MA, USA, 2010; pp. 9–22. ISBN 978-1-4419-5653-8.
17. Shen, L.; Chen, X.; Liu, R.; Wang, H.; Ji, G. Domain-Specific Language Techniques for Visual Computing: A Comprehensive Study. *Arch. Comput. Methods Eng.* **2020**, *28*, 3113–3134. [CrossRef]
18. Paige, R.F.; Kolovos, D.S.; Polack, F.A. A tutorial on metamodelling for grammar researchers. *Sci. Comput. Program.* **2014**, *96*, 396–416. [CrossRef]
19. van Deursen, A.; Klint, P.; Visser, J. Domain-specific languages: An Annotated Bibliography. *ACM SIGPLAN Not.* **2000**, *35*, 26–36. [CrossRef]

20. Rivera, J.E. On the Semantics of Real-Time Domain Specific Modeling Languages. Ph.D. Thesis, University of Malaga, Málaga, Spain, 2010.
21. González-Torres, A.; García-Peñalvo, F.J.; Therón-Sánchez, R.; Colomo-Palacios, R. Knowledge discovery in software teams by means of evolutionary visual software analytics. *Sci. Comput. Program.* **2016**, *121*, 55–74. [CrossRef]
22. Logre, I.; Déry-Pinna, A.-M. MDE in Support of Visualization Systems Design: A Multi-Staged Approach Tailored for Multiple Roles. *Proc. ACM Hum.-Comput. Interact.* **2018**, *2*, 1–17. [CrossRef]
23. Alves, Â.P.; Milani, A.M.P.; Manssour, I.H. Visual Analytics System for Energy Data in Smart Cities and Buildings. In Proceedings of the 2020 IEEE International Smart Cities Conference (ISC2), Virtual, 28 September 2020; pp. 1–8.
24. Pettit, C.; Lieske, S.N.; Jamal, M. CityDash: Visualising a Changing City Using Open Data. In *Planning Support Science for Smarter Urban Futures*; Geertman, S., Allan, A., Pettit, C., Stillwell, J., Eds.; Lecture Notes in Geoinformation and Cartography; Springer International Publishing: Cham, Switzerland, 2017; pp. 337–353. ISBN 978-3-319-57819-4.
25. Rojas, E.; Bastidas, V.; Cabrera, C. Cities-Board: A Framework to Automate the Development of Smart Cities Dashboards. *IEEE Internet Things J.* **2020**, *7*, 10128–10136. [CrossRef]
26. Langer, T.; Meisen, T. Visual Analytics for Industrial Sensor Data Analysis. In *Proceedings of the 23rd International Conference on Enterprise Information Systems; 26–28 April 2021*; SCITEPRESS-Science and Technology Publications: Setúbal, Portugal, 2021; pp. 584–593.
27. Logre, I.; Mosser, S.; Collet, P.; Riveill, M. Sensor Data Visualisation: A Composition-Based Approach to Support Domain Variability. In *Modelling Foundations and Applications*; Springer: Berlin/Heidelberg, Germany, 2014; Volume 8569, pp. 101–116.
28. Kosar, T.; Bohra, S.; Mernik, M. Domain-Specific Languages: A Systematic Mapping Study. *Inf. Softw. Technol.* **2016**, *71*, 77–91. [CrossRef]
29. do Nascimento, L.M.; Viana, D.L.; Neto, P.; Martins, D.; Garcia, V.C.; Meira, S. A Systematic Mapping Study on Domain-Specific Languages. In Proceedings of the Seventh International Conference on Software Engineering Advances (ICSEA 2012), Lisbon, Portugal, 18–23 November 2012; pp. 179–187.
30. Kitchenham, B. *Procedures for Performing Systematic Reviews*; Keele University: Keele, UK, 2004; pp. 1–26.
31. Sánchez-Gordón, M.; Colomo-Palacios, R. Taking the emotional pulse of software engineering—A systematic literature review of empirical studies. *Inf. Softw. Technol.* **2019**, *115*, 23–43. [CrossRef]
32. Wohlin, C.; Runeson, P.; Höst, M.; Ohlsson, M.C.; Regnell, B.; Wesslén, A. *Experimentation in Software Engineering*; Springer Science & Business Media: Berlin, Germany, 2012.
33. Ledur, C.L. *Gmavis: A Domain-Specific Language for Large-Scale Geospatial Data Visualization Supporting Multi-Core Parallelism*; Pontifícia Universidade Católica do Rio Grande do Sul: Porto Alegre, Brazil, 2016.
34. Ledur, C.; Griebler, D.; Manssour, I.; Fernandes, L.G. A High-Level DSL for Geospatial Visualizations with Multi-Core Parallelism Support. In Proceedings of the 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC), Turin, Italy, 4–8 July 2017; Volume 1, pp. 298–304.
35. Vázquez-Ingelmo, A.; García-Peñalvo, F.J.; Therón, R. Domain Engineering for Generating Dashboards to Analyze Employment and Employability in the Academic Context. In Proceedings of the Sixth International Conference on Technological Ecosystems for Enhancing Multiculturality, Salamanca, Spain, 24–26 October 2018; Association for Computing Machinery: New York, NY, USA, 2018; pp. 896–901.
36. Sun, R.; Gui, R.; Neema, H.; Chen, Y.; Ugirumurera, J.; Severino, J.; Pugliese, P.; Laszka, A.; Dubey, A. TRANSIT-GYM: A Simulation and Evaluation Engine for Analysis of Bus Transit Systems. In Proceedings of the 2021 IEEE International Conference on Smart Computing (SMARTCOMP), Irvine, CA, USA, 23–27 August 2021; pp. 69–76.
37. Smeltzer, K.; Erwig, M.; Metoyer, R. A Transformational Approach to Data Visualization. In Proceedings of the 2014 International Conference on Generative Programming: Concepts and Experiences, Västerås, Sweden, 15–16 September 2014; Association for Computing Machinery: New York, NY, USA, 2014; pp. 53–62.
38. Bostock, M.; Heer, J. Protovis: A Graphical Toolkit for Visualization. *IEEE Trans. Vis. Comput. Graph.* **2009**, *15*, 1121–1128. [CrossRef]
39. Heer, J.; Bostock, M. Declarative Language Design for Interactive Visualization. *IEEE Trans. Vis. Comput. Graph.* **2010**, *16*, 1149–1156. [CrossRef]
40. Li, D.; Mei, H.; Shen, Y.; Su, S.; Zhang, W.; Wang, J.; Zu, M.; Chen, W. ECharts: A declarative framework for rapid construction of web-based visualization. *Vis. Inform.* **2018**, *2*, 136–146. [CrossRef]
41. Teng, Z.; Fu, Q.; White, J.; Schmidt, D.C. Sketch2Vis: Generating Data Visualizations from Hand-Drawn Sketches with Deep Learning. In Proceedings of the 2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA), Pasadena, CA, USA, 13–16 December 2021; pp. 853–858.
42. Satyanarayan, A.; Heer, J. Authoring Narrative Visualizations with Ellipsis. *Comput. Graph. Forum* **2014**, *33*, 361–370. [CrossRef]
43. Bostock, M.; Ogievetsky, V.; Heer, J. D^3 Data-Driven Documents. *IEEE Trans. Vis. Comput. Graph.* **2011**, *17*, 2301–2309. [CrossRef] [PubMed]
44. Liu, Y.; Kale, A.; Althoff, T.; Heer, J. Boba: Authoring and Visualizing Multiverse Analyses. *IEEE Trans. Vis. Comput. Graph.* **2020**, *27*, 1753–1763. [CrossRef]

45. Bergel, A.; Maass, S.; Ducasse, S.; Girba, T. A Domain-Specific Language for Visualizing Software Dependencies as a Graph. In Proceedings of the 2014 Second IEEE Working Conference on Software Visualization, Victoria, BC, Canada, 29–30 September 2014; pp. 45–49.
46. Satyanarayan, A.; Moritz, D.; Wongsuphasawat, K.; Heer, J. Vega-Lite: A Grammar of Interactive Graphics. *IEEE Trans. Vis. Comput. Graph.* **2016**, *23*, 341–350. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.