The Journal of Systems and Software 213 (2024) 112029

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Data visualization guidance using a software product line approach ☆

David Romero-Organvidez [a,c,*], Jose-Miguel Horcas [b], José A. Galindo [a,c], David Benavides [a,c]

[a] Department of Computer Languages and Systems, Universidad de Sevilla, Av. Reina Mercedes, Seville, 41012, Spain
[b] ITIS Software, University of Málaga, Málaga, Spain
[c] I3US Research Institute, Universidad de Sevilla, Seville, Spain

ARTICLE INFO

Dataset link: http://uvlhub.io/doi/10.5072/zenodo.39565

Keywords:
Effective communication
Graphs
Tables
Software product line
Variability
Visualization

ABSTRACT

Data visualization aims to convey quantitative and qualitative information effectively by determining which techniques and visualizations are most appropriate for different situations and why. Various software solutions can produce numerous visualizations of the same data set. However, data visualization encompasses a wide range of visual configurations that depend on factors such as the type of data being displayed, the different displays (e.g., scatter plots, line graphs, and pie charts), the visual components used to represent the data (e.g., lines, dots, and bars), and the specific visual attributes of those components (e.g., color, shape, size, and length). A similar problem arises when designing data tables, where the dimensionality of the data and its complexity influence the choice of the most appropriate structure (e.g., unidirectional, bidirectional). Often, this broad spectrum of configurations requires a visualization expert who knows which techniques are best for which type of data source and what is to be conveyed. Typically, researchers and developers lack knowledge of data visualization best practices and must learn the design principles that enable effective communication and the technical details of the specific software tool they use to generate visualizations. This paper proposes a software product line approach to model and realize the variability of the visualization design process, using feature models to encode knowledge about design best practices in graphs and charts. Our approach involves solving visualization design variability through a stepwise configuration process and evaluating the proposal for a specific software visualization tool. Our solution facilitates effective communication of quantitative results by helping researchers and developers select and generate the most effective visualizations for each case. This approach opens up new opportunities for research at the intersection of data visualization and variability.

1. Introduction

We are currently immersed in digital transformation scenarios like Cloud Computing, Internet of Things, and Cyber-Physical Systems. In these scenarios, vast data is generated, stored, and analyzed (Walny et al., 2020). This quantitative data is essential for organizations and practitioners, such as researchers and scientists, to exploit and communicate their results and advancements. However, due to the volume and diversity of the data, it is challenging to extract and present relevant information for end-users. To tackle this challenge, the interdisciplinary field of *Data visualization* (Kirk, 2012) is employed, which deals with visually representing data to communicate information effectively.

However, data visualization exposes a large space of possible visual configurations depending on the type of data to be visualized, the data relationships, and the message we want to transmit (Evergreen, 2019; Knaflic, 2015). The use of graphs and tables is a common practice in organizations today, becoming the fundamental vehicle to represent quantitative information (Few, 2012). There exist considerable variations in graphs and tables (e.g., scatter plots, line graphs, bidirectional tables) that correspond to different quantitative relationships of the data (e.g., correlation, ratio, ranking, hierarchical), and each of these variations can be paired with the visual components and techniques (e.g., points, lines, bars, boxes) that present them most effectively. These visual components can be further configured to improve their visual perception through multiple visual attributes such as form attributes (e.g., length, width, orientation, shape, size, enclosure), color attributes (e.g., hue, intensity), or position attributes (e.g., 2D, 3D position), among others. Both graphs and tables have been developed over time to the point that visualization experts now thoroughly understand which works better for different circumstances and why (Tufte, 1985; Evergreen, 2019). Nevertheless, few practitioners have yet learned the

☆ Editor: Laurence Duchien.

* Corresponding author.
E-mail addresses: drorganvidez@us.es (D. Romero-Organvidez), horcas@uma.es (J.-M. Horcas), jagalindo@us.es (J.A. Galindo), benavides@us.es (D. Benavides).

https://doi.org/10.1016/j.jss.2024.112029
Received 3 March 2023; Received in revised form 7 March 2024; Accepted 17 March 2024
Available online 27 March 2024

design practices that make the use of graphs and tables effective (Cleveland and McGill, 1985; Midway, 2020). Evidence of this fact in the form of countless poorly designed graphs and tables is visible in the literature (Section 2). For example, pie charts are widely used for visual representation, despite the fact that they are not recommended for data that represent a relationship between a part and a whole, because there exists evidence of cognitive perception that indicates that the human eye is not well prepared to detect differences in the size of the angles; and in this case, bar graphs are better options for easy reading and comparing data (Few, 2012).

Moreover, there are a variety of software solutions capable of generating a multitude of different visualizations of the same data. Examples of languages and tools used in data science activities and industry digital transformation are *Data-Driven Documents (D3.js)* (Bostock et al., 2011) for web engineering, *Grafana* (Chakraborty and Kundan, 2021) a platform that acts as a framework for data visualization of any kind, *Pgfplots* and *TikZ* (Tantau, 2013) in LaTeX for research, *Ggplot2* (Wickham, 2009) in R and *Matplotlib* (Hunter, 2007) in Python for data analysis, just to mention a few. These options offer little support to guide users or developers in determining the most correct way to visualize the data, that is, how to decide which visualization is the most appropriate for a dataset with certain properties. The problem increases with the huge number of configurations regarding the visual components and attributes that those libraries offer and that may overwhelm the user, requiring considerable technical skill to know the low-level details of the software library used to generate the visualization.

Of further concern is the lack of scientific rigor that leads to misinterpretations and incorrect decisions. Nguyen et al. (2021) identify the causes of misinformation in visualizations that, deliberately or not, induce error, deception and visual pitfalls. Crucial aspects such as color, shape and size of figures, although the general public is used to visualize and understand, do not always follow good practices, which encourages the use of visualizations with a certain statistical bias. Braga et al. (2020) discusses these statistical biases due to visualizations that are not entirely effective and that are used, for the most part, by journalists and researchers to offer conclusions that are not those drawn from those visualizations. Cognitive biases also seem to be a cause of problems in the interpretation of visualizations, thus Sabrina Bresciani and Martin J. Eppler (Bresciani and Eppler, 2015) propose an enhancement of visual literacy by identifying cognitive, emotional, and social biases and framing them into a list of visualization difficulties. These difficulties reveal the importance of recognizing and understanding common errors in interpreting graphic representations.

In this paper, we propose a *software product line* (SPL) (Pohl et al., 2005) approach to model and materialize the variability of the visualization design process (Section 3). The *visualization design process* (Walny et al., 2020) is concerned with the design practices, development, and application of visual representation of data assisted by computers to display the information for accurate and efficient interpretation by the reader. We make the following contributions:

- We synthesize the best practices of the visualization design process that have been learned through many years of research and real-world trial and error by trailblazers (Few, 2012; Evergreen, 2019). To do so, we encode the variability of the visualization knowledge regarding the best design practices in feature models and follow a *step-wise configuration* (Czarnecki et al., 2005b) approach that allows configuring different displays, visual components, and attributes in a specific software visualization library or tool, so that non-experts in visualization can easily decide the best representation for their data (Section 4).
- We materialize the variability of visualization designs by automatically generating effective visualization implementations using existing software solutions. To do so, we rely on *template-based code generation* (Syriani et al., 2018) to resolve the variability of the visualization at different levels of abstraction, so that users and developers can build effective graphs and tables without a deep understanding of the low-level details of each visualization tool (Section 5).
- We provide an implementation of our SPL approach and apply it to seven practical scenarios to generate visualizations that take into account the best design practices to communicate a quantitative message (Section 6).

Although there are several works discussing the best visualizations for SPL artifacts (Lopez-Herrejon et al., 2018) (e.g., visualization of feature models), to the best of our knowledge, there is no approach that manages the variability of the visualization design process (Section 8). Our solution helps practitioners communicate their quantitative data effectively by assisting them in the selection and generation of the visualizations that work best for each case. We envision that this contribution can open a new line of research where data visualization and variability management, analysis, and implementation can benefit of each other (Section 8).

The challenge of obtaining visualizations that are effective, with good data understanding, and that are supported by good practices and heuristics raises two research questions:

- RQ1: *How does using a Software Product Line (SPL) approach for visualization compare to manual creation in terms of effectiveness for developers not skilled in visualization?* **Rationale:** This question explores the impact of adopting an SPL approach to visualization tasks, particularly for developers who lack experience in visualization techniques. By comparing this structured method with traditional manual approaches, the aim is to understand whether SPL can facilitate the design and development process to produce more appropriate and effective visualizations. This comparison can reveal SPL's potential benefits or drawbacks in improving visualization quality, efficiency and suitability for developers with limited visualization skills.
- RQ2: *How do multi-visualization tools compare to limited-capability ones in flexibility and data representation for users?* **Rationale:** This question examines whether tools offering a variety of visualization types enhance users' ability to effectively and appropriately represent data, compared to more constrained tools. It aims to understand the impact of visualization diversity on user satisfaction and data interpretation quality.

Throughout the paper, we answer these two questions based on the current literature and the response capabilities of our proposal.

This article is an extension of the original "Variability in data visualization: a software product line approach" by Horcas et al. (2022a). In the original paper, we focused on using a product line that encompassed the generation of code that allowed for the visualization of graphs only. In this new contribution, we extend the study to data tables, including new feature models that cover the two case studies. In addition, we have involved a visualization expert to improve the visualization templates or our implementation, enhancing the evaluation of the proposal.

2. Data visualization and best design practices

This section presents the main concepts of the visualization design process and the best design practices, motivating our approach.

2.1. The visualization design process

The visualization design process (Kirk, 2012) is concerned with the design practices, development and application of visual representation of data to effectively communicate information. A formal model of the visualization design process is proposed by Walny et al. (2020) and includes five main stages (Fig. 1): (1) conceptualization of the vision and goals of the project, as well as the dataset; (2) analysis and characterization of the data to be visualized including understanding the data types, amounts, and relationships; (3) abstraction and encoding design steps that encompass creative design and data mapping with

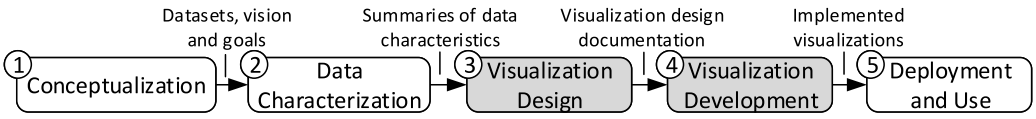


Fig. 1. Stages of the data visualization design process (Walny et al., 2020).

the required representation, resulting in a visualization design concept (or visualization design documentation); (4) implementation of the visualization according to the design performed in the previous phases and using a specific software visualization library; and (5) deployment of the visualization for public use, including its maintenance and data updates. This process may involve not only developers, but also designers, data experts, and clients or end users, each with their own knowledge and technical skills. We focus on stages 3 (Visualization Design) and 4 (Visualization Development) highlighted in Fig. 1 which are the most interesting phases due to their complexity in variability.

2.1.1. Visualization design

Data visualization maps values to visuals, turning numbers into graphs to convey a story or an idea as efficiently as possible. According to Few (2012), tables and graphs are the main visualizations (aka displays) to structure and communicate quantitative information. Therefore, we focus here on the variability exposed by these types of displays.

Visualization design of graphs. Graphs display relationships (e.g., time series, ranking, part-to-whole, distribution) in quantitative information by giving shape to those relationships. Multiple structural forms of graphs can be used to display each relationship by encoding quantitative values as points (e.g., dot plots, scatter plots, strip plots, bubbles), lines (e.g., line graphs that may also include points, frequency polygon graphs), bars (e.g., histograms, table lens), boxes (e.g., box plots with vertical or horizontal orientation), and 2D and 3D shapes (e.g., pie charts, donut charts, radar charts). Deciding the best value-encoding object and graph type for each kind of relationship requires to understand the specific types of relationships that graphs can display (Few, 2012). For instance, data with a correlation relationship should be displayed by encoding values with points as a scatter plot (Few, 2012), but the user needs to be able to identify that the message she wants to transmit is a correlation in the data (e.g., a causal relationship) in contrast to a deviation or a distribution relationship. Furthermore, several visual and textual components work together in graphs to present quantitative information. Design involves not only choosing the primary components that are used to construct the graphs (i.e., points, lines, bars, plots), but also designing the secondary components and non-data components to display the information (e.g., axes, trend lines, reference lines, annotations, scales, tick marks, grid lines, legends, etc.), as well as determining the visual attributes of each component such as its form (e.g., length, width, orientation, size, shape, enclosure), its color (e.g., hue, intensity), or its position (e.g., 2D position).

Visualization design of tables. Tables can present structural variations beyond their general structure of columns and rows such as unidirectional tables (i.e., to arrange the data across the columns or down the rows) and bidirectional tables (i.e., to arrange the data in both directions). Deciding the best structural variation depends on the specific relationship between the individual values that the table should exhibit. For instance, quantitative-to-categorical relationships are primarily used for looking up one quantitative value at a time, and therefore unidirectional tables are preferable; while bidirectional tables work especially well in quantitative-to-quantitative relationships because the values are arranged closely together for easy comparison.

2.1.2. Visualization development

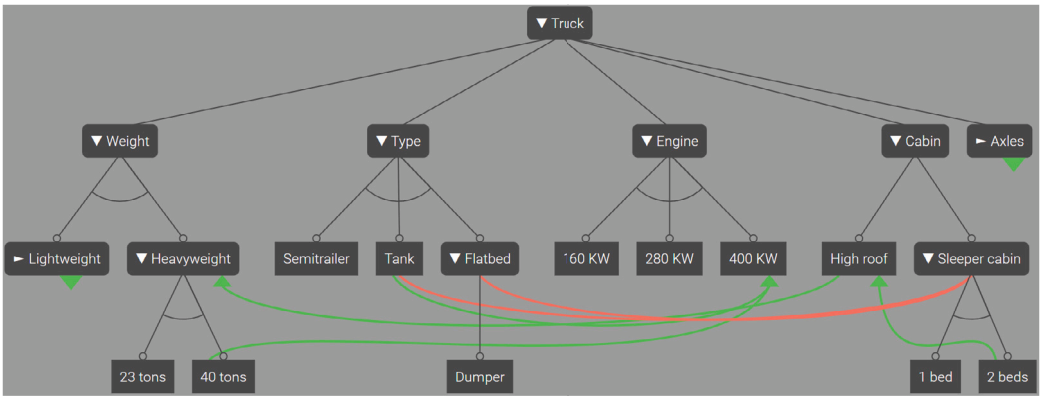
There are a variety of software solutions to develop and generate a visualization of data such as *Matplotlib* (Hunter, 2007), *Ggplot2* (Wickham, 2009), *Pgfplots* (Tantau, 2013), *Grafana* (Chakraborty and Kundan, 2021). Each tool requires considerable technical skills by the users to know its low-level details in order to generate an efficient visualization. Moreover, these tools expose a colossal number of options regarding the visual components and attributes to develop the visualization, but they offer little support to guide users or developers regarding the most correct way to visualize the data.

2.2. Best design practices in data visualization

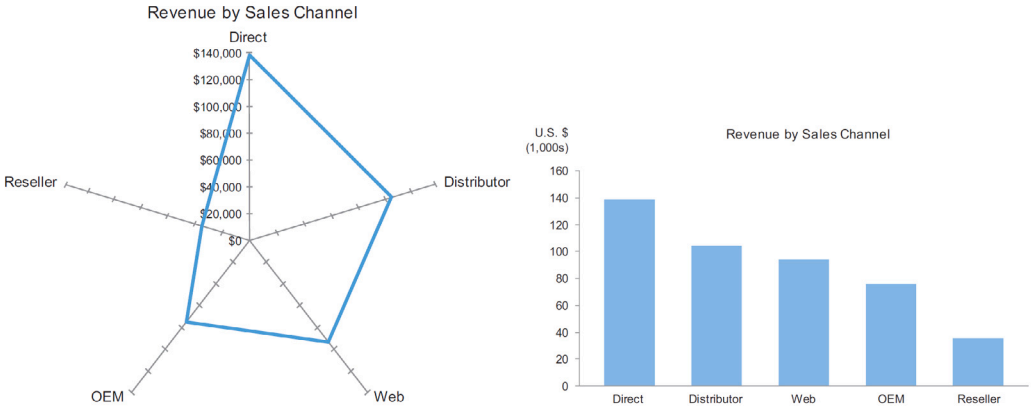
Despite the high number of possible options in the design and implementation of visualization to represent data, only a few of the configurations are appropriate to effectively communicate information. For example, only graphs based on points, lines, bars, and boxes are recommended (with rare exceptions) (Few, 2012) because they rely on visual attributes that can be easily and accurately perceived; while 2D and 3D shapes (e.g., pie charts, donut charts) fail at data representation, mainly due to perceptual reasons, even though their high popularity (Few, 2012). In this regard, many fields of scientific study have contributed to the understanding of visual perception and have applied it to visual design (Bertin, 1983; Cairo, 2012; Cleveland and McGill, 1985; Few, 2012; Moody, 2009; Murray, 2017; Tufte, 1985). A set of patterns known as the *Gestalt Principles of Visual Perception* (Gordon, 2004) uncovered how people perceive patterns, forms, and organizations, revealing that people tend to group objects in particular ways. Many of these findings have been applied in the design of tables and graphs (Cairo, 2012; Few, 2012; Murray, 2017). For instance, the *principle of proximity* (Gordon, 2004) establishes that objects that are close to each other are perceived as belonging to a group, and thus, tables can be arranged better in a particular direction without the need of using horizontal or vertical rules to delineate rows or columns, improving the *data-ink ratio* (Few, 2012) (i.e., the amount of ink that presents important data compared to the total amount of ink used to present the display). Another example is the *principle of similarity* (Gordon, 2004) that establishes that similar objects in color, size, shape, or orientation tend to be grouped together; and thus, shapes with symbols keeping their interiors empty of color (circles, squares, and triangles) are easier to distinguish in graphs. Moreover, some visual attributes considerably affect the visual perception of the display (Cleveland and McGill, 1985), such as the color, where there are a list of nine hues (i.e., gray, blue, orange, green, pink, brown, purple, yellow, red) that meet the requirement of distinctness, and therefore they are easy to recognize and distinct enough to work well together; but its selection is challenging because red and green in combination can be difficult to differentiate for colorblindness people.

Deciding whether to use a table or a graph as a display, the best components to build such a display, and the visual attributes or those components, requires knowing the principles of visual perception behavior and its application to graphical communication (Midway, 2020; Telea, 2014), but also understanding the specific types of relationships that each visualization can display (Few, 2012). However, yet few of the practitioners have learned the design practices that make the use of tables and graphs effective. Evidence of this fact in the form of countless poorly designed tables and graphs is visible in the literature. Concretely, in Fig. 2 and in Section 6, we illustrate some

[1] The content information inside these graphs and tables are not relevant in our paper.



(a) Feature model displayed in the *Glencoe online tool* (Schmitt et al., 2018).



(b) Two visualizations for a nominal comparison.

Type	Model	Z3	Clafer	BBPF
FSE2015	Dune	26.20s	11s	0.01s
	HSMGP	40.70s	14s	0.01s
	HiPAcc	458s	33s	0.01s
	Trimesh	Time-out	2s	0.01s
KConfig	axTLS	Time-out	Time-out	0.01s
	Fiasco	Time-out	Time-out	0.01s
	uClibc-ng	Time-out	Time-out	0.01s

Type	Model	Z3	Clafer	*BBPF*
FSE2015	Dune	26.20s	11s	0.01s
	HSMGP	40.70s	14s	0.01s
	HiPAcc	458s	33s	0.01s
	Trimesh	Time-out	2s	0.01s
KConfig	axTLS	Time-out	Time-out	0.01s
	Fiasco	Time-out	Time-out	0.01s
	uClibc-ng	Time-out	Time-out	0.01s

(c) Comparison of counting time solutions of three solvers (Munoz et al., 2019).

Fig. 2. Examples of visualizations, extracted from the literature, that suffer from design problems[1] (Schmitt et al., 2018).

of the design problems in visual representations extracted from the literature. For instance, in Fig. 2(a), the dark hues used do not meet the requirement of distinctness with features' colors fading more into the background. Also, green and red colors for "requires" and "excludes" constraints are difficult to distinguish for colorblindness people. In contrast, black features over a white background and different line styles for constraints would be easier to read. Another example is the radar chart at the left of Fig. 2(b) (taken from Few, 2012) which is a graph type often available in visualization tools, but it does not present information clearly, accurately, and efficiently, mostly because of perceptual reasons. The same nominal comparison of sales channels can be better represented with the bar graph at the right-hand side of Fig. 2(b) (taken from Few, 2012). The bar graph is easier to read since positions along a quantitative scale are much easier to compare when they are laid out linearly along a single vertical or horizontal axis. Also, a bar graph supports additional meanings to be displayed such as ranking the items, but the radar char does not because it is not clear where the information begins and ends, or whether it should be read clockwise or counterclockwise. Finally, best design practices can be also applied to tables. For instance, the table at the left of Fig. 2(c) (taken from Munoz et al., 2019) contains a high ratio of non-data ink hindering the important data. The same table (at the right of Fig. 2(c)) considers the *principle of proximity* (Gordon, 2004) to reduce the unnecessary rules (non-data ink) and enhance the relevant information.

A summary of the best design practices for visualization to decide the best display according to the data relationships and communicate the information effectively is shown in Tables 1 and 2. While Table 1 summarizes the recommendations to select the best graphs for each data relationship, Table 2 presents the best practices to select the most appropriate structural type for tables according to the data relationships. These recommendation have been extracted from the literature on data visualization, and specially, from Few (2012).

By taking into account the design practices and the principles of graphical design for communication, practitioners will understand what visual techniques work, what do not work, and why. In the next section, we propose an approach to model and manage the variability of the visualization design process and encoding the knowledge about best design practices for graph and tables.

Table 1
Principles and best practices for graphs design.

Data relationships in graphs	Points	Lines	Bars	Boxes
Time series: Values display how something changed through time. • *Keywords:* change, rise, increase, fluctuate, grow, decline, decrease, trend. • *BP:* Use preferably lines and vertical bars to emphasize on overall patterns and individual values, respectively; use points as *dot plots* only when missing values at time intervals; use boxes only for distributions changing through time.	☒	■	■	☒
Ranking: Values are ordered by size (ascending or descending). • *Keywords:* larger than, smaller than, equal to, greater than, less than. • *BP:* Use preferably bars; use *dot plots* when bars cannot be used because quantitative scale does not begin at zero; use boxes only when ranking multiple distributions.	☒	□	■	☒
Part-to-whole: Values represent parts (proportions) of a whole. • *Keywords:* rate or rate of total, percent or percentage of total, share, accounts for X percent. • *BP:* Use preferably bars; use lines to display how parts of a whole have changed through time.	□	■	■	□
Deviation: The difference between two sets of values. • *Keywords:* plus or minus, variance, difference, relative to. • *BP:* Use bars, but always vertical when combined with time series; *dot plots* when quantitative scale does not begin at zero; lines when combined with time series.	☒	☒	■	□
Distribution: Counts of values per interval from lowest to highest. • *Keywords:* normal curve, concentration, frequency, distribution, range, normal distribution, bell curve. • *BP:* Use *strip plots* to emphasis individual features; use a *frequency polygon* to emphasis the overall shape; use bars as *histograms* to emphasis on individual intervals in single distributions; avoid bars for multiple distributions; use *box plots* to compare multiple distributions.	■	■	☒	☒
Correlation: Comparison of two paired sets of values to determine if there is a relationship between them. • *Keywords:* increases with, decreases with, changes with, varies with, caused by, affected by, follows. • *BP:* Use a *scatter plot*; use a *table lens* when the audience is not familiar with *scatter plots*.	■	□	☒	□
Geospatial: Values are displayed on a map to show their location. • *Keywords:* geography, location, where, region, territory, country, state, city. • *BP:* Use bubbles of various sizes on a map; use lines to mark routes on a map.	■	■	□	□
Nominal comparison: A simple comparison of values for a set of unordered items. • *Keywords:* this is bigger than that, this is the biggest of all, this is almost twice as big as that, these two are far bigger than all the others. • *BP:* Use preferably bars; use *dot plots* when bars cannot be used because quantitative scale does not begin at zero.	☒	□	■	□

■: Most appropriate; ☒: Appropriate under certain conditions; □: Avoid.

Table 2
Principles and best practices for tables design.

Data relationships in tables	Unidirectional	Bidirectional
Quantitative to Categorical (Look-up)		
Single set of quantitative values and a single set of categorical items. • *BP:* Use unidirectional tables; bidirectional tables are not applicable because there is only one set of categorical items.	■	□
Single set of quantitative values and the intersection of multiple categories. • *BP:* Unidirectional tables is preferable because of convention; bidirectional tables save space.	■	■
Single set of quantitative values and the intersection of multiple hierarchical categories. • *BP:* Use unidirectional tables to clearly display hierarchy by placing the separate levels side by side in adjacent columns; use bidirectional tables if the separate levels of the hierarchy are not split between the columns and rows.	■	☒
Quantitative to Quantitative (Comparison)		
A single set of quantitative values associated with multiple categorical items. • *BP:* Bidirectional works especially well because the quantitative values are arranged closely together for easy comparison.	■	■
Distinct sets of quantitative values associated with a single categorical item. • *BP:* Use unidirectional tables; bidirectional tables tend to get messy as multiple sets of quantitative values are added.	■	☒

■: Most appropriate; ☒: Appropriate under certain conditions; □: Avoid.

3. An SPL for data visualization

Our approach consists of building an SPL (Fig. 3) to model and manage the variability and encoding the best practices of the visualization design process. We follow the classical SPL framework (Pohl et al., 2005) that separates the domain and the application engineering processes, and distinguishes between the problem and the solution spaces. Note that the result will be a visualization that follows best practices. This visualization may need to be tuned or refined by the design team if something more concrete is needed.

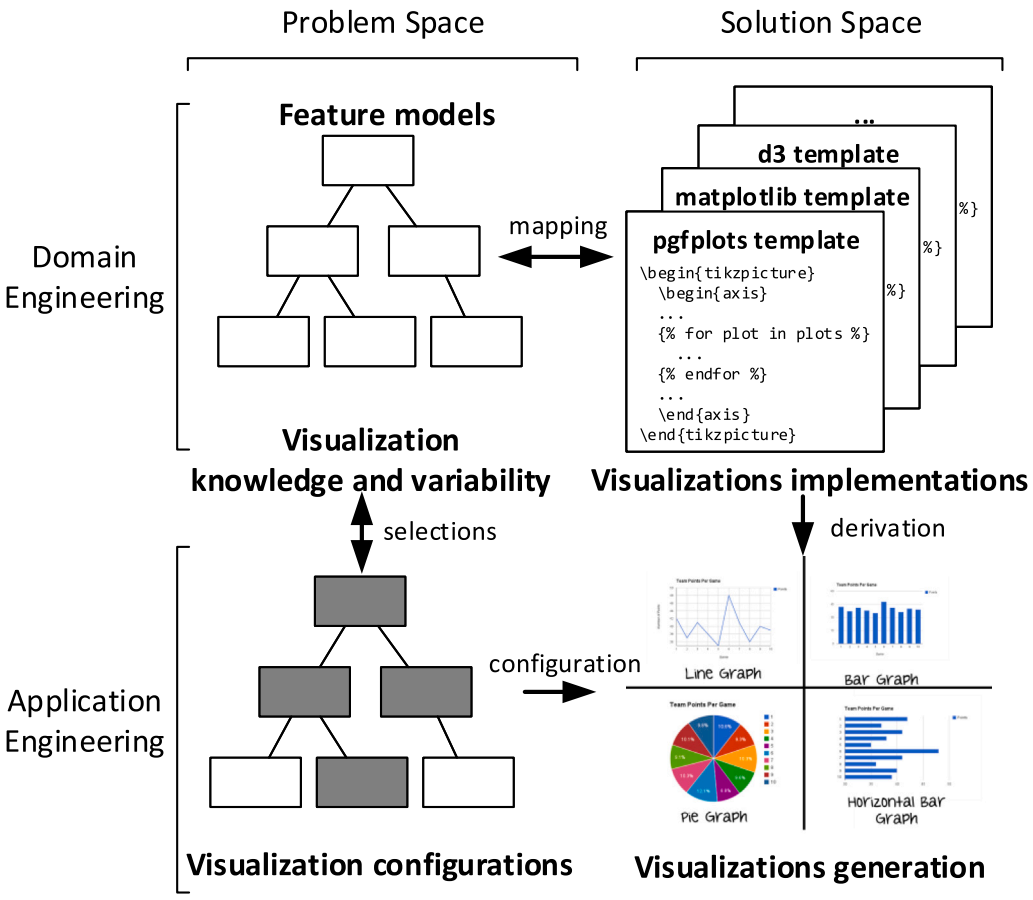


Fig. 3. SPL for the visualization design process.

In the domain engineering process, we analyze the knowledge and variability of the visualization design process encoding them in feature models (top left of Fig. 3), and prepare the reusable and variable artifacts to be used in multiple visualizations in terms of templates for the different visualization software libraries (top right of Fig. 3). In the application engineering process, we specify configurations for the visualization according to the stakeholder's requirements (bottom left of Fig. 3) and generate a final display for a specific software library (bottom right of Fig. 3).

Since we are interested in the separation of the visualization knowledge and variability from the implementation in a specific software library, we will distinguish the perspectives of the problem and the solution space in the following sections. The problem space takes the perspective of stakeholders (e.g., researchers, data analysts) and the problematic of designing visualizations that effectively communicate quantitative information about their data (Section 4). In contrast, the solution space represents the developer's perspective, including the generation of the visualization in a particular visualization software library (Section 5).

3.1. Multi-level variability modeling and step-wise configuration

Fig. 4 illustrates our approach to model and configure the variability of the visualization design process. Following the visualization design and development phases presented in Section 2, we refine the modeling of variability into four different levels (Rabiser et al., 2018; Reiser and Weber, 2007). Concretely, we define three levels according to the three fundamental steps in the design of a visualization identified by Few (2012) and an additional level corresponding with the step for the development of the visualization. The objective of each step is: (1) to determine the message to display; (2) to select the best means to display the information; (3) to design the display to show the information; and (4) to select the implementation for the visualization. The variability of each step is modeled in one level separately from the subsequent levels in individual feature models.

Feature models have no cross-tree constraints between them across the levels but they can be connected between them by a dependency relation implemented by means of *imports* or *interfaces* (Schröter et al., 2016). This way, the feature model of step 1 is connected with the two feature models of step 2 (one of them modeling the variability of graphs and the other the variability of tables). That is, if the user in a configuration chooses a graph to display the message, the following feature model in step 2 to be configured will be the one modeling the variability of graphs; otherwise, it will be the feature model modeling the variability of tables. The following feature models (from steps 2 to 4) are completely independent between them and could be configured in any order. The feature models of step 2 model the variability of the types of graphs (or tables). The feature models of step 3 model the variability of the visual components and attributes of graphs (or tables). Finally, the feature models of step 4 model the variability of the implementation of the visualization by exposing the selection of different visualization software libraries for graphs (or tables).

For configuring the feature models, we follow a step-wise configuration technique (Czarnecki et al., 2005b) in which the user configures the variability step by step starting from the first level. Since there are no cross-tree constraints across the feature models, there is no need for propagation between the models. The only dependency exists between the feature model of step 1 and the feature models of step 2 when selecting the main display: a graph or a table. After that choice, from the point of view of our SPL, the user may configure the variability of steps 2 to 4 in any order. That is, the feature models can be configured top-down (from step 2 to 4) by resolving first the most

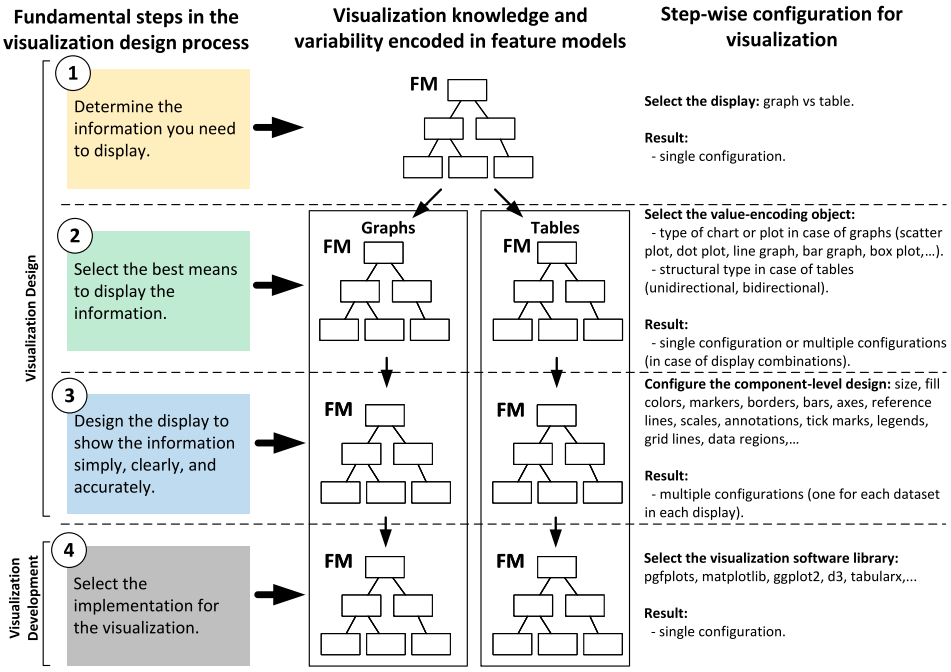


Fig. 4. Variability and configuration in the visualization design process.

generic variability; they can be configured following a bottom-up order (from step 4 to 2) resolving first the variability of the details of the visualization; or they can be configured in an arbitrary order. However, it makes more sense to configure them following the order (top-down) of the visualization design process (Section 2).

Using an SPL for data visualization provides a structured framework that benefits developers with limited visualization techniques. This approach answers RQ1 on the effectiveness of this method compared to manual creation. With its predefined feature models and built-in best practices, the SPL approach allows developers to tackle the complex task of creating visualizations more efficiently. Unlike manual authoring that requires extensive visualization knowledge, the SPL approach streamlines development, minimizes cognitive load and improves the quality of visualizations through a step-by-step guided process.

3.2. Encoding the visualization knowledge

To encode the visualization knowledge and expose the variability, we first analyze and extract the knowledge and best practices of the visualization design process that have been learned through many years of research and real-world trial and error by trailblazers. To do so, we analyze the literature on visualization design techniques and best practices (Cairo, 2012; Evergreen, 2019; Few, 2012; Knaflic, 2015; Murray, 2017; Telea, 2014) in a non-systematic way, and synthesize a set of feature models that encode this knowledge. These references are six books by experts in the field. Among all the books, we focus especially on Few's book (Few, 2012), as it contains most of the principles. This book serves as the main reference in this research. It draws together most of the practices from five other books, establishing a framework for good practice in data visualizations. We rely on this book for the codification of the visualization knowledge. This codification aligns with validated principles and helps to apply best practices in data visualizations and understanding them. The four configuration steps outlined in Section 4 are derived from Few's book. We have codified these best practices into the four levels, as suggested in the book.

The procedure to encode the knowledge in a feature model is based on the variability modeling experience of the authors described as follows:

1. **Identify the recommendation or best practice.** We first identify a recommendation, tip, or best practice in the literature. These are usually given in natural language and must be manually processed and synthesized. For example, about the orientation (horizontal vs vertical) in a bar graph: *"Horizontal bars are the best choice when it would be difficult to get categorical labels to fit side by side under vertical bars"*. extracted from Few (2012). Similar recommendations can be found in a different source. For example: *"The horizontal bar chart is especially useful if your category names are long, as the text is written from left to right, as most audiences read, making your graph legible for your audience"*. extracted from Knaflic (2015).
2. **Featurization of the main visualization concepts involved in the best practice.** After identifying the recommendation or best practice we identify the main concepts involved in the recommendation which will be modeled as features or attributes if they do not exist in the variability model. For instance, from the previous example, we extract the concept of "Horizontal Bars", "Categorical labels", and the length of those labels. The first two concepts will be represented as features in our model, while the length of the labels will be represented as an attribute of the feature "CategoricalLabels".
3. **Creation of the relations between features and/or attributes to satisfy the best practice.** Depending on the nature of the recommendation or best practice, the possible relations between the features (and/or attributes) vary. The relation can be modeled with a cross-tree constraint involving the features, with a relationship of the feature tree (i.e., an or-group, and alternative-group, or an optional/mandatory relation), with attributes of a feature, or with a combination of these. For instance, following the previous example, we create a cross-tree constraint `CategoricalLabels.length` > X → `HorizontalBars` that relates both features and the attribute. The value X can be obtained from a different source of the literature (if available) or can be adjusted to the user's needs or a particular visualization if there is not any more specific recommendations.

The common part of the feature models, i.e. those points of variability that are shared between several libraries, such as titles, axes

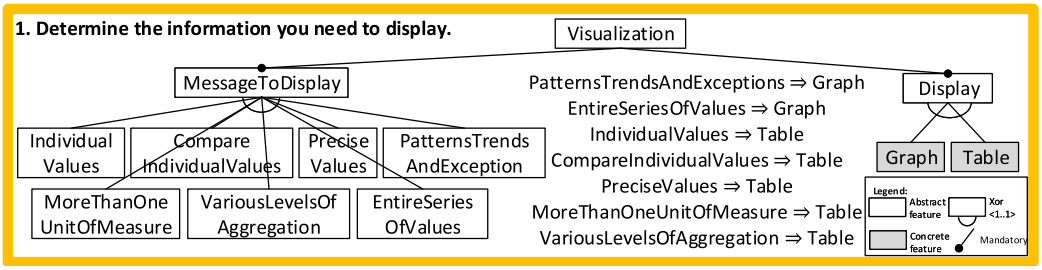


Fig. 5. Step 1. Encoding the knowledge to select a graph or a table to communicate the information.

(necessary or unnecessary) or legends, should be studied by a group of experts in visualizations. If the common part is validated, the range of erroneous visualizations is reduced since it is those essential common parts that tend to repeat an incorrect visualization pattern that induces misinterpretations.

We apply this process for the principles and best practices shown in Tables 1 and 2 for graphs design and tables design, respectively. For example, the last principle of Table 1 states that when the data relationship is a nominal comparison, it is preferable to use bars as value encoding, while we should use dot plots when bars cannot be used because the quantitative scale does not begin at zero. Thus, we synthetize this recommendation as two cross-tree constraints involving the different featurized concepts: `NominalComparison` → `Bars` ∨ `DotPlot`, and `NominalComparison` ∧ `QuantitativeScaleNotBeginAtZero` → `DotPlot` (see feature model of step 2 in 6). Another different example of encoding knowledge is done for the modeling of the possible color's selections, where we restrict the selection to only the nine hues that meet the requirement of distinctness (as present in Section 2.2) avoiding the selection of others colors, and thus reducing the variability and complexity of the final user in the configuration process. We model this recommendation with an alternative group selection with the nine colors (see feature model of step 3 in 7).

Incorporating data visualization best practices into a feature model improves this approach over more rigid tools. These tools ignore variability or require prior knowledge of the user. This approach partially answers research question RQ 2, highlighting the importance of adaptability and integrated knowledge to optimize data visualization.

In the following sections we detail our approach to design effectively visualizations (Section 4) and to automatically generate the visualizations (Section 5) for the two fundamental vehicles used to represent quantitative information: graphs and tables.

4. Variability modeling and configuration in visualization

We analyze and extract the knowledge and best practices of the visualization design process from the literature and synthesize a set of feature models that encode this knowledge. Following with our approach (Fig. 4), we separate the variability modeling in four steps distinguishing the variability modeling and configuration for graphs (Section 4.1) and for tables (Section 4.2). Since graphs and tables are very different visual elements to represent information, only the first step in which we decide which display to use is common.

Step 1. Determine the information you need to display (Fig. 5). The first step is to determine the message to be displayed from the data, and consequently choose the best medium of communication (i.e., a table or a graph). This requires knowing the different roles of graphs and tables when presenting quantitative information. As stated by Few (2012): *"tables make it easy to look up individual values"*, while *"graphs are used to display relationships among and between sets of quantitative values by giving them shape"*. The knowledge of whether a graph or a table is more appropriate has been encoded in the feature model of Fig. 5. This feature model allows deciding whether to use a table or a graph according to the message you want to transmit. The feature model encodes the different types of message as abstract features, and the relations between the message and the best concrete medium of communication to show that message (a table or a graph) as cross-tree constraints.

The following steps in the design of the visualization are very different for graphs and tables, thus we differentiate the variability modeling and configuration of graphs and tables.

4.1. Designing graphs

Multiple types of graphs can be used to display each relationship of the quantitative information. Moreover, graphs are built using several visual components that can be also configured. The following three steps are exclusive for graphs.

Step 2. Select the best means to display the information (Fig. 6). In the second step of the visualization design process, the goal is to select the best means to display the information in terms of the specific chart type according to the relationships of the data to communicate. The available knowledge and best practices for the design of graphs extracted from the literature is summarized in Table 1, and has been encoded in the feature model of Fig. 6. Fig. 9 specifies the fundamental variations of graphs in terms of chart types (concrete features in gray) that correspond to different quantitative relationships (abstract features in white). Knowledge about the type of graph that presents each quantitative relationship more effectively is encoded in the cross-tree constraints. For example, to represent a nominal comparison in the data, it is preferable to use bars by means of either a vertical or a horizontal bar graph; but when then quantitative values of the scale do not begin at zero, using dot plots is recommended (Few, 2012). To help the user determine whether her quantitative message involves a particular data relationship or another, we also encode a set of keywords and phrases that are commonly used to identify each relationship (Few, 2012). So, the user can first state the message in writing and then look to see whether she used any of the keywords to transmit the message. For example, the message likely involves a time series if it includes any of the following words: *change*, *rise*, *increase*, *fluctuate*, *grow*, *decline*, *decrease*, *trend*; while the following words suggest a correlation relationship: *increases with*, *decreases with*, *changes with*, *varies with*, *caused by*, *affected by*, *follows*. The feature model of this second step is normally configured once for a visualization, but can be configured multiple times to represent a combination of displays (e.g., a bar graphs combined with a time series).

Step 3. Design the display to show the information simply, clearly, and accurately (Fig. 7). In this step, we model the variability of the visual component-level graph design. The goal is to make the visual objects that encode data prominent, accurate, and clear to communicate the information. Graphs are constructed from components that can be divided into three groups: (i) primary components, that is, the points, bars, lines, and boxes that encode the quantitative data; (ii) components that serve secondary roles like the scales, trend lines, tick marks, and so on; and (iii) non-data components like the axis lines. Fig. 7 shows an excerpt from the feature model that encodes the variability in the

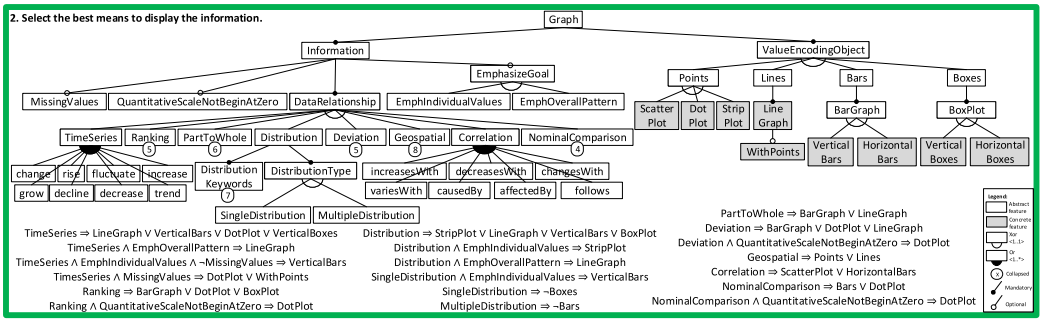


Fig. 6. Step 2 for graphs. Feature model to determine the best graph type according to the data relationship.

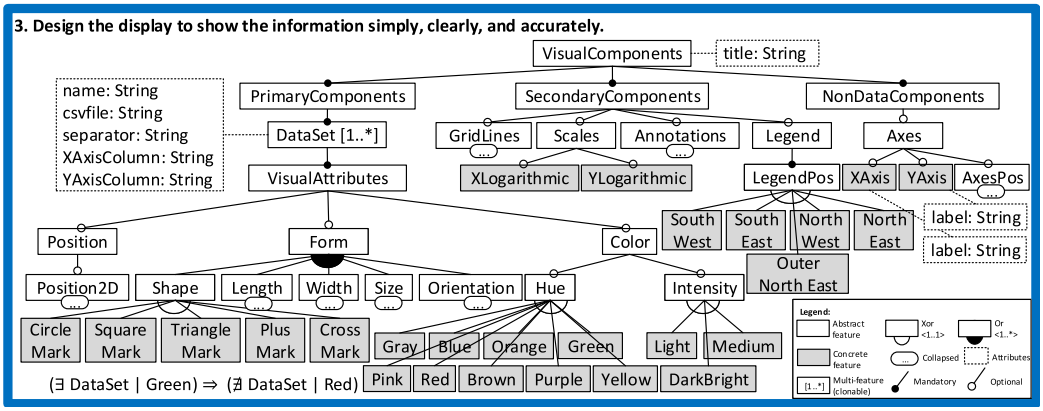


Fig. 7. Step 3 for graphs. Variability of the visual component-level graph design.

design of the visual components. On the left-hand side of Fig. 7, the visual attributes of the primary components need to be configured for each set of data, and thus we model the `DataSet[1..*]` feature as a *multi-feature* (Czarnecki et al., 2005a; Alférez et al., 2019) (aka *clonable feature*) that allows one to configure the entire subtree as many times as necessary. For instance, consider a graph representing a comparison of algorithms; each series of data (or dataset) will represent an algorithm with different visual attributes (position, form, color) to easily distinguish them. Note that the variability of the modeled visual attributes includes only those values that are recommended by visualization experts in the literature (Few, 2012; Telea, 2014; Evergreen, 2019), in contrast to modeling all possibilities of a specific software tool. This allows us to reduce the configuration space and avoid the selection of features exposed by the tool that may result in poor design practices undermining the final visualization. For example, we model only those shapes that are easy to distinguish from one another like circle (○), square (□), triangle (△), plus (+), and times (×) marks; or those color hues that meet the requirement of distinctness (Few, 2012). Some best practices can be encoded as complex cross-tree constraints such as the fact that green and red colors should not be used in combination because they can be difficult to differentiate for colorblindness people; therefore, we explicitly avoid the selection of both colors in different dataset of the same visualization: `(∃ DataSet | Green) ⇒ (∄ DataSet | Red)`. In addition, each dataset requires to provide a set of attributes such as a name to identify the data, the file path to the (.csv) file containing the data, and the column name for the X and Y axes inside the data file. This is modeled as *feature attributes* (Benavides et al., 2005) associated with the `DataSet` feature. The secondary components (middle of Fig. 7) include several other components of graphs that also provide information and can be configured such as the scales (linear or logarithmic), grid lines, annotations, or the legend including its position, among others (Few, 2012). We collapsed some of those features (`...`) and omitted others, such as trend lines or reference lines, due to lack of space. Finally, non-data components (right-hand side of Fig. 7) are axis lines that give graphs dimension, serve as a container for the data and also provide various possibilities for configurations, such as the number of axes and their positions. For instance, under most circumstances, graphs include a single axis either vertical or horizontal for 1D graphs, or two perpendicular axes for 2D graphs. The configuration process of the feature model in this third step results in multiple configurations, one for each dataset due to the presence of the `DataSet[1..*]` multi-feature, as well as due to the possibility of building a combination of displays from step 2 (e.g., a dot plot with a trend line).

Step 4. Select the implementation for the visualization (Fig. 8). The last step consists of choosing the software library or tool to produce the visualization designed in the previous steps. The goal here is to decide the visualization tool depending on the target audience or the publicity method for the visualization (e.g., research publication, data analysis, web publication), without the need to know about the different software solutions available. Fig. 8 shows an example of a feature model that exposes four well-known visualization libraries: `pgfplots` (Tantau, 2013) for research publications in LaTeX, `matplotlib` (Hunter, 2007) in Python and `ggplot2` (Wickham, 2009) in R for data analysis, and D3 (Bostock et al., 2011) in JavaScript for publication on the web.

4.2. Designing tables

Tables are slightly more forgiving of visual design flaws because tables encode data through the use of verbal language (i.e., text) visually displayed (Few, 2012), thus the design of tables is different of graphs and we encode the best design practices of the design of graphs in another set of feature models.

Step 2. Select the best means to display the information (Fig. 9). For tables, the second step of the visualization design process is to select the best means of displaying structured information. The available

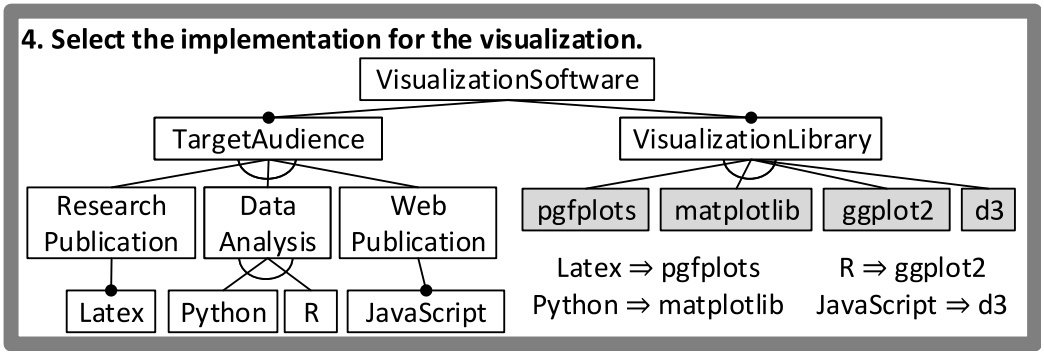


Fig. 8. Step 4 for graphs. Development alternatives for visualization in graphs.

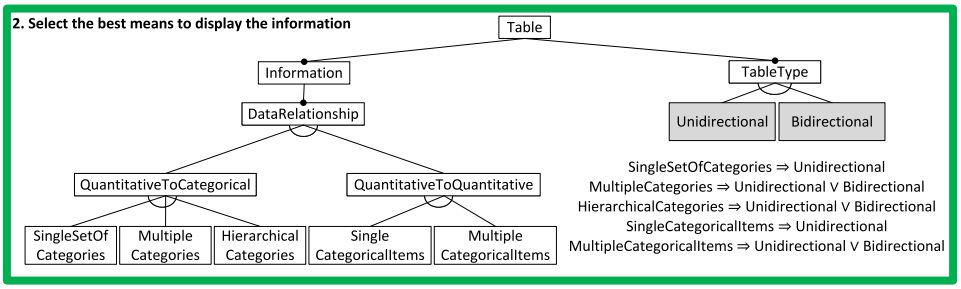


Fig. 9. Step 2 for tables. Feature model to determine the best table layout according to the data relationship.

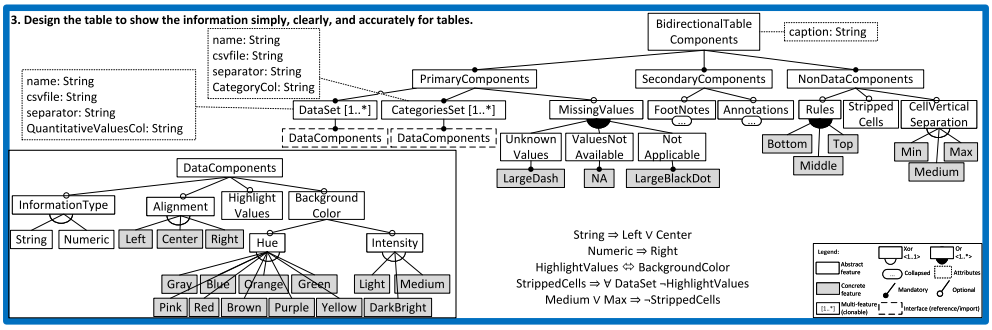


Fig. 10. Step 3 for tables. Variability of the visual component-level table design.

knowledge and best practices for the design of tables extracted from the literature is summarized in Table 2, and has been encoded in the feature model of Fig. 9. The information related to the data in a table can be displayed either (i) quantitative to categorical or (ii) quantitative to quantitative. In case (i), our objective is to display individual values within a category (e.g., to see the total sales of a salesman each month of the year). In case (ii), following the previous example, we visualize two sales of different salespersons each month. Then, we are one-way and two-way categories. In the case of unidirectional tables, this visualization is indicated to visualize data with the quantitative to categorical relationship, as we described in (i). However, although not limited, in the case of comparisons, a bidirectional visualization, where the X-axis (rows) and the Y-axis (columns) are essential, is very suitable for comparisons.

Step 3. Design the display to show the information simply, clearly, and accurately (Fig. 10). As in the graphs, this feature model allows you to configure a table from three main categories: primary components, secondary components, and non-data components. Despite there is not as much variety in tables as there can be in graphs, we have identified some best practices that can be easily coded in our approach. For example, there is one that is alignment. Columns can be aligned left, right and center. However, good practice tells us that columns containing text should be left or center aligned, while numbers, being decimals, should be right-aligned. These constraints are modeled in Fig. 10. In tables, although we do not have any possible overlapping of data as we did in the graphs (e.g., the comparison between two-time sequences, each defined with some components), here, the concept of a dataset can be focused on the columns. Each column has (i) information type, (ii) alignment, (iii) highlight values and (iv) background color. Each columns can have a valid configuration for their dataset. However, it would not make sense to design a stripped table if one of the columns is shaded in its background. In other words, there may be incompatibilities if we study the global configuration of each column compared to the rest. That is why there are restrictions associated with the dataset itself, e.g., the `StrippedCells` feature is not used if there is no dataset with a background of a particular color.

Step 4. Select the implementation for the visualization (Fig. 11). The last step for tables is very similar to the last step for graphs. It consists of choosing the software library or tool to produce the table designed in the previous steps. The goal here is to decide the visualization tool that generating the table depending on the target audience or the publicity method for the visualization (which in fact are the same as for graphs), without the need to know about the different software solutions available. Fig. 11 shows an example of a feature model that

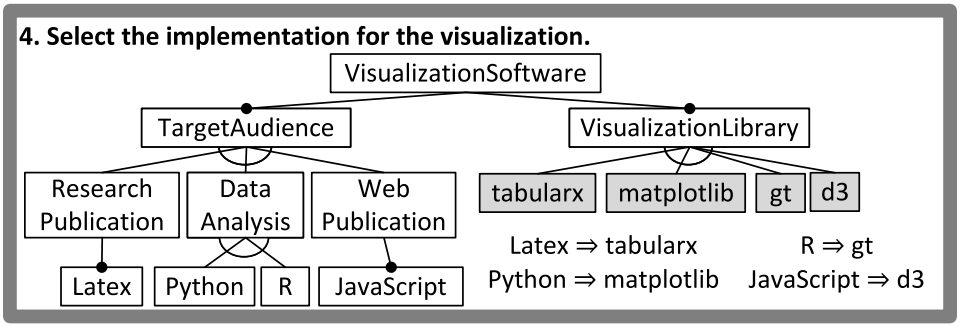


Fig. 11. Step 4 for tables. Development alternatives for visualization in tables.

exposes four alternatives libraries to generate tables: `tabularx` (Tantau, 2013) for research publications in LaTeX, `matplotlib` (Hunter, 2007) in Python and `gt` (Wickham, 2009) in R for data analysis, and D3 (Bostock et al., 2011) in JavaScript for publication on the web. Note that `matplotlib` and D3 alternatives were also available in the fourth step for graphs. These libraries, in addition to graphs, also provides an API to create custom tables. The configuration of the feature model of step 4 for graphs and tables will guide the development of the visualization, as explained in the following section.

Addressing RQ1, the analysis focuses on how the SPL approach benefits non-specialist developers. Step 1 simplifies the choice between graph and table. It reduces initial complexity. Steps 2 and 3 guide selection and configuration and enable the creation of effective visualizations without prior experience. Step 4 assists in choosing the software library and facilitates implementation for developers with limited visualization skills.

Addressing RQ2, the flexibility and representation of data in visualization tools are examined. Step 2 highlights the flexibility of tools with multiple capabilities. It enables more effective data representations. Step 3 emphasizes the importance of diverse configuration options. It increases the clarity and accuracy of visualizations. These steps indicate how versatile tools improve interpretation and user satisfaction.

5. Variability implementation and resolution in visualization

In the solution space of our SPL, we implement the variable visualization artifacts and resolve the variability of those artifacts according to the configurations previously provided, generating the final visualization.

Template-based code generation. In the domain engineering of our SPL (top-right of Fig. 3), we implement and resolve the variability using a *template-based code generation* (Syriani et al., 2018) approach. In a template-based approach, a text-based language (e.g., JavaScript, Python, LaTeX) is enriched with directives to automatically generate custom content. A template system allows us to reuse static elements (e.g., web pages) while defining dynamic elements based on parameters. In visualization, templates support static non-variable content, providing basic structure and appearance, while the dynamic variable content (the visual components and attributes) are specified as parameters. Sections 5.1 and 5.2 detail the template for graphs and tables, respectively.

To map the variability specified in the feature models with the implementation of the visualizations, we define a mapping model that relates the features in the feature models with the variation points and variants in the parameterizable templates. To define such a mapping model, we formalize the concepts of variation points and variants in the context of the visualization artifacts.

Listing 1: Excerpt of the *Jinja2* template for graphs visualizations in LaTeX using the *pgfplots* and *tikz* packages.

```
\begin{tikzpicture}
  \begin{axis}[
    title={{ title }},
    xlabel={{ xaxis_label }}, ylabel={{ yaxis_label }},
    ...
    {{ plot_type }},
    {% if x_log_scale %} xmode=log, {% endif %}
    {% if y_log_scale %} ymode=log, {% endif %}
    {% if legend_pos %} legend pos={{ legend_pos }}, {%
        endif %}
  ]
  {% for plot in plots %}
  \addplot+[
    mark={{ plot.mark }},
    color={{ plot.color }},
    ...
  ]
  table[
    x={{ plot.xaxis_column }},
    y={{ plot.yaxis_column }},
    col sep={{ plot.separator }},
  ]
  { {{ plot.csv_file }} };
  {% if legend_pos %}\addlegendentry{ {{ plot.name }} }{%
      endif %}
  {% endfor %}
  \end{axis}
\end{tikzpicture}
```

Mapping features to variation points and variants in visualization artifacts. Pohl et al. (2005) define a variation point in the context of an SPL as *"a representation of a variability subject within domain artifacts enriched by contextual information"*, and a variant as *"a representation of a variability object within domain artifacts"*. In our case, the domain artifacts are the code templates in charge of generating the visualization (e.g., LaTeX or Python source code), and the contextual information are the directives (e.g., *Jinja2* directives, *#ifdef* directives) that specify the variability subjects (e.g., plot type, color, etc..). The associated variability objects will be every variant (e.g., scatter plot or bar chart for the plot type, or the different color variants). To simplify the formalization of a variation point, we rely on the definition of Jacobson et al. (1997) who define a variation point as *"one or more locations at which the variation will occur"*, and we formally define a variation point and its variants as follows:

Definition 1 (*Variation Point, Variant*)**.** A *variation point* is defined as a 3-tuple (f, h, V):

- f is the feature in the feature model associated with the variation point.
- h is the handler (or identifier) of the location in the artifact where the variation takes place.
- V is the set of possible variants for this variation point. If no variants are specified for the variation point (i.e., $V = \emptyset$), the handler h is processed as a boolean flag (i.e., like a *#ifdef* directive).

A *variant* $v \in V$ is defined as a tuple (ρ, ν):

- ρ is the feature or attribute in the feature model associated with the variant.

Table 3
Mapping model of variation points and variants for graphs.

	Variation point		Variants (V)	
	Feature (f)	Handler (h)	Feature/Attribute (ρ)	Value (v)
Features	ValueEncodingObject	plot_type	ScatterPlot LineGraph VerticalBars HorizontalBars …	only marks smooth ybar xbar …
	XLogarithmic	x_log_scale	–	–
	YLogarithmic	y_log_scale	–	–
	LegendPos	legend_pos	SouthWest SouthEast NorthWest NorthEast OuterNorthEast	south west south east north west north east outer north east
Attributes	VisualComponents	title	VisualComponents.title	(provided)
	XAxis	xaxis_label	XAxis.label	(provided)
	YAxis	yaxis_label	YAxis.label	(provided)
Features of Multi-features	Shape	plot.mark	CircleMark SquareMark TriangleMark PlusMark CrossMark	o square triangle + x
	Color	plot.color	Gray Blue Orange …	gray blue orange …
Multi-feature Attributes	DataSet	plot.name	DataSet.name	(provided)
	DataSet	plot.csv_file	DataSet.csvfile	(provided)
	DataSet	plot.xaxis_column	DataSet.XAxisColumn	(provided)
	…	…	…	…

- v is the concrete value to be substituted/replaced/assigned to the handler h of the variation point. ■

With these definitions, we define a mapping to establish a relation between the feature models specified in Section 4 and the template-based variable artifacts. We define a template and its corresponding mapping for each main display, that is, one for graphs and one for tables.

5.1. Template and mapping for graphs

Listing 1 shows an example of a variable artifact for visualization using a template-based approach, concretely using the *Jinja2*[2] template engine. Variable elements are specified between curly braces and have been highlighted in bold. These variable elements represent variation points and will be replaced by the selected variants in the configurations. We choose a template-based approach in contrast to a pure annotative-based approach (Krüger et al., 2018) for implementing the variability, because the visualization domain exposes a large set of variants for each variation point, and therefore, a pure annotative-based approach (e.g., C preprocessors with *#ifdef* directives) requires introducing a high number of annotations to represent each possible variant. In addition, a template-based approach supports more powerful directives that allow the use of composition-based mechanisms (Apel et al., 2013) such as the substitution and/or replacement of elements, the assignment of values, or the use of control structures such as *if/elif/else*, *for-loops*, *macros*, and *blocks*. For example, the for-loop in Listing 1 (lines 12–25) iterates over the plots that will represent each dataset modeled in the feature model of Step 3 (Fig. 7) as the `DataSet[1..*]` multi-feature. The template engine will generate a copy of the code between lines 12–25 for each instance of `DataSet[1..*]` provided in the configurations.

[2] *Jinja2*: https://palletsprojects.com/p/jinja/.

Listing 2: Excerpt of the *Jinja2* template for template visualizations using the Python *matplotlib* library.

```
1 import matplotlib.pyplot as plt
2
3 fig, ax = plt.subplots()
4 ax.set_title("{{ title }}")
5 ax.set_xlabel("{{ xaxis_label }}")
6 ax.set_ylabel("{{ yaxis_label }}")
7 {% if x_log_scale %} ax.set_xscale('log') {% endif %}
8 {% if y_log_scale %} ax.set_yscale('log') {% endif %}
9 {% for plot in plots %}
10 data = []
11 with open("{{ plot.csv_file }}", 'r') as f:
12     for line in f:
13         parts = line.split("{{ plot.separator }}")
14         x = float(parts[{{ plot.xaxis_column }}])
15         y = float(parts[{{ plot.yaxis_column }}])
16         data.append((x, y))
17 x_data, y_data = zip(*data)
18 ax.plot(x_data, y_data, marker="{{ plot.mark }}", color="
       {{ plot.color }}", label="{{ plot.name }}")
19 {% endfor %}
20
21 {% if legend_pos %} ax.legend(loc="{{ legend_pos }}") {%
       endif %}
22
23 plt.show()
```

Table 3 shows the mapping model of variation points and variants for the *Jinja2* template of Listing 1. For instance, when the feature (e.g., `ValueEncodingObject`) associated with a variation point is selected in a configuration of the feature models, the variant chosen in the configuration (e.g., `ScatterPlot`) will be considered to replace its associated value (e.g., `only marks`) in the handler of the template (e.g., `plot_type`). When the feature is not selected in a configuration, the variation point identified by the handler is removed from the template. The same behavior is applied for those variation points without variants (e.g., `XLogarithmic`). The *Jinja2* template engine receives a map structure of ($handler, value$) pairs for the variation points

Table 4
Mapping model of variation points and variants for tables.

	Variation point		Variants (V)	
	Feature (f)	Handler (h)	Feature/Attribute (ρ)	Value (v)
Features	TableType	table_type	Unidirectional Bidirectional	unidirectional bidirectional
	CellVerticalSeparation	cell	Min Medium Max	min medium max
Features of Multi-features	Alignment	data_component.alignment	Left Center Right	left center right
	Rules	data_component.rules	Bottom Middle Top	bottom middle top
	Hue	data_component.color	Gray Blue Orange …	gray blue orange …
Multi-feature Attributes	DataSet	plot.name	DataSet.name	(provided)
	DataSet	plot.csv_file	DataSet.csvfile	(provided)
	DataComponent	column_title	DataComponent.title	(provided)
	…	…	…	…

whose associated features have been selected in the configurations of the feature models. In case of a multi-feature, the *value* is a list of maps with $(handler, value)$ pairs that can be traversed with the for-loop directive in the template (lines 12–25 in Listing 1).

Listing 2 shows an example of variability for the same input data, but using in this case the *matplotlib* library. Since it is a library used in Python, there are specific lines such as line 1 where it is imported into the script as a module, as well as line 13 to be able to open the file. In the customization aspect, for example, lines 4–6 define some features necessary for understanding such as the title and the names of the X and Y axes. Aspects such as color are defined in line 20.

5.2. Template and mapping for tables

Listing 3 also shows an example of a variable artifact for visualization using a template-based approach, concretely using the *Jinja2* template engine. For example, the for-loop in Listing 3 (lines 12–25) iterates over the data component that will represent each column modeled in the feature model of Step 3 (Fig. 10) as the `DataSet[1..*]` multi-feature. The template engine will generate a copy of the code between lines 12–25 for each instance of `DataSet[1..*]` provided in the configurations.

Table 4 also shows the mapping model of variation points and variants for the *Jinja2* template of Listing 3. For instance, when the feature (e.g., `TableType`) associated with a variation point is selected in a configuration of the feature models, the variant chosen in the configuration (e.g., `Unidirectional`) will be considered to replace its associated value (e.g., `unidirectional`) in the handler of the template (e.g., `TableType`).

6. Implementation and evaluation

In this section, we explore two main topics. First, we delve into the implementation of our Software Product Line (SPL) and its key components. Then, we assess our proposal through a structured evaluation process. This evaluation is pivoted in the usage of our approach in different realistic scenarios and informed by visualization experts to interactively enhance the proposal. Finally, we examine the complexity involved in the configuration process.

6.1. Implementation of the SPL process for visualization

To showcase the applicability of our approach, we have developed an implementation of our SPL for visualization, which incorporates the feature models discussed earlier and includes various configurations that can be used to recreate the visualization scenarios we will describe in Section 6.2.

Listing 3: Excerpt of the *Jinja2* template for template visualizations in LaTeX using the *tabularx* package.

```
\begin{table}[t]
    \caption{ {{ title }} }
    \centering
    \setlength{\tabcolsep}{
    {% if cell %}
        {% if cell == min %} 0.5p {% endif %}
        {% if cell == medium %} 1p {% endif %}
        {% if cell == max %} 2p {% endif %}
    {% endif %}
    }
    \begin{tabularx}{
        {% for data_component in data_components %}
            {% if data_component.alignment %}
                {% if data_component.alignment  == left
                    %} l {% endif %}
                {% if data_component.alignment  == center
                     %} c {% endif %}
                {% if data_component.alignment  == right
                    %} r {% endif %}
            {% else %} l
            {% endif %}
        {% endfor %}
    }

    {% for data_component in data_components %}
        {% if data_component.rules  == top %} \toprule {%
            endif %}
        {% if data_component.rules  == middle %} \midrule
            {% endif %}
        {% if data_component.rules  == bottom %}
             \bottomrule {% endif %}

        {% if loop.index > 0 %} & {% endif %}
        {{ data_component.title }}
    {% endfor %}

    \midrule
    {{ csv_file }}
    \bottomrule
    \end{tabularx}
\end{table}
```

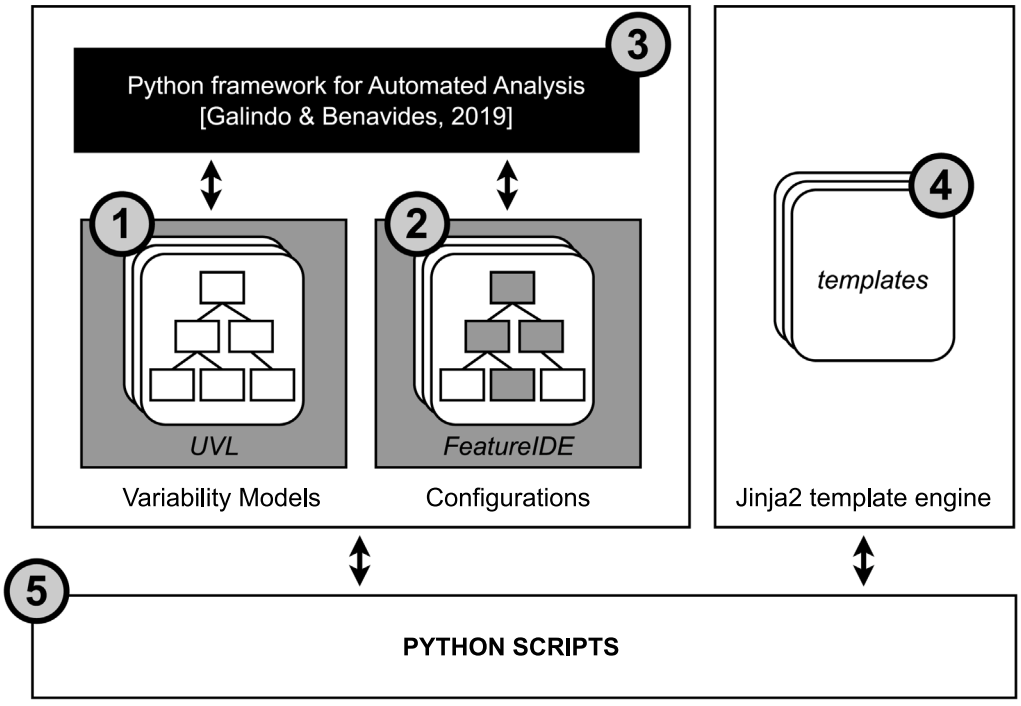


Fig. 12. Architecture used to implement and test our scenarios.

Fig. 12 overviews the main components of our SPL:

1. The variability models (i.e., feature models) that specifies the variability of the visualization design process for each step of our approach. We rely on the Universal Variability Language (UVL) (Benavides et al., 2024) format.
2. The configurations of the feature models for different visualization scenarios. We rely on FeatureIDE (Meinicke et al., 2017) to create those configurations interactively.
3. The Python framework for automated analysis of feature models proposed in Galindo et al. (2023) to manage and deal with the feature models and configurations (i.e., Flama[3]).
4. The *Jinja2* template engine for Python to implement and resolve the variability.
5. A set of Python scripts to manage and orchestrate the feature models and configurations, as well as the mapping according to the appropriate templates.

All artifacts of our SPLs, as well as the resources and visualizations to replicate our experiments in Section 6 are available online.[4]

6.2. Summary of the evaluation

The evaluation of our proposal has been carried out in two phases, as illustrated in Fig. 13. The phases were carried out in a step-wise refinement process which leads to the results presented in this paper. First, the evaluation was carried out by analyzing seven realistic scenarios, four of them focus on different types of graphs and the remaining three focus on tables. Expert feedback is used to inform our proposal and enhance the implementation.

Seven practical scenarios with different quantitative data and messages to be visualized to generate a visualization that takes into account the best design practices to communicate a quantitative message. To do so, we follow the four configuration steps of our approach presented in Section 4 for each of the scenarios.

Table 5
Summary of scenarios covered in the evaluation.

Scenario	Visualization type	Datatype	Data source
Scenario 1	Graph	Cuantitative	The dblp team (2022)
Scenario 2	Graph	Cuantitative	Horcas et al. (2022b)
Scenario 3	Graph	Cuantitative	Benavides et al. (2024)
Scenario 4	Graph	Cuantitative	Dong et al. (2020)
Scenario 5	Table	Cuantitative	The dblp team (2022)
Scenario 6	Table	Cuantitative	Aghababaeyan et al. (2021)
Scenario 7	Table	Cualitative	Rabiser et al. (2019)

Table 5 summarizes the different evaluation scenarios we have conducted. Scenarios 1 to 4 evaluate different types of charts for a particular dataset, while Scenarios 5 to 7 evaluate the visualization of tables. In all scenarios, we indicate whether the information to be visualized is qualitative or quantitative. We also indicate the source of the dataset from which it has been extracted. All data are organic, i.e., not generated by any computer procedure.

Scenarios 1 and 5 focus on generating a visualization from scratch using a quantitative dataset for which a previous visualization does not exist. Therefore, we use our SPL to generate a graph (Scenario 1) and a table (Scenario 5) to transmit a different message for the same dataset. In the later scenarios, we use our SPL to improve existing visualizations in the literature.

In Scenario 2, we focus on selecting the appropriate graph type for a given data relationship (step 2) to select a better alternative for an existing defective visualization. Scenario 3 focuses on the configuration of the visual components and attributes of a specific graph (step 3) to improve an existing defective graph. Finally, Scenario 4 demonstrates the application of our SPL to generate a visualization of a simpler graph that represents the same message as a complicated existing graph, which the visualization experts do not recommend and that is not included in our SPL.

Similarly, for tables, in Scenario 5, we focus on selecting the appropriate tabla type for a given data relationship (step 2), improving

[3] Flama: https://flamapy.github.io/.
[4] https://github.com/diverso-lab/spl_visualization_design

Listing 4: Quantitative data for the visualization of the Scenario 1: Number of papers in proceedings for the SPLC, VaMoS, and ConfWS conferences (extracted from *dblp* (The dblp team, 2022)).

```
Year, SPLC, VaMoS, ConfWS
2000, 27, , ?
2001, , , ?
2002, 24, , ?
2003, , , ?
2004, 42, , ?
2005, 24, , ?
2006, 43, , ?
2007, 27, 21, ?
2008, 64, 17, ?
2009, 48, 23, ?
2010, 67, 30, ?
2011, 58, 11, 7
2012, 41, 22, 9
2013, 49, 21, 16
2014, 52, 23, 13
2015, 65, 16, 21
2016, 51, 14, ?
2017, 37, 15, ?
2018, 49, 17, 19
2019, 54, 17, 13
2020, 41, 25, ?
2021, 37, 19, 13
2022, 60, 15, 17
```

an existing defective table. In contrast, Scenario 6 focuses on the configuration of the visual components and attributes of a table (step 3) to improve an existing defective table. Finally, in Scenario 7, a table visualization is made with various colors suitable for people with color perception problems, such as color blindness.

Regarding expert feedback, two visualization experts have given inputs on the SPL artifacts: the feature model, the templates and the generated visualizations with our implementation. The experts feedback was given using two strategies:

1. *Expert review of student visualization*. One of the authors, who teaches visualization, asked his students to create visualizations of a specific dataset before and after introducing them to good visualization practices. Then, we compared these results with those generated by our tool. In addition, an external visualization expert was consulted to evaluate and compare the three sets of results.
2. *Expert review of templates and results*. Another external visualization expert checked that the templates for constructing the graphs and tables as well as the generated visualizations comply with the heuristic principles described in Section 4. Also the feature model was explained to him (that is not an expert on feature modeling) and checked its validity.

6.3. Evaluation of graphs

Scenario 1: Efficient visualization of quantitative data with graphs. Let us consider the quantitative data (Listing 4) that contain the number of articles published in the proceedings of the SPLC,[5] VaMoS,[6] and ConfWS[7] conferences available in the *dblp computer science bibliography* (The dblp team, 2022). Following the four configuration steps of our approach (see Sections 4 and 4.1), we can generate a visualization for these quantitative data:

1. **Determine the information you need to display.** We are interested in displaying the entire series of values to show how the number of articles have changed through time or identify trends. Therefore, we create a configuration of the feature model of Fig. 5 by just selecting the feature `EntireSeriesOfValues`, which results in the following configuration: Config_FM1 = {`Visualization`, `MessageToDisplay`, `EntireSeriesOfValues`, `Display`, **`Graph`**}. As expected the best communication vehicle for our message is a graph. In the configuration, we have highlighted the concrete features automatically selected (**`Graph`**) and underlined the feature manually selected by the user (`EntireSeriesOfValues`). Others abstract features are also automatically selected due to the feature model relations. and will be omitted in the following to simplify the configurations.
2. **Select the best means to display the information.** We want to display "how the number of articles changes through time", so we can easily identify that the data relationship we want to visualize is a time series. In fact, our message contains the "change" keyword that help us to identify the time series. To create a configuration of the feature model of Fig. 6, we select the `change` feature which results in the selection of the `TimeSeries` feature. At this point, to display a time series, a line graph or vertical bars are preferable (see Table 1). However, our data (Listing 4) contain missing values at some years either because the conference was not celebrated or because the data is not available in the *dblp* (The dblp team, 2022). Thus, we select the `MissingValues` feature, suggesting us that a dot plot or a line graph with points are the most preferable representation. As we are interested in the overall pattern, we also select the `EmphOverallPattern` feature, resulting in the configuration: Config_FM2 = {`Graph`, `Information`, `MissingValues`, `EmphasizeGoal`, `EmphOverallPattern`, `DataRelationship`, `TimeSeries`, `change`, `ValueEncodingObject`, `Lines`, **`LineGraph`**, **`WithPoints`**}; that gives us the line graph with points as the recommended encoding objects to display the information.
3. **Design the display to show the information simply, clearly, and accurately.** In the third step, we configure the visual components for our display (feature model of Fig. 7). We have three datasets that implies to instantiate the multi-feature `DataSet [1..*]` three times, one for each conference (SPLC, VaMoS, and ConfWS), configuring the visual attributes for each dataset differently: DataSet1_SPLC = {`VisualAttributes`, `Form`, `Shape`, `CircleMark`, `Color`, `Hue`, `Blue`}; DataSet2_VaMoS = {`VisualAttributes`, `Form`, `Shape`, `SquareMark`, `Color`, `Hue`, `Red`}; DataSet3_ConfWS = {`VisualAttributes`, `Form`, `Shape`, `TriangleMark`, `Color`, `Hue`, `Brown`}. We also set the `name`, `datafile`, `separator` `XAxisColumn`, and `YAxisColumn` attributes for each dataset accordingly; and select the secondary and non-data components such as the `NorthWest` position for the legend, and set the `title` and `label`(s) for the axes; resulting in the configuration: Config_FM3 = {`VisualComponents.title=''Number of publications in proceedings''`, `PrimaryComponents`, `DataSet1_SPLC`, `DataSet2_VaMoS`, `DataSet3_ConfWS`, `SecondaryComponents`, `Legend`, `LegendPos`, `NorthWest`, `NonDataComponents`, `Axes`, `XAxis.title=''Year''`, `YAxis.title=''Papers''`}. Since the feature model only exposes the recommended variations extracted from the visualization literature, any configuration of the visual attributes will ensure the best design practices for visualization.
4. **Select the implementation for the visualization.** In the last step, we decide to generate the visualization for a research publication in LaTeX. We create a configuration of the feature model of Fig. 8: Config_FM4 = {`VisualizationSoftware`, `TargetAudience`,

[5] SPLC *dblp*: https://dblp.org/db/conf/splc.
[6] VaMoS *dblp*: https://dblp.org/db/conf/vamos.
[7] ConfWS *dblp*: https://dblp.org/db/conf/confws.

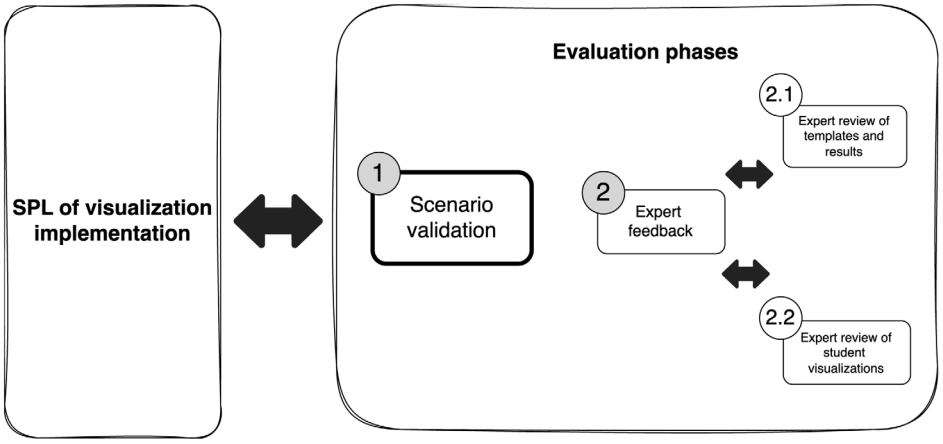


Fig. 13. Evaluation phases.

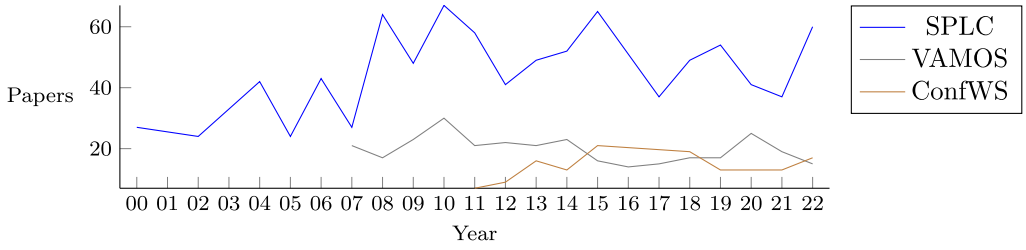


Fig. 14. Visualization generated for the Scenario 1: a lines graph with points representing the time series of Listing 4 (number of publications in proceedings).

`ResearchPublication`, `Latex`, **`pgfplots`**}, which uses the *pgfplots* library, and thus, results in the use of the *pgfplots* template to generate the final visualization.

With the four configurations provided and the *pgfplots* template (Listing 1), we can resolve the variability as explained in Section 5 to generate the visualization of Fig. 14: a time-series graph using a combination of lines and points to effectively encode the quantitative information of Table 1. The lines give the sense of continuity that is required for displays of time, while the points explicitly identify the exact position along the line whether the values are available because our data contain several missing values. We may remove the points and use only the lines in the case of collecting all intermediate values (e.g., years 2001 and 2003 for SPLC, and years 2016, 2017, and 2020 for ConfWS). We may also show individual values as points without connecting them with a line only if the time-series values were not collected at regular intervals of time (e.g., every year).

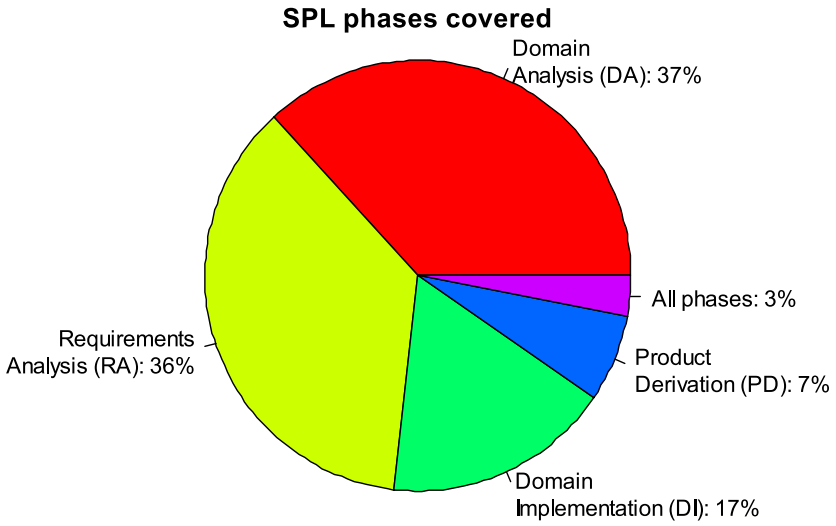
Scenario 2 (Fig. 15). The pie chart in Fig. 15(a) represents a part-to-whole relationship about the tool support for the different SPL processes, extracted from Horcas et al. (2022b). A pie chart is not recommended to represent a relationship between a part and a whole (the tool support for the SPL processes in this case) because differences in the size of the angles are difficult to be perceived, requiring labels to display the values.

To improve this visualization, we use the quantitative data available in Horcas et al. (2022b), and create the configurations that result in the visualization of Fig. 15(b). The recommended display for a part-to-whole relationship is a bar graph; or a line graph, in case of showing how parts of a whole have changed over time, which is not the case. The reason to prefer encoding a part-to-whole relationship as bars in contrast to a pie chart is that people can see the differences easily in the bar graph but with difficulty in the pie chart because the visual perception is highly tuned for seeing differences among the lengths of objects that share a common baseline but not well tuned for discerning difference among 2D areas (Few, 2012). For example, we cannot easily compare how much bigger is the "Domain Analysis (DA)" part from the "Requirements Analysis (RA)" part in the pie chart (Fig. 15(a)) without looking at the numbers on the textual labels. In the bar graph (Fig. 15(b)), we have manually chosen horizontal bars in contrast to vertical bars because the categorical labels are too long to fit side by side in vertical bars.

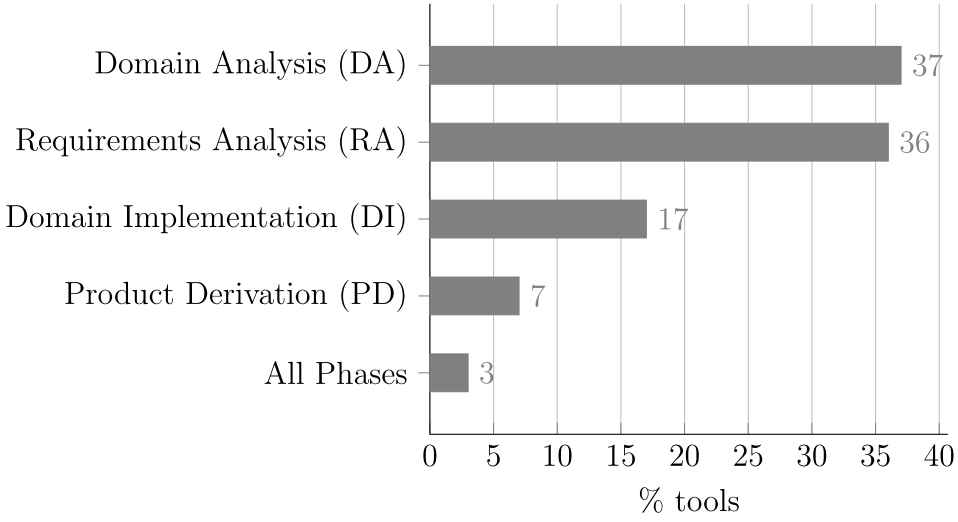
Vertical grid lines, as secondary components in the background, also facilitate quantitative comparison between parts. Finally, colors do not contribute to this visualization and may distract (the original pie chart even uses the red and green colors together); thus, we opt to use the same color for all datasets.

Scenario 3 (Fig. 16). The visualization in Fig. 16(a) illustrates a correlation relationship between the file size and different formats of feature models, extracted from Benavides et al. (2024). In fact, a scatter plot is the preferred display to show a correlation relationship; however, the original visualization suffers from several problems: (1) the visual attributes of the original scatter plot present some flaws such as an *over-plotting* problem (Few, 2012) (i.e., points are hidden or overlapped by other points to the degree that they cannot be distinguished); (2) it uses shapes with the same form that are difficult to distinguish from each other; and (3) it uses different scale units on the Y axis (e.g., mixing KB and MB). Such a perceptual problem (over-plotting) can be minimized by using shapes that are easiest to distinguish from one another like ○, □, △, +, or ×, even without colors. We use the quantitative data available in Benavides et al. (2024) to configure and recreate the display, which results in the visualization of Fig. 16(b). First, we configure different shapes that are easiest to distinguish from each other, in contrast to using the same mark for five of six datasets as in the original display. Second, we enlarge the graph, decrease the size of the points, and keep the interior of the points empty of fill color to remedy the over-plotting problem. Finally, we use a logarithmic scale to maintain the same unit on the vertical axis.

Scenario 4 (Fig. 17). In this scenario, we apply our approach outside the SPL domain. Consider the visualization of Fig. 17(a) showing the total incidence of Covid-19 cases per region in Spain, extracted from



(a) Original visualization (taken from Horcas et al. (2022b)): a defective pie chart.



(b) Visualization generated with our approach: a bar chart.

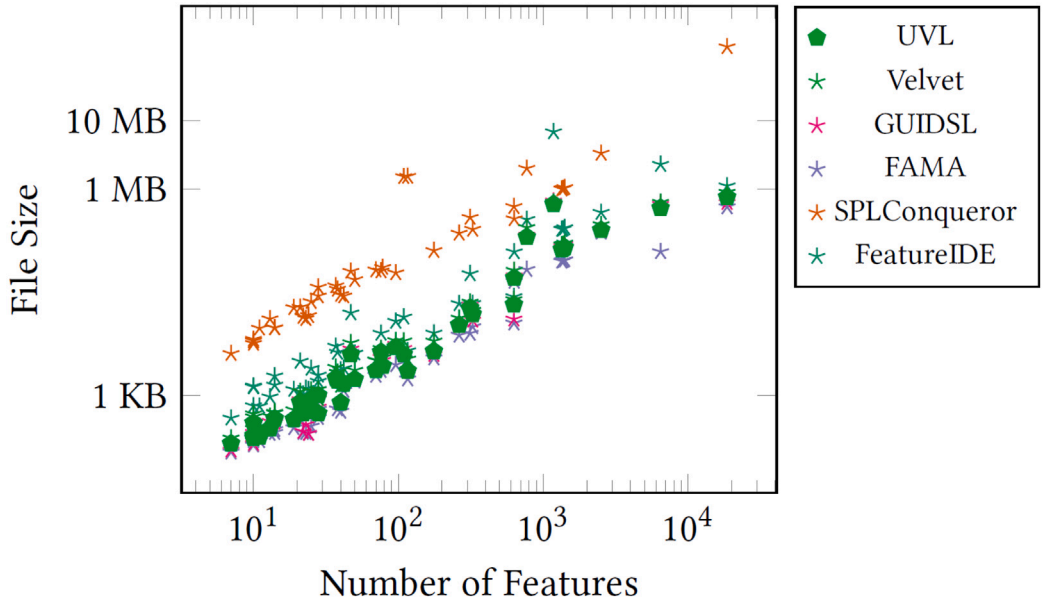
Fig. 15. Scenario 2: Visualization of a part-to-whole relationship.

JHU CSSE COVID-19 Data (Dong et al., 2020). The bubble graph map encodes the quantitative values as points of different sizes, which, despite being perceived as fancy and pretty, is not effective in transmitting the message about which region has more incidence than others or in making comparisons between regions (Cleveland and McGill, 1985). A nominal comparison like this is better represented with a bar graph. The configurations of our feature models generate the visualization of Fig. 17(b). Since we have many regions (19), horizontal bars are more appropriate to fit the regions' labels than vertical bars. In addition, we maintain the same color for all bars as appears on the map and rank the regions by quantities to facilitate comparisons. We can observe on the bar graph (Fig. 17(b)) that the incidence in Cataluña (2,36 million) is much higher than in Madrid (1,64 million), but this difference is not appreciated on the map of Fig. 17(a).

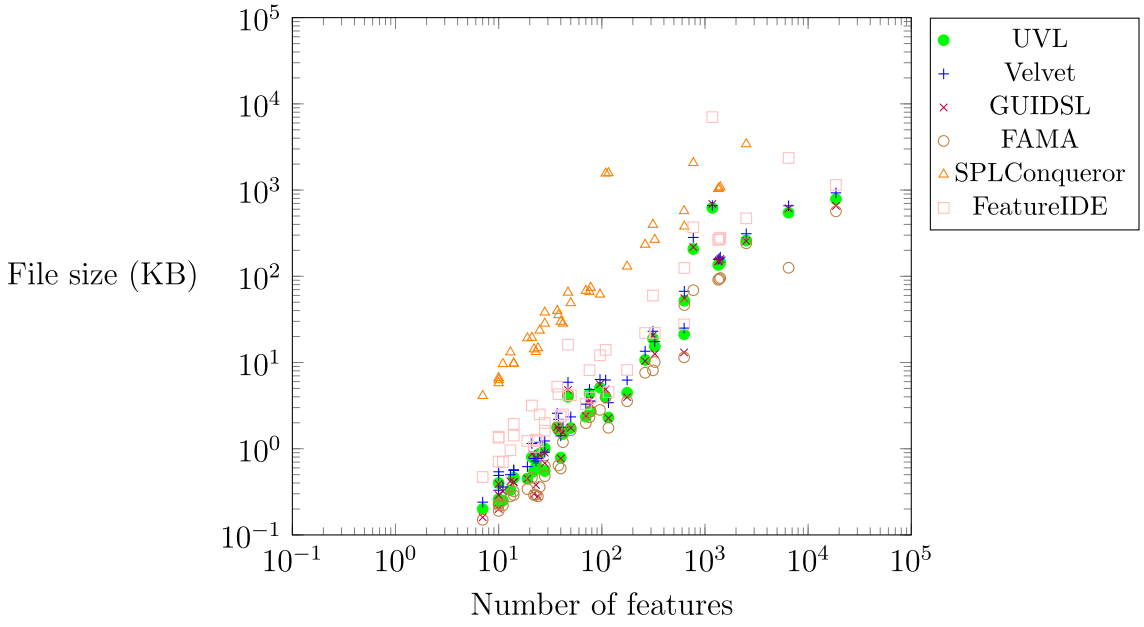
6.4. Evaluation of tables

Scenario 5: Efficient visualization of quantitative data with tables. Considering again the quantitative data (Listing 4) of the Scenario 1. We can represent the details of the data (i.e., the specific values for individual comparison) using a table. Following the four configuration steps of our approach (see Sections 4 and 4.2), we can generate a table for these quantitative data:

1. **Determine the information you need to display.** In this case, we are interested in comparing individual values for each year and conference. Therefore, we create a configuration of the feature model of Fig. 5 by just selecting the feature `CompareIndividualValues`, which results in the following configuration: Config_FM1 = {`Visualization`, `MessageToDisplay`, `CompareIndividualValues`, `Display`, **`Table`**}. In contrast to a graph, using a table is difficult to take a overall pattern of the data. However, here the message we want to transmit is different, and a table is the best communication vehicle for our data when we want to compare individual values.
2. **Select the best means to display the information.** We want to display a quantitative (i.e., the number of papers) to categorical (i.e., conferences) data relationship consisting in a single set of categories (i.e., the years). To create a configuration of the feature model of Fig. 9, we select the `SingleSetOfCategories`. In fact, the best structure of columns and rows for a single set of categories is a unidirectional table as encoded in the cross-tree constraints of the feature model. The complete configuration is as follows: Config_FM2 = {`Table`, `Information`, `QuantitativeToCategorical`, `SingleSetOfCategories`, `TableType`, **`Unidirectional`**.



(a) Original visualization (taken from Sundermann et al. (2021)): a scatter plot with several visual problems.

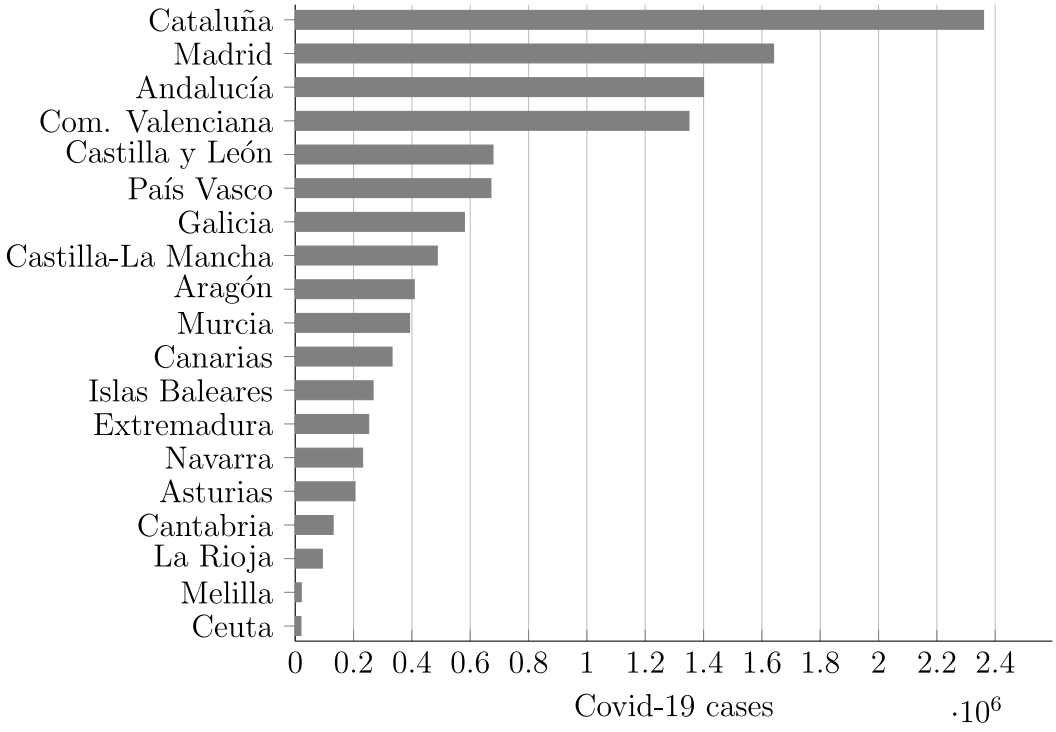


(b) Visualization generated with our approach: an improved scatter plot.

Fig. 16. Scenario 3: Visualization of a correlation relationship (file sizes of different formats for feature models).

3. **Design the display to show the information simply, clearly, and accurately.** In the third step, we configure the visual components for our table (feature model of Fig. 10). As in the graph, we have three datasets that implies to instantiate the multi-feature `DataSet [1..*]` three times, one for each conference (SPLC, VaMoS, and ConfWS), configuring the visual attributes for each dataset differently. However, in this case, we do not need to distinguish or highlight in the table a specific dataset with different colors because the natural alignment of the columns clearly differentiates the values of each dataset. Moreover, the information of the three dataset is numeric, and the values should be aligned to right. Thus, we configure the three dataset identically: DataSet1_SPLC = DataSet2_VaMoS = DataSet3_ConfWS = {`DataComponents`, `InformationType`, `Numeric`, `Alignment`, `Right`}. We also set the `name`, `csvfile`, `separator`, and `QuantitativeValuesCol` attributes for each dataset accordingly. In addition, we have a second multi-feature `CategoriesSet [1..*]` which in this case we only need to instantiate ones for the single set of categories (i.e., the years) since our table is unidirectional: CategoriesSet1_Year = {`DataComponents`, `InformationType`, `String`, `Alignment`, `Left`}. Moreover, our data contains missing values due to two reasons: (1) the conference was not celebrated in a specific year, thus, we select the `NotApplicable` feature; or (2) the *dblp* library does not contains such information, so the values are unknown. Take into account that the values may be available in other academic source, but not in *dblp*, so we select the `UnknownValues` feature. For the secondary components configuration, in particular the orientation of the table is automatically selected to `Vertical` as indicated by the first cross-tree constraints that establishes that if we have more than 5 values for a category in our data, a vertical orientation is

(a) Original visualization (taken from *JHU CSSE COVID-19 Data* (Dong et al., 2020)): a bubble graph map.



(b) Visualization generated with our approach: a bar chart.

Fig. 17. Scenario 4: Visualization of a nominal comparison (Covid-19 incidence in Spain per Region).

recommended (we have 23 values in out category, one per each year to represent). Finally, we configure the non-data components by selecting the three available rules: the `Top` and `Bottom` rules to delimit the table and the `Middle` to distinguish the header from the quantitative values. No other rules are needed because of the natural arrangement of rows and columns of tables, and in fact, the feature model does not allows to configure any other rules or vertical lines maintaining a low non-data ink ratio as recommended by the best design practices of tables (Few, 2012). The final configuration is: Config_FM3 = {`TableComponents.title=''Number of publications in proceedings''`, `PrimaryComponents`, `DataSet1_SPLC`, `DataSet2_VaMoS`, `DataSet3_ConfWS`, `MissingValues`, `UnknownValues`, **`LargeDash`**, `NotApplicable`, **`LargeBlackDot`**, `SecondaryComponents`, `Orientation`, **`Vertical`**, `NonDataComponents`, `Rules`, `Bottom`, `Top`, `Middle`}.

4. **Select the implementation for the visualization.** In the last step, we decide to generate the table for a research publication in LaTeX. We create a configuration of the feature model of Fig. 11: Config_FM4 = {`VisualizationSoftware`, `TargetAudience`, `ResearchPublication`, `Latex`, **`tabularx`**}, which uses the *tabularx* library, and thus, results in the use of the *tabularx* template to generate the final visualization.

With the four configurations provided and the *tabularx* template (Listing 3), we can resolve the variability to generate Table 6 which effectively presents the quantitative values with the recommended structure and rearrangement.

In the following scenarios, we use our approach to improve existing visualizations found in the literature by fixing their flaws and applying

Table 6
Table generated for the Scenario 5: a unidirectional tables with the quantitative values of Listing 4.

Year	SPLC	VaMoS	ConfWS
2000	27	•	–
2001	–	•	–
2002	24	•	–
2003	–	•	–
2004	42	•	–
2005	24	•	–
2006	43	•	–
2007	27	21	–
2008	64	17	–
2009	48	23	–
2010	67	30	–
2011	58	21	7
2012	41	22	9
2013	49	21	16
2014	52	23	13
2015	65	16	21
2016	51	14	–
2017	37	15	–
2018	49	17	19
2019	54	17	13
2020	41	25	–
2021	37	19	13
2022	60	15	17

–: Unknown values (in *dblp*).
•: Not Applicable.

the best design practices encoded in our SPL. For each scenario we show the original visualization and the new visualization generated following our approach.

Scenario 6 (Fig. 18). This scenario presents a table comparing the average accuracy of two algorithms when clustering with two neural networks, extracted from Aghababaeyan et al. (2021). The original table (Fig. 18(a)) exposes several design issues we solve in the new table generated using our approach (Fig. 18(b)). First, the original table represents a data relationship containing multiple categories using a unidirectional table which is acceptable as explained in the best design practices (see Step 2 of our approach in Section 4.2). However, since the goal of the table is comparison, a bidirectional table is more suitable since it is easier for the human eye to compare data from left to right than from top to bottom. Instead of two separate tables, as in the original one, presenting the same information, we now have a single bidirectional table. The rows represent distinct clusters, while the columns are divided into two groups: (i) *Accuracy on the Retaining Cluster* and (ii) *Average Accuracy on the Other Clusters*. Second, the bidirectional table is more compact requiring less space to present the same information because the unidirectional table duplicates information (e.g., cluster categories and the average row). Furthermore, the original table adds to the visual clutter by using white space to separate the header from the data and the row from the average. Third, the natural alignment of the columns facilitates reading and data analysis (principle of proximity (Few, 2012)), thus we can reduce unnecessary vertical and horizontal rules (non-data ink) and enhance the relevant information (i.e., the categories and values). Finally, to improve readability, we have added two decimals to all floating numbers since all values of the same column should have the same precision, and the numerical columns are right-aligned enabling the easy identification of the whole and decimal parts. In addition, we removed the percentage symbol (%) showing it at the header as it was repetitively used throughout the table and all values are the same type.

Scenario 7 (Fig. 19). In this scenario, we improve the configuration of the visual components and attributes of a table. Concretely, we present an example of bad color selection for a data group presented in a table (Fig. 18(a)), extracted from Rabiser et al. (2019). Individuals with color vision deficiencies, such as anomalous trichromacy or dichromacy (Sun et al., 2018), may have difficulty distinguishing between the red and blue rows due to limitations in the human eye's ability to perceive specific colors. This problem can occur in various visualizations, particularly when depicting positive results (green) versus negative results (red). To address this issue, it is recommended to use a greyscale, such as the one shown in Fig. 18(b), in which green is replaced with white, orange is replaced with a greyness of 0.9, and red is replaced with a greyness of 0.8. This greyscale is often the most effective color configuration for individuals with color vision deficiencies. In addition, unnecessary lines have been removed, as well as a blank space to make it easier to understand and so that the white background of the header cell does not confuse with a category cell.

6.5. Complexity of the configuration process

Table 7 shows the complexity exposed by the feature models in our SPL in order to configure them and generate a visualization. We measure the complexity from two viewpoints: (1) the modeling of the feature models encoding the recommendations and best practices from the visualization experts and (2) the configuration of the feature model by the user. In the first case, we use the number of features and constraints that encode the visualization knowledge and the resulting number of possible configurations and final concrete products (visualizations). In the second case, we use the number of decisions or choices that the user needs to make to configure a visualization and the possible alternatives for each decision that the user needs to consider. The number of decisions corresponds with the number of variation points in the feature model. In contrast, the number of alternatives corresponds with the variants (children features) of each variation point, which can also be additional variation points. Table 7 shows the minimum and maximum number of choices the user must make and the average of alternatives that need to be considered in each step of the visualization design process. For example, the feature model of Step 2 for graphs (Fig. 6) exposes 37,706 configurations that result in only 9 possible different visualizations (i.e., the types of graphs recommended by the experts). To obtain one of these visualizations, the user only has to make from 4 to 19 decisions, and in each of these decisions, the user has to decide between 4 possible alternatives on average. The benefit of using our approach is more evident when the feature model increases in size and complexity but not the configuration of the model, as shown in Step 3 for graphs (Fig. 7). Here, the feature model exposes more than 12 million configurations with a total of 23,040 concrete visualizations for the visual components of a graph. However, the user is only aware of about 20 decisions in the worst case (in case the user wants to configure all possible options), and for each decision, there are two possible alternatives on average (Table 7).

Addressing RQ1, complexity analysis highlights how SPL can simplify the visualization process for developers with limited skills. The user decision-making process is streamlined even with large feature models. SPL's approach is practical for developers unfamiliar with the proper principles and heuristics.

Addressing RQ2, discussing the number of possible configurations and visualizations offered by the SPL approach demonstrates the flexibility of multiple visualization tools. This approach provides a broader range of data representation options compared to limited capability tools. This evaluation addresses RQ2's central concern about flexibility and efficiency in data representation.

7. Limitations and discussion

In this section we discuss about the application of software product line techniques to data visualization.

The effectiveness of visualizations cannot be definitively determined as there are various types available (Evergreen, 2019), and their usefulness depends on the specific purpose they are intended for Ware (2019).

Model	Faults C_i	Accuracy on the retraining cluster C_i	Average Accuracy on the other clusters $C_{j\neq i}$
12-layer ConvNet	Cluster 1	80%	28.23%
	Cluster 2	60%	27.42%
	Cluster 3	55%	27.45%
	Cluster 4	71.40%	27.06%
	Cluster 5	66.70%	27.40%
	Cluster 6	60%	27.95%
	Cluster 7	58.33%	27.76%
	Cluster 8	67.70%	28.60%
	Cluster 9	66.30%	27.74%
	Cluster 10	61%	27.80%
Average		**66.64%**	**27.74%**
ResNet20	Cluster 1	77%	30%
	Cluster 2	75%	29%
	Cluster 3	74%	28%
	Cluster 4	62.5%	31%
	Cluster 5	65%	29%
	Cluster 6	56.5%	28.70%
	Cluster 7	78.40%	30.3%
	Cluster 8	75%	29.9%
	Cluster 9	62.5%	31%
	Cluster 10	64%	31%
Average		**69%**	**29.80%**

(a) Original table (taken from Aghababaeyan et al. (2021)) with several design flaws.

Faults C_i	Accuracy on the retraining cluster C_i (%)		Average Accuracy on the other cluster $C_i \neq j$ (%)	
	12-layer ConvNet	ResNet20	12-layer ConvNet	ResNet20
Cluster 1	80.00	77.00	28.23	30.00
Cluster 2	60.00	75.00	27.42	29.00
Cluster 3	55.00	74.00	27.45	28.00
Cluster 4	71.40	62.50	27.06	31.00
Cluster 5	66.70	65.00	27.40	29.00
Cluster 6	60.00	56.50	27.95	28.70
Cluster 7	58.33	78.40	27.76	30.30
Cluster 8	67.70	75.00	28.60	29.90
Cluster 9	66.30	62.50	27.74	31.00
Cluster 10	61.00	64.00	28.78	31.00
Average	**66.64**	**69.00**	**27.74**	**29.80**

Largest value of the column

(b) Table generated with our approach: an effective bidirectional table.

Fig. 18. Scenario 6: Visualization of a data relationship with multiple categories.

For instance, visualizations that employ length and direction are usually easily understood by people and can be a good option (Todorovic, 2008). However, visualizations based on area, volume, and curvature are generally not effective since these features are challenging for humans to comprehend. Typically, incorporating numbers and legends are necessary to facilitate the interpretation of graphs that rely on these characteristics (Freixa Font et al., 2021)

7.1. Visualization process and challenges

Experts in visualization are primarily concerned with the message they intend to convey, and they must consider how the intended audience will perceive and understand the information presented. This involves taking into account various factors such as psychophysics, cognition, and semiotics. To achieve this, a team of experts with diverse backgrounds is often necessary (Bigelow et al., 2014), resulting in multidisciplinary projects that involve multiple teams with varying expertise. The visualization process begins with data that needs to be analyzed and interpreted by others, which then serves as input for an iterative process of generating visualizations that are communicated back to the user. Each phase of this process necessitates effective communication between different teams possessing complementary skills. Concretely, the visualization process consists of three main phases:

Top Well-investigated	Top Further Work Needed	Top Interested
Product Configuration/Derivation	*PL Learning and Adoption, Inst., Training*	*Maintenance / Evolution*
Variability Modeling	***Validation (Testing)***	*PL Implementation*
Variability Management	***Maintenance / Evolution***	*PL Learning and Adoption, Inst., Training*
Variability Analysis / Model checking	Ecosystems / Multi PLs	*Validation (Testing)*
Variability-Asset Relation/Mapping	*PL Requirements Engineering*	***Assets (reqts, arch., code, ...)***
Bottom Well-investigated	**Bottom Further Work Needed**	**Bottom Interested**
PL Learning and Adoption, Inst., Training	Others	***Financing / Economics***
Deployment, Release Mgmt & Operation	Variability Modeling	***Business Activities (PL Marketing, etc.)***
Ecosystems / Multi PLs	Product Configuration/Derivation	***PL Organization***
Maintenance / Evolution	PL Tools	***Runtime Adaptation / DSPLs***
PL Metrics	Variability Management	*PL Tools*

(a) Original table (taken from Rabiser et al. (2019)) with a bad combination of colors.

Top Well-investigated	Top Further Work Needed	Top Interested
Product Configuration/Derivation	PL Learning and Adoption, Inst., Training	Maintenance / Evolution
Variability Modeling	Validation (Testing)	PL Implementation
Variability Management	Maintenance / Evolution	PL Learning and Adoption, Inst., Training
Variability Analysis / Model checking	Ecosystems / Multi PLs	Validation (Testing)
Variability-Asset Relation/Mapping	PL Requirements Engineering	Assets (reqts, arch, code, ...)
Bottom Well-investigated	**Bottom Further Work Needed**	**Bottom Interested**
PL Learning and Adoption, Inst., Training	Others	Financing / Economics
Deployment, Release Mgmt & Operation	Variability Modeling	Business Activities (PL Marketing, etc.)
Ecosystems / Multi PLs	Product Configuration/Derivation	PL Organization
Maintenance / Evolution	PL Tools	Runtime Adaptation / DSPLs
PL Metrics	Variability Management	PL Tools

Topics where survey answers correspond with literature
Topics with some differences with literature
Topics with a large divergence with literature

(b) Table generated with our approach: an appropriate visualization for colorblindness people.

Fig. 19. Scenario 7: Visualization of a data group in a colored table. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 7
Complexity of the feature model configuration process to generate a visualization with our SPL approach. For the feature model in each step we show the number of features ($\#F$), the number of constraints ($\#CTC$), the number of configurations ($\#Configs$) and final products variants ($\#Products$), the minimum ($\#Choices_{min}$) and maximum ($\#Choices_{max}$) number of decisions the user need to do to configure it, and the average number of alternatives in each decision the user must consider ($\#Alternatives_{avg}$).

Feature model	F	$CTCs$	$Configs$	$Products$	$Choices_{min}$	$Choices_{max}$	$Alternatives_{avg}$
Step 1. Determine the information you need to display.							
FM 1 (Fig. 5)	12	7	7	2	1	2	4.5
Step 2. Select the best means to display the information.							
FM 2: Graphs (Fig. 6)	86	19	37,706	9	4	19	4.05
FM 2: Tables (Fig. 9)	13	5	8	2	3	4	2.25
Step 3. Design the display to show the information simply, clearly, and accurately.							
FM 3: Graphs (Fig. 7)	50	1	12,752,640	23,040	8	20	2.15
FM 3: Tables (Fig. 10)	73	5	979,776	10,368	11	23	1.87
Step 4. Select the implementation for the visualization.							
FM 4: Graphs (Fig. 8)	14	4	4	4	2	3	3.00
FM 4: Tables (Fig. 11)	14	4	4	4	2	3	3.00

- Data characterization: the design team examines and characterizes the data that will be utilized in the visualization. This includes comprehending the datatypes implemented and identifying the characteristics that need to be represented. Typically, the initial step of this phase involves data wrangling (Furche et al., 2016) and data reduction techniques (Liang et al., 2021) to refine the raw data.
- Design: Once the initial data has been characterized and comprehended, the responsible practitioners initiate an iterative process (Hehman and Xie, 2021) to generate the visualization. This process comprises creative design, which may involve sketching or manual illustration tools, and data mapping that associates the data with the appropriate representation. The outcome of this process is a visualization concept that is then evaluated by the client. Typically, a team with knowledge of the perceptual scalability (Yost and North, 2006) of the visualization is necessary during this phase to ensure that the resulting product presents the data in a comprehensible manner (Freixa Font et al., 2021). Moreover, systems may be employed to facilitate decision-making by producing various visualizations.
- Development: During the ultimate phase of the visualization process (Bai et al., 2010), the software developer team assumes the responsibility of implementing the visualization that has been designed using a tangible framework that disseminates the information in the desired format. It is in this stage that popular software visualization libraries, such as Grafana, D3.js, and others, are employed.

In the visualization process, numerous challenges have been recently identified (Walny et al., 2019). In the subsequent text, we outline some of these challenges that we think that this paper helps and then clarify how we think that this is helping on tackling those challenges

The first challenge relates to the rapid data change that necessitates constant adaptation. With the proliferation of big data, data is frequently regenerated in real-time, leading to a significant impact on

the entire visualization process and subsequently affecting the quality of the final product. Techniques that permit the reuse of existing data characterization are essential to avoid the re-application of all the data wrangling and data reduction techniques each time a new version of the data is released. Additionally, mechanisms to represent such interaction are required to enable the visualization of the safe-path evolution of the data. With this regard we have tried to encapsulate and automate the whole process of deriving a new visualization. This can help on creating or update the visualization with new data as soon as its fed in the process.

The second challenge pertains to testing interactive inputs, which involves different teams with varying skill sets designing and developing the visualization. This complexity makes it difficult for designers to anticipate and test all potential combinations of interactive inputs that a visualization may receive. With regards to this challenge we help by enabling the constraint propagation between levels of abstractions w.r.t. a visualization.

Finally, the third challenge involves the existence of constraints that are unknown to the entire team during the visualization design. This usually leads to multiple revisions of the visualization, which can result in unwanted costs. We posit that techniques on variability modeling and feature interaction can help provide a more coherent process in which the number of iterations, and thus the costs, are reduced. With out approach we are helping the communication within the teams as all the constraints are clearly specified and shared across.

Our proposal based on four feature models focuses on the fact that each model is managed by a different design team within a single organization. That is, steps 2 and 3 described in Section 4 do target users with some experience in visualization best practices. However, steps 1 and 4 can be managed by a team that is not necessarily an expert in visualization, such as a company's marketing team.

7.2. Cost of SPL implementation

In this context, the implementation of the SPL benefits the communication between the different teams involved in a visualization; however, depending on the size of the visualization project and the importance of the audience, it may not be worth it. This implementation is similar to the convenience or not of building a product line; a visualization product line would be interesting in such cases when designing the project requires engineering work beforehand.

Another point to consider when applying this technique is identifying the product line's implementation strategy. As mentioned in the paper, we rely on template-based implementation for this technique. However, we could rely on considering others such as *feature flags* (Meinicke et al., 2020) (that could be more convenient for other libraries such as *D3* and *JavaScript*) or compiler directives to implement the product line at a code level.

Recalling the knowledge extraction process in Section 4, we note that the first feature model, *Determine the information you need to display* is cross-cutting to graphs and tables. Then there are three other feature models for graphs and three others for tables, thus completing the feature modeling process. When implementing a new template, either for graphs or tables, the cost is similar: it would be necessary to study which points of variability of that template can be fitted into the current features of each of the three models. Possibly, since many libraries share a similar way of working (Joshi and Tiwari, 2023; Vučetić et al., 2023), several features can be reused (e.g. the title and the name of the axes) and only have to add those that are specific to the template we want to add.

7.3. Product building limitation

Although it is not mandatory, our proposal is more in line with a top-down strategy, i.e., we start from specifications, which are the visualization requirements. By propagating configurations through the four feature models, we build the final artifact. The artifact is typically the instance of a visualization library in a particular language (based on our configuration). This is what we know as a top-down approach, as opposed to other bottom-up approaches (Ouali, 2021), where we try to infer the feature model(s) from existing software by refactoring code and detecting bad smells.

However, the product building of our approach relies exclusively on a set of templates that have been verified and validated by a visualization expert. The final artifact is limited to the templates defined by the visualization domain expert. These templates are generic, i.e., the valid SPL configuration results in the insertion of variability points that generate the visualization as discussed above in Section 3. However, if the dataset has specific characteristics not contemplated in the features of our SPL, the generated visualization cannot address that specificity.

Our proposal can generate visualizations in a reduced set of tools and libraries where the dataset presents low granularity, i.e., simple visualizations that can be modeled with our variability points. The four models we represent in the four phases cannot cover all possible casuistry and specificities of a dataset visualization. We offer a best-practice artifact that saves time and cost and is effective, but sometimes does not meet all customer needs.

7.4. Heuristic principles and best practices

Our approach is based on the use of heuristics that follow best practices for the correct visualization and understanding of the data. The most important heuristics are listed in our reference text, "*Show me the numbers*" (Few, 2012)

- *Maximize data/ink ratio*. This heuristic proposed in Tufte (1985) suggests that the amount of ink present in a visualization should be the minimum possible, where what should stand out the most is the data. Any visual element only in decoration mode distracts the reader and makes it difficult to understand the data. An example of this ratio can be seen in the visualization of tables, where there is an excessive use of horizontal and vertical lines to organize information when it is not strictly necessary.
- *Use of color*. In Simmon (2013) we encounter the principle of the proper use of color. Colors are often a widely used resource to highlight important information, indicate similarities and differences, and provide a clear message for the user to assimilate. However, inappropriate use of color, either by mismatching colors or trying to highlight parts that might otherwise be resolved, can distort the results and lead to erroneous conclusions. This use of color is found in graphics, where there are comparisons of elements, where there are colors that are not suitable for people with visual impairments or colors that are not easily distinguishable.
- *Data representation*. In the principle proposed in M. Wong (2010), we are advised that data representation benefits and facilitates the understanding of data. That is, it is the visualization itself that acts as the narrator of the story. An example is found in the use of timelines, where we will use the visual representation of a line chart instead of a bar chart. This decision is made because the X-axis of a graph is commonly used to represent the progression of time. Using another type of visualization may not be as effective.
- *Context*. The principle of Cairo (2012) suggests that visualizations should be placed in a clear context so that they can be understood without adding additional descriptions, which would make for poor visualization. An example of this can be found in using legends and labels in graphs or descriptive headings in tables.

These four fundamental principles are the foundation on which our approach is based. All the features of our models are extracted from these principles and from an exhaustive analysis of the visualization tools that support our SPL. These principles have been mapped to the points of variability, e.g. colors, dividing lines, minimum ink usage, to name a few. The use of these principles, the analysis of the libraries and the design of the feature models are what sustains the basis of the design of the templates that generate the final visualization artefact.

RQ1 examines the effectiveness of the SPL approach versus manual creation for developers without specialized visualization skills. Relevant aspects include (1) the complexity of data visualization and the potential of SPL to simplify the process for those not specialized in the field; (2) the benefit of SPL to improve communication between different teams and simplify the visualization design process, as discussed in Section 7.2; and (3) the reliance on templates verified and validated by visualization experts in the SPL approach, which could help non-specialist developers create compelling visualizations, as mentioned in the discussion of Section 7.3.

7.5. Threats to validity

Even though the experiments presented in this paper provide evidence that the proposed solution is valid, we made some assumptions that may affect their validity. This section discusses the different threats to validity that affect the evaluation (Wohlin et al., 2012).

7.5.1. External validity

The inputs used for the experiments presented in this paper were either realistic or designed to mimic realistic feature models. In this case, we reviewed several papers summarizing the basic heuristics for data visualization and human perception and extracted some of the data. However, since it was developed using a manual design process, it may have errors and not encode all possible configurations. The major threats to the external validity are the following:

- *Population validity*. The feature models that we created may not consider all possible visualization heuristics. To reduce these threats to validity, we consulted multiple experts in the master class the authors give at the University of Seville and reviewed multiple papers and books in a non-systematic way.
- *Ecological validity*. While external validity generally focuses on generalizing the results to other contexts (e.g., using other models), ecological validity focuses on possible errors in the experiment materials and tools used. To prevent ecological validity threats, such as wrong encodings and the like, we relied on the Flama framework (Galindo et al., 2023; Galindo and Benavides, 2020), which has been tested using multiple known techniques, such as metamorphic relationships (Segura et al., 2011).
- *Lack of visualization experts*. A critical aspect of our proposal is whether the evaluation will always be carried out by a team of visualization experts. It is not always easy to find users with this profile, so our SPL relies on heuristics and visualization design principles (see Section 7.4) with which we validate our visualizations. All of our templates have been validated by a visualization expert, but it certainly requires an in-depth review of generated graphs and tables with a number of minor modifications between them to corroborate that they are the best way to represent and make the datasets understandable.
- *Evaluation with students*. In Section 6.2, we mentioned using students to compare manual and automatic graphs of a dataset, which may be simplistic. We propose to use 8–10 varied datasets, make visualizations before and after learning basic principles, and compare with our tool. Visualizations experts will evaluate all results for a more comprehensive evaluation.

7.5.2. Internal validity

This study meticulously designed its experiments to ensure result accuracy. However, human perception factors present challenges in evaluating a data visualization tool. These challenges can introduce biases, potentially skewing conclusions. Critical threats to this study's internal validity include:

- *Expectancy validity*. The expectancy effect is a significant threat to this study's internal validity. It is also known as the placebo effect. Participants may know they are using a tool for better data visualization. Because of this, they might view results as superior. This perception may not reflect the tool's actual quality or clarity. This bias can skew the results. Participants' expectations might influence their views, not just the tool's effectiveness. To prevent this threat, a blind testing approach is suggested. In this method, participants do not know the tool's intended benefits. Also, we ask users to compare the results obtained with heuristics and best practices. This comparison assesses whether the generated visualizations provide a good understanding of the data.
- *Learning validity*. The study faces a learning effect threat. Participants might use the data visualization tool repeatedly. Over time, they become more familiar with its features. This familiarity can boost performance. It is unclear if improvements stem from the tool or repeated use. Randomizing tasks or using a counterbalanced design is crucial to avoid this threat.

7.5.3. Construct validity

Implementation details can largely affect the tool's complexity, effort, and performance. In our case, the threats to the construct validity relate to the effort of implementing all modeled features with a specific visualization library and the performance of the SPL process.

- *Implementation effort.* A potential threat to construct validity in the implementation effort of our SPL lies in providing support for a more complex visualization libraries, such as *D3.js* or *matplotlib* (step 4). While our paper outlines a robust template-based approach for implementing variability in visualizations, there is a crucial oversight in discussing how the same principles may not seamlessly transfer to other libraries. The key challenge arises from the low-level nature of the libraries, which lacks readily available high-level constructs for capturing and replacing visualization types and their parameters within a template, as we have demonstrated in LaTeX using the *pgfplots* library. Unlike *pgfplots*, where these constructs are inherent, other libraries (e.g., *D3.js*) may demand a fundamentally different approach to addressing variability. To enhance the construct validity of our approach, we provide the implementation of the visualizations in our template-based approach using two different visualization libraries: *pgfplots* (in Latex) and *matplotlib* (in Python). This provides a comprehensive understanding of the adaptability and limitations of our approach in different library environments. Although further research, discussion, and potential prototyping are warranted, particularly when considering more intricate libraries such as *D3.js*.
- *Performance.* Another potential threat to construct validity in our study pertains to the efficiency of generating visualizations using a specific library within our SPL. The execution time of generating a visualization using the SPL has not been measured but is negligible for a single visualization configuration. However, due to the large number of possible configurations, generating all of them for comparison and deciding the most effective configuration can be challenging. Generating every possible configuration might seem comprehensive for a complete evaluation, but it raises concerns about practicality and relevance. In real-world scenarios, users typically seek a single, optimized configuration that effectively conveys their intended message. Generating thousands of configurations could result in a time-consuming and resource-intensive

process that may not align with the user's needs. To address this construct validity challenge, it is crucial to acknowledge the discrepancy between exhaustive configuration generation and the user's practical requirements. Our study focuses on identifying and providing the most effective configuration by encoding only the best practices and leaving out ineffective visualizations. This study emphasizes the importance of efficiency over sheer quantity in the context of SPL for visualization generation.

7.5.4. Conclusion validity

Conclusion validity relates to the reliability and robustness of our results, and concretely with the accuracy of the generated visualizations.

- *Accuracy of the visualizations.* A potential threat to conclusion validity in our study is the difference between the generated visualization with our tool and the final visualization present in the communication media (e.g., this article, a website). Automatically generated visualizations usually need manual revisions, especially when adjusting parameters or visual details that were not accounted for in the SPL nor encoded in the feature models (e.g., space between bars in a bar chart). While our SPL aims to provide an automated and systematic approach to visualization generation, the reality often involves scenarios where the user's specific requirements or unforeseen nuances demand specific manual interventions to obtain the final visualization (e.g., adjusting the visualization to the available space in a scientific article or webpage). This discrepancy can impact the construct validity of our tool, as it highlights a gap between the intended automatically generated visualization with our tool and the final visualization shown to final users. To mitigate this threat, we acknowledge the limitations of our SPL in capturing all possible parameters and visual intricacies, and emphasize the importance of user flexibility and the ability to manually fine-tune visualizations to align with their precise communication goals.

8. Related work

The visualization and SPL research have crossed paths on several occasions, but from a different perspective than the one presented in this article. Studies focus on applying visualization techniques in the context of SPLs (Nestor et al., 2008) mainly for the visualization of feature models (Nestor et al., 2007), but also to visualize variability and configurations (Asadi et al., 2016; Pleuss and Botterweck, 2012), feature interactions (Illescas et al., 2016), and constraints (Martinez et al., 2014). Techniques include the use of *colors* to visualize variabilities in source code (Kästner et al., 2008; Feigenspan et al., 2013), polymetric views (Lanza and Ducasse, 2003) in the form of a *variability blueprint* for the decomposition of complex feature models (Urli et al., 2015), *feature relations graphs* (Frogs) for feature constraints (Martinez et al., 2014), *cone trees* to draw feature models in 3D (Trinidad et al., 2008), or the use of statistics to visualize large-scale feature models (Heradio et al., 2019), among others. From this perspective, Lopez-Herrejon et al. (2018) present a systematic mapping study of visualization techniques that have been used for different SPL activities. In contrast, to the best of our knowledge, the variability of the visualization domain has not been studied yet (Galster et al., 2014).

Through a comprehensive literature review, we have identified fundamental principles for accuracy in data visualization. The Uncertainty Principle, emphasized by Cairo (2012) and by Wittenbrink and Lodha (Few, 2012), arises when measurement errors or missing data exist. The authors argue that using plots that encode data using only position and extent helps to represent them in a clear and orderly manner. The Implicit Error concept of Borkin et al. (2013) emphasizes the qualitative consideration of inherent measurement errors not recorded by subject matter experts, especially relevant in public health analysis, as demonstrated by research on viruses and pandemics. Friendly et al. (2001) further distinguish between correlation and causation, emphasizing the need for clarity in data representations to avoid misinterpretation. They stress the importance of recognizing third-party variables in data sets to avoid analytical errors. These findings refine the understanding of information visualization principles.

Several formal processes for visualization design have been proposed Walny et al. (2020), Chi (2002), Munzner (2009), Brehmer et al. (2014), such as the one presented in Section 2 by Walny et al. (2020). Chi (2002) also describes a framework to make information visualization systems easier to develop through the creation of a reference *Data State Model*. Munzner (2009) provided a four-level nested model for visualization design and validation that was further extended in McKenna et al. (2014) and Meyer et al. (2015). Brehmer et al. (2014) discussed pre-design empirical methods for information visualization using four illustrative scenarios, highlighting the methods and challenges unique to the visualization domain. All of these works expose the need to design the visualization at different levels as we have done in our approach in Fig. 4 (Section 4), but none of them takes into account the high variability that exists in the visualization design process. To synthesize variability and best design practices, numerous theoretical books have been published (Few, 2012; Diehl, 2007; Cairo, 2012; Murray, 2017; Evergreen, 2019; Knaflic, 2015; Gordon, 2004; Telea, 2014; Ward et al., 2010), of which we have mainly relied on Few (2012) because it is the most practical, presenting the variations, principles and best practices of visual perception, as well as its application to graphical communication.

Finally, regarding the evaluation of a visualization, there is an ongoing discussion in the visualization community (Kurzhals and Weiskopf, 2018) about the relevant factors that make a visualization effective, expressive, memorable, aesthetically pleasing, etc. These factors are typically led by guidelines and model theories (Lau and Vande Moere, 2007) of how different data representations and interaction concepts are perceived and processed by human users, requiring qualitative evaluation methods based on observation and interviews. For instance, in Kurzhals and Weiskopf (2018), Kurzhals and Weiskopf adopt the *repertory grid* methodology in the context of visualization. The repertory grid is a technique based on the personal construct theory for identifying the ways that a person interprets or gives meaning to, in this case, a visualization. The evaluation of design criteria for visualization can be performed using automatic algorithms (e.g., computation of *visual clutter* (Rosenholtz et al., 2007) and *scagnostics* (Wilkinson et al., 2005)), or with structured approaches, such as *questionnaires* (Forsell and Cooper, 2012), that involve the human user for a quantitative and qualitative assessment of the visualization. Isenberg et al. (2013) reviewed evaluation methods applied in visualization research, based on a coding scheme and seven scenarios by Lam et al. (2012). Our SPL can help with the quantitative assessment of the visualization by identifying those features that should be present in a configuration to fullfill the quality factors of a visualization (e.g., to be effective).

To our knowledge, there is no proposal for generating visualizations based on best practices and heuristics supported in an SPL. This fact makes our proposal original and initiates a new field of study using visualizations and SPLs.

In response to research question 1 (see Section 1) on SPL use of visualizations, we conclude that, while the process of designing such an SPL is several orders of magnitude larger than designing the visualizations we want, in extended use over time, the SPL is reusable in a multitude of contexts, given that the artifacts, i.e., templates, are reusable and can be evolved and corrected. With the SPL, we can generate several versions of the same type of visualization, e.g., a bar chart, without having to make the effort of designing each visualization by hand.

9. Conclusions and future work

We have presented an SPL approach to deal with the high variability that exists in the visualization design process. Our solution helps practitioners communicate their quantitative data effectively by assisting them in the selection and generation of visualizations that best represent the desired information. The proposed SPL allows for the configuration of different displays, visual components, and visual attributes taking into account the principles and best design practices of visualization experts.

The complexity of implementing an SPL, as opposed to implementing visualizations by hand, is offset by the reduction of error and malpractice associated with the incorrect use of libraries and library configurable elements. Although the process of implementing an SPL is always costly, modeling the variability of the significant chart and table visualization libraries more than compensates for having to know all the best practices in advance, which is only sometimes the case. Our proposal acts as a guide through the entire configurable spectrum to provide compelling visualizations that help to understand the data and draw accurate conclusions, avoiding statistical, knowledge or other biases.

In response to research question 2 (see Section 1) on the generalist use of visualization software tools, we appreciate that the use of tools which generate visualizations based on good practices and principles helps to generate effective and easy-to-communicate visualizations. Using these tools saves us the process of knowing all the literature in advance or generating erroneous visualizations without knowing it.

We propose a more in-depth evaluation. In future visualization master's courses taught by some authors, we will do hands-on evaluations with students. We will ask them to create graphs and data tables at three points: before learning the principles, after, and using our methodology. External evaluators who are visualization experts will give feedback on the changes in the three phases.

With this contribution, we open a new window of research where data visualization and variability meet each other, and where several challenges remain open. It is easy to imagine that more variability can be found in other parts of the data visualization process, and therefore finding different options to handle this variability is challenging. Data visualization is an area of research that is increasingly active with the current *data era* we are facing. Data that are related to computers but also data that are all over different areas ranging from biology to archaeology. We envision that variability management knowledge can be used in this trending area to facilitate the data visualization process, as we have partially shown in this contribution. In this regard, as part of our ongoing work, we plan to extend the modeling of other principles and best practices of the visualization knowledge that cannot be directly encoded in feature models, such as, for example, the fact that horizontal bars work better than vertical bars when the categorical labels are too long, as shown in Section 6, or concepts such as the data-ink ratio presented in Section 2. We also plan to include an explicit process for the quantitative assessment of visualizations in our SPL, considering the factors (e.g., effectiveness) that are typically led by guidelines.

Material

All the source code and data can be downloaded and executed from the following repository: https://github.com/diverso-lab/spl_visualization_design.

CRediT authorship contribution statement

David Romero-Organvidez: Writing – original draft, Writing – review & editing, Validation. **Jose-Miguel Horcas:** Conceptualization, Investigation, Methodology, Software, Writing – review & editing. **José A. Galindo:** Investigation, Supervision, Writing – original draft, Writing – review & editing. **David Benavides:** Investigation, Methodology, Supervision, Validation, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: David Romero-Organvidez, Jose-Miguel Horcas, Jose A. Galindo, David Benavides reports financial support was provided by Ministry of Science and Innovation. David Romero-Organvidez, Jose-Miguel Horcas, Jose A. Galindo, David Benavides reports financial support was provided by Government of Andalusia. David Romero-Organvidez reports a relationship with University of Seville that includes: employment and funding grants.

Data availability

Feature models of the SPL configuration steps are available in UVLHub: http://uvlhub.io/doi/10.5072/zenodo.39565. UVLHub is a repository specialized in feature models in UVL Benavides et al. (2024), Sundermann et al. (2023) language and following Open Science principles. Romero-Organvidez et al. (2021, 2023).

Acknowledgments

This work was partially supported by FEDER/Ministry of Science, Innovation and Universities/Junta de Andalucía/State Research Agency/CDTI with the following grants: *Data-pl* (PID2022-138486OB-I00), TASOVA PLUS research network (RED2022-134337-T) and MIDAS (IDI-20230256). Also, the work from the University of Málaga is supported by the projects *IRIS* PID2021-122812OB-I00 (co-financed by FEDER funds), *LEIA* UMA18-FEDERJA-157, and *DAEMON* H2020-101017109. David Romero-Organvidez is supported by PREP2022-000335, financed by MICIN/AEI/10.13039/501100011033 and by FSE+

Thanks to Mario Pérez Montoro (Universitat de Barcelona) and Jorge García Gutiérrez (Universidad de Sevilla) for their contribution to the evaluation of this article.

References

Aghababaeyan, Zohreh, Abdellatif, Manel, Briand, Lionel C., Ramesh, S, Bagherzadeh, Mojtaba, 2021. Black-box testing of deep neural networks through test case diversity. CoRR abs/2112.12591.arXiv:2112.12591.

Alférez, Mauricio, Acher, Mathieu, Galindo, José Angel, Baudry, Benoit, Benavides, David, 2019. Modeling variability in the video domain: Language and experience report. Softw. Qual. J. 27 (1), 307–347. http://dx.doi.org/10.1007/s11219-017-9400-8.

Apel, Sven, Batory, Don S., Kästner, Christian, Saake, Gunter, 2013. Feature-Oriented Software Product Lines - Concepts and Implementation. Springer, http://dx.doi.org/10.1007/978-3-642-37521-7.

Asadi, Mohsen, Soltani, Samaneh, Gasevic, Dragan, Hatala, Marek, 2016. The effects of visualization and interaction techniques on feature model configuration. Emp. Soft. Eng. 21 (4), 1706–1743. http://dx.doi.org/10.1007/s10664-014-9353-5.

Bai, Xiaoyan, White, David, Sundaram, David, 2010. A flexible approach for visualization development. In: 2010 Sixth International Conference on Signal-Image Technology and Internet Based Systems. IEEE, pp. 315–322.

Benavides, David, Martín-Arroyo, Pablo Trinidad, Cortés, Antonio Ruiz, 2005. Automated reasoning on feature models. In: 17th International Conference on Advanced Information Systems Engineering. CAiSE, In: LNCS, vol. 3520, Springer, pp. 491–503. http://dx.doi.org/10.1007/11431855_34.

Benavides, D., Sundermann, C., Vill, S., Feichtinger, K., Galindo, José A., Rabiser, R., Thüm, T., 2024. UVL: Feature Modelling with the Universal Variability Language. J. Syst. Softw. submitted. http://dx.doi.org/10.2139/ssrn.4764657, https://ssrn.com/abstract=4764657.

Bertin, Jacques, 1983. Semiology of graphics. University of Wisconsin Press.

Bigelow, Alex, Drucker, Steven, Fisher, Danyel, Meyer, Miriah, 2014. Reflections on how designers design with data. In: Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces. pp. 17–24.

Borkin, Michelle A., Vo, Azalea A., Bylinskii, Zoya, Isola, Phillip, Sunkavalli, Shashank, Oliva, Aude, Pfister, Hanspeter, 2013. What makes a visualization memorable? IEEE Trans. Vis. Comput. Graphics 19 (12), 2306–2315. http://dx.doi.org/10.1109/TVCG.2013.234.

Bostock, Michael, Ogievetsky, Vadim, Heer, Jeffrey, 2011. D^3 data-driven documents. IEEE Trans. Vis. Comput. Graph. 17 (12), 2301–2309. http://dx.doi.org/10.1109/TVCG.2011.185.

Braga, Jaqueline, Silva, Tiago, Souto, Virginia, 2020. Manipulações estatísticas e anomalias visuais: design de visualização de dados e reconhecimento de vieses estatísticos | statistical manipulations and visual anomalies: data visualization design and statistical bias recognition. InfoDesign - Revista Brasileira de Design da Informação 17, 145–162. http://dx.doi.org/10.51358/id.v17i2.756.

Brehmer, Matthew, Carpendale, Sheelagh, Lee, Bongshin, Tory, Melanie, 2014. Pre-design empiricism for information visualization: Scenarios, methods, and challenges. In: 5th BELIV. pp. 147–151. http://dx.doi.org/10.1145/2669557.2669564.

Bresciani, Sabrina, Eppler, Martin, 2015. The pitfalls of visual representations: A review and classification of common errors made while designing and interpreting visualizations. SAGE Open 5, http://dx.doi.org/10.1177/2158244015611451.

Cairo, Alberto, 2012. The Functional Art: An introduction to information graphics and visualization. New Riders.

Chakraborty, Mainak, Kundan, Ajit Pratap, 2021. Grafana. In: Monitoring Cloud-Native Applications: Lead Agile Operations Confidently using Open Source Software. A Press, pp. 187–240. http://dx.doi.org/10.1007/978-1-4842-6888-9_6.

Chi, Ed Huai-hsin, 2002. A Framework for Visualizing Information, vol. 1, Springer Science & Business Media.

Cleveland, William S., McGill, Robert, 1985. Graphical perception and graphical methods for analyzing scientific data. Science 229 (4716), 828–833.

Czarnecki, Krzysztof, Helsen, Simon, Eisenecker, Ulrich W., 2005a. Formalizing cardinality-based feature models and their specialization. Softw. Process. Improv. Pract. 10 (1), 7–29. http://dx.doi.org/10.1002/spip.213.

Czarnecki, Krzysztof, Helsen, Simon, Eisenecker, Ulrich W., 2005b. Staged configuration through specialization and multilevel configuration of feature models. Sw. Proc. Imp. Prac. 10 (2), 143–169. http://dx.doi.org/10.1002/spip.225.

Diehl, Stephan, 2007. Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software. Springer Science & Business Media.

Dong, Ensheng, Du, Hongru, Gardner, Lauren, 2020. An interactive web-based dashboard to track COVID-19 in real time. Lancet Infect. Dis. 20 (5), 533–534. http://dx.doi.org/10.1016/S1473-3099(20)30120-1.

Evergreen, Stephanie D.H., 2019. Effective Data Visualization: The Right Chart for the Right Data. SAGE publications.

Feigenspan, Janet, Kästner, Christian, Apel, Sven, Liebig, Jörg, Schulze, Michael, Dachselt, Raimund, Papendieck, Maria, Leich, Thomas, Saake, Gunter, 2013. Do background colors improve program comprehension in the #ifdef hell? Empir. Softw. Eng. 18 (4), 699–745. http://dx.doi.org/10.1007/s10664-012-9208-x.

Few, Stephen, 2012. Show Me the Numbers: Designing Tables and Graphs to Enlighten, second ed. Analytics Press, Oakland, CA, USA.

Forsell, Camilla, Cooper, Matthew, 2012. Questionnaires for evaluation in information visualization. In: 4th Workshop: Beyond Time and Errors: Novel Evaluation Methods for Visualization. BELIV, http://dx.doi.org/10.1145/2442576.2442592.

Freixa Font, Pere, Pérez-Montoro Gutiérrez, Mario, Codina, Lluís, 2021. The binomial of interaction and visualization in digital news media: Consolidation, standardization and future challenges. Profesional de la Inf. 2021 30 (4), e300401.

Friendly, Michael, Denis, Daniel, Truman, Harry, 2001. Milestones in the history of thematic cartography, statistica l graphics, and data visualization.

Furche, Tim, Gottlob, Georg, Libkin, Leonid, Orsi, Giorgio, Paton, Norman W., 2016. Data wrangling for big data: Challenges and opportunities. In: EDBT, vol. 16, pp. 473–478.

Galindo, José A., Benavides, David, 2020. A Python framework for the automated analysis of feature models: A first step to integrate community efforts. In: 24th ACM International Systems and Software Product Line Conference, Vol. B. SPLC, ACM, Montreal, Canada, pp. 52–55. http://dx.doi.org/10.1145/3382026.3425773.

Galindo, José A., Horcas, José Miguel, Felfernig, Alexander, Fernández-Amorós, David, Benavides, David, 2023. FLAMA: a collaborative effort to build a new framework for the automated analysis of feature models. In: Proceedings of the 27th ACM International Systems and Software Product Line Conference - Volume B. SPLC 2023, Tokyo, Japan, 28 August 2023- 1 September 2023, ACM, pp. 16–19. http://dx.doi.org/10.1145/3579028.3609008.

Galster, Matthias, Weyns, Danny, Tofan, Dan, Michalik, Bartosz, Avgeriou, Paris, 2014. Variability in software systems—A systematic literature review. IEEE Trans. Softw. Eng. 40 (3), 282–306. http://dx.doi.org/10.1109/TSE.2013.56.

Gordon, Ian E., 2004. Theories of Visual Perception. Psychology Press.

Hehman, Eric, Xie, Sally Y., 2021. Doing better data visualization. Adv. Methods Pract. Psychol. Sci. 4 (4), 25152459211045334.

Heradio, Ruben, Fernández-Amorós, David, Mayr-Dorn, Christoph, Egyed, Alexander, 2019. Supporting the statistical analysis of variability models. In: 41st Int. Conf. on Soft. Eng.. ICSE, pp. 843–853. http://dx.doi.org/10.1109/ICSE.2019.00091.

Horcas, Jose-Miguel, Galindo, Jose A., Benavides, David, 2022a. Variability in data visualization: A software product line approach. In: Proceedings of the 26th ACM International Systems and Software Product Line Conference - Volume A. SPLC '22, Association for Computing Machinery, New York, NY, USA, pp. 55–66. http://dx.doi.org/10.1145/3546932.3546993.

Horcas, José Miguel, Pinto, Mónica, Fuentes, Lidia, 2022b. Empirical analysis of the tool support for software product lines. Softw. Syst. Model. http://dx.doi.org/10.1007/s10270-022-01011-2.

Hunter, J.D., 2007. Matplotlib: A 2D graphics environment. Comput. Sci. Eng. 9 (03), 90–95. http://dx.doi.org/10.1109/MCSE.2007.55.

Illescas, Sheny, Lopez-Herrejon, Roberto E., Egyed, Alexander, 2016. Towards visualization of feature interactions in software product lines. In: 4th IEEE Work. Conf. on Soft. Vis.. VISSOFT, pp. 46–50. http://dx.doi.org/10.1109/VISSOFT.2016.16.

Isenberg, Tobias, Isenberg, Petra, Chen, Jian, Sedlmair, Michael, Möller, Torsten, 2013. A systematic review on the practice of evaluating visualization. IEEE Trans. Vis. Comput. Graphics 19 (12), 2818–2827. http://dx.doi.org/10.1109/TVCG.2013.126.

Jacobson, Ivar, Griss, Martin L., Jonsson, Patrik, 1997. Software Reuse - Architecture, Process and Organization for Business. Addison-Wesley-Longman.

Joshi, Ankush, Tiwari, Haripriya, 2023. An overview of Python libraries for data science. J. Eng. Technol. Appl. Phys. 5, 85–90. http://dx.doi.org/10.33093/jetap.2023.5.2.10.

Kästner, Christian, Trujillo, Salvador, Apel, Sven, 2008. Visualizing software product line variabilities in source code. In: 12th International Conference on Software Product Lines, Vol. 2 (Workshops). SPLC, pp. 303–312.

Kirk, Andy, 2012. Data Visualization: A Successful Design Process. Packt publishing.

Knaflic, Cole Nussbaumer, 2015. Storytelling with Data: A Data Visualization Guide for Business Professionals. John Wiley & Sons.

Krüger, Jacob, Pinnecke, Marcus, Kenner, Andy, Kruczek, Christopher, Benduhn, Fabian, Leich, Thomas, Saake, Gunter, 2018. Composing annotations without regret? Practical experiences using FeatureC. Softw. Pract. Exp. 48 (3), 402–427. http://dx.doi.org/10.1002/spe.2525.

Kurzhals, Kuno, Weiskopf, Daniel, 2018. Exploring the visualization design space with repertory grids. Comput. Graph. Forum 37 (3), 133–144. http://dx.doi.org/10.1111/cgf.13407.

Lam, Heidi, Bertini, Enrico, Isenberg, Petra, Plaisant, Catherine, Carpendale, Sheelagh, 2012. Empirical studies in information visualization: Seven scenarios. IEEE Trans. Vis. Comput. Graphics 18 (9), 1520–1536. http://dx.doi.org/10.1109/TVCG.2011.279.

Lanza, M., Ducasse, S., 2003. Polymetric views - A lightweight visual approach to reverse engineering. IEEE Trans. Softw. Eng. 29 (9), 782–795. http://dx.doi.org/10.1109/TSE.2003.1232284.

Lau, Andrea, Vande Moere, Andrew, 2007. Towards a model of information aesthetics in information visualization. In: 11th International Conference Information Visualization. IV, pp. 87–92. http://dx.doi.org/10.1109/IV.2007.114.

Liang, Xin, Whitney, Ben, Chen, Jieyang, Wan, Lipeng, Liu, Qing, Tao, Dingwen, Kress, James, Pugmire, David, Wolf, Matthew, Podhorszki, Norbert, et al., 2021. Mgard+: Optimizing multilevel methods for error-bounded scientific data reduction. IEEE Trans. Comput. 71 (7), 1522–1536.

Lopez-Herrejon, Roberto Erick, Illescas, Sheny, Egyed, Alexander, 2018. A systematic mapping study of information visualization for software product line engineering. J. Softw.: Evol. Process 30 (2), http://dx.doi.org/10.1002/smr.1912.

M. Wong, Dona, 2010. The Wall Street Journal Guide to Information Graphics: The Dos and Don'ts of Presenting Data, Facts, and Figures. W. W. Norton & Company, USA.

Martinez, Jabier, Ziadi, Tewfik, Mazo, Raul, Bissyandé, Tegawendé F., Klein, Jacques, Traon, Yves Le, 2014. Feature relations graphs: A visualisation paradigm for feature constraints in software product lines. In: 2nd IEEE Working Conference on Software Visualization. VISSOFT, pp. 50–59. http://dx.doi.org/10.1109/VISSOFT.2014.18.

McKenna, Sean, Mazur, Dominika, Agutter, James, Meyer, Miriah, 2014. Design activity framework for visualization design. IEEE Trans. Vis. and Comp. Graph. 20 (12), 2191–2200. http://dx.doi.org/10.1109/TVCG.2014.2346331.

Meinicke, Jens, Thüm, Thomas, Schröter, Reimar, Benduhn, Fabian, Leich, Thomas, Saake, Gunter, 2017. Mastering Software Variability with FeatureIDE. Springer, http://dx.doi.org/10.1007/978-3-319-61443-4.

Meinicke, Jens, Wong, Chu-Pan, Vasilescu, Bogdan, Kästner, Christian, 2020. Exploring differences and commonalities between feature flags and configuration options. In: Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Practice. pp. 233–242.

Meyer, Miriah, Sedlmair, Michael, Quinan, P. Samuel, Munzner, Tamara, 2015. The nested blocks and guidelines model. Inf. Vis. 14 (3), 234–249. http://dx.doi.org/10.1177/1473871613510429.

Midway, Stephen R., 2020. Principles of effective data visualization. Patterns 1 (9), 100141. http://dx.doi.org/10.1016/j.patter.2020.100141.

Moody, Daniel, 2009. The "physics" of notations: Toward a scientific basis for constructing visual notations in software engineering. IEEE Trans. Softw. Eng. 35 (6), 756–779. http://dx.doi.org/10.1109/TSE.2009.67.

Munoz, Daniel-Jesus, Oh, Jeho, Pinto, Mónica, Fuentes, Lidia, Batory, Don S., 2019. Uniform random sampling product configurations of feature models that have numerical features. In: 23rd International Systems and Software Product Line Conference. SPLC, ACM, pp. 39:1–39:13. http://dx.doi.org/10.1145/3336294.3336297.

Munzner, Tamara, 2009. A nested model for visualization design and validation. IEEE Trans. Vis. Comput. Graphics 15 (6), 921–928. http://dx.doi.org/10.1109/TVCG.2009.111.

Murray, Scott, 2017. Interactive Data Visualization for the Web: An Introduction to Designing with D3. " O'Reilly Media, Inc.".

Nestor, Daren, O'Malley, Luke, Quigley, Aaron J., Sikora, Ernst, Thiel, Steffen, 2007. Visualisation of variability in software product line engineering. In: 1st Workshop on Variability Modelling of Software-Intensive Systems. VaMoS, pp. 71–78.

Nestor, Daren, Thiel, Steffen, Botterweck, Goetz, Cawley, Ciarán, Healy, Patrick, 2008. Applying visualisation techniques in software product lines. In: ACM Symp. on Software Visualization. pp. 175–184. http://dx.doi.org/10.1145/1409720.1409748.

Nguyen, Vinh, Jung, Kwanghee, Gupta, Vibhuti, 2021. Examining data visualization pitfalls in scientific publications. Vis. Comput. Ind., Biomed. Art 4, http://dx.doi.org/10.1186/s42492-021-00092-y.

Ouali, Sami, 2021. Generating software product line model by resolving code smells in the products' source code. Int. J. Softw. Eng. Appl. 12, 1–10. http://dx.doi.org/10.5121/ijsea.2021.12101.

Pleuss, Andreas, Botterweck, Goetz, 2012. Visualization of variability and configuration options. Int. J. Softw. Tools Technol. Transf. 14 (5), 497–510. http://dx.doi.org/10.1007/s10009-012-0252-z.

Pohl, Klaus, Böckle, Günter, van der Linden, Frank, 2005. Software Product Line Engineering - Foundations, Principles, and Techniques. Springer, http://dx.doi.org/10.1007/3-540-28901-1.

Rabiser, Daniela, Prähofer, Herbert, Grünbacher, Paul, Petruzelka, Michael, Eder, Klaus, Angerer, Florian, Kromoser, Mario, Grimmer, Andreas, 2018. Multi-purpose, multi-level feature modeling of large-scale industrial software systems. Soft. Sys. Model. 17 (3), 913–938. http://dx.doi.org/10.1007/s10270-016-0564-7.

Rabiser, Rick, Schmid, Klaus, Becker, Martin, Botterweck, Goetz, Galster, Matthias, Groher, Iris, Weyns, Danny, 2019. Industrial and academic software product line research at SPLC: Perceptions of the community. In: Proceedings of the 23rd International Systems and Software Product Line Conference - Volume A. SPLC '19, Association for Computing Machinery, New York, NY, USA, pp. 189–194. http://dx.doi.org/10.1145/3336294.3336310.

Reiser, Mark-Oliver, Weber, Matthias, 2007. Multi-level feature trees. Requir. Eng. 12 (2), 57–75. http://dx.doi.org/10.1007/s00766-007-0046-0.

Romero-Organvidez, David, Galindo, José Á., Horcas, Jose-Miguel, Benavides, David, 2021. A first prototype of a new repository for feature model exchange and knowledge sharing. In: Proceedings of the 25th ACM International Systems and Software Product Line Conference - Volume B. In: SPLC '21, Association for Computing Machinery, New York, NY, USA, pp. 80–85. http://dx.doi.org/10.1145/3461002.3473949.

Romero-Organvidez, D., Galindo, J., Sundermann, Chico, Horcas, J., Benavides, D., 2023. UVLHub: A Feature Model Data Repository Using UVL and Open Science Principles. J. Syst. Softw. submitted. http://dx.doi.org/10.2139/ssrn.4536607.

Rosenholtz, Ruth, Li, Yuanzhen, Nakano, Lisa, 2007. Measuring visual clutter. J. Vision 7 (2), 17. http://dx.doi.org/10.1167/7.2.17.

Schmitt, Anna, Bettinger, Christian, Rock, Georg, 2018. Glencoe – A tool for specification, visualization and formal analysis of product lines. In: 25th International Conference on Transdisciplinary Engineering. In: Advances in Transdisciplinary Engineering, pp. 665–673. http://dx.doi.org/10.3233/978-1-61499-898-3-665.

Schröter, Reimar, Krieter, Sebastian, Thüm, Thomas, Benduhn, Fabian, Saake, Gunter, 2016. Feature-model interfaces: The highway to compositional analyses of highly-configurable systems. In: Proceedings of the 38th International Conference on Software Engineering. ICSE '16, Association for Computing Machinery, New York, NY, USA, pp. 667–678. http://dx.doi.org/10.1145/2884781.2884823.

Segura, Sergio, Hierons, Robert M., Benavides, David, Cortés, Antonio Ruiz, 2011. Automated metamorphic testing on the analyses of feature models. Inf. Softw. Technol. 53 (3), 245–258. http://dx.doi.org/10.1016/J.INFSOF.2010.11.002.

Simmon, Robert, 2013. Subtleties of Color. USA.

Sun, Ying, Wu, Huang, Qiu, Yinghong, Yue, Zhiqiang, 2018. Stereoacuity of black-white and red-green patterns in individuals with and without color deficiency. J. Ophthalmol. 2018, 1–4. http://dx.doi.org/10.1155/2018/1926736.

Sundermann, C., Vill, S., Thüm, T., Feichtinger, K., Agarwal, P., Rabiser, R., Galindo, José A., Benavides, D., 2023. UVLParser: Extending UVL with Language Levels and Conversion Strategies. In: 27th ACM International Systems and Software Product Line Conference (SPLC 2023). ACM, p. to appear.

Syriani, Eugene, Luhunu, Lechanceux, Sahraoui, Houari A., 2018. Systematic mapping study of template-based code generation. Comput. Lang. Syst. Struct. 52, 43–62. http://dx.doi.org/10.1016/j.cl.2017.11.003.

Tantau, Till, 2013. Graph drawing in TikZ. J. Graph Algorithms Appl. 17 (4), 495–513. http://dx.doi.org/10.7155/jgaa.00301.

Telea, Alexandru C., 2014. Data Visualization: Principles and Practice. CRC Press.

The dblp team, 2022. Dblp computer science bibliography. Monthly snapshot release of March 2022, URL https://dblp.org/xml/release/dblp-2022-03-01.xml.gz.

Todorovic, Dejan, 2008. Gestalt principles. Scholarpedia 3 (12), 5345.

Trinidad, Pablo, Cortés, Antonio Ruiz, Benavides, David, Segura, Sergio, 2008. Three-dimensional feature diagrams visualization. In: 12th International Conference on Software Product Lines, Vol. 2 (Workshops). SPLC, pp. 295–302.

Tufte, Edward R., 1985. The visual display of quantitative information. J. Healthc. Qual. (JHQ) 7 (3), 15.

Urli, Simon, Bergel, Alexandre, Blay-Fornarino, Mireille, Collet, Philippe, Mosser, Sébastien, 2015. A visual support for decomposing complex feature models. In: 3rd IEEE VISSOFT. pp. 76–85. http://dx.doi.org/10.1109/VISSOFT.2015.7332417.

Vučetić, Slaviša, Vulović, Marko, Radosavljevic, Dragana, Milic, Petar, Lekic, Julijana, Milosavljević, Bojana, 2023. Open data visualization by using Javascript libraries.

Walny, Jagoda, Frisson, Christian, West, Mieka, Kosminsky, Doris, Knudsen, Søren, Carpendale, Sheelagh, Willett, Wesley, 2019. Data changes everything: Challenges and opportunities in data visualization design handoff. IEEE Trans. Vis. Comput. Graph. 26 (1), 12–22.

Walny, Jagoda, Frisson, Christian, West, Mieka, Kosminsky, Doris, Knudsen, Søren, Carpendale, Sheelagh, Willett, Wesley, 2020. Data changes everything: Challenges and opportunities in data visualization design handoff. IEEE Trans. Vis. Comp. Graph. 26 (1), 12–22. http://dx.doi.org/10.1109/TVCG.2019.2934538.

Ward, Matthew O., Grinstein, Georges, Keim, Daniel, 2010. Interactive Data Visualization: Foundations, Techniques, and Applications. AK Peters/CRC Press.

Ware, Colin, 2019. Information Visualization: Perception for Design. Morgan Kaufmann.

Wickham, Hadley, 2009. ggplot2 - elegant graphics for data analysis. In: Use R, Springer, http://dx.doi.org/10.1007/978-0-387-98141-3.

Wilkinson, Leland, Anand, Anushka, Grossman, Robert L., 2005. Graph-theoretic scagnostics. In: IEEE Symposium on Information Visualization. InfoVis, pp. 157–164. http://dx.doi.org/10.1109/INFVIS.2005.1532142.

Wohlin, Claes, Runeson, Per, Höst, Martin, Ohlsson, Magnus C., Regnell, Björn, 2012. Experimentation in Software Engineering. Springer, http://dx.doi.org/10.1007/978-3-642-29044-2.

Yost, Beth, North, Chris, 2006. The perceptual scalability of visualization. IEEE Trans. Vis. Comput. Graphics 12 (5), 837–844.

David Romero Organvídez is a graduate of the BSc in Software Engineering in 2021, MSc in Software Engineering in 2022 and currently a PhD student in Computer Science at the University of Seville. He has a pre-doctoral contract in the Data-pl project led by Professors David Benavides and José A. Galindo. His areas of interest are variability models, software product lines and model repositories based on Open Science principles. He is the technology leader of the UVLHub project (https://www.uvlhub.io). The main line of his thesis is directed to the intensive use of data in software product lines. He combines his thesis with a Teaching Assistant position at the University of Seville.

José Miguel Horcas is an Associate Professor at the University of Málaga, Spain, where he received his M.Sc. degree in 2012 and his Ph.D. in Computer Sciences in 2018. He is a member of the CAOSD research group of the University of Málaga. He carried out two postdoc stays at King's College London in 2019 for three months, and at the University of Seville in 2021-2022 for 18 months. His main research areas are related to software product lines, including variability and configurability, as well to quality attributes and modularity. More information available at https://sites.google.com/view/josemiguelhorcas.

José Angel Galindo is a Tenured Associate Professor at the University of Seville, he has conducted research and taught in the United States, France, and Spain. Specializing in software product lines and product configuration, he initiated his research by developing variability management tools such as FaMa and BeTTy, which have gained widespread adoption in academia and industry. His thesis, focusing on the application of these techniques and tools to diverse domains beyond traditional product lines, earned him the prestigious SISTEDES Best Thesis Award at the national level. Nowadays, he continues to pioneer new variability analysis techniques and tools such as the Flama Project () as project leader, which aggregates the state of the art on variability analysis.

David Benavides received the B.S. degree in information systems from the Institute Superieur d'Electronique de Paris, France, in 2000, the M.Sc. degree in computer engineering from the University of Seville, Spain, in 2001, and the Ph.D. degree in software engineering from the University of Seville, in 2007, where he is Full Professor of Software Engineering and leads the Diverso Lab (https://grupo.us.es/diversolab). He is in the direction board of UVL (https://universal-variability-language.github.io), flama (https://flamapy.github.io) and UVLHub (https://www.uvlhub.io). His main research interests include software product lines, feature modelling, variability-intensive systems, computational thinking and libre and open-source software development.