ABSTRACT
Software product-line engineering is concerned with developing a set of similar, yet customized, software systems that share a common codebase. To develop such a variant-rich system, various development processes, techniques, and tools have been studied in research and are used in practice. Specifically, to help developers manage the complexity of developing large-scale variant-rich systems, researchers have proposed visualizations to visually present different properties of such systems and their engineering—such as feature models, configurations, the similarity of variants, or process traces. Two recent mapping studies have systematically elicited the state-of-the-art on such visualizations, but neither of them provides a comparative analysis of the underlying visualization concepts and tools. In this paper, we report a qualitative meta-analysis of the 64 papers that we primarily selected from these two mapping studies. Advancing on the previous studies, we compare the use cases, pros, cons, and relations between visualization concepts and tools used with respect to engineering variant-rich systems. Our results provide insights—orthogonal to those from the mapping studies—regarding the purposes for which visualization concepts are used and the tools that are available to implement these concepts. The overview we provide can help researchers as well as practitioners decide to use specific established visualization concepts or design new ones, and identify tools that can help them to implement these.

CCS CONCEPTS
• Software and its engineering → Software product lines;
• Human-centered computing → Visualization techniques;
Visualization systems and tools.

KEYWORDS
software product line, variant-rich system, visualization techniques, visualization systems and tools

ACM Reference Format:

1 INTRODUCTION
Software product-line engineering provides means for engineering a set of similar systems that build on a common codebase [4, 53, 82], thereby establishing a variant-rich system (a.k.a. product family or product line). Since most of today’s software systems have to exist in various variants to account for individual customer requirements, variant-rich systems are widely established in different domains, such as automotive, aerospace, and embedded firmware [29, 51, 52, 55–58, 79, 91, 100, 103]. Typically, developers initiate a variant-rich system via the clone-and-own method of copying and adapting a system to new requirements (e.g., via forking on social-coding platforms) and move to a fully integrated platform later on [21, 28, 49, 55, 86, 91]. An integrated platform requires more planning and investment in advance [1, 13, 17, 50, 51, 89], for instance, to set up the platform architecture, feature model [44, 75], variability mechanism [14], pipelines for configuring and deriving variants [18, 97], as well as testing [24]. However, the benefits (e.g., quality improvements, faster time-to-market) of a platform implemented based on product-line engineering methods usually outweigh the initial investments, which is why most organizations that develop a variant-rich system adopt a platform at some point [13, 51, 89, 100].

Regardless of how they were developed or are organized, real-world variant-rich systems (e.g., the Linux Kernel [94]) can become very large, for instance, in terms of their features, variants, source code, or the number of developers involved. Visualizations are key to helping developers understand and manage the size, complexity, as well as relations of such entities (e.g., feature dependencies, evolution graphs, fork networks). A visualization can provide a clear and intuitive representation of a certain part of a variant-rich system and its properties. Two independent mapping studies [61, 68] have been conducted recently to provide overviews of existing visualizations used in product-line engineering. Both focus on the entities that are visualized and the types of diagrams used. However, neither one reports a comparative analysis of the use cases for which the visualization concepts and respective tools are used. For instance, Medeiros et al. [68] provide an overview of what diagrams are used, what evolution scenarios [93] are covered, what benefits are reported, or what users can interact how with a visualization. In contrast, we are interested in what visualization concepts (e.g., diagrams) have been used for what reasons to visualize a certain entity of engineering variant-rich systems (e.g., features, forks) and the respective tools used. More specifically, we are not interested
There are two recent mapping studies on the various visualization variants-rich systems.

In this paper, we present a conceptual comparison of the different visualization concepts proposed and used in the context of engineering variant-rich systems. For this purpose, we performed a meta-analysis of 64 papers from which we elicited the visualization concepts, the visualized entities, the researchers’ underlying ideas, the evaluations performed, and the tools used. We discuss pros and cons of the underlying visualization concepts, relations, and tools, providing examples of their application in research and practice.

More precisely, we contribute the following in this paper:

- We provide an overview of what visualization concepts have been used for what entities in the context of engineering variant-rich systems.
- We analyze what use cases researchers have aimed to support using specific visualization concepts, as well as the tools they used for this purpose.
- We discuss what visualization concepts have been used, and what evidence for their usefulness has been reported.
- We share our analysis dataset with more detailed descriptions in a persistent open-access repository.¹

Building on our contributions, researchers as well as practitioners can more easily identify appropriate concepts and tools to support their own use cases or to develop novel visualization ideas.

## 2 RELATED WORK

There are two recent mapping studies on the various visualization concepts that have been proposed for engineering variant-rich systems, which are the work most closely related to our own analysis.

Lopez-Herrejon et al. [61] have conducted a systematic mapping study on visualizations for software product-line engineering, summarizing findings for 37 primary sources. They performed their meta-analysis by gathering information on the visualization techniques and providing a categorized overview of the findings. Specifically, Lopez-Herrejon et al. have identified ten (types of) visualization tools (cf. Table 3), which they briefly introduce. However, the review does not explain and analyze what and why visualizations have only been visualized as typical trees, but also using concepts like tables, bubble maps, or scatter charts. We are concerned with exploring such relations between visualization concepts and entities, and with understanding what evidence exists that the visualizations are helpful for developers.

To tackle this goal, we defined three research questions (RQs):

**RQ1** What visualization concepts have been used for what entities? Some visualization concepts are widely established for specific entities, for instance trees (concept) to display feature models (entity) as diagrams. However, variant-rich systems exhibit various entities with relationships, which is why various visualization concepts have been proposed and combined. We elicited such combinations to provide an overview of the paths that have been explored.

**RQ2** What are the ideas of using these visualizations? Each visualization builds on an underlying idea for its specific use case. For instance, trees for feature models are the de facto standard idea [75], but some researchers proposed adding cone trees to provide a better visualization of how multiple variant-rich systems are connected. Exploring the underlying ideas of each visualization contributes to an understanding of what entities and relations the respective researchers aimed to visualize.

**RQ3** What is the evidence on the usefulness of these visualizations? When building on existing or proposing new visualizations, it is key to understand whether they actually help developers. For instance, while 3D cones have been used for visualizing feature models, their actual usefulness for developers has not been empirically evaluated (no insights). We collected an overview of which visualization concepts and underlying ideas have proven to benefit developers.

**RQ4** What tools have been used for implementing visualizations? To develop a visualization, it is helpful to build on established visualization tools. As an example, FeatureIDE [69] is a widely established tool for visualizing feature models, but extending it with new visualizations is quite cumbersome. To support researchers and practitioners in designing new visualizations, we provide an overview of the tools used in previous work that they can build upon.

¹https://doi.org/10.5281/zenodo.8069277
These research questions were outside the scope of the two mapping studies we build upon. Consequently, we argue that we contribute orthogonal and valuable in-depth insights into visualizations for engineering variant-rich systems.

3.2 Selecting Primary Sources

For our meta-study, we relied on the papers identified in the two systematic mapping studies by Lopez-Herrejon et al. [61] and Medeiros et al. [68]. Both works conducted systematic searches for relevant papers: Lopez-Herrejon et al. conducted an automated search on ScienceDirect, IEEEExplore, the ACM Digital Library, and SpringerLink, followed by a snowballing search. This resulted in 37 papers published between 2007 and 2016. Medeiros et al. performed an automated search on ScienceDirect, IEEEExplore, the ACM Digital Library, Scopus, and the Wiley Online Library, also followed by a snowballing search. They identified 41 papers that have been published from 2006 until 2021. Overall, this constituted a total of 58 papers after removing the duplicates between the two studies.

As a cross check, particularly for more recent papers, we performed a manual literature search through the 2016 to 2022 proceedings of the International Conference on Visualisation (VIS), International Working Conference on Software Visualisation (VISSOFT), and the International Systems and Software Product Lines Conference (SPLC). This process led to six new papers [8, 37, 38, 63, 72, 73] that we added to the previous ones, adding up to a total of 64 papers that we considered for our analysis. During our manual search, we selected any paper as relevant if it (i) has been published at the research track and (ii) reports on a visualization for engineering variant-rich systems. For this purpose, we first checked whether we could decide based on the title, then we checked the abstract, and if needed we read the full text of a paper. Since we investigated three prominent conferences, all papers fulfilled the typical criteria of being peer-reviewed and written in English.

3.3 Data Extraction

To answer our research questions, we extracted the following data (besides typical bibliographic data):

**RQ1** A list of keywords summarizing the entities that are visualized (e.g., features, feature model, variants).

**RQ1** A list of keywords describing the concepts used for visualizing (e.g., bar chart, tree, histograms).

**RQ1** Short descriptions of each relation between entities and concepts (i.e., short notes connecting one or more entities with one or more concepts).

**RQ2** The idea or use case for which each relation has been proposed for a visualization (i.e., a short note for each relation, such as “de facto standard of using trees for feature models”).

**RQ3** Information on the evaluations that have been performed (e.g., experiment with 10 students solving tasks).

**RQ3** Any results of the evaluation reported in the paper (e.g., feedback, task-solving performance).

**RQ4** A list of all tools used for implementing a visualization together with short notes on the pros and cons of these tools reported in a paper.

Using this data, we were able to answer our research questions and provide novel insights compared to the related work.

### Table 1: Visualization concepts used in the primary sources.

<table>
<thead>
<tr>
<th>Concept</th>
<th>Sources</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heatmap</td>
<td>[8, 26, 60, 78]</td>
<td>4</td>
</tr>
<tr>
<td>Histogram</td>
<td>[92]</td>
<td>1</td>
</tr>
<tr>
<td>Levelized structure map</td>
<td>[46]</td>
<td>1</td>
</tr>
<tr>
<td>Line chart</td>
<td>[25]</td>
<td>1</td>
</tr>
<tr>
<td>Logic gate</td>
<td>[31]</td>
<td>1</td>
</tr>
<tr>
<td>Sankey diagram</td>
<td>[70, 71, 81]</td>
<td>3</td>
</tr>
<tr>
<td>Scatter chart</td>
<td>[12]</td>
<td>1</td>
</tr>
<tr>
<td>Survival chart</td>
<td>[101]</td>
<td>1</td>
</tr>
<tr>
<td>Table/Matrix</td>
<td>[2, 7, 15, 25, 30, 36, 47, 60, 74, 83, 84, 88]</td>
<td>12</td>
</tr>
<tr>
<td>Tree</td>
<td>[3, 6, 8, 10, 11, 27, 31, 37, 38, 41–43, 63, 76, 90, 92, 96]</td>
<td>17</td>
</tr>
<tr>
<td>Treemap</td>
<td>[40, 60, 70, 72, 73]</td>
<td>5</td>
</tr>
<tr>
<td>Word cloud</td>
<td>[19, 65]</td>
<td>2</td>
</tr>
</tbody>
</table>

3.4 Data Analysis

To ensure consistency, the first author extracted all data by reading each paper in detail. They employed an open-coding process, marking relevant information in the full texts and extracting them into a spreadsheet. The other authors of this paper verified the extracted data during discussions, data analyses, based on their knowledge of these papers, and by picking individual papers for detailed checks. We then analyzed our data qualitatively, using open-card-sorting to derive common themes and terminologies from it.

4 RESULTS

We identified 19 visualization concepts from the primary sources. In Table 1, we display the distribution of what visualization concept has been used in the primary sources; and in Figure 1, we display the entities of engineering a variant-rich system that have been visualized using these concepts (RQ4). To understand the selection of visualization concepts used in research, we explored the reasoning behind these concepts (RQ2). Next, we provide a brief overview of the concepts, their underlying ideas, the existing evidence (RQ3), and the tools used (RQ4)—with more detailed descriptions in our dataset, answering our research questions in combination to avoid redundancy and improve the comprehensibility of this section.

**Answering RQ1: Visualization Concepts Used.** We identified 19 visualization concepts that have been used for eight different entities. While trees (17) and graphs (20) are the most commonly used visualization concepts, due to the hierarchical organization of variant-rich systems, there is a variety of other concepts that have been proposed to more effectively communicate data and insights. For hierarchical data like feature models and software artifacts, trees and treemaps are commonly used. Chronological data, such as changes in the lines of code of a feature over time, are typically visualized using tables, matrices, or survival charts. Categorized data, such as features, have been visualized using various concepts. In contrast,
Some visualization concepts used for engineering variant-rich systems, such as trees, graphs, tables, or matrices, are generally used for visualizing feature models and their derivatives (features and feature dependencies). In contrast, other visualization concepts have an underlying idea specific to their use case, such as colored code, word clouds, and concept lattices. The remaining visualization concepts have been explored for visualizing various entities of a variant-rich system, such as the overall structure, code complexity, or importance of features typically associated with each visualization concept’s ability to provide a unique perspective on the data. By using or combining different visualization concepts, entities within a variant-rich system can be associated with each other, which provides an understanding of the relationships or patterns in the system.

**Answering RQ1: Ideas of Visualizations.** Various evaluation methods have been used in the primary sources, showing evidence on the benefits of the visualization concepts covered. In Table 2, we display the distribution of the four evaluation methods we identified from the primary sources. While field experiments are a common method used in visualization research, other methods, such as judgment studies, experimental simulations, and field studies, have also been employed to evaluate visualizations for variant-rich systems. From such evaluations, we found that a substantial body of evidence supports the notion that trees are an effective means for visualizing feature models, while graphs have been shown to be effective in visualizing features and software artifacts. Additionally, tables and matrices have been found to be effective in visualizing features. Other visualization concepts have, to some extent, been demonstrated to be effective in visualizing some use cases. However, the level of evidence supporting their effectiveness varies, and further research is needed to fully investigate their potential in different contexts. Finally, the 24 primary sources that we do not reference do not include any type of evaluation of their proposed visualizations, indicating a gap in the research and a need to further investigate the effectiveness of these visualizations.

**Answering RQ2: Evidence on Visualizations.** The tools that have previously been identified by Lopez-Herrejon et al. [61] are asterisked (*).

<table>
<thead>
<tr>
<th>Tool</th>
<th>Sources</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>* Adhoc</td>
<td>[7, 8, 12, 15, 20–22, 26, 27, 31, 39, 46, 62, 64, 87, 90, 92, 98, 101, 104]</td>
<td>20</td>
</tr>
<tr>
<td>* Eclipse EMF-GEF</td>
<td>[5, 10, 11, 15, 30, 34, 35, 45, 66, 76, 80, 83, 84, 88]</td>
<td>15</td>
</tr>
<tr>
<td>* Prefuse</td>
<td>[6, 41, 81]</td>
<td>3</td>
</tr>
<tr>
<td>* D3.js</td>
<td>[19, 40, 47, 60, 95]</td>
<td>5</td>
</tr>
<tr>
<td>* Graphviz</td>
<td>[42]</td>
<td>1</td>
</tr>
<tr>
<td>* CCVisu</td>
<td>[5]</td>
<td>1</td>
</tr>
<tr>
<td>* Google Charts</td>
<td>[74]</td>
<td>1</td>
</tr>
<tr>
<td>* ConExp</td>
<td>[59]</td>
<td>1</td>
</tr>
<tr>
<td>* Moose</td>
<td>[99]</td>
<td>1</td>
</tr>
<tr>
<td>* Processing</td>
<td>[67]</td>
<td>1</td>
</tr>
<tr>
<td>FeatureIDE</td>
<td>[2, 70]</td>
<td>2</td>
</tr>
<tr>
<td>Babylon.js</td>
<td>[72, 73]</td>
<td>2</td>
</tr>
</tbody>
</table>

Figure 1: Visualization concepts (left) and the entities they visualize (right). The numbers indicate how often we identified each concept or entity. Note that these numbers do not precisely match those in Table 1, because some concepts/entities are used multiple times in one paper, for different purposes.

**Table 2: Evaluation methods used in the primary sources.**

<table>
<thead>
<tr>
<th>Evaluation</th>
<th>Sources</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field experiment</td>
<td>[5–8, 10, 11, 20, 22, 26, 31, 32, 36, 38–40, 42, 43, 47, 54, 59, 63, 65–67, 72, 73, 76, 78, 80, 81, 83, 88, 98, 99]</td>
<td>34</td>
</tr>
<tr>
<td>Judgement study</td>
<td>[2, 71, 74]</td>
<td>3</td>
</tr>
<tr>
<td>Experimental simulation</td>
<td>[41, 95]</td>
<td>2</td>
</tr>
<tr>
<td>Field study</td>
<td>[101]</td>
<td>1</td>
</tr>
</tbody>
</table>
Not supported. The visualization tool is not designed to implement the visualization concept.

We gathered this overview by inspecting and testing each tool, as well as reading its documentation. To understand why researchers opted for a particular visualization tool over others, we analyzed the advantages and limitations of these tools in the context of engineering variant-rich systems, which we detail in the following. Note that “visualized entities” refers to the entities our primary sources have visualized with a tool, not the tools’ general capabilities.

We found that some of the visualization tools are only designed for specific purposes: Graphviz and CCVisu are used to abstract graphs from structural information, while ConExp is specialized for formal concept analysis. Meanwhile, Prefuse and D3.js provide support for most visualization concepts, but have not been adopted to their full extent in the primary sources. We also note that although most of the primary sources do not explicitly point out their motivation for choosing a visualization tool, researchers tend to select the tool developed or supported by their institution (e.g., CCVisu [5] and Moose [99]). Thus, one factor that affects the choice of visualization tools could be familiarity with tools and languages. In our dataset, we further provide an overview of the key specifications of the tools to help researchers and practitioners select the most suitable one for their context.

5 DISCUSSION

Visualizations play a crucial role in understanding variant-rich systems and communicating their complexities effectively. For RQ1, we explored what and how visualization concepts have been used for different entities and found that, while some visualization concepts (e.g., trees, graphs) are commonly used, there are also entities for which other visualization concepts have been explored. We conclude that the choice of visualization concepts should be tailored to the specific data and perspectives being communicated.

For RQ2, we analyzed the reasons for using a visualization concept for an entity. Some visualization concepts are generally used for visualizing specific entities in variant-rich systems, such as trees and graphs for feature models and their derivatives due to the hierarchical structure. Other concepts have an underlying idea specific to their use case, such as colored code, word clouds, and concept lattices. Our findings highlight the importance of selecting a suitable visualization concept or combination of techniques to communicate effectively, as well as of reflecting on standard visualizations and potential improvements.

For RQ3, we examined the evaluation of each primary source. We found that there is reliable empirical evidence supporting the effectiveness of several visualization techniques used for variant-rich systems, such as trees, graphs, and tables/matrices for feature models and their derivatives. However, the level of evidence supporting the effectiveness of other visualization concepts varies, and most primary sources do not include an evaluation study. This clearly indicates the need for further research and systematic evaluations.

For RQ4, we explored what tools the visualizations were implemented with. Some tools like Eclipse-based platforms are compatible with the development environment, enabling more advanced operations, such as code interactions and configuring. Researchers should carefully consider the available visualization tools and their specifications when selecting the most suitable tool for their requirements and ideas. We contribute an overview of the key specifications of each visualization tool, which can help researchers in selecting the most suitable tool for their requirements.

Our findings further reveal the heterogeneity of visualization tools regarding different use cases. Tools, according to their capabilities and limitations, handle tasks with different visualization techniques and integrate with workflows at different levels. In general, the selection of visualization tools is determined by the task domain, data type, and skills or experience of the user. Considering that skills and experience vary among users and we cannot take that into account, we assume that familiarity with the tools and programming languages does not impact a choice. Under that assumption, the following high-level considerations can help select proper tools for a visualization:

- If the visualization must be embedded into a development workflow, such as interacting with the source code or integrating click-and-configure experience, Eclipse EMF-GEF, FeatureIDE, Prefuse, and Adhoc tools seem most useful. Eclipse EMF-GEF and FeatureIDE work well when the variant-rich system is modeled and built with Eclipse; Prefuse can be integrated with Java projects; and Adhoc tools may be a better option to fit specific development environments.
- If a formal concept analysis is needed, ConExp is a proper choice specifically designed for this use case.
- If software analyses on a complex variant-rich system are needed, Moose can be an option for its capability to handle large-scale systems.
- If feature-model visualizations shall be explored with more flexibility, such as combining several visualization concepts in a view or applying tweaks on an existing visualization, the latest open-source visualization tools with proper documentation and community support, such as D3.js and FeatureIDE, should be considered. Using a 3D-rendering engine like Babylon.js allows to expand visualization concepts into a 3D shape, enabling a more comprehensive perspective.
- If little programming knowledge is a requirement, Google Charts or encapsulated D3.js components seem most useful.

Overall, we hope that our results and these conclusions help researchers as well as practitioners when designing novel techniques for engineering variant-rich systems that require visualizations.

6 THREATS TO VALIDITY

Selection bias is a potential threat to the validity of our study. We based our selection on two recent mapping studies and a manual search in the proceedings of three conferences, but we may still have missed relevant primary sources. Consequently, our study’s findings may not represent the entire universe of research on visualizations for variant-rich system engineering. Still, our findings provide a valuable and detailed overview of the state-of-the-art.

Publication bias may have impacted our findings. We can only rely on published papers, which may not accurately represent the current state-of-the-art. Unpublished research, open-source tools, or industry practices may also be relevant and should be included in future research. Still, we capture a large number of primary sources and provide helpful insights for researchers and practitioners.
7 CONCLUSION

Visualizations are a powerful tool for understanding and analyzing software-engineering data, also in the context of variant-rich systems. We have examined 64 papers that have proposed visualizations for variant-rich systems and, building on 12 different (types of) tools: Eclipse EMP-GEF, Prefuse, D3.js, Graphviz, CCVisu, Google Charts, ConExp, Moose, Processing, FeatureIDE, Babylon.js, and Adhoc tools. To provide a detailed overview, we compared the visualization concepts used for the entities of variant-rich system engineering, the underlying ideas, the existing evidence on their usefulness, and the tools used. Our findings can help researchers and practitioners to understand what visualizations may be most useful to reuse, and to identify opportunities for exploring novel ones. We further provide help when selecting underlying tools for implementing a visualization. However, we want to emphasize that more systematic evaluations and comparisons are needed to better understand which visualization concepts are better suited for what entities of variant-rich system engineering. In future work, we aim to build on our findings to develop novel visualizations for under-explored concepts.

REFERENCES


