Visualizing Architectural Evolution via Provenance Tracking: A Systematic Review

Kaitlynn Burgess
kate_burgess1@baylor.edu
Computer Science, Baylor University
Waco, Texas, USA

Dante Hart
dante_hart1@baylor.edu
Computer Science, Baylor University
Waco, Texas, USA

Amr Elsayed
amr_elsayed1@baylor.edu
Computer Science, Baylor University
Waco, Texas, USA

Tomas Cerny
tomas_cerny@baylor.edu
Computer Science, Baylor University
Waco, Texas, USA

Miroslav Bures
miroslav.bures@fel.cvut.cz
FEE, Czech Technical University
Prague, Czech Republic

Pavel Tisnovsky
ptisnovs@redhat.com
Red Hat
Brno, Czech Republic

ABSTRACT

Provenance tracking is used to record vital information such as user actions and the origin of data, but its potential has not been utilized with software architecture. Given the importance of provenance tracking, it can be seen as beneficial to understand the methods used to track this architecture evolution, as well as having methods to help visualize the architecture evolution. Throughout this paper, a systematic review is conducted addressing how provenance tracking can be used to track software architectural changes. Additionally, open-source provenance tracking tools, Trrack, ProvViewer, VisTrails, InDiProv, and GraphTrail are discussed to show how such functionality can be applied to visualize software architecture. In this study, we analyzed a final selection of 35 papers. Among these papers, we compile content from them to better understand the potential of how provenance tracking can be used to aid the visualization of software architecture. This analysis can be applied to existing provenance tracking visualization tools as well as benefit researchers or practitioners intending to maintain and trace software architecture.

CCS CONCEPTS

• **Software and its engineering → Visual languages**.

KEYWORDS

Provenance Tracking, Software Architecture, Visualization, Log Analysis, Mapping Study

ACM Reference Format:
Kaitlynn Burgess, Dante Hart, Amr Elsayed, Tomas Cerny, Miroslav Bures, and Pavel Tisnovsky. 2022. Visualizing Architectural Evolution via Provenance Tracking: A Systematic Review. In *International Conference on Research in Adaptive and Convergent Systems (RACS '22), October 3–6, 2022.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3538641.3561493

RACS '22, October 3–6, 2022, Virtual Event, Japan

ACM ISBN 978-1-4503-9398-0/22/10…$15.00
https://doi.org/10.1145/3538641.3561493

1 INTRODUCTION

Architecture evolution is defined as a high-level representation specifying the structure and interactions of components of a system that change over an allotted time frame [32]. Maintaining and tracking architectural changes is essential for organizations to produce high-quality products for clientele and produce valid and verifiable results in data-centered research [17, 32]. The expanding size and complexity of these platforms make managing applications more difficult, known as software degradation [14].

The next natural step to prevent this software degradation is using provenance tracking to visualize and trace architecture evolution by recording and showing each change. *Provenance tracking* shows derivation history of data [29]. Provenance graphs can provide easy-to-understand visualizations of architecture evolution and complexity. Therefore, using provenance graphs as a method to monitor architectural change is a viable, visual solution to understanding how component interactions can lead to architecture degradation or anti-patterns. Such component interactions can also be utilized to display the version history of software architecture.

This systematic literature review is focused on provenance tracking architecture evolution as discussed in current relevant publications as well as current software used for provenance tracking visualization. Be that as it may, we discovered limited information on architectural evolution monitored by provenance tracking. However, improving existing provenance visualization tools to generate an accurate representation of architectural changes using provenance tracking is plausible.

In this study, we analyzed 474 studies, from which we selected 35 directly relevant to the topic. The selected 35 publications discuss provenance models [35], provenance frameworks [24], provenance systems [33], provenance visualization tools [8], and integrating provenance with existing frameworks [23]. We identified five tools that are specialized to track provenance visually. Be as this may, collaboration, the capacity of history stored, and managing large data sets all presented to be an issue. Accessing provenance is a goal nearly all tracking tools have. However, support of collaboration, accessing previous versions of data, and being able to store large data are essential to tracing architecture. We discuss possible solutions to these obstacles.

2 PAPER ORGANIZATION

This paper is organized as follows. Section 3 discusses provenance tracking and the integration with architecture as well as this paper's focus and articles that were deemed relevant but did not encompass the scope of our research questions. Section 4 addressed how we formulated research questions and refined a search query. Section 5 discusses how each primary study of the search query was selected and trends pertaining to publication year and author's countries of origin. Section 6 discusses how we categorized the papers to answer the research questions. Section 7 addressed visualization methods used to trace provenance tracking. Section 8 describes the provenance tracking tools discovered which can be used to trace architecture evolution. Section 9 addressed the solutions to our research questions. Section 10 addresses the threats to the validity of our systematic literature review. Section 11 discusses the future directions for research as well as difficulties faced throughout this systematic literature review. Lastly, section 12 concludes this paper and restates what this paper is aiming to achieve and has discovered.

3 BACKGROUND

Software architecture evolves in parallel with software development. Monitoring such changes is becoming increasingly complex as platforms grow. As architecture expands, a system must remain flexible and maintain its original functionality [14]. Therefore, visualizing and tracking how modifications impact an architecture via provenance tracking could be beneficial for software maintenance and to avoid architecture degradation. Thus, using provenance tracking to display such architectural changes is a potential solution to visualize and understand these changes easier.

Provenance is information representing entities, agents, activities and their association with other components [35]. Provenance tracking, in general, is used to monitor changes in data. Provenance tracking can be used to detect architectural changes via the recording of evolving architecture in a series of graphical versions [8]. The importance of understanding modifications in a given architecture is useful for organizations to understand how changes may influence the architecture or impact its components.

Logging architectural changes are not always necessarily adequate on their own. Architects may not understand how software changes will impact the architecture as a whole or how they will impact other components within the architecture. Another issue with monitoring architecture changes is visually showing previous versions of the architecture once a change has been made. More often than not, the methods used to track architecture evolution do not use provenance tracking visualization, as a result, most versions of previous architecture models are quickly rendered obsolete and can create difficulties for communication in project teams. Additionally, a lack of provenance tracking for visualization of a software architecture system over time can potentially increase software degradation as only the original developers of a said system can fully analyze and recall a project's history efficiently.

In one study, 'Big Data Provenance: Challenges, State of the Art and Opportunities' [30] researchers discuss issues that arise with big data provenance tracking. Although helpful to understand issues with provenance tracking, this does not encompass the scope of our research. Another publication, 'A Semantic Foundation for Provenance Management' [27] discusses who, what, when, where, and why the aspect of provenance tracking. Initially thought to be beneficial to answer our research questions, instead, the focus of the paper exceeded the range of our paper's intended goal.

Our focus in this study pertains to how the process of visualizing architecture usually evolves. We aim to identify current works applying provenance tracking to architecture evolution and its visualization.

4 MAPPING STUDY METHOD

We followed the methodology formulated by Kitchenham and conducted a systematic literature review to provide in-depth explanations of how our research progressed over time [19]. As seen in Figure 1 we utilized a multi-phase approach to gather and organize relevant research articles. In phase one, we collaborated to generate a set of research questions. To progress the understanding of our area of research, we frequently revised these questions to best reflect our intentions with this paper. This literature review was centered around the four research questions defined below:

- RQ1 How is provenance tracking used to visualize and trace changes in software architecture/evolution? What are the common visualization and internal representation approaches to provenance tracking of software architecture evolution?
- RQ2 What specifics of provenance tracking are considered in the architecture provenance visualization?
- RQ3 How can we visualize an intermediate change in software architecture given to a selected baseline?
- RQ4 What are the current issues/challenges in provenance tracking in relation to software architecture visualization? How can these methods be improved?

These research questions aim to utilize existing provenance visualization tools to trace architectural evolution. Once these questions were formulated, we progressed to phase two of the filtration process. Phase two consisted of constructing a list of search queries among defined indexers to establish a foundation for our systematic literature review. The defined indexers used were: ACM, SpringerLink, IEEE, Scopus, and Science Direct.

We, tested five search queries with varying terms, however, all but one query would be exceedingly large with over 500 publications or the papers in a given query were not relevant to our research. As our topic centers on provenance tracking, our search query is primarily based on the phrase "provenance tracking". To better focus the query results on publications relevant to provenance visualization, we included the terms "graph OR model OR visual OR view OR representation" which broadened the query to deliver provenance visualization tools. Following this, we concerned ourselves with provenance visualization as it pertains to architecture; thus, the last search terms added to the query were "architecture OR design" which allowed studies focused on architecture visualization to be included in the query. Finally, we sought the most recent discoveries as they pertain to our query; subsequently, we restricted the results to papers published in the last ten years (2012 - 2022). Once these search terms and filters were compiled into one query, we used this query across the indexers for a total of 466 results. Be as this may, during the filtration process, we snowballed eight scholarly publications leading to a total of 474 publications

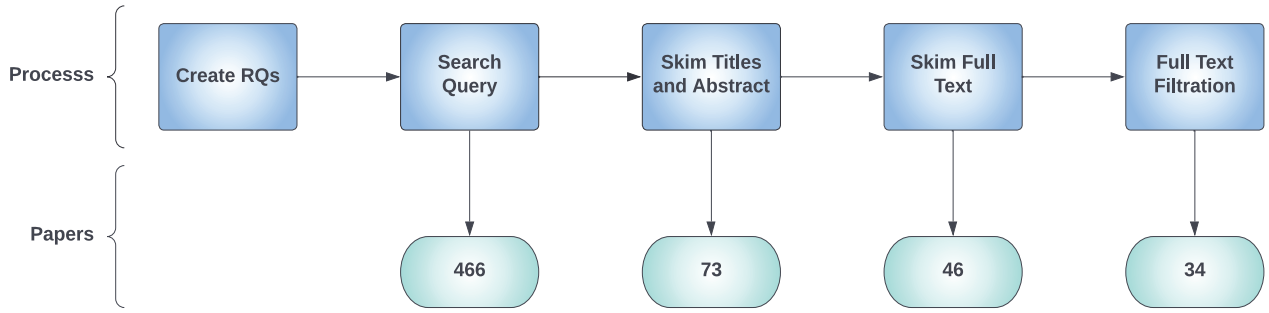

Figure 1: Query Filtration Process

as shown in Table 2 and Figure 1. Therefore, after assessing which query would be best for our research, the final query is shown in Table 1.

Table 1: Search Query

Search Query
"provenance tracking" AND (Graph OR model OR visual OR view OR representation) AND (Architecture OR Design)

Table 2: Search Query Results for Various Index Sites

Indexer	Search Results	Post Abstract and Title Filtration	Total Relevant
ACM DL	232	31	13
IEEE Xplore	2	1	1
SpringerLink	86	10	5
Scopus	21	8	5
ScienceDirect	125	15	6
Others	8	8	8
Total	474	73	35

Once these 474 papers were assembled, the 3-step filtration process began. The first phase consisted of narrowing down papers based on their title followed by the abstract. Papers that did not meet inclusion criteria or met exclusion criteria were considered not relevant to our research and removed from the pool. The third phase consisted of light, full text skimming through the remaining papers. If any paper met the exclusion criteria shown in Table 3 they were considered not relevant for our research. The final phase consisted of full-text reads of the remaining 46 papers, which were then included or excluded based on criteria that will be provided in the below section. During each step of the process, article conflicts and partner disagreements were discussed and resolved before proceeding to the next step. Once the filtration process was completed, the final paper pool consisted of 35 research articles.

Table 3: Inclusion and Exclusion Criteria

Inclusion Criteria
(1) Architecture evolution studies. (2) Provenance tracking studies. (3) Provenance tracking impacting architectural visualization studies. (4) Provenance tracking visualization tools studies. (5) Provenance tracking correlating to visualization of architecture studies. (6) Papers addressing correlations between provenance tracking and architecture. (7) Evolution of architecture or architecture complexity publications.
Exclusion Criteria
(1) Duplicate papers found across indexers. (2) Paper whose findings are out of our scope of research. (3) Publications based on expert opinion. (4) Case studies without generalization. (5) Papers that are non-peer reviewed. (6) Papers with no full-text availability. (7) Publications not written in English.

Of the 35 relevant articles, multiple articles were relevant involving provenance tracking, yet how they utilized provenance tracking often regarded areas that were decided to be too niche to be included as relevant in our study. Despite this, we were able to analyze and study our remaining papers to help answer our research questions that will be demonstrated in the next upcoming sections.

4.1 Author Country of Origin

Among the research papers aggregated, there was only one author overlap in the papers discovered. Additionally, most if not all papers

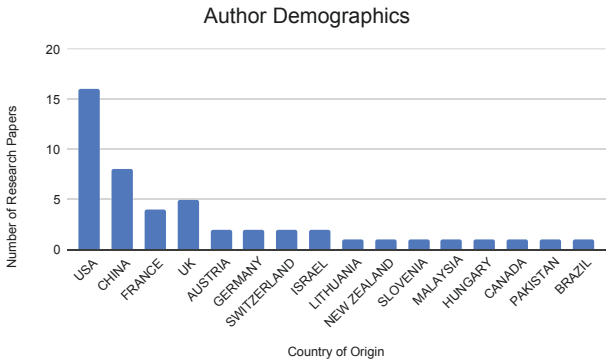

Figure 2: Research Article Country of Origin

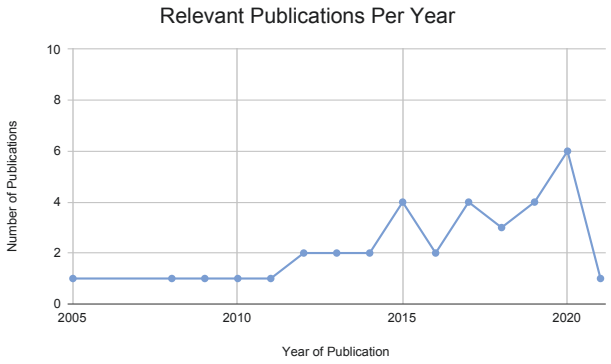

Figure 3: Number of papers found per year

consisted of more than three authors. This discovery demonstrates a balanced and non-biased distribution in regards to our research database. Additionally, in regards to the country of origin as seen in Figure 2, you can see for each paper, there is a country of origin recorded to show the distribution of papers found across the world. Of our 35 papers, most authors come from a mixture of the United States, China, or Europe, with the United States and China being the two most dominant contributors with them taking up 41 percent and 23 percent of all the research articles, respectively.

5 QUERY ANALYSIS RESULTS

Each relevant publication was analyzed, if a paper discussed provenance tracking visualization or architecture evolution it was considered useful for answering what specifics of provenance tracking are used to visualize architectural changes. Moreover, papers analyzing provenance visualization tools were used to assess issues about provenance tracking as a means to monitor architecture changes, how each tool uses a selected baseline for tracking such changes visually, and what are internal approaches to tracing changes in data from a visual standpoint. Papers discussing provenance models or architecture evolution were used to elaborate on how provenance visualization can be used as a means to trace architectural changes and what problems arise when tracing such evolution.

In the upcoming section, we discuss the findings of our investigation and answer the research questions in varying subsections. We then conclude with a brief overview of the validity of our study.

5.1 Modeling Architectural Trends

Monitoring architecture evolution has been a large focus for many industries. In regards to modeling architecture in general, modeling can be generalized into two separate categories: process-centric and data-centric [9]. While it is common to use process-centric modeling to visualize *where* the data is flowing in an architecture graph, we want to focus on the architectural design itself and analyze the components via a data-centric approach. We desire to utilize data provenance tracking to visualize changes on the architectural components in the system itself, rather than the flow of data within that system. However, when using this approach, the varying methods to use provenance tracking to record such changes is scarce with only five tools being discovered in our query. As mentioned previously, most research articles marked as relevant for provenance tracking only desired to analyze changes in their respective area of study, rather than the generalization of the software architecture itself [4, 7, 15, 29, 33]. Nonetheless, from 2005 to 2020 it can be seen in Figure 3, there is a spiked interest in provenance tracking and its applications. This increase in popularity demonstrates how studying provenance tracking is becoming a lucrative field of study.

6 PAPER CATEGORIZATION

Table 4: Citations Per Category

Category	Citations
Graph	[7] [12] [17] [13] [18] [25] [26] [28] [36]
Data Management	[2] [3] [7] [17] [36]
Structural Changes	[2] [8]
Component Relationships	[2] [8][20] [36]
Version History	[2] [8] [10] [20]

Each publication was associated with a respective category as demonstrated in Table 3. Certain primary studies were mapped to multiple categories based on their content. The essential categories are highlighted in Table 4: interactive visualization, graphs, data management, structural changes, component relationships, and version history. These categories were formulated based on reading and clustering. Throughout the literature review, graphs were considered essential to generate a visual representation of provenance tracking. Data management; in our scope, was defined as ways to ensure provenance data is secure and how provenance data can be stored or retrieved. In addition, structural changes were equated to provenance graph behavior: colors, shapes, movement, adding, and deleting. Component relationships were considered vital to describe how nodes should be connected and the relevance of said connection. Since provenance tracking describes the lineage of data version history was deemed vital to recording and monitoring changes. Papers addressing provenance tracking visualization were linked to either interactive visualization, graph-based

visualization, data management, structural changes, or defining relationships. Publications discussing the storage and retrieval of provenance-tracked data were assigned to the data management category. If a paper mentioned data integrity or provenance security the paper was assigned to data management. All papers discussing provenance visualization tools were assigned to a version history to allow for a compare and contrast analysis.

In this systematic literature review, five interactive provenance tracking tools were found and assessed to have a means to visualize changes in architecture evolution: Trrack library [8], Prov Viewer [7], VisTrails [2], GraphTrail [10], and InDiProv [17]. Before preceding, each tool discussed is not developed for monitoring architectural changes specifically. However, extending these tools to fit our research goals is possible. Therefore, we will discuss how each tool uses visualization of provenance, the limitations of each tool, the specifics of the tools, and how each application can manage data from a given baseline.

7 VISUALIZATION METHODOLOGY

7.0.1 Interactive Visualization. Interactive visualization is defined as allowing in-depth analysis of data via chart manipulation with motions, colors, shapes, and optical cues. Most provenance visualization tools rely on interactive functionality to record user interactions. Such actions allow the user to keep a fluid and up-to-date graph which, when used to model software architecture, allows the user to prevent software degradation. Ideally, the most common functionality for node linked interactive graphs and workflow diagrams involved hovering, selecting nodes and moving nodes [2, 8, 20].

7.0.2 Graph. Graphical visualization involves the use of symbols or other visual aids to represent data. Different graphs show non-identical information. For example, as seen in Table 4 we found a multitude of graphs used for provenance tracking: bar graphs, tables, matrices, tag clouds, scatter plots, hybrids, workflows, and node-based graphs. However, their applications were not directed toward software architecture. Be this as it may, utilizing various graphs to see architecture evolution from varying perspectives can be deemed as beneficial [11, 13]. Throughout this systematic literature review, we found bar graphs [7], scatter plots [28], line charts [18], and workflow diagrams [12, 17, 26] used as means to capture interactions between components and aid in provenance tracking visualization. Other researchers utilized directed acyclic graphs (DAG's) as a mechanism to record information [36]. Some publications address provenance graphs employing node size representation of the rank value of nodes, and edge width denoting the impacts one node had on another [25]. Each provenance graph discussed can be used as a visual snapshot of software architecture from a select period of time.

7.1 Data Management

To trace software architectural evolution, provenance tracking tools should provide a tamper-resistant data archive for analysis results [36]. Maintaining the integrity of data for software architecture is vital to generating reproducible and trustworthy data. Certain approaches to maintaining data integrity include the use of novel algorithms, provenance encoding, and adding a tamper-resistant provenance layer [36]. Other publications suggest the establishment of fine-grained access control policies to protect provenance information [3].

Besides the maintenance of data integrity, a provenance tool must be able to store data and retrieve data from previous sessions in order to record architectural changes. Throughout this literature review, we found instances where tools utilize a server-client approach to store and retrieve data. Other publications describe writing data to XML or JSON files for easy retrieval [7]. The tool VisTrails uses an XML dialect that acts as a data pipeline enabling easy sharing capabilities and maintaining visualization provenance [2]. Whilst, InDiProv utilizes PROV-XML to serialize instances of the PROV data model to XML [17]. Be as this may, every provenance task: workflow creation, storing provenance, and tracking is written as a JSON exchange messed and sent to the server [17].

7.1.1 Structural Changes. Throughout this literature review, we found the most essential structural changes required to monitor architecture evolution via provenance tracking in an interactive node-based graph to be adding, deleting, and moving nodes. We found publications that describe provenance tools recording hovering over nodes, moving nodes, undo, and redo functionality on the graph post a modification [8]. Each of these functionalities is beneficial, however, from the tools found in this literature review monitoring adding, and deleting nodes is not a readily available functionality. Other publications address the functionality of direct manipulation of objects in a given visualization; allowing scaling and positioning [2].

7.1.2 Component Relationships. From the node-base provenance graphs described previously, it was found that node grouping established by a developer can be used to generate a state for a given provenance tracking application. The specificity of grouping can be applied to software architecture in which the grouping of nodes represents components such as data sets, services, and portrays how common these two components interact. Other authors use node-based graphs where nodes represent expressions, and edges represent the node's sub-expressions [36]. Furthermore, other interactive node-based graphs show entities as circles, activities as vertices, and agents as pentagons [20]. In addition, edges were used to identify relationships between nodes in order to convey how an interaction may lead to positive or negative effects. Other examples use nodes to correspond to a module element in a given schema [2]. In this same schema, an edge between two nodes corresponds to a connect element where connections show the data dependencies among varying modules. One such application uses an abstract approach to establishing node link interactions in which the user establishes the node interactions [8].

7.1.3 Version History. To monitor architectural changes from a selected baseline a provenance visualization tool should be able to store various versions of given software architecture. To elaborate, upon each change to a software architecture a snapshot of the provenance graph should be recorded in order to allow an easy-to-understand visualization of how the architecture evolves. During our literature review, we found provenance tracking tools which upon the user creating numerous provenance graphs, the tool automatically connects the graphs via links to indicate a chain of actions

[10]. This functionality in turn can be used to analyze changes in a given architecture from a selected baseline or analyze previous versions of a given architecture. Furthermore, we found tools that allow the storage of various provenance graphs in a single domain and allow developers to merge two existing graphs which can be used to trace changes in version history [20]. Even more so, we found a provenance tracking tool that allows a multi-view scenario where researchers can see side-by-side direct changes in a given set of data [2]. In addition, one provenance tracking tool utilizes an interactive node-link tree to record version history; each node represents a previous version of the same provenance graph [8].

8 PROVENANCE VISUALIZATION TOOLS

Table 5: Provenance Tracking Tools Analysis

	Prov Viewer	Trrack-/Track-Vis	Graph-Trail	In-DiProv	Vis-Trail
Bar Graphs		✓	✓		
Tables			✓		
Matrices			✓		
Tag Clouds			✓		
Scatter Plots		✓	✓		
Hybrid Graphs		✓	✓		
Workflows				✓	✓
Node-based graphs	✓	✓	✓		

Throughout this systematic literature review, we discovered five provenance visualization tools that can have the means to monitor architecture evolution. As shown in Table 5 each tool has diverse provenance graphs which can be used to monitor architecture evolution.

To begin, one tool GraphTrail shows exploration history via provenance tracking user actions and displaying visual and textual cues [10]. GraphTrail uses bar graphs, tables, matrices, tag clouds, scatter plots, hybrid graphs, and node-based graphs as visualization techniques to record data. Be as this may, GraphTrail an open source version is currently unavailable.

The next tool to be analyzed is VisTrails. VisTrails records the provenance of visualization data products [2]. VisTrails also has a built-in history tree that keeps track of all changes for a given workflow graph. Besides this provenance utility, VisTrails also has an intuitive user interface for adding and connecting different modules within the program. This allows users to easily reconstruct, analyze, and view software architecture evolution.

Another provenance tool Prov Viewer is a graph-based visualization tool for exploration provenance [20]. Prov Viewer uses provenance tracking and has been used to record game analytics of player-enemy interactions to give developers data related to locations where players face issues the most [7].

The fourth tool to be discussed, InDiProv, is a provenance application built on a server/client architecture[17]. The server is an engine responsible for provenance and data management whilst the client handles workflows and pipelines [17]. This functionality allows for faster data storage for provenance tracking. However, testing with this tool proved unfruitful as the tool is inaccessible. This could be due to it being a non-released prototype or it being a proprietary tool.

The final tool, Trrrack, and its optional visual library TrackVis interact to make a provenance visualization tool that has the ability to trace architectural changes. Developers must import the data desired to monitor and apply local changes to the library to generate a provenance visualization best suited for their intended research. Trrack provides node-based graphs, scatter plots, bar charts, and hybrid graphs as means to monitor software architecture. Additionally, Trrack provides abstract and modular functionality by allowing users to create custom node-link relationships.

Of the tools mentioned throughout this systematic review, each has different limitations in regards to visualization of provenance. A direct issue faced by Trrack and GraphTrail is managing large data sets [8, 20]. Issues pertaining to VisTrails is how collaboration is not supported without explicitly transferring history [17]. InDiProv supports visualization, however, in the publication discussing InDiProv the examples of visual provenance were minuscule. Our research intends on improving existing tools and achieving the best user interface to display architectural changes.

For Trrack two solutions are proposed to overcome large data sets. The first proposed solution is to store the data set as an attribute node but an issue arises when mixing actions and data [8]. The second solution discussed is to write the data set to an output file but this makes the association between the data and application state weak [8]. GraphTrail utilizes aggregation according to node and edge attributes to reduce the impact of large data sets [10].

9 RESEARCH QUESTION ANSWERS

How is provenance tracking used to visualize and trace changes in software architecture/evolution? What are the common visualization and internal representation approaches to provenance tracking of software architecture evolution?

Of the studies found that report how provenance tracking is used to trace data changes visually, most focused on recording user sessions and node behavior in interactive provenance graphs. Other publications address diverse forms of provenance graphs. As stated in section 7, interactive visualization and graphs are the most common forms to visualize how provenance tracking is used in software architecture and evolution. This is usually done by using a provenance tracking visualization tool, which has been discussed in section 8. Using these tools, a user can utilize provenance tracking to show changes in a myriad of different formats, such as scatter plots and workflow diagrams to name a few.

What specifics of provenance tracking are considered in the architecture provenance visualization?

Understanding the specifics of provenance tracking considered for monitoring software architecture evolution is important because it can assist architects in comprehending impact analysis. Throughout our literature review, we found how relationships in provenance graphs are defined, how to maintain data, and the data required to monitor software architecture changes as the essential specifics of provenance tracking architecture evolution. When

considering visualization, provenance tracking is used to trace the data management of a software architecture graph to ensure the data being analyzed is tamper-resistant to ensure a consistent foundation for analysis. Additionally, provenance tracking records the structural change of a software architecture graph, the component relationships of different node/entities in the graph, and different versions of the graph that have been saved or recorded.

How can we visualize an intermediate change in software architecture given to a selected baseline?

It has been shown that accessing the previous version history of architecture is imperative to gain an insight into how a change has impacted a given system [32]. When a modification has been made to software architecture tracing a change from a selected baseline is necessary to understand the impact on components. It is possible to visualize intermediate changes in software architecture through provenance tracking software tools. These tools utilize graphing assets to visualize the software architecture, and use provenance tracking to record and show the different versions of the software architecture when it has been modified. Additionally, the provenance graphing tools can demonstrate component relationships between different nodes in an software architectural system as well as structural changes to the software architecture system itself.

What are the current issues/challenges in provenance tracking in relation to software architecture visualization? How can these methods be improved?

Analyzing provenance tracking visualization throughout the 35 relevant publication suggests monitoring architecture evolution will equate to two main issues: the capacity of how much data can be stored by a tool and storing large data sets in a selected node. Be that as it may, we also found potential solutions to some of the more pressing issues. As discussed in section 8, a plausible solution was to store the data set as an attribute node, with another solution for storage capacity being to write the data set to an output file. However, these are not the only matters of concern with software architecture visualization.

10 THREATS TO VALIDITY

A major consideration in the validity of the relevant publications is the limitation of information describing software architecture provenance tracking in our search query. When considering how provenance tracking intertwines with software architecture it is vital to see how other researchers have or are implementing such an intervention. In our case, this research was limited greatly. Correspondingly, connecting two fields of research was the only viable solution to answering our questions. Due to this, we must assert that there may be tools and articles outside of our finalized search query.

Another consideration is a bias throughout most of the primary studies. For example, in publications proposing new provenance models, frameworks, or visualization tools there was an increase in selective outcome reporting. Thus, when analyzing new provenance visualization tools, challenges pertaining to the application were rarely asserted.

11 DISCUSSION AND FUTURE DIRECTIONS

11.1 Importance of Security

An observation noticed throughout our query was the reoccurring notion of tracking provenance securely and maintaining provenance integrity. Of the papers we considered relevant, nine stressed the importance of their provenance methods being secure.[1, 3, 4, 9, 22, 23, 29, 31, 36] Additionally, a notion of not only being a secure system in it of itself, but also being resistant to tampering was seen. One research article deemed relevant utilized provenance for the express purpose of monitoring potential attacks and undesired modification of data in their system by using the provenance semiring model to apply provenance polynomial expressions to tuples[36]. Therefore, maintaining a tamper-resistant provenance system for architecture was considered an essential function and used to answer research question 2.

11.2 Lack of Generalization

A difficulty encountered during our research revolved around finding papers that discussed visualization of software architecture from a general and abstract perspective. Many papers discussed methodologies to visualize their specific provenance methods in a myriad of graphs earlier described, yet these papers could not help us visualize architecture in general. Usually these papers revolved around proprietary workflow visualization that we deemed to be not as helpful as other visualization tools and methods [5, 6, 15, 16, 24, 25, 34]. We could use these papers as examples, but the tools described work only for the researchers' specific fields. If these tools were used in a more abstract and general setting, a disadvantage would be how a developer not involved in the previous research uses the application for newly intended purposes. For example, a provenance tracking-based research article focused on provenance tracking data-centric processes, and their provenance visualization methods considered their methodology with data-dependent processes and not software architecture as a whole.[9] Had this been the case, we could've used and applied their methods to our research to greater effect.

11.3 Future Directions

One of our future aspirations is to discover or implement a consistent, abstract, and general method to include functionality for software architecture graphs for provenance tracking to allow developers to trace changes from a selected baseline. In this way, one can not only visualize provenance tracking for architecture but also have added versatility when working with a project for a team. Rather than always starting at a consistent and immutable base graph, with the ability to import and update the base graph, we believe that this would drastically improve productivity. Additionally, with the functionality of provenance tracking for a given project, the history of the different architecture graphs would be able to be traced and replicated at will, ensuring data loss prevention. Additionally, another goal is to continue development with existing open-source tools to try to implement a prototype showcasing the functionality of the tool with working on the tracking of provenance with software architecture graphs. Another goal is to continue the search for more abstract provenance tracking visualization tools

that can be applied for software architecture in general or to encourage the development of new tools that can be applied to fulfill the requirements of provenance tracking for software architecture. Additionally, in the interests of visualizing architecture, provenance tracking methods could also be utilized for tracking architectural languages and microservices.[21] A future research goal consists of an understanding of how provenance tracking could be applied to help visualize microservice architecture as well.

12 CONCLUSION

In this paper, we investigated how varying tools use provenance tracking to trace visual changes in data and how this applies to architecture evolution. We address the specifics of provenance tracking used to track changes in a given architecture and how to trace changes from a selected baseline. Furthermore, we discuss issues that pertain to architecture evolution, namely maintaining large data sets, the capacity of history storage, and the ability to have version history.

Visualizing architecture via provenance tracking can improve project design analysis and prevent software degradation for a software architecture-based system by introducing provenance not just on data within the architecture, but the architecture itself. In this way, an architecture system's evolution can be mapped and documented for posterity. Utilizing more abstract and versatile tools, such as Trrack, Prov Viewer, and VisTrails, one can help monitor software architectural evolution over time. However, current support and availability for such software functionality are slim, with some tools either still in beta releases or support ended altogether.[2, 20] Such all-encompassing functionality described previously is not standard for other provenance tools, and it would be beneficial for this field if more options were available or if existing tools were continued to be developed.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 1854049 and a grant from Red Hat Research https://research.redhat.com.

REFERENCES

[1] Idrees Ahmed, Abid Khan, Mansoor Ahmed, and Saif ur Rehman. [n. d.]. Order preserving secure provenance scheme for distributed networks. 82 ([n. d.]), 99–117. https://doi.org/10.1016/j.cose.2018.12.008

[2] L. Bavoil, S.P. Callahan, P.J. Crossno, J. Freire, C.E. Scheidegger, C.T. Silva, and H.T. Vo. [n. d.]. VisTrails: enabling interactive multiple-view visualizations. In *VIS 05. IEEE Visualization, 2005.* (2005-10). 135–142. https://doi.org/10.1109/VISUAL.2005.1532788

[3] Elisa Bertino, Gabriel Ghinita, Murat Kantarcioglu, Dang Nguyen, Jae Park, Ravi Sandhu, Salmin Sultana, Bhavani Thuraisingham, and Shouhuai Xu. [n. d.]. A roadmap for privacy-enhanced secure data provenance. 43, 3 ([n. d.]), 481–501. https://doi.org/10.1007/s10844-014-0322-7

[4] C. Bier. [n. d.]. How usage control and provenance tracking get together - A data protection perspective. 13–17. https://doi.org/10.1109/SPW.2013.24

[5] P. Bourhis, D. Deutch, and Y. Moskovitch. [n. d.]. Equivalence-Invariant Algebraic Provenance for Hyperplane Update Queries. 415–429. https://doi.org/10.1145/3318464.3380578 ISSN: 0730-8078.

[6] Dai Chaofan, Zhang Ran, Li Pei, Wang Wenqian, and Cao Zewen. [n. d.]. A Minimal Attribute Set-oriented Data Provenance Method. In *Proceedings of the International Conference on Big Data and Internet of Thing* (2017-12-20) *(BDIOT2017)*. Association for Computing Machinery, 1–5. https://doi.org/10.1145/3175684.3175686

[7] Troy Costa Kohwalter, Felipe Machado de Azeredo Figueira, Eduardo Assis de Lima Serdeiro, Jose Ricardo da Silva Junior, Leonardo Gresta Paulino Murta, and Esteban Walter Gonzalez Clua. [n. d.]. Understanding game sessions through provenance. 27 ([n. d.]), 110–127. https://doi.org/10.1016/j.entcom.2018.05.001

[8] Zach Cutler, Kiran Gadhave, and Alexander Lex. [n. d.]. Trrack: A Library for Provenance-Tracking in Web-Based Visualizations. In *2020 IEEE Visualization Conference (VIS)* (2020-10). 116–120. https://doi.org/10.1109/VIS47514.2020.00030

[9] Daniel Deutch, Yuval Moskovitch, and Val Tannen. [n. d.]. Provenance-based analysis of data-centric processes. 24, 4 ([n. d.]), 583–607. https://doi.org/10.1007/s00778-015-0390-5

[10] Cody Dunne, Nathalie Henry Riche, Bongshin Lee, Ronald Metoyer, and George Robertson. [n. d.]. GraphTrail: analyzing large multivariate, heterogeneous networks while supporting exploration history. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2012-05-05). ACM, 1663–1672. https://doi.org/10.1145/2207676.2208293

[11] R. Genquan, Z. Li, W. Jianmin, and L. Yinbo. [n. d.]. One method for provenance tracking of product lifecycle data in collaborative service environment. 347–356. https://doi.org/10.1109/CyberC.2011.62

[12] Sandra Gesing, Malcolm Atkinson, Rosa Filgueira, Ian Taylor, Andrew Jones, Vlado Stankovski, Chee Sun Liew, Alessandro Spinuso, Gabor Terstyanszky, and Peter Kacsuk. [n. d.]. Workflows in a dashboard: a new generation of usability. In *Proceedings of the 9th Workshop on Workflows in Support of Large-Scale Science* (2014-11-16) *(WORKS '14)*. IEEE Press, 82–93. https://doi.org/10.1109/WORKS.2014.6

[13] Eric Griffis, Paul Martin, and James Cheney. [n. d.]. Semantics and provenance for processing element composition in dispel workflows. In *Proceedings of the 8th Workshop on Workflows in Support of Large-Scale Science* (2013-11-17) *(WORKS '13)*. Association for Computing Machinery, 38–47. https://doi.org/10.1145/2534248.2534252

[14] Mohamed Oussama Hassan and Henri Basson. [n. d.]. Tracing Software Architecture Change Using Graph Formalisms in Distributed Systems. In *2008 3rd International Conference on Information and Communication Technologies: From Theory to Applications* (2008-04). 1–6. https://doi.org/10.1109/ICTTA.2008.4530365

[15] Lianlian He, Peng Yue, Liping Di, Mingda Zhang, and Lei Hu. [n. d.]. Adding Geospatial Data Provenance into SDI—A Service-Oriented Approach. 8, 2 ([n. d.]), 926–936. https://doi.org/10.1109/JSTARS.2014.2340737 Conference Name: IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing.

[16] Jingmei Hu, Jiwon Joung, Maia Jacobs, Krzysztof Z. Gajos, and Margo I. Seltzer. [n. d.]. Improving data scientist efficiency with provenance. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering* (2020-06-27) *(ICSE '20)*. Association for Computing Machinery, 1086–1097. https://doi.org/10.1145/3377811.3380366

[17] C. Hänel, M. Khatami, T.W. Kuhlen, and B. Weyers. [n. d.]. Towards Multi-user Provenance Tracking of Visual Analysis Workflows over Multiple Applications. 23–27. https://doi.org/10.2312/eurorv3.20161112

[18] KarvounarakisGrigoris, GreenTodd J, IvesZachary G, and TannenVal. [n. d.]. Collaborative data sharing via update exchange and provenance. ([n. d.]). https://doi.org/10.1145/2500127 Publisher: ACM PUB27 New York, NY, USA.

[19] Barbara Kitchenham, Pearl Brereton, David Budgen, Mark Turner, John Bailey, and Stephen Linkman. [n. d.]. Systematic literature reviews in software engineering-A systematic literature review. 51 ([n. d.]), 7–15. https://doi.org/10.1016/j.infsof.2008.09.009

[20] Troy Kohwalter, Thiago Oliveira, Juliana Freire, Esteban Clua, and Leonardo Murta. [n. d.]. *Prov Viewer: A Graph-Based Visualization Tool for Interactive Exploration of Provenance Data.* https://doi.org/10.1007/978-3-319-40593-3_6 Pages: 82.

[21] Luka Lelovic, Michael Mathews, Amr Elsayed, Tomas Cerny, Karel Frajtak, Pavel Tisnovsky, and Davide Taibi. 2022. Architectural Languages in the Microservice Era: A Systematic Mapping Study. In *Architectural Languages in the Microservice Era: A Systematic Mapping Study.* Association for Computing Machinery, New York, NY, USA, 8. https://doi.org/10.1145/3538641.3561486

[22] Zitong Li, Xiang Cheng, Lixiao Sun, Ji Zhang, Bing Chen, and Weizhi Meng. [n. d.]. A Hierarchical Approach for Advanced Persistent Threat Detection with Attention-Based Graph Neural Networks. 2021 ([n. d.]). https://doi.org/10.1155/2021/9961342

[23] Cong Liao and Anna Squicciarini. [n. d.]. Towards provenance-based anomaly detection in MapReduce. In *Proceedings of the 15th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing* (2015-05-04) *(CCGRID '15)*. IEEE Press, 647–656. https://doi.org/10.1109/CCGrid.2015.16

[24] P. Lüthi, T. Gagnaux, and M. Gygli. [n. d.]. Distributed ledger for provenance tracking of artificial intelligence assets. 576 LNCS ([n. d.]), 411–426. https://doi.org/10.1007/978-3-030-42504-3_26 ISBN: 9783030425036.

[25] Shiqing Ma, Yousra Aafer, Zhaogui Xu, Wen-Chuan Lee, Juan Zhai, Yingqi Liu, and Xiangyu Zhang. [n. d.]. LAMP: data provenance for graph based machine learning algorithms through derivative computation. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering* (2017-08-21) *(ESEC/FSE 2017)*. Association for Computing Machinery, 786–797. https://doi.org/10.1145/3106237.3106291

[26] Andrius Merkys, Nicolas Mounet, Andrea Cepellotti, Nicola Marzari, Saulius Gražulis, and Giovanni Pizzi. [n. d.]. A posteriori metadata from automated provenance tracking: integration of AiiDA and TCOD. 9, 1 ([n. d.]), 56. https://doi.org/10.1186/s13321-017-0242-y

[27] Sudha Ram and Jun Liu. [n. d.]. A Semantic Foundation for Provenance Management. 1, 1 ([n. d.]), 11–17. https://doi.org/10.1007/s13740-012-0002-0

[28] Guillaume Rousseau, Roberto Di Cosmo, and Stefano Zacchiroli. [n. d.]. Software provenance tracking at the scale of public source code. 25, 4 ([n. d.]), 2930–2959. https://doi.org/10.1007/s10664-020-09828-5

[29] Marten Sigwart, Michael Borkowski, Marco Peise, Stefan Schulte, and Stefan Tai. [n. d.]. Blockchain-based Data Provenance for the Internet of Things. In *Proceedings of the 9th International Conference on the Internet of Things* (2019-10-22). ACM, 1–8. https://doi.org/10.1145/3365871.3365886

[30] Jianwu Wang, Daniel Crawl, Shweta Purawat, Mai Nguyen, and Ilkay Altintas. [n. d.]. Big data provenance: Challenges, state of the art and opportunities. In *2015 IEEE International Conference on Big Data (Big Data)* (2015-10). 2509–2516. https://doi.org/10.1109/BigData.2015.7364047

[31] David Wilkinson, Luís Oliveira, Daniel Mossé, and Bruce Childers. [n. d.]. Software Provenance: Track the Reality Not the Virtual Machine. In *Proceedings of the First International Workshop on Practical Reproducible Evaluation of Computer Systems* (2018-06-11). ACM, 1–6. https://doi.org/10.1145/3214239.3214244

[32] Byron J. Williams and Jeffrey C. Carver. [n. d.]. Characterizing software architecture changes: A systematic review. 52, 1 ([n. d.]), 31–51. https://doi.org/10.1016/j.infsof.2009.07.002

[33] Yinjun Wu, Val Tannen, and Susan B. Davidson. [n. d.]. PrIU: A Provenance-Based Approach for Incrementally Updating Regression Models. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (2020-06-11) *(SIGMOD '20)*. Association for Computing Machinery, 447–462. https://doi.org/10.1145/3318464.3380571

[34] Yulai Xie, Dan Feng, Xuelong Liao, and Leihua Qin. [n. d.]. Efficient monitoring and forensic analysis via accurate network-attached provenance collection with minimal storage overhead. 26 ([n. d.]), 19–28. https://doi.org/10.1016/j.diin.2018.05.001

[35] Mingda Zhang, Peng Yue, Zhaoyan Wu, Danielle Ziebelin, Huayi Wu, and Chenxiao Zhang. [n. d.]. Model provenance tracking and inference for integrated environmental modelling. 96 ([n. d.]), 95–105. https://doi.org/10.1016/j.envsoft.2017.06.051

[36] Nan Zheng and Zachary G. Ives. [n. d.]. Compact, tamper-resistant archival of fine-grained provenance. 14, 4 ([n. d.]), 485–497. https://doi.org/10.14778/3436905.3436909