

This version of the contribution has been accepted for publication, after peer review but is not the Version of Record and does not reflect post-acceptance improvements, or any corrections. The Version of Record is available online at: [https://doi.org/10.1007/978-3-031-98033-6\\_4](https://doi.org/10.1007/978-3-031-98033-6_4)

Use of this Accepted Version is subject to the publisher's Accepted Manuscript terms of use <https://www.springernature.com/gp/open-research/policies/accepted-manuscript-terms>

# VR-SBOM: Visualization of Software Bill of Materials and Software Supply Chains in Virtual Reality

Roy Oberhauser<sup>[0000-0002-7606-8226]</sup>

Computer Science Dept., Aalen University, Aalen, Germany  
[roy.oberhauser@hs-aalen.de](mailto:roy.oberhauser@hs-aalen.de)

**Abstract.** As software capabilities increase and are delivered more frequently, further external software components (e.g., services, frameworks, platforms, or libraries) are incorporated. Thus, organizations and IT increasingly depend on more complex and dynamic Software Supply Chains (SSCs). Yet, the entire set of components involved are often opaque for end users, businesses, and even developers. Software Bill of Materials (SBOM) formats (SPDX, CycloneDX) provide essential information regarding components, yet their voluminous text and 2D tool visualization limitations obscure the underlying models and SSC. This paper contributes an immersive Virtual Reality (VR) solution concept VR-SBOM towards holistic contextualized multi-layout visualization of SSCs with heterogeneous SBOMs. Our prototype implementation demonstrates its feasibility, while a scenarios-based case study exhibits its potential and scalability.

**Keywords:** Software Bill of Materials, SBOM, Software Supply Chain, Virtual Reality, Visualization, SPDX, CycloneDX.

## 1 Introduction

Modern software development is highly dependent on external components (e.g., libraries, packages, frameworks, Web APIs), yet often unmindful of its inclusion “under the hood.” A 2024 industry analysis of 20K+ enterprise applications found [1]: 180 component dependencies on average (10% having 400+), with modern commercial software consisting of up to 90% Open-Source Software (OSS) components; 6.6T+ downloads across 7M+ OSS projects/components involving 60M+ releases (averaging 16 per OSS project annually). As to deployment frequency due to Continuous Delivery (CD) automation, already in 2012 estimated average daily deployments for Google were 5.5K and Amazon 23K [2][3]. A 2021 survey of 1200 professionals revealed elite performers (26%) were deploying on demand multiple times a day [4]. High dynamicity with more external component dependencies results in larger, more complex, and changing Software Supply Chains (SSCs). While SSC Management (SSCM) aims to plan, monitor, control, optimize, and analyze SSCs, the essential characteristics inherent in software’s nature, namely complexity, conformity, changeability, and invisibility [5], can be transmuted to SSCs. These characteristics, in con-

junction with obscured external dependencies, make SSC transparency, comprehension, and analysis a challenge, and hence their management.

A Bill of Materials (BOM) defines the material components necessary to produce a product, and can be used to assess costs, risks, supply and support aspects, etc. The associated material distribution flows and multi-stage production dependencies can be viewed as a supply chain, with each stage processing materials (goods or services) to a consumer. For product-centric businesses, modeling and analysis of supply chains are inherent and vital to the business. Applying the BOM paradigm to software acquisition/development results in a Software BOM (SBOM) utilized by a SSC, which includes the components, libraries, tools, and activities needed to develop, build, procure, provision, and/or distribute some software artifact. As the need for SSC transparency became evident to address security vulnerabilities and license conformance, a push towards SBOM formats and their adoption by software suppliers became apparent. The System Package Data Exchange (SPDX) [6] (formerly Software Package Data Exchange) SBOM format by the Linux Foundation was released in 2011 and published in 2021 as ISO/IEC 5962 [7]. OWASP’s alternative SBOM format CycloneDX (CDX) [8] was published in 2018 and later as ECMA-424 [9]. As of 2023, GitHub offers an “Export SBOM” function, simplifying SBOM generation and increasing the likelihood of further SBOM adoption. Over 72K SBOM’s were published by the end of 2023 [1]. Hitherto, the lack of practical and accessible SBOM information and adoption had made SSC *modeling* impractical, a prerequisite for SSCM. Current SBOM/SSC tools lack comprehensive visualization, affecting model transparency, comprehension, and analysis. This, in turn, impacts the veracity and premises of software business models that count on coherent, correct, conformant, sufficient, and resilient SSCs - often only noticed after SSC disruptions occur. SSCM necessitates models offering transparency and insights that can address software’s complexity, conformity, changeability, invisibility, and external dependencies.

To address the comprehensive visualization of SBOM and SSC models, this paper proposes and investigates an immersive experience using Virtual Reality (VR). Our prior VR-related contributions in the Enterprise Architecture (EA) space include: VR-EA+TCK [10] supports EA models, integrating enterprise repositories, Atlas, IT blueprints, and knowledge and content management systems, with VR-EvoEA+BP [11] animating enterprise evolution and Business Processes (BPs). In the Software Engineering (SE) area: VR-ISA [12] enables informed software architecture, VR-SDLC [13] models development lifecycles, VR-GitCity [14] and VR-Git [15] model Git repos, VR-UML [16] for Unified Modeling Language models, and VR-DevOps [17] for CD pipeline models. This paper contributes VR-SBOM, a VR solution concept for context-enhanced multi-layout visualization of SBOMs and SSCs, supporting comprehensive visualization and inter-model SSC analysis. Our realization demonstrates its feasibility, while a scenario-based case study exhibits its potential and scalability. Furthering SSC transparency and comprehensibility enhances SSC management and optimization options.

The paper is structured as follows: Section 2 discusses related work; Section 3 presents our solution concept; Section 4 details our realization; our evaluation is described in Section 5, followed by a conclusion.

## 2 Related Work

Work related to the visualization of SBOMs includes Jones and Tate [18] for interactively comparing BOM graphs in 2D. DepVis [19] interactively visualizes third-party dependencies and vulnerabilities in 2D. V-Achilles [20] visualizes npm package vulnerabilities for GitHub in 2D. Google’s central repository Open Source Insights (OSI) [21] provides OSS package security information with 2D graph-based dependency visualization. OWASP’s Dependency Track [22] is an intelligent component analysis platform for CDX files, offering a 2D dashboard with monitoring and risk analysis. OWASP CycloneDX Sunshine [23] visualizes CycloneDX files in 2D. Regarding SSC visualization, Kula et al. [24] propose a generalized model for visualizing library popularity, adoption, and diffusion via a software universe graph, library coexistence pairing heat maps, and dependents diffusion plots. We are unaware of any tools that currently offer 3D or VR-based SBOM and SSC visualization.

Aside from visualization, work related to SBOM tooling includes Mirakhorli et al. [25], who conducted an extensive empirical analysis of 84 open-source and proprietary SBOM tools to assess the current landscape, highlighting many issues including interoperability, quality, and many having a niche focus and immaturity. Yousefnezhad and Costin [26] evaluated real-world SBOM tools with regard to DevSecOps, SSC, and compliance scenarios. Wang et al.’s book on SSCM [27] surveys research literature and describes SSC modeling, analysis, issues, and techniques, yet SSC visualization is not addressed.

Hence, further work is needed to investigate (immersive) SBOM/SSC visualization concepts to support various analysis, management, and collaboration scenarios.

## 3 Solution Concept

Our solution approach leverages VR for visualizing one or more SBOM models and relations simultaneously, mapped to a spatial structural model that can be immersively explored and experienced in 3D.

### 3.1 Grounding of our Solution Concept in VR-Related Research

To address possible reservations about the appropriateness of VR in our solution concept, our reasoning is based on prior VR research in areas we view as related to modeling, analysis, and collaboration, some of which we highlight here. In their systematic meta-analysis, Akpan & Shanker [28] showed VR and 3D offer significant advantages in the area of discrete event modeling, including model development, analysis, and Verification and Validation (V&V). Of 23 articles examining 3D analysis, 95% concluded using 3D was more potent and lead to better analysis than 2D, e.g., when evaluating a model’s behavior or performing a what-if analysis. They also found a consensus that 3D/VR can present results convincingly and understandably for decision-makers. 74% of 19 papers concluded that 3D/VR significantly improves the model development task (supporting teams and improving

precision and clarity). To investigate VR’s suitability for analytical tasks for an information architecture, Narasimha et al. [29] used a card sorting collaboration experiment. They found that VR was at least as good as in-person card sorting, and for certain variables VR was even better than both conventional and video-based conditions. Qualitative data evaluating awareness indicated that during collaborative interaction, participants were aware of their task, others, and their context, while collaborating similarly to an in-person setting. Additionally, the qualitative data showed evidence of positive views towards VR. The outcomes suggest that both a sense of presence and collaboration (equivalent to an in-person setting) is possible within VR. A survey of Immersive Analytics (IA) by Fonnnet & Prie [30] analyzed 177 papers. They found concurring evidence that for graph and spatial data analysis, IA provides benefits vs. non-IA when the scene complexity exceeds the 2D display, while for multi-dimensional data, the advantages are more task-dependent. They remark that while IA enables exploration of large-scale data worlds, context-aware navigation techniques are insufficiently exploited - although they are critical for users. We thus conclude that an immersive contextual VR experience has significant potential for comprehensively depicting large models in 3D while supporting awareness, modeling, analysis, V&V, decision support, stakeholder inclusion, and collaboration.

### 3.2 Relation to Our Prior VR-Related Research

Our solution map in Fig. 1 positions VR-SBOM relative to our other VR-based solutions. Besides our own prior work introduced earlier, our generalized VR Modeling Framework (VR-MF), described in [31], provides a domain-independent hypermodeling framework addressing key aspects for modeling in VR: visualization, navigation, interaction, and data. VR-EA [31] supports EA models in VR, including ArchiMate and BPMN via VR-BPMN [32]; VR-ProcessMine [33] supports process mining; VR-EAT [34] integrates and models enterprise repositories and the Atlas EA tool and blueprints. In the SE and Systems Engineering (SysE) area, there is VR-V&V [35] for V&V, VR-TestCoverage [36] for test coverage. As SBOMs/SSCs can be relevant to at least two perspectives, we position VR-SBOM as spanning two broad areas: EA & BP from the IT perspective; and SE & SysE from the software development perspective. Broad holistic solutions would be feasible in combination with our other solutions. None of our prior work has investigated the modeling of SBOM or SSCs in VR nor addressed their specific challenges.

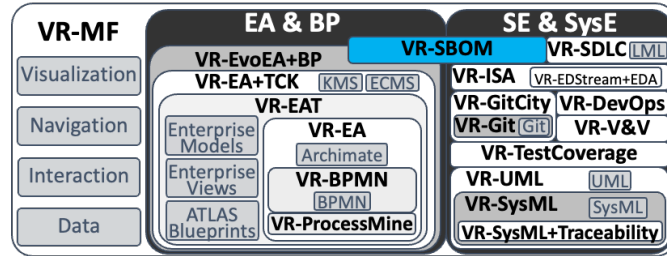


Fig. 1. VR solution concept map showing VR-SBOM (blue) in relation to our other concepts.

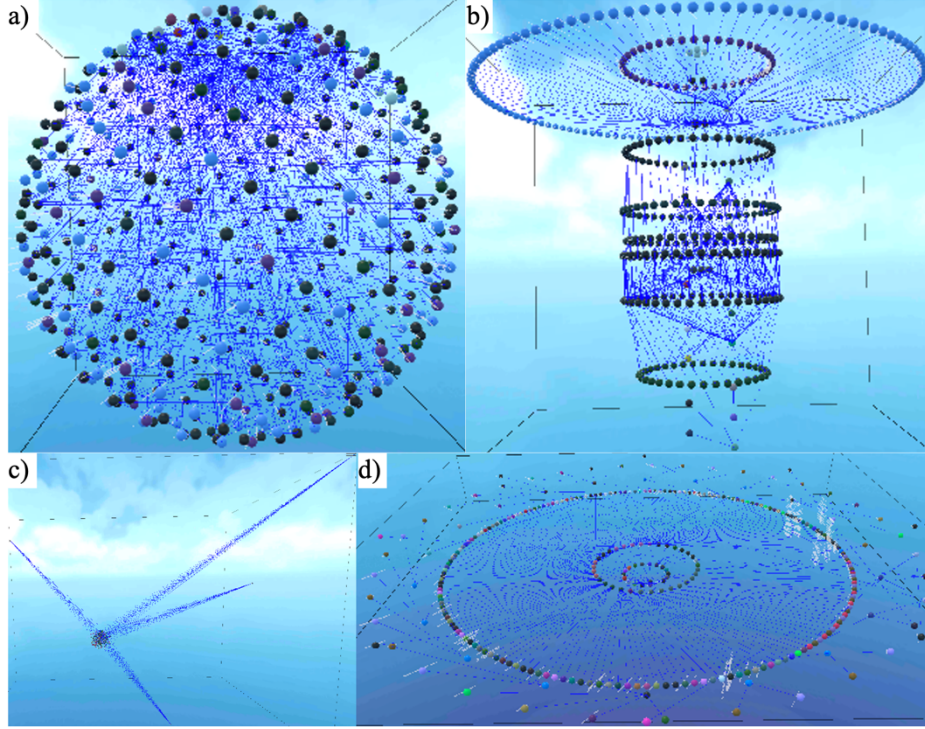
### 3.3 The VR-SBOM Solution Concept

The contribution of this paper is a VR solution concept for context-enhanced multi-layout visualization of SSCs and SBOMs. This solution concept is abstract, extensible, and independent of SBOM modeling format (SPDX, CDX). By annotating contextualized connected SBOM models, SSCs can be immersively visualized and experienced. Thus, the comprehension and analysis of SSCs becomes feasible, offering a larger picture of how various software artifacts relate to the SSC. Furthermore, accessibility, comprehension, and collaboration via concern-based (risk, resilience, security, etc.) perspectives for viewing and filtering a model for diverse stakeholders, such as software developers, enterprise architects, business analysts, quality assurance, IT administrators, security specialists, compliance auditors, or managers. These objectives are addressed as follows:

**Data.** A Data Hub supports flexible data integration via ports and adapters in combination with a data repository. SBOMs in both formats, CVEs, and any other SSC contextual data is integrated and stored in our JSON format. Alternatively, if data freshness is paramount, data can forego storage and be integrated live via Web APIs.

**Visualization.** A Directed Acyclic Graph (DAG) visualization paradigm was chosen to generically and scalably visualize extensive SBOM and SSC models. SBOM elements are represented by spherical nodes, while relations (edges) are depicted as lines. Element types are differentiated by customizable node colors displayed via a legend. Node labels provide naming information, while detailed element metadata can be retrieved via our VR-Tablet concept. Labeled glass boxes enclose models to support model orientation, differentiation, and contextualization. Multiple DAG layout options are offered: 1) a spatially-dense 3D *Sphere* (or *nexus*) (Fig. 2a) with node placement on the sphere’s surface and all relations within, for comprehensive overview while minimizing spatial distances; 2) *Category-and-Level* (Fig. 2b) bundles nodes by type (category) on cylindrical layer surface (levels) positioned on vertical axis near layers having most relations with its type, for determining (un-)common types and relations; 3) *Force-Directed Graph* (Fig. 2c) positions nodes using attractive and repulsive forces to approximately equalize edge lengths while minimizing collisions, for ascertaining highly (dis-)connected nodes; 4) *Radial Tidy Tree* (Fig. 2d) orders nodes hierarchically along a vertical radial tree (cone-like) with ever larger lower rings, for navigating granularity and depth; 5) *Stacking Radial Tree* (shown later), like Radial Tidy Tree but stacking multiple rings within a category level, for reducing ring circumference, and 6) *Custom placement*. Thus, hitherto intangible SBOM elements and relations are flexibly visualized and graph-based visual comparisons support stakeholder concerns.

**Navigation.** To reduce the potential for VR sickness symptoms during immersive navigation, our solution concept supports two navigation modes: 1) locomotion gliding controls (default), enabling users to fly through the VR space and get an overview of the entire model from any angle they wish, or 2) teleporting permits a user to select a destination and be instantly placed there (i.e., moving the camera to that position), reducing movement through a virtual space.

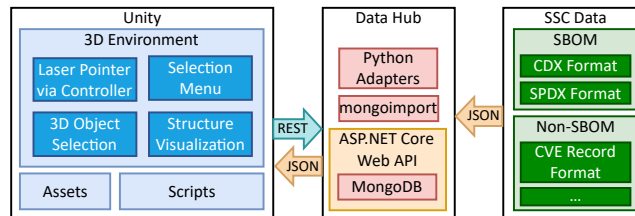


**Fig. 2.** Visualization layouts for an SBOM containing 448 nodes: a) Sphere/nexus, b) Category-and-Level, c) Force-directed, and d) Radial Tidy Tree.

**Interaction.** The VR controllers and our VR-Tablet concept support interaction. Our VR-Tablet paradigm provides: interaction support, detailed information regarding a selected element, or browsing, filtering, searching, and settings. Any browser-based (multimedia) content could also be displayed as shown in our prior work [10].

## 4 Realization

To determine the feasibility of our solution concept, we realized a prototype having the logical architecture in Fig. 3 and described below.



**Fig. 3.** VR-SBOM Logical Architecture.

**Data integration.** The Data Hub integrates and stores data in a NoSQL document-oriented local database MongoDB in JSON and offers endpoints (ports) to the VR frontend via the ASP.NET Core Web API. Adapters are used to convert SSC-relevant data in various SBOM formats such as SPDX, CDX, and non-SBOM data such as CVE data in the CVE Record Format [37]. MongoDB was used as a local database consisting of two collections, one for SBOM-specific data (SPDX, CDX, etc.) and another for non-SBOM (e.g., additional SSC) data (e.g., CVE records).

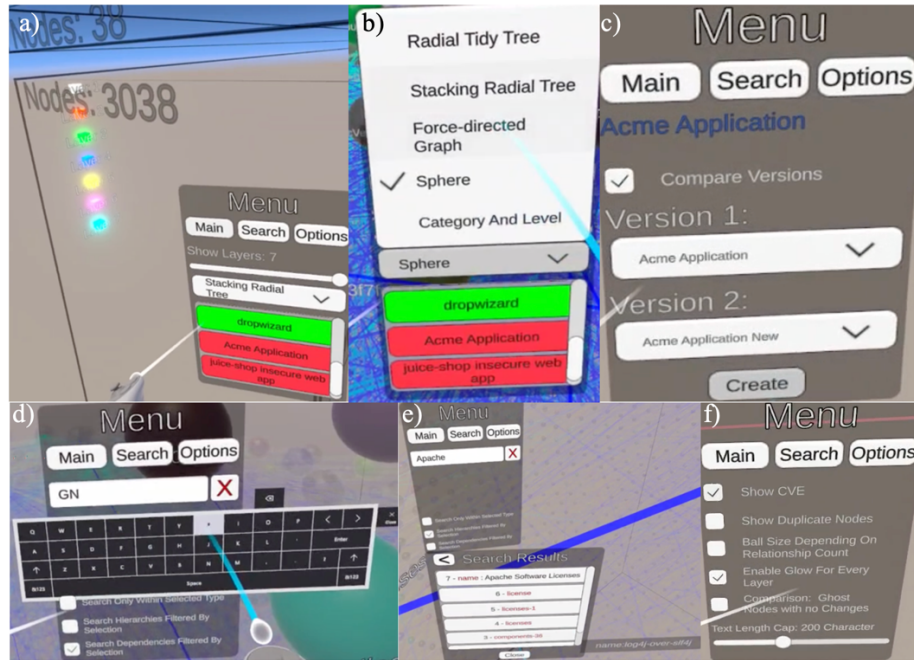
**Visualization.** VR support was realized in C# with Unity 2022.3.21f1, accessing the Data Hub via REST (REpresentational State Transfer) and retrieving JSON data. Nodes are depicted as spheres, key-value pairs as text or as a popout tooltip, and references as lines. For directional lines, the darker (closer) end (blue/red) is the enclosing source and the lighter (aqua/pink) end the point referred to, reducing the clutter arrowheads would create while providing direction. Multiple SBOM graphs can be depicted concurrently and are labeled and contextually distinguished via boundary (transparent glass) boxes, which are labeled on the bottom by SBOM name/ID and indicate node metrics in the upper left corner (Fig. 4a). The graph layouts (Sphere, Category-and-Level, Radial Tidy Tree, Force-directed Graph, and Stacking Radial Tree) can be switched to support a different focus (Fig. 4b). For the *Sphere* (nexus) layout, all nodes are placed on the sphere’s surface equidistant from each other, while all lines (connections) are inside the sphere. For *Category-and-Level*, the nodes are first segregated by category (type) and then positioned along a vertical line based on its level. For the *Radial Tidy Tree* layout, a vertical hierarchy is used with each lower-layer ring placed even further out in the x/z direction (even if the level is sparse), causing it to typically be wider than higher. In the *Force-directed Graph* layout, connected nodes receive an attractive force, while unconnected nodes receive a repelling force; the vectors are then combined to determine a node’s position, causing the most highly-connected nodes to be more centric. *Stacking Radial Tree* reduces ring radius by stacking multiple rings at the same level. The implementation can be readily extended to support additional graph visualization types and store *Custom placement*. A legend of the node types and their randomly-assigned colors is placed on top of (or in proximity of) the boundary box. To reduce visual clutter, by default when nodes have the same property values, they are merged, but this can be toggled. Storing custom layout adjustments was not yet implemented due to time constraints.

**Navigation.** Both locomotion and teleporting are supported, including teleporting to a search result.

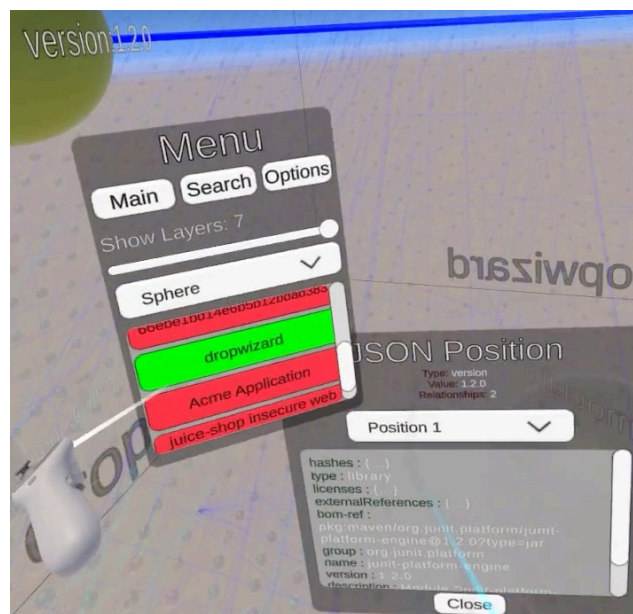
**Interaction.** Interaction is supported with our VR-Tablet. To prevent the VR-Tablet from interfering with comprehension or navigation, it is hidden and appears when the left controller is rotated outwards by about 90 degrees. It offers a menu consisting of three tabs: Main, Search and Options, as shown in Fig. 4:

- a) Main offers: a slider “Show Layers” for adjusting the maximum graph layer depth, a dropdown for desired layout type, and a scroll view of SBOM names or object IDs. In Scroll View, the selection acts as a toggle: if the SBOM is already depicted (green), then it is hidden (red), otherwise it is loaded.
- b) The dropdown Graph Types offers various graph layouts (Sphere, Category-and-Level, Radial Tidy Tree, Force-Directed Graph, and Stacking Radial Tree),





**Fig. 4.** VR-Tablet menu showing tab sections: a) Main, b) Layout options, c) Compare Versions submenu, d) Search/Filter, e) Search Results, and f) Options.



**Fig. 5.** Selecting top left green node opens JSON position panel (bottom right) in VR-Tablet.



- c) “Compare Versions” submenu is offered when two graphs are selected; if checked, the SBOM node differences are highlighted as a colored ring, with green/red indicating new/missing node (can ghost rest).
- d) Search offers both searching and filtering based on an input field that offers a pop-up prefab platform-independent MRTK keyboard to enter a search string. The search includes both types and values and ghosts (makes transparent) all elements not in the result set. Different search and filter types are supported: “Search Only Within Selected Type” searches nodes within type, “Search Hierarchies Filtered by Selection” to search nodes of a subgraph, and “Search Dependencies Filtered by Selection” to search dependencies of a subgraph.
- e) Search results are displayed in a separate extra VR-Tablet pane with scrollable search results, which may be numbered based on the hierarchy level (1 = top-level). Selecting a result offers a Tooltip to the right of the selection indicating node name, etc. Teleporting to any search result is supported.
- f) Options offers these additional settings:
  - “Show CVE” depicts CVE data related to a loaded SBOM as separate graphs in bounded glass boxes, with a red connection to its location in the SBOM graph.
  - “Show duplicate Nodes” will depict all nodes separately, since by default all nodes with the same property values are depicted by a single node.
  - “Ball Size Depending On Relation Count” will cause the node size to be larger when it has more relations relative to other nodes.
  - “Enable Glow For Every Layer”
  - “Comparison: Ghost Nodes with no Changes” causes unchanged nodes to be ghosted (transparent) to reduce visual clutter during comparison analysis.
  - “Text Length”: constrains the allowable max text length of labels.

Selecting a node will keep its node type opaque, while ghosting all other nodes of different types and lines unassociated with a node of that type. Node-specific JSON information is displayed on an additional freely movable pane, offering a dropdown list of all positions of the node in the SBOM file, as shown in Fig. 5.

The evaluation then demonstrates how these realized capabilities are utilized.

## 5 Evaluation

For the evaluation of our solution concept with our prototype realization, we refer to the design science method and principles [38], in particular, a viable artifact, problem relevance, and design evaluation (utility, quality, efficacy). For this, a scenario-based case study focuses on supporting SBOM and SSC comprehension, analysis, and contextualization, the scenarios being: Heterogeneous Multi-SBOM Interoperability, Comparison and Dependency Analysis, License (search/filtering) Analysis, Security and SSC Analysis, and Scalability. Further SBOM use cases can be readily mapped to these scenarios (provenance, foreign ownership, outdated components, etc.). Abbreviations referred to for the SBOM files utilized in the evaluation scenarios are given in Table 1. Further SBOMs were also tested but are not listed.

**Table 1.** SPDX and CDX files used in evaluation.

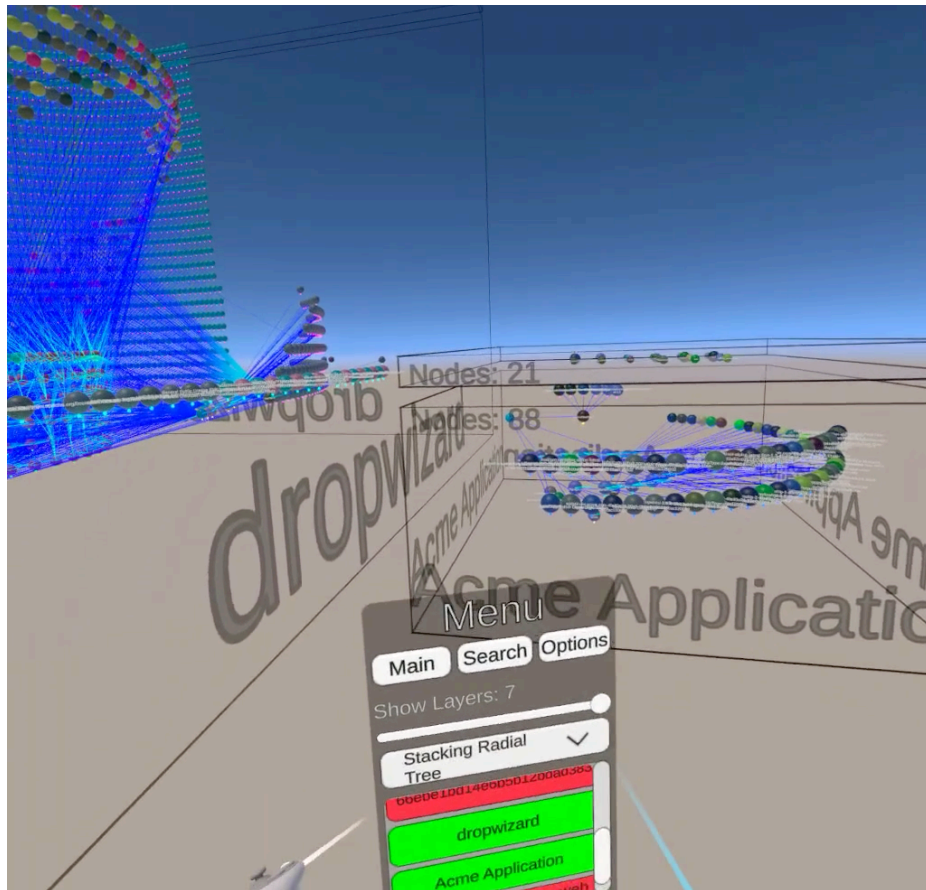
Name	Abbrev.	Nodes (w/o duplicates)	Types	Lines	Format	Modifications
Dropwizard <sup>1</sup>	DW	(3038)	38	10621	CDX 2.0.2	3 CVEs linked
Acme Application <sup>2</sup>	AA	129(88)	21	154	SPDX 3.0	-
Acme Application New <sup>2</sup>	AAN	132(88)	35		SPDX 3.0	+/- 3 nodes

<sup>1</sup> <https://github.com/CycloneDX/bom-examples/blob/master/SBOM/dropwizard-1.3.15/bom.json>

<sup>2</sup> <https://github.com/spdx/spdx-examples/blob/master/software/example13/spdx3.0/example13.spdx3.json>

### 5.1 Multiple Heterogeneous SBOM Interoperability Scenario

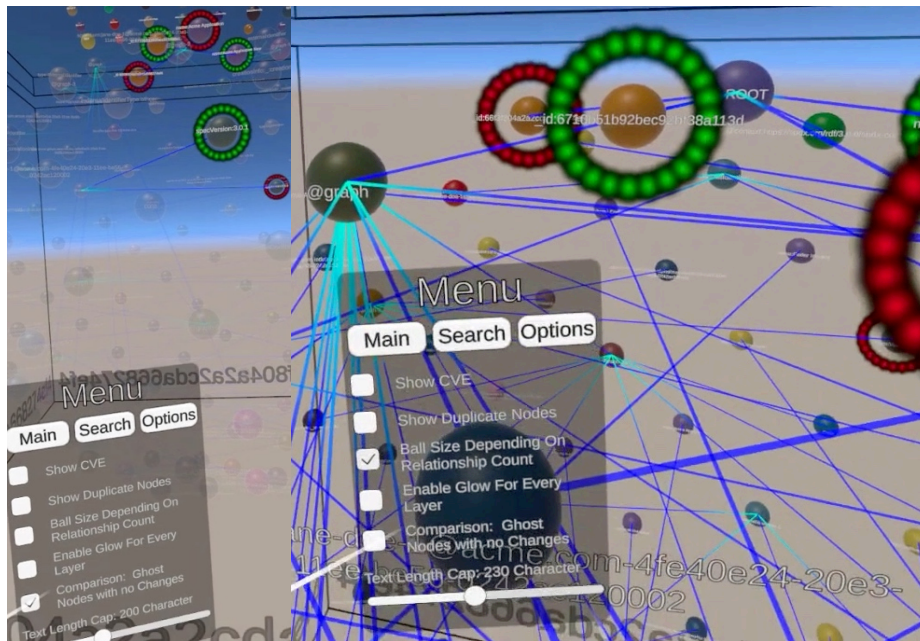
To be practical, SSCs must be able to view multiple SBOMs having differing formats simultaneously. For this scenario, SBOM interoperability support is demonstrated by depicting multiple heterogeneous SBOM models, with DW based on a CDX model in the left boundary box and AA based on an SPDX model on the right in Fig. 6.



**Fig. 6.** Multiple heterogeneous SBOMs loaded (green) (DW CDX left, AA SPDX right).

## 5.2 Comparison and Dependency Analysis Scenario

This scenario demonstrates comparison analysis support via a visual delta of graphs, in particular selecting SBOM versions in VR-Tablet (Fig. 4c). As shown in Fig. 7, green rings highlight additions, red rings deletions (modifications use both). Dependency analysis is supported via node size based on relation count (Fig. 7 right) - relevant, e.g., if high relation risk is a concern. Dependency navigation is shown later.



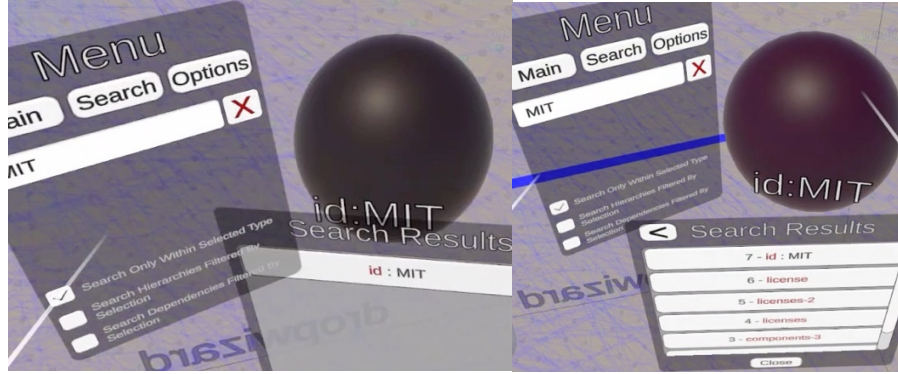
**Fig. 7.** AA SBOM version comparison (left) highlighting additions/deletions (green/red ring) and ghosting “Nodes with no Changes”; on right, “Ball Size Depending on Relation Count”.

## 5.3 License (Search/Filtering) Analysis Scenario

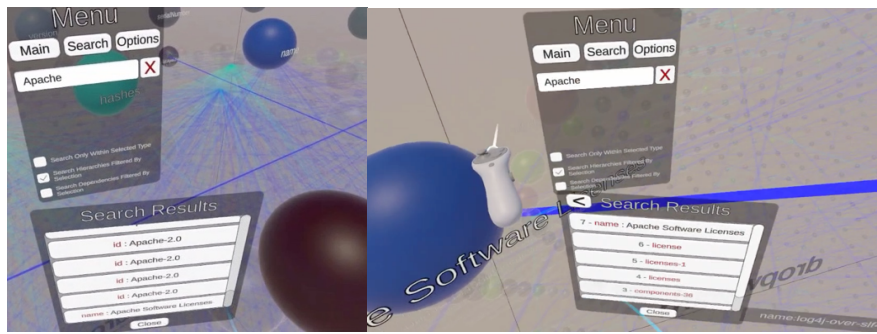
A license analysis scenario is a typical SBOM use case. We use search and filtering capabilities to demonstrate how this scenario is supported, yet these capabilities can support further scenarios and stakeholder concerns (security, risk, compliance, etc.).

**Search:** A basic search finding all occurrences of a given string across all nodes.

**Filtered by Type:** Searching only within a certain property type, e.g., name, description, id. As shown in Fig. 8, selecting the node type “id”, enabling “Search Only Within Selected Type”, and providing with search string “MIT” shows a results panel listing any nodes of type “id” containing the string MIT (ghosting others). Selecting a specific search result teleports to that node, with the panel then showing that node’s hierarchical path to the root node grasp its context. Hovering over any hierarchy element shows a tooltip to the right providing that node’s name. Selecting a path element teleports to it. Via the top panel arrow, one can return to the previous search results to pick another.

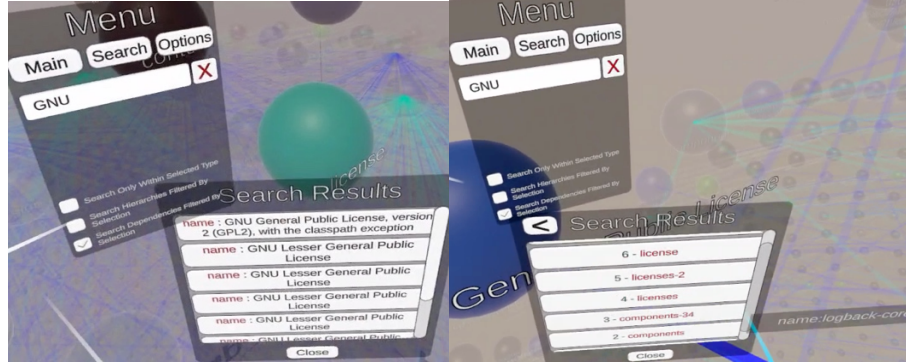


**Fig. 8.** Left: selecting type “id”, with “Search Only Within Selected Type” for “MIT” searches all nodes of that type containing that string, providing a “Search Results” panel; selecting any search result then teleports to it, which then shows its contextual path to its root node (right).



**Fig. 9.** Left: selecting node “licenses” then “Search Hierarchies Filtered by Selection” for “Apache” gives search results only within the license subhierarchy (independent of type); selecting a search result (e.g., name:) teleports to it (right), showing its contextual path to its root node; hovering on result shows tooltip with the component’s name (here log4j-over-slf4j).

**Filtered Node Hierarchy.** This type of search filter can be useful, e.g., when concerned only about a selected component or some subgraph hierarchy. Conceivable search examples include determining if some database is the open source or enterprise version, or the version of some library within a software component. For our license analysis case example in Fig. 9, “Search Hierarchies Filtered by Selection” was enabled and the “license” node selected (ghosting all nodes outside hierarchy). Then the search string “Apache” is entered via the virtual keyboard, which shows the search results panel listing all matching hierarchically deeper nodes in the SBOM (ignoring any higher up or parallel hierarchies or relations). While we selected the license hierarchy here, we could have selected any component hierarchy. Selecting a search result teleports to that node, whereafter the panel shows the complete path to the root node (with tooltip support), to help contextualize it.



**Fig. 10.** Left: selecting node “license,” then “Search Dependencies Filtered by Selection” for “GNU” gives search results for nodes *directly related* to “license” and containing “GNU”; selecting any search result teleports to it (right) then showing contextual path to root node; hovering on result shows tooltip that provides node name (here component name logback-core).

**Filtered Dependencies:** Via this search type, directly related dependencies are searched/filtered. This can find any nodes across all components as long as they directly relate to a node of interest. For our license scenario shown in Fig. 10, “Search Hierarchies Filtered by Selection” was enabled, the “license” node selected, and the search string “GNU” given via the virtual keyboard. The search results panel shows any nodes *directly related* to the node (license) containing the string “GNU” independent of its hierarchical position. Hence, no GNU components are shown that don’t relate to license. Selecting any search result teleports to that node, whereafter the panel shows the complete contextual path from that node to the root node. This context hierarchy can then be explored by hovering over any result, causing a tooltip to pop out on the right that provides its element name, in our example “logback-core”. Selecting an element causes one to teleport to it. The top panel arrow returns to the previous search results.

#### 5.4 Security Analysis and SSC Analysis Scenario

SSCs necessitate chaining (linking) elements and incorporating supplementary data. While SBOMs are standardized, SSC models as such are not, yet SBOMs foresee and provide link support (e.g., via references). Both CDX (e.g., via BOM-Link, externalReferences) and SPDX (e.g., via ExternalRef) support flexible intra- or inter-linking of various type-specific data (SBOMs, models, Web APIs, documentation, etc.). Due to space constraints, as the security analysis scenario involves linking, it serves to demonstrate support for both analysis cases; linking non-SBOM CVE vulnerability models can represent linking to any extrinsic data/models for an SSC. In this case, the original DW SBOM was modified to fictionally link three extrinsic CVE JSON records from CVE List Downloads [39], consisting of two general ones (CVE-2020-11002 and CVE-2020-5245) and one specific to the Log4j component (CVE-2021-44228), as shown in Fig. 11. Each CVE record model was placed in its own



boundary box linked via red lines to applicable SBOMs (Fig. 12, Fig. 13 left). Selecting any “CVE” search result shows its context path (Fig. 13 right), which can be used for teleporting to get more detailed (CVE) data. The CVE references demonstrate how other references to data/models could be included to support SSCs and their analysis.

```

2178 |
2179 |   "type": "library",
2180 |   "bom-ref": "pkg:maven/org.slf4j/log4j-over-slf4j@1.7.26?type=jar",
2181 |   "publisher": "QOS.ch",
2182 |   "group": "org.slf4j",
2183 |   "name": "log4j-over-slf4j",
2184 |   "version": "1.7.26",
2185 |   "description": "Log4j implemented over SLF4J",
2186 |   "hashes": [
2187 |     {
2188 |       "alg": "SHA-256",
2189 |       "content": "8a120413111126d081c610c1d180c838ba7a61223b96aa11c4203894afbc22e1"
2190 |     }
2191 |   ],
2192 |   "licenses": [
2193 |     {
2194 |       "name": "Apache Software Licenses",
2195 |       "url": "http://www.apache.org/licenses/LICENSE-2.0.txt"
2196 |     }
2197 |   ],
2198 |   "externalReferences": [
2199 |     {
2200 |       "type": "website",
2201 |       "url": "http://www.qos.ch"
2202 |     },
2203 |     {
2204 |       "type": "distribution",
2205 |       "url": "https://oss.sonatype.org/service/local/staging/deploy/maven2/"
2206 |     },
2207 |     {
2208 |       "type": "vcs",
2209 |       "url": "https://github.com/qos-ch/slf4j"
2210 |     }
2211 |   ],
2212 |   "vulnerabilities": [
2213 |     "CVE-2021-44228"
2214 |   ]
2215 | }

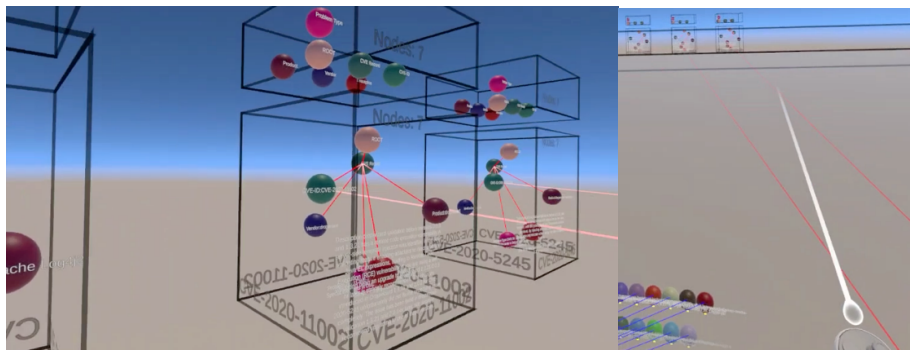
```

```

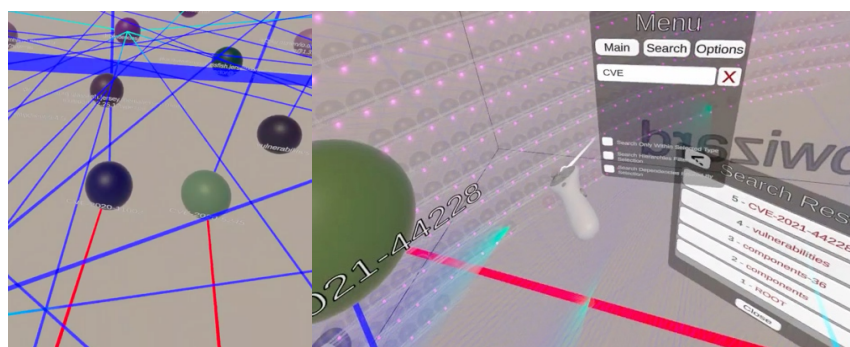
1 {
2   "bomFormat": "CycloneDX",
3   "specVersion": "1.2",
4   "serialNumber": "urn:uuid:b4
5   "version": 1,
6   "metadata": { ...
7     "timestamp": "2021-11-02T10:00:00Z",
8     "tools": [
9       {
10        "name": "cyclonedx",
11        "version": "1.10.0",
12        "vendor": "CycloneDX"
13      }
14    ],
15     "component": {
16       "name": "dropwizard",
17       "version": "2.1.10",
18       "vendor": "Dropwizard",
19       "purl": "pkg:maven/io.dropwizard/dropwizard@2.1.10?type=jar"
20     },
21     "dependencies": [
22       {
23         "ref": "dropwizard",
24         "name": "dropwizard",
25         "version": "2.1.10",
26         "vendor": "Dropwizard",
27         "purl": "pkg:maven/io.dropwizard/dropwizard@2.1.10?type=jar"
28       },
29       {
30         "ref": "log4j-over-slf4j",
31         "name": "log4j-over-slf4j",
32         "version": "1.7.26",
33         "vendor": "QOS.ch",
34         "purl": "pkg:maven/org.slf4j/log4j-over-slf4j@1.7.26?type=jar"
35       },
36       {
37         "ref": "slf4j-api",
38         "name": "slf4j-api",
39         "version": "1.7.26",
40         "vendor": "QOS.ch",
41         "purl": "pkg:maven/org.slf4j/slf4j-api@1.7.26?type=jar"
42       }
43     ]
44   }
45 }

```

**Fig. 11.** SBOM modifications to DW CDX linking CVE vulnerabilities: CVE-2021-44228 specific to component library log4j and two additional CVEs appended to entire library bundle.



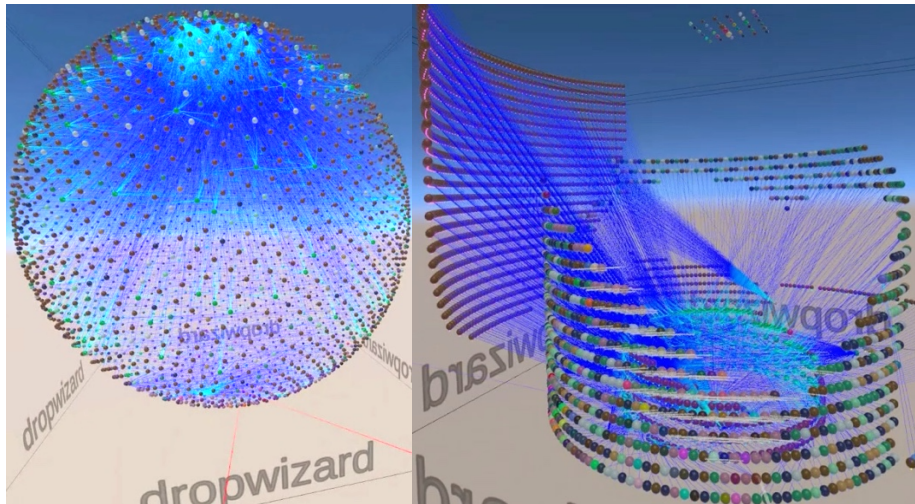
**Fig. 12.** Left: three juxtaposed CVEs boxes; Right: inter-relations (red) with SBOM shown.



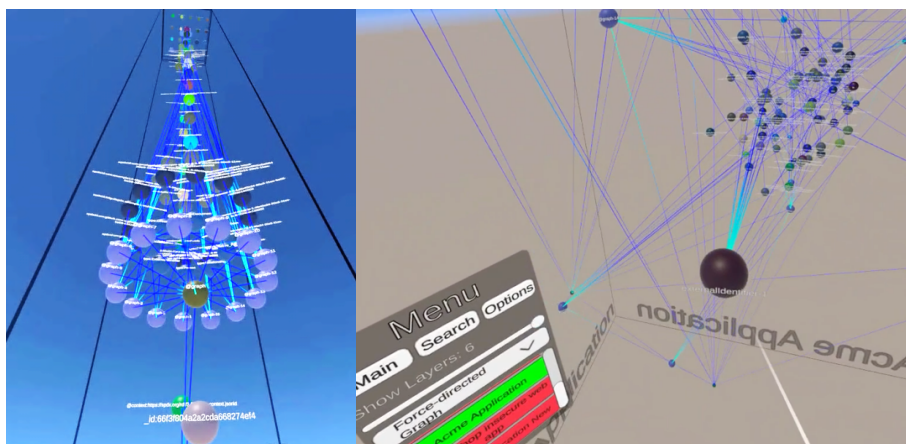
**Fig. 13.** CVEs-SBOMs inter-linked via red lines (left); CVE search result context (right).

### 5.5 Scalability Scenario

To evaluate visualization scalability, we depict SBOMs of varying sizes and layouts, each supporting a different stakeholder focus. The DW SBOM has 3038 non-duplicated nodes, 38 types, and 7 layers of depth. It is displayed as a Sphere (compact nexus) and Stacking Radial Tree (hierarchically-stacked) layouts in Fig. 14. The smaller AA SBOM is shown with Category-and-Level (type-bundled) and Force-directed Graph (dependency-centric) layouts in Fig. 15. To support comprehension at scale, the Limit Layers slider reduces unwanted layer depth, while layer glows (cubes) support layer orientation when navigating large models (especially when ghosting), as shown with AA SBOM in Fig. 16. Adjusting the Text Length slider would reduce text label clutter.

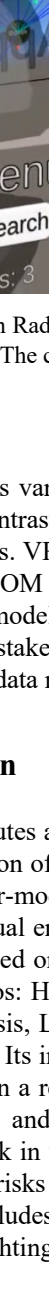


**Fig. 14.** Large DW SBOM as a Sphere (left) and Stacking Radial Tree (right).



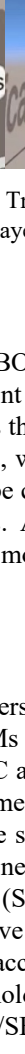
**Fig. 15.** AA SBOM as Category-and-Level (left) and Force-directed Graph (right).





s var  
 ntras  
 s. VI  
 OM  
 node  
 stake  
 data n  
 n  
 tes a  
 on of  
 r-mo  
 al ex  
 ed o  
 s: H  
 sis, I  
 Its i  
 n a r  
 and  
 k in  
 risks  
 ludes  
 hting

pti



with

## References

1. Sonatype: State of the Software Supply Chain. (2024). [https://sonatype.com/hubfs/SSCR-2024/SSCR\\_2024-FINAL-optimized.pdf](https://sonatype.com/hubfs/SSCR-2024/SSCR_2024-FINAL-optimized.pdf), last accessed 2025/05/09
2. IT Revolution: DevOps Guide: Selected Resources to Start Your Journey. (2015). [https://web.archive.org/web/20211010072856/http://images.itrevolution.com/documents/ITRev\\_DevOps\\_Guide\\_5\\_2015.pdf](https://web.archive.org/web/20211010072856/http://images.itrevolution.com/documents/ITRev_DevOps_Guide_5_2015.pdf)
3. Micco, J.: Tools for continuous integration at google scale. Google Tech Talk (2012)
4. DORA Team: Accelerate State of DevOps report (2021). <https://web.archive.org/web/20250402081619/https://services.google.com/fh/files/misc/state-of-devops-2021.pdf>
5. Brooks, F. P. Jr.: The Mythical Man-Month. Addison-Wesley (1995)
6. System Package Data Exchange (SPDX®), <https://spdx.dev>
7. ISO/IEC 5962:2021 Information technology — SPDX® Specification V2.2.1 (2021)
8. CycloneDX, <https://cyclonedx.org>
9. ECMA-424 CycloneDX Bill of materials specification (2024)
10. Oberhauser, R., Baehre, M., Sousa, P.: VR-EA+TCK: Visualizing Enterprise Architecture, Content, and Knowledge in Virtual Reality. In: Shishkov, B. (eds) Business Modeling and Software Design (BMSD 2022), pp. 122-140. LNBIP, vol 453. Springer, Cham (2022). [https://doi.org/10.1007/978-3-031-11510-3\\_8](https://doi.org/10.1007/978-3-031-11510-3_8)
11. Oberhauser, R., Baehre, M., Sousa, P.: VR-EvoEA+BP: Using Virtual Reality to Visualize Enterprise Context Dynamics Related to Enterprise Evolution and Business Processes. In: Shishkov, B. (eds) Business Modeling and Software Design. BMSD 2023. LNBIP, vol 483. Springer, Cham (2023). [https://doi.org/10.1007/978-3-031-36757-1\\_7](https://doi.org/10.1007/978-3-031-36757-1_7)
12. Oberhauser, R.: VR-ISA: Immersively Visualizing Informed Software Architectures Using Viewpoints Based on Virtual Reality. International Journal on Advances in Software, 17(3 & 4), pp. 282-300 (2024)
13. Oberhauser, R.: VR-SDLC: A Context-Enhanced Life Cycle Visualization of Software-or-Systems Development in Virtual Reality. In: Shishkov, B. (eds) Business Modeling and Software Design. BMSD 2024. LNBIP, vol 523. Springer, Cham (2024). [https://doi.org/10.1007/978-3-031-64073-5\\_8](https://doi.org/10.1007/978-3-031-64073-5_8)
14. Oberhauser, R.: VR-GitCity: Immersively Visualizing Git Repository Evolution Using a City Metaphor in Virtual Reality. Int'l Journal Adv. Software, 16(3&4), 141-150 (2023).
15. Oberhauser, R.: VR-Git: Git Repository Visualization and Immersion in Virtual Reality. In: Proceedings of the Seventeenth International Conference on Software Engineering Advances (ICSEA 2022), pp. 9-14. IARIA (2022)
16. Oberhauser, R.: VR-UML: The Unified Modeling Language in Virtual Reality – An Immersive Modeling Experience. In: Shishkov, B. (eds) Business Modeling and Software Design. BMSD 2021. LNBIP, vol 422. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-79976-2\\_3](https://doi.org/10.1007/978-3-030-79976-2_3)
17. Oberhauser, R.: VR-DevOps: Visualizing and Interacting with DevOps Pipelines in Virtual Reality. In: Proceedings of the Nineteenth International Conference on Software Engineering Advances, pp. 43-48 (2024).
18. Jones, R., Tate, T.: Visualizing Comparisons of Bill of Materials. In: 2023 IEEE Symposium on Visualization for Cyber Security (VizSec), pp. 12-16. IEEE (2023)
19. Groman, M.: Visualization of Vulnerabilities in Open Source Software Dependencies. Master Thesis, Masaryk University (2023)
20. Jarukitpipat, V., et al.: V-Achilles: An Interactive Visualization of Transitive Security Vulnerabilities. In: Proc. 37th IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE '22). ACM, Article 169, pp. 1–4 (2022). <https://doi.org/10.1145/3551349.3559526>

21. Open Source Insights, <https://deps.dev/>, last accessed 2025/05/09
22. Dependency-Track, <https://dependencytrack.org>, last accessed 2025/05/09
23. Sunshine, <https://github.com/CycloneDX/Sunshine/>, last accessed 2025/05/09
24. Kula, R. G., De Roover, C., German, D. M., Ishio, T., Inoue, K.: A generalized model for visualizing library popularity, adoption, and diffusion within a software ecosystem," 2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER), Campobasso, Italy, 2018, pp. 288-299, doi: 10.1109/SANER.2018.8330217
25. Mirakhorli, M., et al.: A Landscape Study of Open Source and Proprietary Tools for Software Bill of Materials (SBOM). *arXiv preprint arXiv:2402.11151* (2024)
26. Yousefnezhad, N., Costin, A.: Understanding SBOMs in Real-World Systems – A Practical DevOps/SecOps Perspective. In: Shishkov, B. (eds) Business Modeling and Software Design. BMSD 2024. LNBIP, vol 523. Springer, Cham (2024)
27. Wang, Y., Cheung, S. C., Yu, H., Zhu, Z.: Managing Software Supply Chains: Theory and Practice. Springer Nature, Singapore (2024) <https://doi.org/10.1007/978-981-96-1797-5>
28. Akpan, I. J., Shanker, M.: The confirmed realities and myths about the benefits and costs of 3D visualization and virtual reality in discrete event modeling and simulation: A descriptive meta-analysis of evidence from research and practice. *Computers & Industrial Engineering*, 112, 197-211 (2017)
29. Narasimha, S., Dixon, E., Bertrand, J. W., Madathil, K. C.: An empirical study to investigate the efficacy of collaborative immersive virtual reality systems for designing information architecture of software systems. *Applied ergonomics*, 80, 175-186 (2019)
30. Fonnet, A., Prie, Y.: Survey of immersive analytics. *IEEE transactions on visualization and computer graphics*, 27(3), 2101-2122 (2019)
31. Oberhauser, R., Pogolski, C.: VR-EA: Virtual Reality Visualization of Enterprise Architecture Models with ArchiMate and BPMN. In: Shishkov, B. (eds) Business Modeling and Software Design. BMSD 2019. LNBIP, vol 356. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-24854-3\\_11](https://doi.org/10.1007/978-3-030-24854-3_11)
32. Oberhauser, R., Pogolski, C., Matic, A.: VR-BPMN: Visualizing BPMN Models in Virtual Reality. In: Shishkov, B. (eds) Business Modeling and Software Design. BMSD 2018. LNBIP, vol 319. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-94214-8\\_6](https://doi.org/10.1007/978-3-319-94214-8_6)
33. Oberhauser, R.: VR-ProcessMine: Immersive Process Mining Visualization and Analysis in Virtual Reality. In: The Fourteenth International Conference on Information, Process, and Knowledge Management (eKNOW 2022), pp. 29-36. IARIA (2022)
34. Oberhauser, R., Sousa, P., Michel, F.: VR-EAT: Visualization of Enterprise Architecture Tool Diagrams in Virtual Reality. In: Shishkov, B. (eds) Business Modeling and Software Design. BMSD 2020. LNBIP, vol 391. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-52306-0\\_14](https://doi.org/10.1007/978-3-030-52306-0_14)
35. Oberhauser, R.: VR-V&V: Immersive Verification and Validation Support for Traceability Exemplified with ReqIF, ArchiMate, and Test Coverage. *International Journal on Advances in Systems and Measurements*, 16(3 & 4), 103-115 (2023)
36. Oberhauser, R.: VR-TestCoverage: Test Coverage Visualization and Immersion in Virtual Reality. In Proceedings of The Fourteenth International Conference on Advances in System Testing and Validation Lifecycle (VALID 2022), pp. 1-6. IARIA (2022)
37. cve-schema, <https://cveproject.github.io/cve-schema/schema/>, last accessed 2025/04/29
38. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. *MIS Quarterly*, 28(1), pp. 75-105 (2004)
39. CVE List Downloads, <https://www.cve.org/Downloads>, last accessed 2025/04/29