Journal of Systemics, Cybernetics and Informatics (2025) 23(5), 1-4
https://doi.org/10.54808/JSCI.23.05.1

A Critical Retrospect of OSS License Compliance: Lessons Learned and Next Steps

Sergius DYCK, Daniel HAFERKORN
Fraunhofer IOSB
Karlsruhe, Germany

ABSTRACT

In the rapidly evolving software development landscape, the integration of Open Source Software (OSS) has become commonplace, providing developers with extensive libraries and tools that enhance productivity and accelerate project timelines. However, the use of OSS comes with significant legal responsibilities, particularly regarding compliance with various Open Source Software Licenses (OSSL). An initial framework was designed to ensure OSS compliance, centering on automated creation of Software Bill of Materials (SBOMs) and a "License Playbook". Automated checks were executed with tools such as Maven and Nexus, verifying license acceptability and required source-code inclusion.

In follow-up work, OSS notice lists were automated, domain-driven design was applied to improve communication, and Java-based tools for Maven were introduced to structure compliance data and reduce errors.

Over time, it became clear that the original framework no longer aligns with evolving requirements, especially as various web projects with focus on OSSL gained in importance. The existing license-management tool encounters challenges in handling large dependency sets, and post-release adjustments in Maven repositories remain difficult to perform. Consequently, alternative software suites are being evaluated to determine whether the proprietary tool should be adapted or replaced to meet evolving needs and strengthen the overall OSS compliance strategy.

Keywords: Open Source Software, Open Source Compliance, Software Bill of Materials, Critical Retrospect, Lessons Learned.

1. INTRODUCTION

Open Source Software (OSS) integration has become standard practice in modern software development, offering extensive libraries and tools to improve productivity and accelerate project timelines. However, the use of OSS imposes significant legal responsibilities through various Open Source Software Licenses (OSSL).

1.1 Current state

In earlier work, a theoretical framework was introduced as a first step [1] to ensure basic OSS compliance. That framework emphasized structured processes for addressing OSS usage and license terms, supplemented by a "License Playbook" that provided concise reference guidelines.

Building on these foundational elements, practical solutions for automating compliance processes were then explored [2]. A Domain-Driven Design (DDD) approach was applied to establish a shared language that defined concepts such as "Third Party Library," "License Type," and "License Information." That language was realized through a domain model, which guided the structured storage of metadata about each license (including required disclaimers or acknowledgments). A Java-based tool was developed for Maven, namely the "osslist-maven-plugin (OMP)". This tool enables dependency information to be gathered, validated, and automatically converted into notice lists or SBOMs in a proprietary format. Each dependency's license text, source code, and additional JSON-based license data were retrieved from the Nexus repository using Maven classifiers (e.g., "license," "licenseInformation," and "sources").

Automatic OSS-compliance checks could be embedded into the build processes of software components using Nexus-sourced data processed by the OMP. In accordance with the predefined exclusion list mechanisms, license constraints were enforced, and the process was aborted if a license contravened any internal policy. It was also determined whether specific OSS components required the inclusion of source code or acknowledgments. If source code was unavailable under the required classifier, the build was halted to prevent inadvertent license infringements. License texts, traceability rationales, and disclaimers mandated by certain licenses were stored in JSON format, which was uploaded into the artifact repository alongside the OSS binaries. As described in [2], this approach made it possible to detect missing artifacts (e.g., license text not found) and to fail the build accordingly, thereby reducing manual errors.

This paper undertakes a critical assessment of the initially proposed concept, highlighting areas that need adjustments. The original framework was designed mainly for Java projects, which had long served as the primary focus. In recent years, project priorities have shifted toward web development with additional JavaScript or TypeScript code and dependencies within the same development projects. This expanded scope now requires integrated management of not only JAR dependencies but also NPM packages, introducing new technical demands.

It has been recognized that the present OSS license management tool meets original objectives but has reached its limits. Although a dedicated license management solution is in use, adding or updating entries has proven cumbersome, and performance issues arise when retrieving license information for multiple dependencies simultaneously. The License Playbook remains only loosely integrated and lacks robust support for large collections of interrelated dependencies that share similar license details. Storing license data in Maven release repositories has also complicated error corrections, since the contents of such repositories are usually not intended to be changed retroactively. Moreover, the retrieved metadata remains confined to build artifacts, and extensive reports are generated only within individual projects but not in a central location or across multiple projects.

1.2 Assessing the OSS Compliance process

A fundamental baseline for OSS compliance was established through the combination of the organizational-technical framework, the in-house OSS management tool, and the curated License Playbook. Nonetheless, several alternative solutions are now being explored so that evolving requirements can be better addressed. This process has only just started and is still a work-in-progress.

The possibility of extending or replacing the existing tools with a more flexible software management suite is proposed, particularly for projects that incorporate both JAR-based and NPM-based dependencies. By moving toward broader automation and improved integration, it is anticipated that repetitive manual maintenance will be reduced and that reliable, centralized license information will be preserved across both new and legacy codebases.

In line with these measures, additional enhancements for more robust OSS compliance have been identified. In particular, it has been planned to:

- Introduce tools for analyzing OSS source code, so that embedded and shaded transitive components not explicitly declared can be detected in an automated manner.
- Employ a SBOM difference ("diff") tool to compare the SBOMs for consecutive product versions (e.g., from 1.0 to 1.1), thereby revealing any new or removed OSS components.
- Broaden application of the existing OSS compliance process so that other free assets like XML schemas or icon and theme files are also covered.

Through these efforts, it is expected that a higher degree of automation will be achieved as well as a higher level of understanding about changes between versions of developed software components, thereby offering better support to development teams and strengthening the overall compliance strategy.

2. STATE OF THE ART IN OPEN SOURCE LICENSING

As mentioned in the previous section, OSS is vital to modern development. Over time, various best practices have been developed to address legal, security, and license-related challenges, particularly those arising from complex software supply chains.

Given the broad scope of OSS compliance, the section focuses on a narrower selection of key aspects to foster efficiency, particularly since more general aspects have already been addressed in [1]. Specifically, these aspects include:

- Analyzing a library for applicable license terms
- Storing OSS metadata information, including concluded license and traceability information
- Generating SBOMs including license information for both internal and external purposes
- Validating OSSL compliance during CI/CD processes

By narrowing the focus to these core elements, current best practices can be more systematically examined, and a clearer understanding of the interdependencies between technical and organizational measures can be provided.

2.1 Software Bill of Materials

It has been observed that SBOMs play a pivotal role in ensuring transparency. These SBOMs, inspired by traditional bills of materials, are commonly generated through automated tools and provide a detailed list of OSS dependencies and associated metadata. In the Java realm, SBOMs have been regarded as enriched representations of Maven dependency graphs; however, language-agnostic SBOM support has become increasingly important for web-oriented projects and other multi-language environments. In recent years, a growing trend has been noted whereby SBOM artifacts are published directly to repositories such as Maven Central [3]. These artifacts are intended to reflect actual dependency relations, yet discrepancies sometimes emerge due to incomplete or outdated project metadata, underscoring the continued need for systematic SBOM validation.

2.2 Processes and Standards

Various scanning and analysis tools have been employed to facilitate comprehensive OSS compliance monitoring. Common standards like CycloneDX and SPDX support automated workflows and data sharing, while tools such as Grype[1] and Syft[2] focus on vulnerability detection in container images. Platforms including Docker Hub or Quay.io are also used to store and distribute SBOMs alongside software images, thereby streamlining responses to emerging security threats or licensing updates.

Information Quality: Automated scanning and analysis approaches are dependent on accurate descriptors and metadata, which can be prone to human error or misconfiguration.

In the effort to refine OSS compliance, a method described in [4] introduced different categories for the varying origin of license information, such as declared license, concluded license, and extracted license, illustrating different levels of license detail and the potential need for human review. Although automation alleviates many repetitive tasks, it has been recognized that verification steps remain essential to prevent the propagation of license reporting, SBOM generation, or repository maintenance errors.

SBOM formats - CycloneDX & SPDX: CycloneDX is often used for security-centric scenarios, focusing on delineating software components along with their known vulnerabilities to improve supply chain security. While it can incorporate SPDX license identifiers to standardize license metadata and SPDX license expressions for describing complex AND/OR multi-licensing situations, it does not support the more advanced licensing metadata capabilities of SPDX such as per-file licensing or cross-document references.

SPDX, on the other hand, was originally devised to streamline the license identification and compliance process, placing strong emphasis on standardized license declarations and provenance data to support legal audits and documentation requirements with the aim of facilitating compliance with legal regulations.

[1] https://github.com/anchore/grype

[2] https://github.com/anchore/syft

Although both standards convey metadata about dependencies, CycloneDX excels in vulnerability management, whereas SPDX provides broader coverage of licensing details relevant to compliance requirements [5].

Tools: Based on insights from [6], a range of tools has been evaluated to address various aspects of SBOM management. These tools span different functional categories that cover OSS component analysis, data storage, and completeness checks, and they serve diverse use cases within software supply chain management.

In terms of proprietary solutions, platforms such as GitLab stand out due to their integrated approach, which combines source code management, CI/CD pipelines, artifact management, supply chain security and embedded license scanning capabilities. These platforms aim to streamline compliance processes by offering a single interface for tracking and reporting license obligations. However, it remains unclear how such a platform can handle inaccurate or missing license information [7].

Among open source solutions, the Eclipse SW360[3] platform represents a comprehensive approach that integrates specialized tools such as Fossology[4] for artifact inspection and license detection. Fossology itself provides file-level scanning, license text extraction, and automated reporting, all of which contribute to detailed documentation and traceability. Given its modular architecture, Fossology could potentially replace or enhance existing in-house compliance tooling, particularly for organizations seeking a transparent, community-driven ecosystem [8].

Additional open source options such as ScanCode[5] or Licensee[6] address similar challenges, providing scanners and parsers that automate license discovery. Their outputs can be fed directly into SW360 or comparable systems, enabling unified storage and management of compliance data. Consequently, an ecosystem that combines SW360 and Fossology (or similar tools) can deliver end-to-end support for the library analysis lifecycle, collecting OSS information, verifying license details, and maintaining a centralized repository of compliance artifacts. This integrated approach can be an effective replacement for current processes, provided that performance and interoperability requirements are met.

Problems with inaccurate license information in dependencies are nowadays also being taken more seriously by the maintainers of open source projects. In several cases, it was possible to inform them about inaccuracies and have these issues resolved, so that their projects now provide correct license information. Furthermore, joint OSS community efforts such as the curated OSS license data store ClearlyDefined[7] that has been initiated by Microsoft have now gained momentum, with the aim of addressing such issues [9]. For the authors' specific dependencies, it should still be verified whether the service already contains equivalent corrections. If this is not yet the case, the authors have a valuable opportunity to contribute their own corrections and thereby give back to the community.

3. ADAPTING THE INTERNAL OPEN SOURCE PROCESS

Since the initial organizational and technical framework for OSS license compliance was established, the field has been observed to grow more widespread and mature. Many defined processes and best practices appear to have been adopted across a growing number of industries, supported by a wide variety of both proprietary and open source tools. These tools are applied in diverse scenarios and continue to be updated to address evolving use cases.

At the same time, a dedicated consortium has been established: the international "Open Source Tooling Group" [10]. It was formed by companies and other organizations that provide or use OSS-based license-compliance tools. Within this consortium, work is being carried out to create an interoperable ecosystem of OSS tools for license compliance, enabling organizations to fulfill their compliance obligations more effectively.

Eclipse SW360 has been identified as a promising candidate for further consideration as an open source project that can be employed as a central catalog for software components in enterprise environments. A web application and a repository are provided, enabling the collection, organization, and sharing of data on software components. As a result, a unified view of compliance information can be maintained.

On the other hand, a more comprehensive coverage of existing use cases could also be achieved by leveraging GitLab, since projects are already managed within that environment.

At this stage, no final decision has been made regarding which specific tools will be deployed. However, given the challenges encountered with self-developed tooling, it has been concluded that unnecessary duplication of existing solutions should be avoided. Instead, it is considered more efficient to profit from the ongoing improvement of established tools. By following this approach, the issues identified so far are expected to be addressed, and compliance processes are anticipated to align with emerging standards and future requirements.

4. CONCLUSION AND FUTURE WORK

4.1 Conclusion

The OSS compliance landscape has seen rapid growth in tools that automate key tasks of license identification, SBOM generation, vulnerability scanning, and artifact reporting. By combining multiple utilities, such as CycloneDX tools (for SBOM creation), OWASP Dependency-Track[8] (for continuous monitoring), Eclipse SW360 (for component cataloging and notice management), Fossology (for detailed license scanning), and container scanning tools (e.g., Grype, Snyk), organizations can implement end-to-end compliance processes for both traditional (e.g., Maven-based) and modern (e.g., NPM, containerized) projects. Moreover, publishing SBOM artifacts in online registries such as a GitLab Package Registry or JFrog Artifactory closes the loop, ensuring that both internal stakeholders and external consumers receive an up-to-date record

[3] https://eclipse.dev/sw360
[4] https://fossology.github.io
[5] https://aboutcode.org/scancode
[6] https://licensee.github.io
[7] https://clearlydefined.io
[8] https://dependencytrack.org

of included open source components and their respective licenses.

4.2 Future Work

The next steps will involve a comprehensive evaluation of the compliance tools identified during this study to verify their alignment with our technical and organizational needs. Once suitable candidates have been identified and confirmed, existing compliance artifacts and metadata will be ported to the selected platforms. Finally, an updated OSS compliance process will be introduced, integrating the validated tools into the standard development and release pipeline.

Although an integrated toolchain automates large portions of the OSS compliance workflow, several issues remain that must be addressed at the organizational level:

- Residual false-positive and false-negative findings have to be triaged and corrected manually
- Dedicated organizational roles (e.g., an Open-Source Compliance Officer) are required so that responsibilities are clearly assigned and inquiries are centrally handled.
- Continuous training programs must remain in place to keep engineering and legal staff informed about license obligations and tool usage.
- Periodic internal or third-party audits must be scheduled to verify that tooling outputs align with contractual and regulatory requirements.
- Exception-handling workflows must be documented to accommodate policy-compliant alternatives that may fall outside the coverage of the automated tool-chain.

5. REFERENCES

[1] S. Dyck, D. Haferkorn, J. Sander, An organizational-technical concept to deal with open source software license terms, Proceedings of the 20th World Multi-Conference on Systemics, Cybernetics and Informatics (WMSCI 2016), July 5–8, 2016, Orlando, Florida, USA.

[2] S. Dyck, D. Haferkorn, C. Kerth, A. Schoebel, Automating Open Source Software License Information Generation in Software Projects, Proceedings of the 22nd World Multi-Conference on Systemics, Cybernetics and Informatics (WMSCI 2018), July 8–11, 2018, Orlando, Florida, USA.

[3] Y. Gamage, et al., Software bills of materials in Maven Central, Proceedings of the 2025 IEEE/ACM 22nd International Conference on Mining Software Repositories (MSR 2025), IEEE, 2025.

[4] Georgia M. Kapitsaki, Frederik Kramer, Nikolaos D. Tselikas, Automating the license compatibility process in open source software with SPDX, Journal of Systems and Software, Volume 131, 2017, Pages 386-401, ISSN 0164-1212, https://doi.org/10.1016/j.jss.2016.06.064.

[5] A. Muneza, R. Manzi, et al., SBOM generation tools and formats affect compliance with US standard, Proceedings of the 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER 2022), March 15–18, 2022, Honolulu, Hawaii, USA.

[6] A Landscape Study of Open Source and Proprietary Tools for Software Bill of Materials (SBOM), Mehdi Mirakhorli and Derek Garcia and Schuyler Dillon and Kevin Laporte and Matthew Morrison and Henry Lu and Viktoria Koscinski and Christopher Enoch, 2024, https://arxiv.org/abs/2402.11151

[7] C. Cowell, N. Lotz, C. Timberlake, Automating DevOps with GitLab CI/CD Pipelines: Build efficient CI/CD pipelines to verify, secure, and deploy your code using real-life examples, Packt Publishing, 2023.

[8] O. Fendt, M. C. Jaeger, Open Source for Open Source License Compliance, Proceedings of the 15th IFIP WG 2.13 International Conference on Open Source Systems (OSS 2019), May 26–27, 2019, Montreal, Quebec, Canada.

[9] ClearlyDefined and fostering multi-stakeholder collaboration for accurate licensing metadata at scale, Niccholas Rodriguez Vidal, Submitted to Open Infrastructure Fund / Fondo de Infraestructura Abierta}, 2023

[10] A. Azhakesan, F. Paulisch, Sharing at scale: an open-source-software-based license compliance ecosystem, Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP 2020), May 23–29, 2020, Seoul, Republic of Korea.