SBOMS AS A SOLUTION IN THE SOFTWARE SUPPLY CHAIN

Trevor Stalnaker

William and Mary

Abstract

Software Bills of Materials (SBOMs) have emerged as a tool to facilitate the management of software dependencies, vulnerabilities, licenses, and the supply chain. While significant effort has been devoted to increasing SBOM awareness and developing SBOM formats and tools, recent studies have shown that SBOMs are still an early technology and not yet adequately adopted in practice. This report* expands on previous research by comprehensively investigating the challenges stakeholders encounter when creating and using SBOMs. We survey 138 practitioners belonging to five stakeholder groups (practitioners familiar with SBOMs, members of critical OSS projects, AI/ML, cyber-physical systems, and legal practitioners) using differentiated questionnaires, and interview 8 survey respondents to gather further insights about their experience. We identify 12 major challenges facing the creation and use of SBOMs, including those related to SBOM content, deficiencies in SBOM tools, SBOM maintenance and verification, and domain-specific challenges. We highlight 4 actionable solutions to these challenges and present the major avenues for future research and development.

Introduction

The software supply chain has increasingly grown in complexity with the proliferation of open-source software (OSS) and AI/ML components. Consistent with NASA's stated directives on technology sharing, organizations and developers often accomplish tasks by integrating components from a variety of vendors. However, leveraging external packages does not come without a cost. The fate of a software product becomes intrinsically tied to its evolving dependencies. If a dependency displays a vulnerability, then so too could the final product, potentially leading to severe consequences. Moreover, failing to comply with the license terms of software dependencies can lead to severe legal and economic consequences for organizations.

Software Bills of Materials (SBOMs) have emerged as a way to facilitate the management of dependencies[10], leading to improved management of software vulnerabilities, enhanced license compliance, and increased transparency in the software supply chain[9].

The 2021 US Presidential Executive Order 14028 on Improving the Nation's Cybersecurity[3] gave momentum to SBOM formalization and adoption as it requires companies selling software to the US government to provide SBOMs. This was prompted by recent supply chain attacks, such as the SolarWinds breach, and critical vulnerabilities such as those affecting the Log4J library. SBOMs are currently championed by the NTIA and organizations such as the Linux Foundation and OWASP. Significant effort has been put into promoting SBOM formats and tools that can create and process SBOMs, with the goal of increasing adoption and fully enabling the benefits that SBOMs offer[9].

Although organizations and developers have acknowledged the importance of SBOMs and anticipate using them more frequently in the coming years[8,15], recent research highlights concerns regarding commitment to SBOMs and the actualization of SBOM benefits[8,19]. These concerns arise due to the lack of industry agreement regarding the content of SBOMs across different domains, as well as how they should be employed and integrated into their development and operational processes[18,70]. An additional barrier is the lack of mature tools for SBOM production and consumption[8,19,21].

Considering this it is imperative to understand (i) how developers and other stakeholders currently create and use SBOMs, (ii) additional opportunities/benefits that SBOMs can offer for different types of software and stakeholders, (iii) the specific challenges that prevent stakeholders from fully enjoying SBOM benefits, and (iv) actionable solutions to overcome such challenges.

Background

A Bill of Materials (BOM) refers to a list of raw materials, components, and parts needed to manufacture an end product. The concept has been applied to software systems as Software BOMs (SBOMs), which identify a project's dependencies and their provenance. SBOM

* The content of this report is supported by work that is to be published at ICSE '24[6].

use cases include component inventory, vulnerability analysis, and license compliance. Two Major SBOM format specs currently exist: SPDX and CycloneDX.

As modern software systems go beyond the mere integration of libraries and frameworks, various initiatives have proposed different types of BOMs to account for other components typically integrated into a software system (e.g., hardware devices, firmware, APIs, or AI/ML models). To this end, our study targets specific populations of software stakeholders (e.g., AI and Cyber-Physical Systems practitioners) to understand needs that could be fulfilled by various kinds of BOMs.

While SBOMs have existed for some time[1,2,7], they are only now becoming more widely known[12,20]. The analysis of their uses and shortcomings has been investigated only by a few recent studies[5,19,21]. A comparison between our study and the most related prior studies, regarding methodology & scope can be found in our ICSE submission[6], and a more detailed comparison can be found in our replication package[17].

Study Design

This study aims to investigate the challenges encountered by stakeholders when creating and using SBOMs, and how such challenges can be addressed.

We aim to address the following research questions:

RQ1: How do stakeholders create and use SBOMs?
RQ2: What challenges occur in this process?
RQ3: What are solutions to SBOM challenges?

Next, we describe the study methodology, which includes five distinct surveys and follow-up interviews with participants from different stakeholder groups. See Fig. 1 for a high-level overview of our methodology.

As the study involves human subjects, the methodology was approved by the ethical board of the college.

Survey Design

We designed survey questions considering our RQs, previous literature on SBOMs, and general guidelines for survey design.

Since the study involves a general population of: (1) software developers and other stakeholders that have interacted with SBOMs, and (2) domain specialists (AI/ML, CPSs, and legal practitioners), we designed questionnaires with questions asked to all stakeholder groups and questions asked to the specific groups.

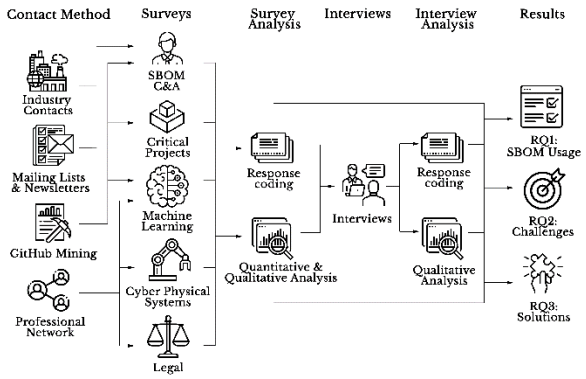

Figure 1: Methodology Overview

The surveys contain a mix of (five-point) Likert-scale, multiple-option, and open-ended questions that asked about: SBOM content, use cases, benefits, distribution preferences, challenges, potential solutions, dependency management practices, and legal aspects. All questionnaires also featured a consent form, a statement about data confidentiality, and a demographics section. Participants who completed the survey entered a lottery to win one of ten $50 USD Amazon gift cards.

Participant Identification

To explore different facets of SBOM usages, we identified 5 participant groups: SBOM Community and Adopters, contributors of critical OSS[13], AI/ML, Cyber-Physical Systems (CPS), and legal practitioners.

SBOM Community and Adopters (SBOM C&A)

Contacting people who directly use SBOMs and related technologies allowed us to obtain firsthand feedback on how SBOMs are currently used, as well as any perceived deficiencies in current SBOM standards and tools. Within this group, we identified five sub-groups of stakeholders. We asked participants to self-identify as belonging to one or more of the following groups:

SBOM Consumers: Read existing SBOM to gather information on dependencies, vulnerabilities, or licenses.
SBOM Producers: Document a software system and its dependencies with an SBOM using a particular format.
SBOM Tool Makers: Contribute to the development of tools that facilitate the creation or use of SBOMs.
SBOM Educators: Create or compile educational resources about SBOMs, including guides and tutorials.
SBOM Standard Makers: Contribute to specifications for the creation and usage of SBOMs.

Eligible participants for this group were identified based on their potential experience with SBOMs, the

supply chain, and software development, via a combination of three different approaches:

Keyword-based search of GitHub repositories. Combining manual effort and automated tools (based on GitHub APIs), we located public GitHub repositories by searching issues, commits, and files for keywords and traces related to SBOMs and the supply chain. Next, we considered contributors to those repositories and gathered only publicly available contact information.

Identifying dependencies between GitHub repositories. We found additional participants by (1) examining GitHub profiles/organizations that listed projects with SBOM-related tags as topics, and (2) using GitHub's dependency feature to locate dependent projects with SBOM-related tags. These repositories and their contributors logically represent groups currently using SBOMs. In total, we identified 4,423 developer email addresses via GitHub mining.

Sharing the survey in mailing lists. To locate additional individuals familiar with SBOMs, we published a call for participants through SBOM-related mailing lists, including the SPDX and the OpenChain mailing lists.

Developers of Critical Open-Source Systems

The Open-Source Software Foundation's workgroup on Securing Critical Projects compiled a list of the 102 most critical OSS, comprising 564 total repositories[13]. The projects include the Linux Kernel, programming languages, package managers, build systems, databases, etc. Given the role of SBOMs in the software supply chain, we sought to administer a targeted survey examining these critical projects, which are widely depended on and may have a greater need to produce, use, and distribute SBOMs. The actions of these projects are also likely to represent and set the tone for the rest of the open-source landscape. Also, examining these projects allowed us to assess how SBOMs have spread beyond early adopters.

Using the GitHub API, we mined the top-10 contributors (by # of commits) for each of these 564 repositories. Where there were fewer than ten total contributors, we examined all that were available.

CPS Developers and Researchers

These are people with expertise in cyber-physical systems (autonomous vehicles, medical monitoring and industrial control systems, robots etc.), which entail a close interaction between hardware and software. Given these systems have their own supply chains and are becoming more popular in certain domains, surveying this group allowed us to examine unique challenges facing the usage of SBOMs and HBOMs, as well as how the two may interact. CPS participants were identified from our professional network.

AI/ML Developers and Researchers

These are: (i) Top-10 (by number of commits) developers that contribute to a machine learning project hosted on GitHub (with 100+ stars) and expose a public profile. AI/ML projects were identified by matching the projects' topics to keywords such as "machine learning" or "artificial intelligence" (see the full list of keywords in our replication package[17]); and (ii) AI/ML practitioners in our academic/professional network.

AI/ML components have their own supply chains but are also increasingly integrated into traditional software products. Model/data provenance is essential to security (e.g., model poisoning), licensing, usage, and research of AI/ML systems. The needs, challenges, and use cases facing AI/ML developers may be similar and different from those of typical SBOM users. By surveying this group, we aimed to understand these similarities and differences.

Legal Practitioners

Through our professional network, we identified a legal practitioner with a technical background who could answer questions about non-technical challenges facing SBOM use. This includes examining how SBOMs interact with regulations, contractual obligations, and more. The views of one respondent are not representative of the field at large, but with only a small pool of legal practitioners having software development and SBOM experience, this group is the hardest group to survey at scale.

Survey Response Collection and Analysis

Survey responses were collected using Qualtrics[4]. Survey participants were only presented with questions related to the group(s) they selected. The survey for SBOM community and adopters was kept open for four months, with three waves of invitations. The remaining surveys were kept open for two to four weeks.

Via email and mailing list posts, we invited 4.4k+ individuals to participate in the surveys and received 229 complete responses in total. After removing personal information, the responses were analyzed following the procedure described below, resulting in 150 valid responses. Table 1 provides details on our responses.

Survey	Full Resps	Valid Resps	Fam. w/ SBOMs	Inter-views
SBOM C&A	179	101	61	4
Critical	22	22	13	1
ML	21	20	8	1
CPS	6	6	1	1
Legal	1	1	1	1
Total	**229**	**150**	**84**	**8**

Role	#
P	34
C	31
TM	24
E	14
SM	16
O	7

P=Producer, C=Consumer, TM=Tool Maker, E=Educator, SM=Std. Maker, O=Other

Table 1: Survey Respondents

For closed-ended questions, we aggregated results using descriptive statistics and discussed them. We examined responses from Likert-scale questions to determine practitioner sentiments, as well as frequently selected answers to multiple-choice questions to identify common SBOM use cases and challenges. We report the most frequently selected answers in Study Results.

For open-ended questions, a coding approach was applied in line with[16]. Two researchers ("annotators" in the following) performed a first phase of open coding on the first 28 valid responses of 101 received for the SBOM community and adopters survey. They independently assigned one or more codes to each response.

Once both annotators completed the open coding for the first 28 valid responses, they convened to settle disagreements and consolidated a set of labels. Since multiple codes could be assigned to each response and disagreements were discussed, we did not base our analysis on inter-rater agreements.

From this point, the remaining responses were coded by the annotators independently. During the further coding, the annotators started from the previously established codes (available in a shared spreadsheet); yet, they had the option of adding new codes, that would, in turn, become available to the other annotator.

When coding was completed, annotators met to discuss their coding and reconcile the disagreement cases. Results were analyzed by leveraging descriptive statistics on the codes the annotators assigned to each question.

Throughout the whole coding process, the annotators flagged and reviewed answers that were nonsensical, did not answer the survey questions, were copy-pasted from the web, or appeared to be generated through ChatGPT. In this way, 41 responses were removed from the analysis. Another 20 responses were removed because of numerous blank or repeated answers, and 18 were discarded as spam (e.g., same email/IP addresses or identical responses).

Interview Design and Response Analysis

We conducted one-hour semi-structured interviews with eight participants of the surveys, to gather deeper knowledge about their experience and responses.

We selected respondents from the 5 surveys whose responses warranted further investigation. We sought interviews with respondents who (1) gave detailed replies highlighting interesting use cases, challenges, and potential solutions; (2) demonstrated experience in their field; and (3) diversified our interviewee pool in terms of their role (consumers, producers, etc.). We hoped to capture a variety of perspectives from respondents familiar with SBOMs and those that were not but had interesting thoughts on how SBOMs might affect them.

The interviews were conducted in two parts. The first part asked follow-up and clarification questions which varied depending on the survey responses of each interviewee (e.g., You highlight the importance of identifiers for each software element. Why are these identifiers so important?). For interviewees in the SBOM C&A group, a second part of the interview featured five questions that were common across interviews in that group.

Interviews were conducted over Zoom and recorded with the participants' permission. The recordings were transcribed using the Whisper speech recognition tool. The interviews included two researchers, taking notes about the given responses. The same authors parsed and analyzed participant responses and notes individually, employing an open coding strategy like that used in the analysis of the survey responses and discussing the coding when needed. Interviewees were given a $50 USD Amazon gift card.

Study Results

56% of the study participants are familiar with SBOMs.

RQ1: SBOM Creation and Usage

SBOM Awareness and Formats

Of the 50 producers, consumers, and tool makers surveyed, 16 reported using SPDX, 8 CycloneDX, and 12 both. Those that consume SBOM do so frequently: 35.5% (11/31) of participants stated they use them daily and 29% (9/31) weekly. Of the 22 critical OSS survey participants, 9 were unfamiliar with SBOMs and 7 were aware of SBOMs, while not adopting them yet. One interviewee mentioned how the limited interest is also due to the limited tool support and the need for manually maintaining SBOMs.

Of 6 CPS respondents, 3 were familiar with HBOMs and 2 had used them, but with bespoke formats.

No ML practitioners surveyed were aware of BOM formats for AI systems or datasets, but one interviewed standard maker was on an SPDX team that had worked on adding fields to SPDX 2.x for ML systems. Since our initial survey, we have learned that CycloneDX has added an ML-BOM to its specification.

Participants expressed that pressure to maintain SBOMs primarily targets industry and projects at the end of a supply chain, while projects near the beginning have little incentive to produce them. Some projects, such as the Linux kernel, may have no real dependencies of their own and so do not require dependency management methods.

This results in downstream components creating SBOMs on behalf of their dependencies. Other than being a cumbersome task done for somebody else; as one interviewee said, "[the risk is] miss[ing] something because you got to go back and dig back through all these different dependencies."

SBOM Use Cases, Benefits, and Data Fields

Among SBOM practitioners we found that dependency tracking (55), security (22), and licensing (22) are the main use cases for SBOMs. Other responses include software versioning (14), provenance (10), documentation (6), and transparency (4).

While tracking vulnerabilities was a main use case for consumers (80.7%), producers (100%), and tool makers (83.3%), some respondents were concerned that SBOMs might provide a road map of vulnerabilities for attackers.

When 41 SBOM producers, tool makers, and standard makers were asked which data fields should be included in SBOMs, responses varied. The most common answers were general information about the software components: version number (24 of 41), license (22), component name (18), and a URL to the component (18). Notably, 13 respondents indicated that the SBOM should contain unique identifiers for the software component the SBOM is documenting and/or its dependencies.

Although we found little evidence to suggest AI and DataBOMs are being used in practice, respondents mentioned two potential use cases. These BOMs could facilitate ML model reproducibility and help to identify / verify datasets across academic papers. Specifically, AIBOMs can provide transparency into how a model was trained, providing information about its architecture, hyper-parameters, and any pre-trained base models used. By providing provenance and usage information, a linked DataBOM can also make developers aware if a model was trained using a poisoned, biased, or illegally sourced dataset.

The surveyed and interviewed CPS practitioners mentioned that BOMs could serve as regulatory documents for critical embedded systems and that they could increase the transparency and reproducibility of research results in academic communities. For these tasks, the BOMs must communicate information related to the physical hardware components (part numbers, manufacturer, etc.), firmware, and other software (including configurations) of the system.

SBOM Generation and Distribution

Despite the NTIA recommendations[103], there is currently no agreed-upon method for distributing SBOMs. That said, respondents have the expectation that the developers of third-party components should be the ones creating, maintaining, and distributing SBOMs along with their software.

Concerning support for DataBOMs and AIBOMs, two survey participants mentioned that Hugging Face dataset cards could serve as DataBOMs. Three respondents mentioned the same service's model cards, providing similar information to AIBOMs.

When asked when SBOMs should be generated, producers said: during each build (28/34), when publishing a major release (21/34), during deployment (19/34), and at the developer's discretion (7/34).

RQ2: SBOM Challenges

C1: Complexity of SBOM Specifications

A common concern among participants is the complexity of SBOM specifications, as stated in this comment: "[...] one core issue [...] is definitely a tension between use case coverage and the complexity of the spec." Adding support for new use cases lengthens and complicates SBOM specifications.

We noticed that the user's perception of the SBOM specification is in part determined by their use case. "If all you're interested in is licensing, [...] [you] don't want to have to learn [about other domains like security] just to be able to use the spec." However, "even if

[SBOM producers] don't have that use case in mind, [their] consumers [might]."

Participants also mentioned the lack of adequate educational resources about the SBOM specifications to better communicate their content. One interviewee mentioned: "It's not just simplicity in the spec. It's not simplicity in the tooling, but how we message it and how we communicate it."

C2: Determining Fields to Include in SBOM

While some fields (software versions, licenses, or component names) are commonly agreed upon, others depend on the use case. For example, practitioners seeking to analyze their software for vulnerabilities may require BOMs to link to an external vulnerability database.

Interesting is the case of BOMs for AI/ML. AI/ML respondents expressed the need to include provenance information about datasets and models in SBOMs, to enable model verification and reproducibility. Other than standard SBOM fields, the 20 respondents from this group pointed out fields such as descriptions of the training data (17) and validation/testing data (14), preprocessing steps taken on the data (13), dataset version (13), and used optimizers/loss functions (13). When asked about fields needed in DataBOMs, they highlighted data sources (18), data transformations (18), preprocessing steps (17), dataset size (16), known/potential biases (14), and data collection procedures (14).

Of the 6 surveyed CPS practitioners, 3 expressed a need for hardware part numbers, 2 for testing and quality assurance data, 1 for system deployment information, 1 for manufacturer information and location (e.g., company and geographical location), and 1 for known limitations about parts (e.g., if they are not suitable for certain tasks due to security risks).

Adding additional fields to SBOM specifications makes the documents more useful, but as mentioned previously, also contributes to the complexity of the specification (C1).

C3: Incompatibility Between SBOM Standards

Responses show that competing standards confuse developers. When consuming SBOMs, 23.33% of the SBOM practitioners stated that different standards pose a challenge, due to interoperability issues between standards and inconsistency between standards and tooling. Despite this, one practitioner said: "Competition is good [...] I definitely think that we have moved faster because of CycloneDX and SPDX having this kind of competition."

There are also multiple ways of creating an SBOM for the same piece of software, often for backward compatibility reasons. One practitioner remarked: "You may have two SBOMs that technically represent the same software, but they're being produced by two different tools, and they look radically different."

Fortunately, respondents suggested there are plans to increase and maintain interoperability among different standards. As one interviewee put it, "I think [the standards are] on two different paths now. [...] To say one's going to die over the other or try to do the grand convergence and bring them together, you're just not going to, it's just going to take too long. [...] it makes much more sense to try to get the two groups to collaborate."

Addressing incompatibility between standards would likely require a community-led effort, creating clear mappings between them, and developing tools that support these mappings.

C4: Keeping SBOM Updated

Once an SBOM has been created, it must be maintained along with the software it represents. Substantial changes to an SBOM over time are known as SBOM drift. Such changes can occur suddenly, such as a dramatic increase in the number of dependencies when an application is added to a container, or when new vulnerabilities are discovered in dependencies asynchronously from changes in the software — one interviewee described SBOMs as "a static vulnerability snapshot of the state of a [piece of] software at a certain point of time."

When asked about deficiencies in standards, 4.35% of participants expressed issues concerning keeping SBOM updated (1), upkeep requirements (1), and the syncing of SBOM versions (1). Of 3 critical OSS developers that consume SBOM, 1 mentioned difficulty in keeping SBOMs up-to-date. This motivates a need for tools which can dynamically update SBOMs as changes occur [114].

C5: Insufficient SBOM Tooling

Figure 2 shows stakeholders' views on whether current SBOM tools address the needs of their users. While we generally found a lack of consensus among participants,

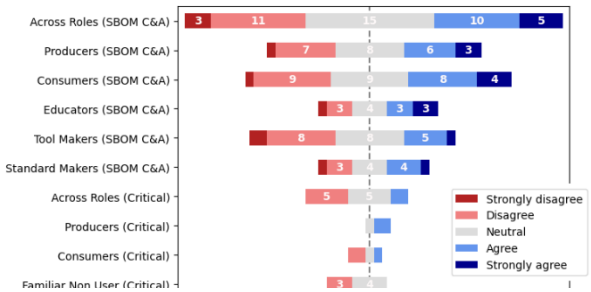

Figure 2 – Perceived sufficiency of SBOM tooling

we observe that tool makers are slightly more negative. These results, combined with the participants' open-ended answers, suggest that current tool support is insufficient. One participant identified a lack of "automated ways to generate SBOM for embedded code like assembly, C, C++."

Across stakeholder groups, there was little familiarity with tools. 85% of the ML respondents were unaware of any tool support for generating AIBOMs, and 90% were unaware of tooling for DataBOMs. Only one CPS practitioner was aware of existing tools. Part of the problem may be low demand. One practitioner had used "a few [SBOM] tools [but] they [didn't] work very well," noting that "it would be nice if they were fixed" but "nobody seems to care because maybe nobody's using them."

Some projects with specific features may be unable to use current tooling, as no support exists for them yet. For example, one practitioner noted that current tooling could not "run fast on projects with tens of thousands of files... They're not designed to work with very, very large projects." Two producers faced challenges involving projects that used multiple programming languages, suggesting an unmet need for tools to support multi-language projects. Similarly, tools should be available for SBOMs to be created when only certain types of information are available, such as building SBOMs from binaries: "[T]here's source SBOMs. There's binary only SBOMs. There's SBOMs that have dependency information. There's SBOMs that have really just information about the package [...]."

C6: Inaccurate and Incomplete SBOMs

An SBOM is only as good as the information that it provides. If the information is inaccurate or incomplete, it becomes difficult for teams to make informed decisions concerning the dependencies, licensing, and security of their projects.

According to the results, SBOMs are of varying quality and are often found wanting. 33% of SBOM consumers from the SBOM C&A survey mentioned poor quality SBOMs as one of the challenges they had faced in using SBOMs. 25% of the consumers from the critical OSS groups stated the same. Surprisingly, 12% of the SBOM producers had the same complaint.

Consider that the minimum SBOM requirement would be to include all direct and transitive dependency information, including the URLs of their sources. The legal practitioner we interviewed mentioned that, in his/her experience, this condition is rarely met.

Participants also discussed "false positives" in BOMs. For example, using a dependency that has a vulnerability does not necessarily mean the software will be impacted. Determining if a project is actually impacted is a difficult problem and requires sophisticated tooling.

The problem of inaccurate SBOMs also impacts tool developers. One respondent described how "it's been difficult to build tooling that accepts an SBOM when I'm not sure if all the fields that I'll need to depend on have been filled out."

C7: Verifying SBOM Accuracy / Completeness

33% of the critical OSS contributors mentioned how SBOM verifiability is a major challenge. This was also reported by 3 participants of the SBOM practitioner survey. The enforcement of SBOM correctness should not be so strict that it impedes SBOM creation and adoption. For example, the legal practitioner we contacted cautioned that holding BOM creators liable for inaccuracies in the documents they produce is a disincentive to creating SBOMs at all. For security reasons, consumers will also need mechanisms to validate the integrity of an SBOM, to check that nobody has (maliciously) altered it in transit. Well-known solutions, e.g., those based on hashing and checksums, can be applied to this context.

C8: Differences Across Ecosystems

Participants indicated that SBOM support varies across languages and package ecosystems. One interviewee mentioned: "a big part of the bottleneck is just retrieving all the information that needs to go into the SBOM and getting it from different sources [...] some language communities do a better job of capturing the metadata [to] include in the SBOM." Some respondents even suggested that tools from the same standard (e.g., CycloneDX) drastically vary in quality across languages.

As another participant mentioned, this "creates an ecosystem challenge for getting that data in an SBOM in a reliable way, because there are some data sources that you can't really trust."

We also observed challenges of creating SBOMs for languages with limited or no package managers. A survey respondent mentioned: "For C/C++ projects, dependencies are typically defined in autotools or cmake files, and Node, Ruby, Python, Golang, etc. all have their own dependency management systems; typically recording exact versions is an output of the build process, although this doesn't come "out of the box" with C/C++ projects". 25% of the critical OSS developers surveyed who were familiar with SBOMs listed a lack of language support as a deficiency in current SBOM specifications, while 8.7% of SBOM practitioners agreed. When asked about tool deficiencies, 41.67% of critical OSS developers surveyed who were familiar with SBOMs expressed a need for more language-specific tooling.

C9: SBOM Completeness vs. Data-Privacy

AI/ML participants indicated that AIBOMs and DataBOMs may entail a tradeoff between completeness and privacy on large datasets, given that these datasets may contain personally identifiable, private, sensitive, or proprietary information. CPS respondents also mentioned privacy concerns in BOMs, as CPS may actively collect and process private and sensitive data from the environment.

C10: SBOM for Legacy Software

One interviewee expressed the challenge of generating SBOMs for legacy software, which may be deployed and used by certain user groups. Even if SBOMs become well-adopted and automatically generated during software builds, the question of what to do about legacy software remains. Software that is still regularly maintained could feasibly have an SBOM created, but it is more challenging for older systems where the original source code is missing or for systems written in languages that are now substantially less common (e.g., COBOL). These languages are less likely to be supported by open-source SBOM tooling. This is particularly problematic for entities like the US government or the banking industry. Community-driven effort may be needed to generate, store, and share SBOMs in such situations.

An important question is whether, for existing systems, only the newest releases require an SBOM, or if older releases that are still used by dependents also require SBOMs. The respondent said: "if ecosystems did start to publish SBOMs, [...] it would be great to see [centralized repository maintainers] go back in time, generate SBOMs for older packages"

C11: Inability to Locate Dependencies

There may be cases where during the production or consumption of an SBOM, a certain dependency cannot be located. This could happen if a dependency was removed from a package manager (perhaps it was malicious or no longer maintained) or from the associated repository. One practitioner mentioned: "They [dependencies] may have been yanked and removed from the upstream package registries, meaning that the mere fact of detecting that they exist could be a challenge" and "In some cases, [finding your dependency is] a lost cause in the sense that your source may be dead, the repository has disappeared and you're left to have to sift through random snapshots of archive.org calls made on the website. That's rare, but that happens."

A centralized database indexed on global IDs and containing provenance information for software repositories / distributions could allow developers to access critical information for projects that are no longer hosted or available. This would essentially be a third-party SBOM archive.

C12: Unclear SBOM Direction / Low Adoption

While a recent executive order[3] requires companies selling software to the US government to provide SBOMs, our results indicate that adoption and knowledge of SBOM are still limited. Moreover, while incentives for library users are clear, those for library creators are not. Given the effort and knowledge needed for creating SBOMs, most developers forgo this effort.

This suggests a fear that the work required to create and maintain SBOMs will outweigh their benefits. As one practitioner said, "I hope that the hype around SBOM will lead to something that's productive [...] and will not just be something which is a compliance requirement that's going to be met in a minimal way."

Lastly, SBOMs are still a new technology that will take time to mature. There is still a need to motivate and implement support for consumer use cases. In an interview, one respondent stated, "You know, if you are a large organization and, say, you take a magic wand, and tomorrow all your software vendors start to provide accurate SBOMs, what are you going to do with this?"

(S1) Multi-dimensional SBOM specifications		
- Structure SBOM specifications considering three dimensions, i.e., use cases, types of software, and amount of information needed (*a.k.a.* information level) - Create more structured, easy-to-navigate, and easy-to-search specifications - Improve educational material about SBOM specifications		
Challenge	**How the Solution Addresses the Challenge**	**Roles**
(C1) Complexity of SBOM specifications	Shorter and easy-to-browse SBOMs specs. without unneeded information (per use case, system, *etc.*)	P, C, TM, E, SM
(C2) Determining data fields to include in SBOMs	Optional SBOM fields added only when required. Information levels determine optional, recommended, and mandatory SBOM fields.	P, C, TM, E, SM
(C6) Inaccurate and incomplete SBOMs	SBOMs not incomplete if irrelevant/hard-to-find info for a given use case is not required.	P, C, TM
(C9) SBOM completeness and data privacy trade-off	SBOMs can tailor the required fields for data privacy according to defined information levels.	P, C, TM, SM
(S2) Enhanced SBOM tooling and build system support		
- Develop libraries and base infrastructure for SBOM production, consumption, and verification - Develop SBOM tooling for binaries and programming languages with no package managers - Integrate SBOM creation into build and continuous integration (CI) systems and AI/ML frameworks (TensorFlow, *etc.*)		
Challenge	**How the Solution Addresses the Challenge**	**Roles**
(C4) Keeping SBOMs up to date	SBOM tools compatible with build and CI/CD automation to create/update SBOMs at each build.	P, C, TM
(C5) Insufficient SBOM tooling	Improved SBOM tools with support for multiple programming languages and AI/ML frameworks.	P, C, TM
(C6) Inaccurate and incomplete SBOMs	SBOM tools integrated with build automation and AI/ML frameworks create SBOMs with dependencies actually used in binaries, releases, and AI/ML models.	P, C, TM
(C7) Verifying SBOM accuracy and completeness	Tools to check that SBOMs created from source code and binaries contain the same dependency info.	P, C, TM
(C8) Differences across ecosystems and communities	Improved SBOM tools would lead to increased SBOM adoption across languages and ecosystems.	P, C, TM
(S3) Strategies for SBOM verification		
- Third-party (community-based) certification/verification of SBOMs		
Challenge	**How the Solution Addresses the Challenge**	**Roles**
(C6) Inaccurate and incomplete SBOMs	With verification mechanisms in place, certified SBOMs would be more accurate and complete.	P, C, TM
(C7) Verifying SBOM accuracy and completeness	Verifying and certifying SBOM content leads to enhanced accuracy and completeness.	P, C, TM
(S4) Increasing incentives for SBOM adoption		
- Create mandates to create and use SBOMs for different stakeholders - Minimize the effort to create and maintain SBOMs (*e.g.*, by developing tools integrated with existing systems and processes) - Increase motivation to develop (open-source) SBOM tooling (e.g., via integration and badging in code repositories such as GitHub) - Promote SBOMs benefits/usage and improve educational materials (e.g., by promoting successful cases of SBOM usage and tooling)		
Challenge	**How the Solution Addresses the Challenge**	**Roles**
(C5) Insufficient SBOM tooling	Increased incentives for SBOM adoption would drive further development of SBOM tooling.	P, C, TM, E, SM
(C12) Unclear SBOM direction	SBOM mandates and promotion/education materials clarify SBOM benefits and usage costs. SBOM incentives would better involve open-source communities in SBOM creation/usage/promotion.	P, C, TM, E, SM

Table 2: Mapping SBOM Challenges to Solutions

RQ3: Solutions to SBOM Challenges

S1: Multi-dimensional SBOM specifications.

We identify 3 dimensions that contribute to the complexity of SBOM specifications: (1) the intended use case of an SBOM, (2) the type of software the SBOM is generated for, and (3) the amount of information documented in an SBOM. Providing clear guidance for these dimensions is needed to inform consumers/producers which fields an SBOM should contain (C2), hopefully reducing the cognitive load placed on users (C1). Only requiring relevant fields is also likely to improve the completeness of SBOM (C6).

S2: Enhanced tooling and build system support.

Across all surveys, three respondents suggested better libraries as a tooling solution. One said, "Increased investment in open-source libraries that can be incorporated in end user commercial and open-source tools [can address current deficiencies in tooling]." Well-maintained, easy-to-use libraries would serve as the foundation and motivation to develop SBOM tools (C5) providing functionality for creating, maintaining (C4), parsing, and managing SBOMs, enhancing the user experience and, potentially, SBOM adoption. Our findings indicate the need for language specific SBOM production tools. A language-agnostic tool is unlikely to adequately support all scenarios. As such, there is work to be done creating SBOM generation tools for different ecosystems, including resolving disparities in the quality of available tools. Language-specific tooling can be built on language-agnostic libraries (C8). SBOMs will likely become more accurate and complete with better tool support (C6). Moreover, in the current landscape of varying SBOM quality, consumption tools may also be responsible for checking the accuracy of the SBOMs consumed (C7). A respondent noted that consumption tools "have a perhaps harder job to make sure that the data that's being generated is accurate." Furthermore, existing build systems (e.g., Maven or Gradle) should be made SBOM-aware: capable of reading and generating SBOMs: "[O]ne way [for SBOMs to be easier to use] would be for build tools to start generating them without asking." We have observed from our surveys that developers tend to prefer processes or tools that are commonly used or predetermined: "when the recommended way of doing something is the default, then it gets done more often." SBOM generation functionality in build tools would more easily facilitate the update of SBOMs (C4). We have seen that developers rely on package management systems to obtain a list of their project's dependencies. Many of these systems also

provide quasi-SBOM files. If SBOM generation and acquisition could be handled at the package manager level, we would likely see a large uptick in adoption (C12). SBOMs could be stored along with other package information and queried through APIs. Indeed, interviewed practitioners suggested that SBOMs should be kept as close to the source as possible. As an SBOM moves further from the source, it is less likely to be up-to-date (C4). GitHub recently unveiled new functionality capable of generating SPDX documents for a cloud repository. Through integration with GitHub's Dependency Graph tool, this capability supports SBOM generation for a number of popular languages and is easily accessible to developers, marking a strong start for SBOM integration. It was also suggested that ML libraries could generate AIBOMs or play an integral part in easily accessing required information: "eventually there'll be [...] something built into TensorFlow or PyTorch [...] that outputs a document [...] that tells you the key elements [like] the hyper-parameters."

S3: Strategies for SBOM verification.

One initially apparent method to approach incomplete or incorrect SBOMs would be to hold parties accountable for the SBOMs they generate (C6), but this could lead to unintended consequences. A legal practitioner said, "[a] requirement for them to certify that it is complete or correct is only going to result in fear of creating SBOMs. 'Perfect' should not be the enemy of 'good.'" Beyond this, SPDX SBOMs are licensed under Creative Commons 0 (CC0), meaning no warranty is included and the producer assumes no liability. The open-source licensing of tools protects their creators from litigation since many licenses also do not provide a warranty. Two other solutions emerged from our surveys (C7). A third-party certification or review board could approve SBOMs and endorse them. However, as one respondent put it, "central authorities have never seemed to work too well in our industry [...]". A decentralized approach could involve the assessment of SBOMs by their consumers and other stakeholders, with issues reported to the SBOM producer or posted in a shared database.

S4: Increasing incentives for SBOM adoption.

There is a need to either minimize the effort needed to create and maintain SBOMs or by gaining other benefits, such as having tools that consume SBOM and help with developer tasks. Similarly, it is necessary to motivate the creators of the development toolkits to support SBOM creation (C5). Issuing badges might be a simple incentive that might promote the adoption of SBOMs (as it has been in other domains[18]) (C12). Other stakeholders could require their participants to provide SBOMs. For example, the scientific publication of tools and models could require that artifacts be accompanied by SBOMs (C12). At the same time, better marketing and educational materials that emphasize the importance of SBOMs are needed, both for software developers and consumers. Ultimately, creating and using SBOMs should be done because it helps to create and maintain better, more secure, and reliable software, and that ultimately benefits society.

Bibliography

[1] CycloneDX History. https://cyclonedx.org/about/history/.
[2] SPDX Overview. https://spdx.dev/about/.
[3] 2021. EXECUTIVE ORDER 14028. https://www.nist.gov/itl/executive-order-14028-improving-nations-cybersecurity.
[4] [n.d.]. Qualtrics. https://www.qualtrics.com/.
[5] Musard Balliu, Benoit Baudry, Sofia Bobadilla, Mathias Ekstedt, Martin Monperrus, Javier Ron, Aman Sharma, Gabriel Skoglund, César Soto-Valero, and Martin Wittlinger. 2023. Challenges of Producing Software Bill Of Materials for Java. arXiv preprint arXiv:2303.11102 (2023).
[6] T. Stalnaker, N. Wintersgill, O. Chaparro, M. Di Penta, D. M. German, and D. Poshyvanyk, "Boms away! inside the minds of stakeholders: A comprehensive study of bills of materials for software systems," in Proceedings of the 46th IEEE/ACM International Conference on Software Engineering, 2024, pp. 1–13.
[7] "Bill Bensing". 2022. History of the Software Bill of Material (SBOM). https://billbensing.com/software-supply-chain/history-software-bill-of-material-sbom/.
[8] Stephen Hendrick. 2022. Software Bill of Materials (SBOM) and Cybersecurity Readiness. https://tinyurl.com/293v3xte.
[9] NTIA. 2019. Roles and Benefits for SBOM Across the Supply Chain. https://ntia.gov/files/ntia/publications/ntia_sbom_use_cases_roles_benefits-nov2019.pdf.
[10] NTIA. 2021. SBOM at a Glance. https://tinyurl.com/txyvbhfu.
[11] NTIA. 2021. Sharing and Exchanging SBOMs. https://www.ntia.gov/files/ntia/publications/ntia_sbom_sharing_exchanging_sboms-10feb2021.pdf.
[12] NTIA. 2021. Software Bill of Materials Elements and Considerations. https://ntia.gov/sites/default/files/publications/uscc_-_2021.06.17_0.pdf.
[13] OpenSSF. 2022. Securing Critical Projects Workgroup: List of Projects Identified as 'Critical'. https://tinyurl.com/sxpeasey.
[14] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2022. Robust speech recognition via large-scale weak supervision. arXiv preprint arXiv:2212.04356 (2022).
[15] Neil Sheppard. 2023. SBOMs (Software Bill of Materials): Why Do They Matter? https://www.leanix.net/en/blog/sboms-matter
[16] Donna Spencer. 2009. Card sorting: Designing usable categories. Rosenfeld Media.
[17] Nathan Wintersgill Oscar Chaparro Massimilano Di Penta Daniel M German Denys Poshyvanyk Stalnaker, Trevor. 2023. Online replication package. https://github.com/TStalnaker44/boms_away_study.
[18] Asher Trockman, Shurui Zhou, Christian Kästner, and Bogdan Vasilescu. 2018. Adding sparkle to social coding: an empirical study of repository badges in the npm ecosystem. In Proceedings of the 40th International Conference on Software Engineering, ICSE 2018, Gothenburg, Sweden, May 27 - June 03, 2018, Michel Chaudron, Ivica Crnkovic, Marsha Chechik, and Mark Harman (Eds.). ACM, 511–522.
An Empirical Study on Software Bill of Materials: Where We Stand and the Road Ahead. arXiv preprint arXiv:2301.05362 (2023).
[19] Boming Xia, Tingting Bi, Zhenchang Xing, Qinghua Lu, and Liming Zhu. 2023. An Empirical Study on Software Bill of Materials: Where We Stand and the Road Ahead. arXiv preprint arXiv:2301.05362 (2023).
[20] Henry Young. [n. d.]. SBOMs: Considerable Progress, But Not Yet Ready for Codification. https://tinyurl.com/y2xzxs8m.
[21] Nusrat Zahan, Elizabeth Lin, Mahzabin Tamanna, William Enck, and Laurie Williams. 2023. Software Bills of Materials Are Required. Are We There Yet? IEEE Security & Privacy 21, 2 (2023), 82–88.