SPDX SBOMs: Enabling Automation of Safety & Security Analysis

Kate Stewart

VP Dependable Embedded Systems, The Linux Foundation

kstewart@linuxfoundation.org

THE LINUX FOUNDATION

Software is Used in Critical Systems Today


Chemical


Financial


Commercial Facilities


Food & Agriculture


Communications


Government Facilities


Critical Manufacturing


Healthcare & Public Care


Dams


Information Technology


Defense Industrial Base


Nuclear Reactors, Materials, & Waste


Emergency Services


Transportation Systems


Energy


Water & Wastewater Systems

THE LINUX FOUNDATION

Source: https://www.cisa.gov/critical-infrastructure-sectors

Critical Infrastructure Today: Mix of Open & Proprietary

98%

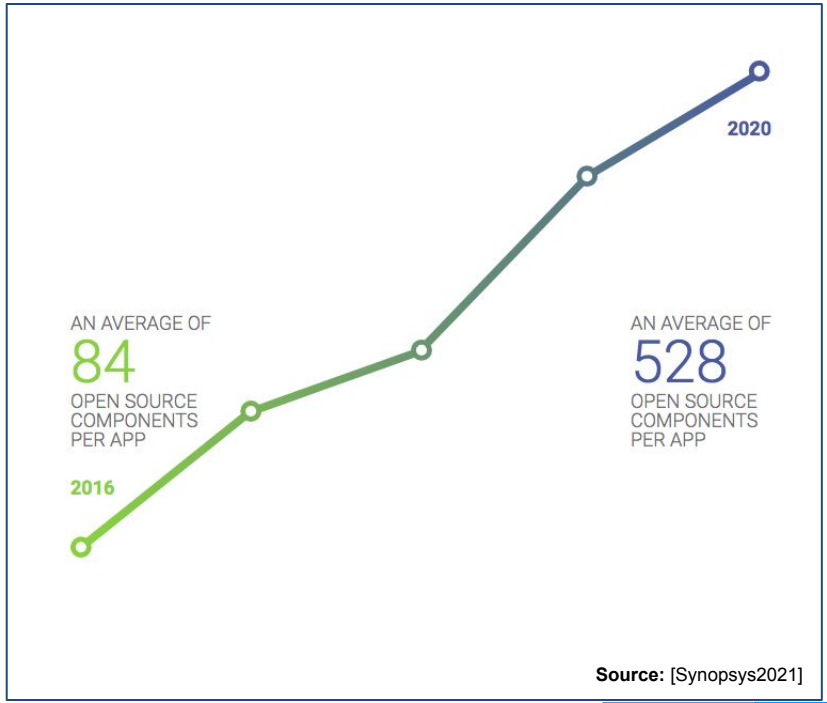
Percent of general codebases and Android apps that contained OSS

[Synopsys2021]

70%

Percent of codebase that was OSS on average

[Synopsys2020]



Source: [Synopsys2021]

Source:

[Synopsys2020] “2020 Open Source Security and Risk Analysis Report” by Synopsys
https://www.synopsys.com/content/dam/synopsys/sig-assets/reports/2020-ossra-report.pdf
[Synopsys2021] "2021 Open Source Security and Risk Analysis Report” by Synopsys
https://www.synopsys.com/software-integrity/resources/analyst-reports/open-source-security-risk-analysis.html

THE LINUX FOUNDATION

Cybersecurity & Critical Infrastructure

Critical Infrastructure

Since 2005, the 'Cybersecurity Policy for Critical Infrastructure Protection' has been set as a common action plan shared between the government, which bears responsibility for promoting independent measures by CI operators relating to CI cybersecurity and implementing other necessary measures, and CI operators which independently carry out relevant protective measures, and the new edition was published in 2022.
This document identifies the 14 sectors as critical infrastructure and it expects stakeholders to undertake the five measures as below.

1. Enhancement of Incident Response Capability
2. Maintenance and Promotion of the Safety Principles
3. Enhancement of Information Sharing System
4. Utilization of Risk Management
5. Enhancement of the Basis for CIP

2. Maintenance and promotion of the safety principles	Basically keep the element of "[1] Maintenance and promotion of the safety principles"	○ Clarify that safety standards, etc., that contribute to the enhancement of incident response capability and risk management are to be developed. ○ Consider survey methods capable of continuously improving the activities of CI operators.

The Cybersecurity Policy for Critical Infrastructure Protection
Full Text
Guideline for Establishing Safety Principles for Ensuring Information Security of Critical Infrastructure(5th Edition)(Revised on May 2019)
Risk Assessment Guide Based on the Concept of Mission Assurance in Critical Infrastructure (1st Edition)(Revised on May 2019)

source: https://www.nisc.go.jp/eng/index.html#sec4

Maintenance and Promotion of Safety Principles

Safety Standards are looking for:

- **Unique ID**, something to uniquely identify the version of the software you are using.
  - Variations in releases make it important to be able to distinguish the exact version you are using.
  - The unique ID could be as simple as using the hash from a configuration management tool, so that you know whether it has changed.
- **Dependencies of the component**
  - Any chained dependencies that a component may require.
  - Any required and provided interfaces and shared resources used by the software component. A component can add demand for system-level resources that might not be accounted for.
- The component's **build configuration** (how it was built so that it can be duplicated in the future) and sources
- Any **existing bugs and their workarounds**
- **Documentation** for application manual for the component
  - The **intended use** of the software component
  - **Instructions** on how to **integrate** the software component correctly and **invoke it properly**
- **Requirements** for the software component
  - This should include the results of any testing to demonstrate requirements coverage
  - Coverage for nominal operating conditions and behavior in the case of failure
  - For highly safety critical requirements, test coverage should be in accordance with what the specification expects (e.g., Modified Condition/Decision Coverage (MC/DC) level code coverage)
  - Any safety requirements that might be violated if the included software performs incorrectly. This is specifically looking for failures in the included software that can cause the safety function to perform incorrectly. (This is referred to as a cascading failure.)
  - What the software might do under anomalous operating conditions (e.g., low memory or low available CPU)

Source: https://www.linux.com/featured/sboms-supporting-safety-critical-software/

Maintenance and Promotion of Safety Principles

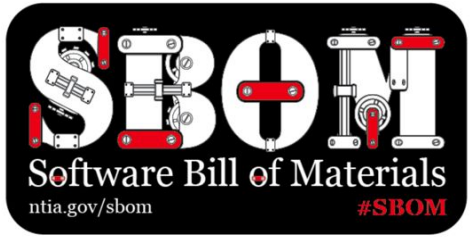
Requirements are needed to know you're "**done**" after applying a patch:

- Need to be able to ensure you have compliance to the updated system requirements after applying a patch
- Given the rate of change and vulnerabilities, we need a way to make this automated, so it needs to be machine readable
- For each file patched, what requirements does it interact with, what tests need to be rerun to regenerate the evidence

Software Bill of Materials (SBOMs) today:

- Machine readable - Identities & Dependencies are part of the minimum definition
- SPDX SBOMs can also enables recording and connecting the sources, assessments, vulnerabilities & patches, build & calibration data, tests, requirements and evidence ⇒ path to automation

Common Understanding of “SBOM”



“An SBOM is a formal record containing the details and supply chain **relationships** of various **components** used in **building software.**

These components, including libraries and modules, can be open source or proprietary, free or paid, and the data can be widely available or access-restricted.”

Source: NTIA’s SBOM FAQ

CYBERSECURITY & INFRASTRUCTURE SECURITY AGENCY

THE LINUX FOUNDATION

NTIA SBOM Guidance

Minimum Elements	
Data Fields	Document baseline information about each component that should be tracked: Supplier, Component Name, Version of the Component, Other Unique Identifiers, Dependency Relationship, Author of SBOM Data, and Timestamp.
Automation Support	Support automation, including via automatic generation and machine-readability to allow for scaling across the software ecosystem. Data formats used to generate and consume SBOMs include SPDX, CycloneDX, and SWID tags.
Practices and Processes	Define the operations of SBOM requests, generation and use including: Frequency, Depth, Known Unknowns, Distribution and Delivery, Access Control, and Accommodation of Mistakes.

Source:https://www.ntia.gov/files/ntia/publications/sbom_minimum_elements_report.pdf.

NTIA Software Bill Of Materials (SBOM) Guidance - Minimum Elements

Data Field	Description
Supplier Name	The name of an entity that creates, defines, and identifies components.
Component Name	Designation assigned to a unit of software defined by the original supplier.
Version of the Component	Identifier used by the supplier to specify a change in software from a previously identified version.
Other Unique Identifiers	Other identifiers that are used to identify a component, or serve as a look-up key for relevant databases.
Dependency Relationship	Characterizing the relationship that an upstream component X is included in software Y.
Author of SBOM Data	The name of the entity that creates the SBOM data for this component.
Timestamp	Record of the date and time of the SBOM data assembly.

SPDX 2.2 + (ISO/IEC 5962:2021) **supports all required minimum elements** (as well the optional that are mentioned in report)

Checker available at: https://github.com/spdx/ntia-conformance-checker

Source: https://www.ntia.gov/files/ntia/publications/sbom_minimum_elements_report.pdf

When should an SBOM be created or consumed?

Safety and Security expect that Configuration Management (CM) information will be **maintained throughout the software lifecycle**.

SBOMs provide a mechanism to **track key artifacts and dependencies** as well as other useful CM information

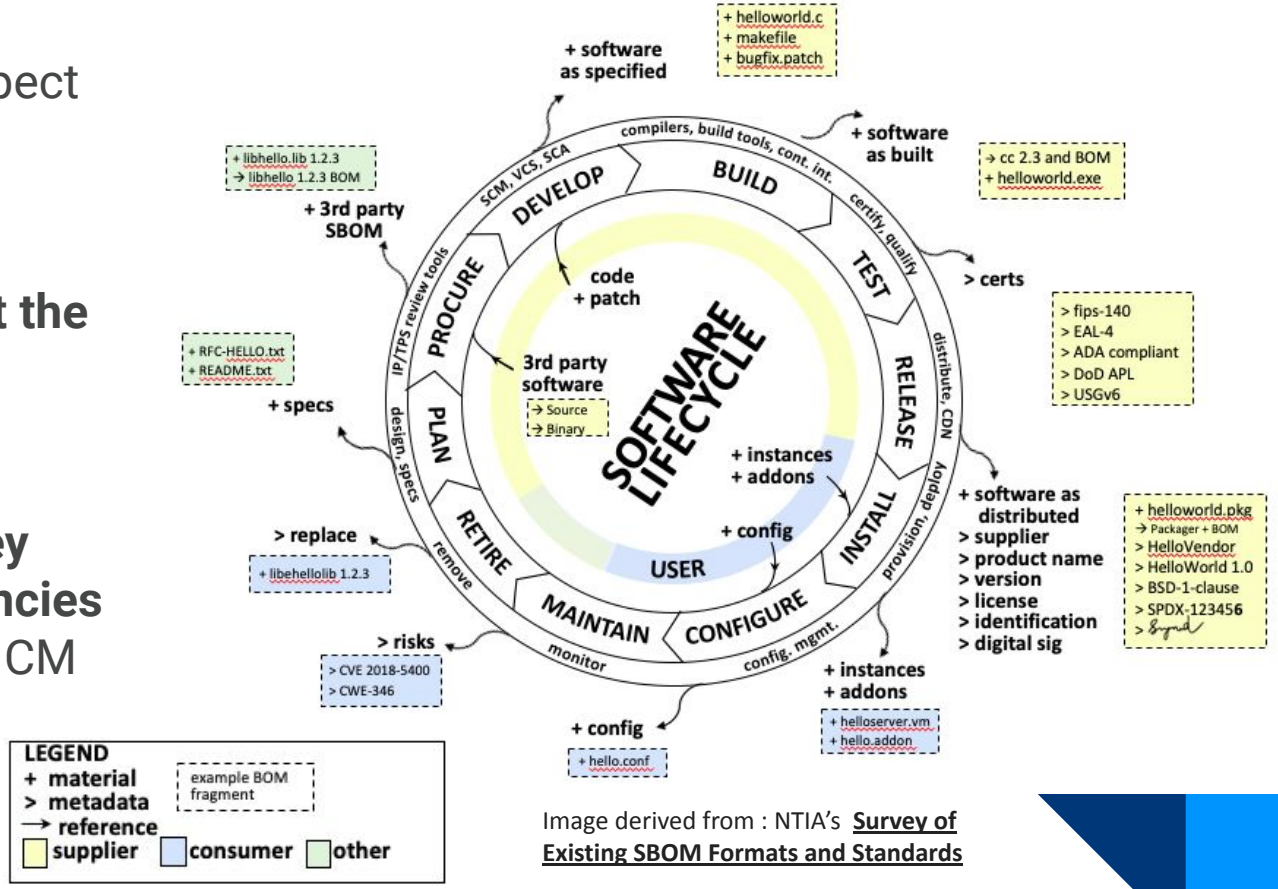


Image derived from : NTIA's **Survey of Existing SBOM Formats and Standards**

THE LINUX FOUNDATION

SBOM Types

SBOM TYPE	DEFINITION
Design	SBOM of intended, planned software project or product with included components (some of which may not yet exist) for a new software artifact.
Source	SBOM created directly from the development environment, source files, and included dependencies used to build an product artifact.
Build	SBOM generated as part of the process of building the software to create a releasable artifact (e.g., executable or package) from data such as source files, dependencies, built components, build process ephemeral data, and other SBOMs.
Deployed	SBOM provides an inventory of software that is present on a system. This may be an assembly of other SBOMs that combines analysis of configuration options, and examination of execution behavior in a (potentially simulated) deployment environment.
Runtime	BOM generated through instrumenting the system running the software, to capture only components present in the system, as well as external call-outs or dynamically loaded components. In some contexts, this may also be referred to as an “Instrumented” or “Dynamic” SBOM.
Analyzed	SBOM generated through analysis of artifacts (e.g., executables, packages, containers, and virtual machine images) after its build. Such analysis generally requires a variety of heuristics. In some contexts, this may also be referred to as a “3rd party” SBOM.

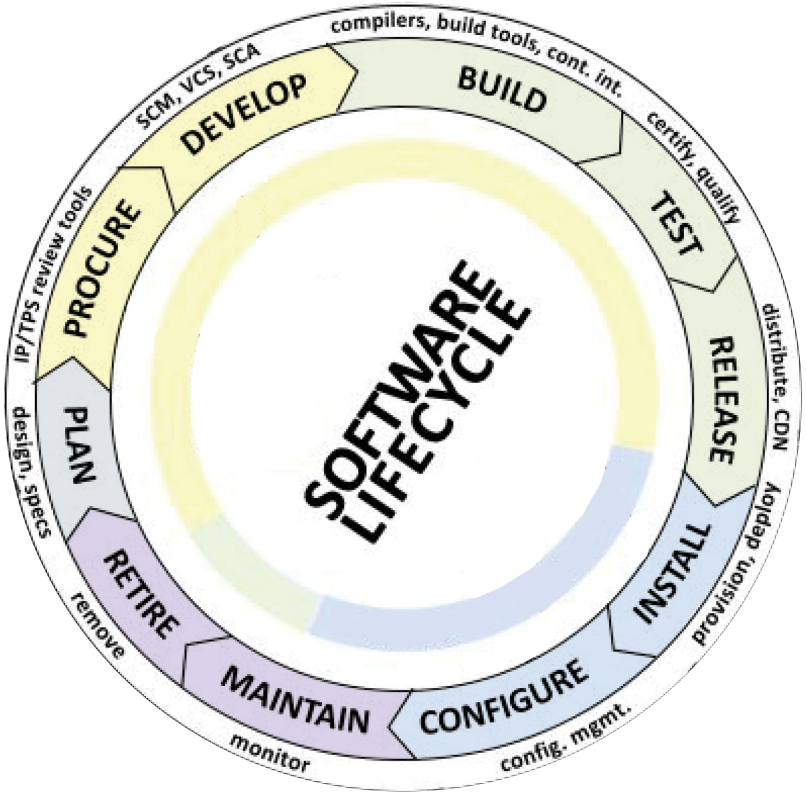
Source: Types of Software Bills of Materials (SBOM) published by CISA on 2023/4/21

KEY: Generate SBOMs **when** the data is available

Source SBOM

Design SBOM

Runtime SBOM

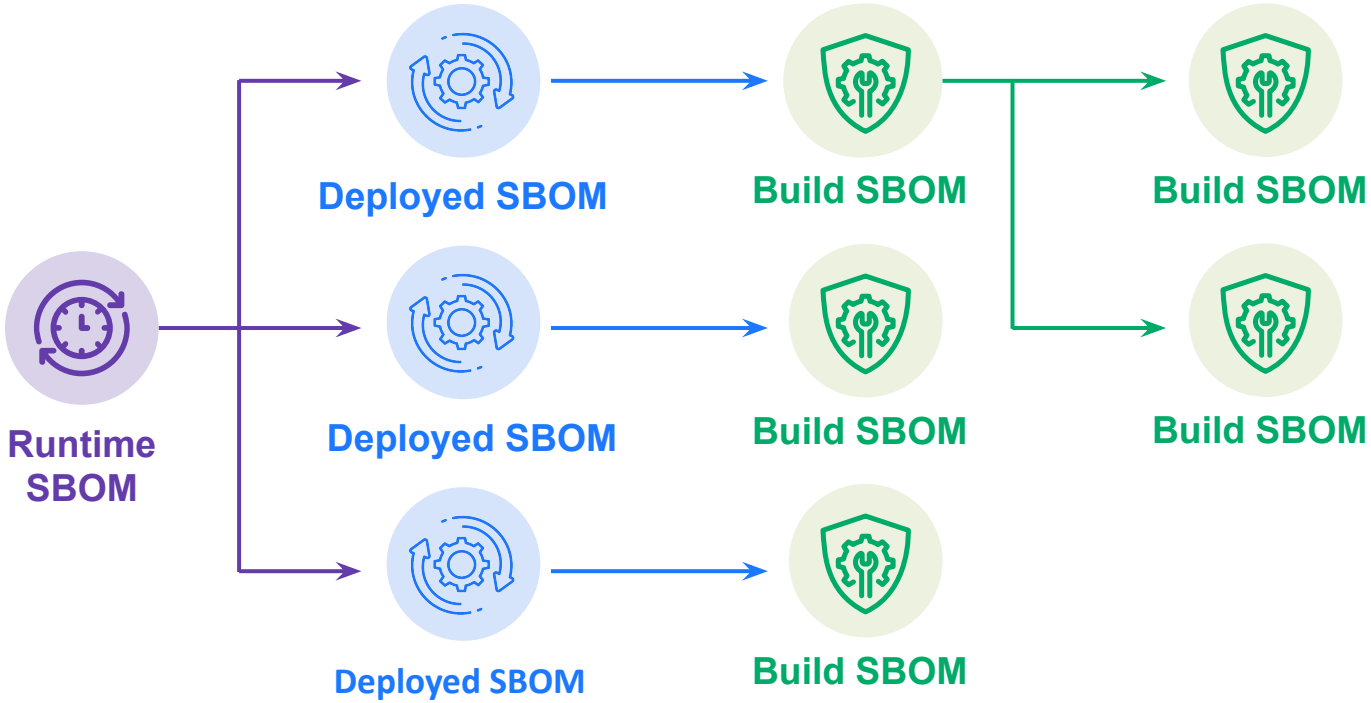


Build SBOM

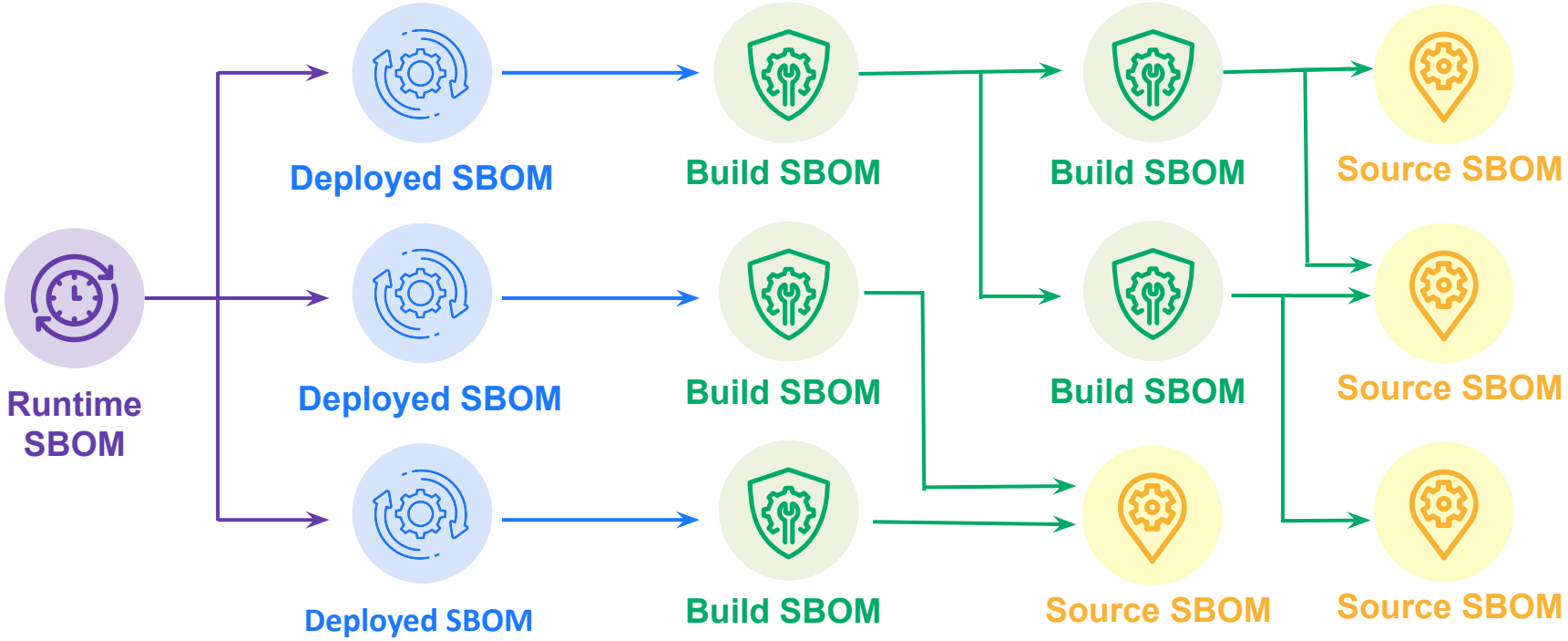
Deployed SBOM

THE LINUX FOUNDATION

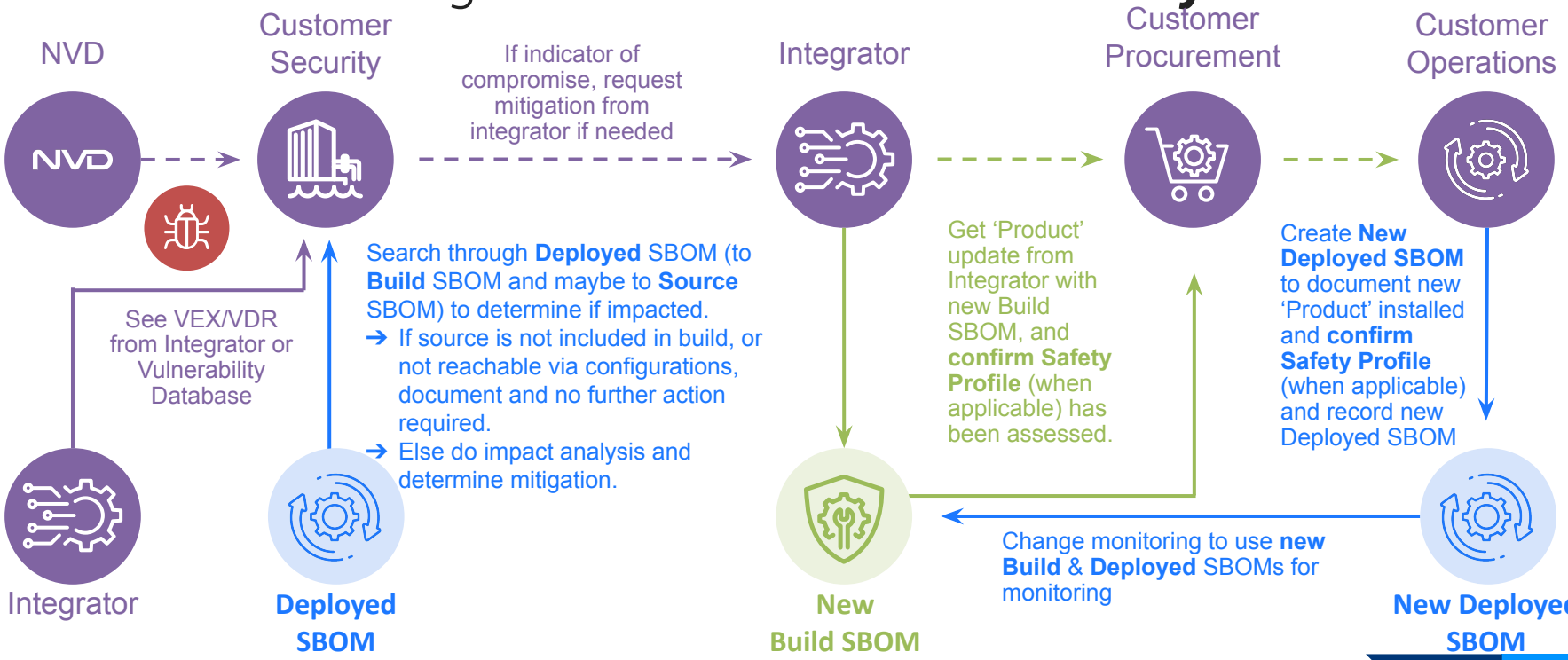
Understanding System: Traceability



Understanding Safety Critical System: Traceability

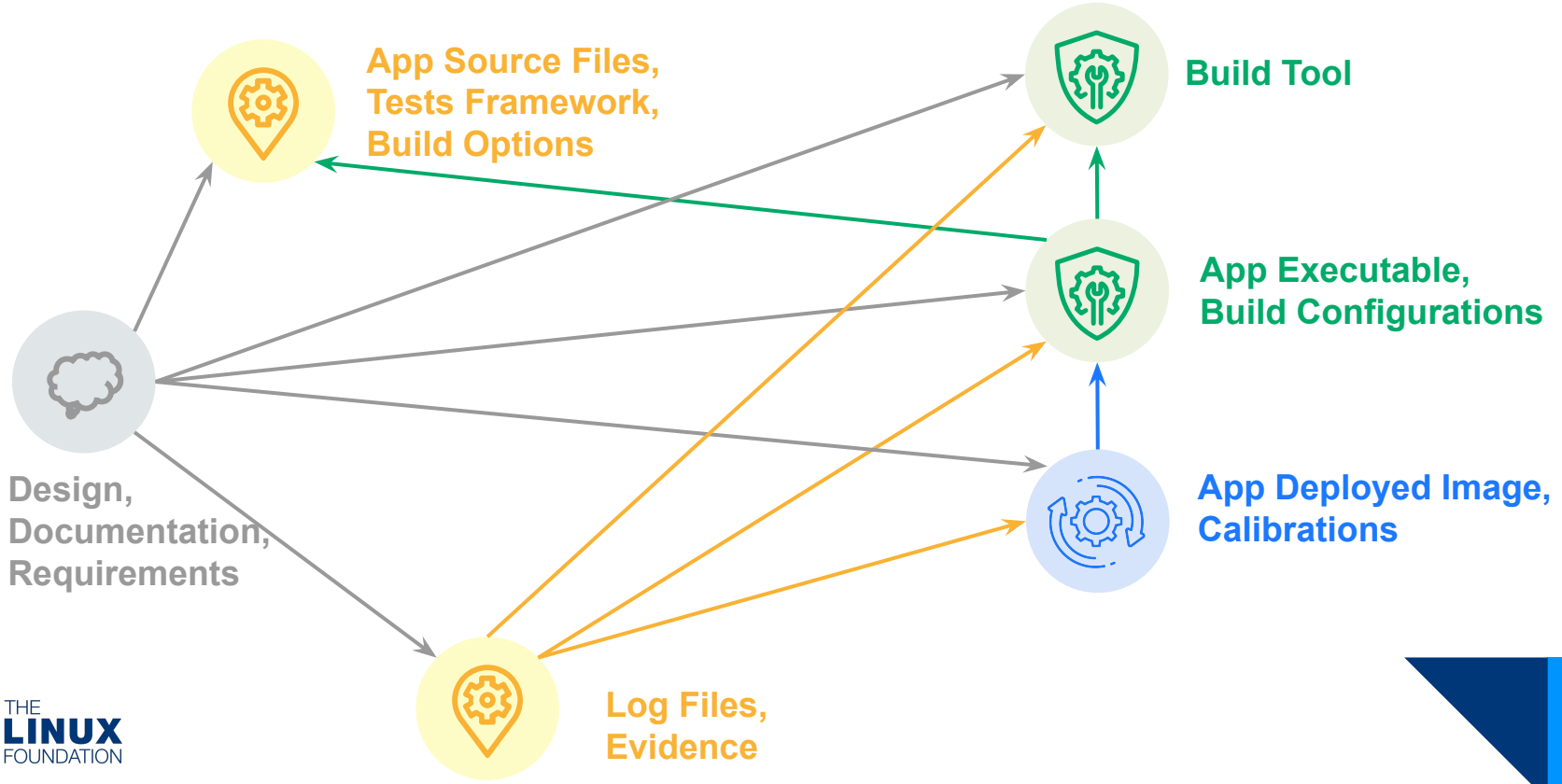


Managing a security fix: Customer & Integrator need to **check Safety Profile**

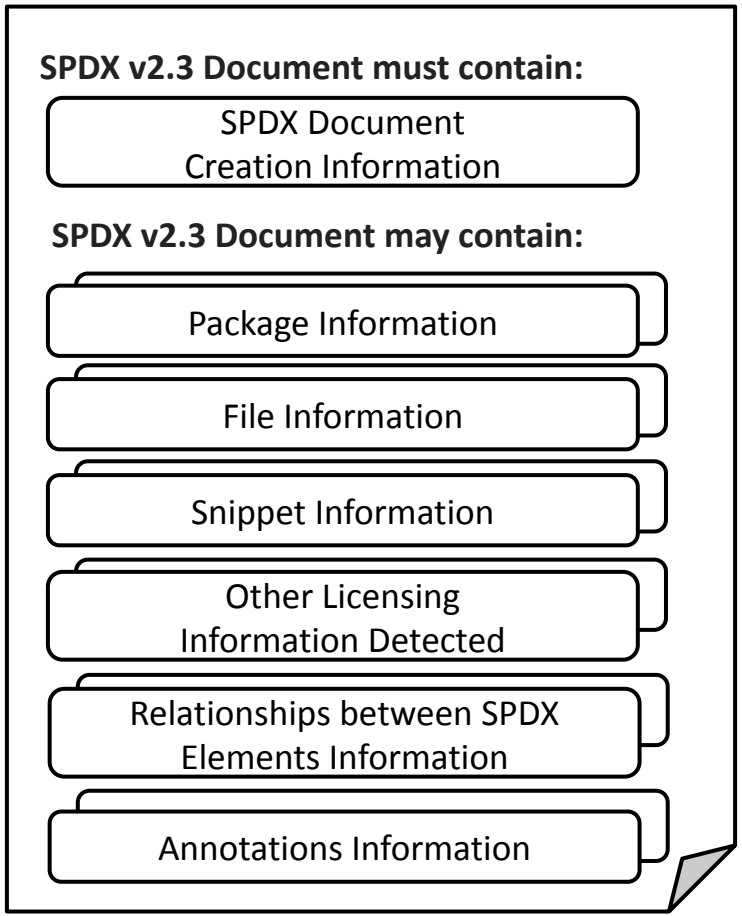


THE LINUX FOUNDATION

SPDX SBOM's Enable Linking: Requirements to Code to Tests to Evidence



SPDX

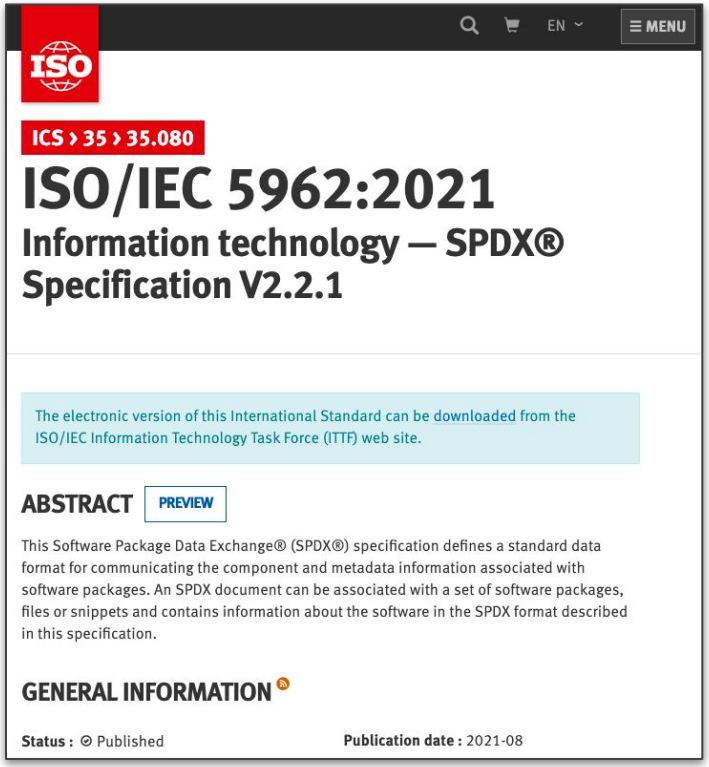


Software Package Data Exchange (SPDX®) specification is a standard for communicating the component and metadata information associated with software

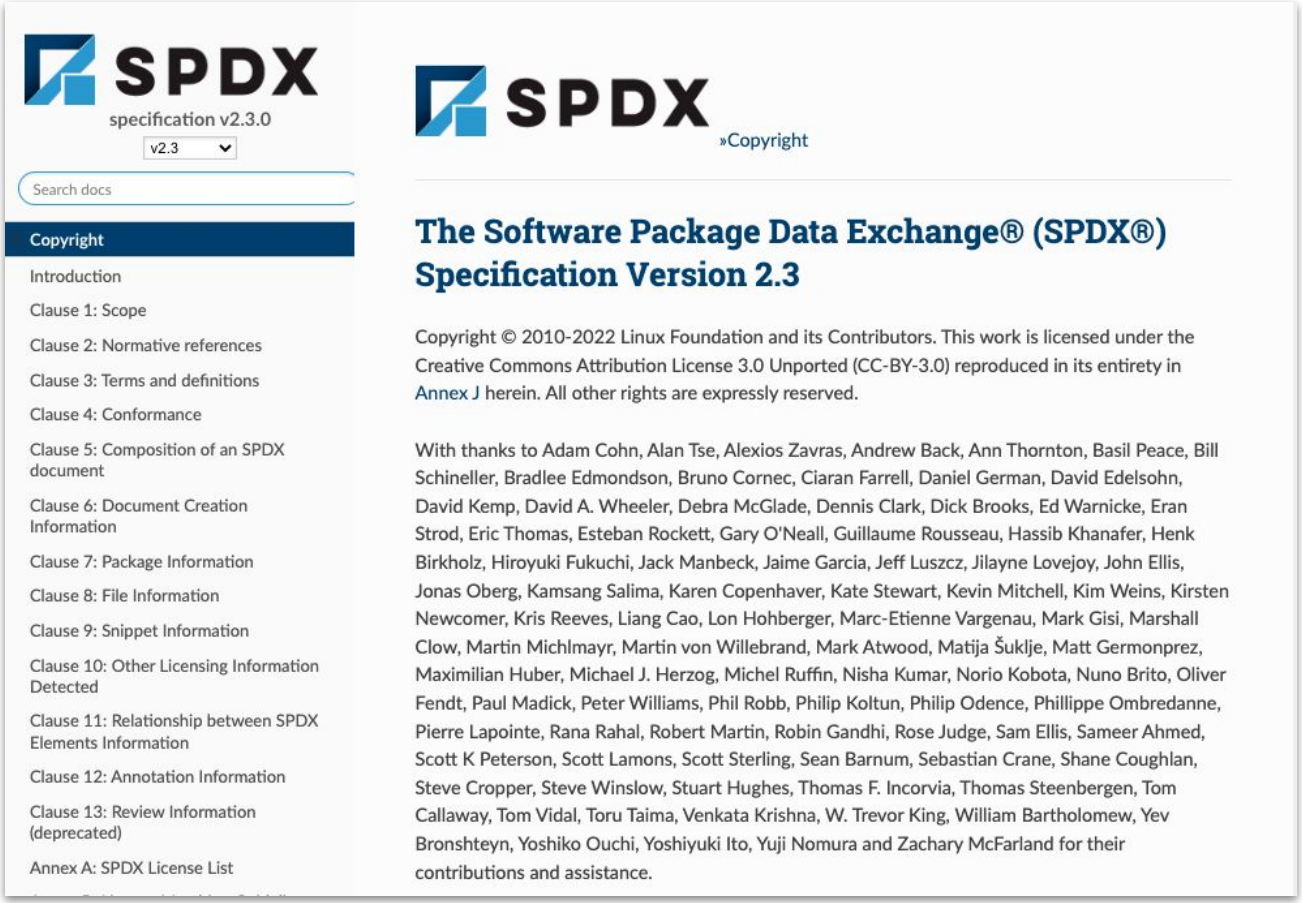
Charter: To create a set of **data exchange** standards that enable companies and organizations to share human-readable and machine-processable **software package metadata** to facilitate **software supply chain processes**.

ISO/IEC 5962:2021

- Able to represent SBOMs from binary images and track back to the source files and snippets.
- Specification is freely available from ITTF site
- Future updates are live tracked at: https://spdx.github.io/spdx-spec and work on satisfying safety requirements is being included
- More information at spdx.dev



Source: https://www.iso.org/standard/81870.html accessed on 2021/11/19



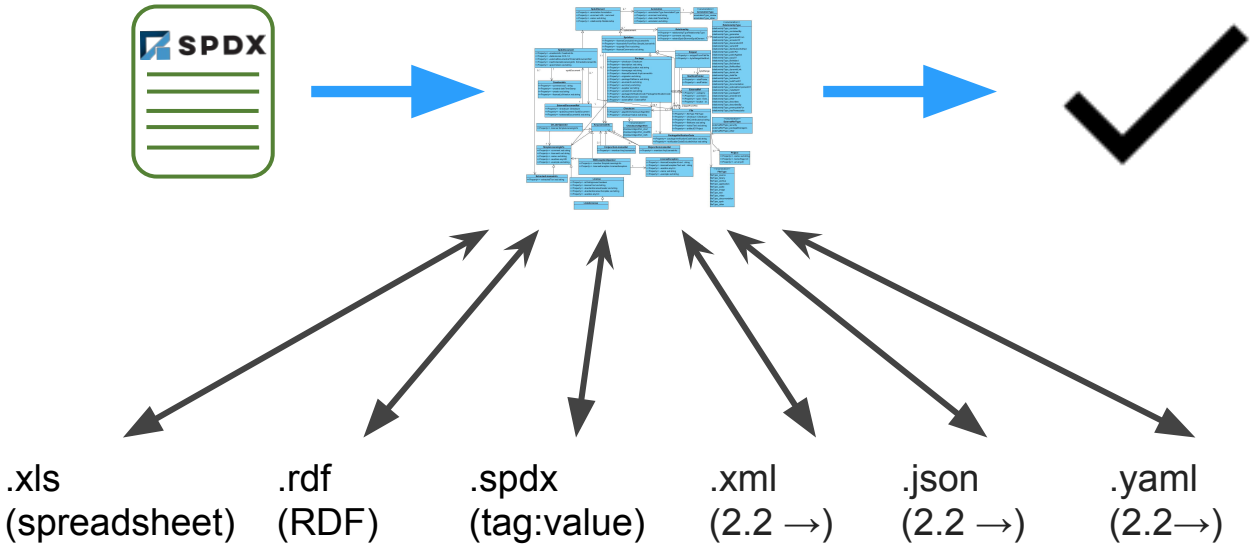
source: https://spdx.github.io/spdx-spec/v2.3/

SPDX Continuously Improves

- **2010/02** - specification drafting began in a work-group of FOSSBazaar under Linux Foundation that came to be called "SPDX", was originally referred to as Package Facts.
- **2010/08** - "SPDX" announced as one of the pillars of the Linux Foundation's Open Compliance Program.
- **2011/08** - SPDX 1.0 specification - handles packages.
- **2012/08** - SPDX 1.1 specification - fixed flaw in verification algorithm
- **2013/10** - SPDX 1.2 specification - improved interaction with license list, additional fields for documenting project info.
- **2015/05** - SPDX 2.0 specification - added ability to handle multiple packages, relationships between packages and files, annotations.
- **2016/08** - SPDX 2.1 specification - added snippets, support for associating packages with external reference sources of information about packages, using SPDX License identifiers in files
- **2019/06** - SPDX 2.1.1 - conversion of specification from google docs to github as repository
- **2020/05** - SPDX 2.2 - Includes SPDX-lite
- **2020/08** - SPDX 2.2.1 prepared for submission to ISO.
- **2021/08** - ISO/IEC 5962 available
- **2022/08** - SPDX 2.3 published to improve interoperability with other formats
- **2023/Q2** - SPDX 3.0 release candidate and prototyping in progress …

SPDX

Formal Model Enables Validation & Interchange Between Specific File Formats



SPDX

SPDX Relationships Clarify Dependency Types

DESCRIBES	DEPENDENCY_OF	PREREQUISITE_FOR	GENERATES	VARIANT_OF
DESCRIBED_BY	RUNTIME_DEPENDENCY_OF	HAS_PREREQUISITE	TEST_OF	FILE_ADDED
CONTAINS	BUILD_DEPENDENCY_OF	ANCESTOR_OF	TEST_TOOL_OF	FILE_DELETED
CONTAINED_BY	DEV_DEPENDENCY_OF	DESCENDENT_OF	TEST_CASE_OF	FILE_MODIFIED
DYNAMIC_LINK	OPTIONAL_DEPENDENCY_OF	DOCUMENTATION_OF	EXAMPLE_OF	PATCH_FOR
STATIC_LINK	PROVIDED_DEPENDENCY_OF	BUILD_TOOL_OF	METAFILE_OF	PATCH_APPLIED
AMENDS	TEST_DEPENDENCY_OF	EXPANDED_FROM_ARCHIVE	PACKAGE_OF	REQUIREMENT_FOR
COPY_OF	OPTIONAL_COMPONENT_OF	DISTRIBUTION_ARTIFACT	DATA_FILE_OF	SPECIFICATION_FOR
DEPENDS_ON	DEPENDENCY_MANIFEST_OF	GENERATED_FROM	DEV_TOOL_OF	OTHER

For more details see: https://spdx.github.io/spdx-spec/v2.3/relationships-between-SPDX-elements/

SPDX

SPDX Generation Tooling

- tools-java
  - Aug 12, 2022 - v1.1.0 update to support 2.3
  - Sept 2022 → Feb 2023 5 releases for performance improvements & fixes
- tools-python
  - OpenSSF Funded cleanup & restructuring
  - Dec 8, 2022 - v0.7.0 - update to support 2.3 & clean up bug backlog
  - Prototyping of 3.0 in progress
- tools-golang
  - Jan 12, 2023 - v0.4.0 - update to support 2.3
- spdx-online-tools (validator & translator)
  - Aug 12, 2022 - v1.0.7 add support SPDX 2.3
  - Nov 15, 2022 - v1.0.9 add in NTIA Conformance Checker Tool

SPDX Consumption Tooling

- spdx-online-tools (validator & translator)
  - Aug 12, 2022 - v1.0.7 add support SPDX 2.3
  - Nov 15, 2022 - v1.0.9 add in NTIA Conformance Checker Tool
- SPDX-to-OSV (vulnerability lookup)
  - Produce an Open Source Vulnerability JSON file based on information in an SPDX document
  - Jan 10, 2022 - v0.1.1 - pick up tooing updates
- ntia-conformance-checker (minimum SBOM fields present)
  - Check that an SBOM meets the minimum field requirements
  - Started as GSOC project, and maintainer from Chainguard has adopted
  - Feb 8, 2022 - v0.2.1 - fix NOASSERTION supplier case

Also worth looking at: https://github.com/nyph-infosec/daggerboard

SBOMs Everywhere in 2022...

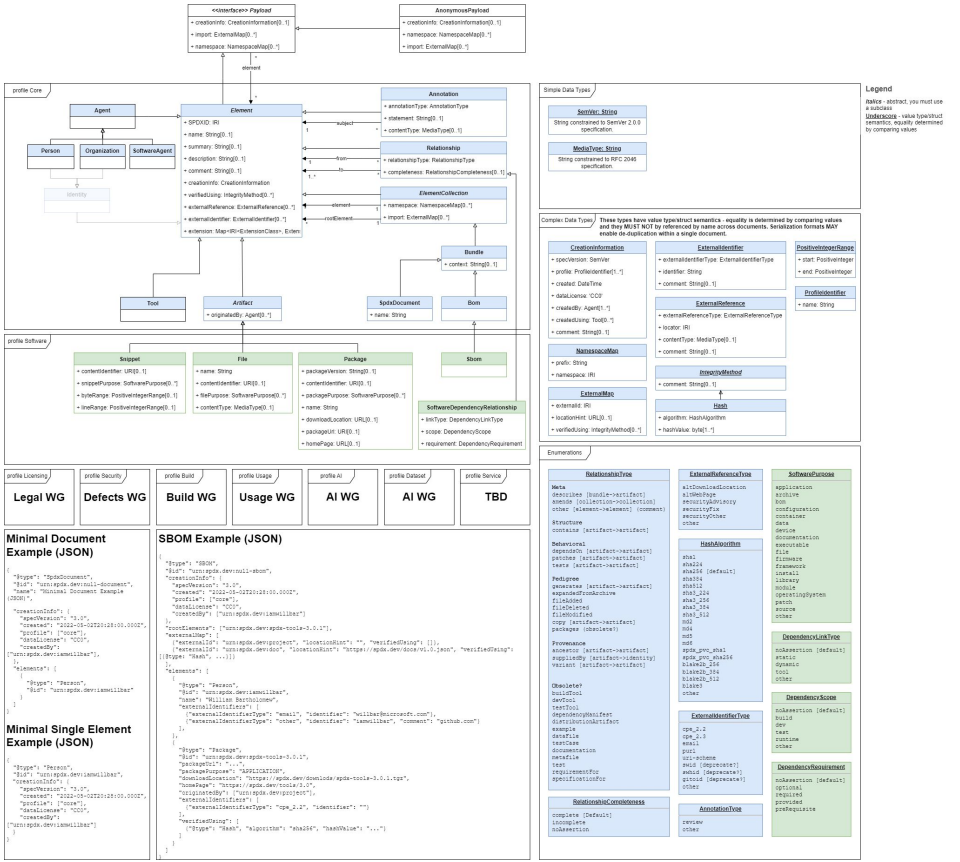
- OpenSSF - Work Stream 9 → **SBOM Everywhere SIG**
  - SPDX Python Library rework funded.
    - Test suite and release candidate available now
  - Started work on consolidating definitions of types of SBOMs → CISA working group to get broader adoption
  - Started documentation of use cases
- NTIA efforts have transferred to US DHS CISA
  - 4 working groups
    - SBOM Sharing, SBOM Adoption, SBOM Cloud, SBOM Tooling
  - International coordination with CERTs (like Japan's CERT) and other international government agencies

SPDX 3.0-rc1

Why the Changes for SPDX 3.0?

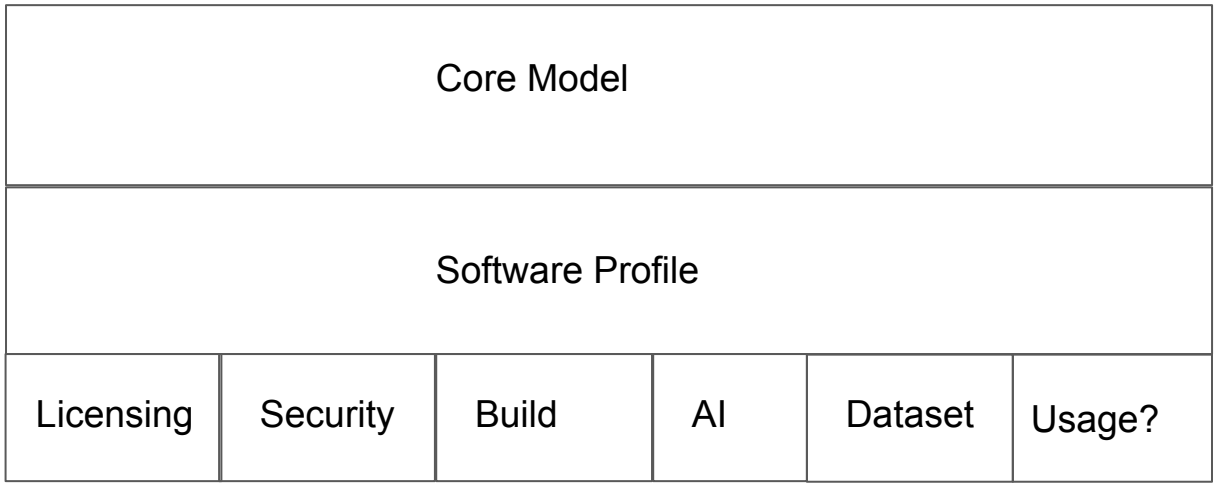
- Additional Use Cases
  - AI and Data
  - Security and Defect information
- Simplify
  - Profiles
  - Remove confusing names
- Flexibility
  - Can communicate a single Element
  - Enhanced relationship structure with less relationship types (work in progress)

SPDX 3.0 Model



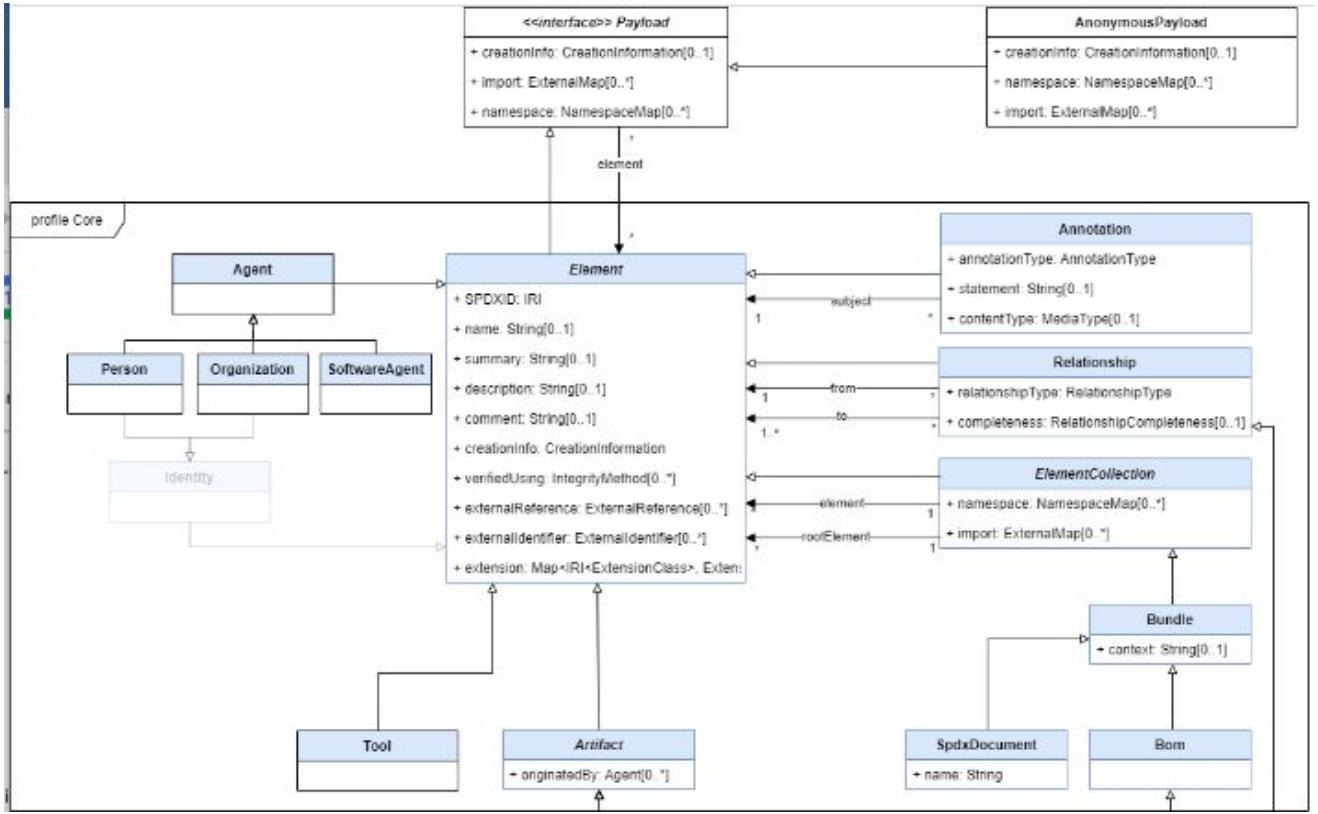
Source:https://github.com/spdx/spdx-3-model/blob/main/model.png

SPDX 3.0 - Increases Modularity

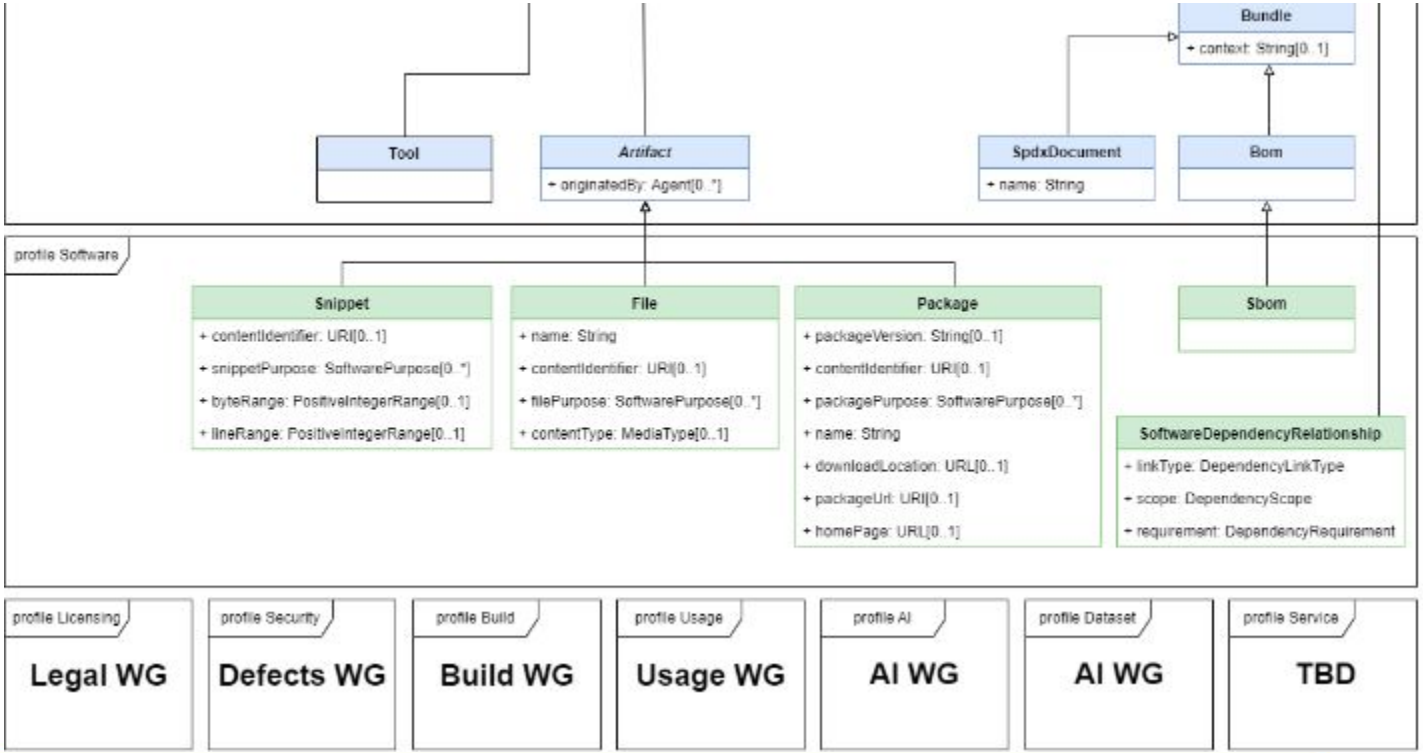


SPDX 3.0 (Core Model + Software Profile + Licensing Profile) == SPDX 2.3

SPDX 3.0 Core Model Permits Extensions



SPDX 3.0 Software Profile



SPDX 3.0 Specification Infrastructure

Specification is being transformed into markdown describing

- Classes, Properties, Enumerations
- Metadata (type & cardinality) and description for each element.
- Will be able to automatically generate schema from this version (for JSON, YAML, RDF, XML, tag-value, etc.) and reduce errors.

Profiles can add their own Classes and Properties and may also restrict other profiles (e.g. values, cardinalities, ...)

See: https://github.com/spdx/spdx-3-model

SPDX

Licensing Profile Update

- Based on licensing-related fields in pre-existing SPDX spec, with updates:
  - More consistency across artifact types (package, file, snippet)
  - Aligning with SPDX 3.0 data model
  - Documenting the object model for license expressions

- Current status:
  - Fields were discussed in joint tech/legal calls in late 2020 and early 2021
  - Initial starting point draft shared in Mar. 2021
  - Revised earlier draft for new SPDX 3.0 model formatting
  - Initial draft included in main at: https://github.com/spdx/spdx-3-model/tree/main/model/Licensing

For more information, contact: Steve Winslow or Alexios Zavras

SPDX

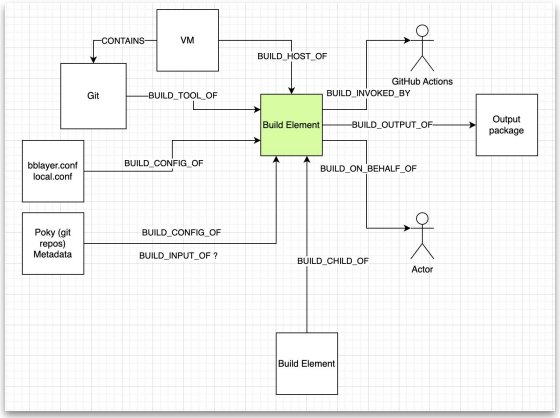
Security Profile Update

- Communicating vulnerabilities in software
  - Associate vulnerabilities with specific elements, like packages
  - Conveying vulnerability assessment (severity, impact, exploitability)
- Linking to external security information
  - *securityAdvisory*
    - Advisories and miscellaneous security related document
    - Common Security Advisory Framework (CSAF)
    - CycloneDX formatted security information
    - Open Source Vulnerability (OSV) document
    - Vulnerability Disclosure Report (VDR per NIST EO 14028)
  - *securityFix*
    - Code fix or patch for a security issue
  - *securityOther*
    - Any unspecified type of security information
- VEX support to assert status of a vulnerability
- Details at: https://github.com/spdx/spdx-3-model/tree/main/model/Security

For more information contact: Thomas Steenbergen, Jeff Schutt or Rose Judge

SPDX

Build Profile Overview



Use Cases:

- Security
- Reproducibility
- Auditing quality/pedigree of build
- Safety
  - The source code at the time of release
  - The configuration used to build the software
  - The specific versions of the tools used to build the software

Producers: Build systems, secondarily, analysis tools

Initial draft has been submitted to model for discussion at :
https://github.com/spdx/spdx-3-model/tree/main/model/Build

Contact: Brandon Lumm or Nisha Kumar

SPDX

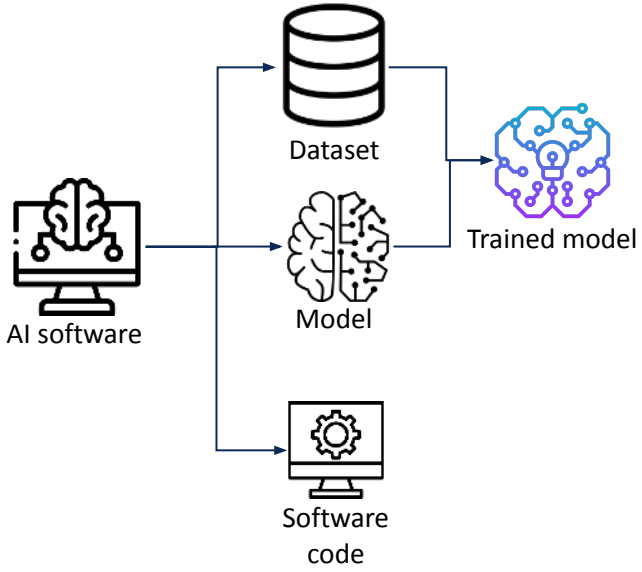
AI BOM ⇒ SPDX AI profile + SPDX Dataset profile

Traditional Software

Software code

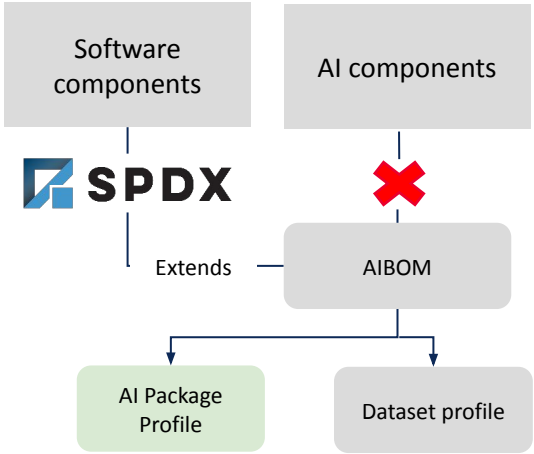
AI Software

Has additional elements that nuances that need to be captured to ensure its traceability



AI BOM

Enhances SPDX to describe AI software including and AI Software's components, licenses, copyrights, and security references.



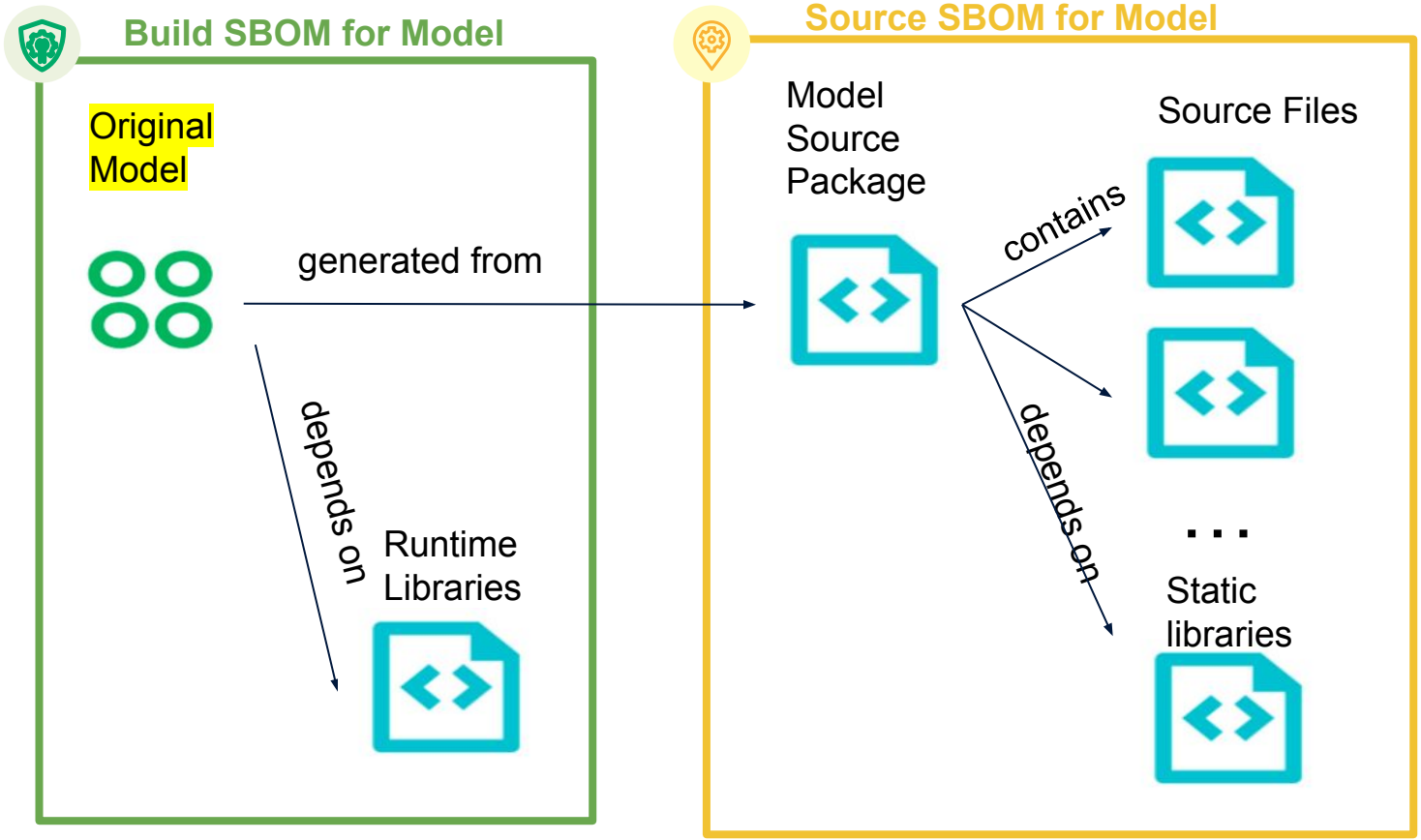
SPDX

Components of AI Application

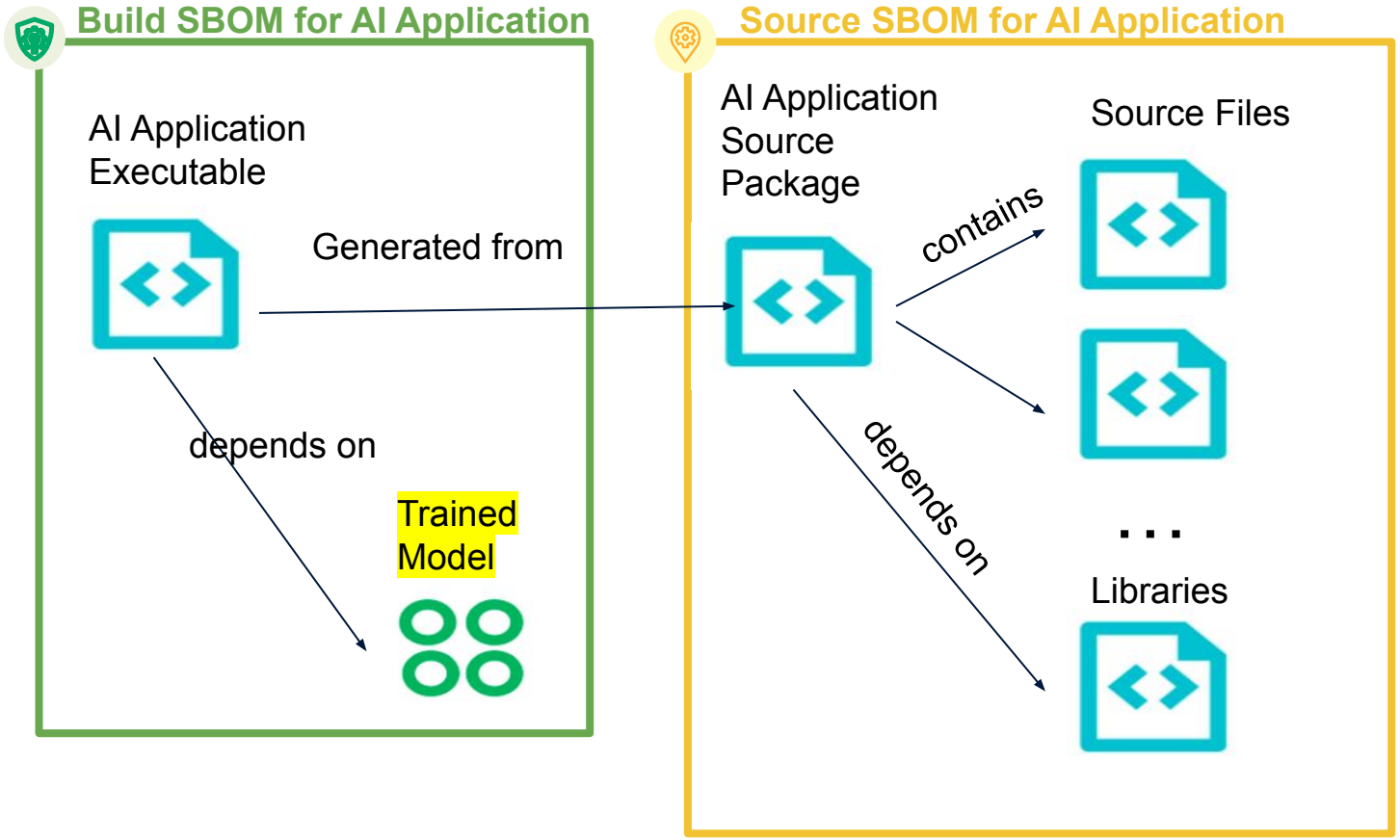
Data	Model	Code
Schema	Algorithms	Business Needs
Sampling over Time	More Training	Bug Fixes
Volume	Experiments	Configuration

Source: IBM Data Science Best Practices
(https://ibm.github.io/data-science-best-practices/versioning.html)

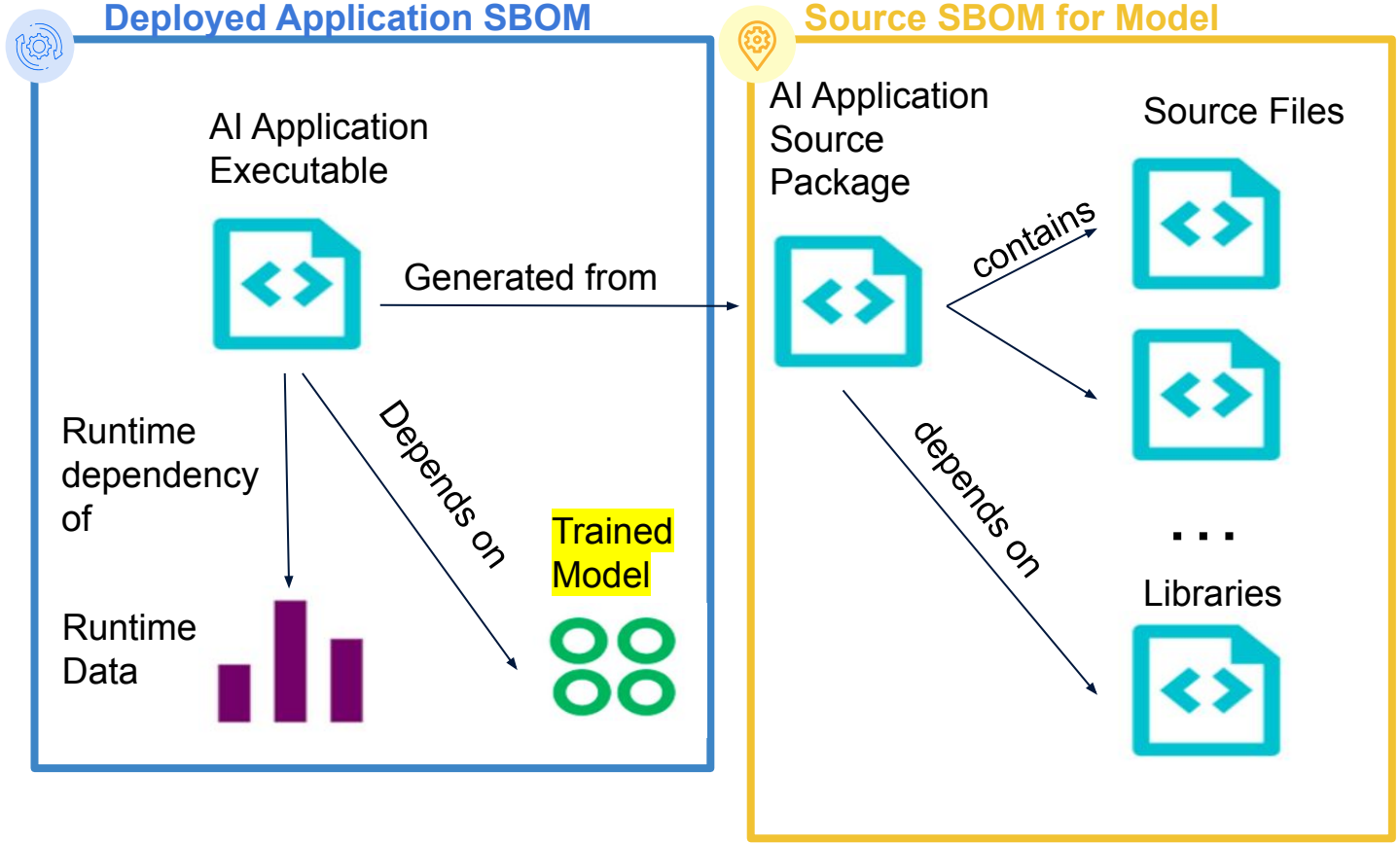
Build Up Original Model



Building the AI Application



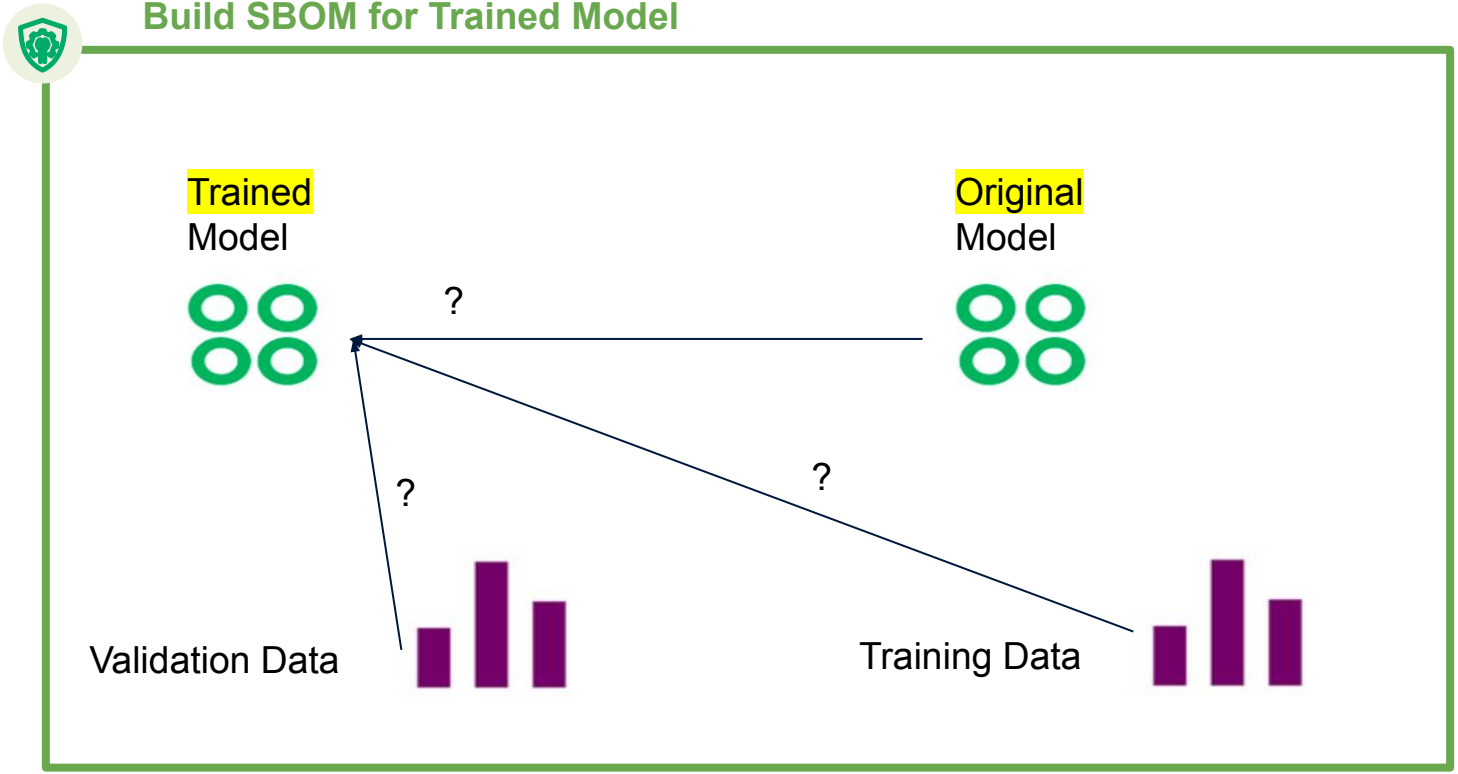
Deploying the AI Application



SPDX can represent most of this today.

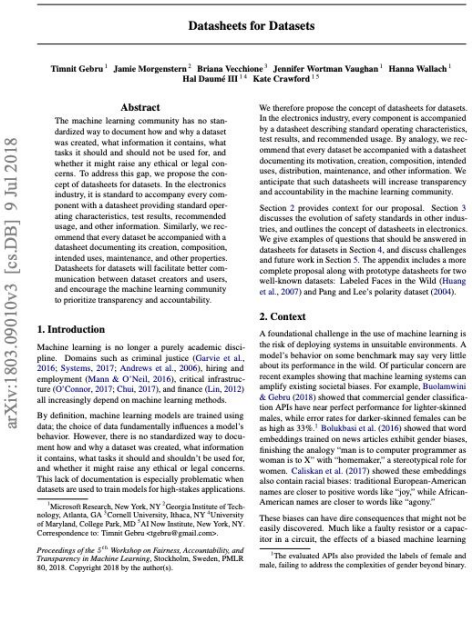
But what about the trained model?

GAP: Representing Training the model



AI BOM Transparency Survey: Fields Required?

Datasheets



arXiv:1803.09010v3 [cs.DB] 9 Jul 2018

Datasets for Datasets

Timnit Gebru [1] Jamie Morgenstern [2] Briana Vecchione [3] Jennifer Wortman Vaughan [1] Hanna Wallach [1]
Hal Daumé III [1,4] Kate Crawford [1,5]

Abstract

The machine learning community has no standardized way to document how and why a dataset was created, what information it contains, what tasks it should and should not be used for, and whether it might raise any ethical or legal concerns. To address this gap, we propose the concept of datasheets for datasets. In the electronics industry, it is standard to accompany every component with a datasheet providing standard operating characteristics, test results, recommended usage, and other information. Similarly, we recommend that every dataset be accompanied with a datasheet documenting its creation, composition, intended uses, maintenance, and other properties. Datasheets for datasets will facilitate better communication between dataset creators and users, and encourage the machine learning community to prioritize transparency and accountability.

1. Introduction

Machine learning is no longer a purely academic discipline. Domains such as criminal justice (Garvie et al., 2016; Systems, 2017; Andrews et al., 2006), hiring and employment (Mann & O'Neil, 2016), critical infrastructure (O'Connor, 2017; Chui, 2017), and finance (Lin, 2012) all increasingly depend on machine learning methods.

By definition, machine learning models are trained using data; the choice of data fundamentally influences a model's behavior. However, there is no standardized way to document how and why a dataset was created, what information it contains, what tasks it should and shouldn't be used for, and whether it might raise any ethical or legal concerns. This lack of documentation is especially problematic when datasets are used to train models for high-stakes applications.

We therefore propose the concept of datasheets for datasets. In the electronics industry, every component is accompanied by a datasheet describing standard operating characteristics, test results, and recommended usage. By analogy, we recommend that every dataset be accompanied with a datasheet documenting its motivation, creation, composition, intended uses, distribution, maintenance, and other information. We anticipate that such datasheets will increase transparency and accountability in the machine learning community.

Section 2 provides context for our proposal. Section 3 discusses the evolution of safety standards in other industries, and outlines the concept of datasheets in electronics. We give examples of questions that should be answered in datasheets for datasets in Section 4, and discuss challenges and future work in Section 5. The appendix includes a more complete proposal along with prototype datasheets for two well-known datasets: Labeled Faces in the Wild (Huang et al., 2007) and Pang and Lee's polarity dataset (2004).

2. Context

A foundational challenge in the use of machine learning is the risk of deploying systems in unsuitable environments. A model's behavior on some benchmark may say very little about its performance in the wild. Of particular concern are recent examples showing that machine learning systems can amplify existing societal biases. For example, Buolamwini & Gebru (2018) showed that commercial gender classification APIs have near perfect performance for lighter-skinned males, while error rates for darker-skinned females can be as high as 33%.[1] Bolukbasi et al. (2016) showed that word embeddings trained on news articles exhibit gender biases, finishing the analogy "man is to computer programmer as woman is to X" with "homemaker," a stereotypical role for women. Caliskan et al. (2017) showed these embeddings also contain racial biases: traditional European-American names are closer to positive words like "joy," while African-American names are closer to words like "agony."

These biases can have dire consequences that might not be easily discovered. Much like a faulty resistor or a capacitor in a circuit, the effects of a biased machine learning

[1]Microsoft Research, New York, NY [2]Georgia Institute of Technology, Atlanta, GA [3]Cornell University, Ithaca, NY [4]University of Maryland, College Park, MD [5]AI Now Institute, New York, NY.
Correspondence to: Timnit Gebru <tgebru@gmail.com>.

Proceedings of the 5[th] Workshop on Fairness, Accountability, and Transparency in Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

[1]The evaluated APIs also provided the labels of female and male, failing to address the complexities of gender beyond binary.

Source:https://arxiv.org/pdf/1810.03993.pdf

Model Cards

arXiv:1810.03993v2 [cs.LG] 14 Jan 2019

Model Cards for Model Reporting

Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, Timnit Gebru
{mmitchellai,simonewu,andrewzaldivar,parkerbarnes,lucyvasserman,benhutch,espitzer,tgebru}@google.com
deborah.raji@mail.utoronto.ca

ABSTRACT

Trained machine learning models are increasingly used to perform high-impact tasks in areas such as law enforcement, medicine, education, and employment. In order to clarify the intended use cases of machine learning models and minimize their usage in contexts for which they are not well suited, we recommend that released models be accompanied by documentation detailing their performance characteristics. In this paper, we propose a framework that we call model cards, to encourage such transparent model reporting. Model cards are short documents accompanying trained machine learning models that provide benchmarked evaluation in a variety of conditions, such as across different cultural, demographic, or phenotypic groups (e.g., race, geographic location, sex, Fitzpatrick skin type [15]) and intersectional groups (e.g., age and race, or sex and Fitzpatrick skin type) that are relevant to the intended application domains. Model cards also disclose the context in which models are intended to be used, details of the performance evaluation procedures, and other relevant information. While we focus primarily on human-centered machine learning models in the application fields of computer vision and natural language processing, this framework can be used to document any trained machine learning model. To solidify the concept, we provide cards for two supervised models: One trained to detect smiling faces in images, and one trained to detect toxic comments in text. We propose model cards as a step towards the responsible democratization of machine learning and related artificial intelligence technology, increasing transparency into how well artificial intelligence technology works. We hope this work encourages those releasing trained machine learning models to accompany model releases with similar detailed evaluation numbers and other relevant documentation.

CCS CONCEPTS

• **General and reference** → **Evaluation;** • **Social and professional topics** → *User characteristics*; • **Software and its engineering** → **Use cases**; *Documentation*; *Software evolution*; • **Human-centered computing** → *Walkthrough evaluations*;

KEYWORDS

datasheets, model cards, documentation, disaggregated evaluation, fairness evaluation, ML model evaluation, ethical considerations

ACM Reference Format:

Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, Timnit Gebru. 2019. Model Cards for Model Reporting. In *FAT* '19: Conference on Fairness, Accountability, and Transparency, January 29–31, 2019, Atlanta, GA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3287560.3287596

1 INTRODUCTION

Currently, there are no standardized documentation procedures to communicate the performance characteristics of trained machine learning (ML) and artificial intelligence (AI) models. This lack of documentation is especially problematic when models are used in applications that have serious impacts on people's lives, such as in health care [14, 42, 44], employment [1, 13, 29], education [23, 45] and law enforcement [2, 7, 20, 34].

Researchers have discovered systematic biases in commercial machine learning models used for face detection and tracking [4, 9, 49], attribute detection [5], criminal justice [10], toxic comment detection [11], and other applications. However, these systematic errors were only exposed after models were put into use, and negatively affected users reported their experiences. For example, after MIT Media Lab graduate student Joy Buolamwini found that commercial face recognition systems failed to detect her face [4], she collaborated with other researchers to demonstrate the disproportionate errors of computer vision systems on historically marginalized groups in the United States, such as darker-skinned women [5, 41]. In spite of the potential negative effects of such reported biases, documentation accompanying trained machine learning models (if supplied) provides very little information regarding model performance characteristics, intended use cases, potential pitfalls, or other information to help users evaluate the suitability of these systems to their context. This highlights the need to have detailed documentation accompanying trained machine learning models, including metrics that capture bias, fairness and inclusion considerations.

As a step towards this goal, we propose that released machine learning models be accompanied by short (one to two page) records we call model cards. Model cards (for model reporting) are complements to "Datasheets for Datasets" [21] and similar recently proposed documentation paradigms [3, 28] that report details of the datasets used to train and test machine learning models. Model cards are also similar to the FDA.gov statement proposal in medicine [25]. We provide two example model cards in Section 5: A smiling detection model trained on the CelebA dataset [36] (Figure 2), and a public toxicity detection model [32] (Figure 3). Where Datasheets highlight characteristics of the data feeding into the model, we

FAT '19, January 29–31, 2019, Atlanta, GA, USA*

ACM ISBN 978-1-4503-6125-5/19/01
https://doi.org/10.1145/3287560.3287596

Source:https://www.microsoft.com/en-us/research/uploads/prod/2019/01/1803.09010.pdf

FactSheets

arXiv:1808.07261v2 [cs.CY] 7 Feb 2019

FactSheets: Increasing Trust in AI Services through Supplier's Declarations of Conformity

M. Arnold,[1] R. K. E. Bellamy,[1] M. Hind,[1] S. Houde,[1] S. Mehta,[2] A. Mojsilović,[1] R. Nair,[1] K. Natesan Ramamurthy,[1] D. Reimer,[1] A. Olteanu,* D. Piorkowski,[1] J. Tsay,[1] and K. R. Varshney[1]
IBM Research
[1]Yorktown Heights, New York, [2]Bengaluru, Karnataka

Abstract

Accuracy is an important concern for suppliers of artificial intelligence (AI) services, but considerations beyond accuracy, such as safety (which includes fairness and explainability), security, and provenance, are also critical elements to engender consumers' trust in a service. Many industries use transparent, standardized, but often not legally required documents called supplier's declarations of conformity (SDoCs) to describe the lineage of a product along with the safety and performance testing it has undergone. SDoCs may be considered multi-dimensional fact sheets that capture and quantify various aspects of the product and its development to make it worthy of consumers' trust. Inspired by this practice, we propose FactSheets to help increase trust in AI services. We envision such documents to contain purpose, performance, safety, security, and provenance information to be completed by AI service providers for examination by consumers. We suggest a comprehensive set of declaration items tailored to AI and provide examples for two fictitious AI services in the appendix of the paper.

1 Introduction

Artificial intelligence (AI) services, such as those containing predictive models trained through machine learning, are increasingly key pieces of products and decision-making workflows. A service is a function or application accessed by a customer via a cloud infrastructure, typically by means of an application programming interface (API). For example, an AI service could take an audio waveform as input and return a transcript of what was spoken as output, with all complexity hidden from the user, all computation done in the cloud, and all models used to produce the output pre-trained by the supplier of the service. A second more complex example would provide an audio waveform translated into a different language as output. The second example illustrates that a service can be made up of many different models (speech recognition, language translation, possibly sentiment or tone analysis, and speech synthesis) and is thus a distinct concept from a single pre-trained machine learning model or library.

In many different application domains today, AI services are achieving impressive accuracy. In certain areas, high accuracy alone may be sufficient, but deployments of AI in high-stakes decisions, such as credit applications, judicial decisions, and medical recommendations, require greater trust in AI services. Although there is no scholarly consensus on the specific traits that imbue trustworthiness in people or algorithms [1, 2], fairness, explainability, general safety, security, and transparency are some of the issues that have raised public concern about trusting AI and threatened the further adoption of AI beyond low-stakes uses [3, 4]. Despite active research and development to address these issues, there is no mechanism yet for the creator of an AI service to communicate how they are addressed in a deployed version. This is a major impediment to broad AI adoption.

Toward transparency for developing trust, we propose a *FactSheet* for AI Services. A FactSheet will contain sections on all relevant attributes of an AI service, such as intended use, performance, safety, and security. Performance will include appropriate accuracy or risk measures along with timing information. Safety, discussed in [5, 3] as the minimiza-

*A. Olteanu's work was done while at IBM Research. Author is currently affiliated with Microsoft Research.

Source: https://arxiv.org/pdf/1808.07261.pdf

AI Profile Properties

AI/ML Application

Model

Data

Required Fields:

- **Creator**
- **Supplier**
- **PackageVersionInfo**
- **DownloadLocation**
- **PackageDescription**
- **LicenseConcluded**
- **LicenseDeclared**
- **ReleaseTime**

Optional Fields:

- **Originator**
- **Checksum**
- **ValidUntilTime**
- **BuildTime**
- **PackageComments**
- *SensitivePersonalInformation*
- *EnergyConsumption*
- *StandardsCompliance*
- *InformationAboutTraining*
- *Hyperparameters*
- *SafetyRiskAssessment*
- *DataPreprocessingSteps*
- *ModelExplainabilityMechanisms*
- *MetricsDecisionThresholds*
- *Metrics*
- *Autonomy*
- *Domain*
- *Limitations*
- *Type*

Dataset Profile Properties

AI/ML Application

Model

Data

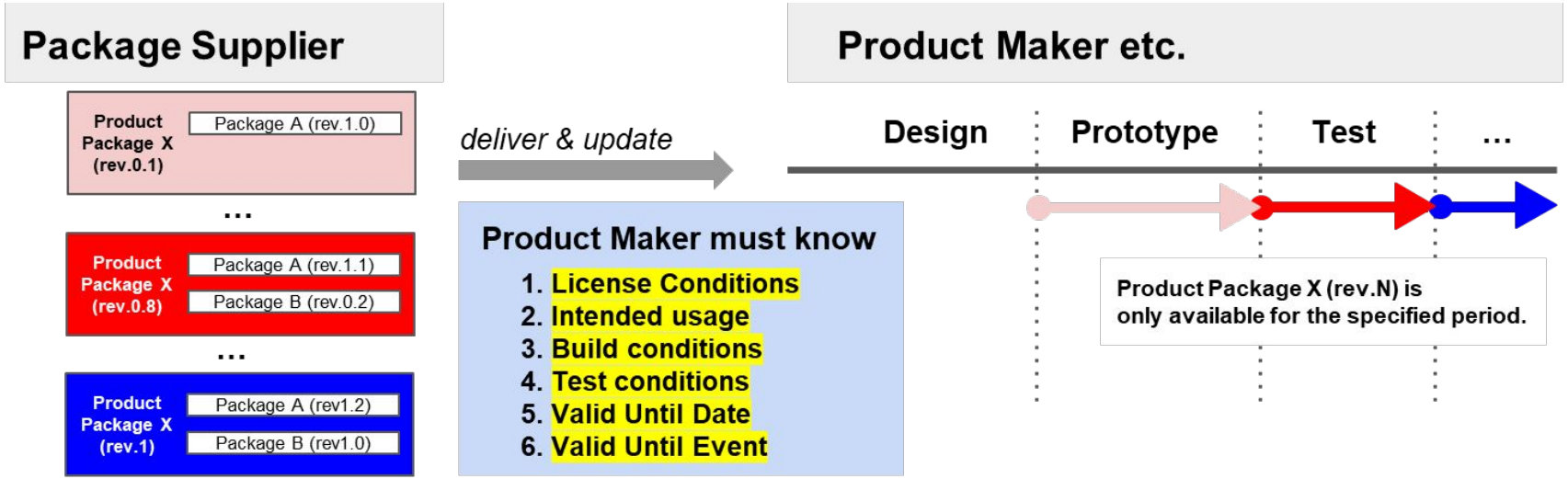
Required Fields:

- **Name**
- **Originator**
- **DownloadLocation**
- **LicenseConcluded**
- **LicenseDeclared**
- **PackageDescription**
- **BuiltTime**
- **ReleaseTime**
- *DatasetType*

Optional Fields:

- **Supplier**
- **VersionInfo**
- **Checksum**
- **ValidUntilTime**
- *IntendedUse*
- *DatasetCollectionProcess*
- *DatasetUpdateMechanism*
- *DatasetSize*
- *DatasetNoise*
- *KnownBias*
- *Errata*
- *SensorsUsed*
- *StandardCompliance*
- *SensitivePersonalInformation*
- *ConfidentialityLevel*
- *AnonymizationMethodUsed*

Usage Profile: to tell intentions as "**Usage**" for Delivery Product



Both parties must control the Terms and Conditions of using of the Delivery

SPDX

What’s next after After 3.0?

Future Direction: Hardware

- Safety Standards expect to know “system” that software is running on
- Vulnerabilities come from interaction between hardware and software (ie.heartbleed)
- Potential participants:
  - RISC-V & ARM core adopters,
  - Chips Alliance Members
  - Board Manufacturers
- For more information, contact: Kate Stewart

Future Direction:
Safety Standards Automation

- Safety Standards expect to know
  - The source code at the time of production release
  - The documentation associated with the code
  - The configuration used to build the production software
  - The specific versions of the tools used to build the software
- Safety Standards Configuration Management (CM) Requirements are greatly simplified by following an effective SBOM process.
  - An SBOM supports **capturing the details** of what is in a specific release and supports determining what went wrong if a failure occurs.
  - The goal is to be able to **rebuild exactly** what the executable or binary was at the time of release.
- To learn more, see:
  - https://www.linux.com/featured/sboms-supporting-safety-critical-software/

Leverage SPDX Relationships to Support Safety Analysis

Using SPDX Relationship Information

AlektoMetis ...we enable digital innovation.

Assumption: process to create and maintain all artefacts (requirements, architecture, tests, analysis report) is accepted and applied

Still the biggest pain: Keeping a complete and consistent set of documentation and verifying that the evidences are complete and consistent

SPDX style solution: Create SPDX Relationships between all documentation artefacts to track all possible system combinations!

SPDX for product documentation

AlektoMetis ...we enable digital innovation.

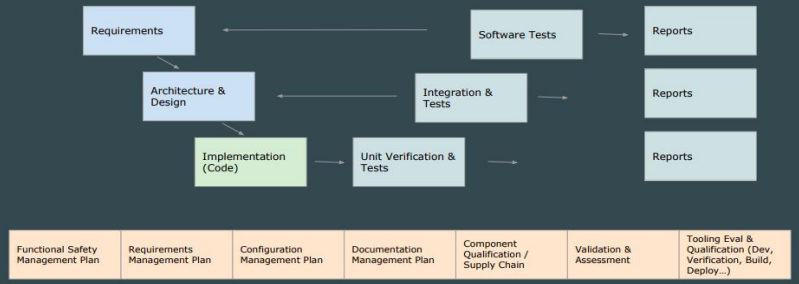
What kind of product documentation do we need to manage?

Specifications, Reports, Tests...

Safety Requirement Specification	a SPECIFICATION for functional requirements, architectural elements etc.
Unit Test	the TEST_CASE related to code or a specification artefact
Unit Test Report	DOCUMENTATION of a unit test EVIDENCE all tests have been performed as planned
Code	usually is GENERATED from or according to some specification artefact
Coding Guidelines	SPECIFICATION about the project specific details for the code

V-Model style documentation model

AlektoMetis ...we enable digital innovation.

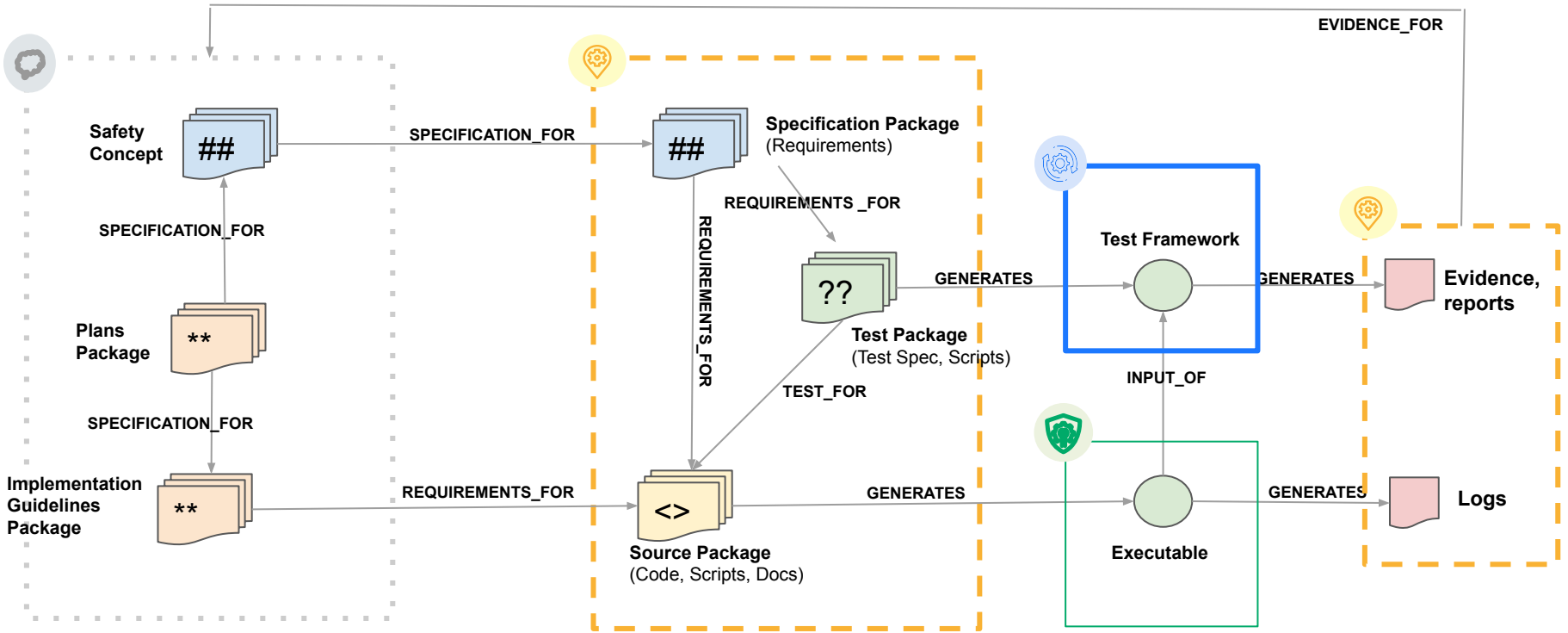


Source:https://fosdem.org/2023/schedule/event/sbom_fusa/

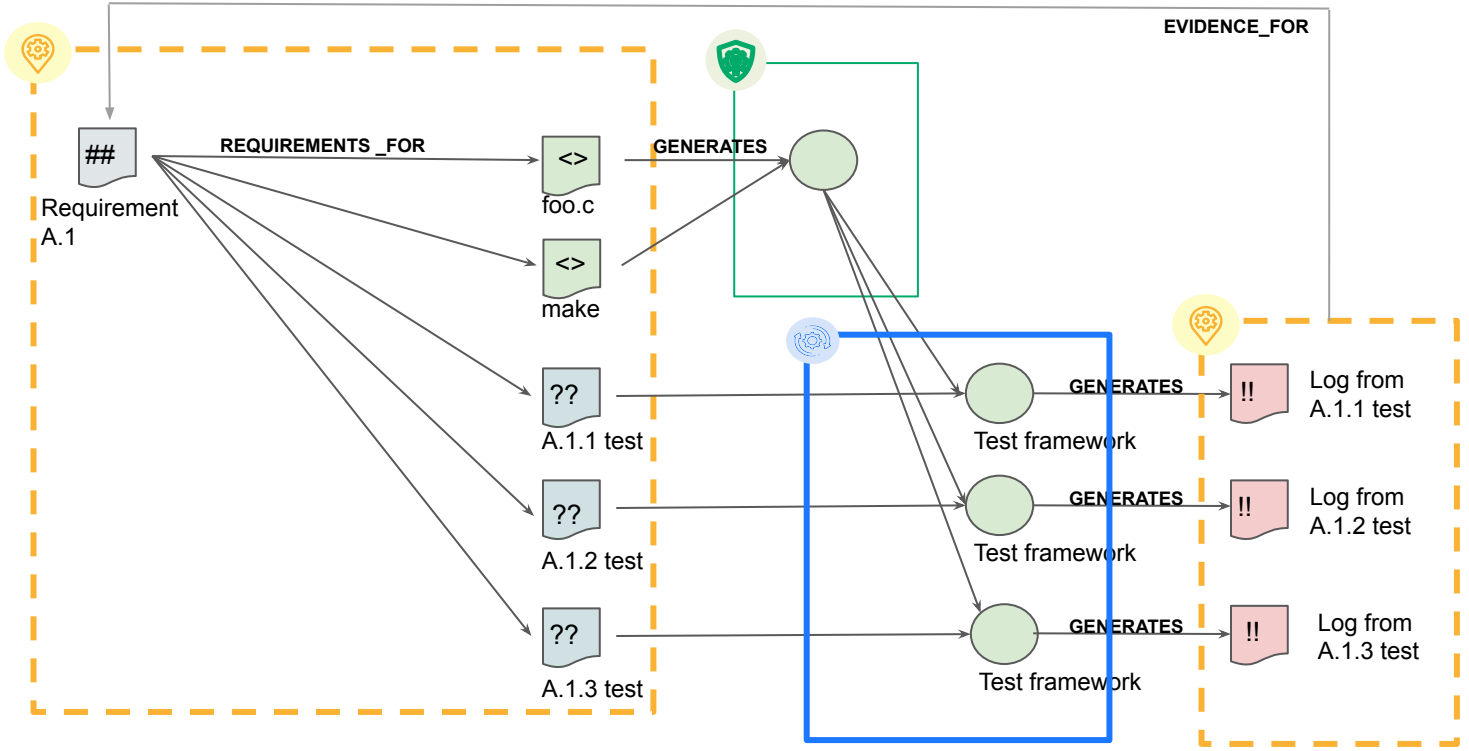
SPDX

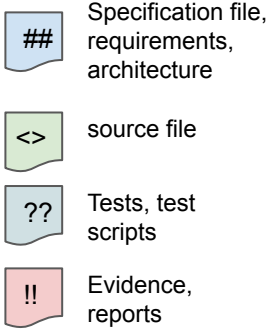
THE LINUX FOUNDATION

Leveraging SPDX Relationships

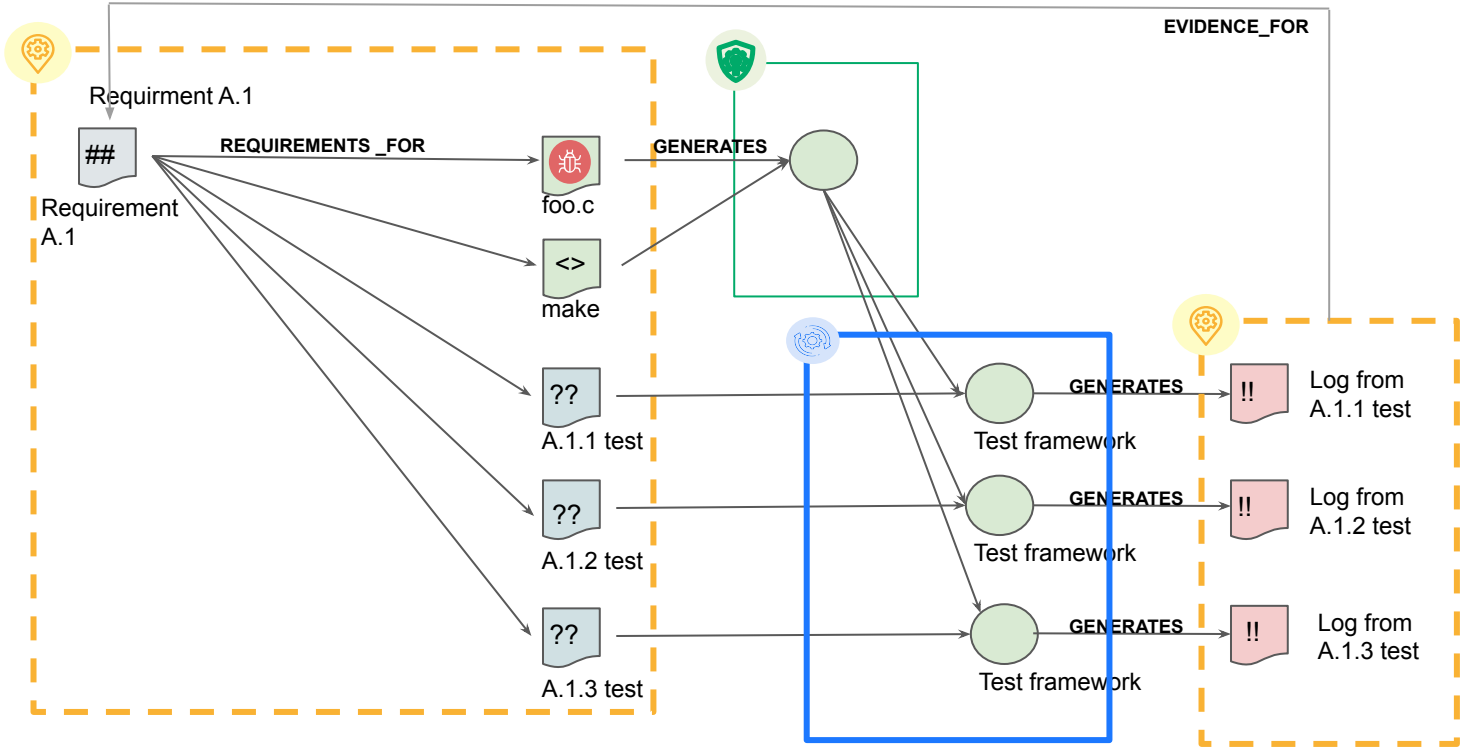


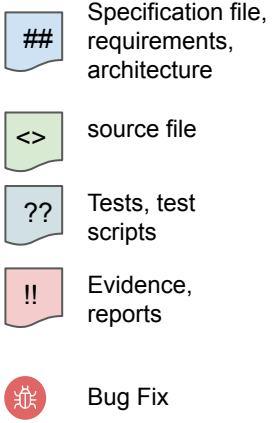
Requirement to Code to Tests to Evidence Traceability



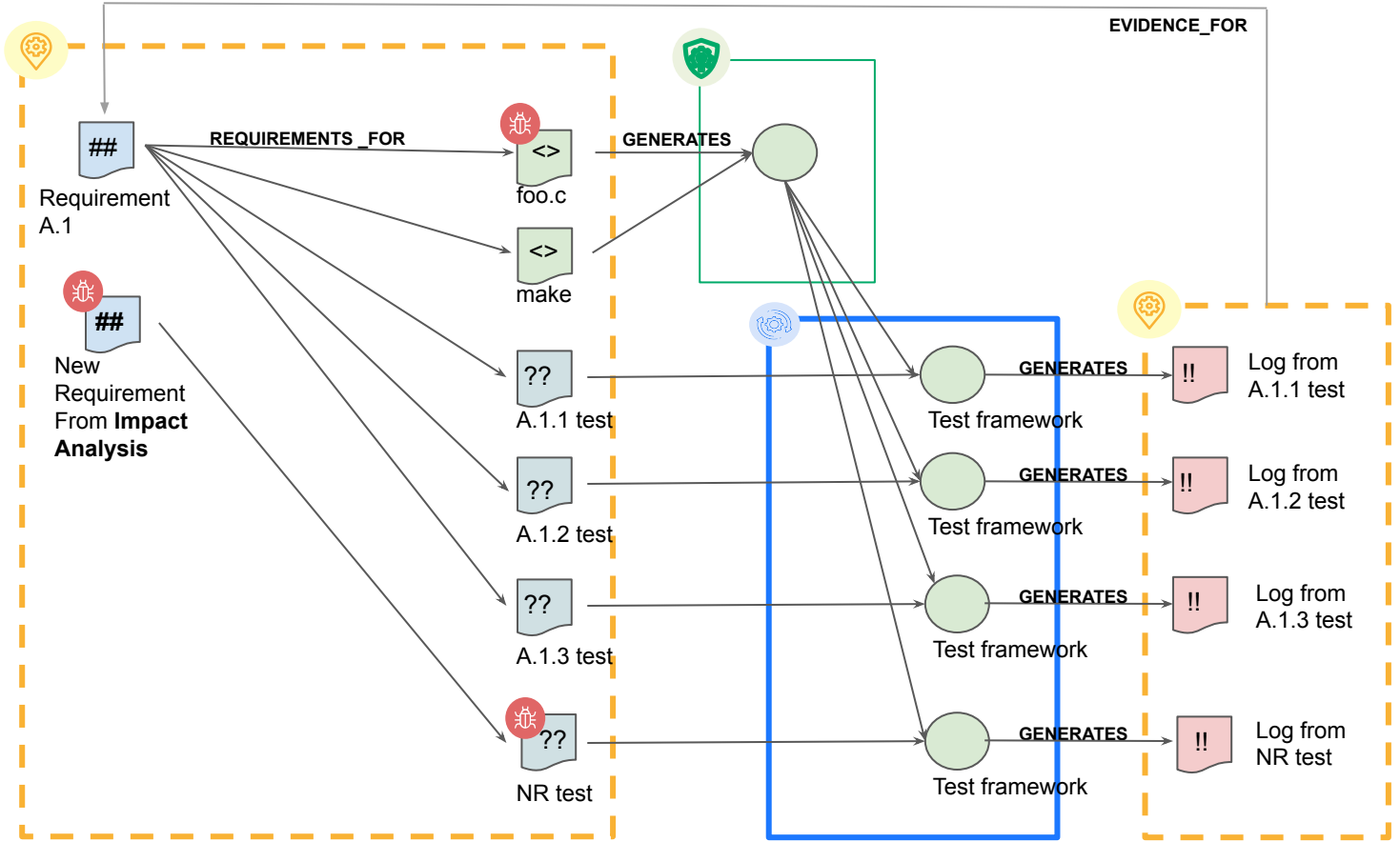


Requirement to Code to Tests to Evidence Traceability





Requirement to Code to Tests to Evidence Traceability



Specification file, requirements, architecture

source file

Tests, test scripts

Evidence, reports

Bug Fix

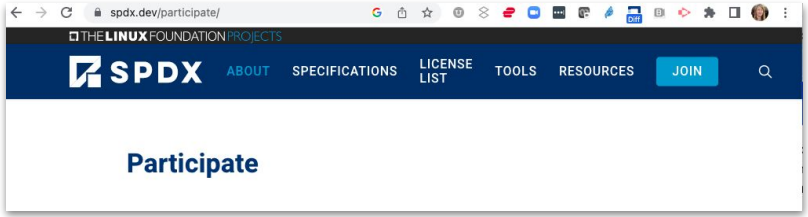
Safety Profile Introduction (3.1 target)

- Also known as Functional Safety (or FuSa).
- Purpose is to link together all the safety artifacts (including code and relevant tests) with the aim of being able to automatically detect what a file update may need to force retesting.
- Goal is to support continuous certification of safety artifacts after security updates are applied.
- Overview can be found at: https://fosdem.org/2023/schedule/event/sbom_fusa/

For more information, contact: Nicole Pappler or Kate Stewart

SPDX

Want to Help?



- If you have a **use-case** you want to make sure can be supported in the future SPDX specification,
  - join the SPDX tech team mailing list (https://lists.spdx.org/g/Spdx-tech) ,
  - open an issue in https://github.com/spdx/spdx-spec and
  - join in on the discussion!

- Try it, and let us know if you see issues.

Thank you! Questions?

How to Get Involved - PRs & Issues

https://github.com/spdx

- Specification
  - https://github.com/spdx/spdx-spec ← ISO submission format
  - https://github.com/spdx/spdx-3-model ← 3.0 development
  - https://github.com/spdx/spdx-examples
- Tooling
  - https://github.com/spdx/tools-python
  - https://github.com/spdx/tools-golang
  - https://github.com/spdx/tools-java
- License List
  - https://github.com/spdx/license-list-XML

SPDX

Embedded Projects Generating SBOMs

Zephyr's west spdx

Presentation / Demo:

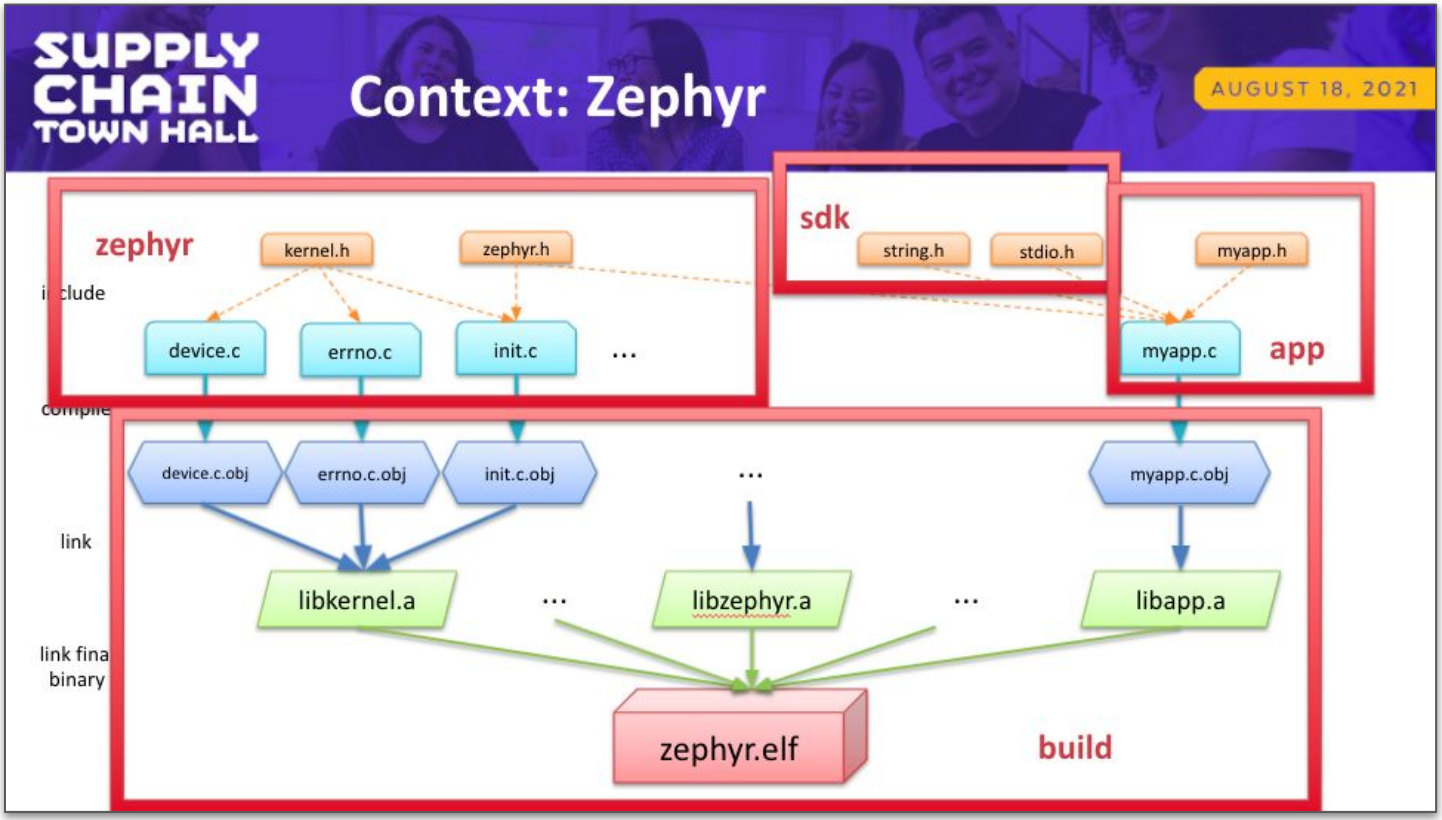
https://www.youtube.com/watch?v=KYC3YpSu9zs

Yocto builds

Presentation / Demo:

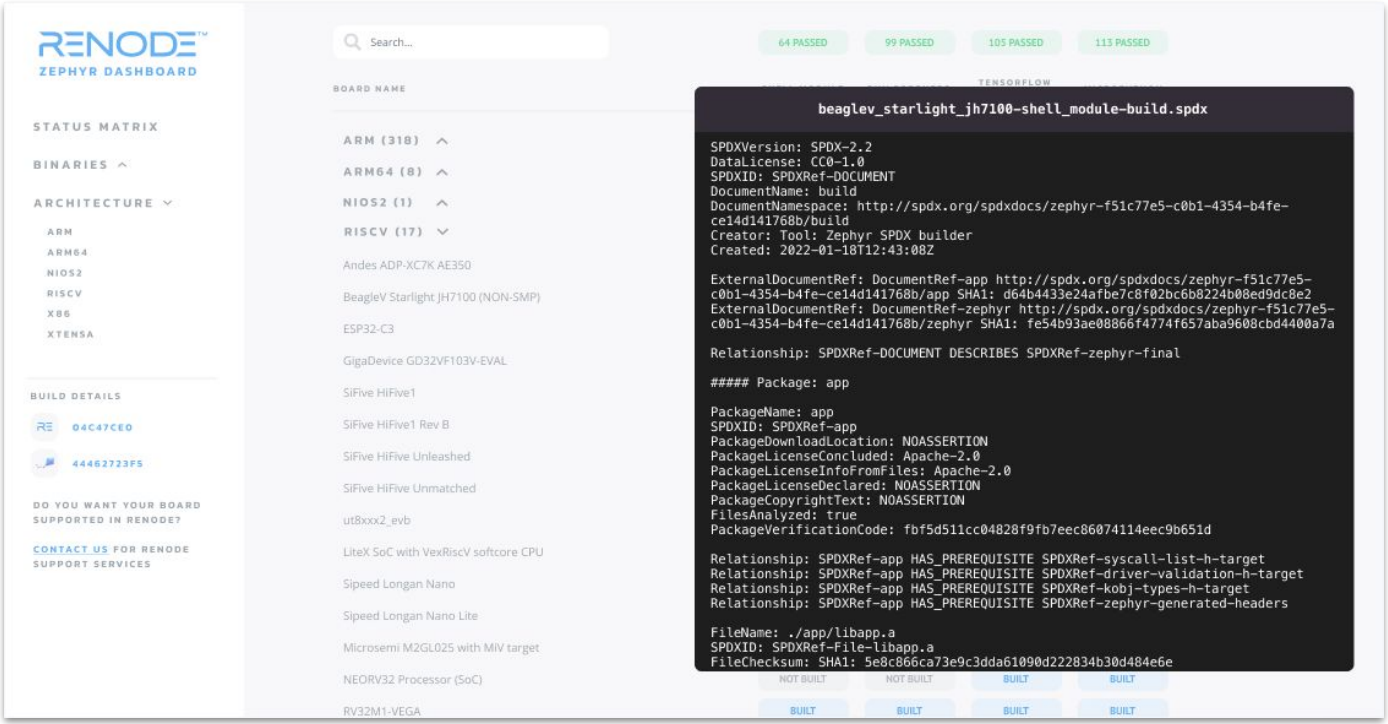
https://www.youtube.com/watch?v=y0N4FnkwTOY

Relationship between SBOMs



Learn more at: https://www.youtube.com/watch?v=KYC3YpSu9zs

SBOMs Included By Default … Automatically



Source:https://www.linux.com/featured/enhancing-supply-chain-security-for-embedded-systems-renode-dashboard-for-zephyr-rtos-adds-new-software-bill-of-materials-sbom-capabilities-by-default/