SPDX Information Modeling
NTIA SBOM Plugfest #1

David Kemp, 16 April 2021

Introduction

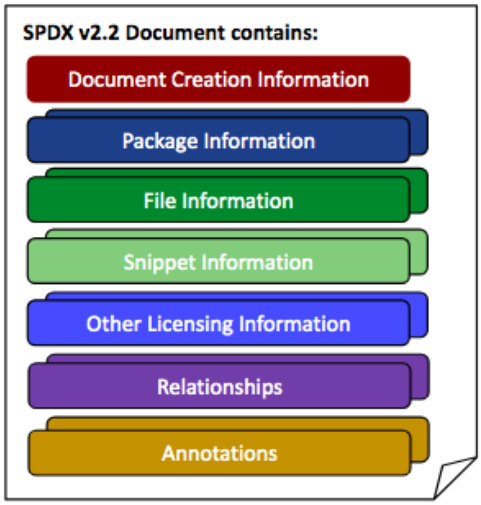
We created two OpenC2 information models for SPDX: first by deriving directly from the SPDX version 2.2 specification, then by updating until the model successfully validated JSON test data submitted by participants at the 9 April NTIA SBOM virtual plugfest. The purpose was to develop "spec-derived" and "reverse-engineered" representations of SPDX data in a format that enables direct comparison. A number of differences between the models were found, suggesting areas where:

1. the spec could be written more precisely to eliminate ambiguity,
2. the underlying design of the spec could be improved, or
3. the submitted data was incorrect and no changes to the spec were indicated

Observations

1. DocumentCreationInfo

Section 1.4 describes what the SPDX specification covers, listing seven types of content:



A top-level information model can be constructed from this layout, augmented by examining the spec as a whole, and making assumptions where necessary:

```
Document = Record
   1 creationInfo             DocumentCreationInfo
   2 packages                 PackageInfo [0..*]
   3 files                    FileInfo [1..*]
   4 snippets                 SnippetInfo [0..*]
   5 otherLicenses            OtherLicensingInfo [0..*]
   6 relationships            Relationship [0..*]
   7 annotations              Annotation [0..*]
```

Section 2 explicitly states that Document Creation Information has a cardinality of Mandatory, one, and section 3 says package cardinality is Optional, one or many (i.e., cardinality is 0..*). The remaining sections 4-8 do not explicitly state a cardinality but it can be inferred from the wording, e.g., from section 5: "Snippets can optionally be used ...".

- we assume that it is not useful to have an SPDX Document without at least one file. If zero files makes sense the cardinality of "files" can be set to [0..*].
- we assume that all other sections are optional
- we could not find a field name for Other Licensing Information in the specification, the JSON schema, or the JSON files examined, so "otherLicenses" seemed appropriately parallel to the other plural field names

Recommendation #1: Section 1 should explicitly state a Document's fields and cardinalities, as in the remaining sections. This would collect that information in one location for easy reference and, more importantly, it would prevent the inconsistencies described next.

2. Document Creation Information

Section 2 defines the fields in Document Creation Information.

```
DocumentCreationInfo = Record
   1 spdxVersion              SpdxVersion
   2 dataLicense              SpdxLicense
   3 SPDXID                   Key(SPDXREF-DOCUMENT)
   4 name                     String
   5 documentNamespace        URI
```

```
   6 externalDocumentRefs     ExternalDocumentRef [0..*]
   7 licenseListVersion       MajorMinorVersion optional
   8 creators                 Entity [1..*]
   9 created                  Timestamp
  10 comment                  String optional
  11 documentComment          String optional
```

Information modeling allows types to be given a name and referenced early in the design process, with details filled in later. In this section:

- SPDXREF-DOCUMENT is later defined as a string with the constant value "SPDXREF-DOCUMENT".
- The Key() notation does not affect instance values, but indicates that that field is a primary key that can be used to reference instances of this type. (A primary key with a constant value implies that only one instance of DocumentCreationInfo can exist in a document, since primary keys uniquely identify rows of a table or instances of a type).
- We chose to call the creator type "Entity" since a similar type is also used in Annotations. It may be preferable to define separate Creator and Annotator types, since the Tool type can be a creator but not an annotator.

In the plugfest data, Document Creation Information had the greatest divergence from the spec. Although section 2 defines 11 fields, only six of them appeared in DocumentCreationInfo instances. The other five appeared at the top level (Document) mixed in with the seven types defined in section 1. In order to validate the submitted data, we had to add five document creation info fields to Document, and make them optional in DocumentCreationInfo.

The Source Auditor SPDX examples also included two top-level fields that are not defined in the spec but that plugfest participants found very helpful:

```
Document = Record
   1 creationInfo             DocumentCreationInfo
   2 packages                 PackageInfo [0..*]
   3 files                    FileInfo [1..*]
   4 snippets                 SnippetInfo [0..*]
   5 otherLicenses            OtherLicensingInfo [0..*]
   6 relationships            Relationship [0..*]
   7 annotations              Annotation [0..*]
```

```
   8 spdxVersion              SpdxVersion
   9 dataLicense              SpdxLicense
  10 SPDXID                   Key(SpdxRef)
  11 name                     String
  12 documentNamespace        URI
  13 hasExtractedLicensingInfos LicensingInfo [0..*]
  14 documentDescribes        SpdxRef [0..*]
```

```
DocumentCreationInfo = Record
   1 spdxVersion              SpdxVersion optional
   2 dataLicense              SpdxLicense optional
   3 SPDXID                   Key(SPDXREF-DOCUMENT) optional
   4 name                     String optional
   5 documentNamespace        URI optional
   6 externalDocumentRefs     ExternalDocumentRef [0..*]
   7 licenseListVersion       MajorMinorVersion optional
   8 creators                 Entity [1..*]
   9 created                  Timestamp
  10 comment                  String optional
  11 documentComment          String optional
```

```
CrossRef = Record
  1 isLive Boolean
  2 isValid Boolean
  3 isWayBackLink Boolean
  4 match String
  5 order Integer
  6 timestamp Timestamp
  7 url URI
```

Recommendation #2: "Document Creation Information" could be interpreted as descriptive rather than prescriptive, but the v2.2 specification should eliminate that ambiguity by clearly defining which fields appear at the top level (section 1) and which are elements of the DCI type (section 2). The v3.0 specification should place all fields that apply to the document as a whole in a common location..

Recommendation #3: The SPDX technical committee should consider standardizing the hasExtractedLicensingInfos and documentDescribes fields, preferably within the DCI type. The Cross-reference example data included in LicensingInfo an anomaly that should be

fixed: the "match" field has string values "true" and "false"; they should be boolean values as in the preceding fields.

Recommendation #4: For v3.0 rename the "SPDXID" field to "spdxId" to align with the form of other field names. Consider whether other field names with a redundant prefix can be shortened, e.g., within the Annotation type: "annotationDate" -> "date", "annotationType" -> "type".

3. Package Information

Several fields are marked as Mandatory that are logically optional - the SPDX document creator either does not have the information or chooses not to include it. These fields are omitted in submitted plugfest data.

```
PackageInfo = Record
   1 name                      String
   2 SPDXID                    Key(SpdxRef)
   3 versionInfo               VersionInfo optional
   4 packageFileName           PackageFileName optional
   5 supplier                  PackageSupplier optional
   6 originator                Entity optional
   7 downloadLocation          DownloadLocation optional
   8 filesAnalyzed             Boolean optional
   9 packageVerificationCode   PackageVerificationCode optional
  10 checksums                 Checksum [0..*]
  11 homepage                  URI optional
  12 sourceInfo                String optional
  13 licenseConcluded          LicenseExpression optional
  14 licenseInfoFromFiles      LicenseIdentifier [0..*]
  15 licenseDeclared           LicenseExpression optional
  16 licenseComments           String optional
  17 copyrightText             String optional
  18 summary                   String optional
  19 description               String optional
  20 comment                   String optional
  21 externalRefs              ExternalRef [0..*]
  22 attributionText           String [0..*]
  23 annotations               Annotation [0..*]
  24 hasFiles                  SpdxRef [0..*]
```

Recommendation #5: Designate the downloadLocation, licenseConcluded, licenseDeclared, and copyrightText fields as optional rather than populating them with a "NOASSERTION" value. The absence of those fields in Package Information indicates that the document makes no assertion about their value.

Recommendation #6: The SPDX technical committee should consider standardizing the hasFiles field, as plugfest participants considered the information useful. The semantic distinction between comment and annotation should be documented, or annotation (being a superset) should replace comment.

Recommendation #7: Section 8 (Annotations) says: "For RDF, the annotations are a property of the SPDX Document, Package, File, or Snippet they are annotating." Sections 3, 4 and 5 should define the annotations field of the Package, File, and Snippet, as the submitted test data shows for Package. The "For RDF" prefix in section 8 should be removed; the conceptual design of a document should not depend on how it is serialized.

4. File Information

As with Package Information, the licenseConcluded, licenseInfoInFiles, copyrightText and contributors fields were omitted in some plugfest contributions and should be optional.

The versionInfo (3.3), licenseDeclared (3.15) and externalRefs (3.21) fields from Package Information do not have counterparts in Section 4 of the SPDX v2.2 spec, but Cybeats submitted examples including these fields.

The artifactOf (4.9-11) and dependencies (4.16) fields are deprecated in version 2.2 and will presumably be gone in version 3.0.

```
FileInfo = Record
   1 fileName                 String
   2 SPDXID                   Key(SpdxRef)
   3 versionInfo              VersionInfo optional
   4 fileTypes                FileType [0..*]
   5 checksums                Checksum [0..*]
   6 licenseConcluded         LicenseExpression optional
   7 licenseInfoInFiles       LicenseExpression [0..*]
   8 licenseDeclared          String optional
   9 licenseComments          String optional
  10 copyrightText            String optional
  11 comment                  String optional
```

```
12 noticeText              String optional
13 contributors            String optional
14 externalRefs            ExternalRef [0..*]
15 attributionText         String [0..*]
16 annotations             Annotation [0..*]
17 artifactOf              Artifact [0..*]
18 dependencies            String [0..*]
```

Recommendation #8: If versionInfo, licenseDeclared, and externalRefs are intended to be applicable to Files, they should be added to the spec. If not, the submitted example documents should be corrected.

5. Checksum structure

The spec includes examples of Package Checksum (3.10) in tag and RDF formats. The tag format shows the algorithm and value directly; but the RDF defines separate keywords for "algorithm" and "checksumValue". JSON objects are already key: value pairs, and a design pattern using the keywords "algorithm" and "checksumValue" repeated for each checksum is unnecessarily complex.

```
"checksums": [
   {
     "algorithm": "MD5",
     "checksumValue": "701f4316d3b7e19969bd2f6a92912dbc"
   },
   {
     "algorithm": "SHA1",
     "checksumValue": "cf5502039c527112eafe1999871a085c2f4d961a"
   }
 ],
```

```
"checksums": {
   "MD5": "701f4316d3b7e19969bd2f6a92912dbc",
   "SHA1": "cf5502039c527112eafe1999871a085c2f4d961a"
 },
```

Recommendation #9: Use algorithm: value pairs directly as Checksum values. In addition to simplicity, this also has the advantage of semantically linking the value to the algorithm, so "MD5": `"cf5502039c527112eafe1999871a085c2f4d961a"` is immediately known to be incorrect because of the length mismatch.

6. Enumerations

Section 3.21.4 defines Categories for external references. Some submitted data includes a typo "PACKAGE_MANAGER" with an underscore instead of a hyphen. The data should be corrected.

Section 4.3 defines File Types. Some submitted data includes a SHARED file type not defined in the spec.

```
ReferenceCategory = Enumerated
   1 SECURITY
   2 PACKAGE-MANAGER
   3 PERSISTENT-ID
  99 OTHER

FileType = Enumerated
   1 SOURCE
   2 BINARY
   3 ARCHIVE
   4 APPLICATION
   5 AUDIO
   6 IMAGE
   7 TEXT
   8 VIDEO
   9 DOCUMENTATION
  10 SPDX
  11 SHARED
  99 OTHER
```

Recommendation #10: The SPDX technical committee should consider adding SHARED to the list of file types.

7. SPDX Information Models and Tools

The complete spec-derived and reverse-engineered models and the Python script used in this report are available at https://github.com/davaya/ntia-sbom-plugfest. The code currently supports translating an information model to multiple formats: IDL text, HTML tables, Markdown tables, GraphViz diagrams, and JSON Schema files. We are planning additional translations for XSD and Protobuf schemas, to support generating documentation and serialization from a single source. We welcome collaboration on information model translators for RDF and other documentation and data formats.

Recommendation #11: The SPDX technical committee should consider adopting the JSON Schema generated from an information model as the normative specification for JSON and YAML document creators.

Recommendation #12: The SPDX technical committee should consider adopting the JADN schema language as either a behind-the-scenes consistency tool for generating artifacts in the current format, or for publication directly as an information model artifact.