Topics to consider for Next Rev of Spec (SPDX 2.2 or 3.0)

- Deprecating SHA1 - now tracked in https://github.com/spdx/spdx-spec/issues/11
    - (CII - MUST?) PVC to use something
    - Uday interested in putting this proposal, Brad, Yev +1
    - SPDX 3.0 - is going to be needed.   General ok from all on call.

- Relationships b/t elements - now tracked in https://github.com/spdx/spdx-spec/issues/12
    - (spec 7.1) - expand - which are missing.   & definitions.
    - Yev would like to see inverse of EXPANDED_FROM_ARCHIVE,   Archive of.
    - Thomas would like to add following relationships to better define context of license finding:
        - EXAMPLE_OF - Source code included with OSS package for example purposes.
        - TEST_TOOL_OF - To distinguish test frameworks such as jUnit from source dependencies.
        - TOOL_OF or DEV_TOOL_OF - To indicate tooling included in OSS package for development or utility purposes which are not used to build a package. Think git commit hooks, FTP upload scripts, etc.
    - Clarifying prerequisite description in existing spec (2.2?), target system
    - Others to comment… provide input.

- Relationships b/t SPDX license-list files (new element?)
  https://github.com/spdx/spdx-spec/issues/13
    - Problem: sets of related licenses, esp. Translations
    - One solution: treat every license as separate file, then describe relationship w/ new element?
    - Any way to describe relationships among license groups such as official/unofficial translations, ported/unported, etc.?
        - For EU Public License in German, that might look something like this:
        - …
        - <relatedLicenses>
        - <relatedLicense relationshipType="official-translation" targetLicenseIdentifier="EUPL-1.1">EUPL-1.1</relatedLicense>
        - </relatedLicenses>
        - …
        - Up to 24 for EUPL, etc.
    - Could this include license stacks (like newlib)? License stacks used as licenses? How to differentiate from license stacks used as informal package-license manifests?

- Thomas: See a need for syntax to capture how a package dependencies where the dependencies are specified with version range tracked in https://github.com/spdx/spdx-spec/issues/14
    - e.g. resulting non deterministic builds. SPDX only now offers to specify dependencies using fixed one-on-one relationships e.g Package A depends on Package B v1.1. In reality Package A specifies it relies on Package B v1.0 or newer. Having this in the spec provides package maintainers a technology agnostic way to specify their dependencies closer to reality. Provides consumers of these packages with an indicator that including package may result license mix that can change with every build. May also be useful to handle the difference between the declared (by maintainer) and resolved dependencies (by package manager).

      Example - SPDX specifies dependency on angular 4.1.1, see it's package.json specifies depends on core-js 2.4.1 or newer

      Note: approach is not figured out yet, but general agreement that this is a problem and we should look into solving it for the next release.

Size of File tracked in https://github.com/spdx/spdx-spec/issues/21
(optional) - express as number of bytes similar to mechanism used for snippet. Will be useful for heuristics working with snippets and licensing.

Questions from Gary - single format or not? If multiple, which ones going forward?
- SPDX as simpler JSON or YAML
- SPDX/Turtle - see: example

From discussion, multiple formats seem to be in our future. Emphasis on translating the XML to easier to read RDF variant (Turtle) in specification.
Thomas: Tag value isn't quite YAML, really want a standard parser to be able to work. Will help adoption. Easy as possible to manipulate is going to be key.
Yev: All SPDX terminology into YAML.
Kate: The fields are important, like license identifiers are for license list. Format that they show up in needs to have standard parsers available to promote further adoption.

- Signing SPDX documents - basic PKI support? Blockchain?

Comments in SPDX documents (depends on file formats). - https://github.com/spdx/spdx-spec/issues/22
    - In RDFa/XML - there is a specific term defined.
    - In tag:value - # - at start of line - need to be added to document. Entire line comments.
    - No middle of comment line in SPDX document.

Comments take the form of '#', as the first non-blank character, and continue to the end of line (marked by characters U+000D or U+000A) or end of file if there is no end of line after the comment marker. Comments are treated as white space.

⇒ Alexios recommends aligning with Turtle

What about SPDX-License-Identifier: ( ) in source code.

- # ok if disallowed character? Need to check specification.

<u>Licenses List Changes</u>

- Replace Deprecated=true w/ Obsoleted-By per Trevor's suggestions: https://github.com/spdx/license-list-XML/pull/392
    - Note: Concern about compatibility, with prior implementations. Good idea to probably add obsoleted-by.

- XSD specification for legal team's XML format: https://github.com/spdx/license-list-XML/issues/391
    - Gary notes that he's working on this (WIP). Working on schema-dev branch on license xml list. All licenses are converted to new schema.

- Support formatting in Notes field: https://github.com/spdx/license-list-XML/issues/397
    - Done as of yesterday at 5pm. :-)

Extend Appendix VI: External Repository Identifiers category PACKAGE-MANAGER with
    - Add common developer service cocoapods, rubygems, pip, sbt, etc..
- See: https://github.com/spdx/spdx-spec/issues/24

Enable Appendix VI: External Repository Identifiers category PACKAGE-MANAGER with referring to package repositories by URL such Artifactory, Bintray, Nexus, etc.

Examples:
```
ExternalRef: PACKAGE-MANAGER cocoapods FBSDKCoreKit/4.17.0
ExternalRef: PACKAGE-MANAGER pypi numeral/0.1.0.8
ExternalRef: PACKAGE-MANAGER sbt+repo.scala-sbt.org/scalasbt/sbt-plugin-releases com.sc.sbt
%sbt-eslint%1.0.0
ExternalRef: PACKAGE-MANAGER gradle+http://repo.jfrog.org/artifactory#easymock
easymock:easymock:2.0
```
- See: https://github.com/spdx/spdx-spec/issues/25

`Fix Typo:` (note PACKAGE_MANAGER in Appendix VI should be PACKAGE-MANAGER) Thanks for highlighting Alexios :-) https://github.com/spdx/spdx-spec/issues/27

`AttributionText` → https://github.com/spdx/spdx-spec/issues/28

- Agreement from those on the call,   yes it should be at File & Package level.
- This will be an optional field.
- Useful for generating notice files, etc.
- Consider does it make sense for snippets?

- Adding “Attributions” to package field to store required attributions (per Oliver)
- Alexios: property named FileAttributionText which will hold the text that has to be reproduced.
  It can be considered as a combination of information found in properties like FileCopyrightText, LicenseInfoInFile, FileContributor, but it might not simply the sum of these values.

  The relative part in the spec could be something like:

```
4.xx   File Attribution Text
4.xx.1 Purpose: This field provides a place for the SPDX data creator to record all
attributions found in the file that are required to be communicated. These typically
include copyright statement(s), license text, and a disclaimer.
4.xx.2 Intent: The intent is to provide the recipient of the SPDX file with all the
legally required attributions in the file, therefore complying with the license
obligations.
4.xx.3 Cardinality: Optional, one.
4.xx.4 Data Format: free form text that can (and usually will) span multiple lines
4.xx.5 Tag: "FileAttributionText:"
In Tag:value format, the multiple lines are delimited by "<text>" and "</text>".
Example:
FileAttributionText: <text>
#   Copyright (C) 2004 Free Software Foundation, Inc.
#   Written by Scott James Remnant, 2004
#
# This file is free software; the Free Software Foundation gives
# unlimited permission to copy and/or distribute it, with or without
# modifications, as long as this notice is preserved.
</text>
4.xx.6 RDF: property fileAttributionText in class spdx:File
Example:
       <File rdf:about="...">
              <fileAttributionText>
#   Copyright (C) 2004 Free Software Foundation, Inc.
#   Written by Scott James Remnant, 2004
#
# This file is free software; the Free Software Foundation gives
# unlimited permission to copy and/or distribute it, with or without
# modifications, as long as this notice is preserved.
              </fileAttributionText>
       </File>
```

←-- ended review on 2017/5/29 ----->

Thomas: SPDX 2.1 has 34 mandatory tags. Propose to reduce the number of mandatory fields to minimal fields needed for exchange to reduce friction to participate. See: https://github.com/spdx/spdx-spec/issues/29

Believe the following fields could be made optional:

- §2.4 DocumentName
- §2.5 DocumentNamespace - Most of the time these URL are totally artificial as producers do not maintain SPDX as linked data
- §3.9 PackageVerificationCode - Verification should be optional, license scanners ignore different type of files such as .git dirs and as such two scanners can produce different PackageVerificationCode for the same package
- §3.15 PackageLicenseDeclared - PackageLicenseConcluded and PackageLicenseInfoFromFiles provide the same information
- §5.3 SnippetByteRange - Making this optional reduces friction to SPDX participation. Maintainer can easily manually specify SnippetLineRange but SnippetByteRange requires tooling

- Gary: consider going to profiles?   Simpler - same field names, make optional or not. Some redundancy in documentation.

SPDX-Document
2.1.5 SPDXVersion
2.2.5 DataLicense
2.3.5 SPDXID
2.4.5 DocumentName
2.5.5 DocumentNamespace
2.8.5 Creator
2.9.5 Created

SPDX-Package
3.1.5 PackageName
3.2.5 SPDXID
3.7.5 PackageDownloadLocation
3.9.6 PackageVerificationCode
3.13.5 PackageLicenseConcluded
3.14.5 PackageLicenseInfoFromFiles
3.15.5 PackageLicenseDeclared
3.17.5 PackageCopyrightText

SPDX-File
4.1.5 FileName

4.2.5 SPDXID
4.4.6 FileChecksum
4.5.5 LicenseConcluded
4.6.5 LicenseInfoInFile
4.8.5 FileCopyrightText

SPDX-Snippet
5.1.5 SnippetSPDXID
5.2.5 SnippetFromFileSPDXID
5.3.5 SnippetByteRange
5.5.5 SnippetLicenseConcluded
5.8.5 SnippetCopyrightText

Non-SPDX License Identifier
6.1.5 LicenseID
6.2.5 ExtractedText
6.3.5 LicenseName

SPDX–Annotation
8.1.5 Annotator
8.2.5 AnnotationDate
8.3.5 AnnotationType
8.5.5 AnnotationComment

← ended review on 2017/6/13 -->

Yev: Allow relationship types to be predicates → https://github.com/spdx/spdx-spec/issues/30

(e.g. http://mynamspace#mypackage spdx:contains http://mynamespace#myfile).

- Verbose due to the presence of optional comment field
- Details from discussion from Yev:

  Package → Relationship
  Relationship → File

  Package:contains → File

  contains: http://myname:myFile

```
<.... id="myPackage">
        <spdx:contains id="http://myname:myFile" />
</rdf:Description>
```

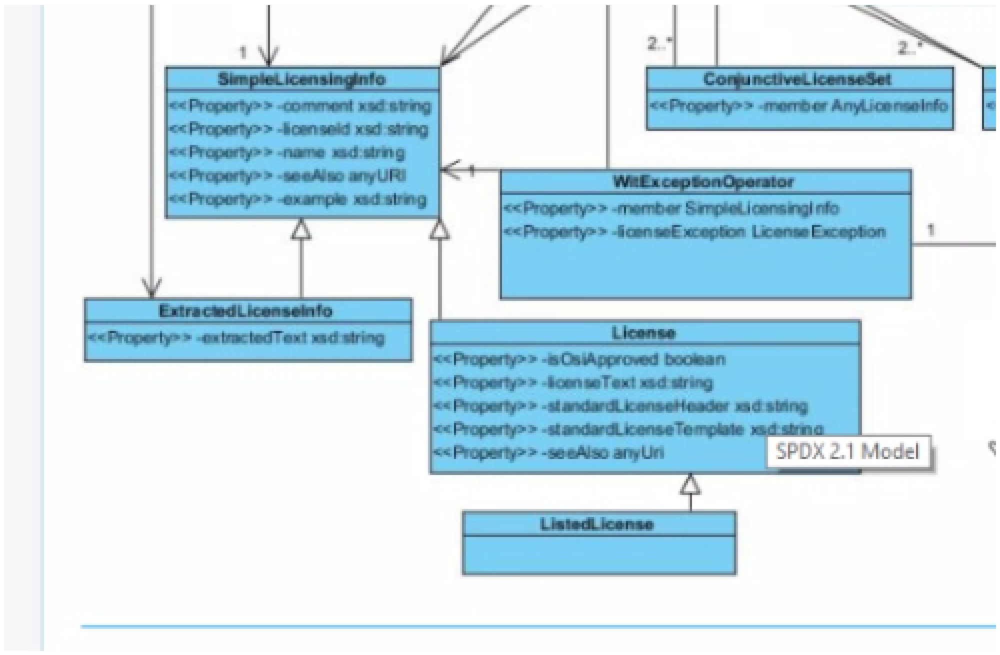
http://myNamespace#myPackge spdx:contains http://myNamepsace#myFile

Thomas notes they are using relationship comments to customize relationship. So would not like to see this ability remove. Likes the proposal, but not want to see "other" removed.

Yev, both should be valid (short version, as well as original). Yev to provide example. Not make old ones go away, just enable addition of concise way of expressing relationships.
Thomas agrees Tag value will become clearer as result of having this additional syntax.
Open question - can annotation describe a relationship? Based on model, not able to. So not a solution at this point.

Thomas: Extend SPDX-Package with optional "PackageNameAliases" field to record aliases to PackageName. Use case - support renamed packages or the same package that have different names on various distribution platforms. Example MySQL <-> MariaDB
Idea is being able to record alternate names for same package.
This would be optional. Want to be able to semantically detect this.
Yev: possibly change the cardinality? Prevents the document creator on what the "official name" is.
Gary: Might be valuable to retain an official name? Can see it both way. Use case, package originator.
Yev: Gets to declared/concluded dichotomy, are we sure we want to go there?
Gary: Not sure, can think of some cases either way.
Yev: Maybe introduce AlternateName (for package, file, license, snippet, etc.) and leave the package name specified is the one that SPDX document author as authorative voice.
Gary: Apply at element level makes sense.
Yev: Not sure if files should have alternate name though. File encoding could be probablamatic. We'll need to apply to each with semantic, so not sure we got to there.
Yev: Prefers AlternateName over "..Alias" concept. Cardinality 0 to many.
Thomas: Is good with AlternateName terminology. Only question is should we prefix it? Ie. PackageAlternateName in tag:value. And then AlternateName in RDF.
Yev & Thomas in agreement.
Yev: If one or more alternate names provided, does PackageName needed?
Gary, Kate: Yes.
Alexios: Only for Packages right now? Yev: Yes, lets limit it to this right now.
Can think of case that it would be good to have license name alternates…
Yev: Possibly, but may want to consult further with legal. Snippet names avoid. File names - no.
Conclusion: ok to add PackageAlternateName as optional field with 0-many. Tracked in https://github.com/spdx/spdx-spec/issues/31

← ended review on 2017/6/27 -->

- Thomas: Make the matching template formats of license part of the spec - both SPDX listed licenses and NON-SPDX listed licenses. Would like to add matching guidelines annotation to SPDX licenses and to NON-SPDX licenses. Also add templating for copyright holders and dates.
    - XML specification of license texts. Has templating. Matching guidelines.
    - Want to add cross references to license that are on the SPDX license list.
    - Concern: schema to store information about license is ok, but matching templates could become problematic. May be differnently to apply consistently. Old templating language in specification is only available on listed licenses. Make other properties to listed licenses. XML language is being used by legal team to line up with guidelines, but may not be standardized enough. Non-standardized input format, move to output format.
    - This is possibly 3 different proposals:
        - Add additional properties to OTHER LICENSE INFORMATION file to bring up to same level as SPDX listed licenses.
        - Add additional fields for listed licenses, so information present in XML can be made visible as start of output representations (for instance bullets, copyright) (we don't want them using the input format)
        - Add in OTHER LICENSE INFORMATION that is not in SPDX license list model to the SPDX license lists (ie. comment)
    - 
1
SimpleLicensingInfo
<<Property>> -comment xsd:string
<<Property>> -licenseId xsd:string
<<Property>> -name xsd:string
<<Property>> -seeAlso anyURI
<<Property>> -example xsd:string
2..*
2..*
ConjunctiveLicenseSet
<<Property>> -member AnyLicenseInfo
WithExceptionOperator
<<Property>> -member SimpleLicensingInfo
<<Property>> -licenseException LicenseException
1
ExtractedLicenseInfo
<<Property>> -extractedText xsd:string
License
<<Property>> -isOsiApproved boolean
<<Property>> -licenseText xsd:string
<<Property>> -standardLicenseHeader xsd:string
<<Property>> -standardLicenseTemplate xsd:string
<<Property>> -seeAlso anyUri
SPDX 2.1 Model
ListedLicense

- Some harmonization here is going to be needed. We probably want to include license exceptions in remodeling discussion. This is probably a 3.0 feature.

- See: https://github.com/spdx/spdx-spec/issues/32
-

← ended review on 2017/7/4 -->

- Thomas: New tag PackageTag (optional, cardinality - multiple) enable users to add custom (user-defined?) tags to package to group. Useful to automatic assessment of license results

  Example:

  PackageName: jUnit
  # Similar to Maven dependency scopes e.g. compile, runtime, etc.
  PackageTag: scope:test
  # Defines type of SW such as build_tool, test_framework, sw_library, utility
  PackageTag: type:build_tool
- Kate: Currently using Package Comment - but problematic with filtering with other comments that are in the comments. Overloaded, so problematic.
- Kate: Could it be a Package Type - like https://spdx.org/spdx-specification-21-web-version#h.7vzbl5vywpa7 ? Yev not sure worth going in this direction.
- Gary: Could Annotation be used? Extend annotation types? https://spdx.org/spdx-specification-21-web-version#h.wlc7jg3vsu43
- Kate: Use case of package compiled to binary file - would we want to tag a binary or jar file? Thomas, Gary - yes can see it used but not as common as packages
- Yev: when documenting a supply chain - what should be tagged, what should be in relationship?
- Gary: can see it being useful. Maybe annotations isn't best approach. But having it applicable to all elements may make sense? Property of element.
- Yev: Images, containers, etc.
- Alex: Licenses - …
- New High level property appropriate to any element?
- Yev: We need to be able careful with tagging license…
- Alex: Whatever you put in tags, should be interpreted of author of SPDX document.
- Freeform tagging vs. specific categories - user assertion. Specify tags are declaration of documentation author, vs. Custom tags not declarative. Create adhoc tags, and no meaning as far as SPDX concerns. Spec says it can exist.
- Yev: That makes it useful for insourcing
- Thomas: Also could be used when customer relationship.
- Tending towards: May apply to any element, optional, signal author may want to convey. Enumeration 0 or more.

- Yev: Field in Document Scope to describe the meanings of tags? Also could be done in Comments. Explanation of used tags included in creator comment - as best practice.
- Conclusion: Keyword Section - may apply to any element, its optional, with cardinality 0 to many, Signal that authors to convey. Schedule for 3.0 → https://github.com/spdx/spdx-spec/issues/33
-

← ended review on 2017/7/11 -->

- Thomas: Expand §8.3.4 Annotation Type with new values → https://github.com/spdx/spdx-spec/issues/35
- Originally just context of "LICENSE" | "PATENT".
- Enables annotator to more precisely indicate type of annotation
    - Discussed use case brought up by customers - will save research. OpenCV is case cited. Be able to annotate that a patent has expired, etc.
    - Use LICENSE when its not 100% clear, so may want to provide information about equivalence or not with another.   Zlib 1.0.6 and another close to it.
    - Different lawyers handle different roles, want to give lawyers comments that apply to the appropriate reviewers.
    - Thomas: Copyrighter holder may be out of business,   so may want to have a "COPYRIGHT" as well.
    - Discussion of who adds the and roles between Alexios & Thomas.
    - Desire to permit multiple TYPES to be used with ANNOTATION
    - 1) Add in new values LICENSE, PATENT, COPYRIGHT, EXPORT, TRADEMARK as valid types.
    - 2) Permit cardinality from one to many.
    - Gary ok with cardinality change, sees as useful for automation.
    - Looking at this as a 2.2 feature.
    - TO BE DISCUSSED FURTHER: Do we want LICENSE here in this, as we can't point Annotation to License at this point in the model for this.   Adding Annotations to Licenses? Will this hold up in the data model.   Open question is how you can express for a complex license expression?

← ended review on 2017/7/18 -->

- Thomas: Instead of PackageCopyrightText and CopyrightText add introduce new PackageCopyrightHolder and CopyrightText. Both optional and more than one entry can exists per SPDX-File or SPDX-Package. Better indicates individual rights holders and makes parsing of this data easier.

  Kate: https://spdx.org/spdx-specification-21-web-version#h.2grqrue - relax one to many?
  Yev: Copyright holder implies present tense, could have been reassigned.   Declared vs concluded may be required, not sure we want to go there.

Thomas: Changing cardinality would help.
Yev: We'll need to apply this to Files & Snippets, as well.
Gary, Kate, Yev, Alexios, Thomas - all +1 on relaxing cardinality. 2.2, permit cardinality to go from one to many. Shelve copyrightholder until more compelling use case.
Tags: PackageCopyrightText, FileCopyrightText, SnippetCopyrightText;
RDF: property spdx:copyrightText → https://github.com/spdx/spdx-spec/issues/36

- Thomas: Add data "views" to the specification - a "view" deducts large SPDX documents into a small subset SPDX document providing a specific reduced "views" on larger data. Views provide can a tailored subset of SPDX data optimized for its viewer. Lower threshold to use SPDX without need for specific tooling.

  Yev: Orthogonal to specification process, and hard to define what all lawyers want to see. Provide best practices - possibly sparql ;-)
  Gary: Different profiles with different required fields, defining an SPDX lite. Different profiles for different purposes. Different required fields.
  Yev: If not required field, should not be required in the specification.
  Gary: Well defined use cases for SPDX documents, agree to use profile. Minimum required fields can count on.
  Yev: Would be a tightening of specification. What about more fields than SPDX specification require.
  Gary: would be a different use case. Borrowed from networking standardization efforts. Keeping spec loose, will let personalization. Treat it like best practices.

  Thomas: One case - open source projects adopting SPDX, but current mandatory fields too much.
  Gary: Agrees, best if we have a document that doesn't look big and scary.

  AI: Thomas to provide a write up on use cases he's seeing, and then the exercise is to compare to the mandatory fields in the specification, and see if there are insights to be gained for view/profile/extension. This will be 3.0 timescale at a minimum.

  Use cases:
  - Provide legal counsel instead of 1 SPDX-Package and 1 SPDX-File entry per found license instead of a SPDX-File for each scan file. Legal counsel does not need to know that are 400 source file under Apache-2.0 license for his/her conclusion she only needs one source file to verify Apache-2.0 is applicable.
  - Provide developers an easy way to correct SPDX data. No scanner can get complex packages 100% right. By providing package maintainer(s) a simplified customized SPDX representation it will be easier for them to make corrections.

→ 2017/9/12 - want to restrict to subset, filter to apply to SPDX file, and filter to specific exerpt and include.   New tools request??   Like a git patch to apply to an SPDX document.   Diff.

- Limited information for some file contents of package, but not requiring all the files of the license.
    - Yev suggests using relationship of package (files analyzed = false), and CONTAINS to make relationships to those specific files.
    - Thomas - limited view proposal, but not sufficient, have done all analysis. Add data "views" to the specification - a "view" deducts large SPDX documents into a small subset SPDX document providing a specific reduced "views" on larger data. Views provide can a tailored subset of SPDX data optimized for its viewer. Lower threshold to use SPDX without need for specific tooling

Thomas: we do a package analysis, files analyzed = true, but don't show the full contents, but only show a subsest of the files.   Proof of it, rather than exhaustive.
Gary: For clients, never include the proprietary files in the analysis, even though files=analyzed is true.
Thomas: We're doing the same thing.
Gary: Case where we can't have all the files present.
Yev: Files analyzed should be files documented.
Kate: Files analyzed has some cardinailty assumption.
Yev: Files analyzed → all files documented.   Then just add some files, and add explicit files with relationships.   Reminder files analyzed exempts you from having to document full file contents.   Makes wrong assertions
Gary: "cheating just a little bit" depends on consumers.   Include proprietary files in SPDX document.   Was doing this before the files analyzed field was added to the specification.   For his use case, could adapt to Yev's proposal.
Yev: package with files inside, is probably not best way to describe relationship.
https://spdx.org/spdx-specification-21-web-version#h.ve38itss85w
Thomas - tricky thing is the verification code.
Alexios - spec is close though.
Gary - litte ambiguous in spec - means all file analyzed that are included in verification code is how he uses it.
Thomas - name is causing problems.   Looking at meta data vs actually scanning files is an important distinction.
Gary - files analyze is false could be problem with client.
Yev - can we change name?

Thomas - likes field name to indicate analyzed - useful as such, maybe introduce all-files-documented.
Yev: could create compatibility problems (meaning of old field changes).
Gary: What about All files, some files, etc.

Yev: Annotations and creator comments could document this?
Gary: It's an unfortunate name….
Yev: Or full-contents-documented.
Gary: could say profile?   Yev: Should be independent of spec, no more than absolutely necessary.
Yev: Verification code, file information cardinality would need to change if profile used for files analyzed.
Gary: ok with Yev's solution of files analyzed set to false.
Yev: Can change name in 2.2
Gary: will create incompatibility - as a 3.0?
Kate: Yes, if incompatibility then 3.0
Gary: Ok to live until then, and will do annotation comment until then.
Thomas: Will work around, similar to cheating as Gary.   Behind package. Conflicts.

Gary: Looks like we need to clean up the wording description of Files-analyzed in 2.2 (allow including of files through relationship if files-analyzed is false) then look at changing the name for 3.0 or change values, yes, some, all or some varient.
Yev ok. Thomas ok. Kate ok.
From prior related discussion in May:
Yev: Make everything that is not explicit default to be NOASSERTION? And not be printed.
Thomas: Useful for reviewers indicate field that needs to be something looked at.
Gary: Translations, auto insert default values, create.
Yev: Some fudging - minimal SPDX?   Worth doing it.
Alexios: Tooling would need to generate it.
Yev: Backwards compatibility should be ok.
Gary: Verification would need to look and assume.   Translation might need to do it.
Yev: idempotent is desirable, want it to be self consistent.
Gary: Translators would need to populate it.   Delete them? Spreadsheet would have to be filled in with no assertion.   Policy on read - default.   If got default - don't write out?

← ended review on 2017/7/25 →

- Thomas: Include in Appendix 1 on top of the license list a reference that machine readable versions of list can be downloaded from SPDX GitHub org. Note need to make same change to spdx.org/licenses.
    - Thomas: In some places we refer to web site, others we don't need to make this consistent.   There are also some stale references to old SPDX repository.
    - Gary: There are several supported formatted - stored in license list github repository. SPDX.org/license - there is JSON, RDF, HTML template - official and stable.   The XML is used internally by legal team, to generate the official

supported output files. Eventually want to make it the output format.   Make copies of XML and license list.   People have referenced XML as a stable source, as its about to change.   Better to use stable output versions.   There are two places documented - license list data & repository.   Please log an issue if you have ideas on how to improve this.

- Thomas: Was looking at the stable ones.   Find the spec, and can't find them in machine readable format in github.   Machine readable versions should be published in own page on SPDX.org.
- Gary: Agrees document in SPDX spec itself.   Possibly copy the README into the spec.   Its better documented in technotes → move to spec.
- PDF documentation available at https://spdx.org/sites/cpstandard/files/pages/files/accessing_spdx_licenses_v-2.0.pdf
- Kate: 2.1.1 or 2.2?   Thomas: leaning towards 2.1.1
- Alexios: Do we need it in the spec, or just point to web site?
- Thomas: Link to web in spec, and more details on web page is ok.
- Gary: Clones of license list people refer to happen. Clearly need a better job.
- Kate: reach out to Jack to create new page?
- Gary: Lets move this to outreach team as to best way to represent this on web? Are Tech reports really working?   Making information more available.
- On SPDX license list - need to point to the output format, not the input format (github site).   So need to clear up.   Refer to license list data.
- Kate: Look at cleaning up text at top of license list page, then clone it into appendix for 2.1.1 -- Gary agrees.   Reach out to Jilayne to make
- Gary: Possibly look at retiring tech report(s), move things to github readme's. Topic for outreach.
- Current Version:
    - **Master Files**: The HTML pages you see here are generated from the master files for the SPDX License List. The master files include a spreadsheet listing all the licenses, deprecated licenses, and license exceptions; and the text for each license in a .txt file.

- Machine Readable Files (https://github.com/spdx/license-list-data) : These files are available by adding suffix (.json, .rdfa, ...
- Master Input Files: …   ---> Raw Source: ??
- Proposed added language: This website contains machine readable files both in RDFa and JSON formats. See the [Accessing the SPDX License List](https://github.com/spdx/license-list-data/blob/master/accessingLicenses.md) document for more information.
- General agreement we'll put this in 2.1.1 update.

        - Gary: Make visible JSON format available for licenses, either through appendix or ???
            - Will be handled as part of prior discussion.

<- August 1 -> From now on File Github issues now.   Work now in github issues

- Conditional License Expressions
    - Need more details.   David Wheeler email? To certain context at package summary level?
    - Follow up with Mark Gisi?
    - Thomas - if build in one environment then one licenses, source, etc. Conditional build spaces.   Linux Packages, Windows Package, etc.
    -
    -

Need to figure out:

- Which ones make sense and people care about?
    - Using green highlighting to indicate which there seems to be consensus around
- Who wants to drive spec proposal for each of the topic?

Tentative Date: June 1 proposal for features?