Bringing Functional Safety to the SBOM

SPDX Safety Profile Release Candidate

Elisa Workshop 19.11.2025

Nicole Pappler, AlektoMetis

Agenda

SPDX and its profiles

- What is SPDX
- SPDX core model and profiles

The Safety Profile in SPDX 3.1

- Motivation
- Model for SPDX 3.1
- Outlook

Whoami – Nicole Pappler

Founder and Safety Consultant at AlektoMetis

Professional History:
Been working in production maintenance, automotive, ECU software development
All my projects had some safety criticality
Started to focus on Functional Safety about 15 years ago

Currently:
Tech consulting as part of AlektoMetis
Supporting my customers regarding Functional Safety, Security & compliant use of open source
Involved in some open source projects:

- Zephyr (Functional Safety Manager)
- ELISA (Medical & Systems Group)
- FuSa for SPDX Profile Group
- OpenChain (3rd party certification with TÜV SÜD)

What else?
Contact handle at GitHub, Discord, etc: @nicpappler

About SPDX

System Data Package Exchange

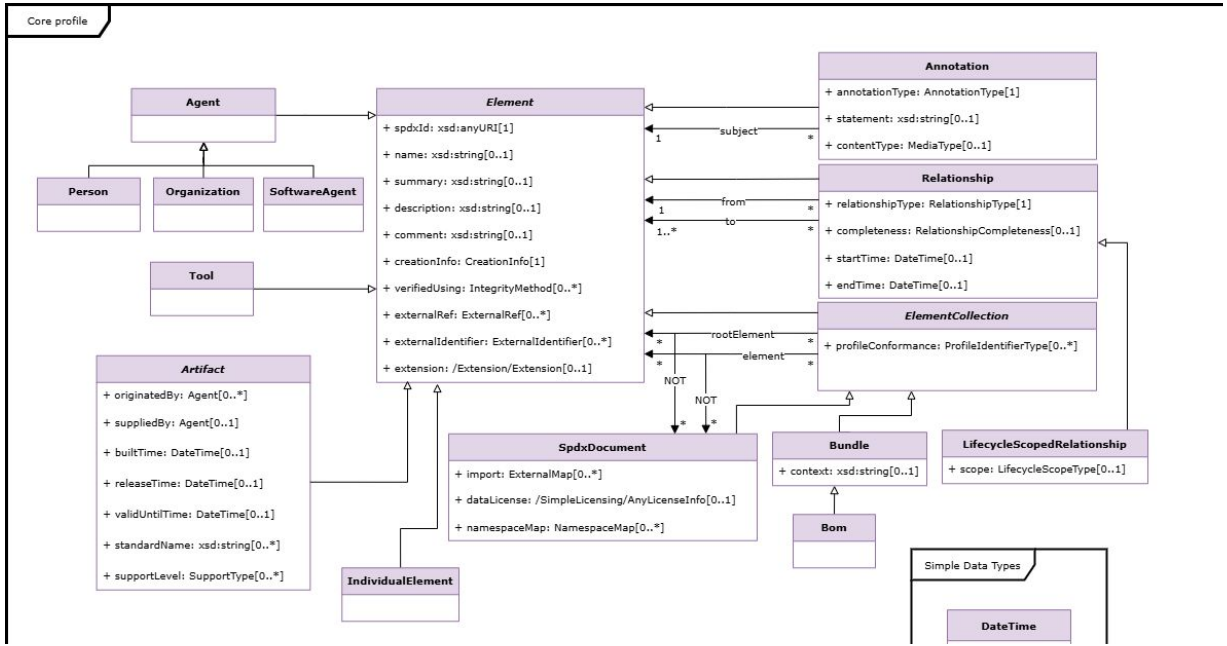
SPDX SAFETY

- Open standard, providing a format to describe software in both machine and human readable way
- Communicate SBOM information, including provenance, license, security, and other related information
- SPDX 2.3 -> ISO/IEC 5962:2021,
- SPDX 3.0 currently on the way to become and ISO/IEC standard
- SPDX Project consists of the
  - SPDX Specification,
  - SPDX License List, and
  - SPDX tools and libraries

SPDX

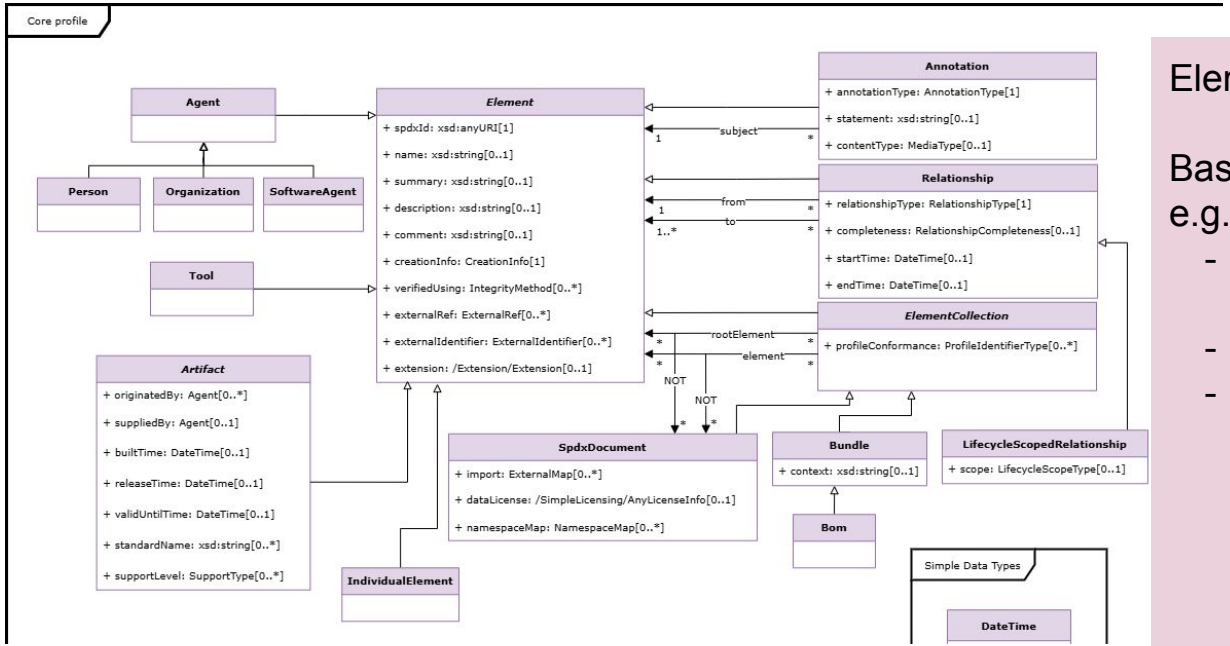
SPDX 3.0 Specification

Core



SPDX 3.0 Specification

Core



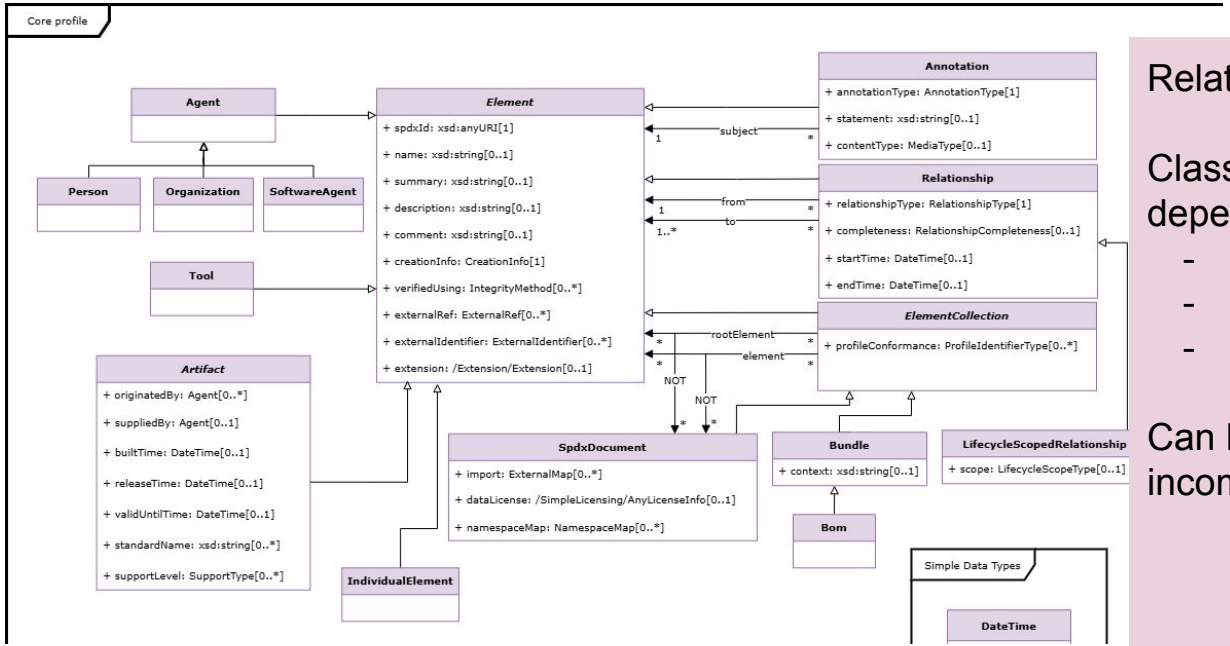
Element:

Basic class, includes e.g. information on

- Creation (who, when)
- ID, name
- description

SPDX 3.0 Specification

Core



Relationship:

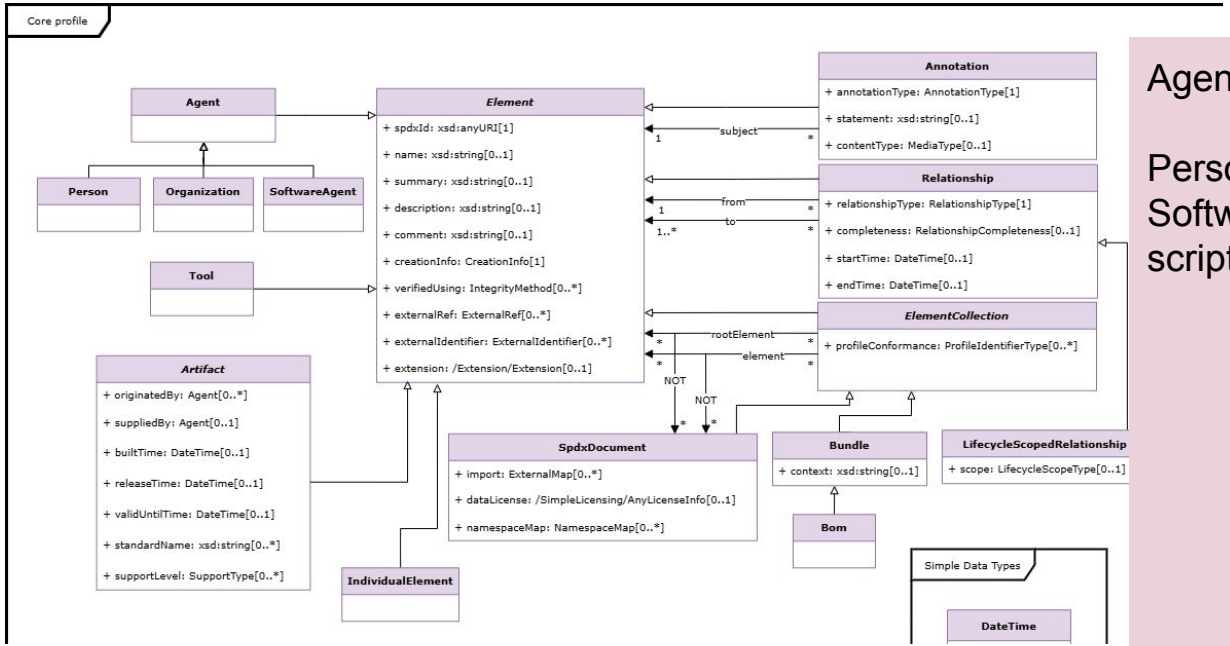
Class describing dependencies, like

- hasEvidence
- hasSpecification
- hasTest

Can be complete or incomplete

SPDX 3.0 Specification

Core



Agent:

Person, Organisation, SoftwareAgent (tool, script, etc.)

SPDX 3.0 Spec

Relationships

Core Enumerations

RelationshipType

Meta	
amendedBy	[Element -> Element]
describes	[Element -> Element]
modifiedBy	[Element -> Element]
other	[Element -> Element] (comment)
Structure	
contains	[Element -> Element]
Behavioral	
configures	[Element -> Element]
delegatedTo	[Element -> Element]
dependsOn	[Element -> Element]
Pedigree	
copiedTo	[Element -> Element]
expandsTo	[Artifact -> Artifact]
generates	[Artifact -> Artifact]
hasAddedfile	[Element -> Element]
hasDatafile	[Element -> Element]
hasDeletedfile	[Element -> Element]
Provenance	
ancestorOf	[Element -> Element]
availableFrom	[Element -> Element]
descendantOf	[Element -> Element]
variant	[Artifact -> Artifact]
Serialization	
serializedInArtifact	[SpdxDocument -> Artifact]
Build	
hasDependencyManifest	[Element -> Element]
hasDistributionArtifact	[Element -> Element]
hasDocumentation	[Element -> Element]
hasDynamicLink	[Element -> Element]
hasExample	[Element -> Element]
hasHost	[Build -> Element]
hasInput	[Build -> Element]
hasMetadata	[Element -> Element]
hasOptionalComponent	[Element -> Element]
hasOptionalDependency	[Element -> Element]
hasOutput	[Build -> Element]
hasPrerequisite	[Element -> Element]
hasProvidedDependency	[Element -> Element]
hasRequirement	[Element -> Element]
hasSpecification	[Element -> Element]
hasStaticLink	[Element -> Element]
hasTest	[Element -> Element]
hasTestCase	[Element -> Element]
hasVariant	[Element -> Element]
invokedBy	[Element -> Agent]
packagedBy	[Element -> Element]
patchedBy	[Element -> Element]
usesTool	[Element -> Element]
Licensing	
hasConcludedLicense	[SoftwareArtifact -> AnyLicenseInfo]
hasDeclaredLicense	[SoftwareArtifact -> AnyLicenseInfo]
Security	
affects	[Vulnerability -> Element]
doesNotAffect	[Vulnerability -> Element]
exploitCreatedBy	[Vulnerability -> Agent]
fixedBy	[Vulnerability -> Agent]
foundBy	[Vulnerability -> Agent]
hasAssessmentFor	[Vulnerability -> Element]
hasAssociatedVulnerability	[Artifact -> Vulnerability]
publishedBy	[Vulnerability -> Agent]
reportedBy	[Vulnerability -> Agent]
republishedBy	[Vulnerability -> Agent]
underInvestigationFor	[Vulnerability -> Element]
AI/Dataset	
hasEvidence	[Element -> Element]
testedOn	[Element -> Element]
trainedOn	[Element -> Element]

ExternalRefType

altDownloadLocation
altWebPage
binaryArtifact
bower
buildMeta
buildSystem
certificationReport
chat
componentAnalysisReport
documentation
dynamicAnalysisReport
eolNotice
exportControlAssessment
funding
issueTracker
license
mailingList
mavenCentral
metrics
npm
nuget
other
privacyAssessment
productMetadata
purchaseOrder
qualityAssessmentReport
releaseHistory
releaseNotes
riskAssessment
runtimeAnalysisReport
secureSoftwareAttestation
securityAdvisory
securityAdversaryModel
securityFix
securityOther
securityPenTestReport
securityPolicy
securityThreatModel
socialMedia
sourceArtifact
staticAnalysisReport
support
vcs
vulnerabilityDisclosureReport
vulnerabilityExploitabilityAssessment

HashAlgorithm

adler32
blake2b256
blake2b384
blake2b512
blake3
crystalsDilithium
crystalsKyber
falcon
md2
md4
md5
md6
other
sha1
sha224
sha256 [default]
sha384
sha512
sha3_224
sha3_256
sha3_384
sha3_512

AnnotationType

other
review

ExternalIdentifierType

cpe22
cpe23
cve
email
gitoid
other
packageUrl
securityOther
swhid
swid
urlScheme

RelationshipCompleteness

complete [default]
incomplete
noAssertion

LifecycleScopeType

build
design
development
other
runtime
test

ProfileIdentifierType

ai
build
core
dataset
expandedLicensing
extension
lite
security
simpleLicensing
software

PresenceType

no
noAssertion
yes

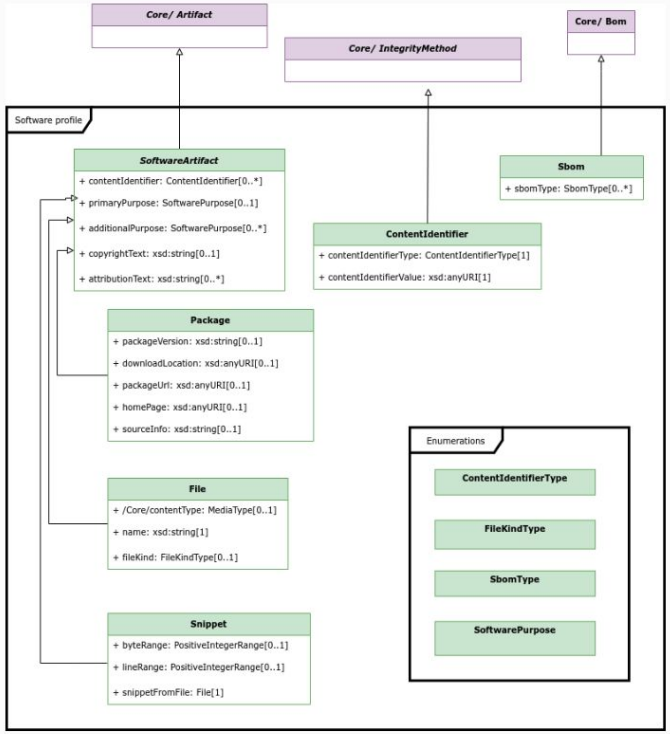
SupportType

deployed
development
endOfSupport
limitedSupport
noAssertion
noSupport
support

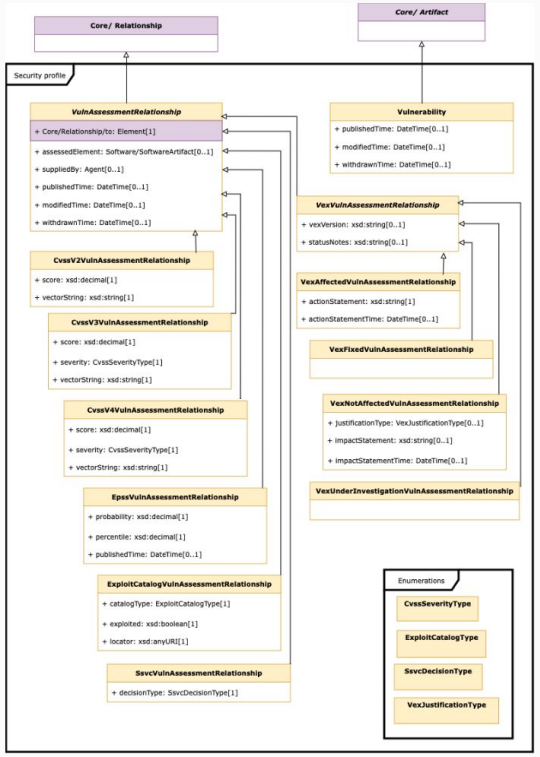
SPDX 3.0 Profiles

Examples

Software Profile



Security Profile



SPDX FuSa

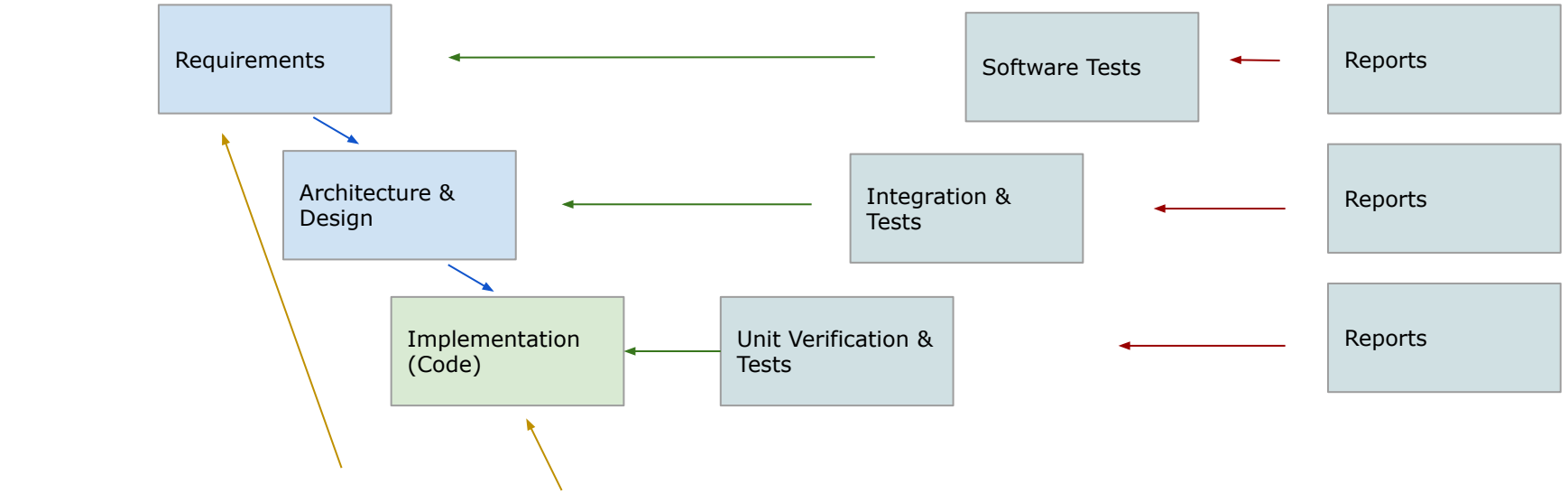
Goal:

To create a SPDX profile, based on SPDX 3.0 that enabled the delivery of the documents created in a safety lifecycle to enable the automation of building, exchanging and processing safety evidences

Use Cases:

- Generation of the Safety Case documentation
- Safety SBOM as exchange format in the supply chain
- Integrating the build of the safety documentation into the pipeline

Dependencies in a FuSa Project

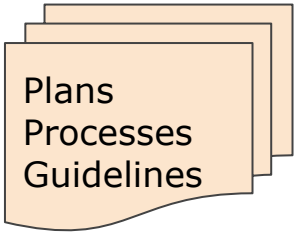


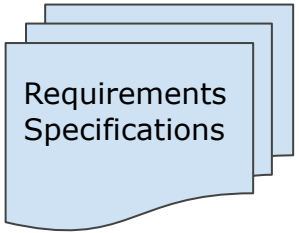
Functional Safety Management Plan	Requirements Management Plan	Configuration Management Plan	Documentation Management Plan	Component Qualification / Supply Chain	Validation & Assessment	Tooling Eval & Qualification (Dev, Verification, Build, Deploy…)

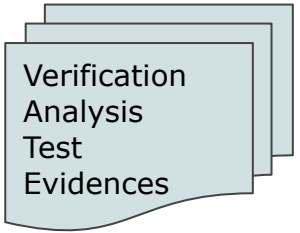
FuSa documentation structure

All FuSa related documentation is part of the Safety Case!

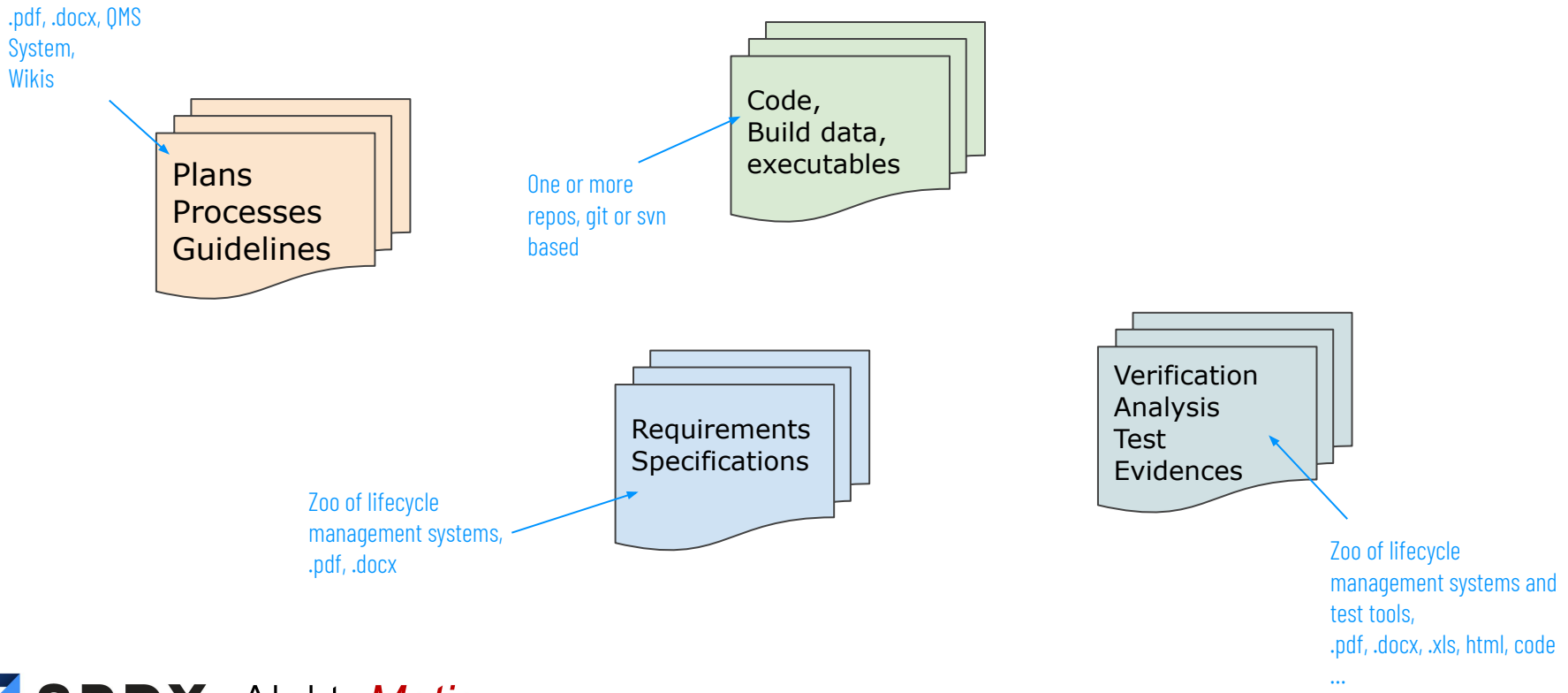
Think of all these documents as part of the release - each document is part of the Bill of Material, as is each screw, each microcontroller and each piece of software!



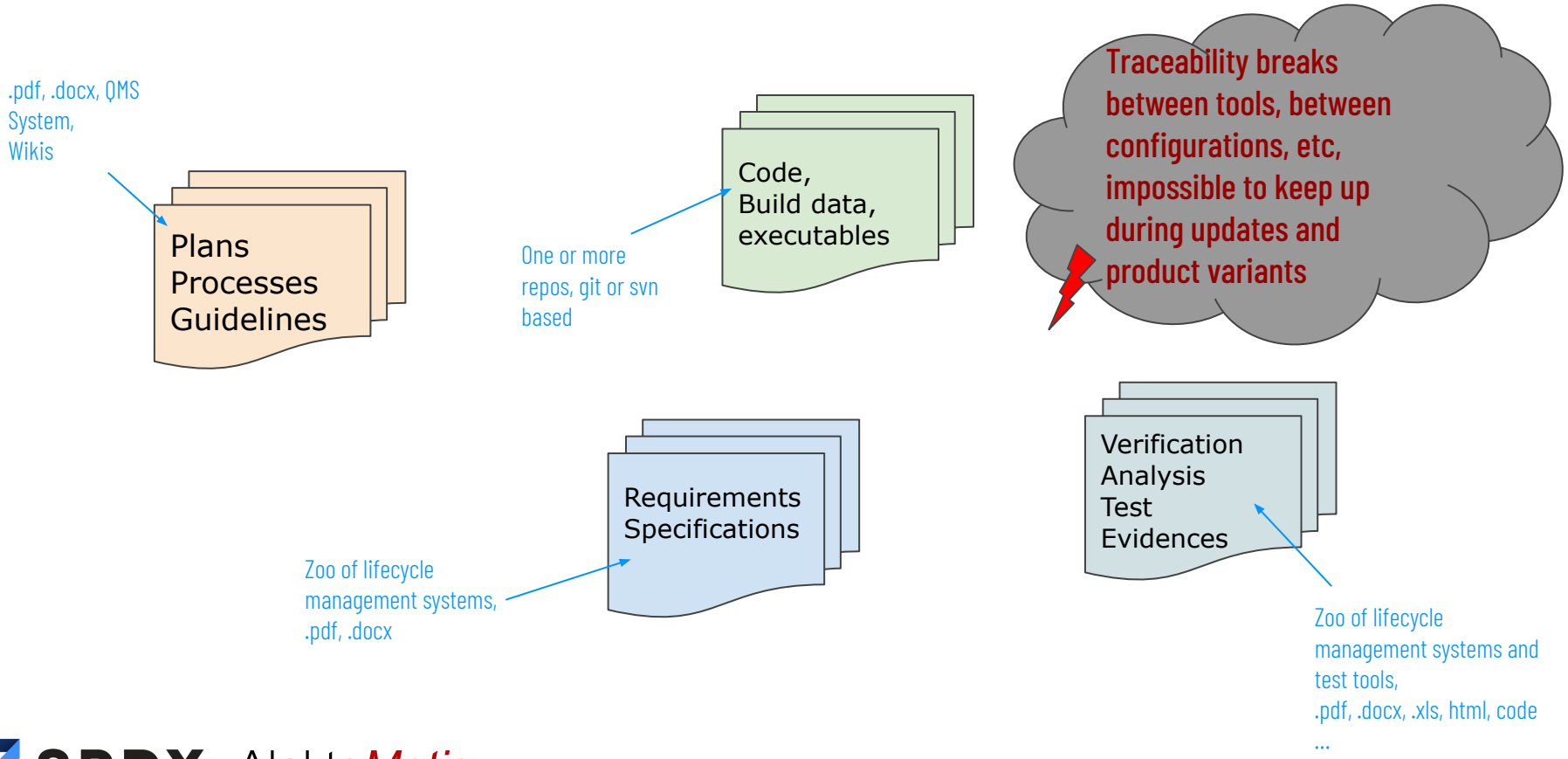




Data Structure of current FuSa projects…



Data Structure of current FuSa projects…



No 1 Safety Information Exchange Format

SPDX SAFETY

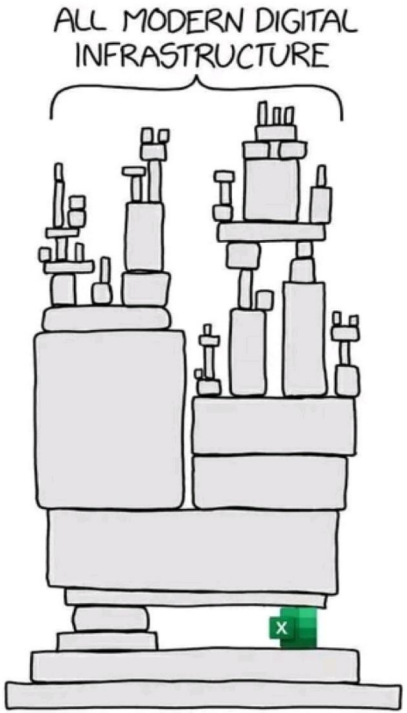
Any guesses????

SPDX AlektoMetis

Emoji by emojidex

No 1 Safety Information Exchange Format

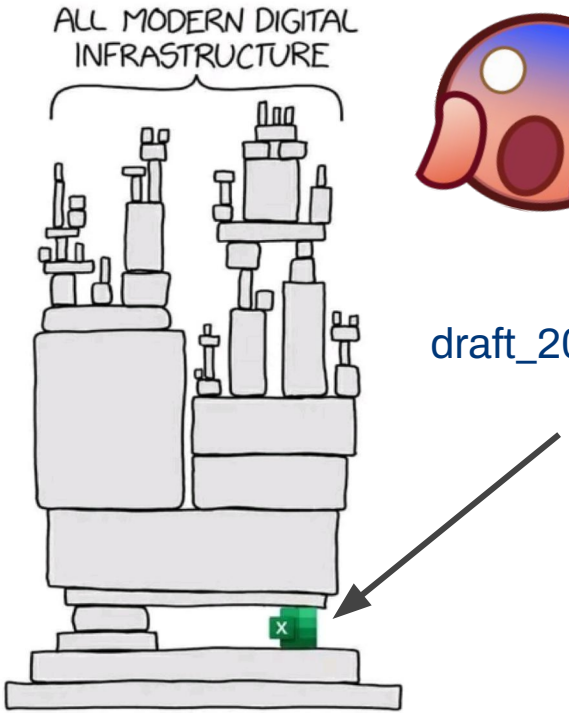
SPDX SAFETY



SPDX AlektoMetis

Emoji by emojidex

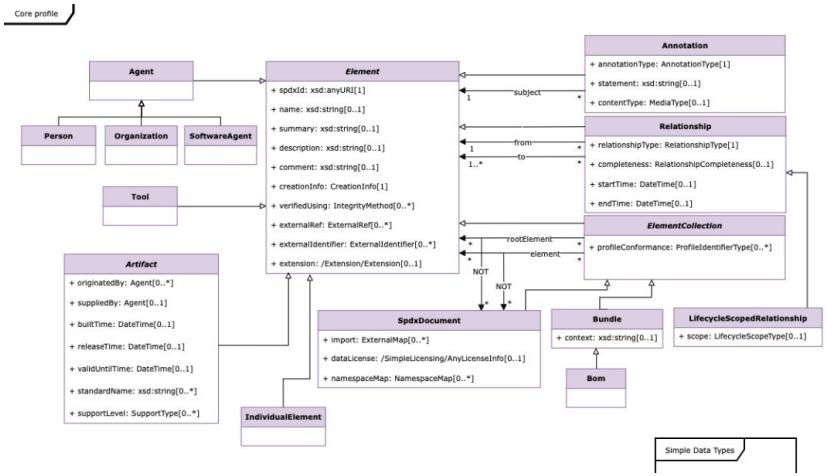
No 1 Safety Information Exchange Format

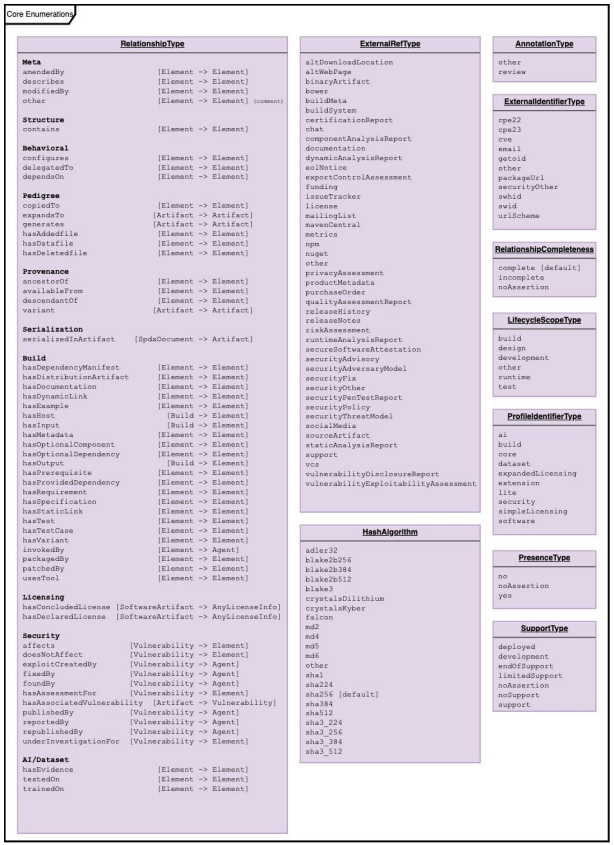


draft_2005TemplateSafetyCase_thisproject_final_forTraceingv06.xls

SPDX AlektoMetis

Emoji by emojidex





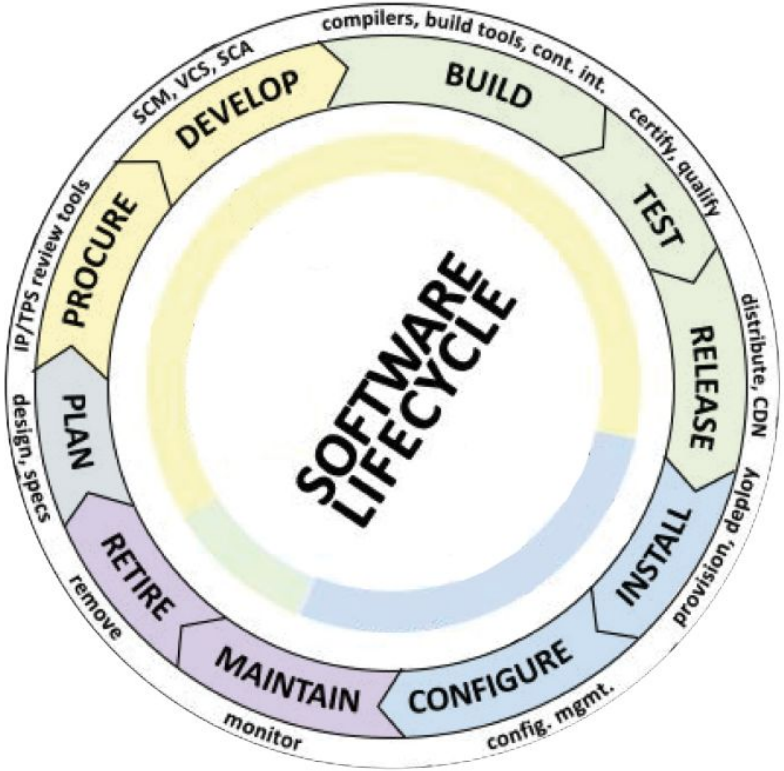
Generate SBOMS when the data is known - by the projects

SPDX SAFETY

Source SBOM

Design SBOM

Runtime SBOM



Build SBOM

Deployed SBOM

SPDX Alektoviews

Exchange SPDX Safety SBOMs

SPDX SAFETY

Evaluation & Implementation | **Build & Test** | **Final integrated system**

Design SBOM

Source SBOM

Build SBOM

Runtime SBOM

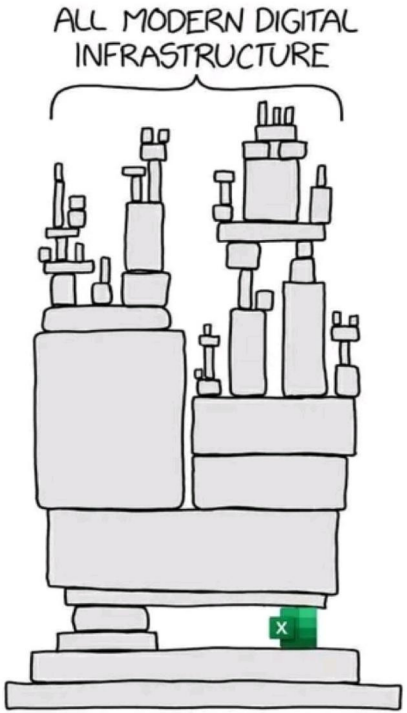
Deployed SBOM

SPDX AlektoMetis

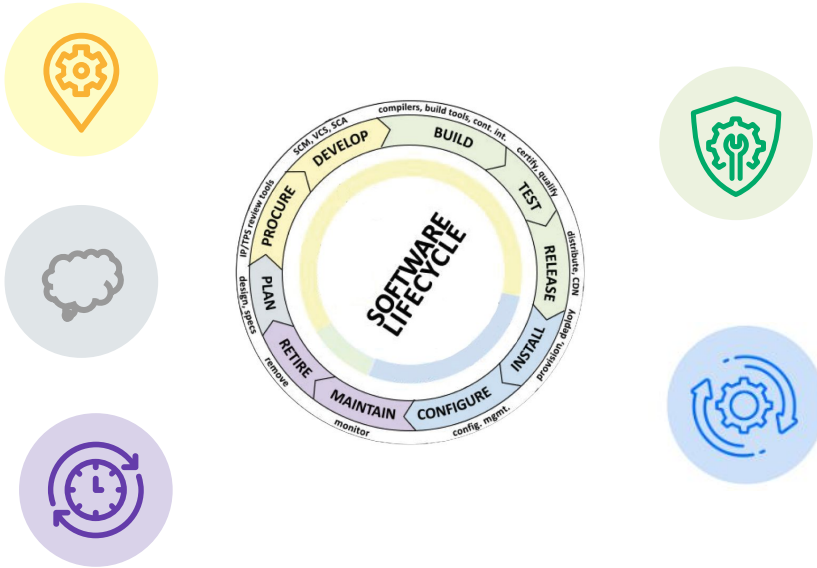
Safety Information Exchange Format?

SPDX SAFETY

SPDX SBOM



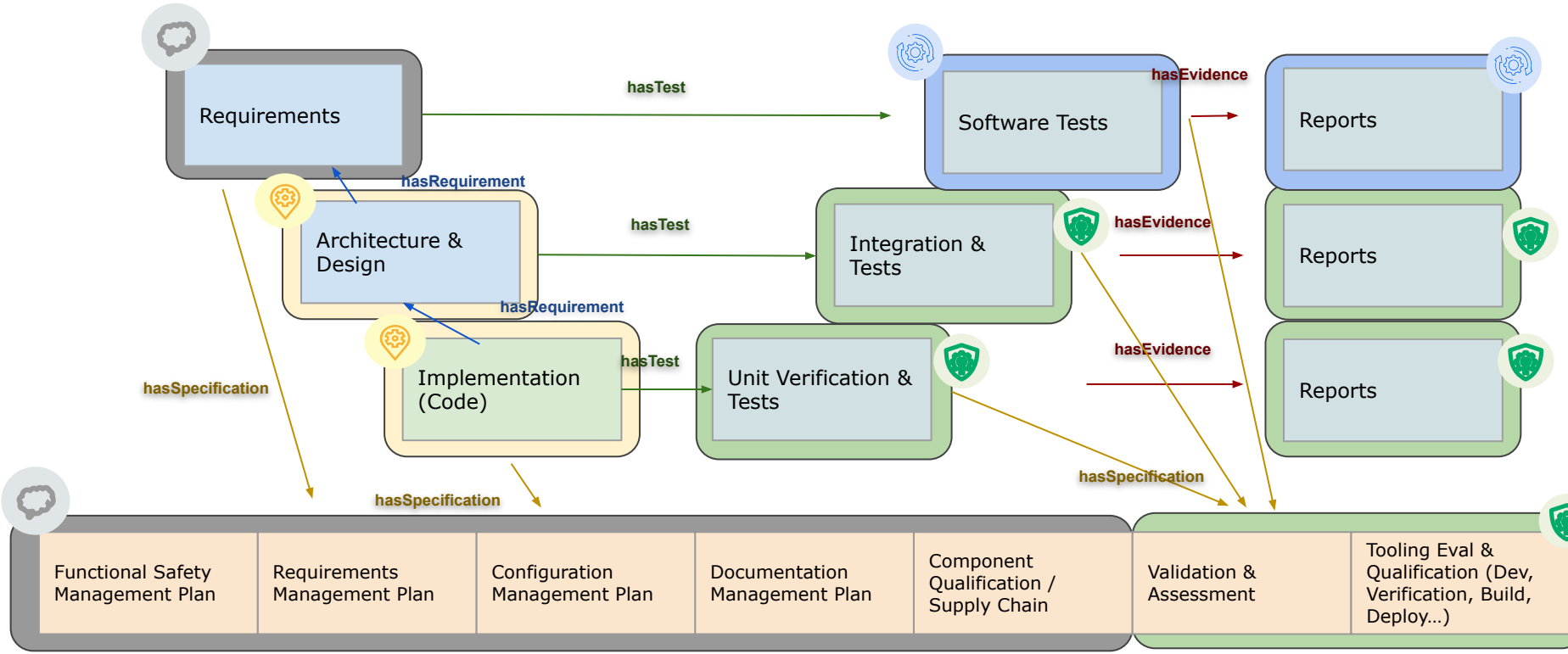
SPDX 3.1



... instead of inconsistent Spreadsheets, manual import/export of half decent ReqIFs...

SPDX AlektoMetis

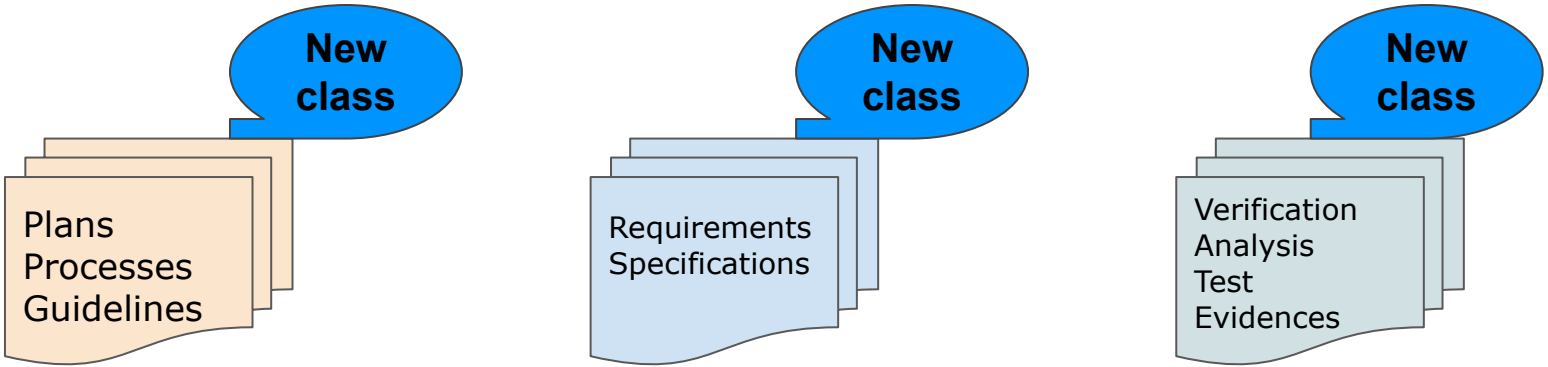
Dependencies in a FuSa Project*



*Using SPDX 3.0 relationships

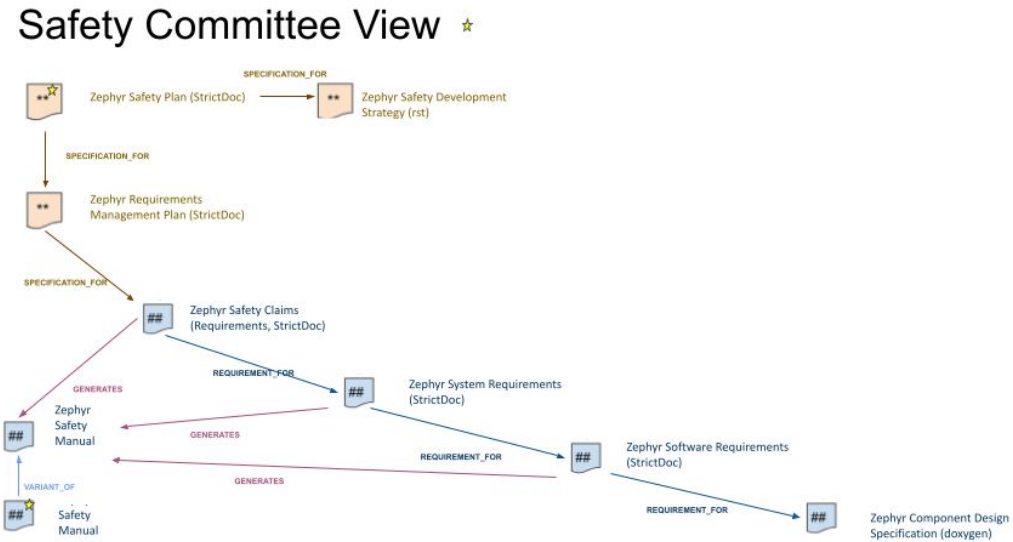
FuSa documentation structure

All FuSa related documentation is part of the Safety Case!
Think of all these documents as part of the release - each document is part of the Bill of Material, as is each screw, each microcontroller and each piece of software!



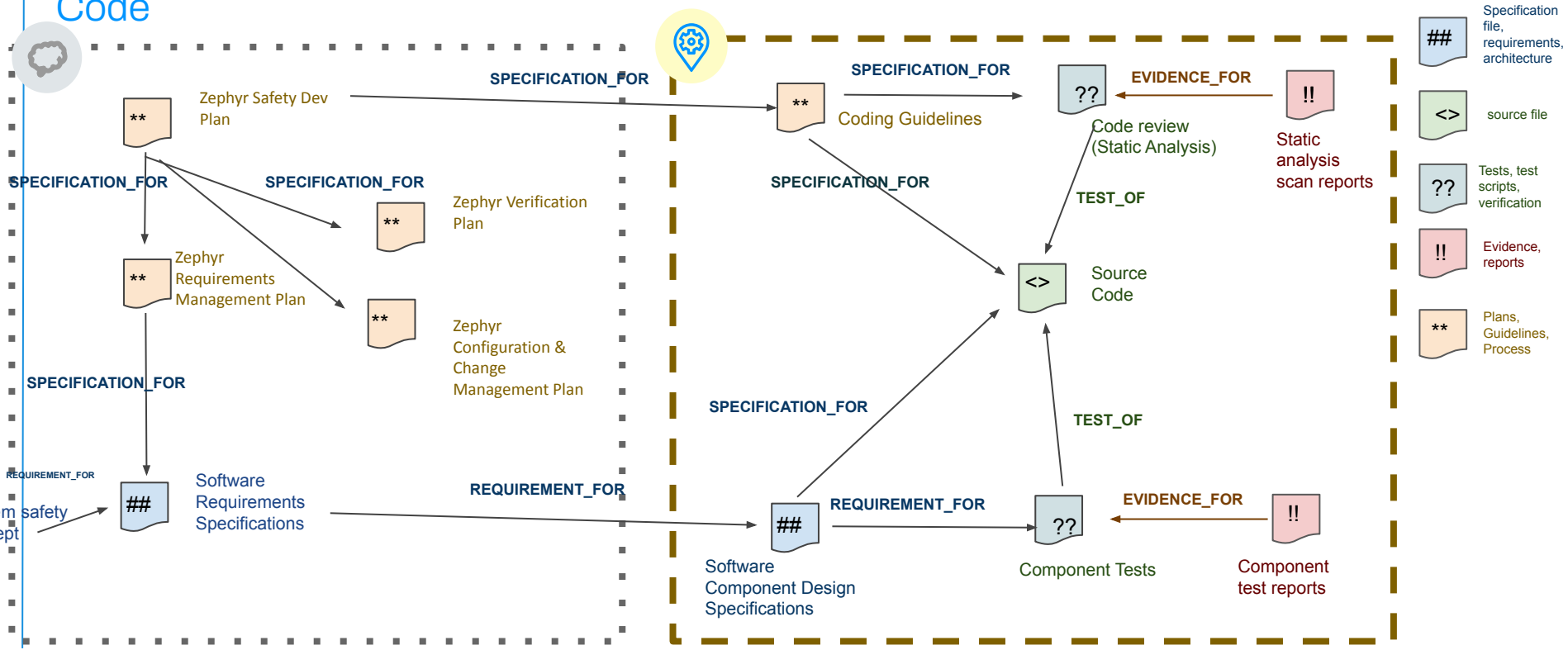
Zephyr Requirements Management

Requirements Management Knowledge Model



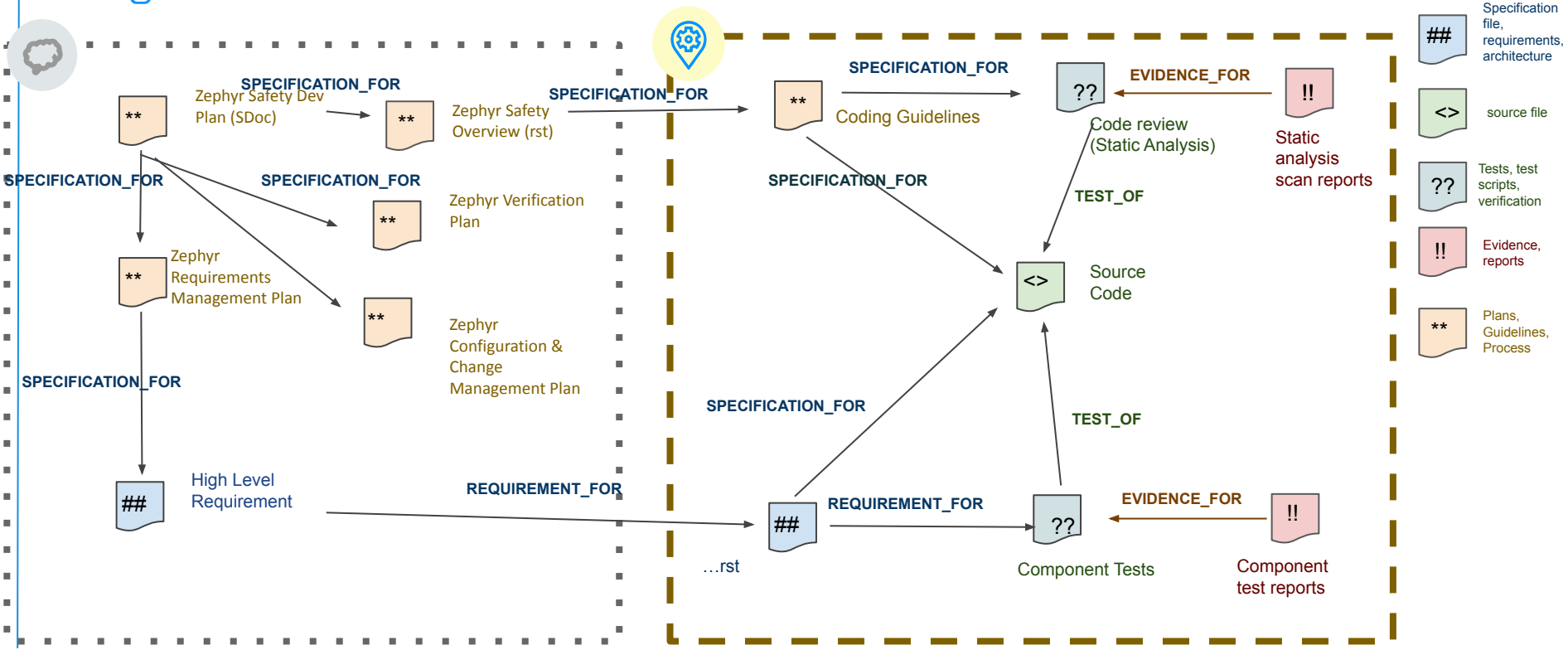
*prototyping with SPDX 2.3 relationships

Dependencies of Safety Plan, Safety Claim, Req, Design and Code



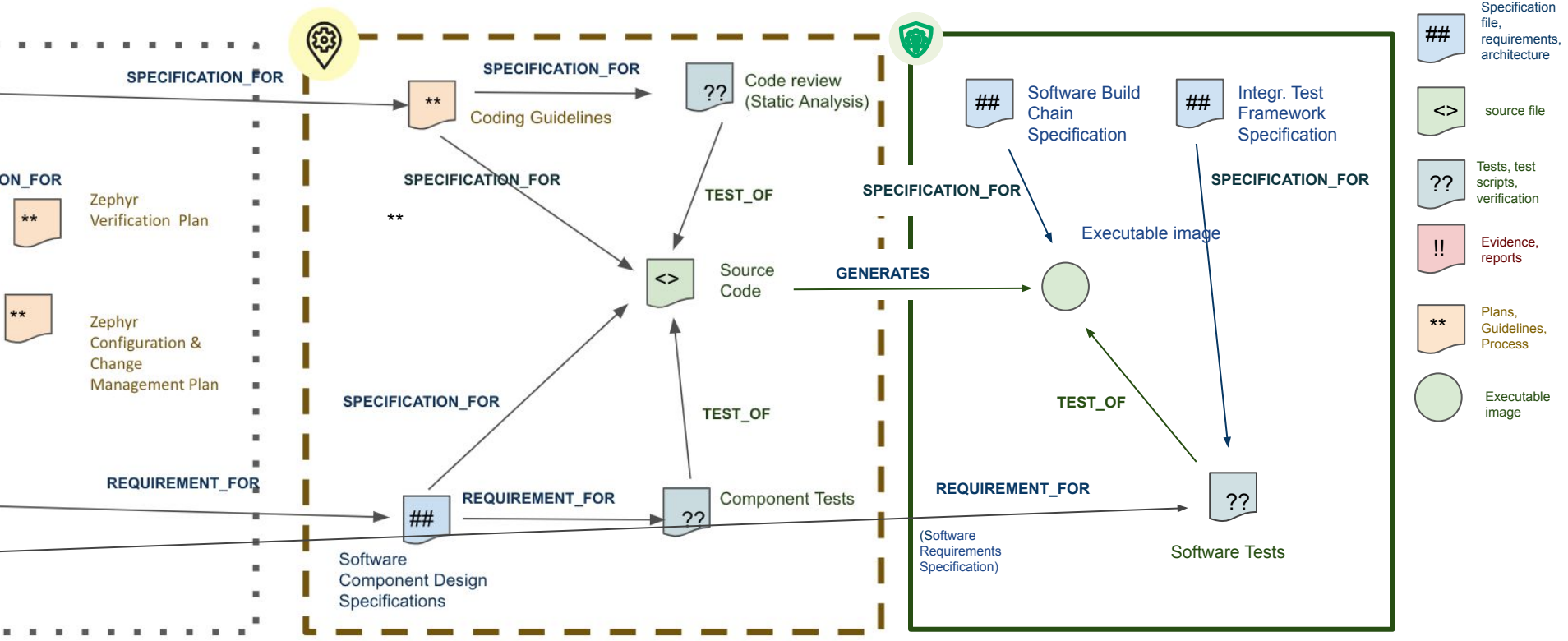
*prototyping with SPDX 2.3 relationships

Design SBOM to Source SBOM



*prototyping with SPDX 2.3 relationships

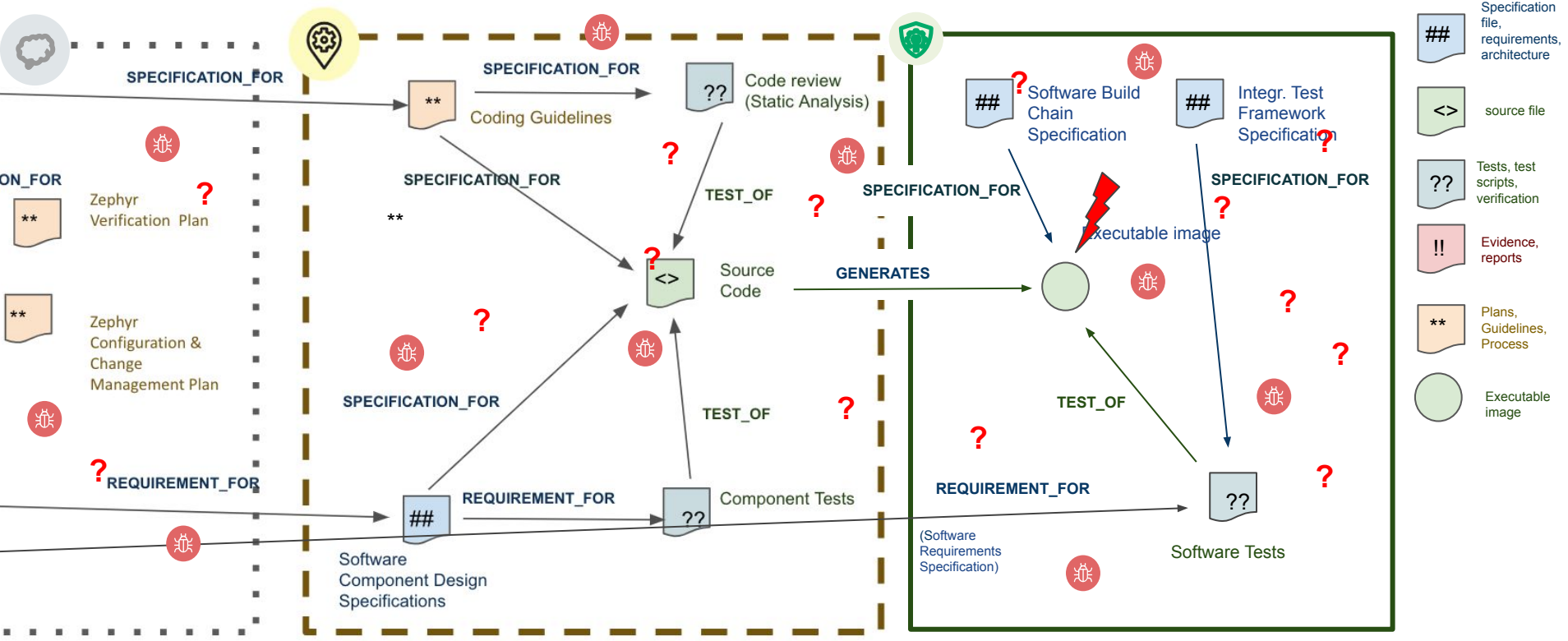
Source SBOM to Build SBOM



*prototyping with SPDX 2.3 relationships

Dependency Identification on Component Level

SPDX SAFETY

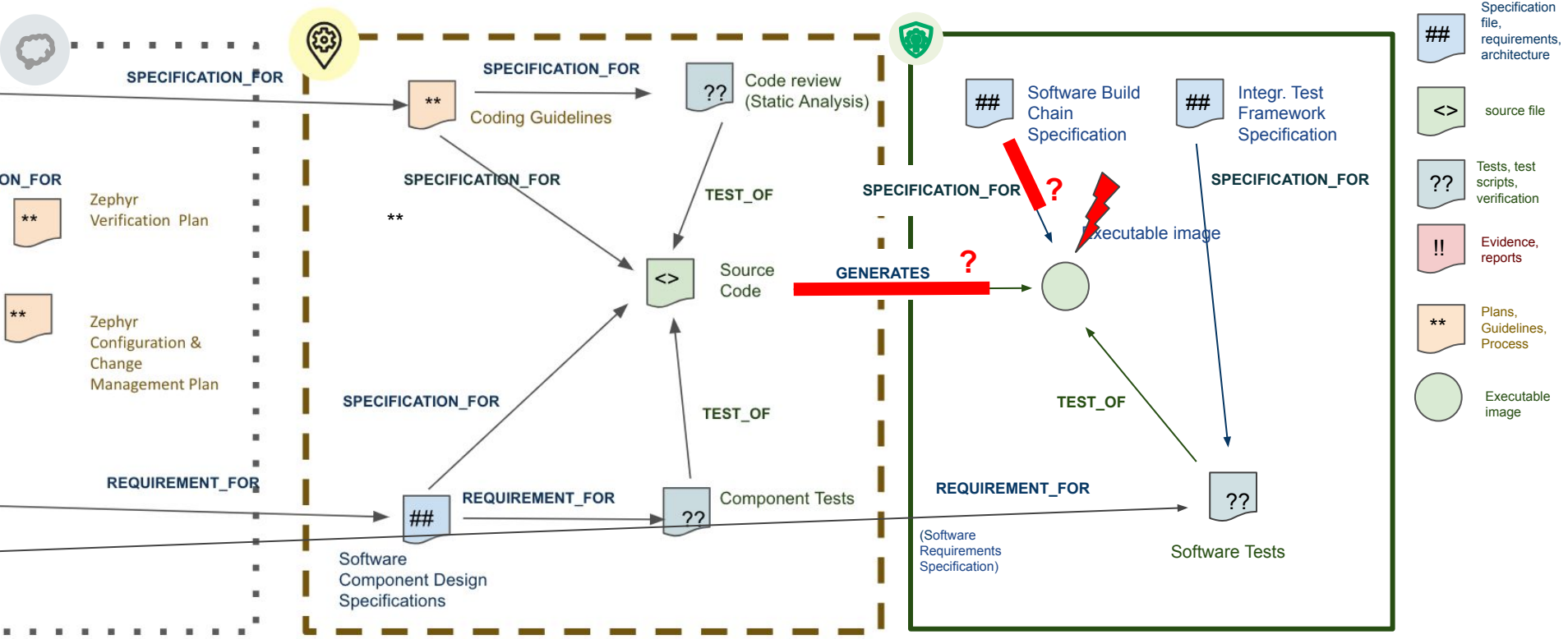


SPDX

*prototyping with SPDX 2.3 relationships

Dependency Identification on Component Level

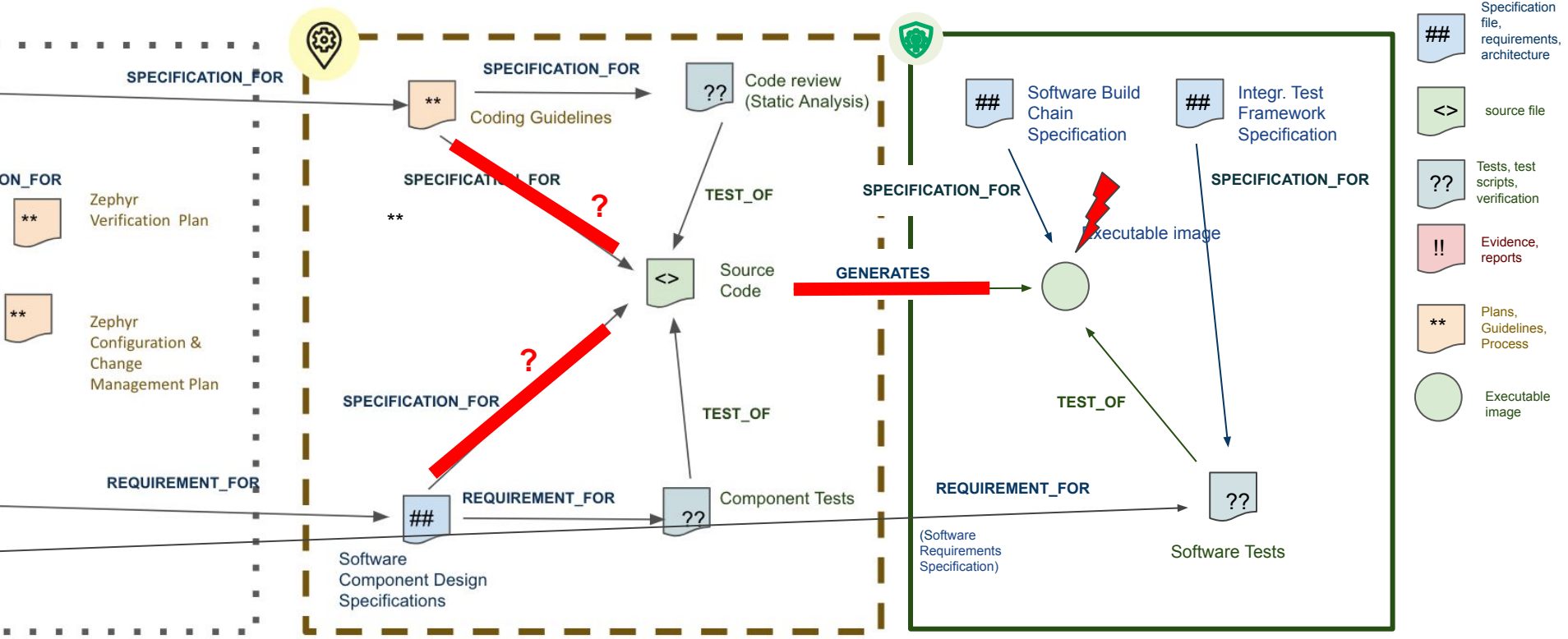
SPDX SAFETY



SPDX

Dependency Identification on Component Level

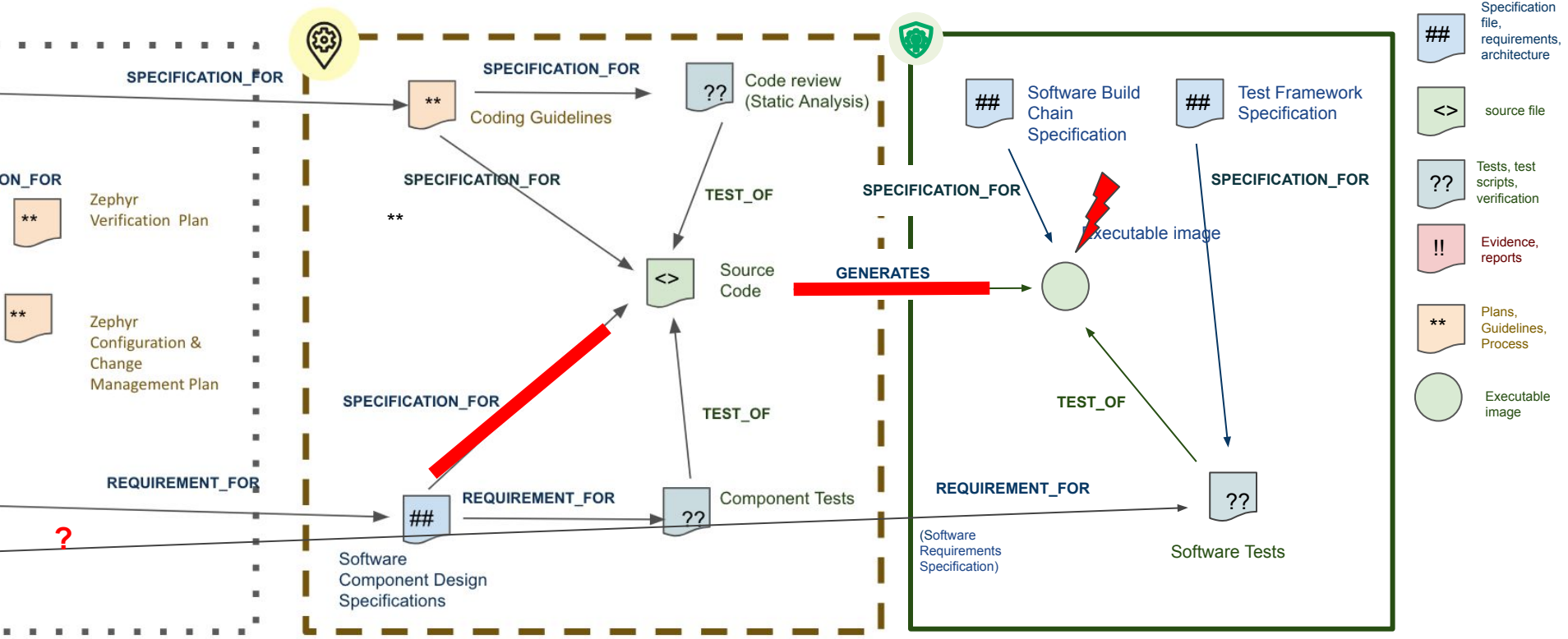
SPDX SAFETY



SPDX

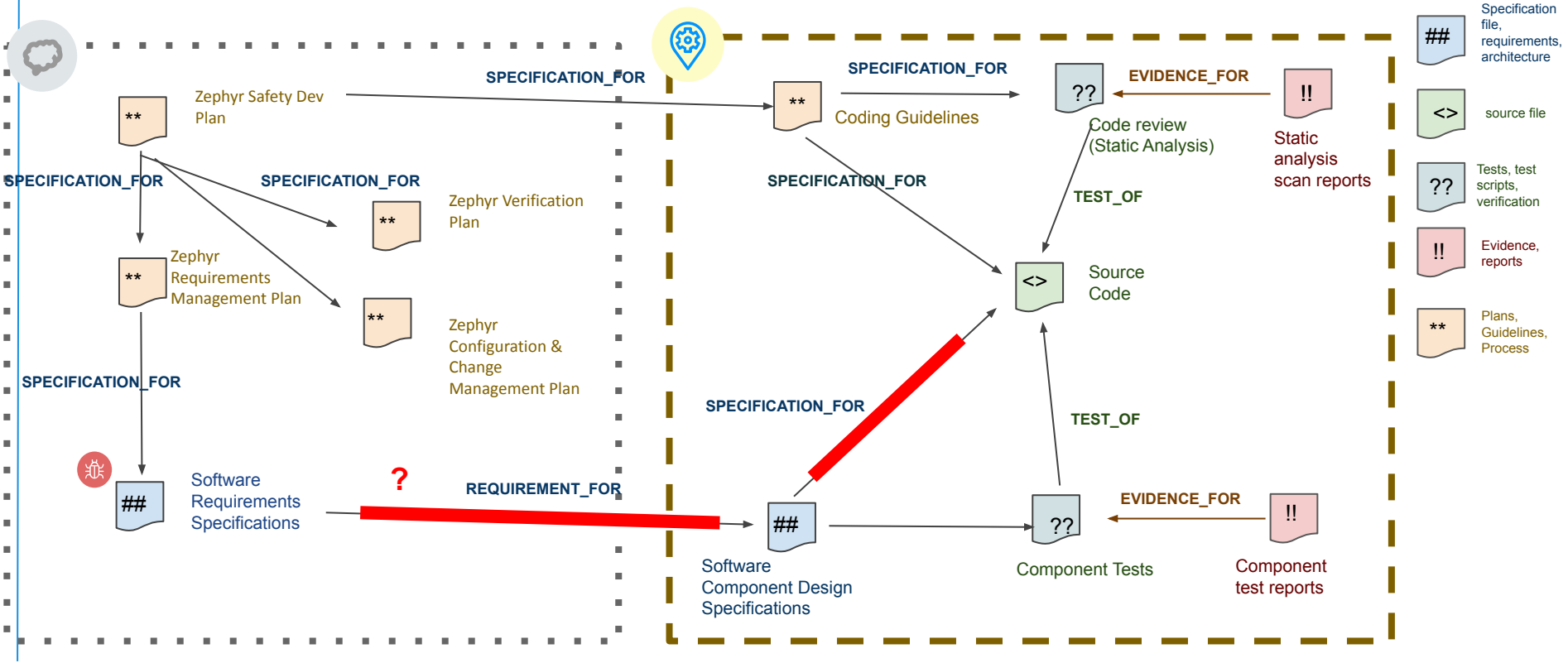
Dependency Identification on Component Level

SPDX SAFETY



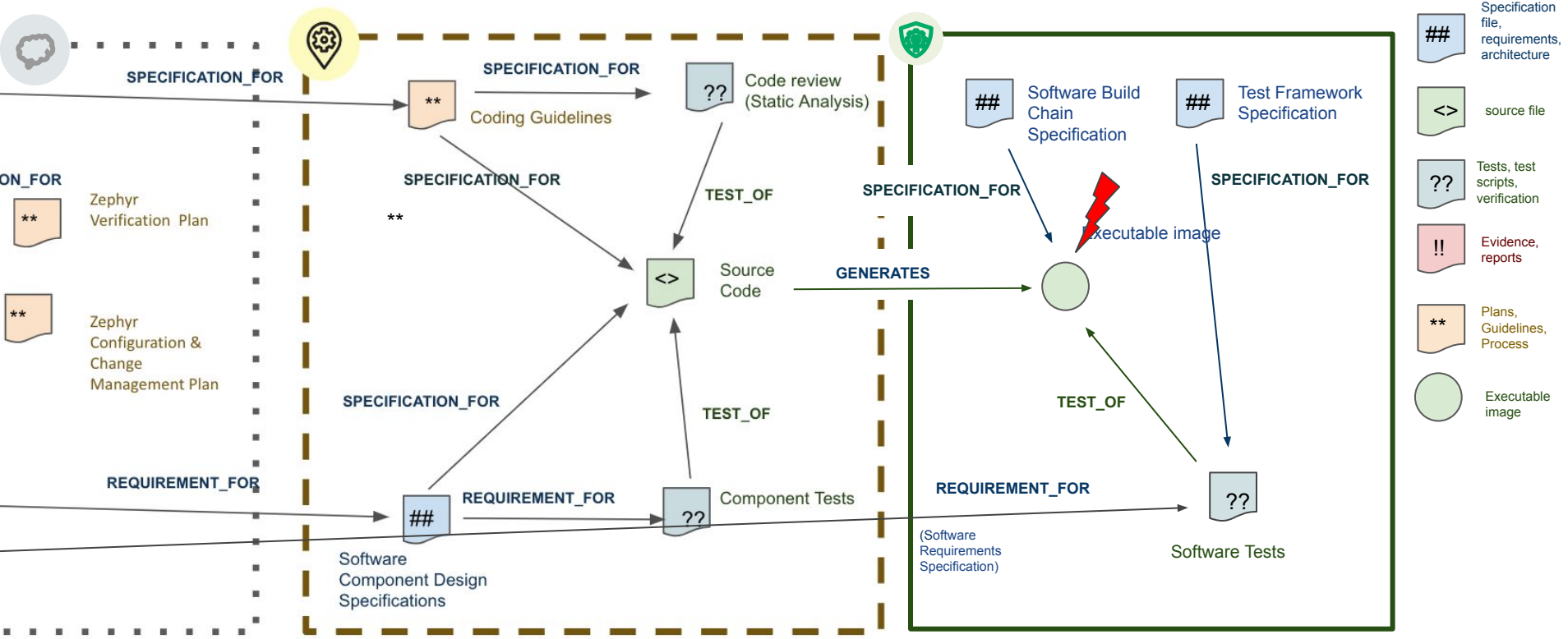
SPDX

Dependency Identification on Component Level



Dependency Identification on Component Level

SPDX SAFETY



SPDX

Content for SPDX 3.1

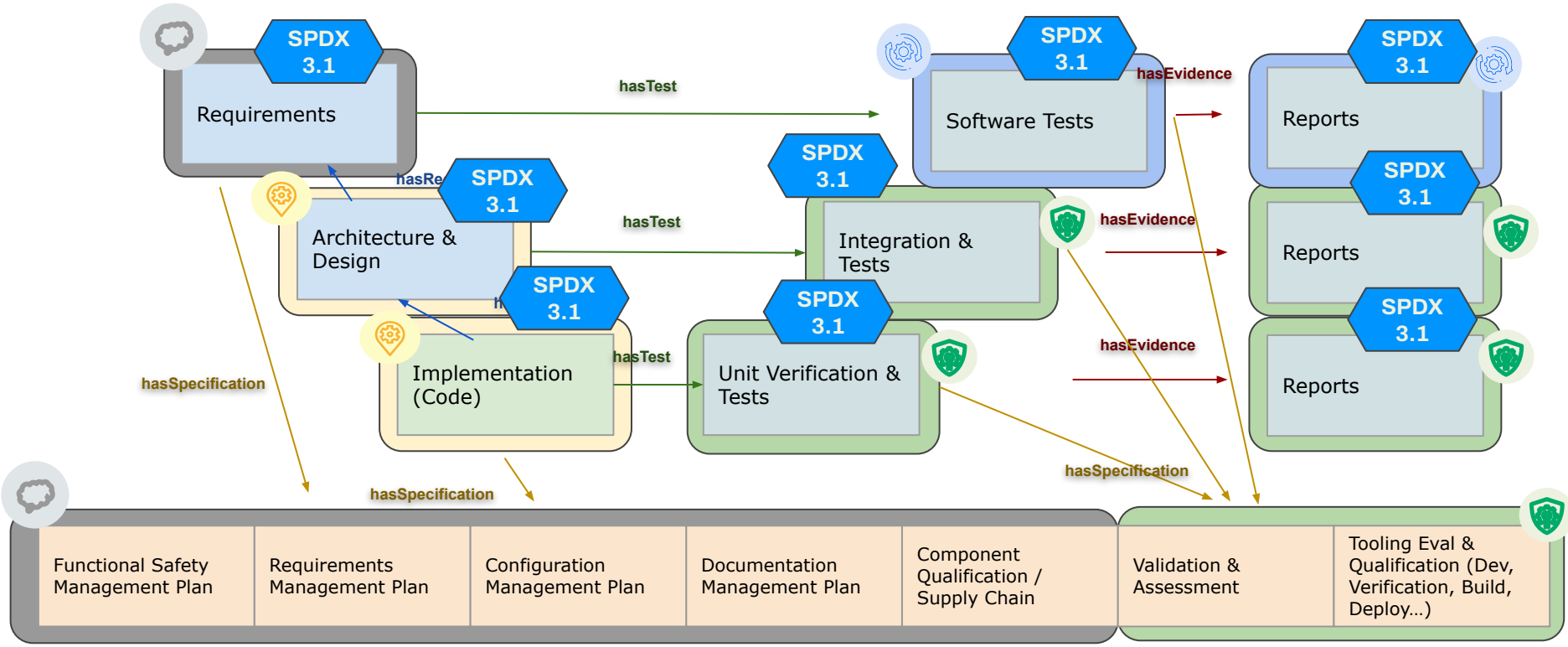
SPDX SAFETY

- Requirements Class
- Verification Class
- traceToDetail entry to Relationship Types to connect hierarchies of requirements
- Evidence Class & new Relationship Class for evidences

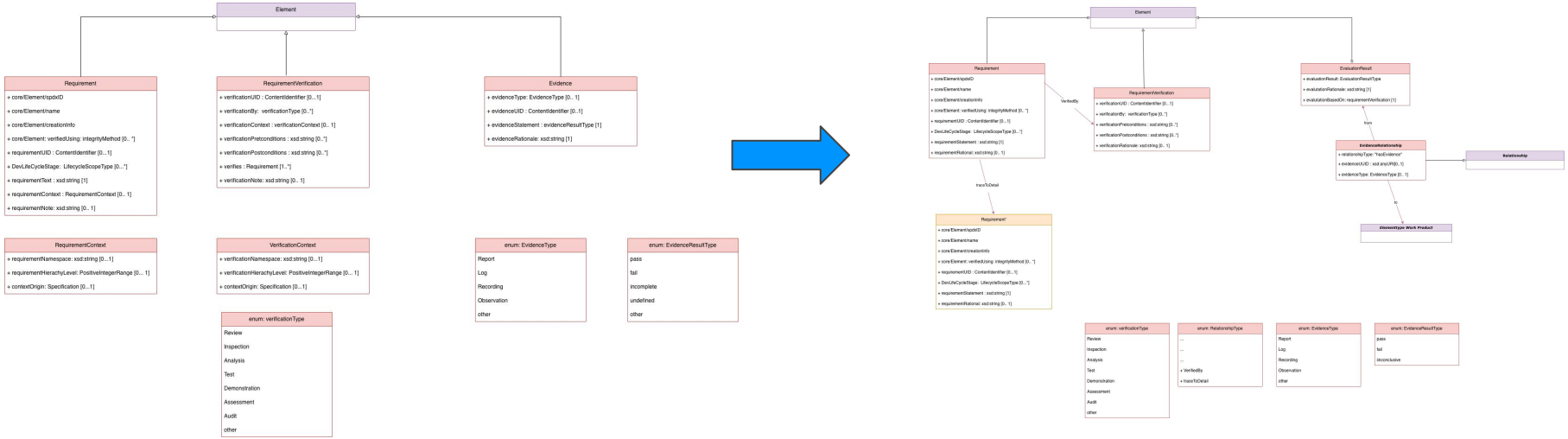
SPDX

Dependencies in a FuSa Project

RC SPDX 3.1

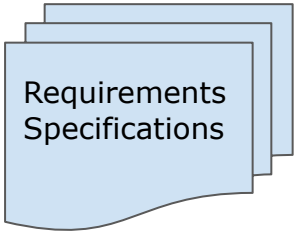


Recent updates on the model



Classes for WPs - REQUIREMENT

RC SPDX 3.1

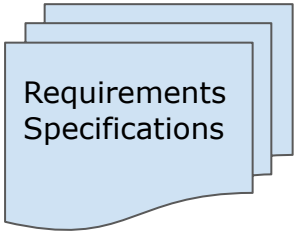


Determining factors and assumptions:

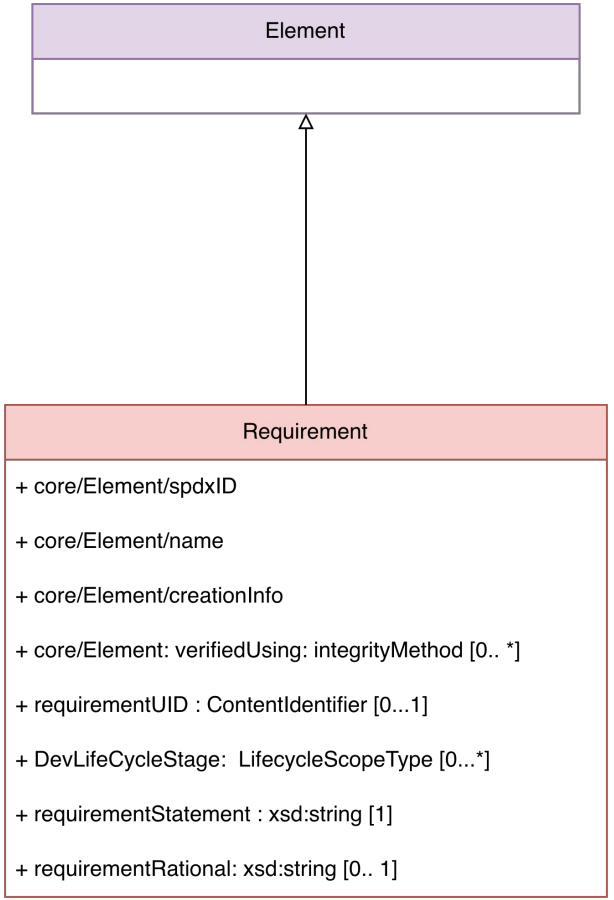
- A requirement describes a functional, non-functional or design need placed on an item (HW, SW, system, whatever can be the product)
- There are different sources of requirements
- Atomic REQUIREMENTS entities can be packaged to Requirement sets that then can become part of specifications ⇒ no new class needed, use existing SPDX functionality to bundle requirements to represent specifications

Classes for WPs - REQUIREMENT

RC SPDX 3.1

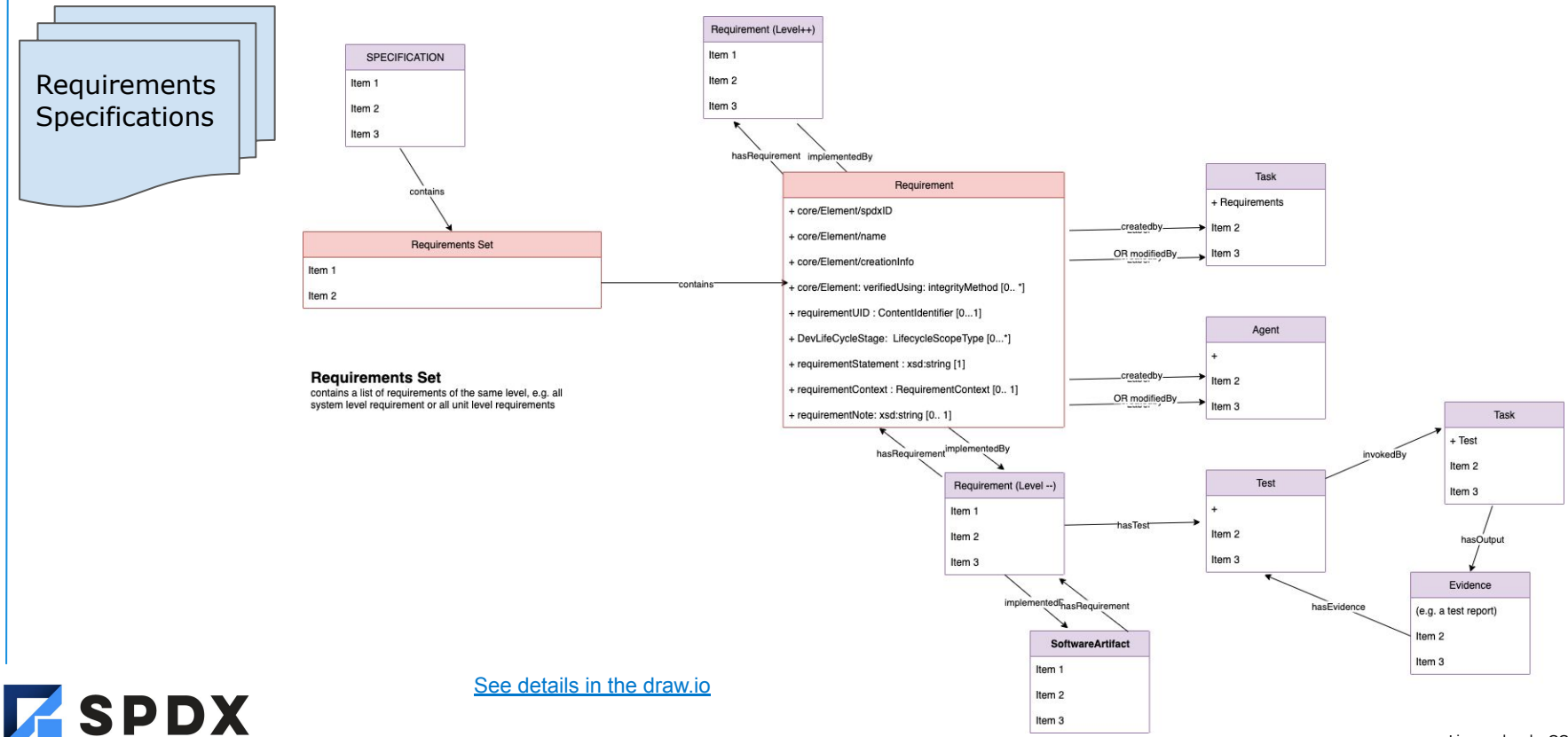


REQUIREMENT class



REQUIREMENT - relationships

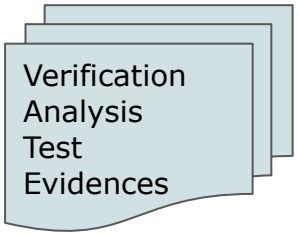
Not all in 3.1



[See details in the draw.io]

Classes for WPs - VERIFICATION

RC SPDX 3.1



Determining factors and assumptions:

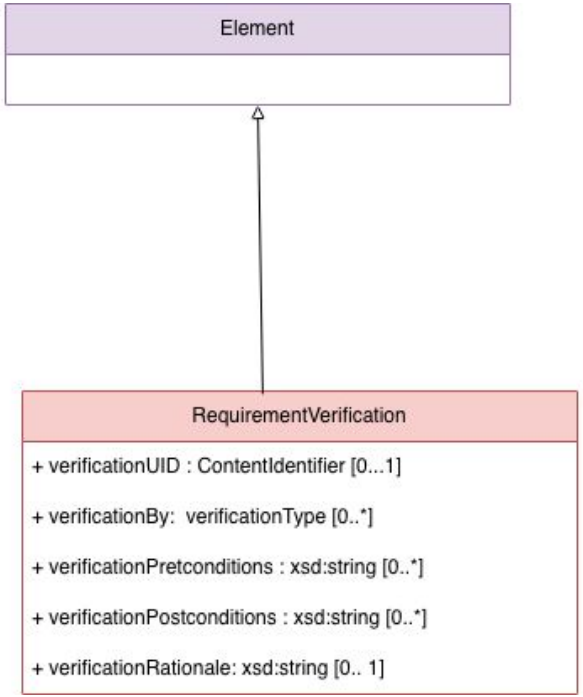
- There are different types of verifications, eg.
  - Test
  - Review/Inspection
  - Analysis
  - Demonstration
- Verification means we have a PROCESS how to do VERIFICATION and some evidence that this verification was performed and what were the environmental and runtime conditions of these tests
- While the verification PROCESS is a process that can be defined using the PROCESS class, a test case/suite/checklist looks very much like a REQUIREMENT, but not exactly

⇒ need class for VERIFICATION specification to have something that describes test cases

Classes for WPs - VERIFICATION

RC SPDX 3.1

Verification
Analysis
Test
Evidences

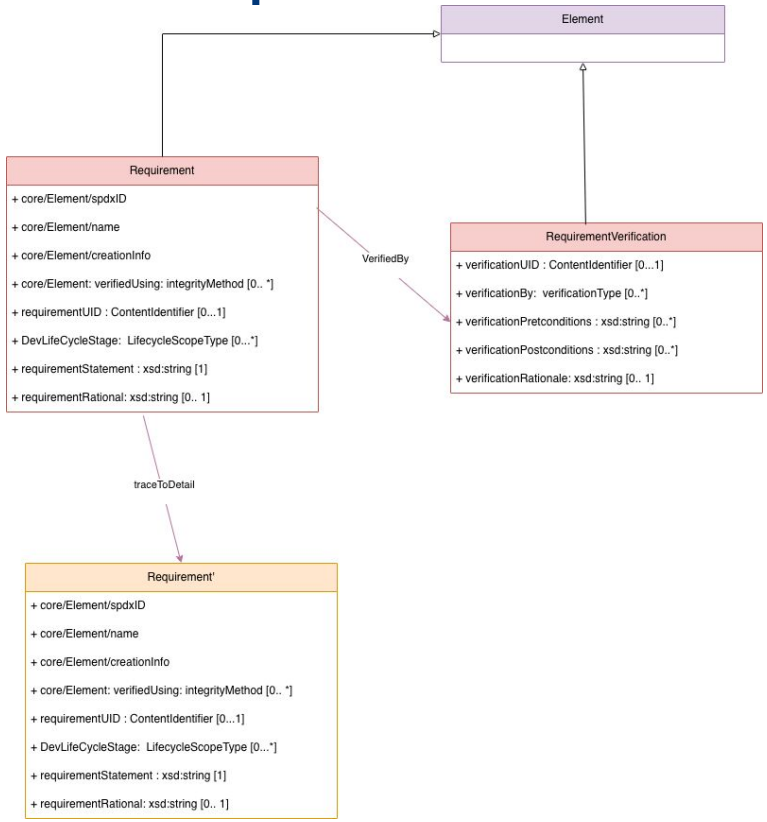


enum: verificationType
Review
Inspection
Analysis
Test
Demonstration
Assessment
Audit
other

Req and Ver - Relationships

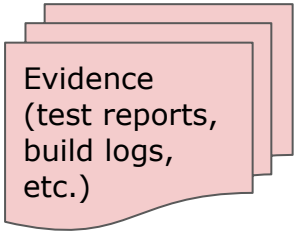
RC SPDX 3.1

Requirements Specifications



Classes for WPs - EVIDENCE

RC SPDX 3.1



Determining factors and assumptions:

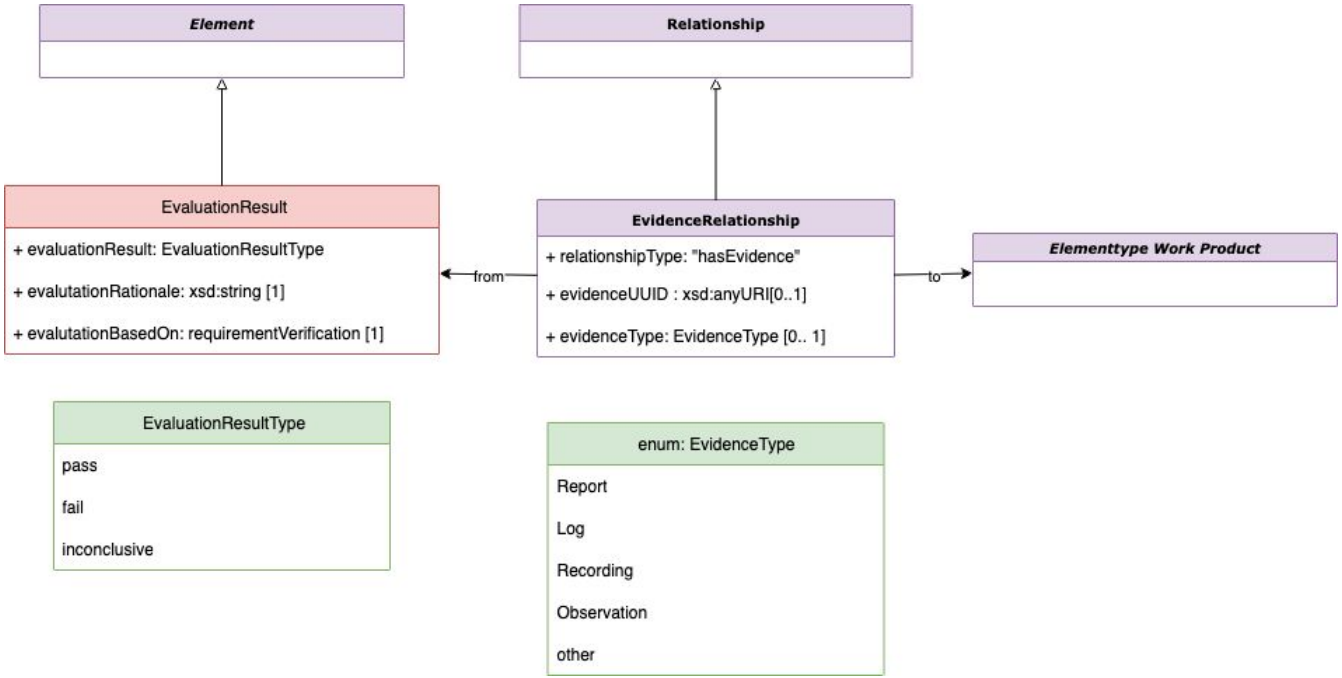
- EVIDENCEs are created based on a PROCESS with a ProcessType for verification
- EVIDENCEs are created applying TASKs by an Agent
- EVIDENCES attest a certain level of compliance of
  - a tested item (code) with its acceptance criteria (requirement), using the test process and
- EVIDENCES are highly coupled with VERIFICATION

Classes for WPs - EVIDENCE

RC SPDX 3.1

Evidence (test reports, build logs, etc.)

EVIDENCE class



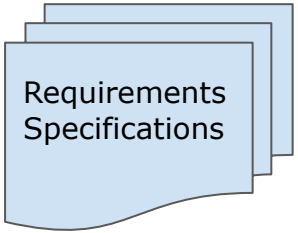
A few things in the pipeline…

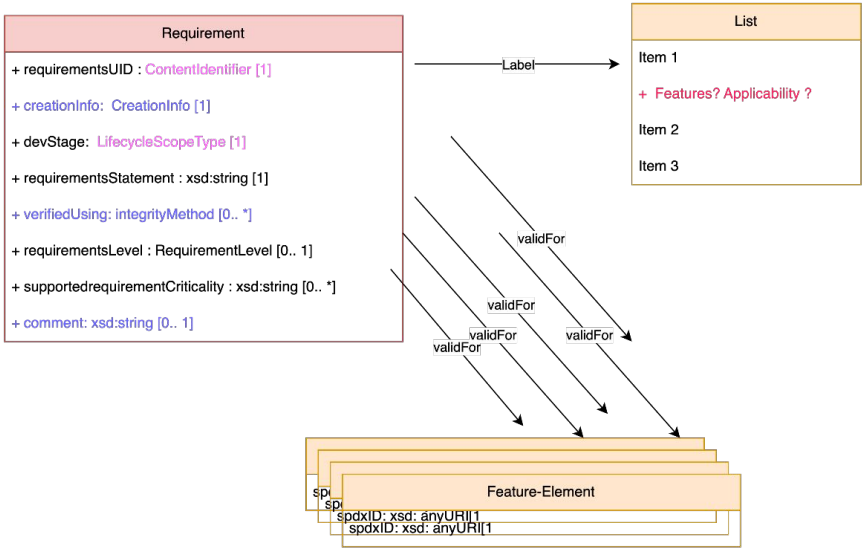
Pushed to SPDX 3.2

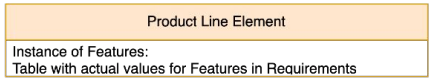
- Product line engineering - Product configuration, calibration etc.
- Task and Process
- Agent: Types, Qualifications etc.

REQUIREMENT & product line engineering

Pushed to SPDX 3.2







Configuration Element

Comes from the software artifact, defines the actual configuration of feature elements

Example:

Requirement 1: The VEHICLE_TYPE vehicle with ENGINE_TYPE shall have a TRANSMISSION_SYSTEM between engine and wheels

Requirement 2:

Features-applicability FA1:

VEHICLE_TYPE: small; medium, large

Features-applicability FA2:

TRANSMISSION_SYSTEM: manual, automatic, fixed

Feature-applicability FA3:

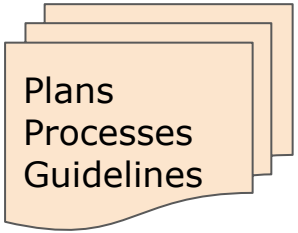
BEV, PHEV, FUEL

Product Configuration 1:

SuperNewVehicle: FA1:small, FA2:fixed, FA3:BEV

Classes for work products - TASK and Process

Pushed to SPDX 3.2



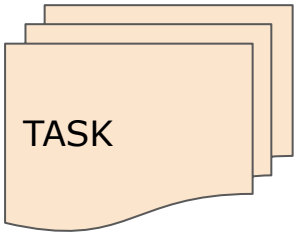
Determining factors and assumptions:

- we can generalize what it is a plan, process or guideline as a set of things, that can be done/performed ⇒ TASKs
- A PROCESS is a list of TASKs,
- While a PLAN will look very much like a PROCESS, a PLAN is a project specific instantiation of a PROCESS
- There are a lot of PROCESS types, e.g. for software it can be things like, dev-process, test-process, build-process, assessment-process etc
- In the definition of IEC 61508, there are requirements for systematic capability, which in the engineering reality translate to process and methods, and therefore these “requirements for systematic capability” -

TASK class includes everything process, plan or guideline, as these types of documents always look the same

Other classes - TASK

Pushed to SPDX 3.2

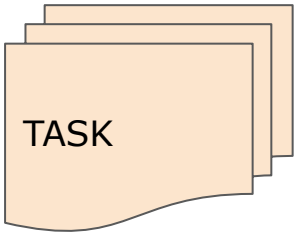


Determining factors and assumptions:

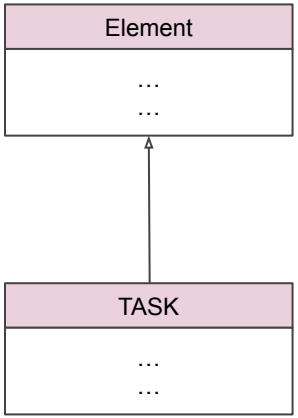
- A TASK is a specific unit of work that contributes to the completion of a project or an item
- TASKs can be of different types
- Need at least the content of ELEMENT
- An TASK as a minimum needs:
  - An Objective what the tasks aims to achieve
  - Preconditions and necessary inputs, resources
  - An Agent (human, tool) with assigned ROLE
  - Completion Conditions, Definition of Done
  - Optionally an environment and its configuration needed to perform the task
  - Optionally supporting information

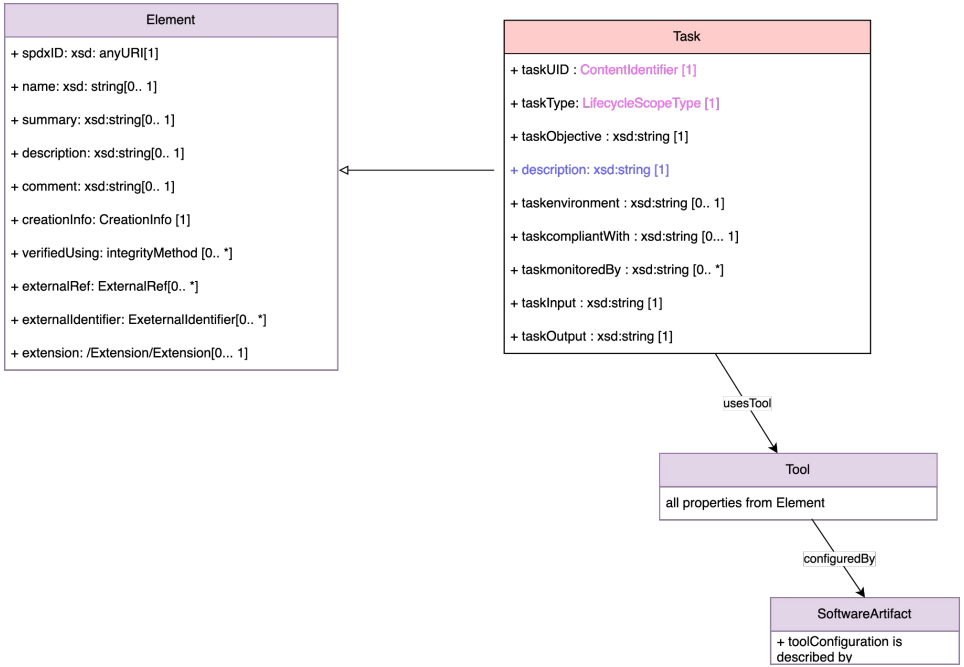
Classes for work products - TASK

Pushed to SPDX 3.2



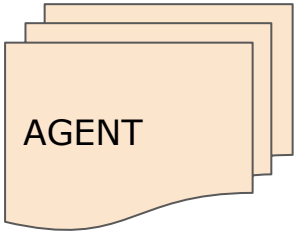
TASK class



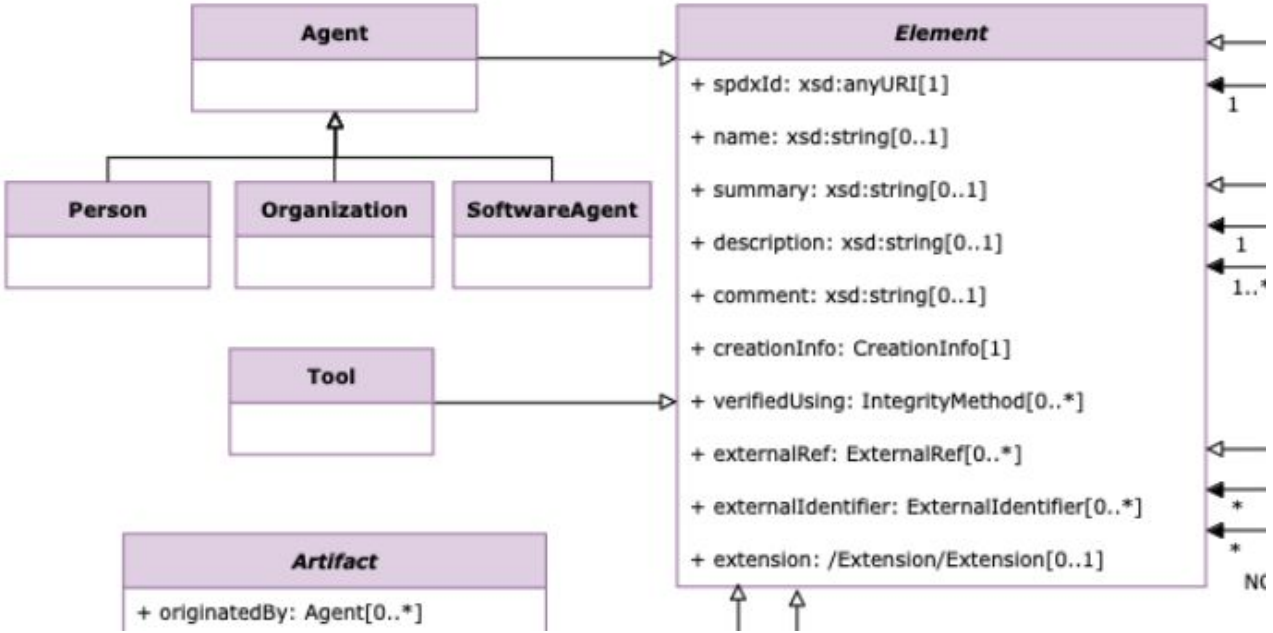


Classes for work products - AGENT

Pushed to SPDX 3.2

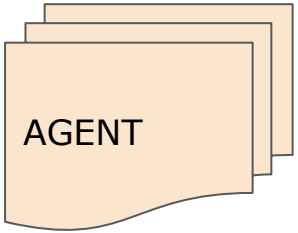


AGENT class



Classes for work products - AGENT

Pushed to SPDX 3.2



AGENT class

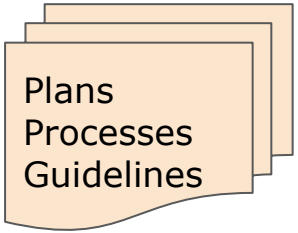
- AGENT can be an actual person, a tool, script, infrastructure, organization...
- For FuSa (and other dependability topics) the agents performing tasks must have sufficient expertise/qualification

Agent
+ AgentType: agentType [1.. *] + AgentQualification: QualificationType [1...*] + AgentQualificationVerified: QualificationVerificationType [0...1]

QualificationType
+ qualifiedFor: RoleType [1] + qualificationLevel: QualificationLevelType [1] + qualificationCompleteness: qualificationCompletenessType [1] + qualificationEvidence: ExternalRef [1]

Agent properties - enums

Pushed to SPDX 3.2



Enums to describe roles and qualifications

QualificationLevelType (enum)
Beginner
Advanced
Expert
Senior Expert
Genius
QualifiedSoftware
...
other

QualificationCompletenessType (enum)
Complete
Incomplete
other

QualificationVerificationType (enum)
Verified
Unverified
other

Role (enum)
RequirementsEngineer
VerificationEngineer
SoftwareDeveloper
SoftwareGenerator
SoftwareDeveloper
Tester
SafetyAssessor
TaskCoordinator
ProcessPlanner
SafetyAnalysisModerator
SafetyEngineer
SafetyManager
RiskAnalyst
DeploymentManager
DeploymentTester
...

Conclusions (so far)

SPDX SAFETY

- Standardized, automated format to exchange safety case documentation
- Tailored SBOMs for design phase, dev phase (source SBOM), runtime and deployed phase
- Reproducible impact analysis
- Tool agnostic information exchange
- Compliance as code approach

SPDX

… to be continued

Talk to us:
nicole@alektometis.com
kstewart@linuxfoundation.org
Mailing List
Weekly meeting Friday 18:00 CET/CEST

Appendix

Task vs HW Action

SPDX internals

Definition:

Task: a task is a specific unit of work that contributes to the completion of a project. E.g. a plan is created, code is changed, a test is specified, a test log is recorded

Action: something that is physically done to a physical object. E.g. a PCB is manufactured, an ECU is mounted into a machine,

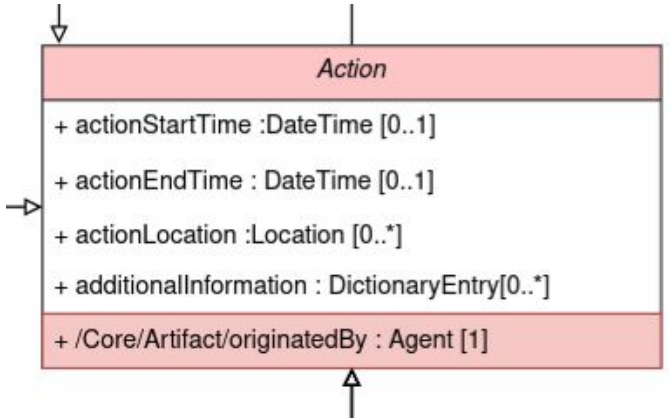
Task vs HW Action

SPDX SAFETY

TASK

Task
+ taskUID : ContentIdentifier [1]
+ taskType: LifecycleScopeType [1]
+ taskObjective : xsd:string [1]
+ description: xsd:string [1]
+ taskenvironment : xsd:string [0.. 1]
+ taskcompliantWith : xsd:string [0... 1]
+ taskmonitoredBy : xsd:string [0.. *]
+ taskInput : xsd:string [1]
+ taskOutput : xsd:string [1]

ACTION



SPDX

Task vs HW Action

Example:

For a vehicle ECU, the hardware (PCB, housing, assembling of all of it) is manufactured, which is recorded by an ACTION.

The software that has been created to run on this ECU, has been created following work steps that are defined in TASKs. These TASKs define e.g. how a software requirement must be created, how it must be written, which tooling is needed. The code for the software is created using TASKs that define how code is written (coding guidelines) and how it is managed in the config management system (e.g. GitHub workflow)

The software is then tested, using test procedures also defined by TASKs. The way the test results are captured and evaluated is also described in its TASKs.

Building the software and running the tests, as well as the creation of the Evidence artefacts for it, is all described by TASKs, there is no specific need for a Location, Duration or similar. The buildTime is already defined by ARTEFACT.

Once we talk about something physically being manufactured, location and duration will become interesting also for safety.

Corner Cases: Is flashing a binary to an ECU a TASK or an ACTION? What about downloading a binary to model to run on an FPGA?