Bringing Functional Safety to the SBOM

SPDX Safety Profile Release Candidate 3.1-rc1

FOSDEM 2026,

SBOM and Supply Chains Devroom

01.02.2026

Nicole Pappler, AlektoMetis

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

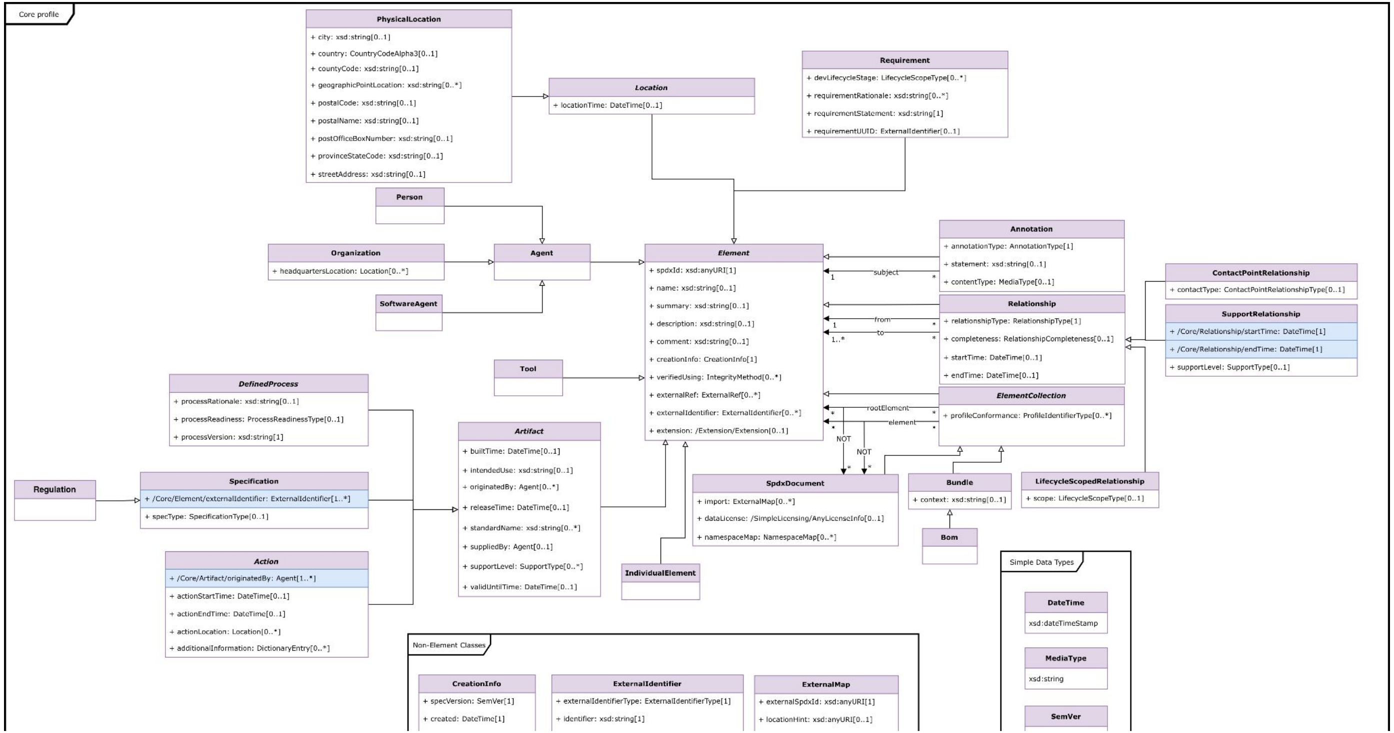
About SPDX

System Data Package Exchange

- 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 3.1-rc core model

[Core]



Core profile

PhysicalLocation
+ city: xsd:string[0..1]
+ country: CountryCodeAlpha3[0..1]
+ countyCode: xsd:string[0..1]
+ geographicPointLocation: xsd:string[0..*]
+ postalCode: xsd:string[0..1]
+ postalName: xsd:string[0..1]
+ postOfficeBoxNumber: xsd:string[0..1]
+ provinceStateCode: xsd:string[0..1]
+ streetAddress: xsd:string[0..1]

Location
+ locationTime: DateTime[0..1]

Requirement
+ devLifecycleStage: LifecycleScopeType[0..*]
+ requirementRationale: xsd:string[0..*]
+ requirementStatement: xsd:string[1]
+ requirementUUID: ExternalIdentifier[0..1]

Person

Organization
+ headquartersLocation: Location[0..*]

Agent

SoftwareAgent

Tool

Element
+ spdxId: xsd:anyURI[1]
+ name: xsd:string[0..1]
+ summary: xsd:string[0..1]
+ description: xsd:string[0..1]
+ comment: xsd:string[0..1]
+ creationInfo: CreationInfo[1]
+ verifiedUsing: IntegrityMethod[0..*]
+ externalRef: ExternalRef[0..*]
+ externalIdentifier: ExternalIdentifier[0..*]
+ extension: /Extension/Extension[0..1]

Annotation
+ annotationType: AnnotationType[1]
+ statement: xsd:string[0..1]
+ contentType: MediaType[0..1]

subject 1 *

Relationship
+ relationshipType: RelationshipType[1]
+ completeness: RelationshipCompleteness[0..1]
+ startTime: DateTime[0..1]
+ endTime: DateTime[0..1]

from 1 *
to 1..* *

ContactPointRelationship
+ contactType: ContactPointRelationshipType[0..1]

SupportRelationship
+ /Core/Relationship/startTime: DateTime[1]
+ /Core/Relationship/endTime: DateTime[1]
+ supportLevel: SupportType[0..1]

ElementCollection
+ profileConformance: ProfileIdentifierType[0..*]

rootElement * *
element * *
NOT *
NOT *

DefinedProcess
+ processRationale: xsd:string[0..1]
+ processReadiness: ProcessReadinessType[0..1]
+ processVersion: xsd:string[1]

Regulation

Specification
+ /Core/Element/externalIdentifier: ExternalIdentifier[1..*]
+ specType: SpecificationType[0..1]

Action
+ /Core/Artifact/originatedBy: Agent[1..*]
+ actionStartTime: DateTime[0..1]
+ actionEndTime: DateTime[0..1]
+ actionLocation: Location[0..*]
+ additionalInformation: DictionaryEntry[0..*]

Artifact
+ builtTime: DateTime[0..1]
+ intendedUse: xsd:string[0..1]
+ originatedBy: Agent[0..*]
+ releaseTime: DateTime[0..1]
+ standardName: xsd:string[0..*]
+ suppliedBy: Agent[0..1]
+ supportLevel: SupportType[0..*]
+ validUntilTime: DateTime[0..1]

IndividualElement

SpdxDocument
+ import: ExternalMap[0..*]
+ dataLicense: /SimpleLicensing/AnyLicenseInfo[0..1]
+ namespaceMap: NamespaceMap[0..*]

Bundle
+ context: xsd:string[0..1]

Bom

LifecycleScopedRelationship
+ scope: LifecycleScopeType[0..1]

Non-Element Classes

CreationInfo
+ specVersion: SemVer[1]
+ created: DateTime[1]

ExternalIdentifier
+ externalIdentifierType: ExternalIdentifierType[1]
+ identifier: xsd:string[1]

ExternalMap
+ externalSpdxId: xsd:anyURI[1]
+ locationHint: xsd:anyURI[0..1]

Simple Data Types

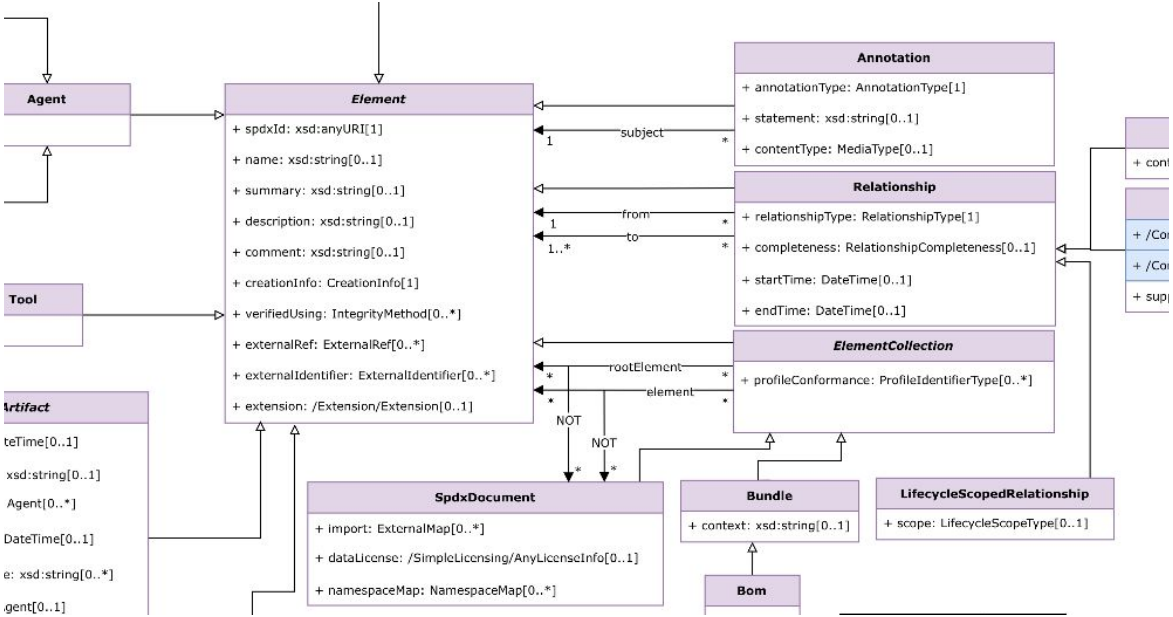
DateTime
xsd:dateTimeStamp

MediaType
xsd:string

SemVer

SPDX 3.1-rc1 Element class

Core


Agent
Element
+ spdxId: xsd:anyURI[1]
+ name: xsd:string[0..1]
+ summary: xsd:string[0..1]
+ description: xsd:string[0..1]
+ comment: xsd:string[0..1]
+ creationInfo: CreationInfo[1]
+ verifiedUsing: IntegrityMethod[0..*]
+ externalRef: ExternalRef[0..*]
+ externalIdentifier: ExternalIdentifier[0..*]
+ extension: /Extension/Extension[0..1]
Annotation
+ annotationType: AnnotationType[1]
+ statement: xsd:string[0..1]
+ contentType: MediaType[0..1]
subject
1
*
Relationship
+ relationshipType: RelationshipType[1]
+ completeness: RelationshipCompleteness[0..1]
+ startTime: DateTime[0..1]
+ endTime: DateTime[0..1]
from
to
1
1..*
*
*
ElementCollection
+ profileConformance: ProfileIdentifierType[0..*]
rootElement
element
*
*
*
*
NOT
NOT
Tool
rtifact
teTime[0..1]
xsd:string[0..1]
Agent[0..*]
DateTime[0..1]
e: xsd:string[0..*]
gent[0..1]
SpdxDocument
+ import: ExternalMap[0..*]
+ dataLicense: /SimpleLicensing/AnyLicenseInfo[0..1]
+ namespaceMap: NamespaceMap[0..*]
Bundle
+ context: xsd:string[0..1]
Bom
LifecycleScopedRelationship
+ scope: LifecycleScopeType[0..1]
+ cont
+ /Co
+ /Co
+ sup

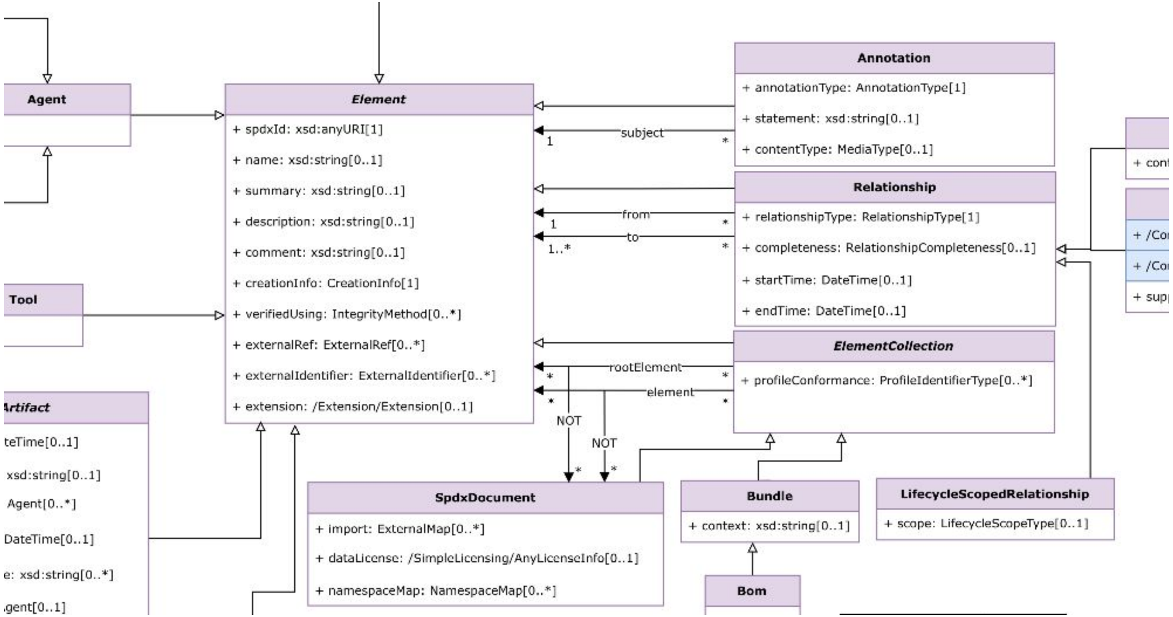
Element:

Basic class, includes e.g. information on

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

SPDX 3.1-rc1 Relationship class

Core



Agent

Tool

Artifact
teTime[0..1]
xsd:string[0..1]
Agent[0..*]
DateTime[0..1]
e: xsd:string[0..*]
gent[0..1]

Element
+ spdxId: xsd:anyURI[1]
+ name: xsd:string[0..1]
+ summary: xsd:string[0..1]
+ description: xsd:string[0..1]
+ comment: xsd:string[0..1]
+ creationInfo: CreationInfo[1]
+ verifiedUsing: IntegrityMethod[0..*]
+ externalRef: ExternalRef[0..*]
+ externalIdentifier: ExternalIdentifier[0..*]
+ extension: /Extension/Extension[0..1]

Annotation
+ annotationType: AnnotationType[1]
+ statement: xsd:string[0..1]
+ contentType: MediaType[0..1]

Relationship
+ relationshipType: RelationshipType[1]
+ completeness: RelationshipCompleteness[0..1]
+ startTime: DateTime[0..1]
+ endTime: DateTime[0..1]

ElementCollection
+ profileConformance: ProfileIdentifierType[0..*]

SpdxDocument
+ import: ExternalMap[0..*]
+ dataLicense: /SimpleLicensing/AnyLicenseInfo[0..1]
+ namespaceMap: NamespaceMap[0..*]

Bundle
+ context: xsd:string[0..1]

LifecycleScopedRelationship
+ scope: LifecycleScopeType[0..1]

Bom

+ cont
+ /Co
+ /Co
+ sup

Relationship:

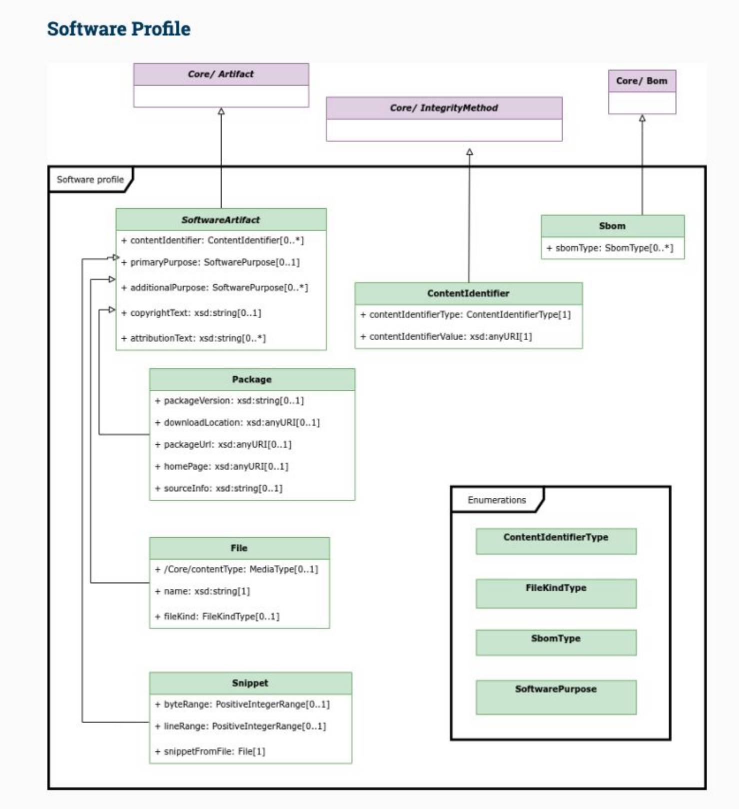
Class describing dependencies, like

- hasEvidence
- hasSpecification
- verifiedBy
- traceToDetail

A relationship can be complete or incomplete

SPDX 3.0 Profiles

Examples



Software Profile

Core/ Artifact

Core/ IntegrityMethod

Core/ Bom

Software profile

SoftwareArtifact
+ contentIdentifier: ContentIdentifier[0..*]
+ primaryPurpose: SoftwarePurpose[0..1]
+ additionalPurpose: SoftwarePurpose[0..*]
+ copyrightText: xsd:string[0..1]
+ attributionText: xsd:string[0..*]

ContentIdentifier
+ contentIdentifierType: ContentIdentifierType[1]
+ contentIdentifierValue: xsd:anyURI[1]

Sbom
+ sbomType: SbomType[0..*]

Package
+ packageVersion: xsd:string[0..1]
+ downloadLocation: xsd:anyURI[0..1]
+ packageUrl: xsd:anyURI[0..1]
+ homePage: xsd:anyURI[0..1]
+ sourceInfo: xsd:string[0..1]

File
+ /Core/contentType: MediaType[0..1]
+ name: xsd:string[1]
+ fileKind: FileKindType[0..1]

Snippet
+ byteRange: PositiveIntegerRange[0..1]
+ lineRange: PositiveIntegerRange[0..1]
+ snippetFromFile: File[1]

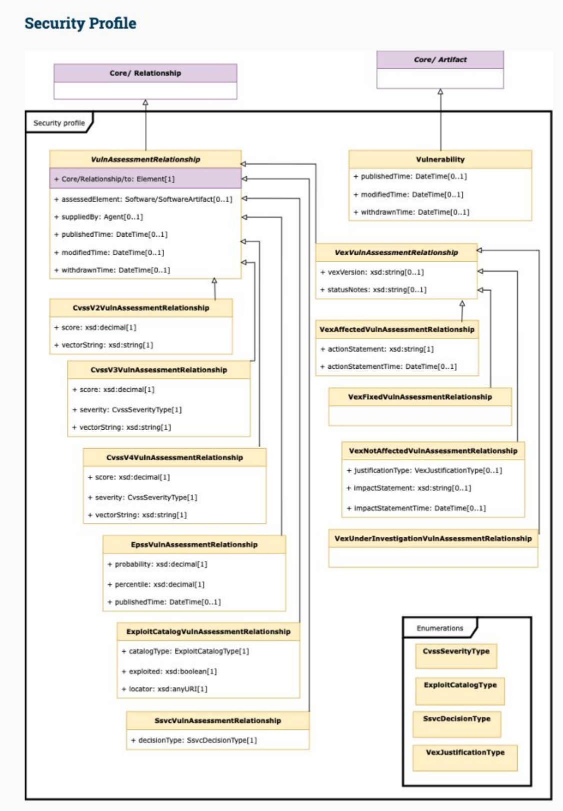
Enumerations

ContentIdentifierType

FileKindType

SbomType

SoftwarePurpose



Security Profile

Core/ Relationship

Core/ Artifact

Security profile

VulnAssessmentRelationship
+ Core/Relationship/to: Element[1]
+ assessedElement: Software/SoftwareArtifact[0..1]
+ suppliedBy: Agent[0..1]
+ publishedTime: DateTime[0..1]
+ modifiedTime: DateTime[0..1]
+ withdrawnTime: DateTime[0..1]

Vulnerability
+ publishedTime: DateTime[0..1]
+ modifiedTime: DateTime[0..1]
+ withdrawnTime: DateTime[0..1]

CvssV2VulnAssessmentRelationship
+ score: xsd:decimal[1]
+ vectorString: xsd:string[1]

CvssV3VulnAssessmentRelationship
+ score: xsd:decimal[1]
+ severity: CvssSeverityType[1]
+ vectorString: xsd:string[1]

CvssV4VulnAssessmentRelationship
+ score: xsd:decimal[1]
+ severity: CvssSeverityType[1]
+ vectorString: xsd:string[1]

EpssVulnAssessmentRelationship
+ probability: xsd:decimal[1]
+ percentile: xsd:decimal[1]
+ publishedTime: DateTime[0..1]

ExploitCatalogVulnAssessmentRelationship
+ catalogType: ExploitCatalogType[1]
+ exploited: xsd:boolean[1]
+ locator: xsd:anyURI[1]

SsvcVulnAssessmentRelationship
+ decisionType: SsvcDecisionType[1]

VexVulnAssessmentRelationship
+ vexVersion: xsd:string[0..1]
+ statusNotes: xsd:string[0..1]

VexAffectedVulnAssessmentRelationship
+ actionStatement: xsd:string[1]
+ actionStatementTime: DateTime[0..1]

VexFixedVulnAssessmentRelationship

VexNotAffectedVulnAssessmentRelationship
+ justificationType: VexJustificationType[0..1]
+ impactStatement: xsd:string[0..1]
+ impactStatementTime: DateTime[0..1]

VexUnderInvestigationVulnAssessmentRelationship

Enumerations

CvssSeverityType

ExploitCatalogType

SsvcDecisionType

VexJustificationType

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

Use Cases

SPDX SAFETY

Use Case 1: Exchange FuSa related information in the supply chain

- Complete project information as Safety Case
- Planning of Functional Safety Management and (Safety) Concept
- Obligations for integration and operation

Use Case 2: Traceability within the project

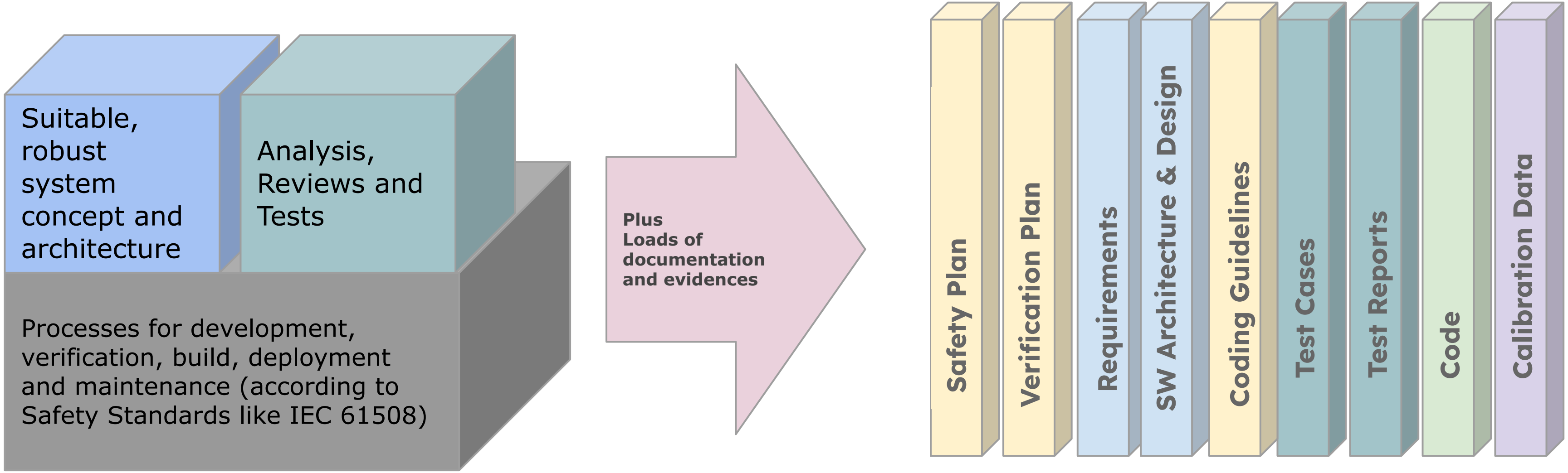
- Process application and guidelines to implemented work products
- Specifications, requirements, architecture, safety analysis, code, tests and test reports

Use Case 3: Support of automated assessments

- Standardized assessment interfaces

Use Case - Exchange Information

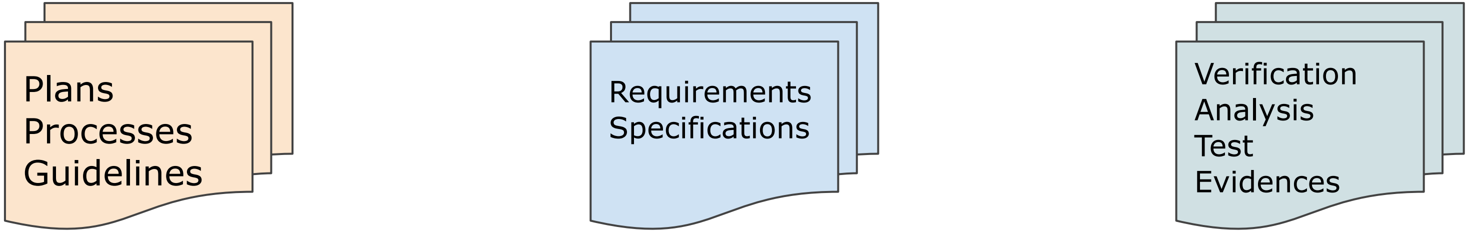
Safety Case documentation


Suitable, robust system concept and architecture
Analysis, Reviews and Tests
Processes for development, verification, build, deployment and maintenance (according to Safety Standards like IEC 61508)
Plus Loads of documentation and evidences
Safety Plan
Verification Plan
Requirements
SW Architecture & Design
Coding Guidelines
Test Cases
Test Reports
Code
Calibration Data

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!

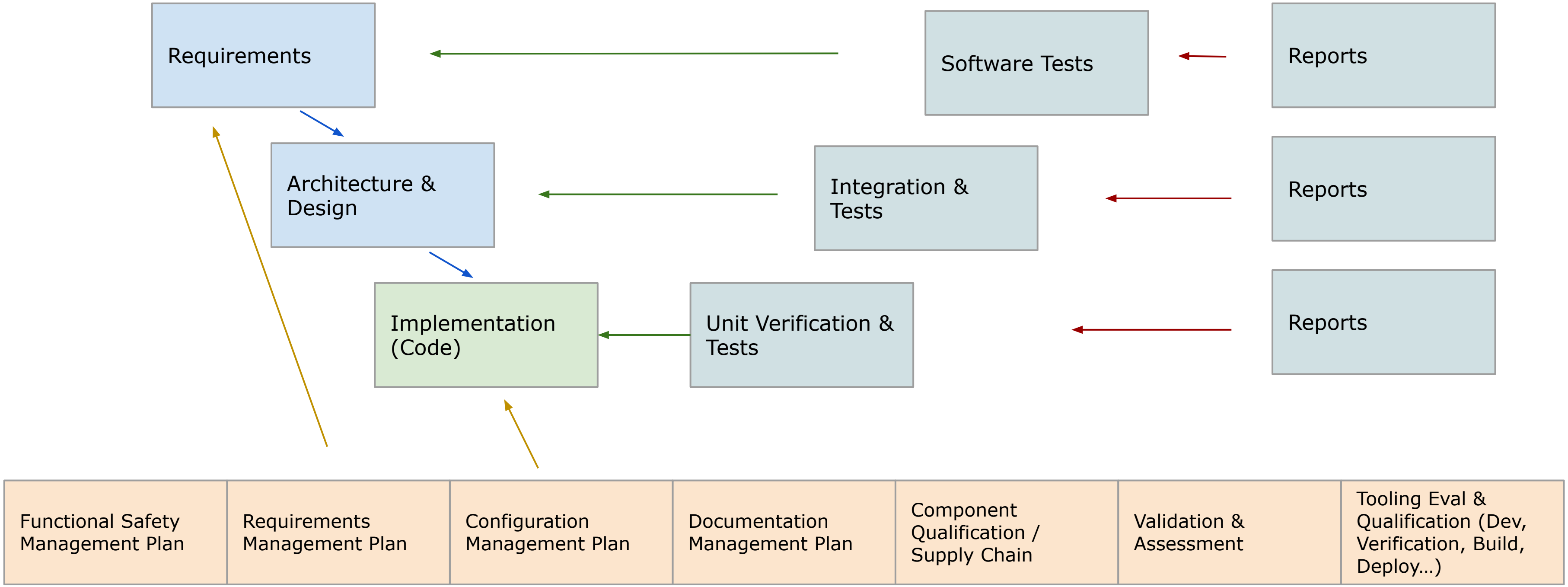


Plans
Processes
Guidelines

Requirements
Specifications

Verification
Analysis
Test
Evidences

Dependencies in a FuSa Project



Requirements

Architecture & Design

Implementation (Code)

Software Tests

Integration & Tests

Unit Verification & Tests

Reports

Reports

Reports

Functional Safety Management Plan

Requirements Management Plan

Configuration Management Plan

Documentation Management Plan

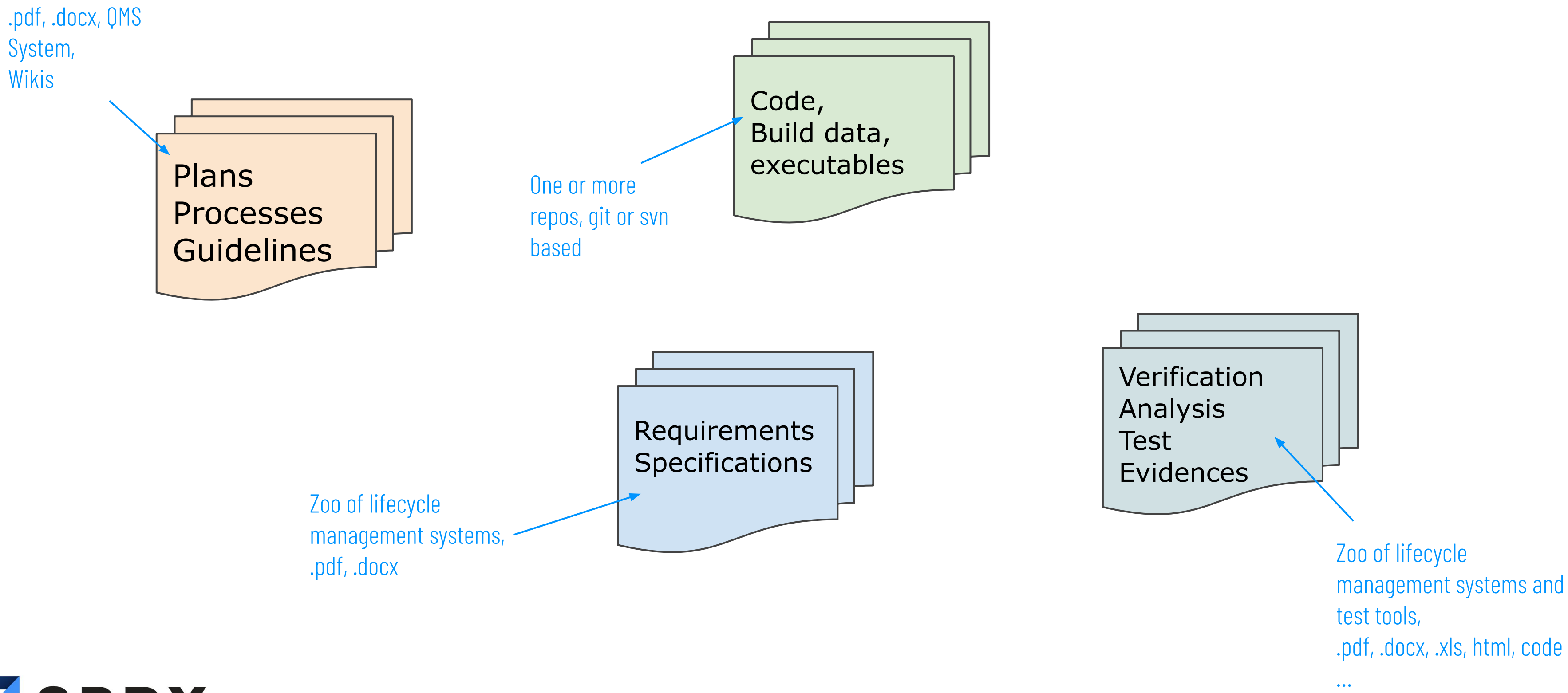
Component Qualification / Supply Chain

Validation & Assessment

Tooling Eval & Qualification (Dev, Verification, Build, Deploy...)

Data Structure of current FuSa projects...

SPDX SAFETY



.pdf, .docx, QMS System, Wikis

Plans
Processes
Guidelines

One or more repos, git or svn based

Code,
Build data,
executables

Zoo of lifecycle management systems, .pdf, .docx

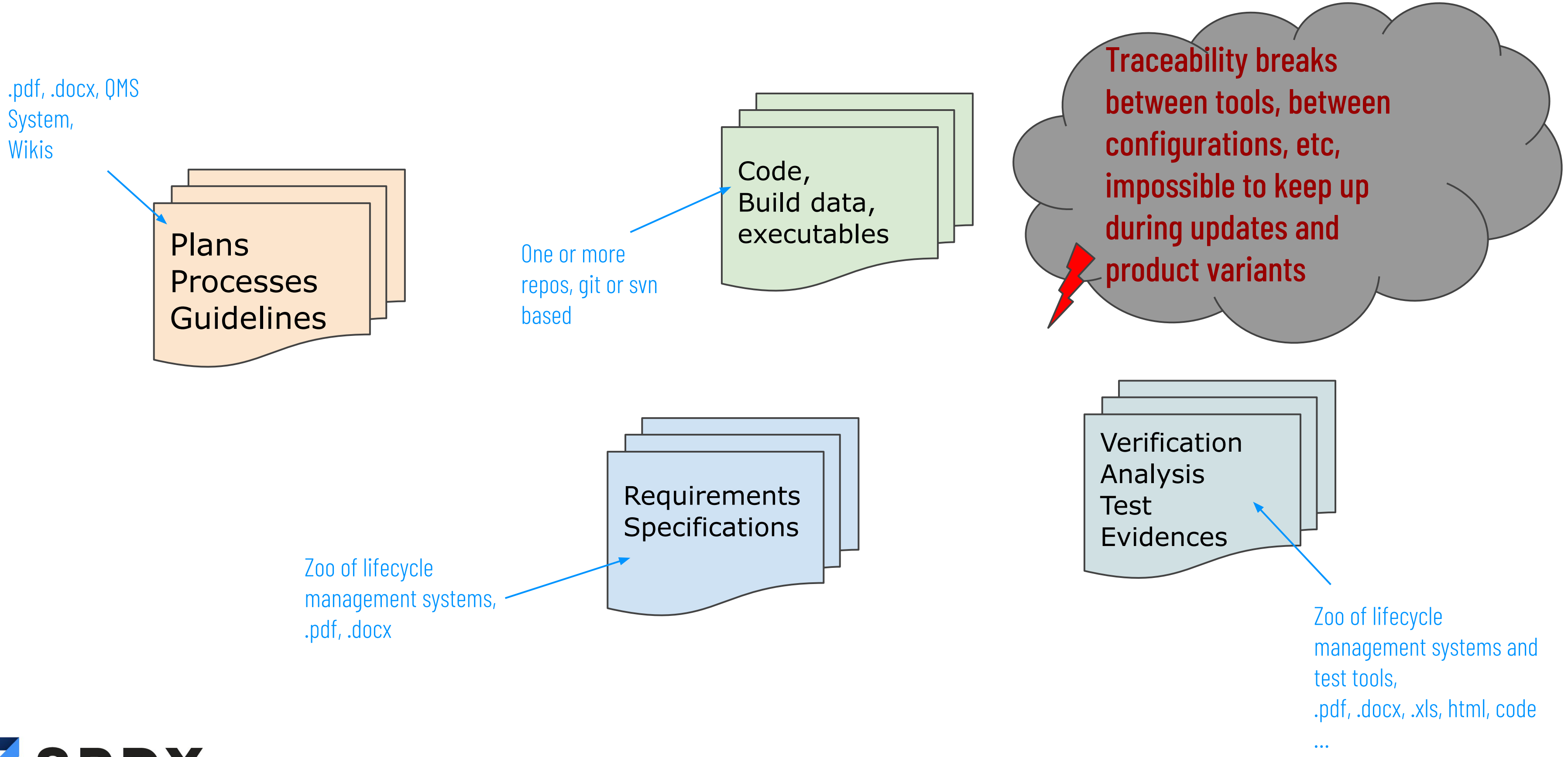
Requirements
Specifications

Verification
Analysis
Test
Evidences

Zoo of lifecycle management systems and test tools, .pdf, .docx, .xls, html, code ...

SPDX

Data Structure of current FuSa projects…



.pdf, .docx, QMS System, Wikis

Plans
Processes
Guidelines

One or more repos, git or svn based

Code,
Build data,
executables

Traceability breaks between tools, between configurations, etc, impossible to keep up during updates and product variants

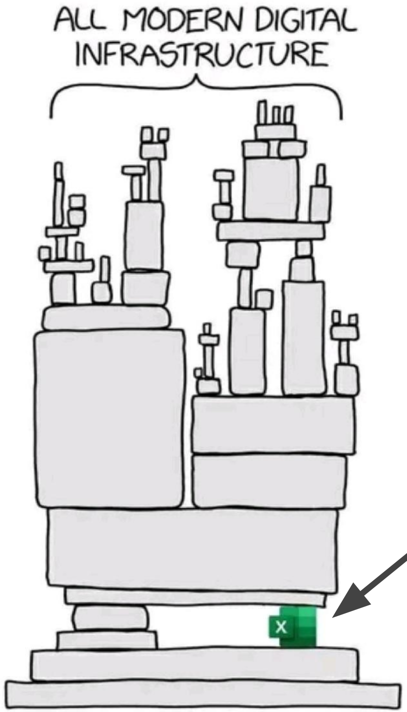
Zoo of lifecycle management systems, .pdf, .docx

Requirements
Specifications

Verification
Analysis
Test
Evidences

Zoo of lifecycle management systems and test tools,
.pdf, .docx, .xls, html, code
…

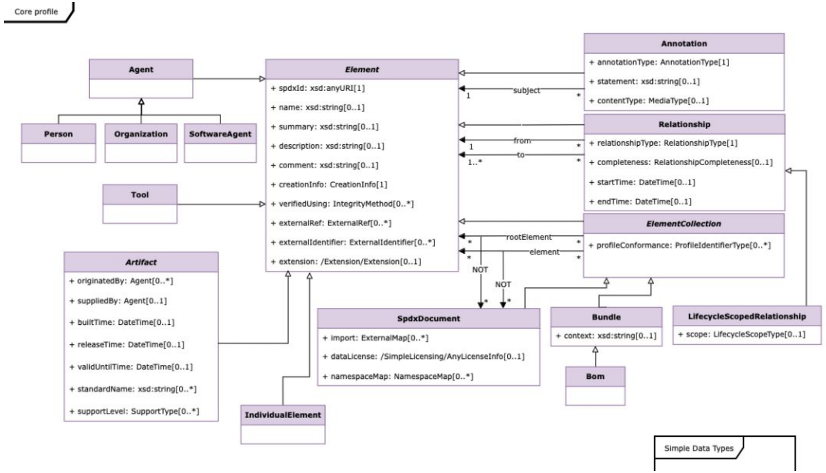
No 1 Safety Information Exchange Format



draft_2005TemplateSafetyCase_thisproject_final_forTraceingv06.xls

Emoji by emojidex

Why not use this instead?



Core profile

Agent

Person

Organization

SoftwareAgent

Tool

Element
- + spdxId: xsd:anyURI[1]
- + name: xsd:string[0..1]
- + summary: xsd:string[0..1]
- + description: xsd:string[0..1]
- + comment: xsd:string[0..1]
- + creationInfo: CreationInfo[1]
- + verifiedUsing: IntegrityMethod[0..*]
- + externalRef: ExternalRef[0..*]
- + externalIdentifier: ExternalIdentifier[0..*]
- + extension: /Extension/Extension[0..1]

Artifact
- + originatedBy: Agent[0..*]
- + suppliedBy: Agent[0..1]
- + builtTime: DateTime[0..1]
- + releaseTime: DateTime[0..1]
- + validUntilTime: DateTime[0..1]
- + standardName: xsd:string[0..*]
- + supportLevel: SupportType[0..*]

IndividualElement

SpdxDocument
- + import: ExternalMap[0..*]
- + dataLicense: /SimpleLicensing/AnyLicenseInfo[0..1]
- + namespaceMap: NamespaceMap[0..*]

Annotation
- + annotationType: AnnotationType[1]
- + statement: xsd:string[0..1]
- + contentType: MediaType[0..1]

Relationship
- + relationshipType: RelationshipType[1]
- + completeness: RelationshipCompleteness[0..1]
- + startTime: DateTime[0..1]
- + endTime: DateTime[0..1]

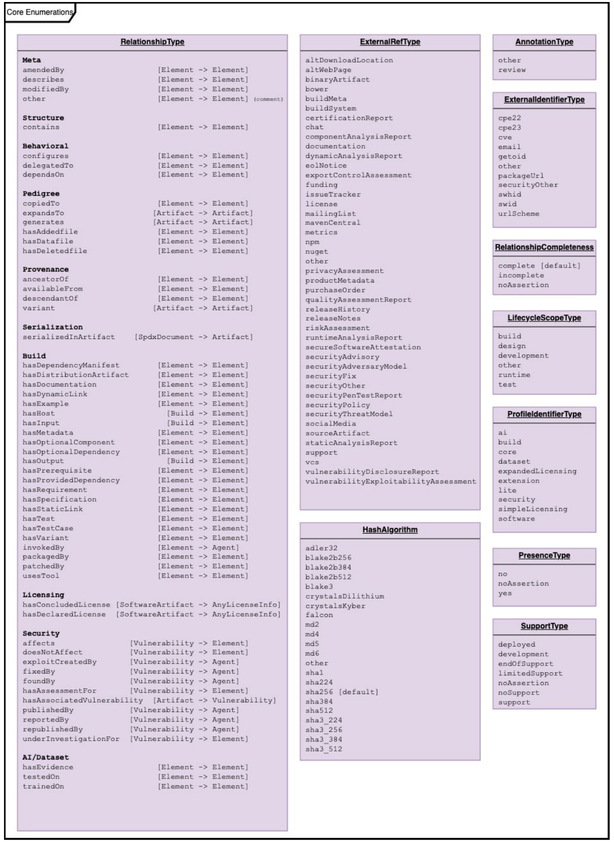
ElementCollection
- + profileConformance: ProfileIdentifierType[0..*]

Bundle
- + context: xsd:string[0..1]

Bom

LifecycleScopedRelationship
- + scope: LifecycleScopeType[0..1]

Simple Data Types



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
getoid
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

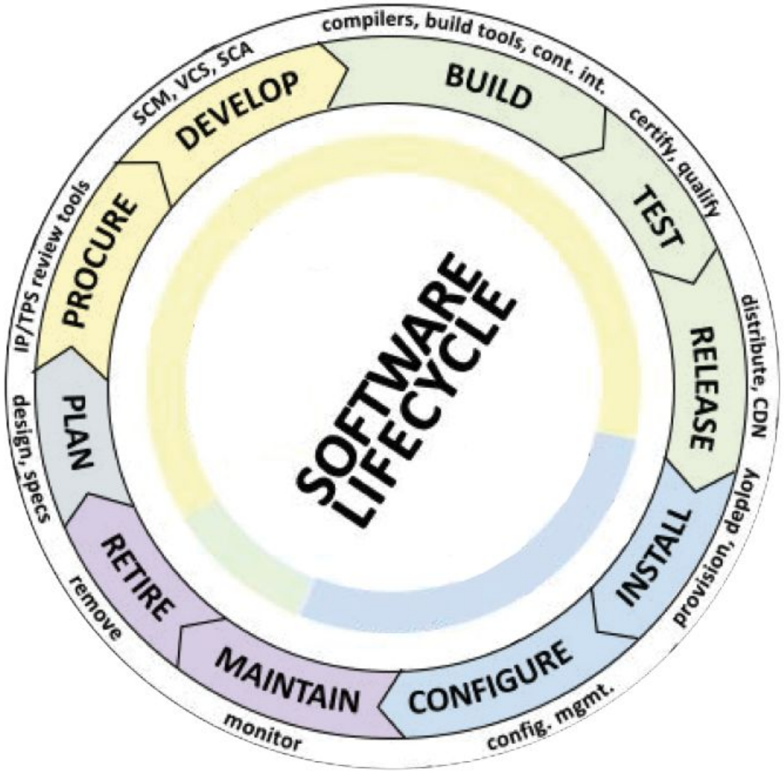
deployed
development
endOfSupport
limitedSupport
noAssertion
noSupport
support

Generate SBOMS when the data is known

Source SBOM

Design SBOM

Runtime SBOM



Build SBOM

Deployed SBOM

Different project phases - SPDX Safety SBOMs

SPDX SAFETY

Concept phase & Implementation

Design SBOM

Source SBOM

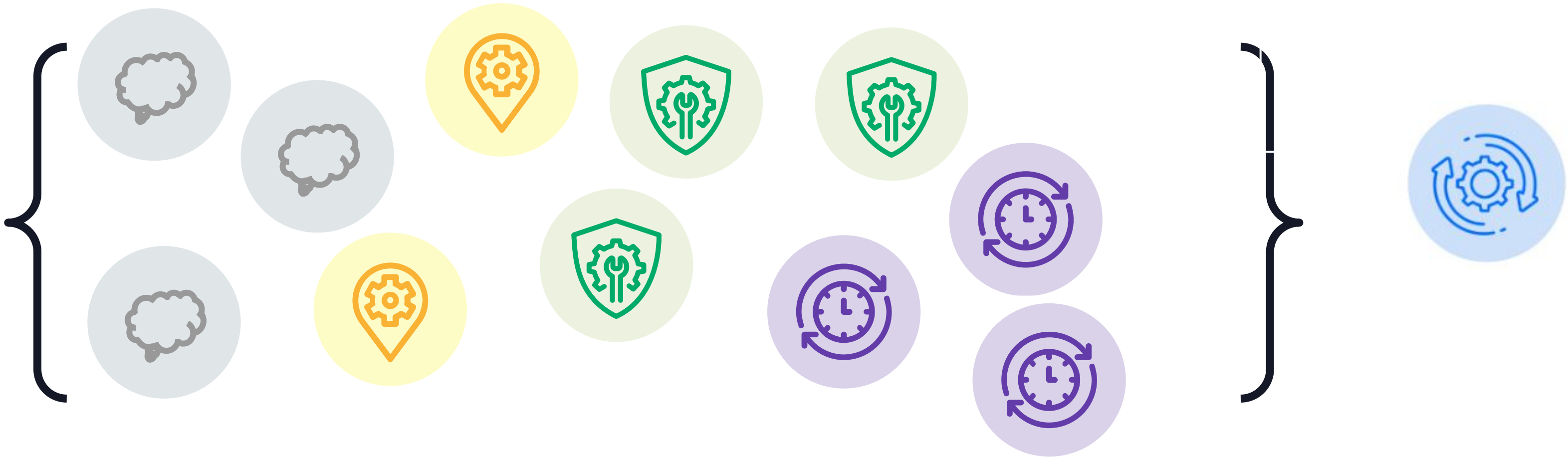
Build & Test

Build SBOM

Runtime SBOM

Final integrated system

Deployed SBOM

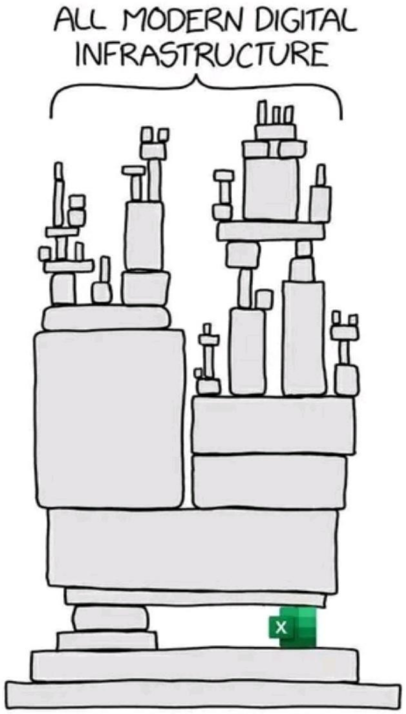


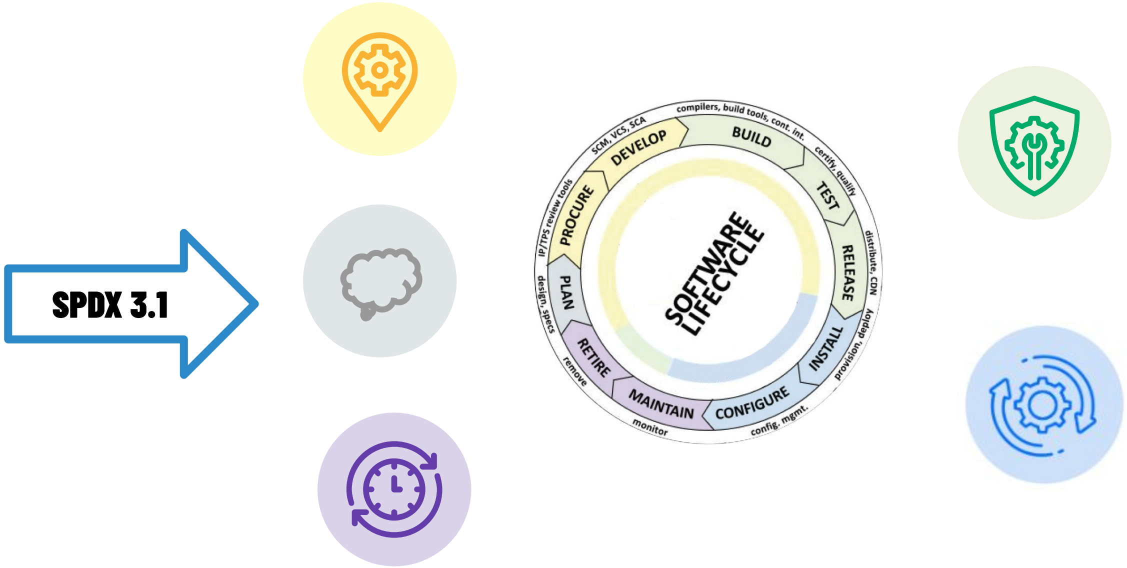
SPDX

Safety Information Exchange Format?

SPDX SAFETY

SPDX SBOM



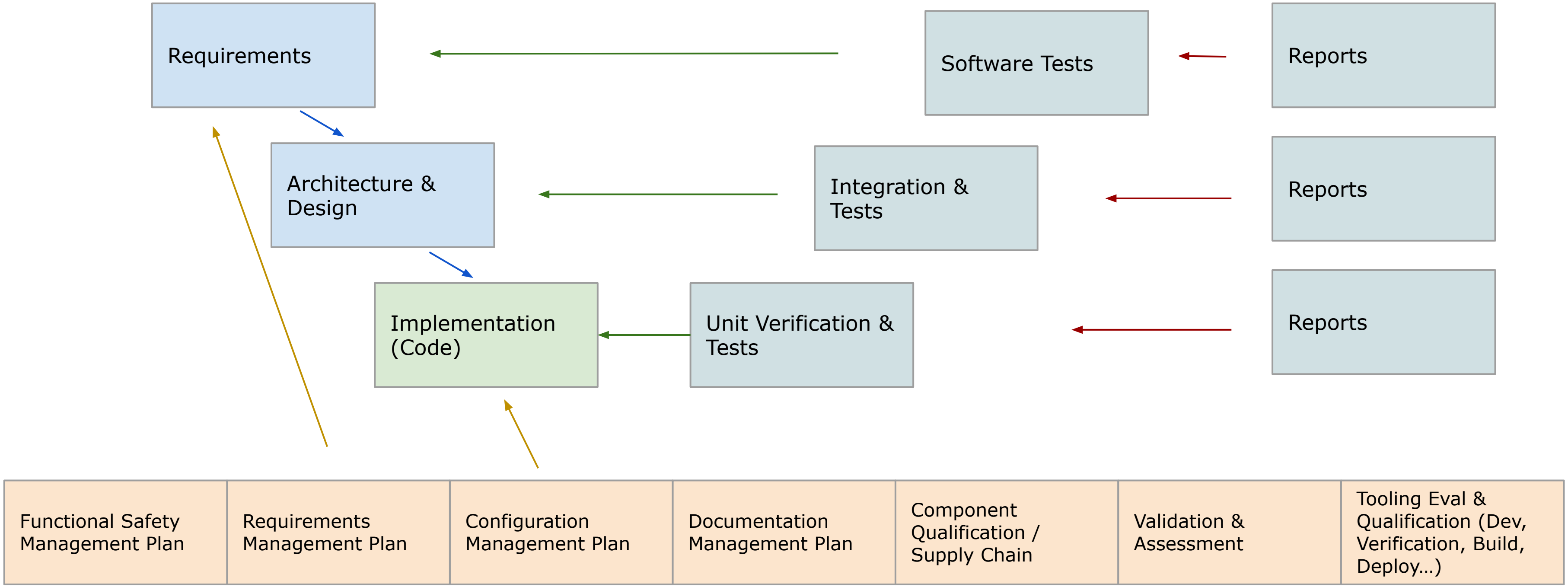


... instead of inconsistent Spreadsheets, manual import/export of half decent ReqIFs... why not use SBOMS?

SPDX

Use Case - Traceability

Dependencies in a FuSa Project



Requirements

Software Tests

Reports

Architecture & Design

Integration & Tests

Reports

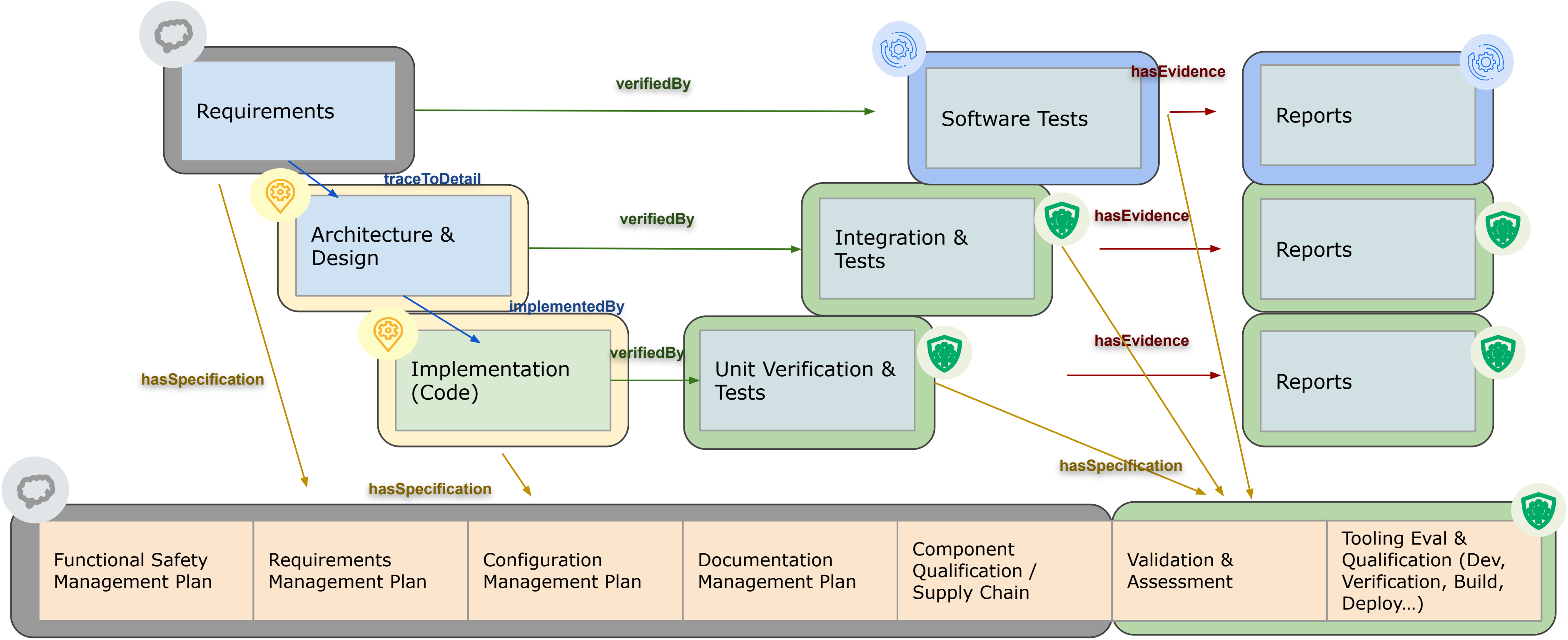
Implementation (Code)

Unit Verification & Tests

Reports

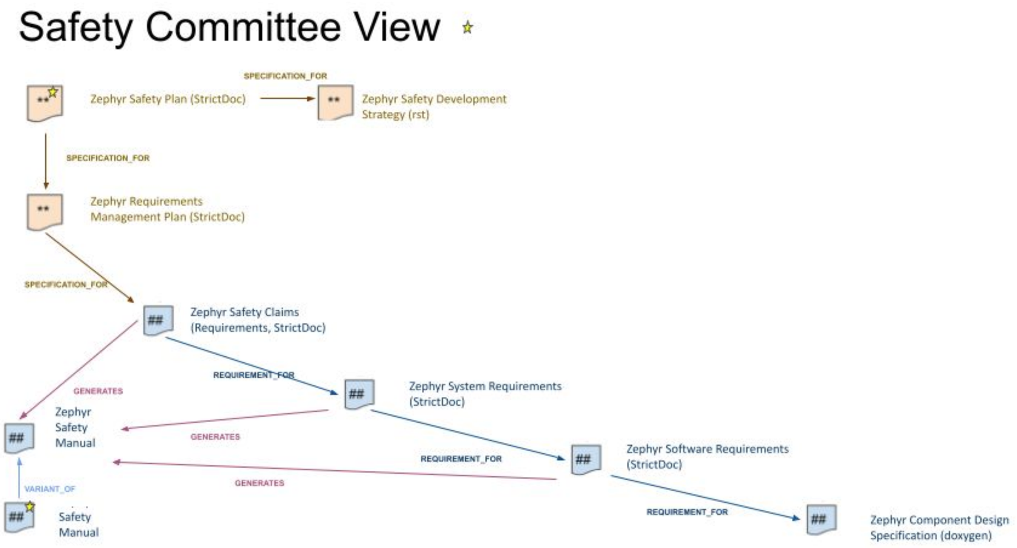
Functional Safety Management Plan	Requirements Management Plan	Configuration Management Plan	Documentation Management Plan	Component Qualification / Supply Chain	Validation & Assessment	Tooling Eval & Qualification (Dev, Verification, Build, Deploy...)

Dependencies in a FuSa Project*



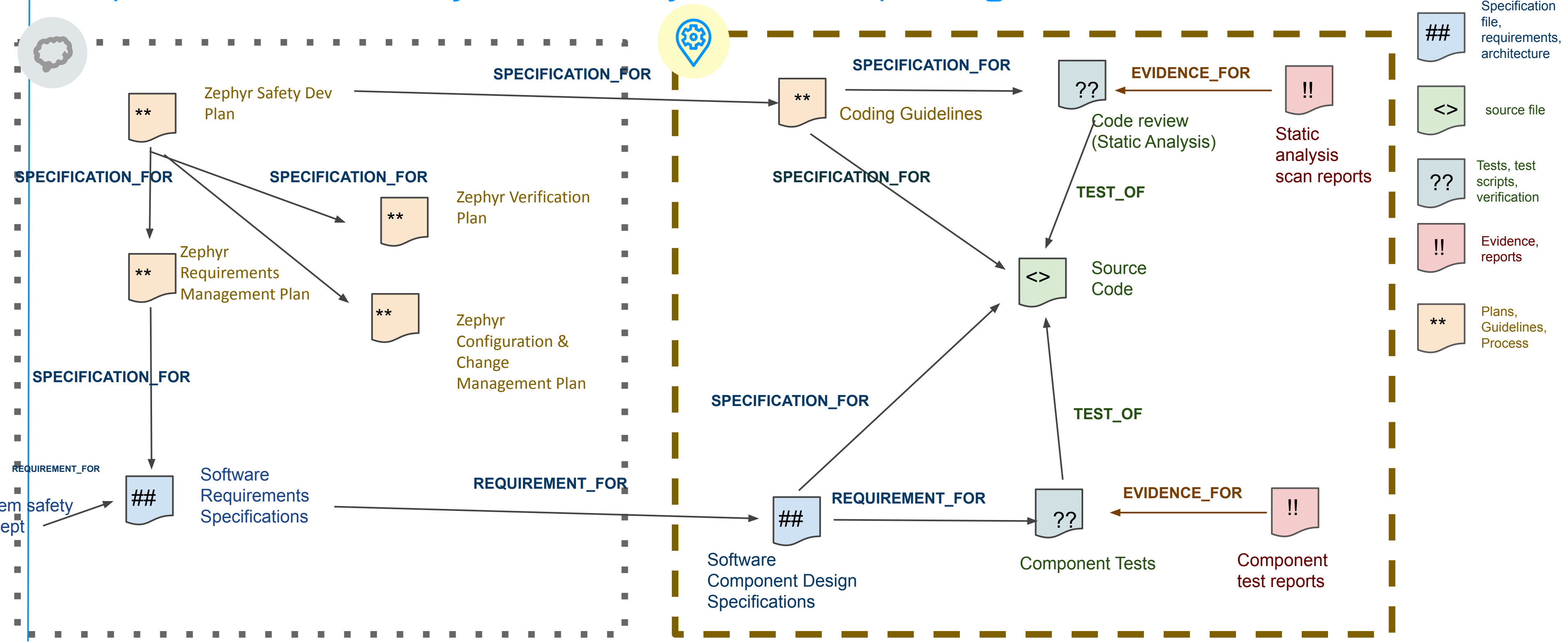
Zephyr Requirements Management

Requirements Management Knowledge Model


Safety Committee View ☆
Zephyr Safety Plan (StrictDoc)
SPECIFICATION_FOR
Zephyr Safety Development Strategy (rst)
SPECIFICATION_FOR
Zephyr Requirements Management Plan (StrictDoc)
SPECIFICATION_FOR
Zephyr Safety Claims (Requirements, StrictDoc)
REQUIREMENT_FOR
Zephyr System Requirements (StrictDoc)
GENERATES
Zephyr Safety Manual
GENERATES
REQUIREMENT_FOR
Zephyr Software Requirements (StrictDoc)
GENERATES
VARIANT_OF
Safety Manual
REQUIREMENT_FOR
Zephyr Component Design Specification (doxygen)

*prototyping with SPDX 2.3 relationships

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



Zephyr Safety Dev Plan

SPECIFICATION_FOR

Coding Guidelines

SPECIFICATION_FOR

Code review (Static Analysis)

EVIDENCE_FOR

Static analysis scan reports

SPECIFICATION_FOR

SPECIFICATION_FOR

Zephyr Verification Plan

SPECIFICATION_FOR

TEST_OF

Zephyr Requirements Management Plan

Source Code

Zephyr Configuration & Change Management Plan

SPECIFICATION_FOR

SPECIFICATION_FOR

TEST_OF

REQUIREMENT_FOR

em safety
ept

Software Requirements Specifications

REQUIREMENT_FOR

REQUIREMENT_FOR

EVIDENCE_FOR

Software Component Design Specifications

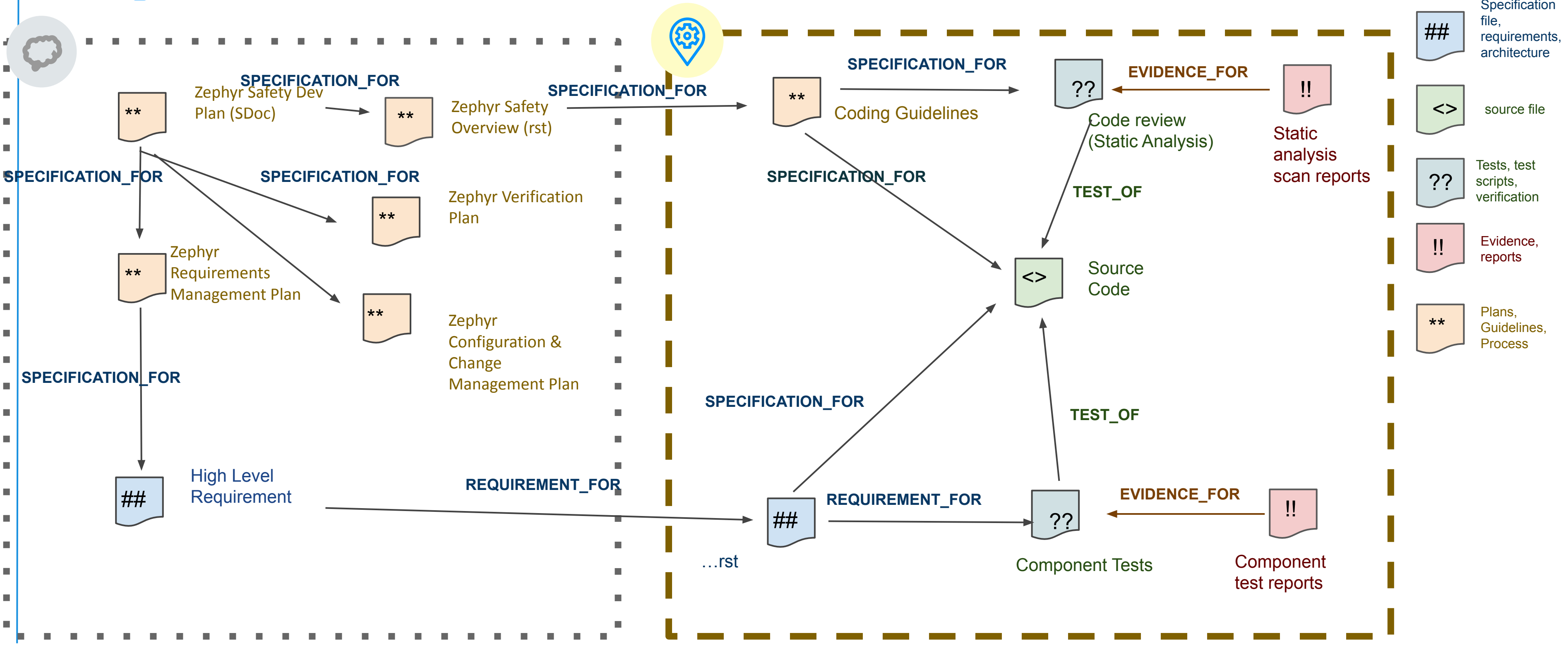
Component Tests

Component test reports

##	Specification file, requirements, architecture
<>	source file
??	Tests, test scripts, verification
!!	Evidence, reports
**	Plans, Guidelines, Process

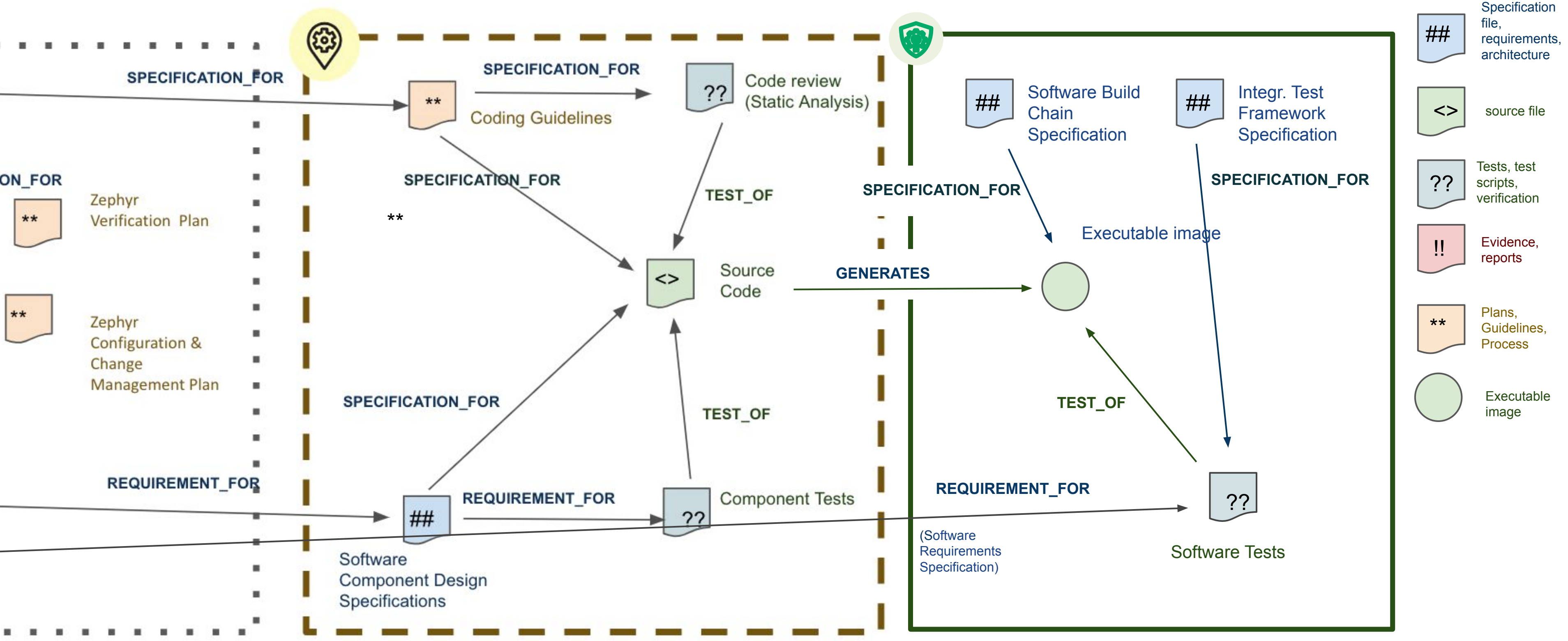
*prototyping with SPDX 2.3 relationships

Design SBOM to Source SBOM



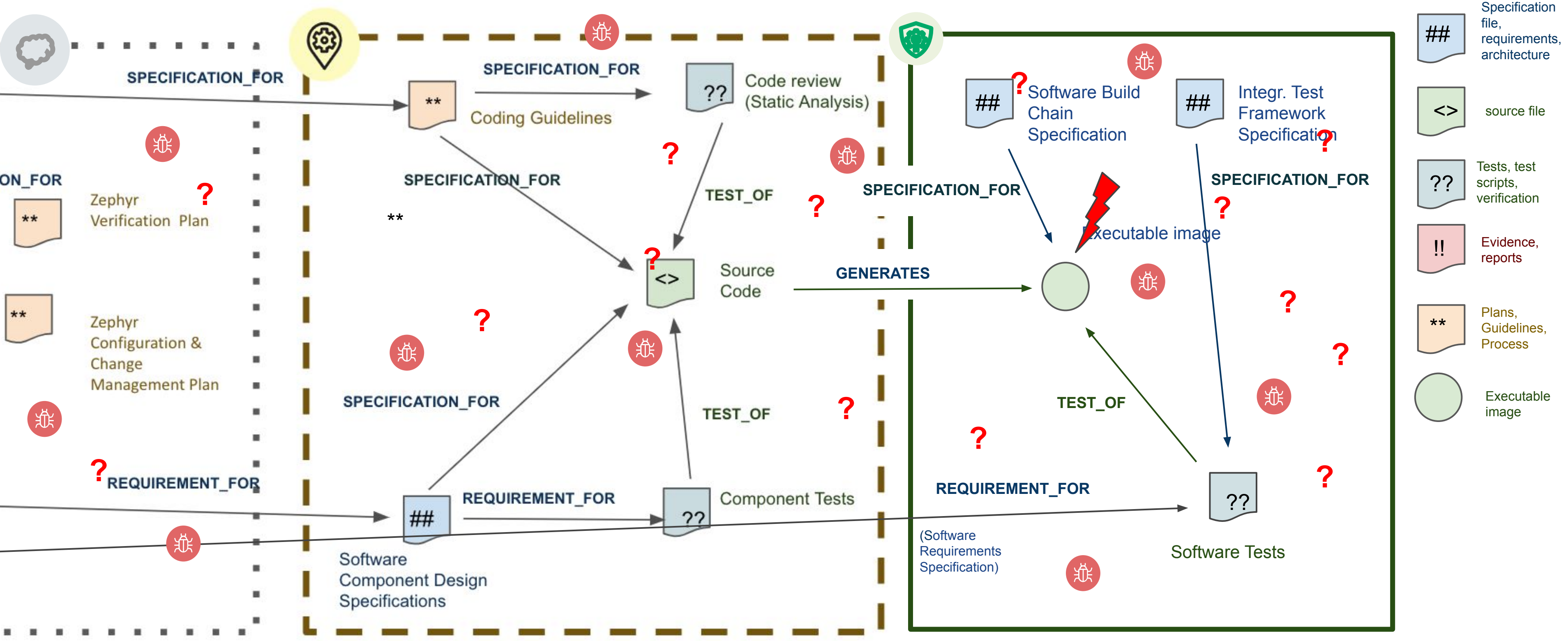
*prototyping with SPDX 2.3 relationships

Source SBOM to Build SBOM



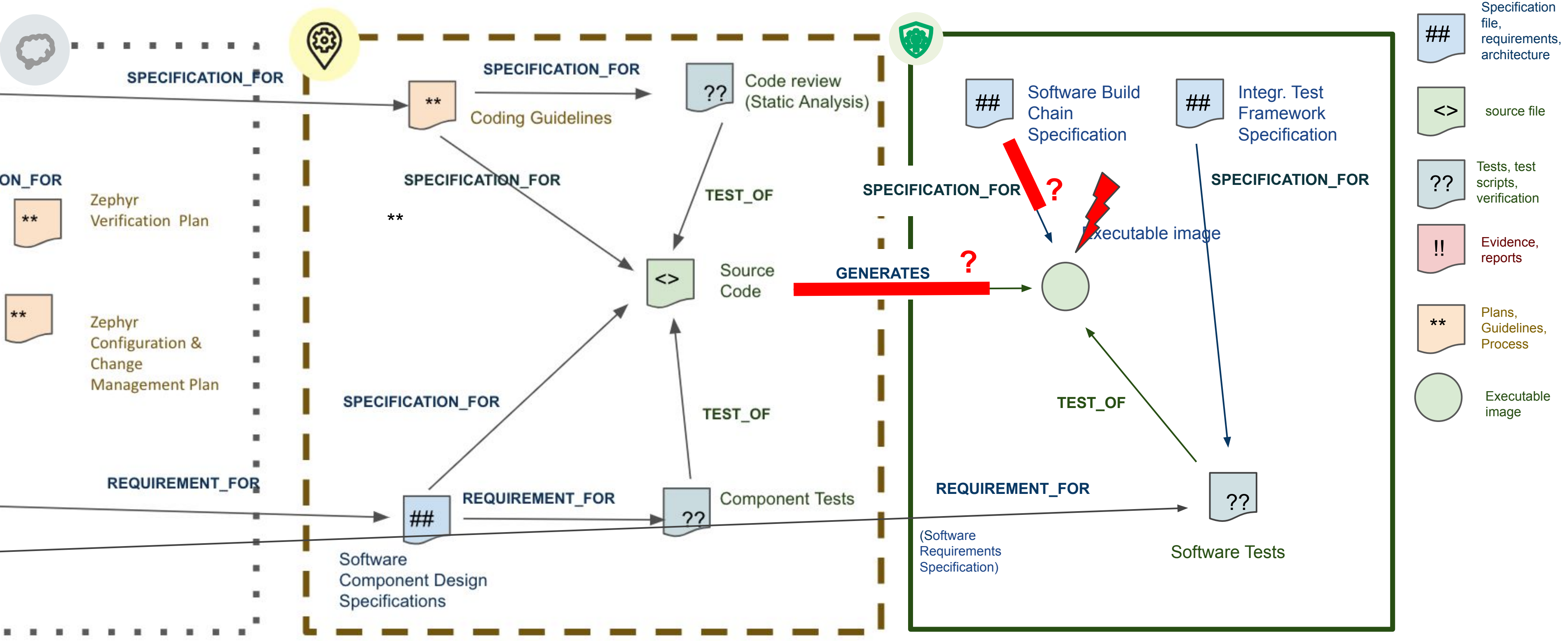
*prototyping with SPDX 2.3 relationships

Dependency Identification on Component Level

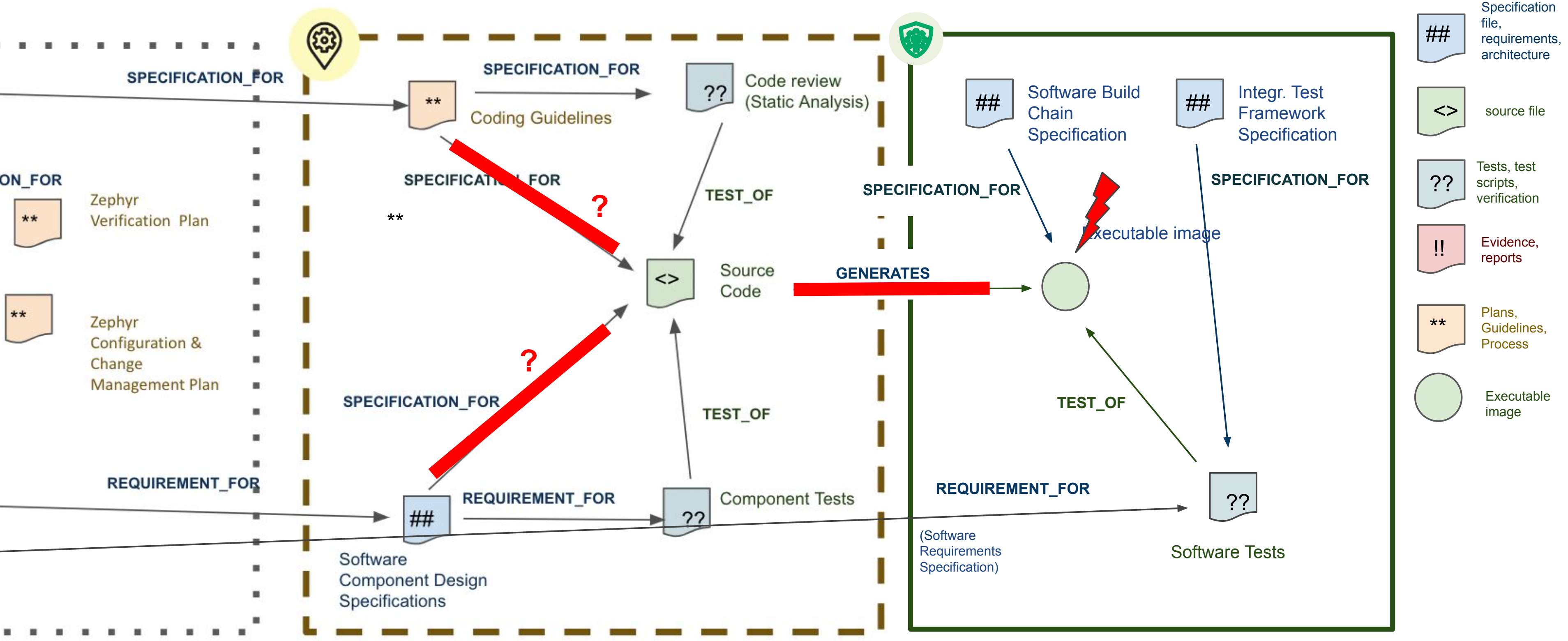


*prototyping with SPDX 2.3 relationships

Dependency Identification on Component Level

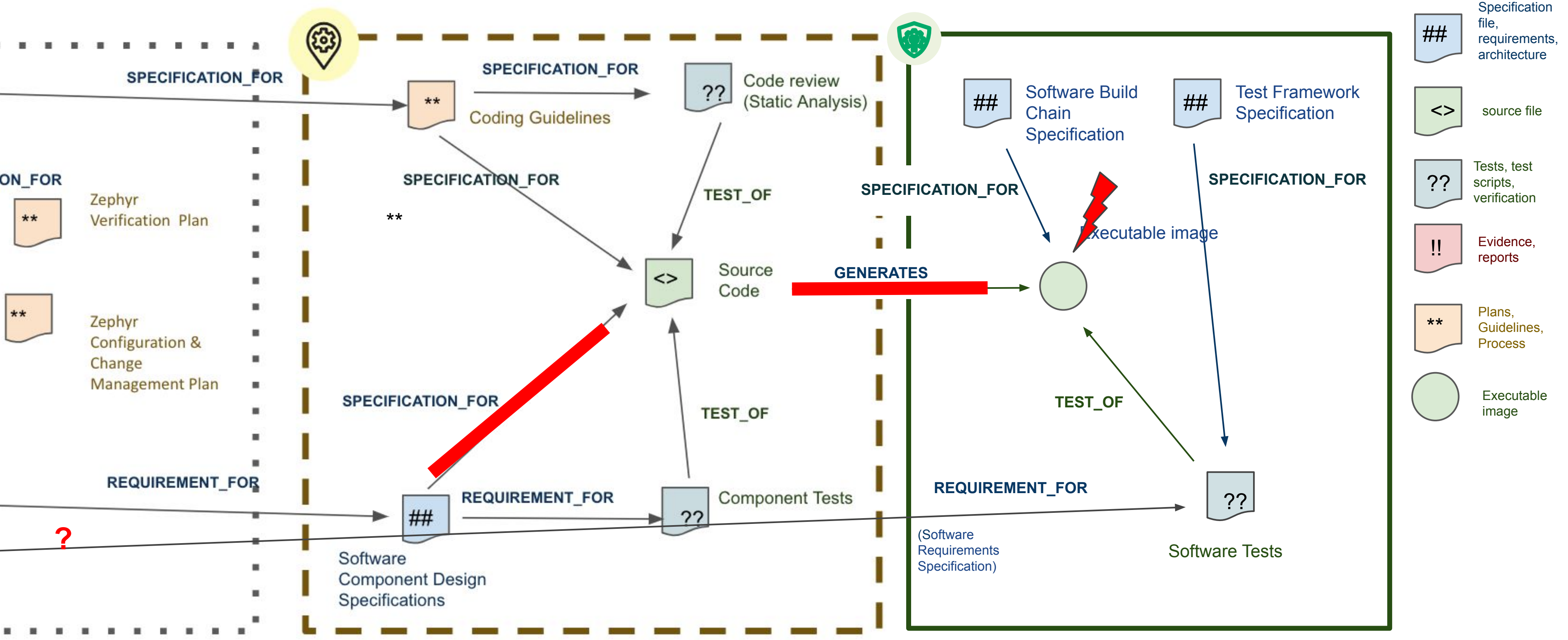


Dependency Identification on Component Level



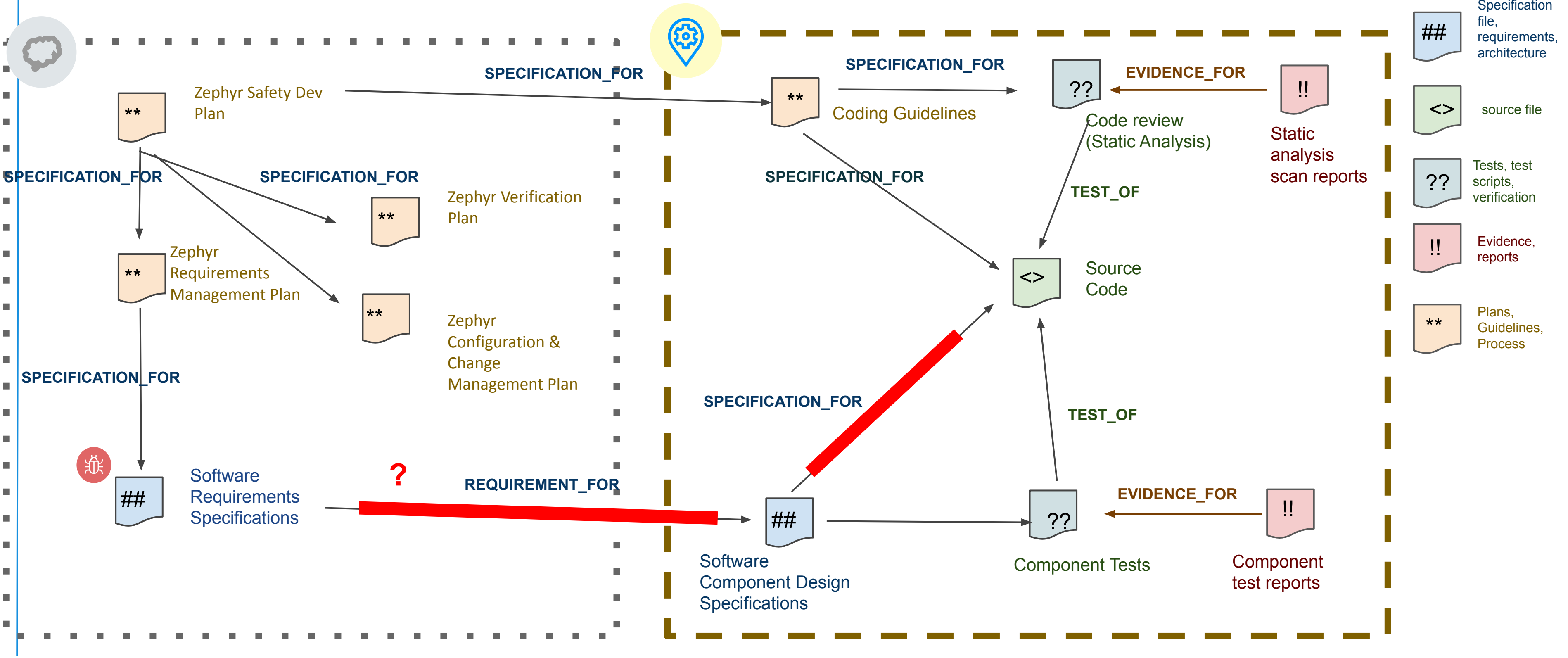
Dependency Identification on Component Level

SPDX SAFETY

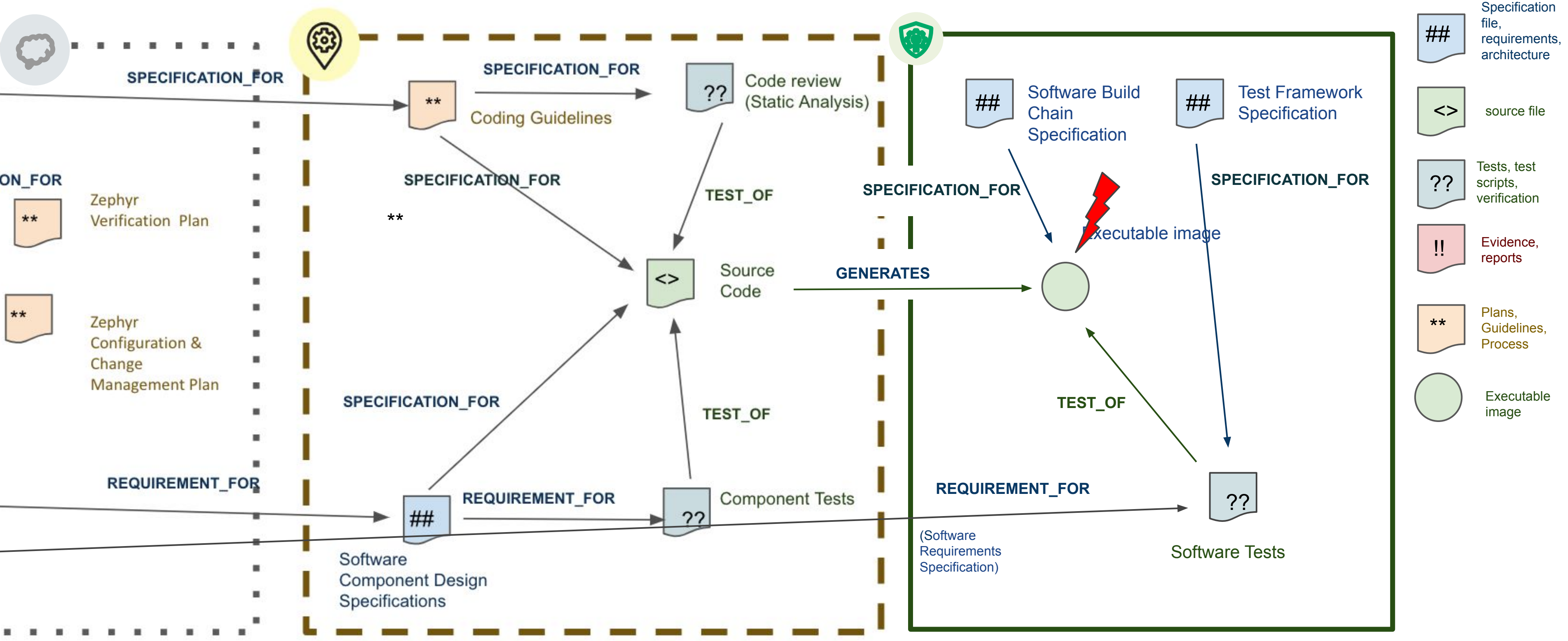


SPDX

Dependency Identification on Component Level



Dependency Identification on Component Level



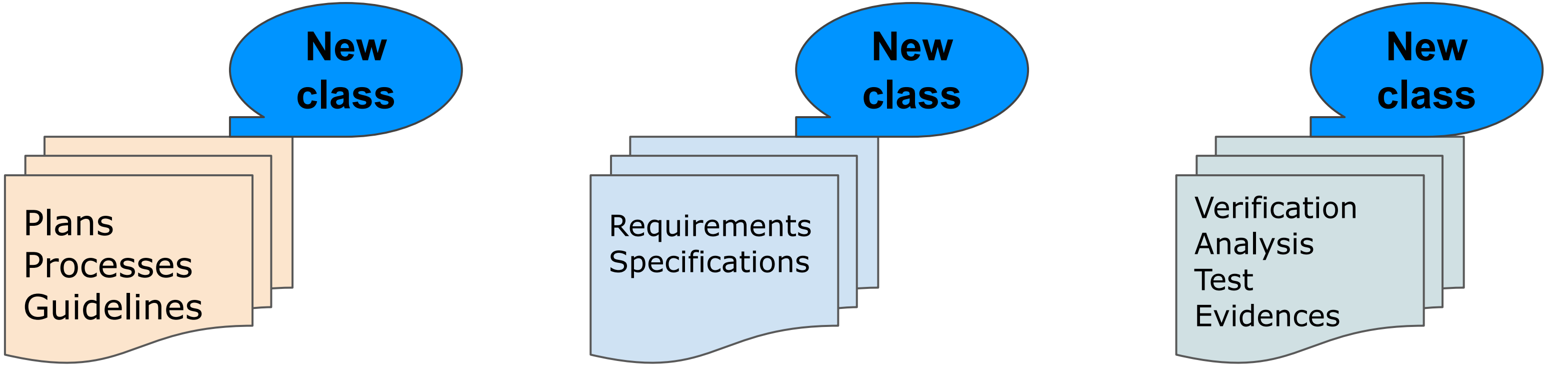
Content for SPDX 3.1

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

FuSa documentation structure

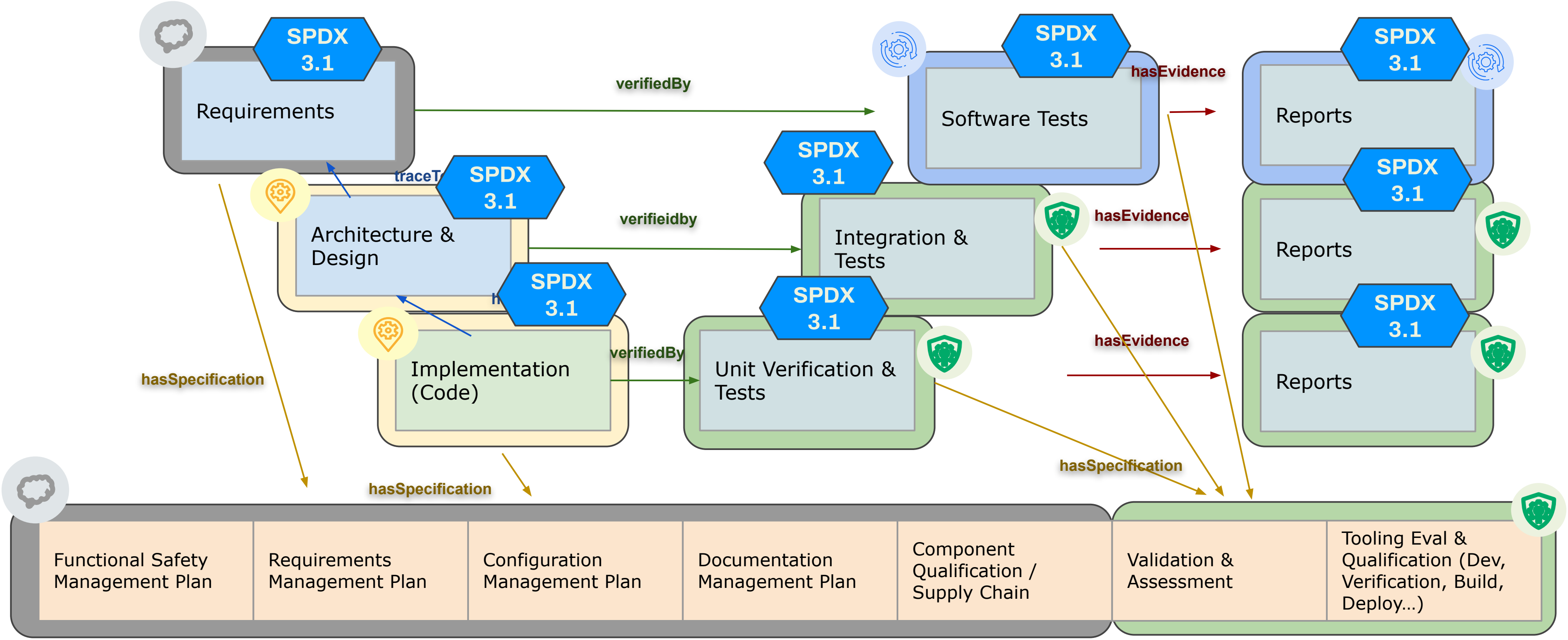
All FuSa related documentation is part of the Safety Case!

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!



Dependencies in a FuSa Project

SPDX 3.1-rc1



Requirements

Architecture & Design

Implementation (Code)

Software Tests

Integration & Tests

Unit Verification & Tests

Reports

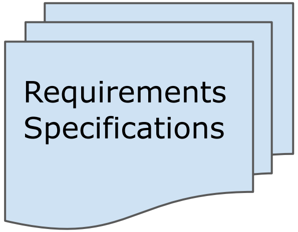
Reports

Reports

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...)

Classes for WPs - REQUIREMENT

RC SPDX 3.1



Requirements Specifications

Determining factors and assumptions:

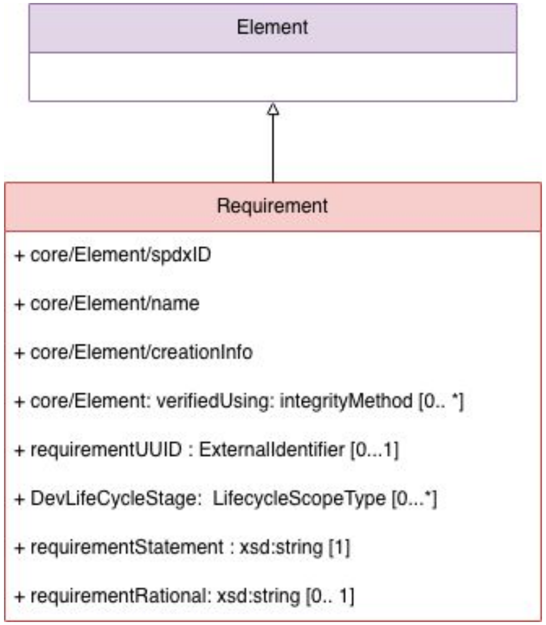
- A requirement describes a functional, non-functional or design need placed on an item (HW, SW, system, whatever can be the product)
- There are different sources of requirements
- Atomic REQUIREMENTS entities can be packaged to Requirement sets that then can become part of specifications ⇒ no new class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

Classes for WPs - REQUIREMENT

RC SPDX 3.1

Requirements

REQUIREMENT class



Element

Requirement

+ core/Element/spdxID

+ core/Element/name

+ core/Element/creationInfo

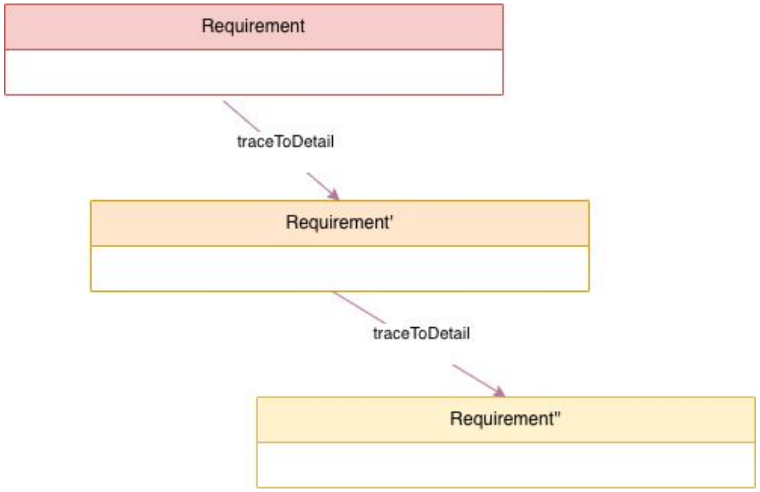
+ core/Element: verifiedUsing: integrityMethod [0.. *]

+ requirementUUID : ExternalIdentifier [0...1]

+ DevLifeCycleStage: LifecycleScopeType [0...*]

+ requirementStatement : xsd:string [1]

+ requirementRational: xsd:string [0.. 1]



Requirement

traceToDetail

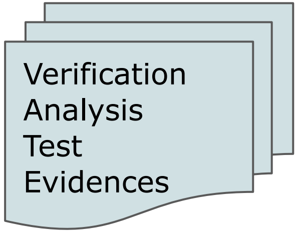
Requirement'

traceToDetail

Requirement''

Classes for WPs - VERIFICATION

RC SPDX 3.1



Verification Analysis Test Evidences

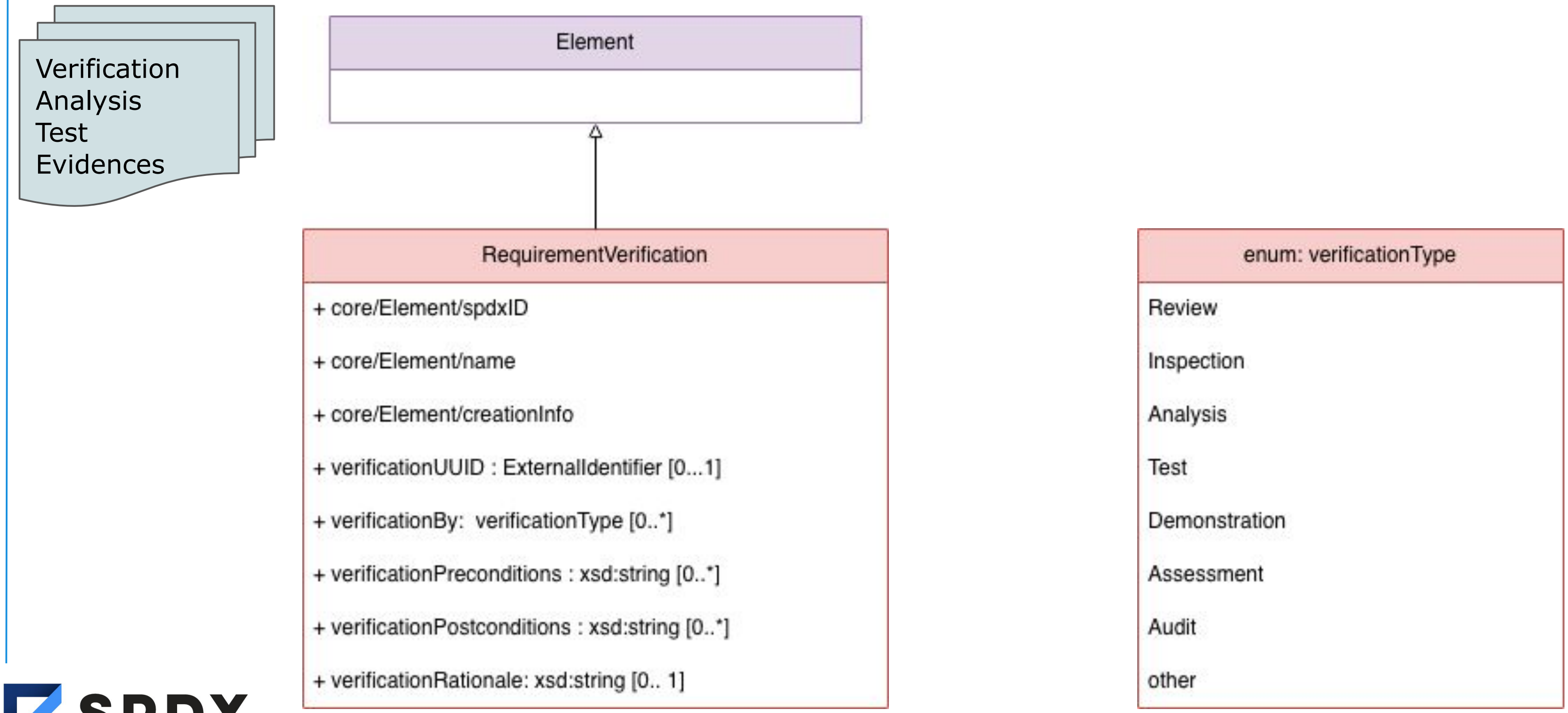
Determining factors and assumptions:

- 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

Element

RequirementVerification

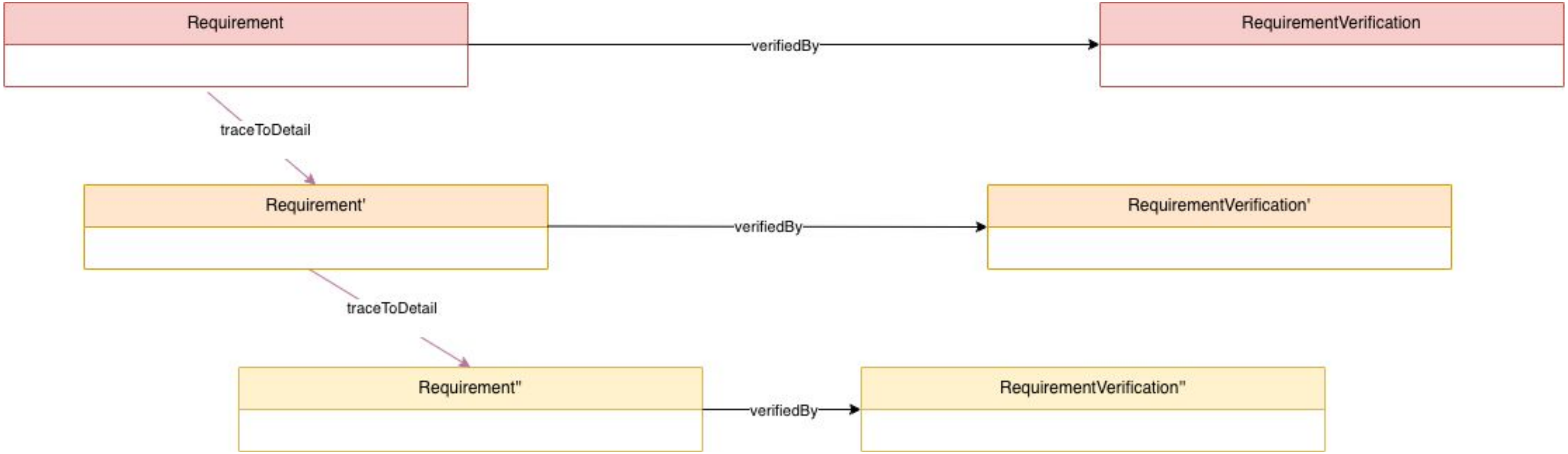
- + core/Element/spdxID
- + core/Element/name
- + core/Element/creationInfo
- + verificationUUID : ExternalIdentifier [0...1]
- + verificationBy: verificationType [0..*]
- + verificationPreconditions : xsd:string [0..*]
- + verificationPostconditions : xsd:string [0..*]
- + verificationRationale: xsd:string [0.. 1]

enum: verificationType

- Review
- Inspection
- Analysis
- Test
- Demonstration
- Assessment
- Audit
- other

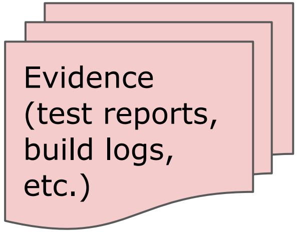
Req and Ver - Relationships

RC SPDX 3.1



Classes for WPs - EVIDENCE

RC SPDX 3.1



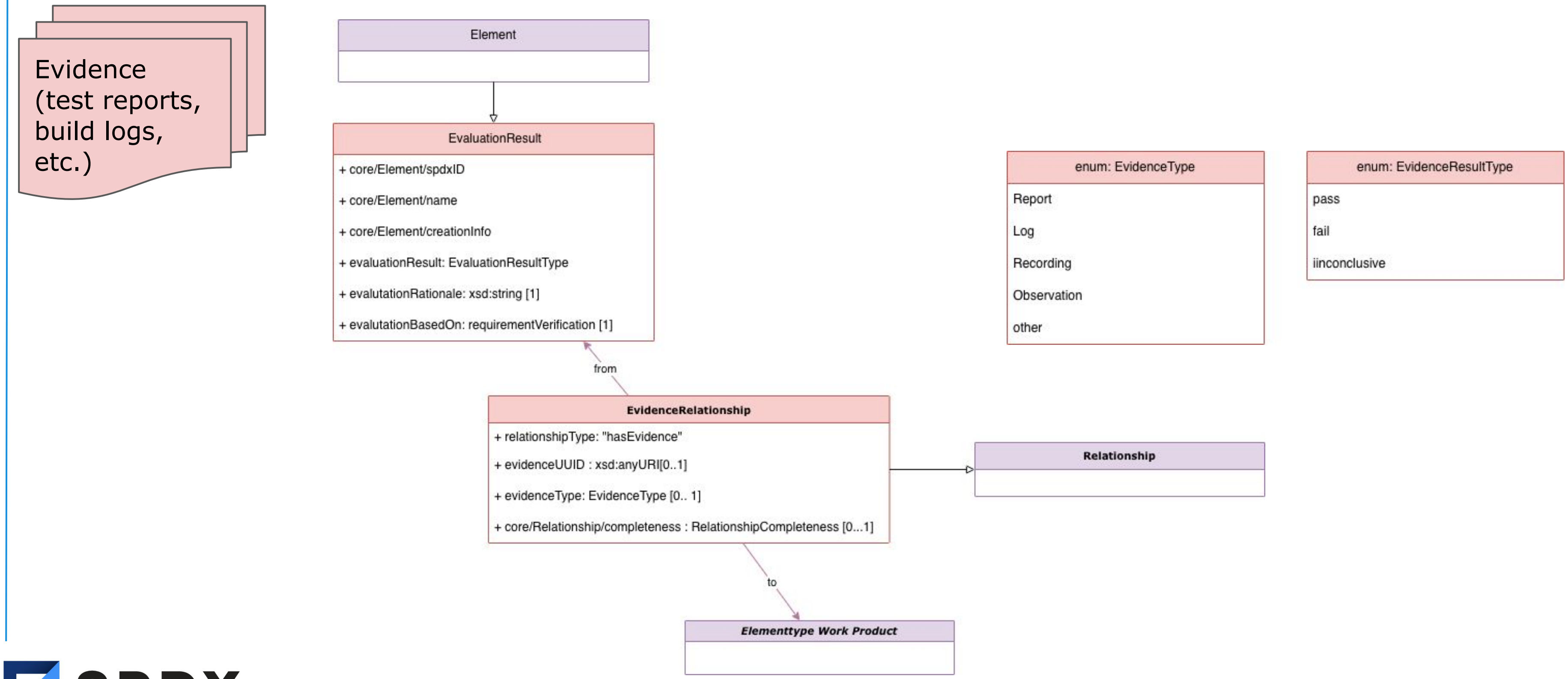
Evidence (test reports, build logs, etc.)

Determining factors and assumptions:

- EVIDENCES attest a certain level of compliance of
  - a tested item (code) with its acceptance criteria (requirement), using the test process and
  - For a specific pair of verification input, verification specification and verification results
- EVIDENCES are highly coupled with VERIFICATION

Classes for WPs - EVIDENCE

RC SPDX 3.1



Evidence (test reports, build logs, etc.)

Element

EvaluationResult

- + core/Element/spdxID
- + core/Element/name
- + core/Element/creationInfo
- + evaluationResult: EvaluationResultType
- + evalutationRationale: xsd:string [1]
- + evalutationBasedOn: requirementVerification [1]

EvidenceRelationship

- + relationshipType: "hasEvidence"
- + evidenceUUID : xsd:anyURI[0..1]
- + evidenceType: EvidenceType [0.. 1]
- + core/Relationship/completeness : RelationshipCompleteness [0...1]

Elementtype Work Product

Relationship

enum: EvidenceType

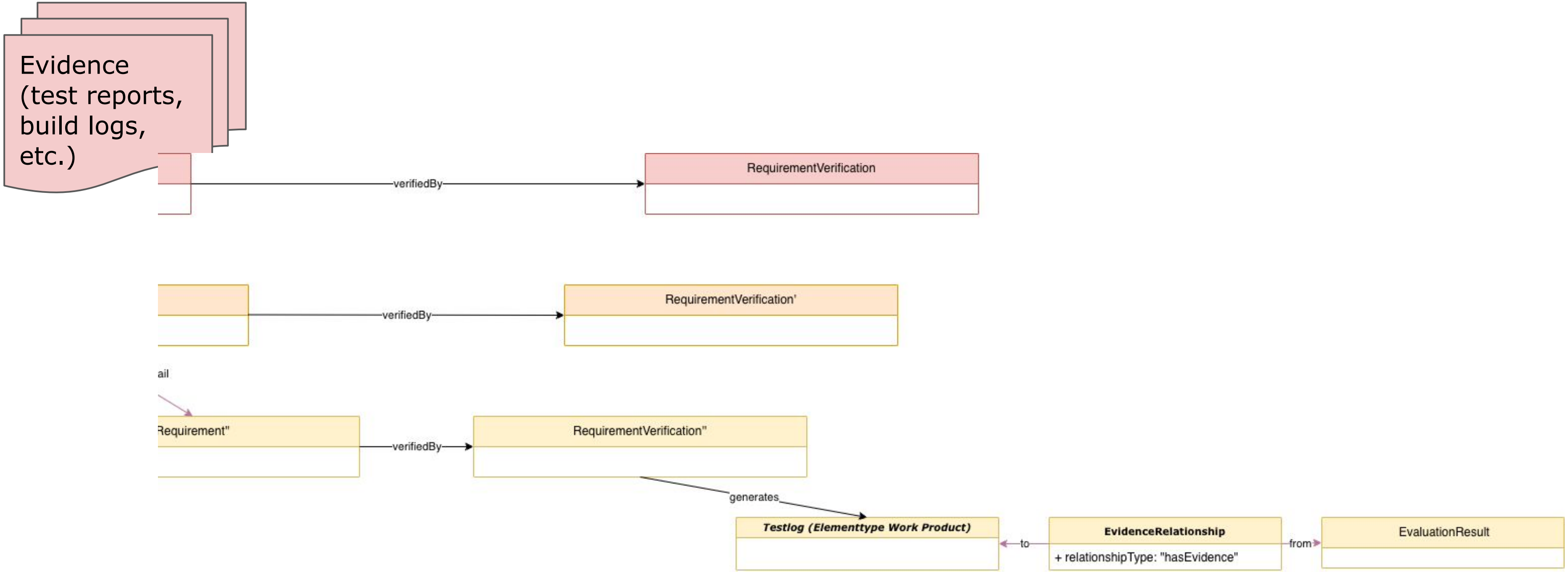
- Report
- Log
- Recording
- Observation
- other

enum: EvidenceResultType

- pass
- fail
- iinconclusive

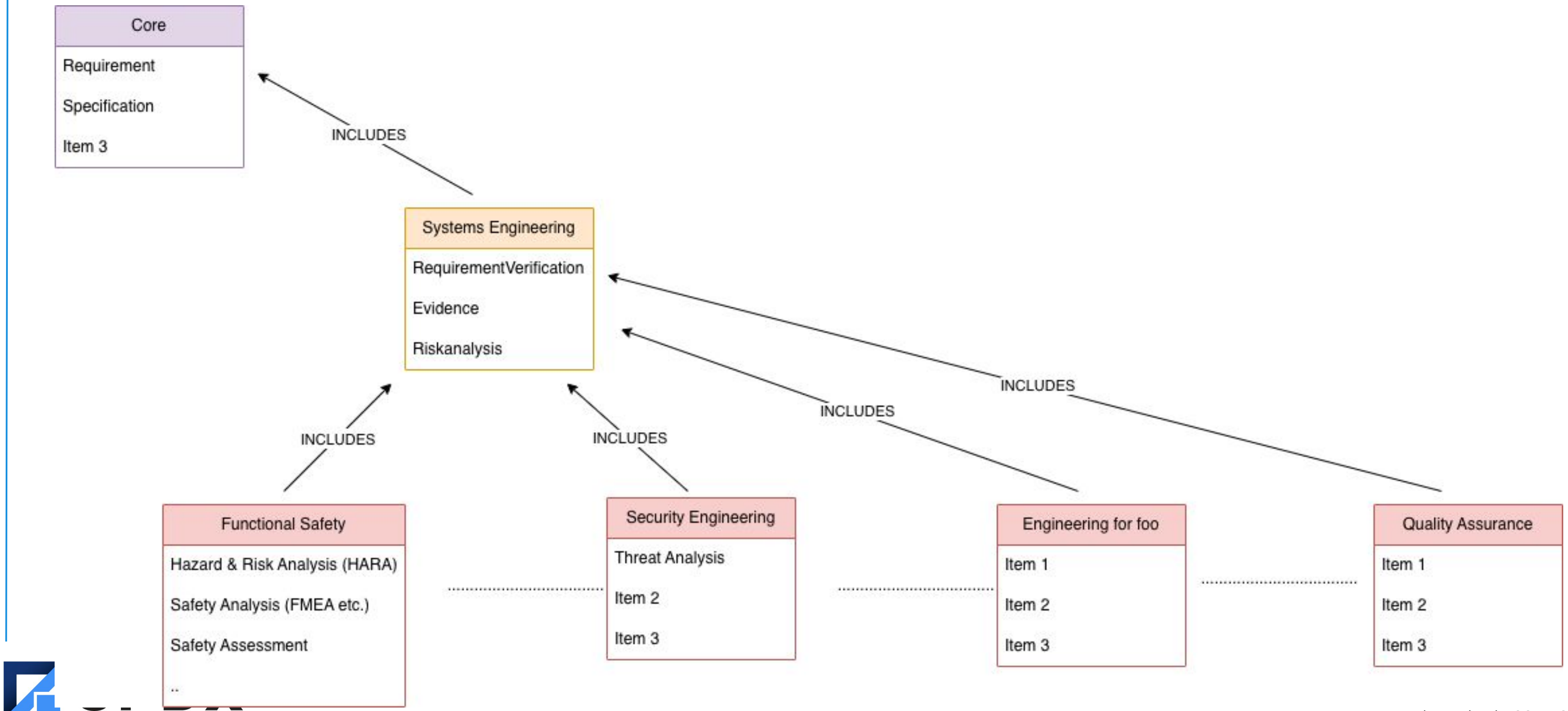
Classes for WPs - EVIDENCE

RC SPDX 3.1



A few things in the pipeline…

YES - we know its not just Functional Safety



Core
- Requirement
- Specification
- Item 3

Systems Engineering
- RequirementVerification
- Evidence
- Riskanalysis

Functional Safety
- Hazard & Risk Analysis (HARA)
- Safety Analysis (FMEA etc.)
- Safety Assessment
- ..

Security Engineering
- Threat Analysis
- Item 2
- Item 3

Engineering for foo
- Item 1
- Item 2
- Item 3

Quality Assurance
- Item 1
- Item 2
- Item 3

A few things in the pipeline…

Pushed to RC 2 (or SPDX 3.2)

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

Conclusions (so far)

- What we started for Functional Safety is universally valid for other System Engineering flavors
- Enabling 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

… to be continued

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