SPDX FuSa Status

Elisa Workshop Lund 2025

Nicole Pappler, AlektoMetis

SPDX FuSa

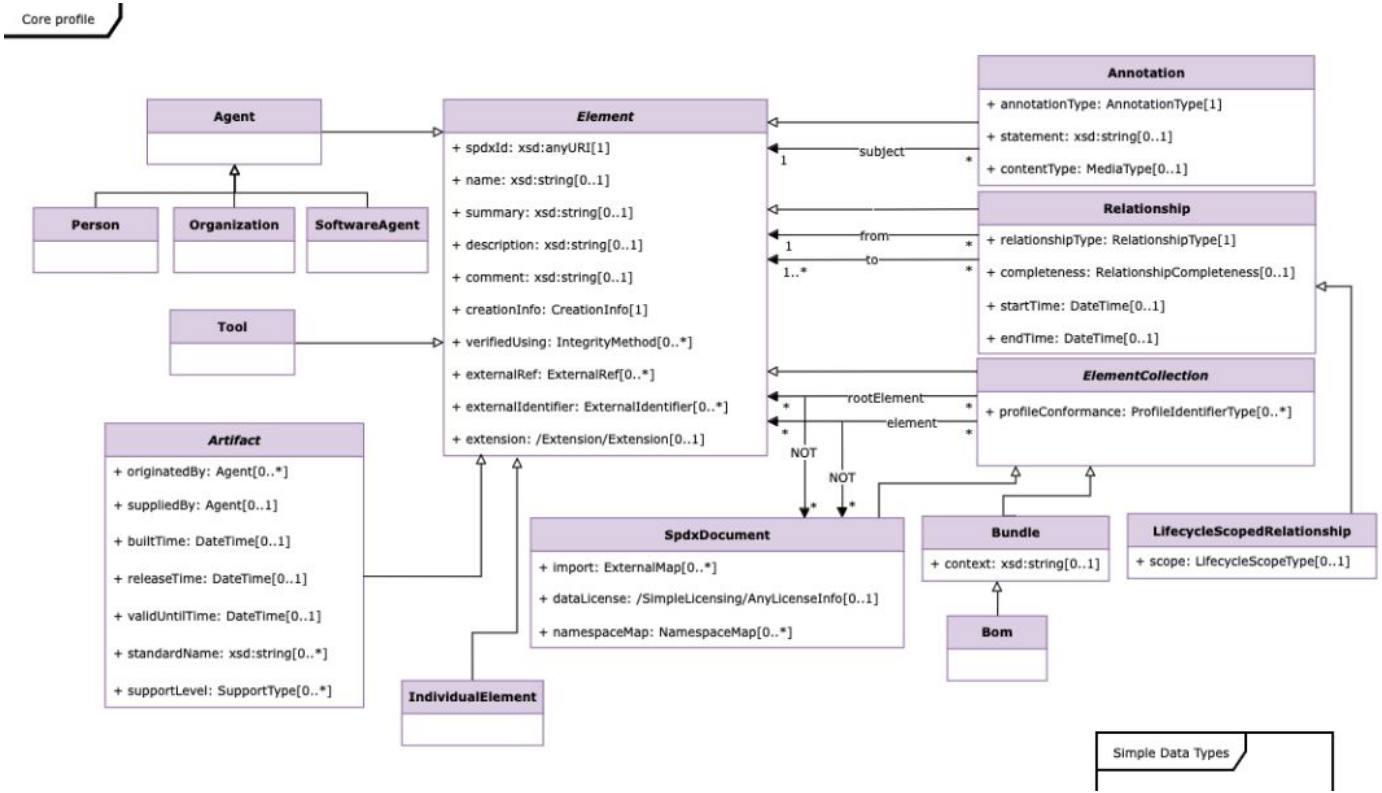
Goal:

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

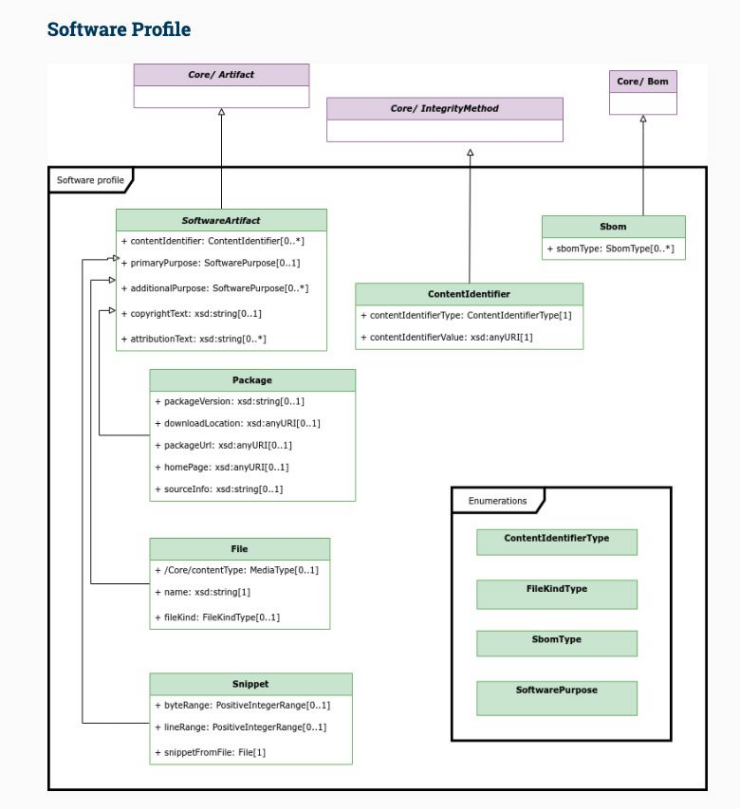
SPDX 3.0 model

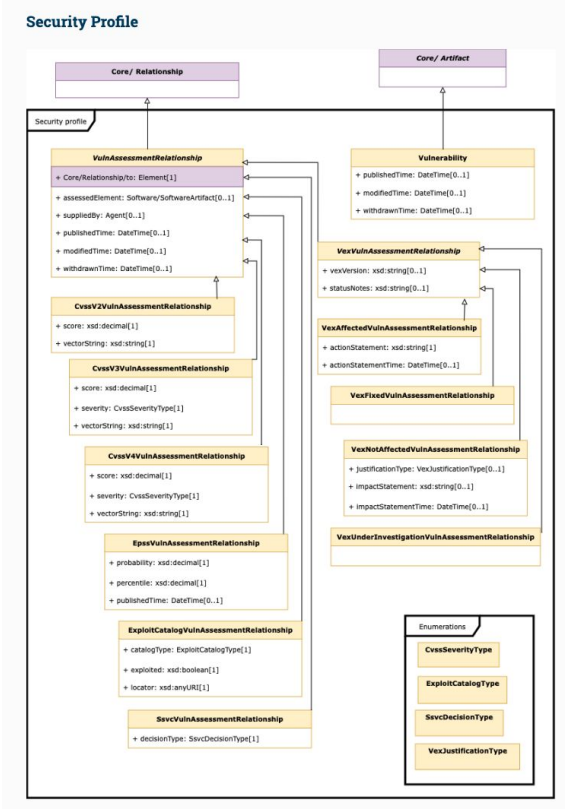


Use existing 3.0 model

Similar SPDX profiles

SPDX SAFETY

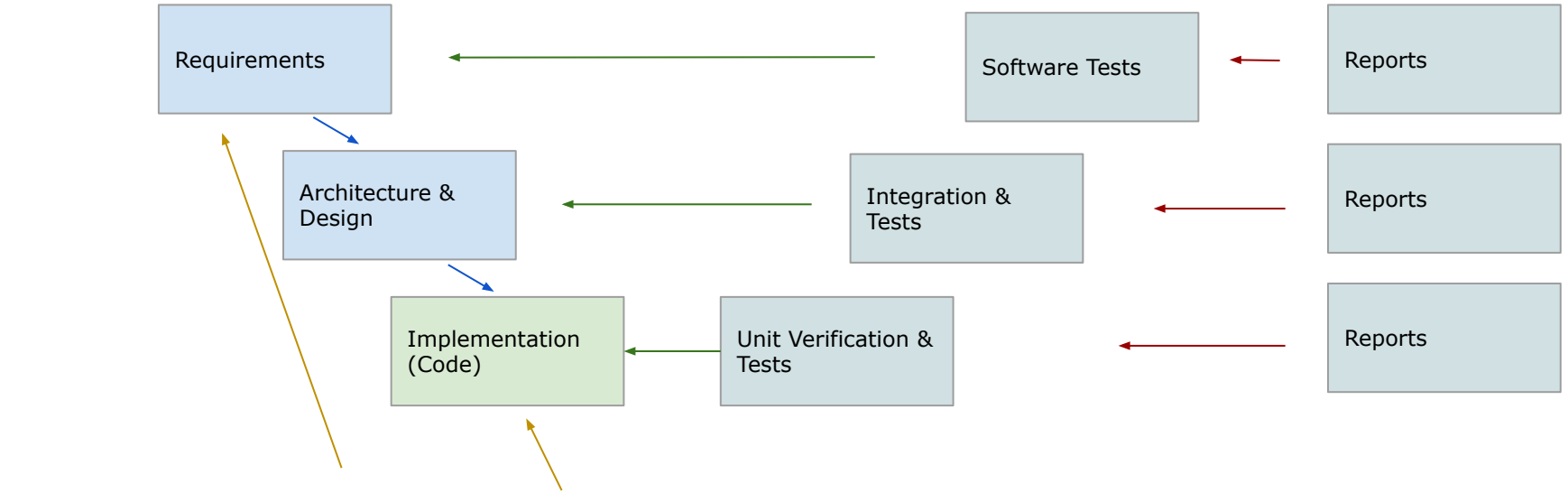




… and many more

SPDX

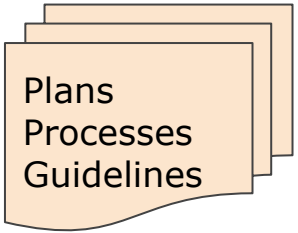
Dependencies in a FuSa Project

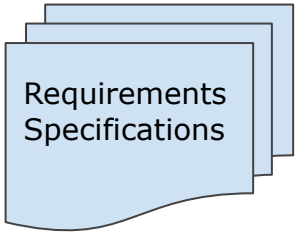


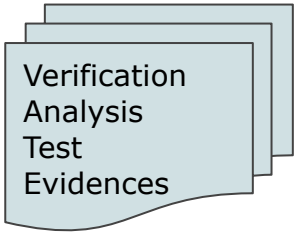
Functional Safety Management Plan	Requirements Management Plan	Configuration Management Plan	Documentation Management Plan	Component Qualification / Supply Chain	Validation & Assessment	Tooling Eval & Qualification (Dev, Verification, Build, Deploy…)

FuSa documentation structure

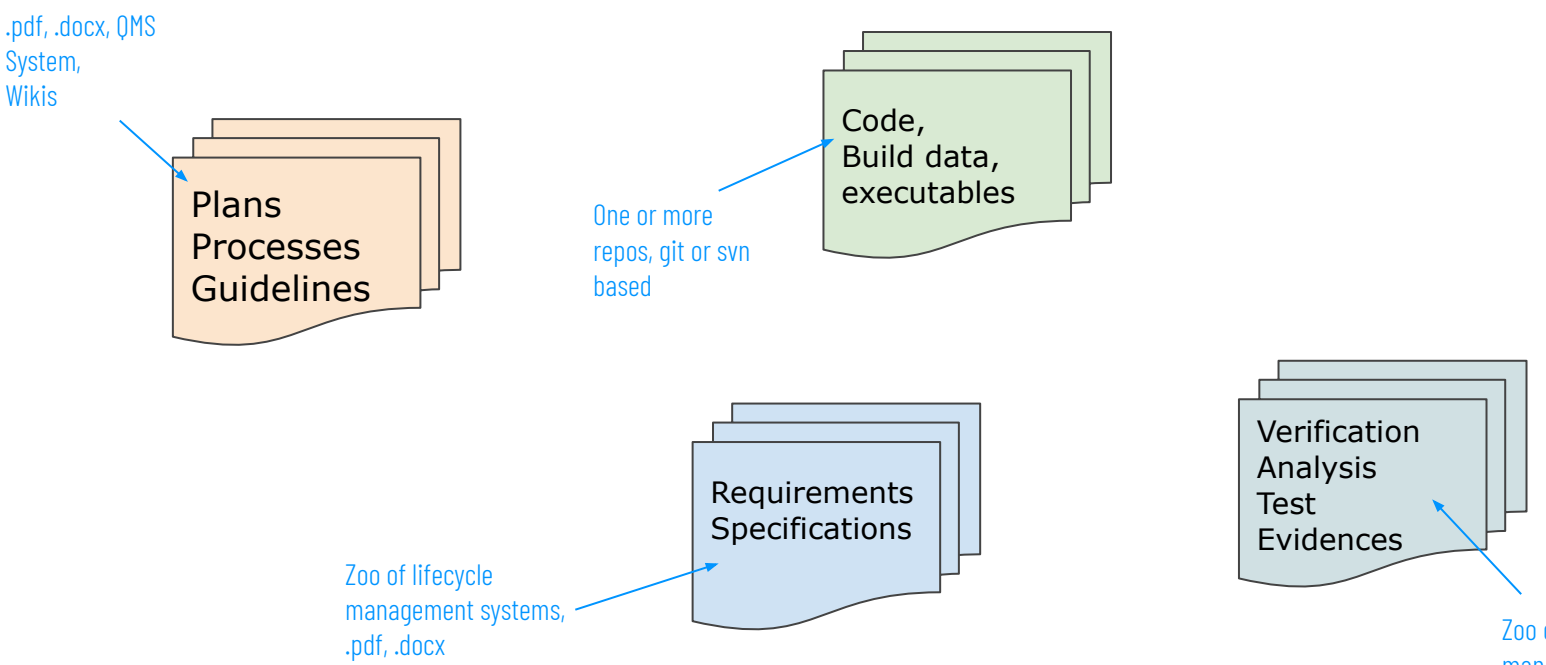
All FuSa related documentation is part of the Safety Case!
Think of all these documents as part of the release - each document is part of the Bill of Material, as is each screw, each microcontroller and each piece of software!







Data Structure of current FuSa projects...



Data Structure of current FuSa projects

.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

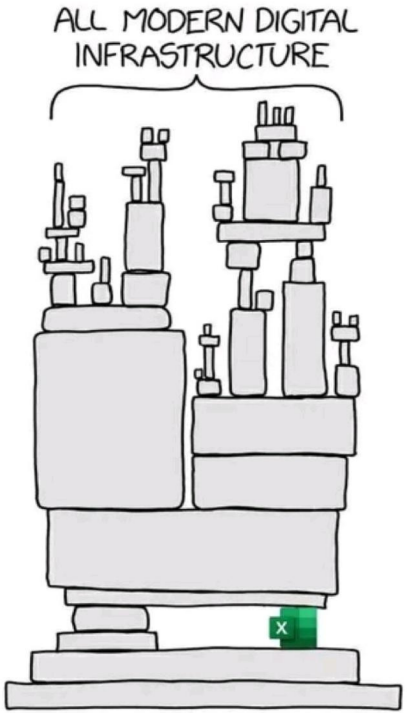
Verification
Analysis
Test
Evidences

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

No 1 Safety Information Exchange Format

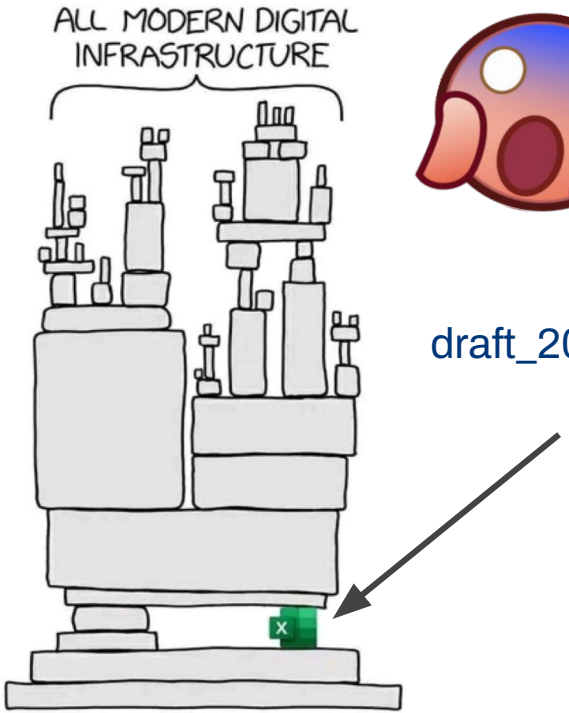
Any guesses????

No 1 Safety Information Exchange Format



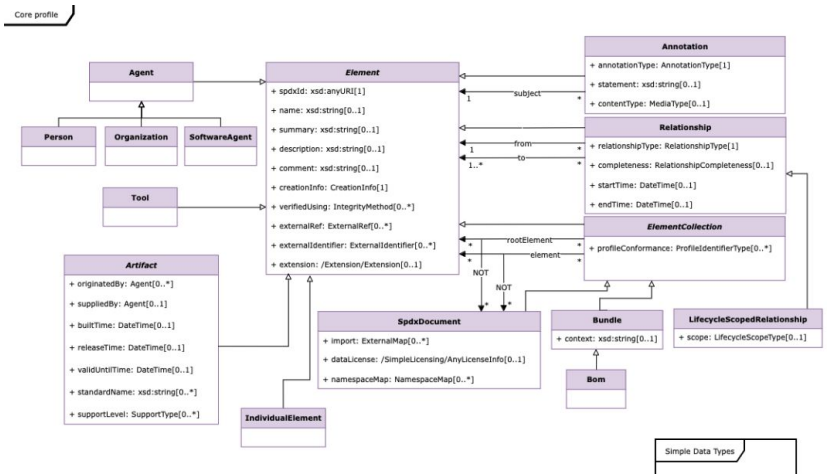
Emoji by emojidex

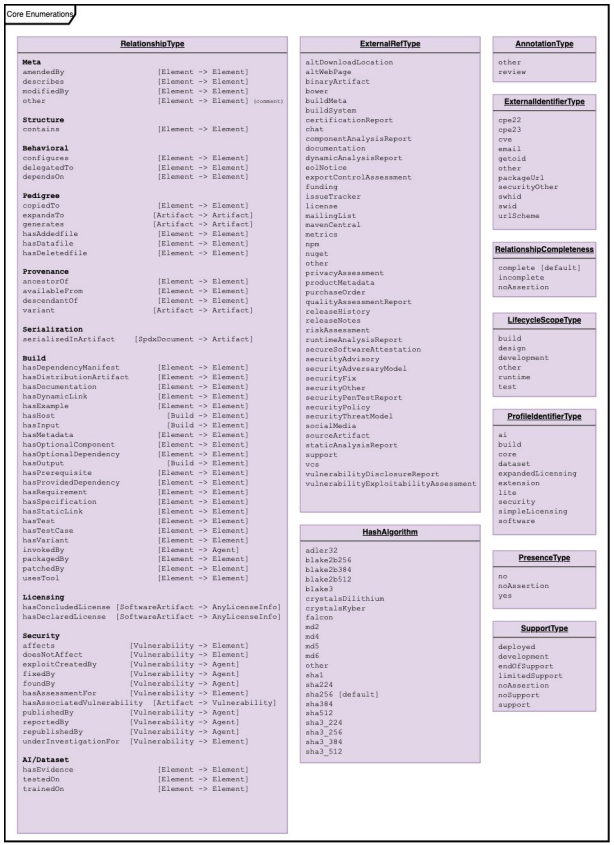
No 1 Safety Information Exchange Format



draft_2005TemplateSafetyCase_thisproject_final_forTraceingv06.xls

Emoji by emojidex





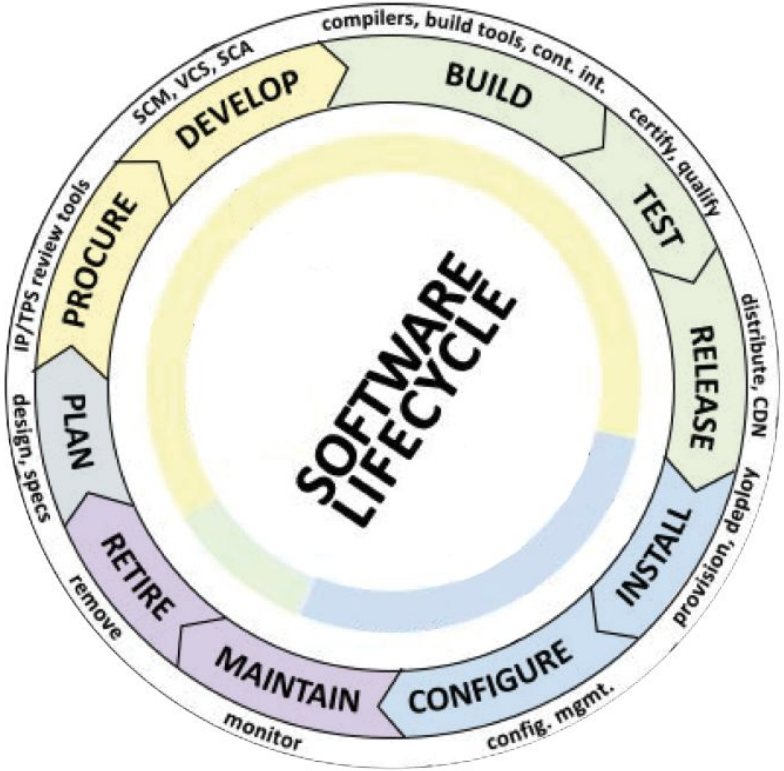
Generate SBOMS when the data is known - by the projects

SPDX SAFETY

Source SBOM

Design SBOM

Runtime SBOM



Build SBOM

Deployed SBOM

SPDX

Exchange SPDX Safety SBOMs

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

Design SBOM

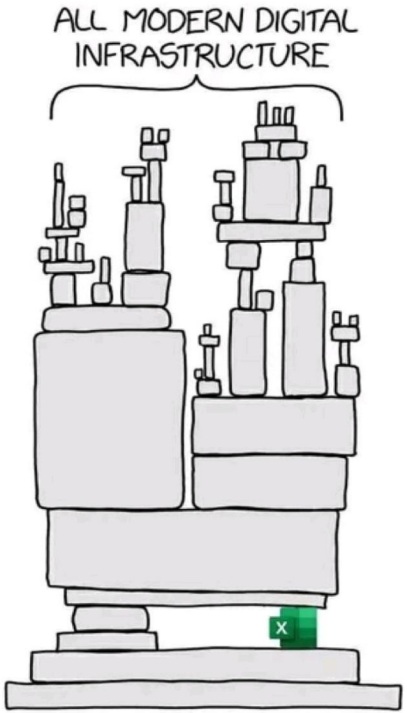
Source SBOM

Build SBOM

Runtime SBOM

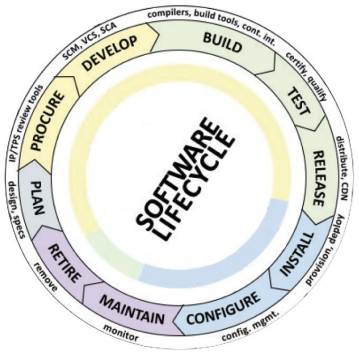
Deployed SBOM

Safety Information Exchange Format?



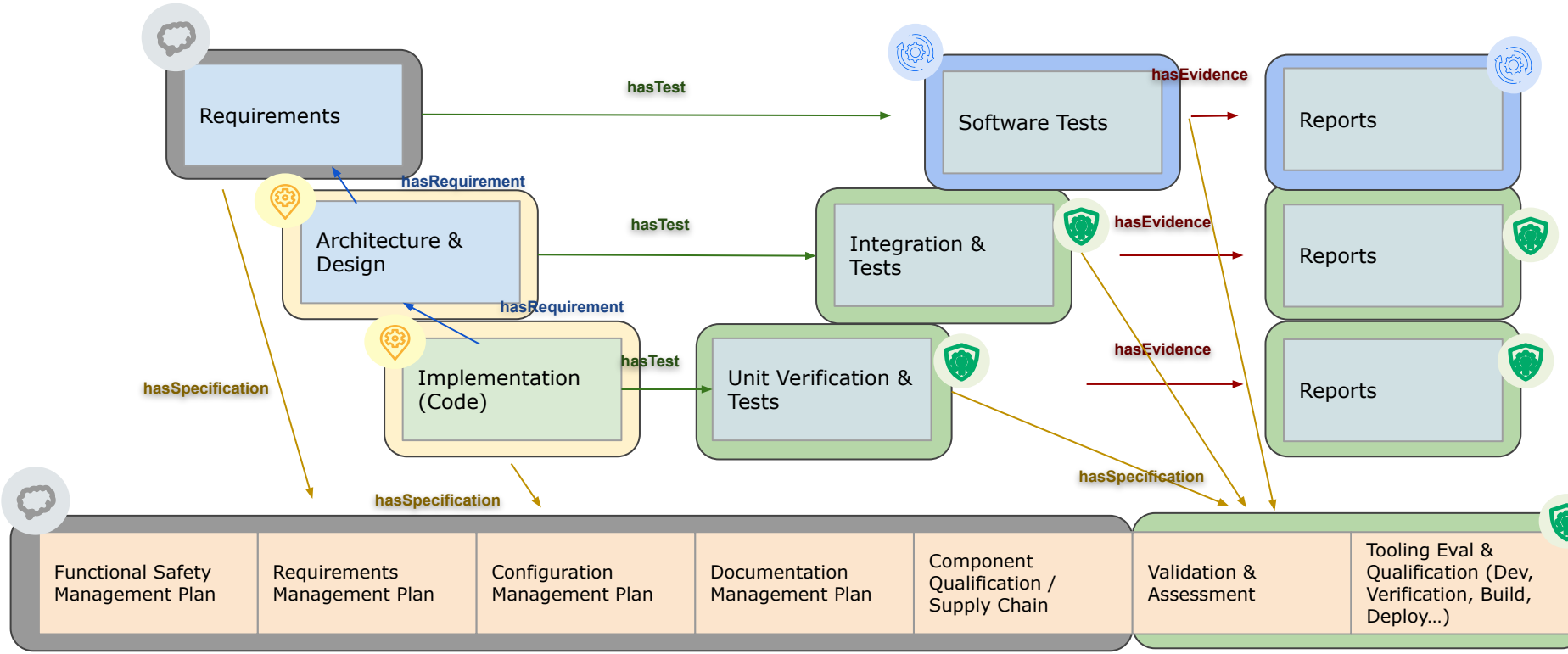
SPDX 3.1

SPDX Safety SBOM!



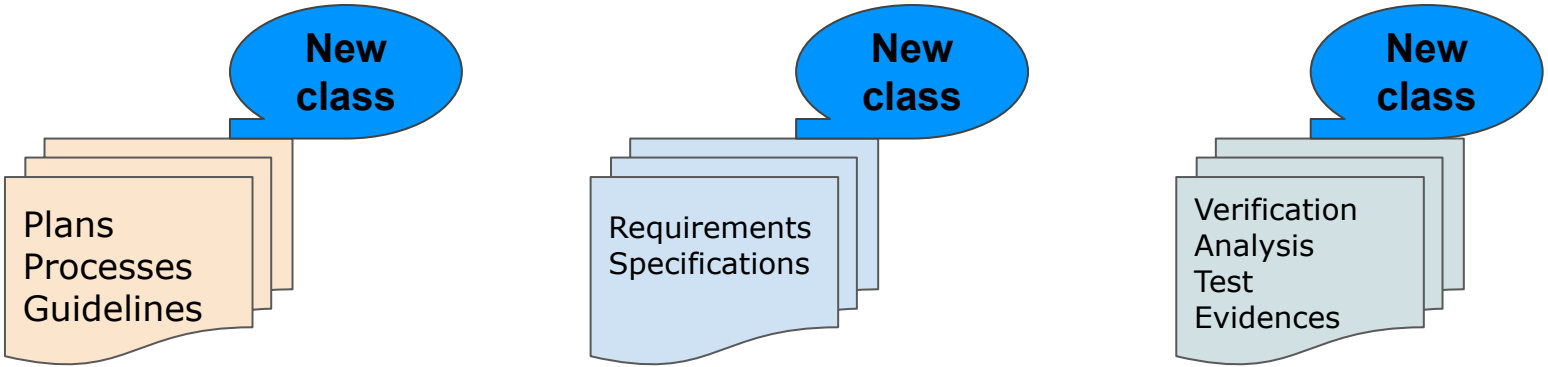
... instead of inconsistent Spreadsheets, manual import/export of half decent ReqIFs...

Dependencies in a FuSa Project

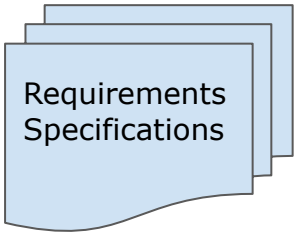


FuSa documentation structure

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!



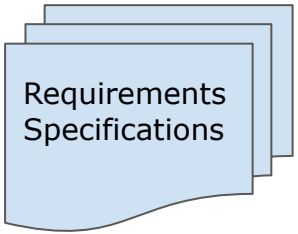
Classes for WPs - REQUIREMENT



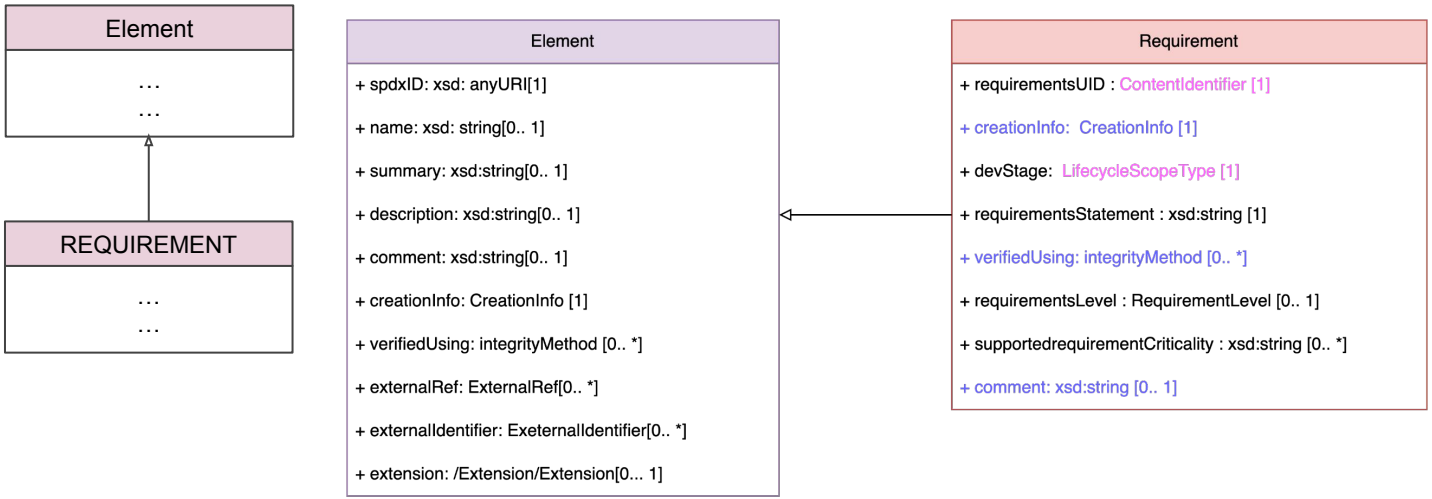
Determining factors and assumptions:

- 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

Classes for WPs - REQUIREMENT

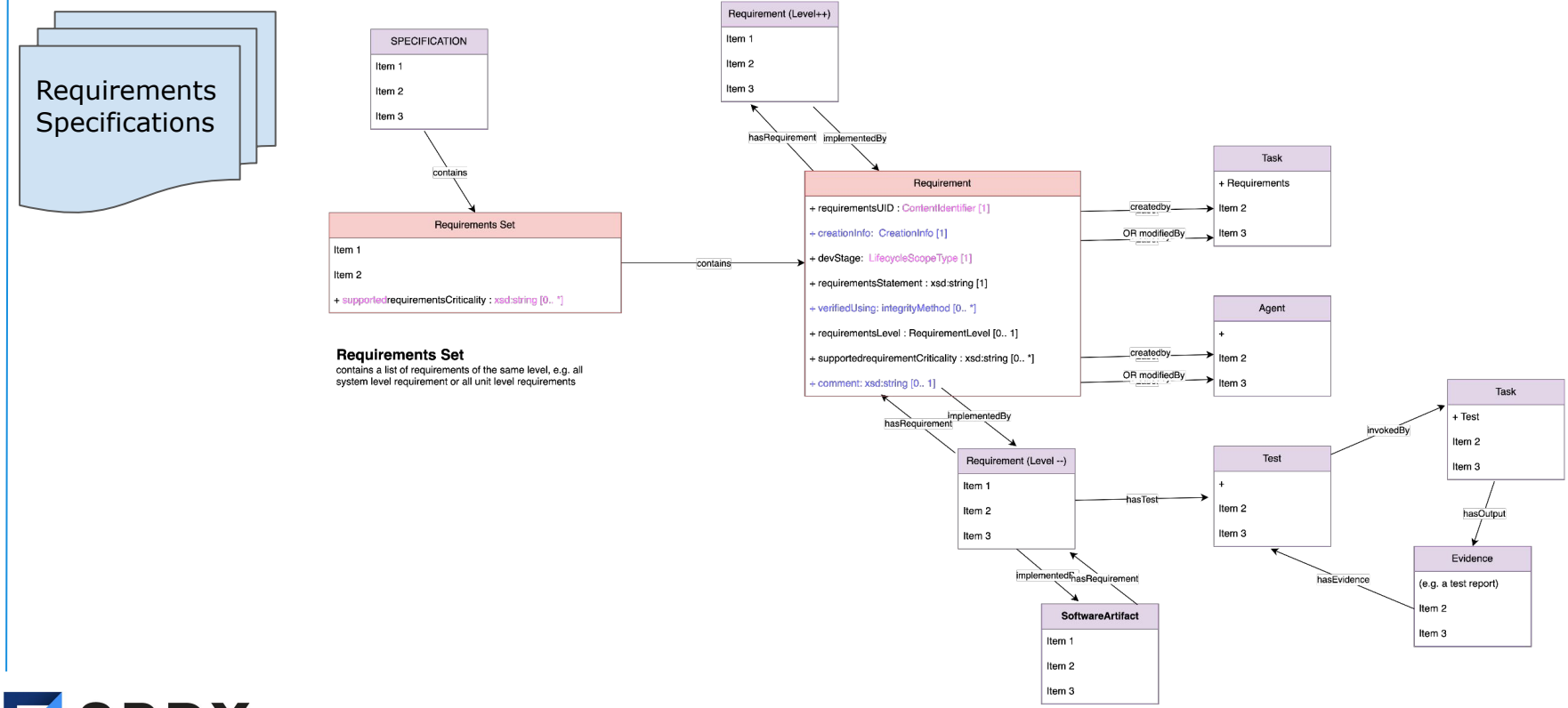


REQUIREMENT class



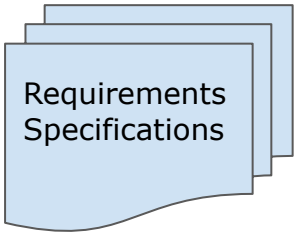
Example for implemented Requirements:
https://github.com/zephyrproject-rtos/reqmgmt/blob/main/docs/system_requirements/system_requirements.sgra

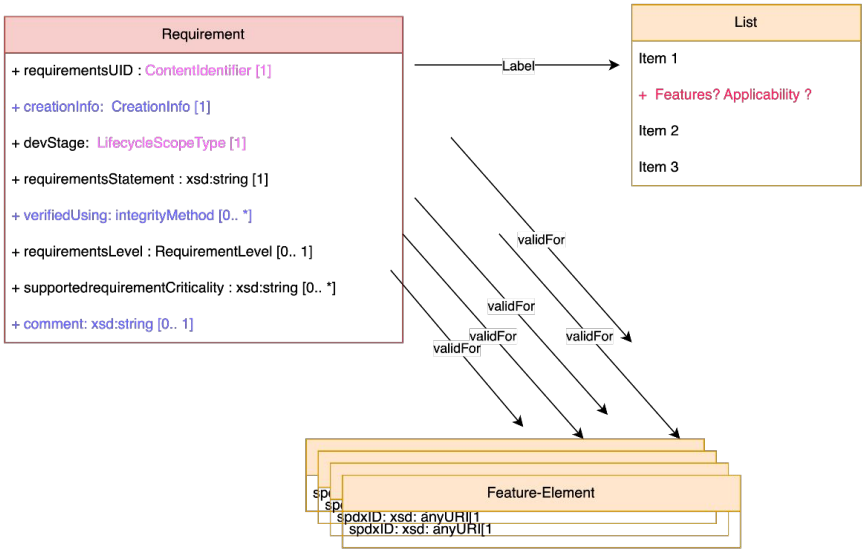
REQUIREMENT - relationships

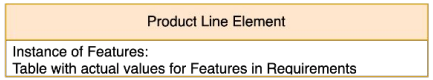


REQUIREMENT & product line engineering

WIP - Open for suggestions!







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

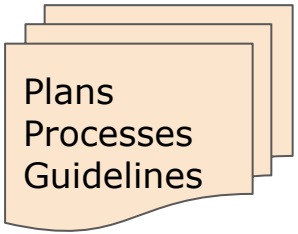
Feature-applicability FA3:

BEV, PHEV, FUEL

Product Configuration 1:

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

Classes for work products - TASK and Process

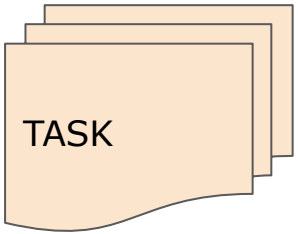


Determining factors and assumptions:

- 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

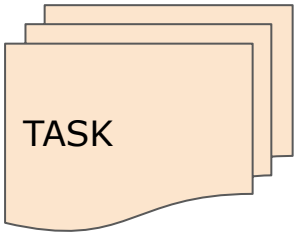
Other classes - TASK



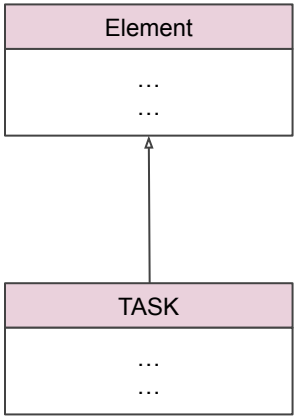
Determining factors and assumptions:

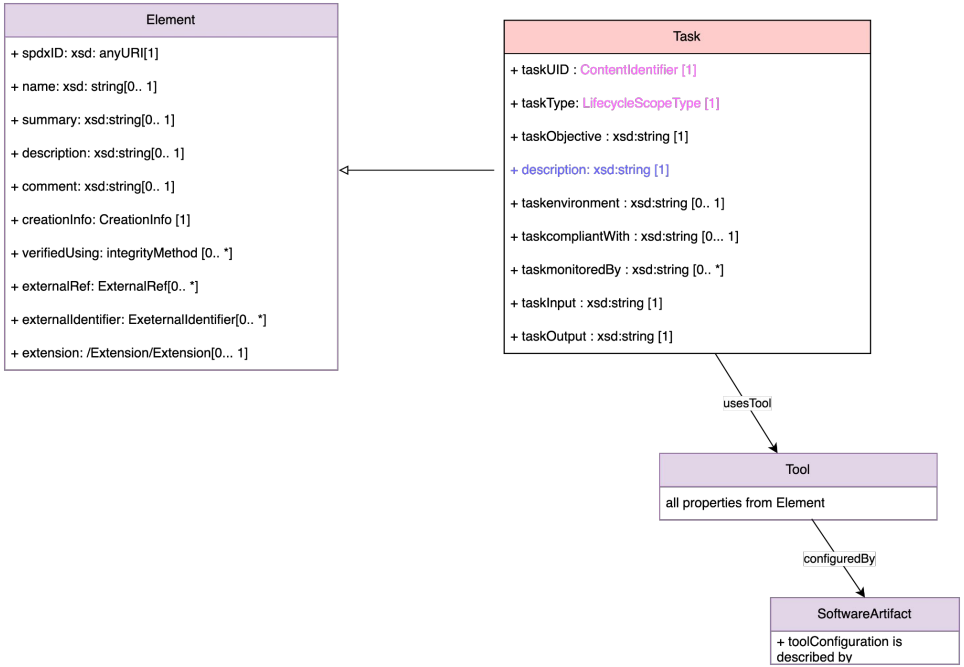
- A TASK is a specific unit of work that contributes to the completion of a project or an item
- TASKs can be of different types
- Need at least the content of ELEMENT
- An TASK as a minimum needs:
  - An Objective what the tasks aims to achieve
  - Preconditions and neccessary inputs, resources
  - 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

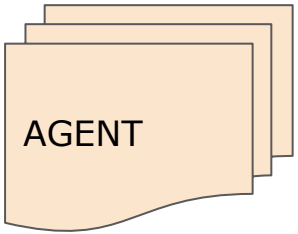


TASK class

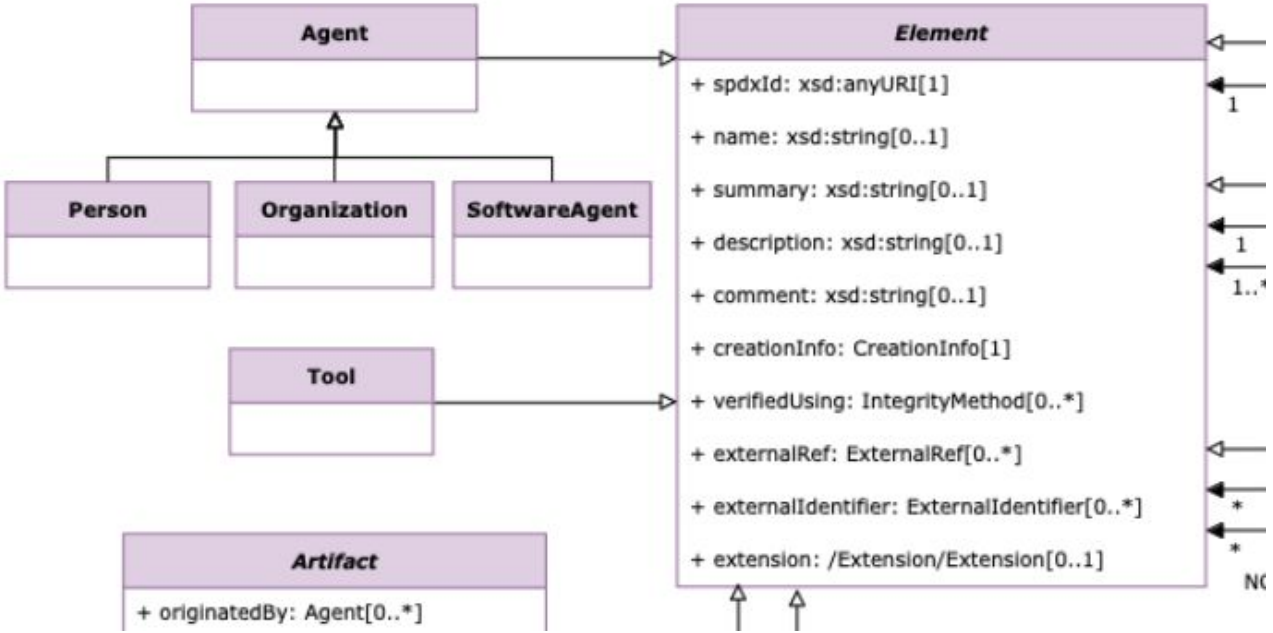




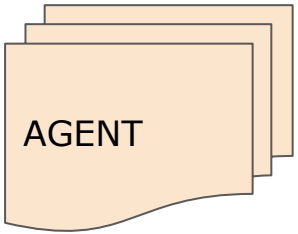
Classes for work products - AGENT



AGENT class



Classes for work products - AGENT



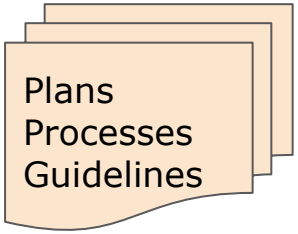
AGENT class

- 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]

Classes for work products - enums



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
...

Task vs HW Action

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,

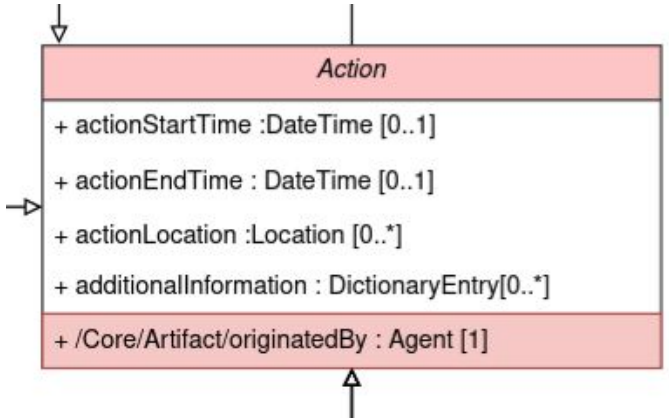
Task vs HW Action

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.

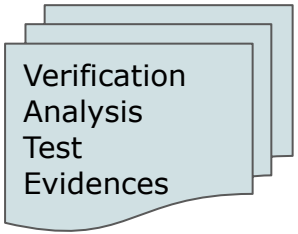
Building the software and running the tests, as well as the creation of the Evidence artefacts for it, is all described by TASKs, there is no specific need for a Location, Duration or similar. The buildTime is already defined by ARTEFACT.

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?

Classes for WPs - VERIFICATION

TBD - Open for suggestions!



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

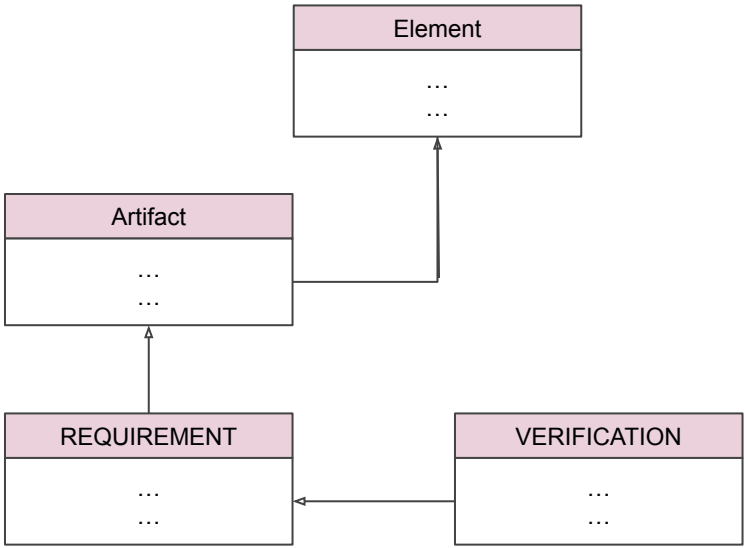
⇒ need class for VERIFICATION specification to have something that describes test cases

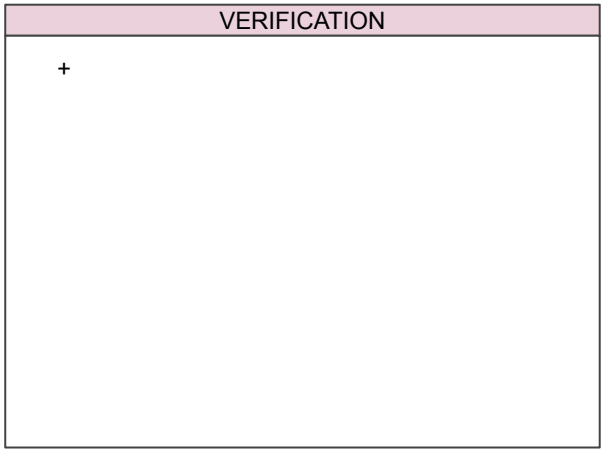
Classes for WPs - VERIFICATION

TBD - Open for suggestions!

Verification
Analysis
Test
Evidences

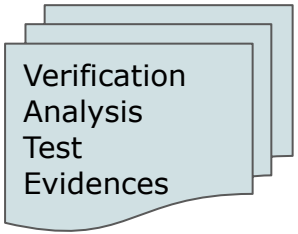
VERIFICATION class





Classes for WPs - VERIFICATION

TBD - Open for suggestions!

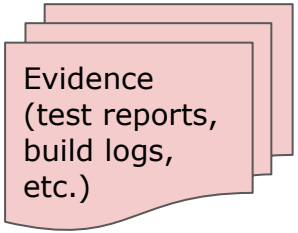


VERIFICATION class

Classes for WPs - EVIDENCE

SPDX SAFETY

TBD - Open for suggestions!



Determining factors and assumptions:

- EVIDENCEs are created based on a PROCESS with a ProcessType for verification
- EVIDENCEs are created applying ACTIONs 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

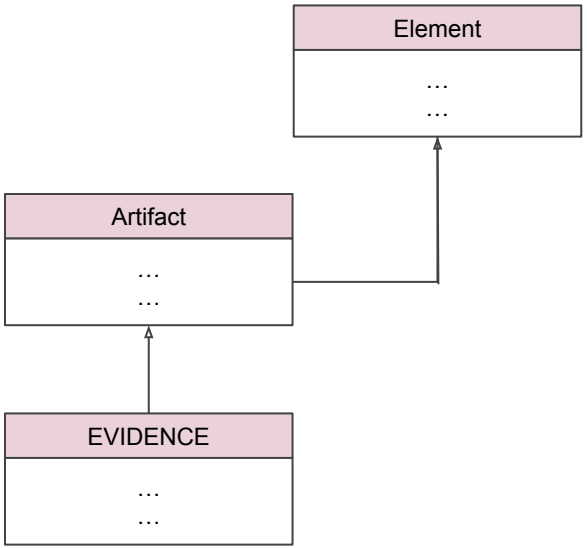
SPDX

Classes for WPs - EVIDENCE

TBD - Open for suggestions!

Evidence
(test reports,
build logs,
etc.)

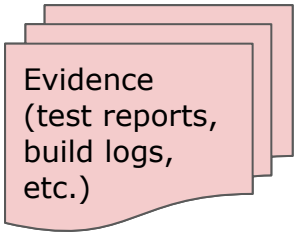
EVIDENCE class



EVIDENCE
+ EvidenceType: enum + EvidenceResult: enum +

Classes for WPs - EVIDENCE

TBD - Open for suggestions!



EVIDENCE class - enums we need

- **EvidenceType**: report, log, video, ...
- **EvidenceResult**: passed, partially passed, partially run, skipped, failed, ..

Example for implemented Requirements:
https://github.com/zephyrproject-rtos/reqmgmt/blob/main/docs/system_requirements/system_requirements.sgra

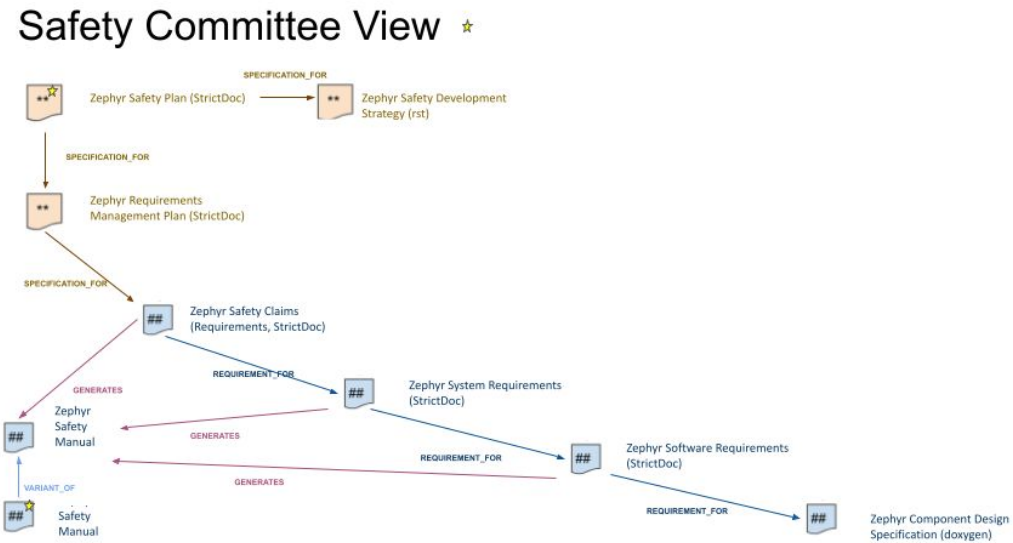
… to be continued

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

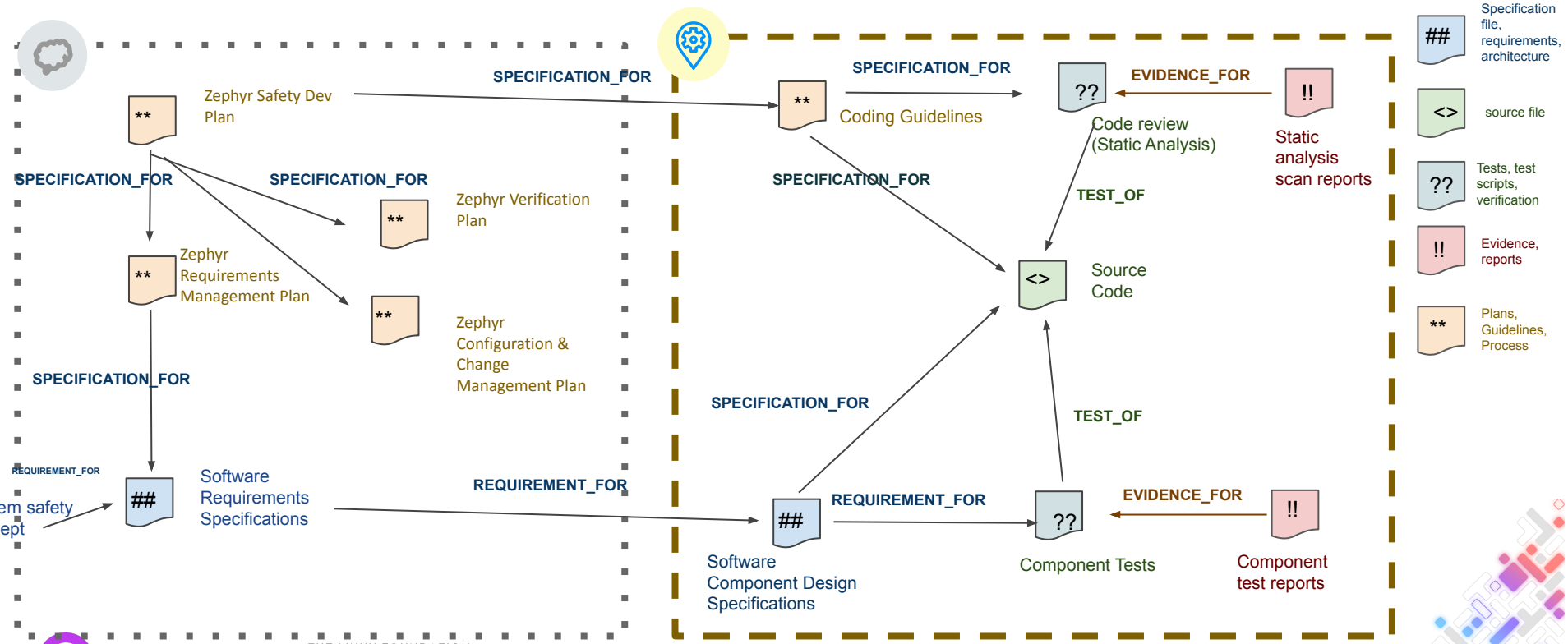
Appendix

Zephyr Requirements Management

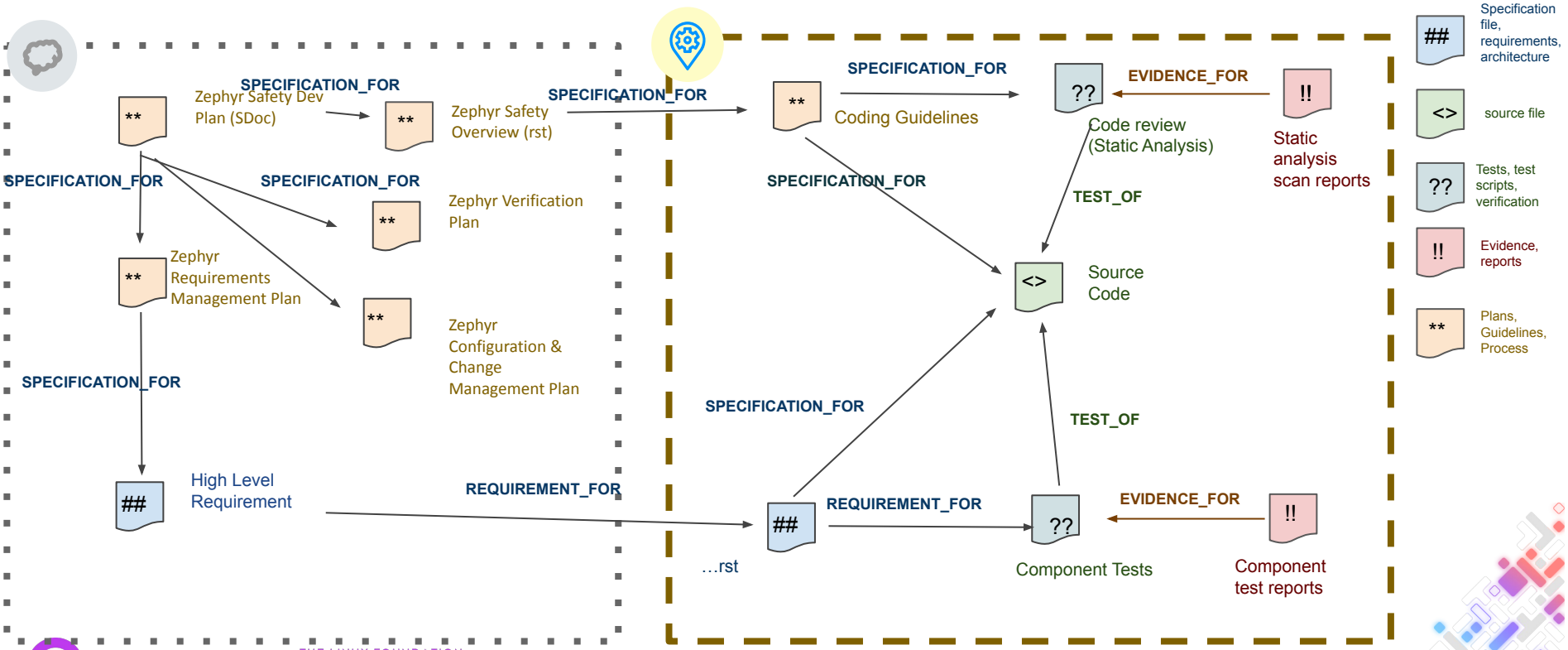
Requirements Management Knowledge Model



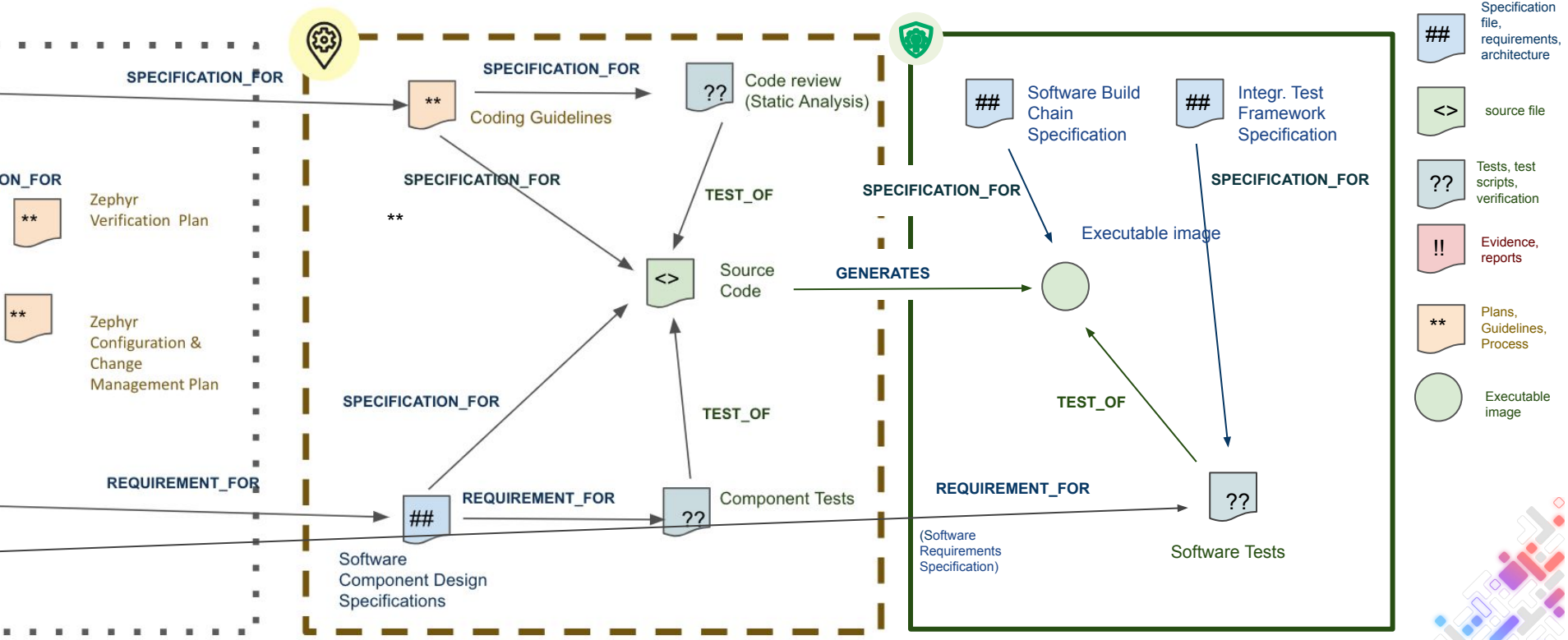
Dependencies of Safety Plan, Safety Claim, Req, Design and Code



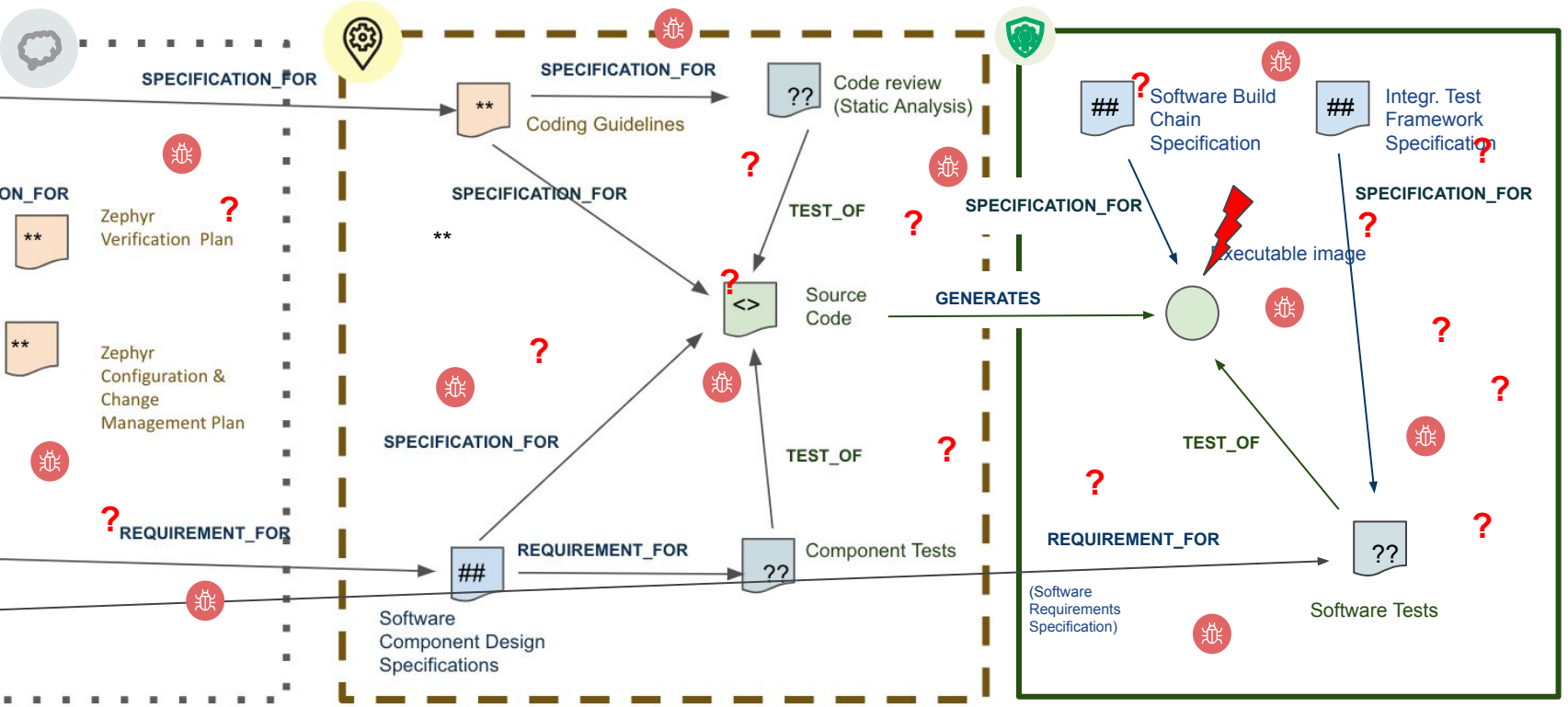
Design SBOM to Source SBOM



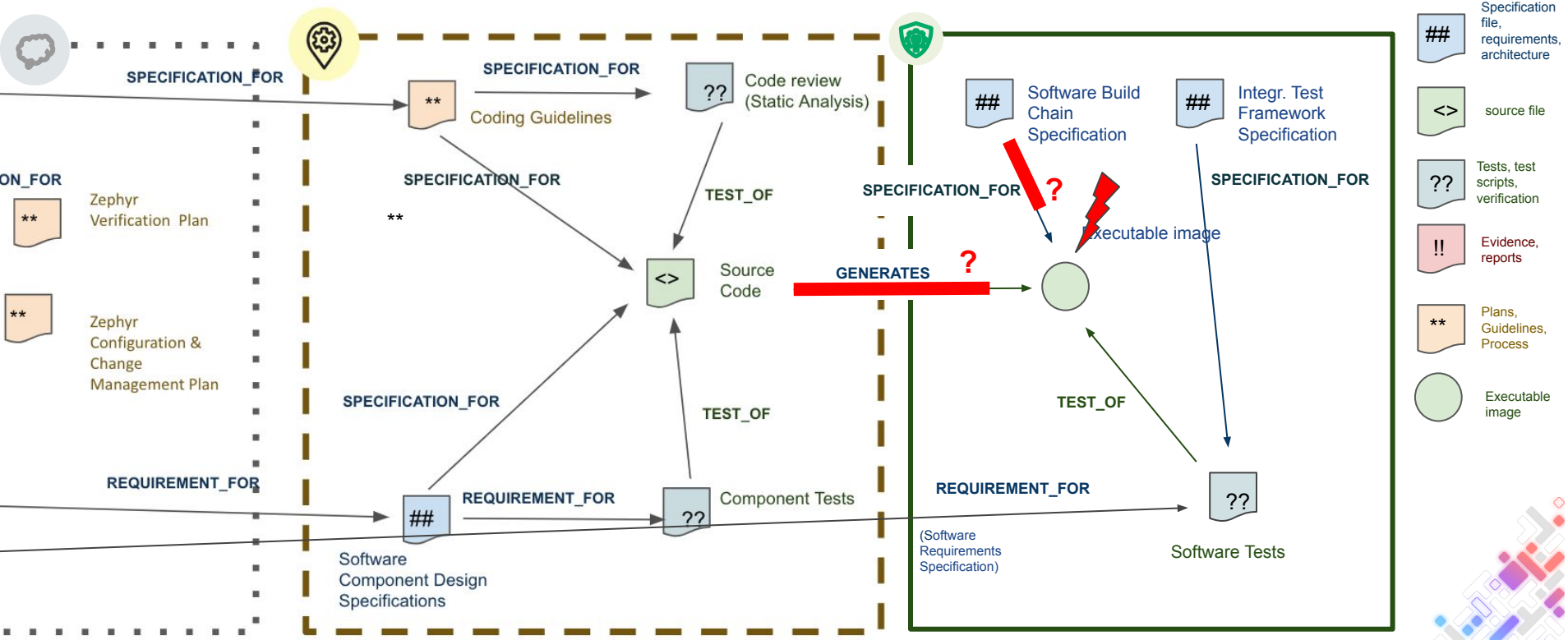
Source SBOM to Build SBOM



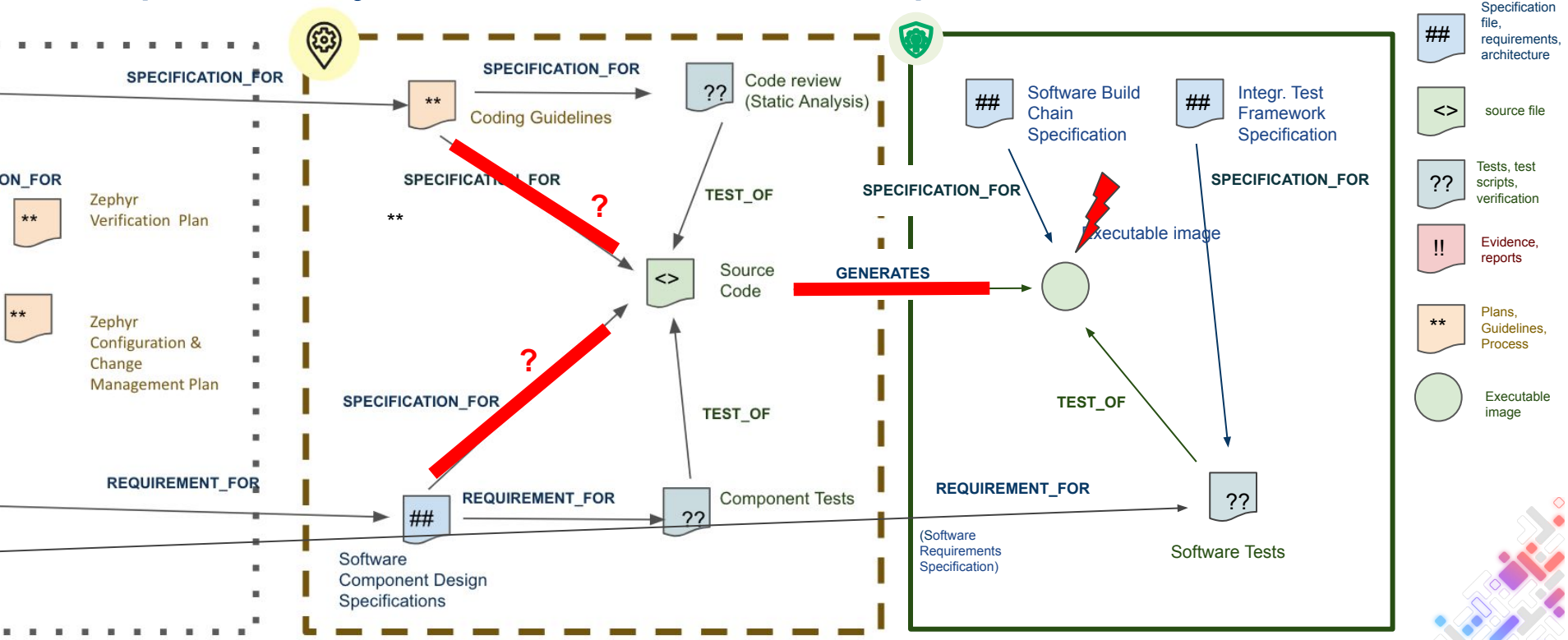
Dependency Identification on Component Level



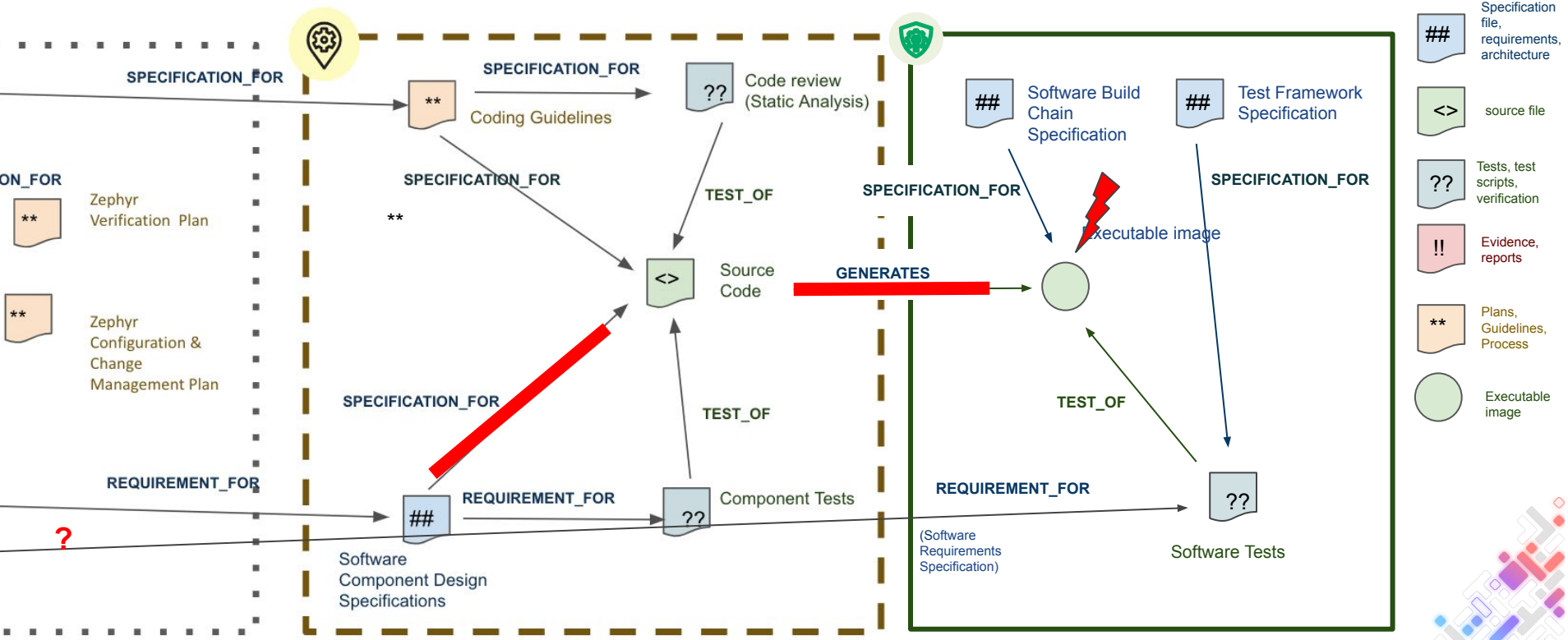
Dependency Identification on Component Level



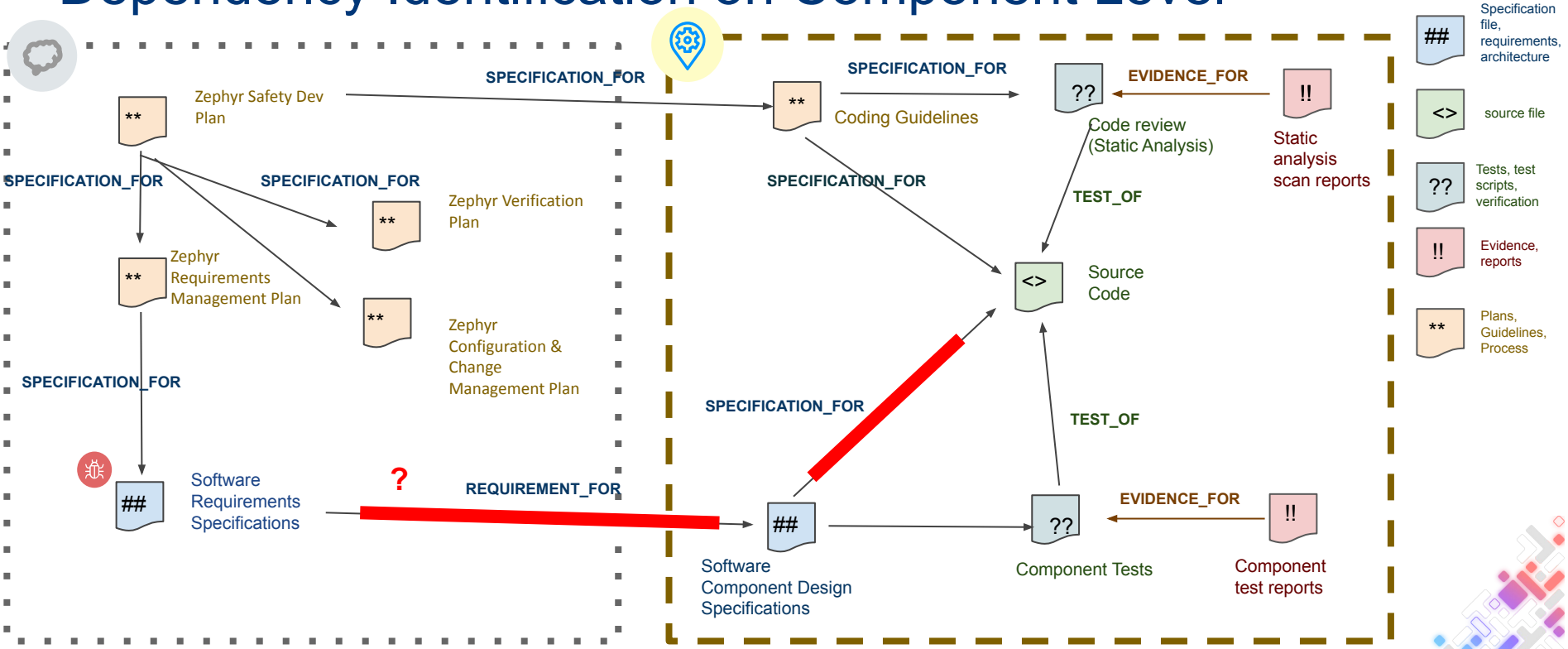
Dependency Identification on Component Level



Dependency Identification on Component Level



Dependency Identification on Component Level



Dependency Identification on Component Level