ORT in a CMake / C++ environment

ORT Community Days 2024

Ummo Schwarting
Carl Zeiss GOM Metrology GmbH
Frank Viernau
EPAM Systems GmbH

The Project
Why creating a SBOM is a challenge

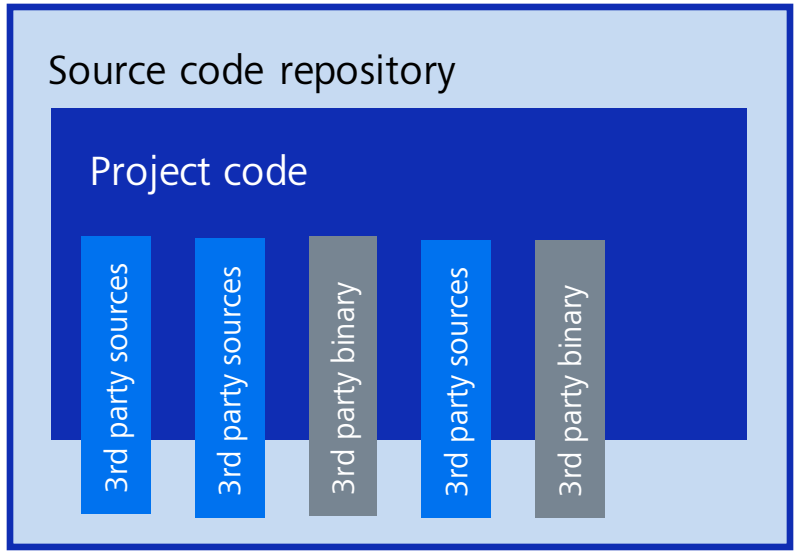
The Project environment

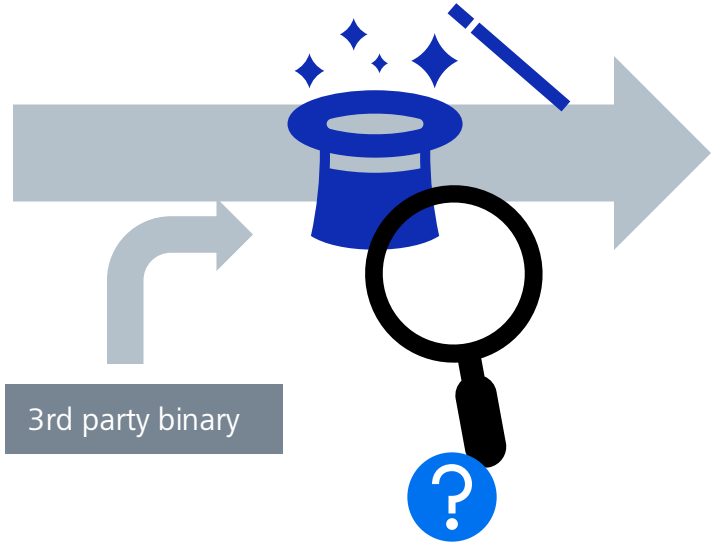
- Metrology 3D Software
- Legacy monolithic CMake / C++ project with ~25 years development, ~ 10.000.000 lines of code
- Multiple teams committing, up to 90 pull requests a day
- SVN background: Migrated to git recently but still one repository without any submodules
- Many custom solutions to configure the build process
- No single build step but several layers
- 3rd party package included at many different steps
- Firmware builds attached
- On site build environment

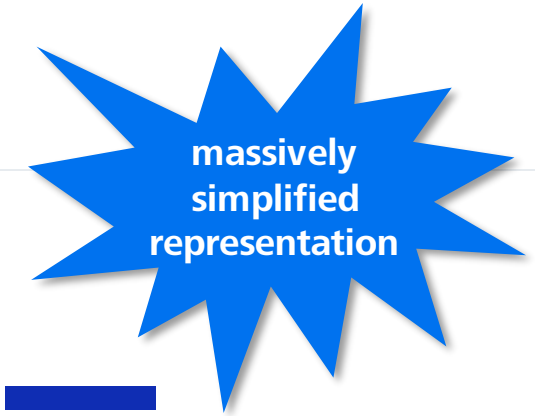
Further Challenge

- No changing a running system:
  setting up everything new and clean is no option.

Screening a monolithic legacy project







Options for Software composition analysis (SCA):

- Scan the whole source repo?
  → Infinite work for curation (plus maintaining curations)
  → Binary artefacts might be missed
- Binary analysis on final artifact?
  → massively incomplete.

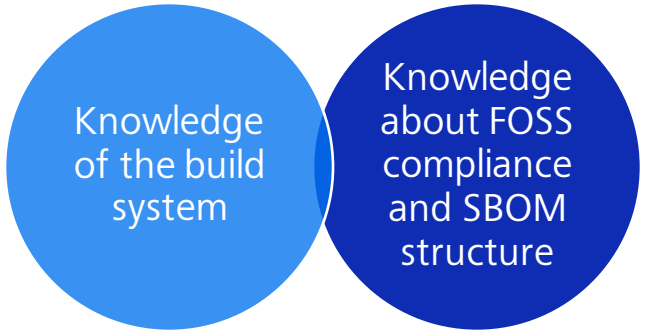
What to do?

1. **Clean up!**

- Migrate 3rd party packages into a central collection
- Sort between distributed and internal packages

2. **Add SCA 'abstraction layer' to funnel custom build solution into ORT workflow**

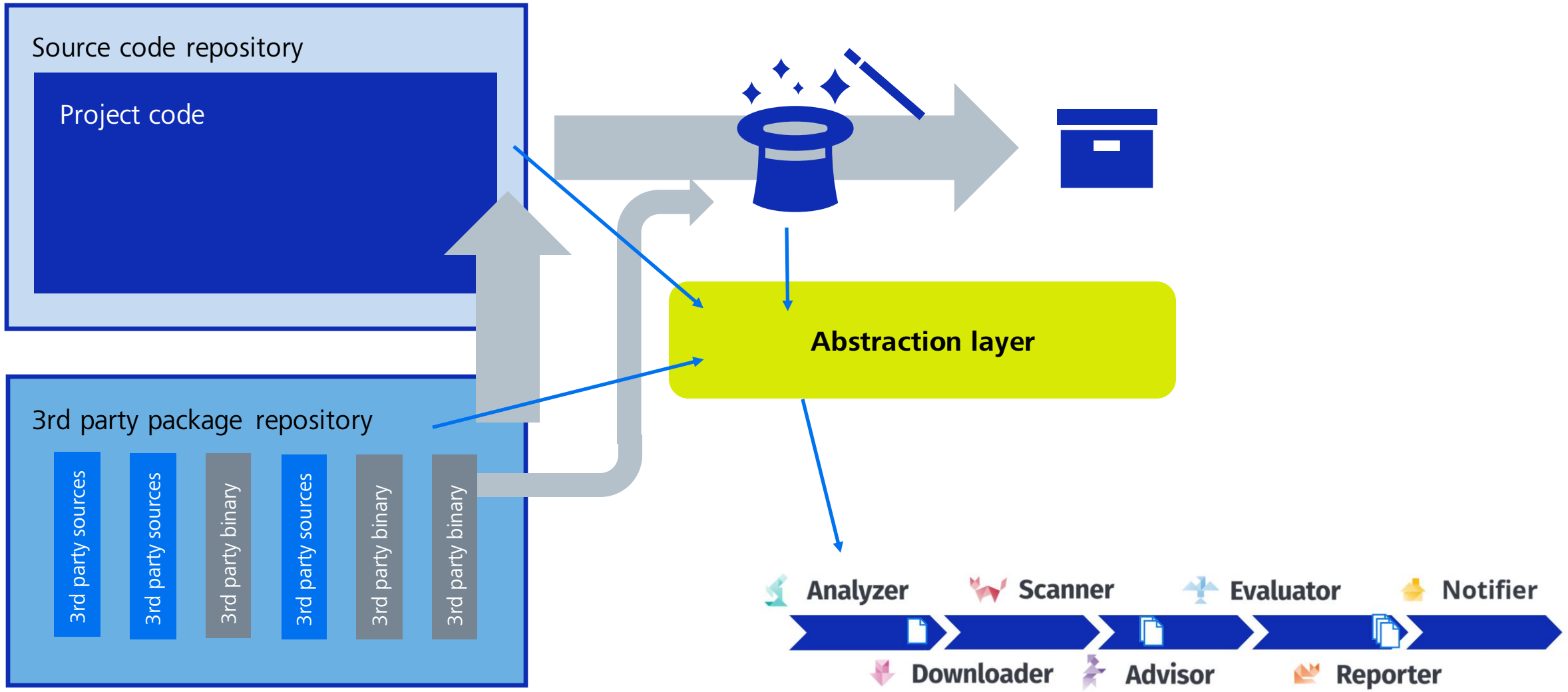
- The existing build system "magic" remains in place, but abstraction allows further processing.
- The target data structure for the abstraction must be simple
  - Avoid having to train the build system specialists for SBOM structure



3. **Prioritize: Focus on what is important and define increments**

- Completeness over structure
  - Package hierarchy is no priority → ORT won't know, how packages depend on each other.
  - But every package will be listed
- Binary package analysis is currently out of scope

Screening a monolithic legacy project



Motivation
CMAKE -> analyzer result

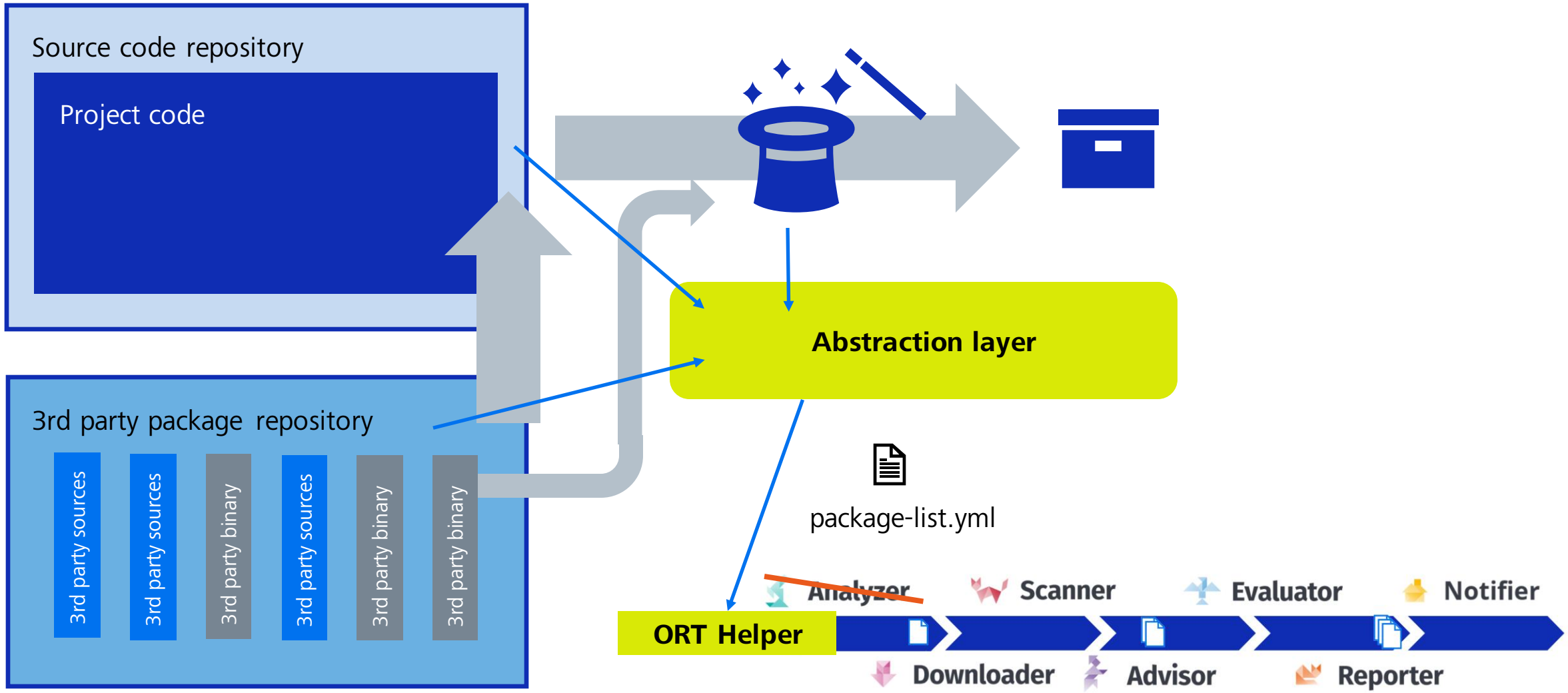
Requirements

- Enable ORT's package-by-package license scanning / clearance workflow (re-use work across projects)
- Use ORT to create SBOMs containing dedicated entries per dependency
- Future: Enable querying vulnerabilities for the dependencies via ORT's advisor(s)

Just need an analyzer result representing the CMAKE project, but where from?

- Implement CMAKE support as package manager in ORT? ..very hard, not timely, not feasible.
- Put information into project.spdx / package.spdx files and analyze them with ORT?
  - CMAKE scripts have all information, they could generate SPDX documents, but
  - SPDX document data model more complex than needed here
  - Relying on external format implies limitations: Risk to supporting unforeseen future use cases
  - ORT's SPDX analyzer is relatively hard to change: analyzer result and SPDX documents do not match 1:1
- **Idea**: Introduce a minimal file format dedicated to this use case
  - Generate an analyzer result based on such file + ort configuration

Screening a monolithic legacy project



Motivation
CMAKE -> package list YAML -> analyzer result

Minimum data needed:

- A flat list of packages, per package:
  - identifier
  - provenance (for scanning for detected licenses)
  - is_excluded, is_dynamically_linked (for policy rules / license clearance)
- Some data from ORT configuration which the analyzer adds: package curations, …

Decision to allow choosing identifiers freely (including type)

- Allows creating analyzer results with arbitrary set of packages: useful for other use case such as license clearance.
- User becomes responsible for uniqueness of identifiers

Short Demo

…

Limitations:

- Not all package metadata can be set yet. Further fields to be added, e.g. PURLs
- Inject further configuration into analyzer result: resolutions, .ort.yml file
- Querying vulnerabilities only work if identifier type is known in ORT / if queries are constructed from PURLs

Outlook
Ongoing and future goals

- Transmit package dependency tree into ORT
- Use additional metadata, e.g. package URL (PURL)
- Read completed SBOMs for binary packages into ORT pipeline (equivalent to Scanner step)
- Vulnerability identification for C++ dependencies (packages without ecosystem ID)

Discussion

Seeing beyond