Open Source based Software Composition Analysis at scale

Marcel Kurzmann, Robert Bosch GmbH
FOSDEM 2024
Reference Tooling Work Group

We are building an open source compliance toolchain ecosystem with open source tools as an open source project. To accomplish this we:

- Use existing independent tooling projects
- Provide reference workflows to allow their adoption
- Provide the concepts and glue to ensure easy interoperability and integration in existing environments
- Provide reference turnkey toolchains that can be used without fees by anybody

Join Us in Creating a New Era for Open Source Compliance

Mailing List: oss-based-compliance-tooling@groups.io
Subscription page: https://groups.io/g/oss-based-compliance-tooling
Online meetings: Bi-weekly - Invitations are sent to the mailing list
Website: https://oss-compliance-tooling.org/
And of course we are on GitHub:
https://github.com/Open-Source-Compliance/Sharing-creates-value
Background
Background

Our journey – the beginning

Mission: Open Source Management automation for JAVA/Maven projects.

Target Fact Sheet (simplified) - JAVA/Maven

Environment Parameters

- Business context: Server-based applications, fat clients
- Distribution context: hosted/distributed
- Development context: explorative / deterministic
- Development Mode: Agile / classic using agile methods
- Build mode: CI/CD, Jenkins

Open Source Parameter

- Open Source Use: only permissive licenses
- Open Source snippets: forbidden
- OSM Concept: binary identification via hashes, hash matching
- Package identification: package manager
- Component paradigm: 1 component ⇔ 1 source
- Metadata Source: central (commercial) database

Mission completed?
Background

Our journey – orbit transfer

Orbit A

JAVA/Maven

Orbit B

JAVAScript/
NPM

Open Source
Management
Background

Our journey – the next mission

Open Source Management automation for JAVAscript/NPM projects.

Target Fact Sheet (simplified) - JAVAscript/NPM

Environment Parameters

- Business context: Web applications
- Distribution context: distributed
- Development context: explorative / deterministic
- Development Mode: Agile / classic using agile methods
- Build mode: CI/CD, Jenkins

Open Source Parameter

- Open Source Use: only permissive license
- Open Source snippets: forbidden
- OSM Concept: binary identification via hashes, hash matching => recursive dependency resolution
- Package identification: package manager
- Component paradigm: 1 component ⇔ 1 source => n:m; download sources and scan
- Metadata Source: central (commercial) database => local database with scan results and curations
Background

Our journey – transfer of learnings

JAVA/Maven

Open Source Management

ORBiT A

ORBiT B

JAVA_SCRIPT/ NPM

JAVA/Maven
Background

Our journey – utilizing the momentum

Open Source Management automation for Embedded systems.

Target Fact Sheet (simplified) – Embedded C / Conan

Environment Parameters
- Business context: Embedded Software for devices
- Distribution context: distributed
- Development context: deterministic
- Development Mode: scaled agile framework
- Build mode: regular incremental builds, Github action, limited scaling options => ORT-Server

Open Source Parameter
- Open Source Use: permissive licenses, weak copyleft licenses
- Open Source snippets: forbidden, use with exception
- OSM Concept: project.spdx.yml-files combined with snippet and license and copyright scanning
- Package identification: manually maintained project spdx.yml-files
- Component paradigm: 1 source ⇔ different binaries
- Metadata Source: source code

Source: https://nssdc.gsfc.nasa.gov/planetary/factsheet/saturnfact.html
Going back in time in: https://github.com/oss-review-toolkit/ort/
Background
„at scale“ – commuter flight path

Experience from „Onboarding“

▪ „Fact sheets“ helpful to quickly gain speed an putting everybody in the picture
  - For new team members
  - For the „customer“ development teams that needed support

▪ Mandatory concept documentation based on standardized template accelerated evolution
  - Initial documentation => reuse => iterative improvement => standardization => automation
  - Find reusable solutions faster by reducing search range with the help of „fact sheets“
Background
Our journey – the next stop

Open Source Management automation for Embedded IoT Linux systems.

Target Fact Sheet (simplified) – Embedded IoT LINUX

Environment Parameters

- Business context: Internet of things
- Distribution context: distributed
- Development context: deterministic
- Development Mode: classic using agile methods
- Build mode: development builds/release builds

Open Source Parameter

- Open Source Use: copyleft license
- Open Source snippets: forbidden
- OSM Concept: SBOM generated by build, component scanning or matching against database
- Package identification: purl, hashes, ...
- Component paradigm: source2binary-files, recipes, ...
- Metadata Source: collaboratively maintained public database; upstream first

Source:
https://nssdc.gsfc.nasa.gov/planetary/factsheet/jupiterfact.html
**Background**

**Goals and needs**

- Find match: Map your needs and
  - find existing solutions
  - find birds of a feather

- Share and reuse

- Standardizing while keeping flexibility

---

**Example: Finding clothes online**

1st limitation of search range

Women OR Men OR Kids

2nd limitation of search range

Clothing OR Shoes OR Sportswear OR ...

3rd limitation of search range

Jackets OR T-Shirts OR Pants OR ...

4th limitation of search range

Size? 
Determine parameters

Get overview of all clothes matching to your parameters

---

**Source:** https://commons.wikimedia.org/wiki/File:Body_measures_SVG.svg
Eclipse Apoapsis
Eclipse Apoapsis

New project proposal

apoapsis noun

 apo·apsis əˈpō-

plural apoapses or apoapsides "+

: the apsis that is farthest from the center of attraction: the high point in an orbit

Source: https://www.merriam-webster.com/dictionary/apoapsis

Source: https://www.dictionary.com/browse/apoapsis

- Apoapsis is a good opportunity, if you want to transfer to another object’s orbit.

- Details see

- https://projects.eclipse.org/proposals/eclipse-overlay
Eclipse Apoapsis
Overview and planned Outputs

Process Level

- **Document Collection**
  - Repo 1
  - Markdown, Powerpoint

- **Abstraction Layer concept ALSCA**

- **Generic Architecture**

- **Usage Case Collection**

- **Usage Blueprints**

Technical Level

- **ORT-Server**
  - Repo 2
  - Kotlin

- **API**

- **Orchestrator**

- **Workers**

**Match your needs to available solutions and jump-start process definition**

**Build your own recursive Dependency Resolution Service**
Eclipse Apoapsis Dependencies

SPDX3.x Operation Profile

Capability Map

Process Level

Technical Level

OSS Review Toolkit

Source: https://oss-compliance-tooling.org/tooling/Landscape/Toolchain-description/
Process Level Outputs
Eclipse Apoapsis

Generic Architecture Description

=> Starting with Open Source License Compliance, further increments with security, export control, ...
Eclipse Apoapsis
Abstraction Layer Concept

Can be used holistically across all domains

Manager
... Embedded SmartSensor... ORT via Gitlab Pipelines
Product Owner
... MobileApp... Fossology with sw360
Development Team
... Cloud-Service... FOSSLight
Audit Team

Covering all kinds of products
Covering all kinds of OSM concepts

Keeps flexibility for the development teams to choose whatever OSM solution is suitable
Eclipse Apoapsis
Usage BluePrints

Open Compliance Reference Tooling
Range of application

- No single reference but depending on context e.g. heteroportunous vs. homogenous OSM setups

**Usage BluePrints**

- Eclipse Apoapsis
- Fact sheets

**Open Source Tooling Group**

- ORT via Gitlab Pipelines
- ORT via Jenkins
- FOSSology with sw360
- FOSSLight...
Eclipse Apoapsis
Use case collection

As a ... I want to ... so that ...

Stakeholder:
- Development Teams
- Compliance Manager
- Security Manager
- Quality Manager
- Audit Team
- ...

Also base for Test-Cases of the solutions => e.g. using Dummy repositories from OpenChain Automation Workgroup
Technical Level Outputs – ORT Server
Vision
ORT Server Goals

- API (REST)
- Scalable (cloud agnostic)
- Easy setup and integration
- Keep flexibility
- Web frontend => see Outlook
- Access management
- Inventory management
Vision

Setup

Usage Blueprints

DEV-Team

SCA Service

API*

ORT-Server

API*

Abstraction Layer concept ALSCA

JVM API**

JVM API**

JVM API**

JVM API**

JVM API**

ORT Analyzer

Tool X***

ORT Downloader

ORT Scanner

ORT Advisor

ORT Evaluator

ORT Reporter

*based on ALSCA

** based on ALSCA definition of mandatory data that need to be handed over

*** Good practice: Plugin for ORT tools

Good practice:

Plugin für ORT Tools

Abstraction Layer concept ALSCA

JVM API**

JVM API**

JVM API**

JVM API**

JVM API**
MVP

Project Hierarchy
- Organizations
  - Products
    - Repositories
- Access management
- User management
- User configuration
  - Credential management

REST API
- Manage project hierarchy
- Trigger runs
  - Flexible configuration
- Status updates
- Generate reports
- Query data

Components
- API
- Orchestrator
  - Manage jobs
  - Prevent duplicate work
- Workers (analyzer, scanner, ...)
  - Run individual tools
  - Separate Docker images

Integrations
- Kubernetes
- Github Action
- OpenAPI
Next steps
Next steps

- OpenChain Tooling Group meeting 7.2. with Martin Nonnenmacher
  - Meeting details see OpenChain Global Calendar: https://www.openchainproject.org/participate
- Preparation of Initial contribution
- Detailed presentation of ORT-server in ORT Community Days 6.-7.3.2024 Berlin

Outlook:
- Frontend
THANK YOU!

Join Us in Creating a New Era for Open Source Compliance

Mailing List: oss-based-compliance-tooling@groups.io

Subscription page: https://groups.io/g/oss-based-compliance-tooling

Online meetings: Bi-weekly – see OpenChain Global Calendar
https://www.openchainproject.org/participate

Website: https://oss-compliance-tooling.org /

And of course we are on GitHub:
https://github.com/Open-Source-Compliance/Sharing-creates-value