

**TOSHIBA**

# **Understanding and Managing the Dependency in SBOM with the New Feature of SW360**

Toshiba Corporation  
Kouki Hama  
2023.02.05

# Who is presenting?

Kouki Hama

Toshiba Corporation  
Software Engineering & Technology Center



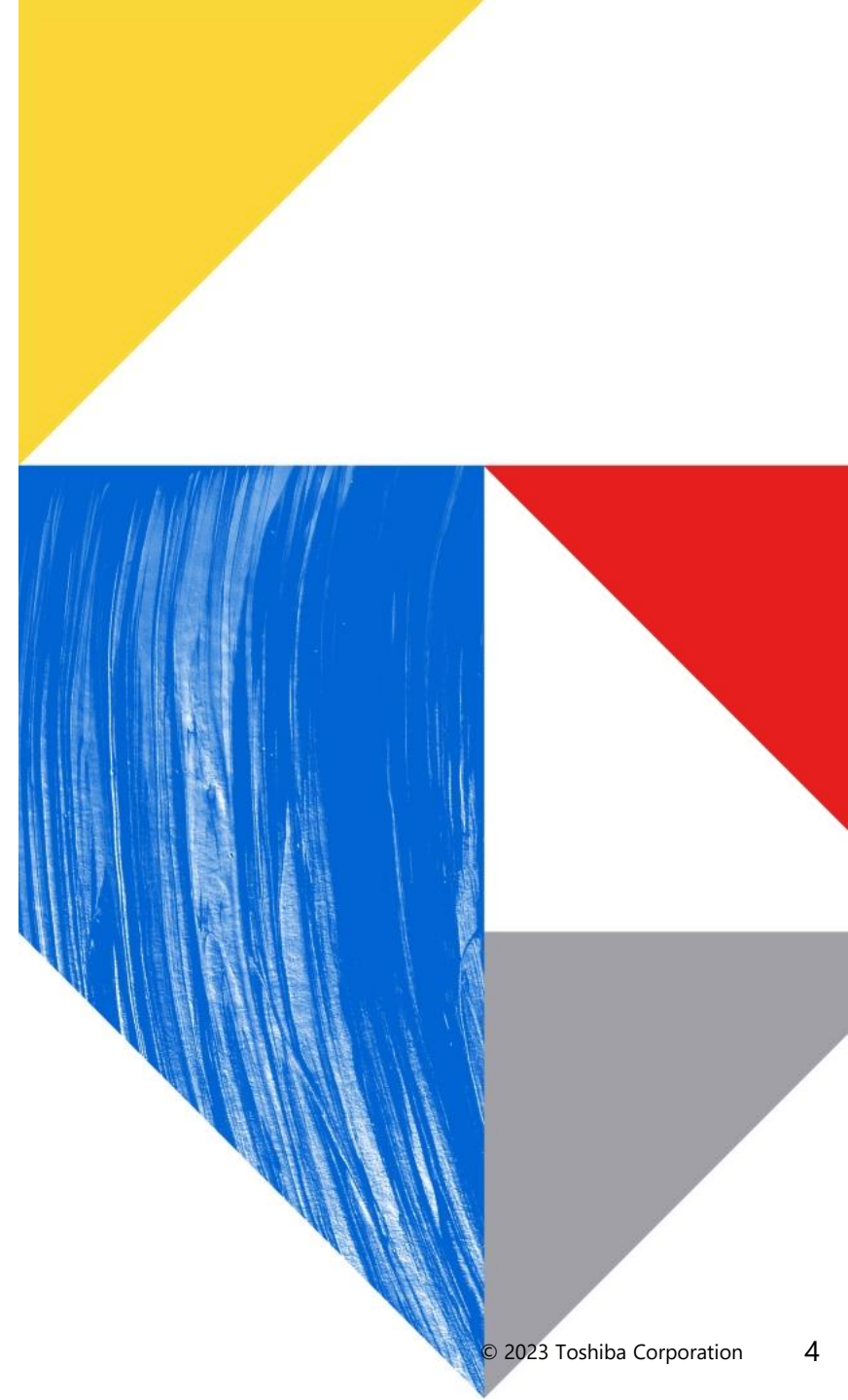
- Researches open source compliance, management process, and these tools.
- One of the co-leader of the Eclipse SW360 project.

# Contents

- 01 What is SW360?
- 02 Background; Software Dependency
- 03 Software Dependency registration issue in SW360
- 04 Solving Software Dependency Registration issue in SW360
- 05 SBOM standards format define dependency
- 06 Future Work for SBOM standards
- 07 Summary

# 01

What is SW360?



## What is SW360?

SW360 is an open-source software project licensed under the EPL-2.0 that provides both a web application and a repository to collect, organize and make available information about software components. It establishes a central hub for software components in an organization.

SW360 allows for

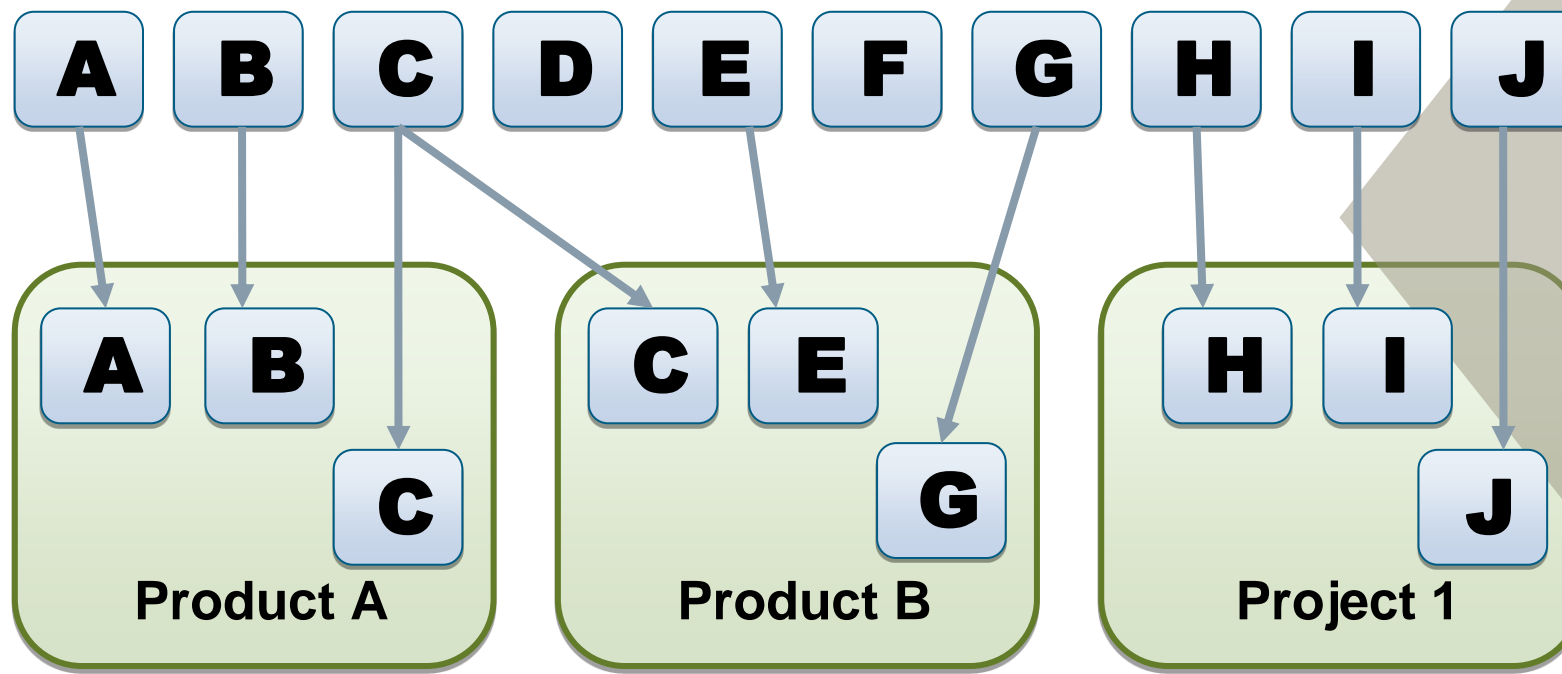
<https://www.eclipse.org/sw360/about/>

- tracking components used by a project/product,
- assessing security vulnerabilities,
- maintaining license obligations,
- enforcing policies, and
- generating legal documents.



SW360 is a 3<sup>rd</sup> party software component catalogue

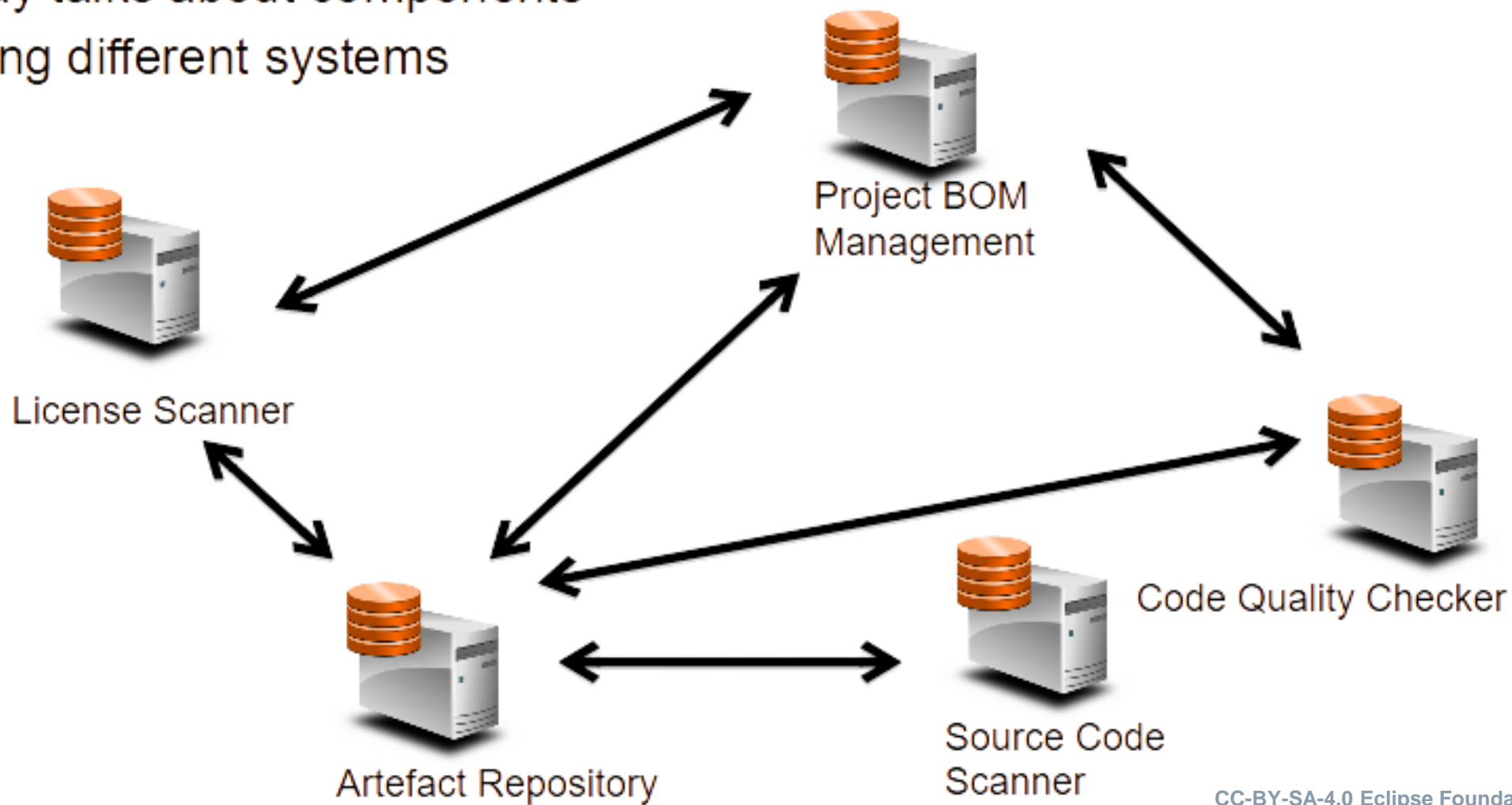
Assigns 3<sup>rd</sup> party and own components  
to products or projects



- **Goals and Benefits**
- Reuse information about components
- Coordinate product documentation process
- Supports OSS license clearing

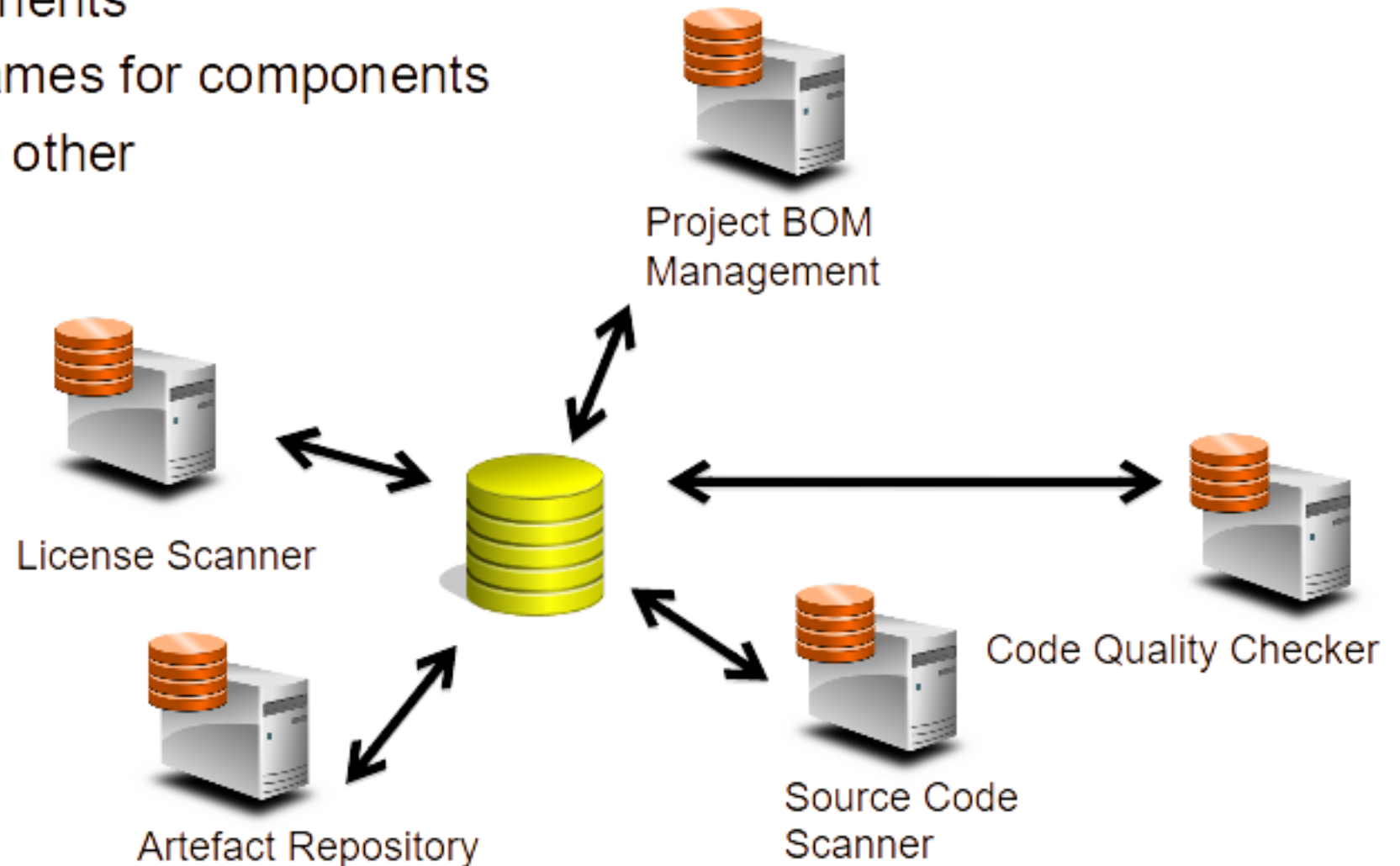
# Handling of Software Components

- IT today talks about components
- Involving different systems



# Solution: Centralise mappings

- “Phonebook for Components”
- Central database for names for components
- Systems to talk to each other
- Like person directory for IT systems in company already





# Example of SW360's registration Items for Software component

- Component Name
- Categories
- Component Type
- Languages
- Software Platforms
- Operating System
- Vendors
- Main Licenses
- Programming Languages
- Operating Systems
- CPE ID
- Software Platforms
- Release Date
- Download URL

The screenshot shows the 'Create Release' form in SW360. The form is titled 'Create Release' and has a 'Cancel' button. It is divided into several sections: 'Release Summary', 'Release Date', 'Clearing State', 'Created by', 'Licenses', 'Contributors', 'Download URL', 'Moderators', 'Programming Languages', 'Operating Systems', 'CPE ID', and 'Software Platforms'. Each section contains input fields or buttons for data entry. For example, 'Name' is set to 'Android', 'Version' is 'Enter Version', 'Release Date' is 'Enter Release Date', 'Clearing State' is 'New', 'Release Mainline State' is 'Open', and 'Created on' is '09/12/2019'.

## WIP: Support for SPDX (GUI, Import, Export... )

- <https://github.com/eclipse/sw360/pull/1682>
- <https://github.com/eclipse/sw360/pull/1503>

## Support for Cyclone DX is on Roadmap

- <https://github.com/eclipse/sw360/issues/1548#issuecomment-1146919177>

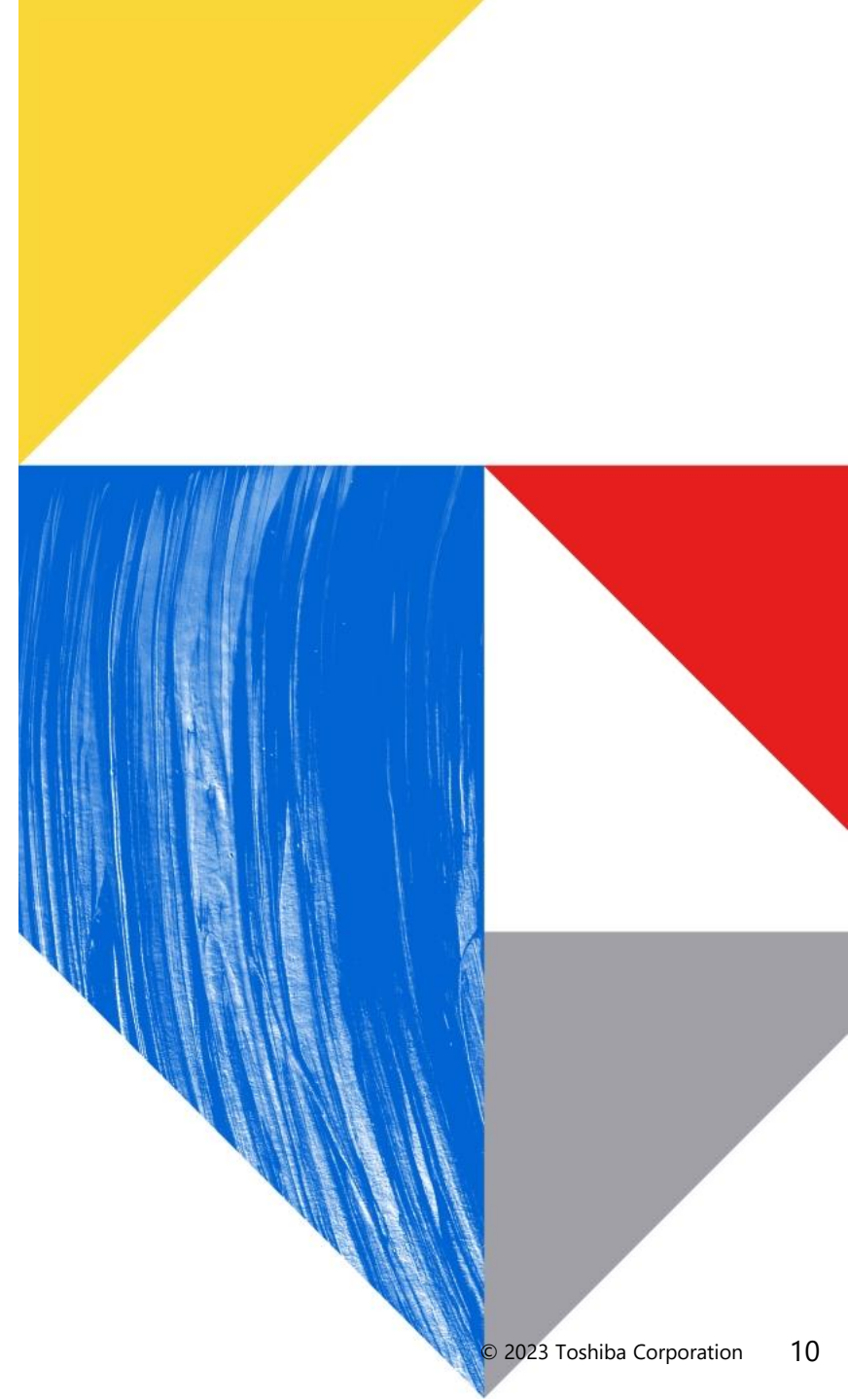
## 4 language GUI

- English, Vietnamese
- Japanese, **New: Chinese**
  - [fix\(language\): Fix the properties file and add some other needed files for Chinese language support by shi9qiu · Pull Request #1820 · eclipse/sw360 \(github.com\)](https://github.com/eclipse/sw360/pull/1820)

# 02

Background;

Software Dependency

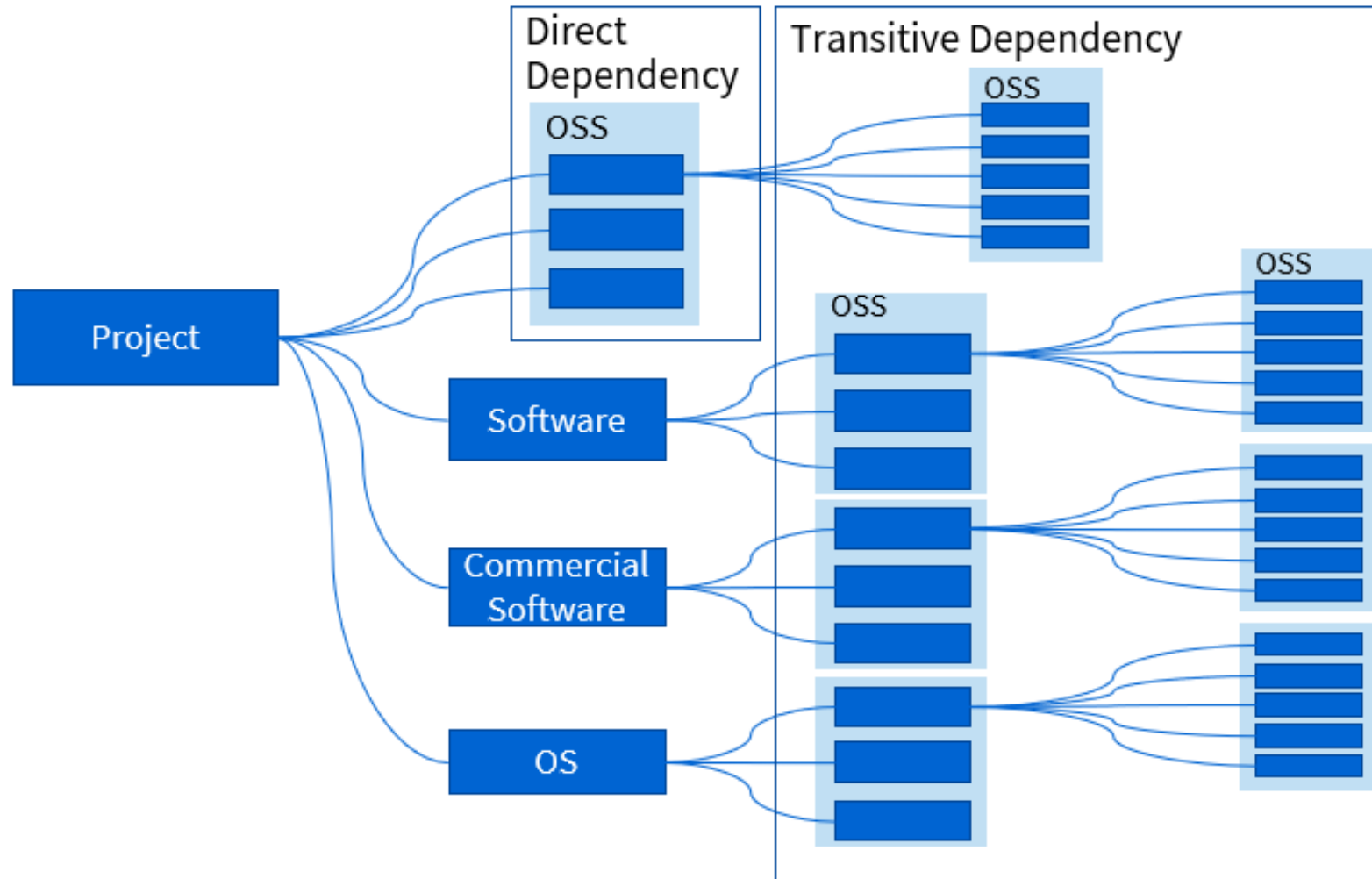


# In nutshell; Why Dependency management is important

- Software dependency information is huge and complex
- It needs to be properly managed to comply with license obligations and to manage vulnerabilities.

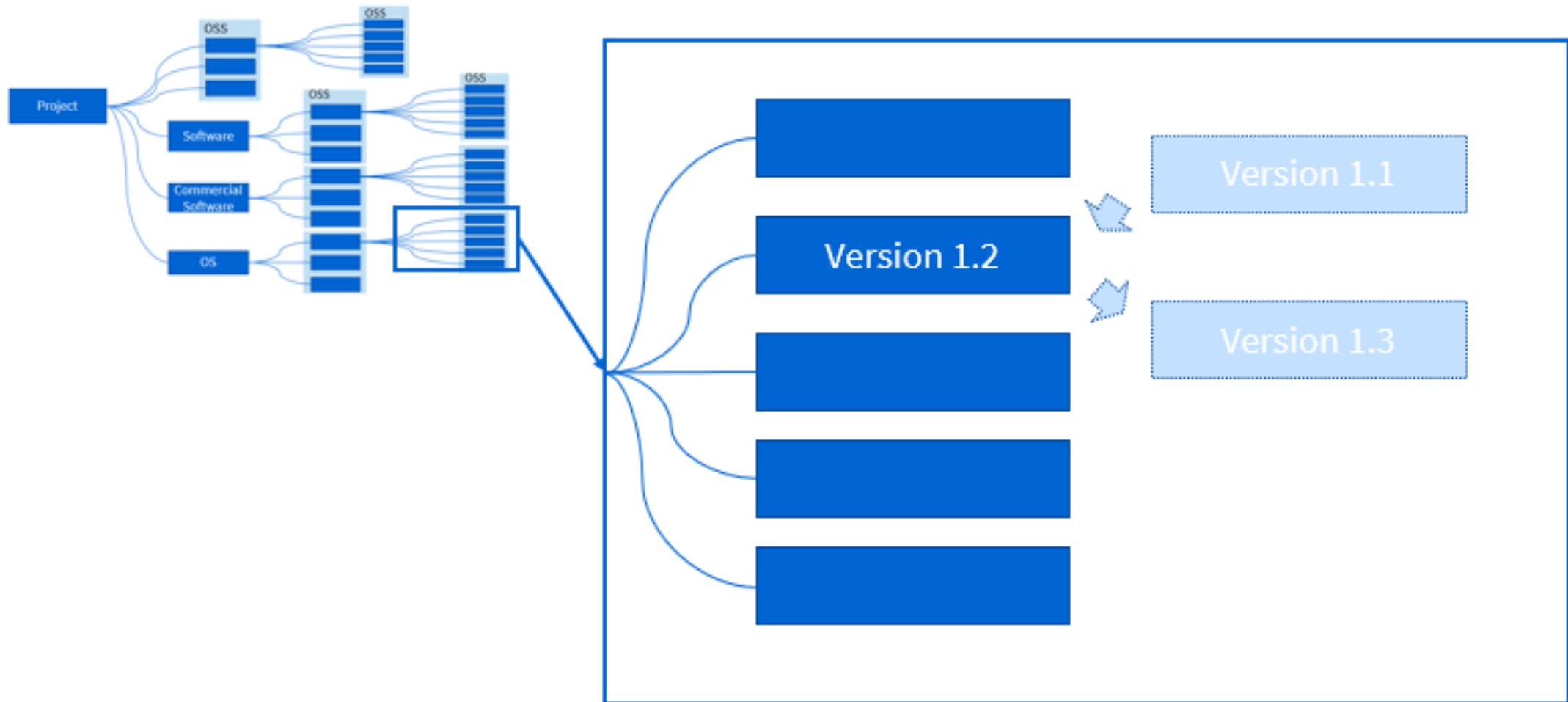
# Background About Software Dependency

The software dependency of a project refers to third-party open source software that this project depend on. Software dependencies can be direct or transitive.



# Softwares in Dependency update version

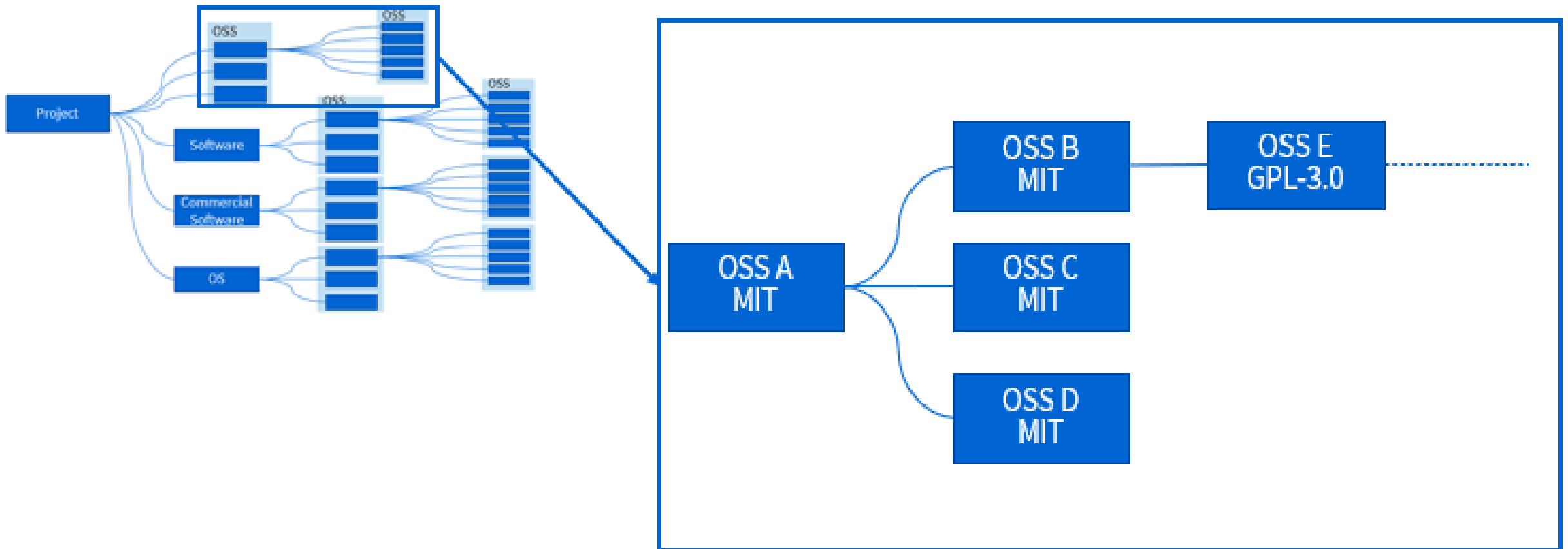
Dependency graph is continually changing because of version updates, the different build time and build options, etc.





# Need to check all licenses in dependency software

Because of the complicated dependency graph, the obligation of the open source license of a dependency may be ignored.



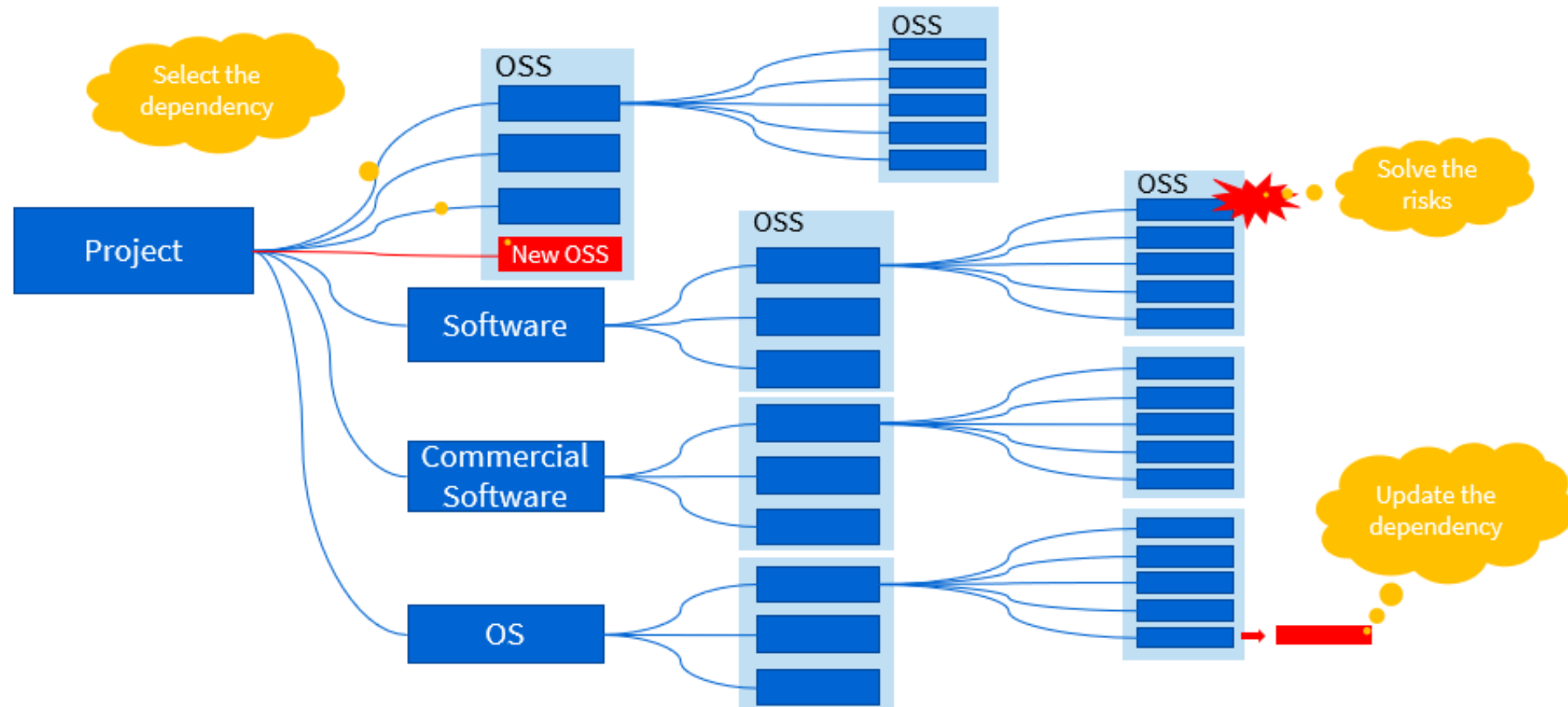




# Dependency Management

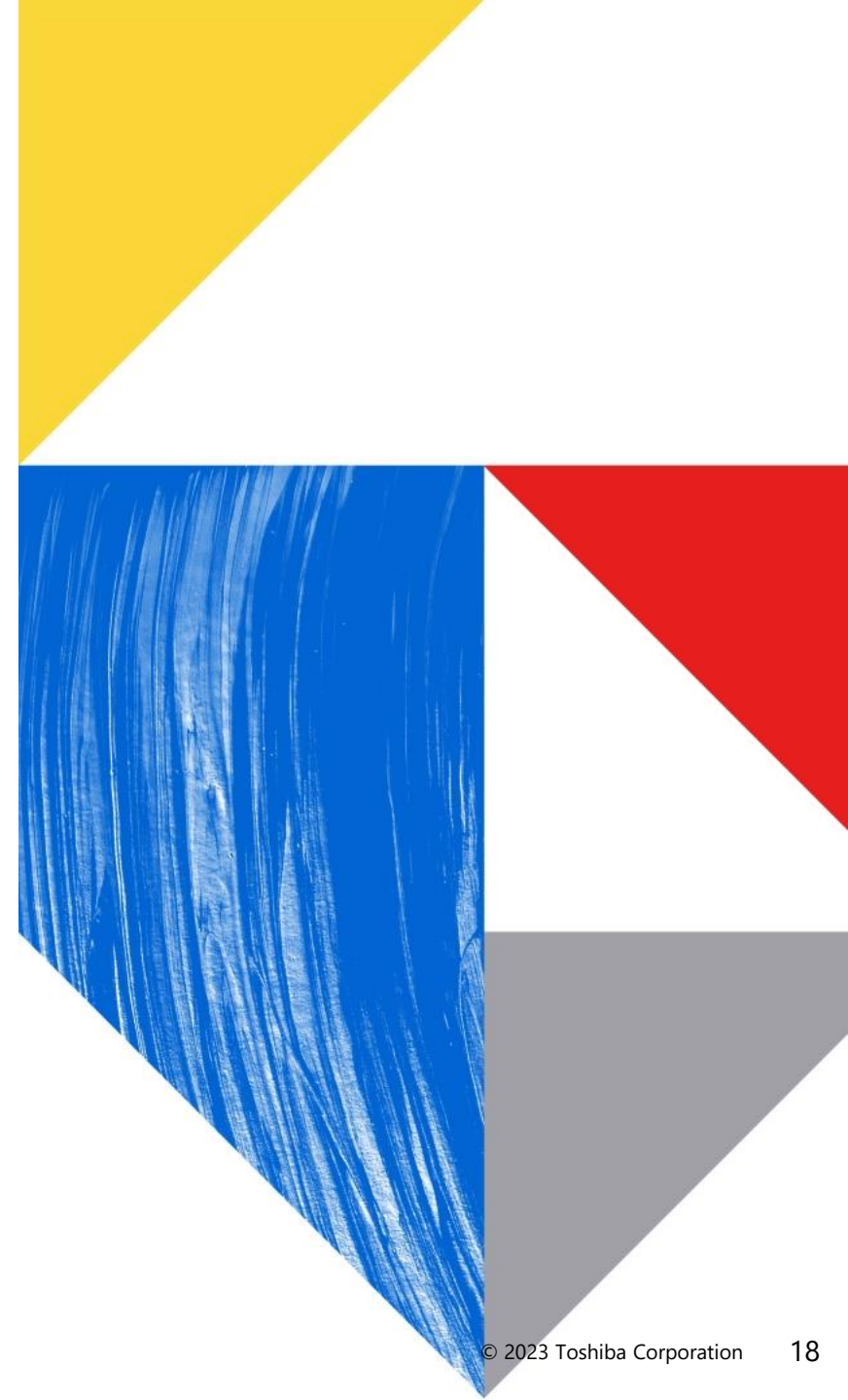
The management activities towards the dependencies of a project

- Selecting the proper dependency
- Updating the outdated dependency
- Solving the risks caused by dependencies



# 03

Software Dependency registration  
issue in SW360

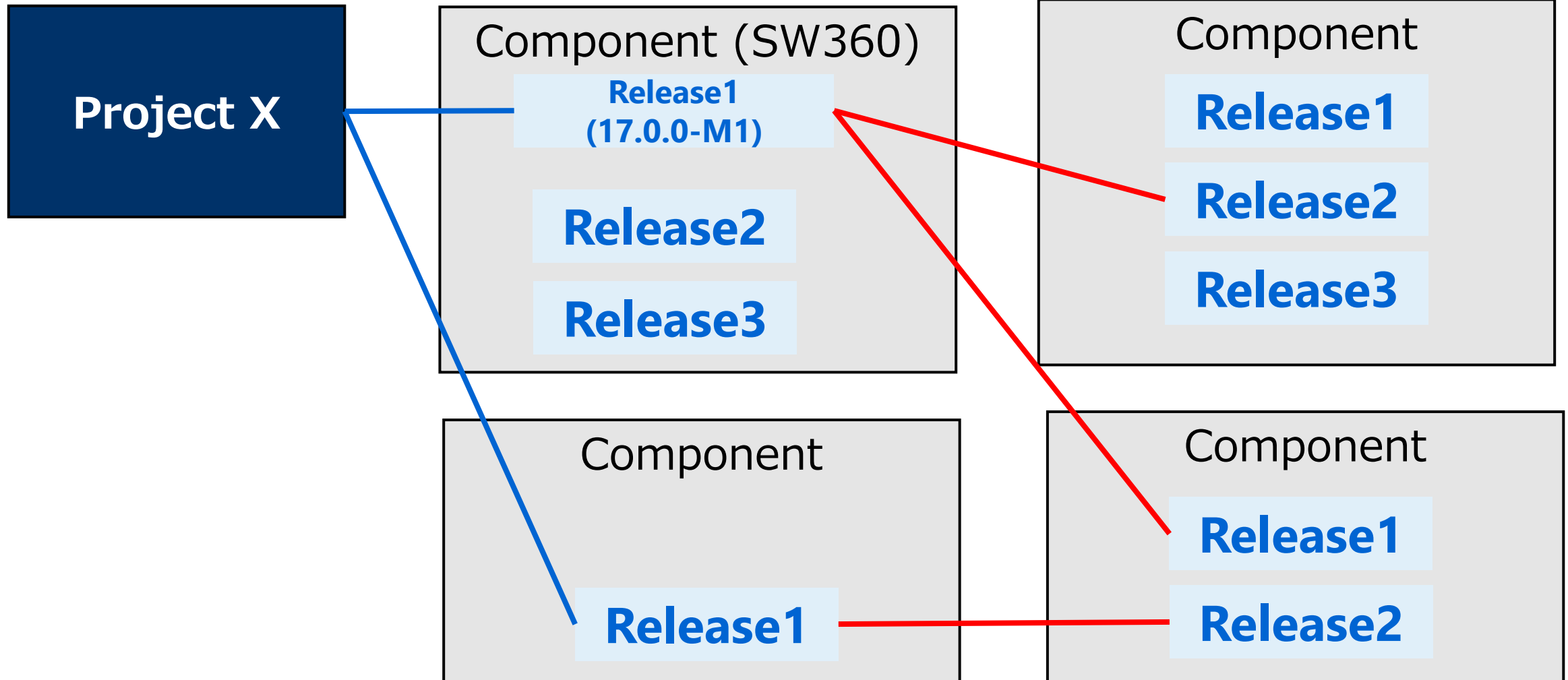


## The issues caused:

- SW360 can register only one software dependency information.
- Different dependencies cannot be registered for different projects.

# Data Architecture of SW360

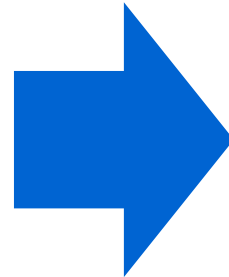
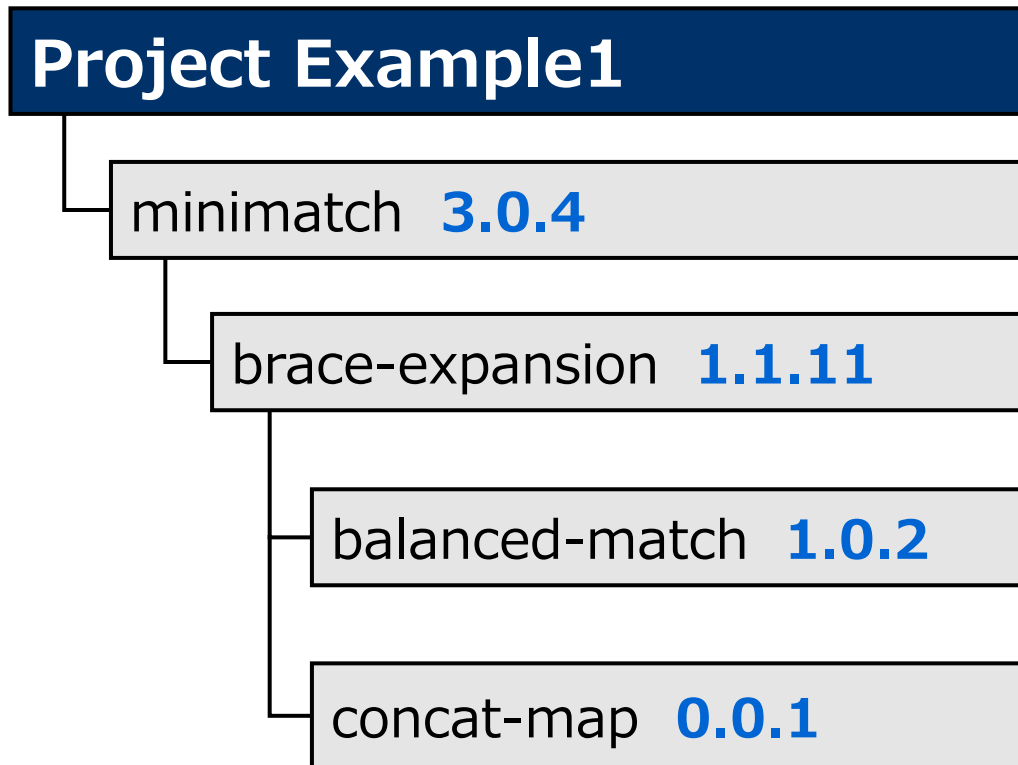
Ex) Component : SW360, Release ( $\hat{=}$  version) : sw360-17.0.0-M1



# How to manage dependencies in current SW360

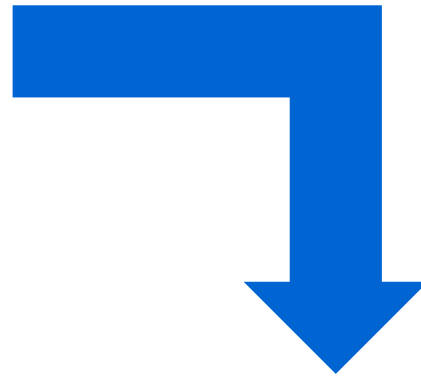
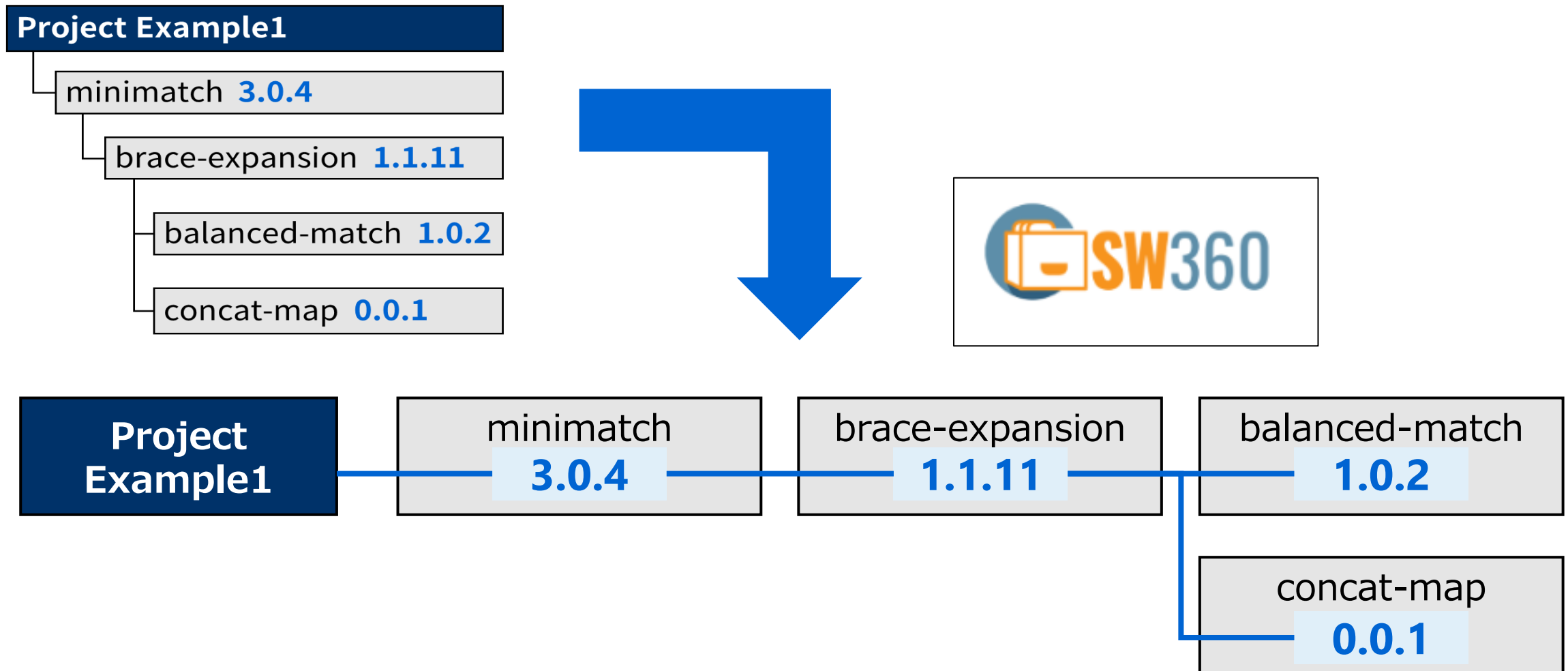
## How to manage **the dependencies of a project** in SW360

- Example: A project "Project Example1" depending on a npm package *minimatch 3.0.4*



# Dependencies registration on SW360 architecture

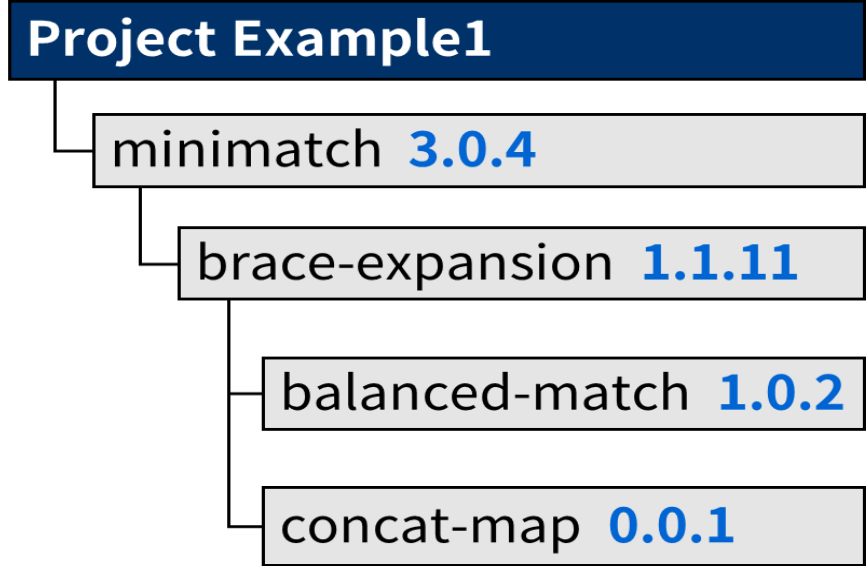
Example1 registered with SW360



# SW360's issue about dependencies registration

Example SW360 registration

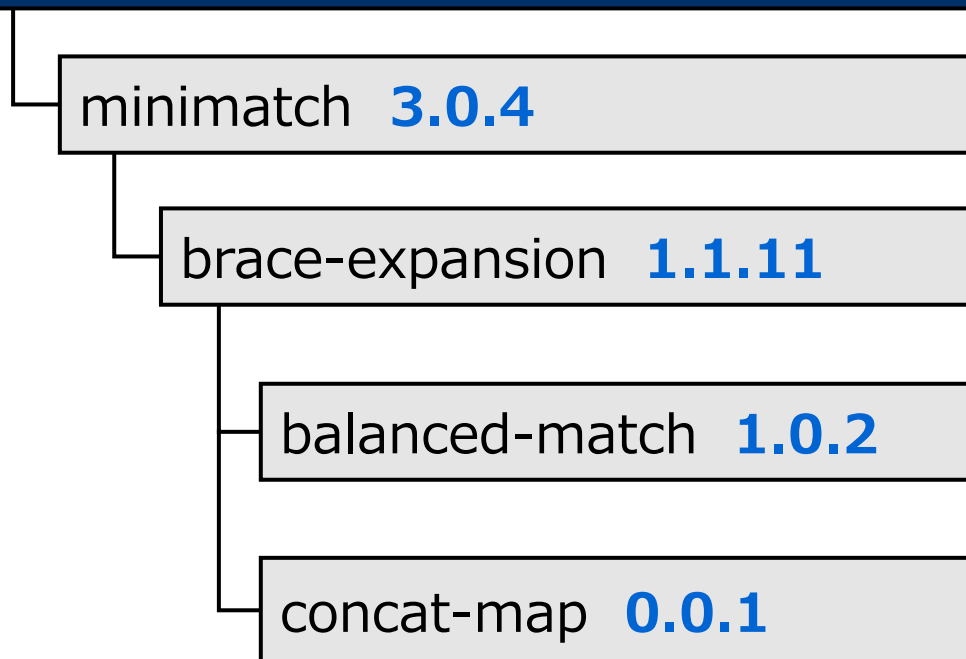
The screenshot shows the SW360 interface for 'Project Example1'. The left sidebar contains navigation tabs: Summary, Administration, License Clearing (highlighted), Obligations (0/0), ECC, Attachments, Attachment Usages, and Vulnerabilities (?/?). The main content area includes buttons for 'Edit Project', 'Link to Projects', 'Tree View', 'List View', and 'Export Spr'. Below these is a section for 'Add License Info to Release' and a list of 'Linked Releases' with 'Expand all' and 'Collapse all' options. The 'Linked Releases' list contains four entries: 'minimatch 3.0.4', 'brace-expansion 1.1.11', 'balanced-match 1.0.2', and 'concat-map 0.0.1'. A blue box highlights this list, and a callout points to it with the text 'Registered Software Information'. Another callout points to the 'Project Example1' breadcrumb with the text 'Project Example1'.



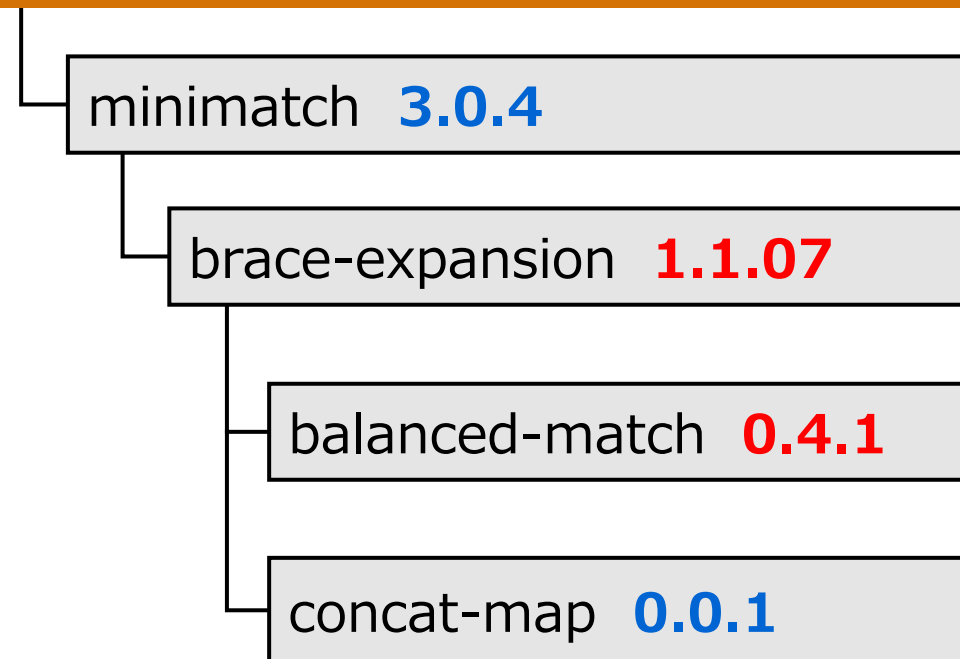
# Use case : Register 2 projects dependencies

Use case : two products using "minimatch 3.0.4"

## Project Example1



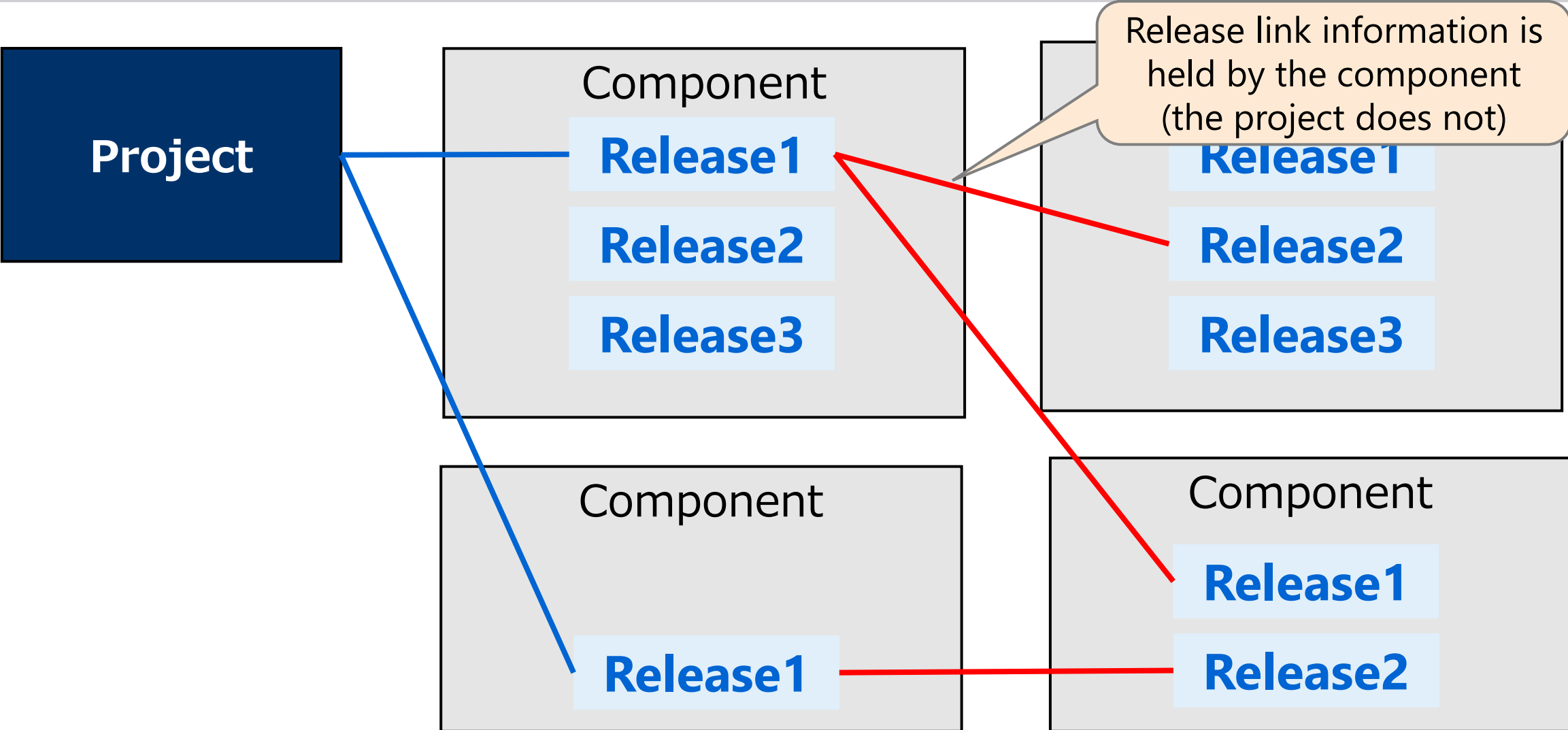
## Project Example2



**The same "minimatch 3.0.4" is used,  
but the dependent OSS versions are different.**



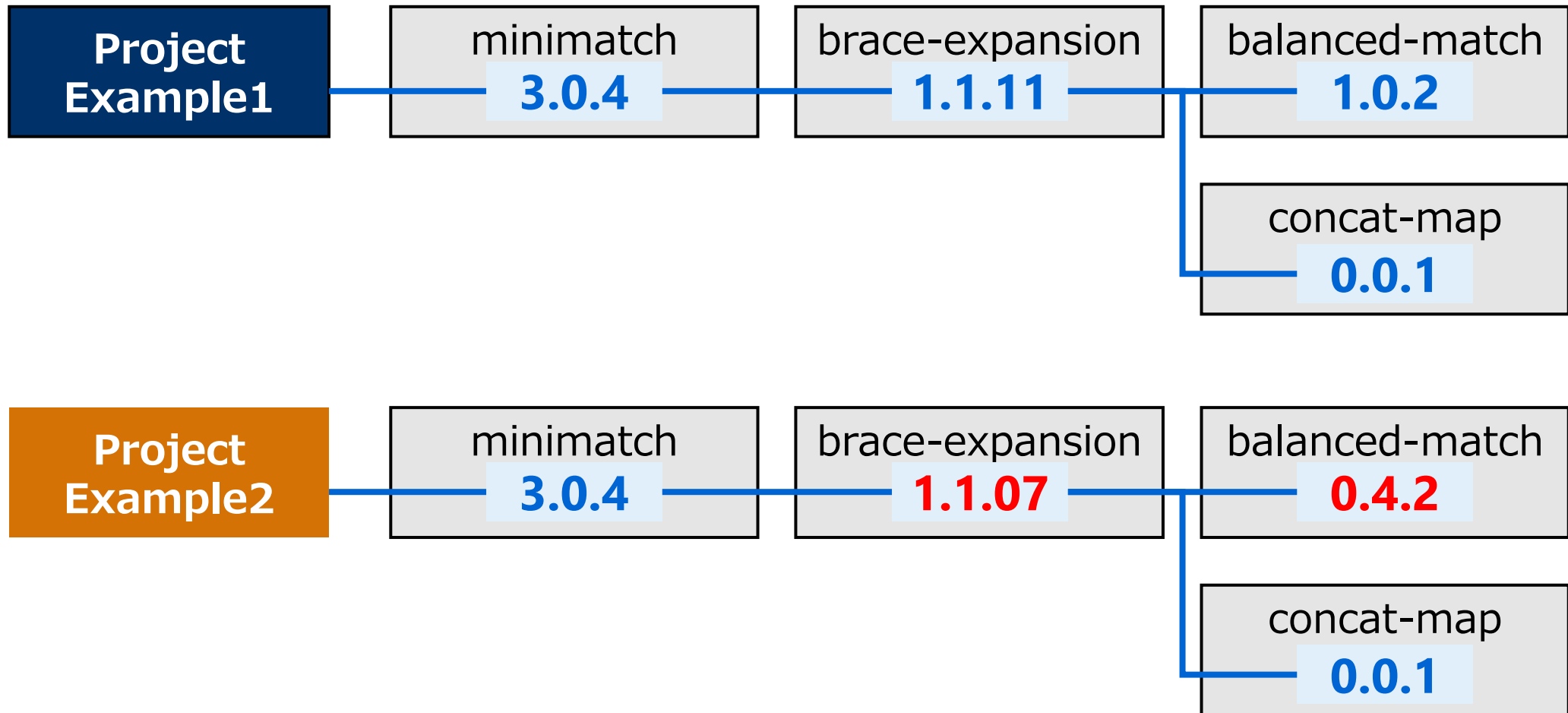
# Why Data Architecture of SW360 is bad for dependencies



**Cannot register different dependency information for each project**

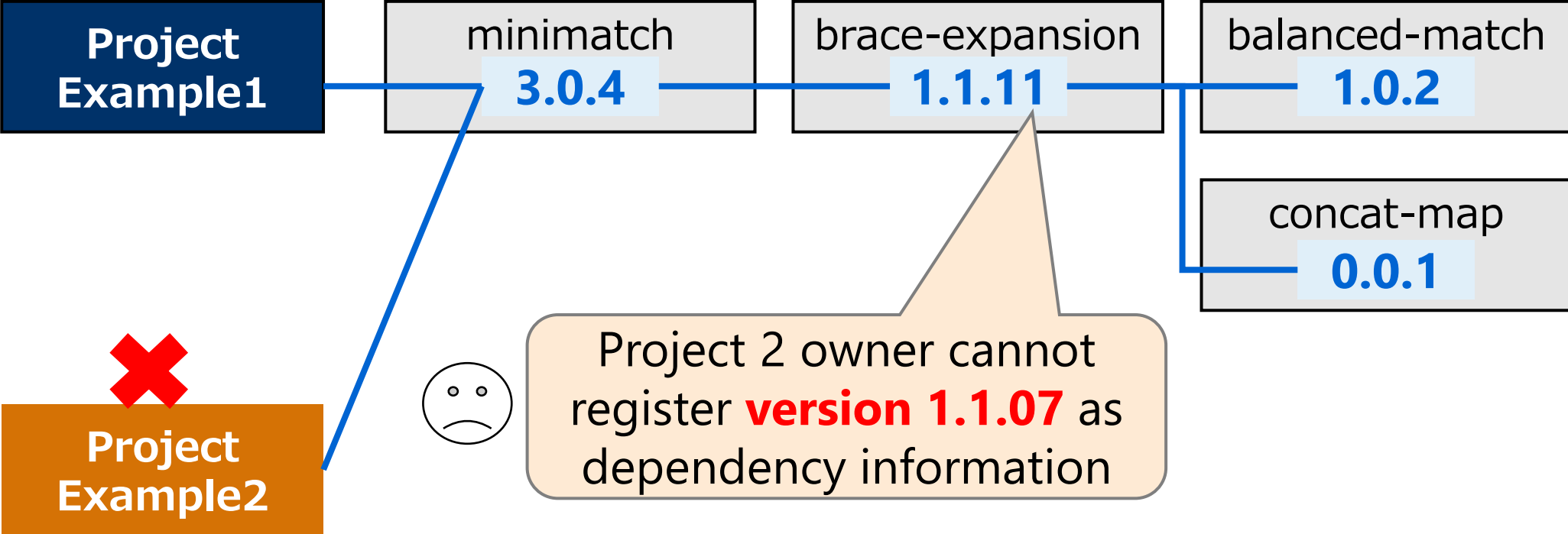
# Issue about dependencies registration of 2 projects

What we really need to do: is have a different tree structure for each project.



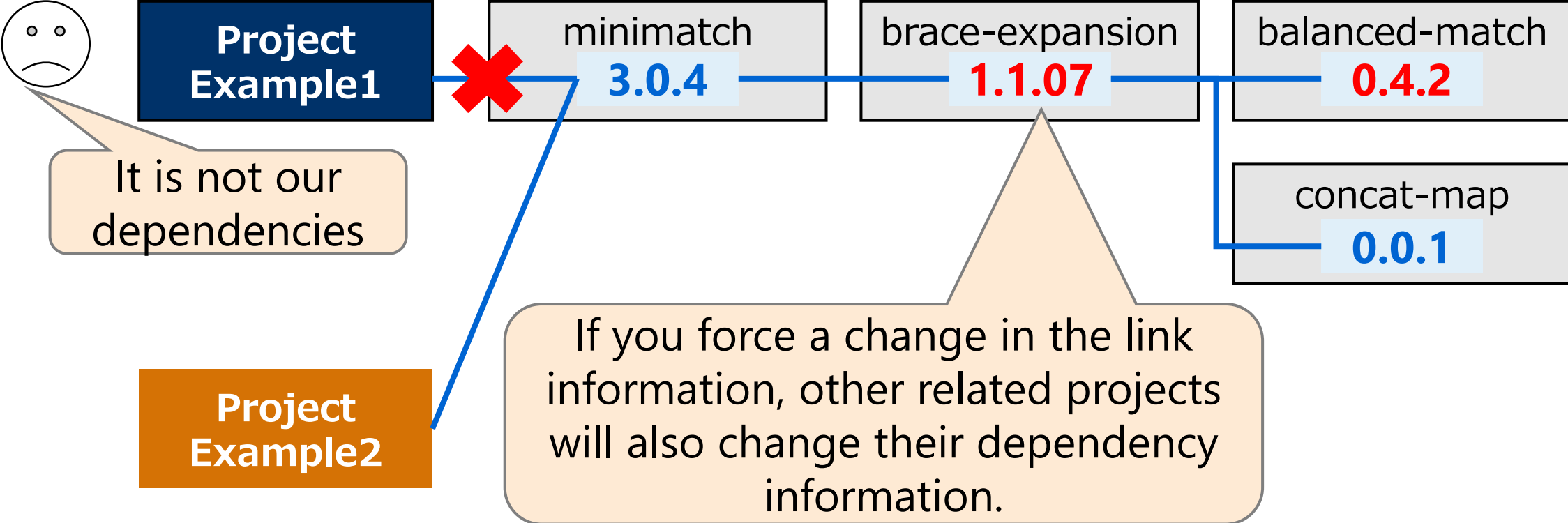
# Issue about dependencies registration for new project

Manage the same component dependency information in different projects.



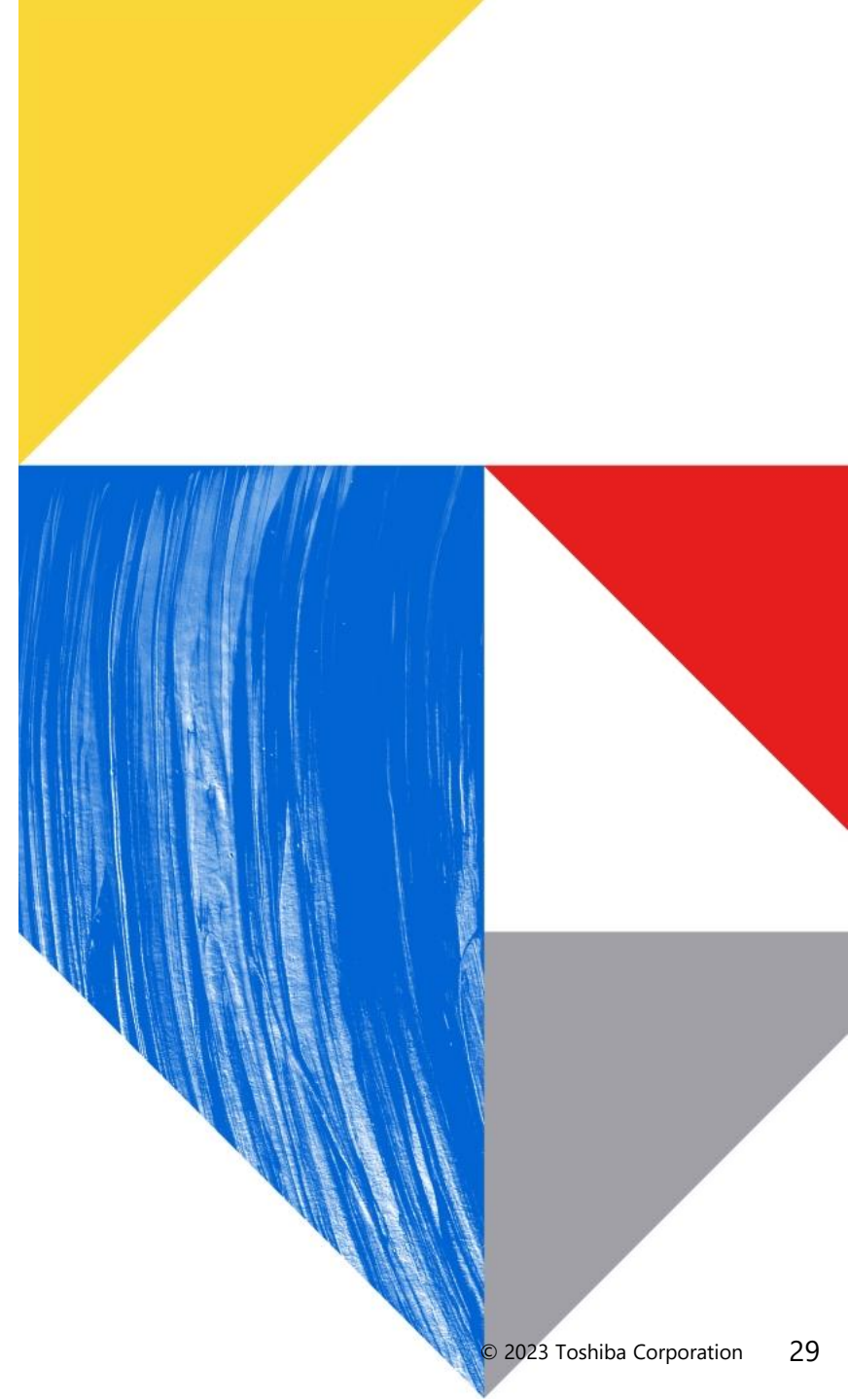
# Issue about dependencies registration for existing Project

Manage the same component dependency information in different projects.



# 04

## Solving Software Dependency Registration issue in SW360



# How to solve Software Dependency registration Problems in SW360

Being able to register different dependency information for each project

By Updating “**data architecture**” and “**GUI**”

- The code can be found in the SW360 branch managed by Toshiba  
[https://github.com/toshiba/sw360/tree/release/feature\\_dependency-network-management](https://github.com/toshiba/sw360/tree/release/feature_dependency-network-management)
- PR will be conducted soon.

# Change the data architecture for dependencies registration in SW360

A new feature for dependency management in SW360

- Feature allowing project to set up its own dependency network

Old Specification

Project A

Comp A  
Release1.0

Comp B  
Release1.0

Comp C  
Release1.0

Comp D  
Release1.0

When link a Release to project,  
Release link info is common, it  
can't be set for each project

New Specification

Project A

Project B

Comp A  
Release1.0

Comp B  
Release1.0

Comp C  
Release1.0

Comp D  
Release1.0

Comp A  
Release1.0

Comp B  
Release1.0

Comp D  
Release1.0

Comp C  
Release1.0

Projects have independent  
Dependency Network

# Change the GUI for dependencies registration in SW360

A new feature for dependency management in SW360

- Feature allowing project to set up its own dependency network

The image compares two versions of the SW360 GUI for dependency management. The left side shows the 'Old Specification' and the right side shows the 'New Specification'. An arrow points from the old to the new version.

**Old Specification:**

- Buttons: Update Project, Delete Project, Cancel
- Section: LINKED PROJECTS
- Table header: Project name
- Table content: minimatch
- Button: Add Projects
- Section: Release name, Release version
- Table content: minimatch, 3.0.4
- Button: Add Releases
- Callout: Unable to set link info for Release

**New Specification:**

- Buttons: Update Project, Delete Project, Cancel
- Section: LINKED PROJECTS
- Table header: Project name, Project Version
- Button: Add Projects
- Section: Release name, Release version
- Table content: minimatch, 3.0.4, brace-expansion, 1.1.11, balanced-match, 1.0.2, concat-map, 0.0.1
- Callout: Can set link info for Release



# New GUI for dependencies : Registration dependencies page

The functions of this new feature

- The Linked Releases And Projects tag (Edit page)

The screenshot shows the 'Linked Releases And Projects' page for 'PROJECT EXAMPLE1 (1.1)'. The page includes a sidebar with navigation options: Summary, Administration, Linked Releases And Projects (selected), Attachments, and Obligations (0/0). At the top, there are buttons for 'Update Project', 'Delete Project', and 'Cancel'. Below this is a section for 'LINKED PROJECTS' with a table header: Project name, Project Version, Project Relation, and Enable SVM. An 'Add Projects' button is located below the table. The main section is 'LINKED RELEASES', which contains a table with the following columns: Release name, Release version, Reload Info, Release relation, Project Mainline State, and Comments. The table lists three dependencies: 'minimatch' (version 3.0.4), 'brace-expansion' (version 1.1.11), and 'balanced-match' (version 1.0.2). Each row has a '+' button in the 'Release name' column, a dropdown arrow in the 'Release version' column, a circular refresh icon in the 'Reload Info' column, a dropdown arrow in the 'Release relation' column, a dropdown arrow in the 'Project Mainline State' column, and an 'Enter Comment' button in the 'Comments' column. A trash can icon is located at the end of each row. Callouts highlight the following functions: 'Add dependency' function (pointing to the '+' button), 'Select version' function (pointing to the dropdown arrow in the 'Release version' column), 'Load default dependencies' function (pointing to the circular refresh icon), and 'Delete dependency' function (pointing to the trash can icon). An 'Add Releases' button is located at the bottom left of the table.

Release name	Release version	Reload Info	Release relation	Project Mainline State	Comments
minimatch	3.0.4	🔄	Contained	Open	Enter Comment
brace-expansion	1.1.11	🔄	Contained	Open	Enter Comment
balanced-match	1.0.2	🔄	Contained	Open	Enter Comment
		🔄	Contained	Open	Enter Comment

# New GUI for dependencies: Registration Component page

- The “release” page is not changed.
- The dependency information here will be seen as the place storing the “default” information. It will keep the same with the latest information in the ecosystem (maven, npm, etc.)

Components > minimatch > minimatch (3.0.4)

Summary

linked releases

Clearing details

ECC Details

Attachments

Update Release Delete Release Cancel

Vendor Name	Release name	Release version
<input type="text" value="Enter vendor"/>	<input type="text" value="brace-expansion"/>	<input type="text" value="1.1.11"/>

Other restricted items (0)

Click to add Releases

# New GUI for dependencies : View page

## GUI: the "License Clearing" tag (View page)

Projects > Project Example1 (1.1)

Summary

Administration

**License Clearing**

Obligations **0/0**

ECC

Attachments

Attachment Usages

Vulnerabilities **??**

Edit Project Link to Projects

Add License Info to Release

Name ▾ (Expand all | Collapse all)

▼ minimatch 3.0.4

▼ brace-expansion 1.1.11 ●

balanced-match 1.0.2

concat-map 0.0.1

Projects > Project Example2 (1.1)

Summary

Administration

**License Clearing**

Obligations **0/0**

ECC

Attachments

Attachment Usages

Vulnerabilities **??**

Edit Project Link to Projects

Add License Info to Release

Name ▾ (Expand all | Collapse all) Linked F

▼ minimatch 3.0.4

▼ brace-expansion 1.1.7 ●

balanced-match 0.4.1

concat-map 0.0.1

Each dependency graph can be committed.

# New GUI for dependencies : Edit page

## GUI: the "Linked Releases And Projects" tag (Edit page)

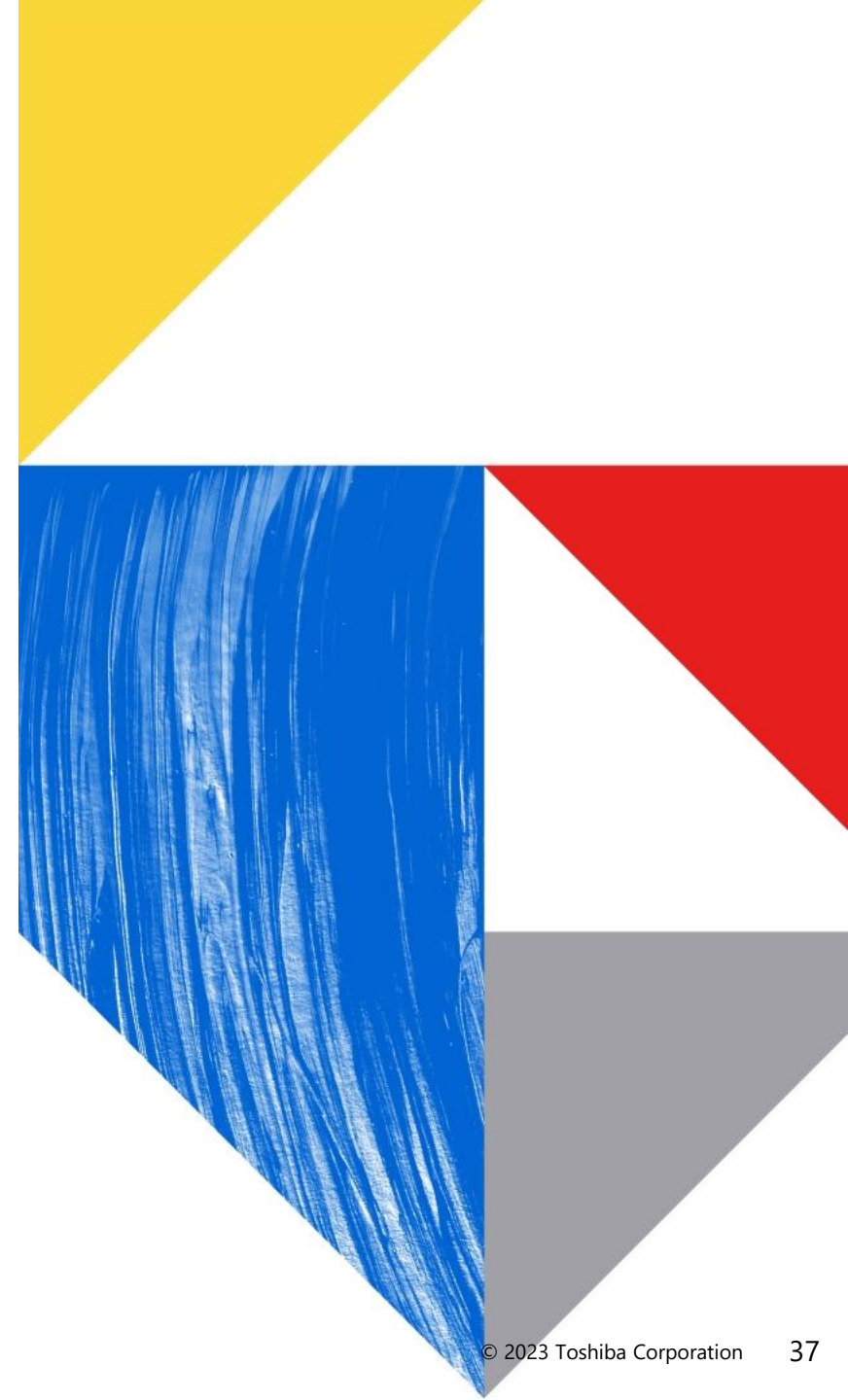
Each dependency graph can be committed.

Release name	Release version	Reload Info
minimatch	3.0.4	
brace-expansion	1.1.11	
balanced-match	1.0.2	
concat-map	0.0.1	

Release name	Release version	Reload Info
minimatch	3.0.4	
brace-expansion	1.1.7	
balanced-match	0.4.1	
concat-map	0.0.1	

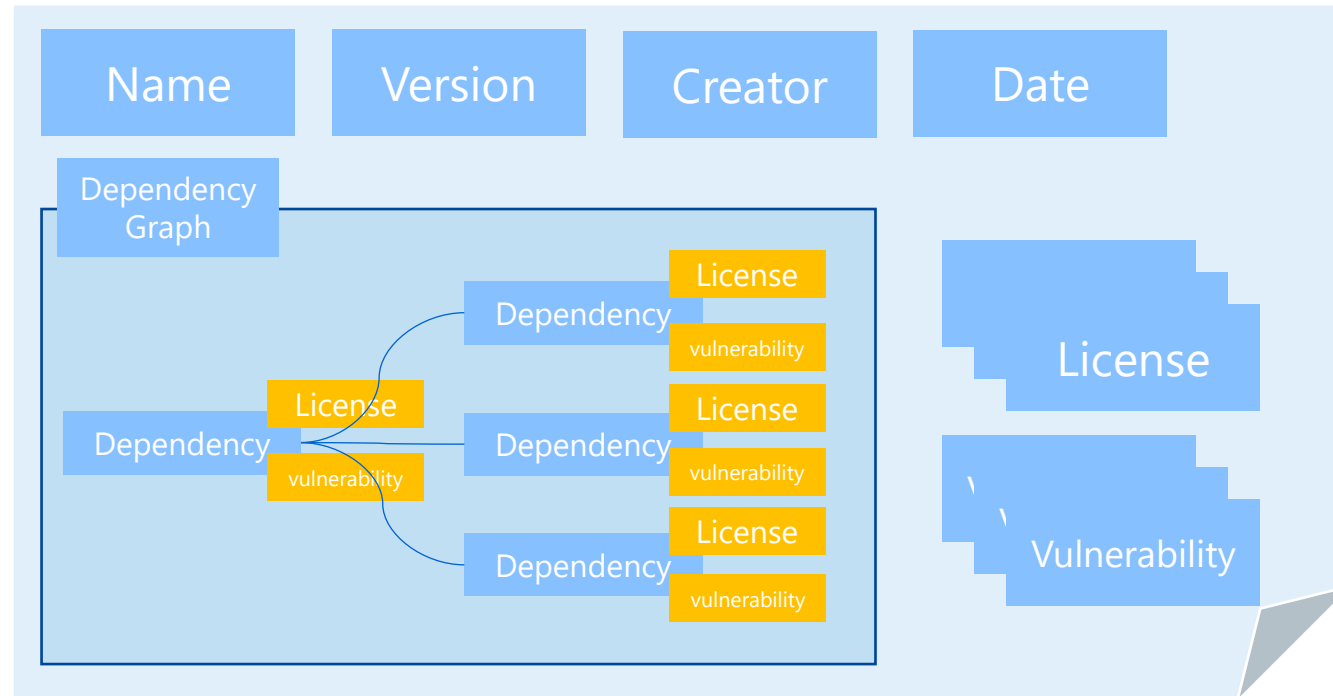
# 05

SBOM standards format define  
dependency

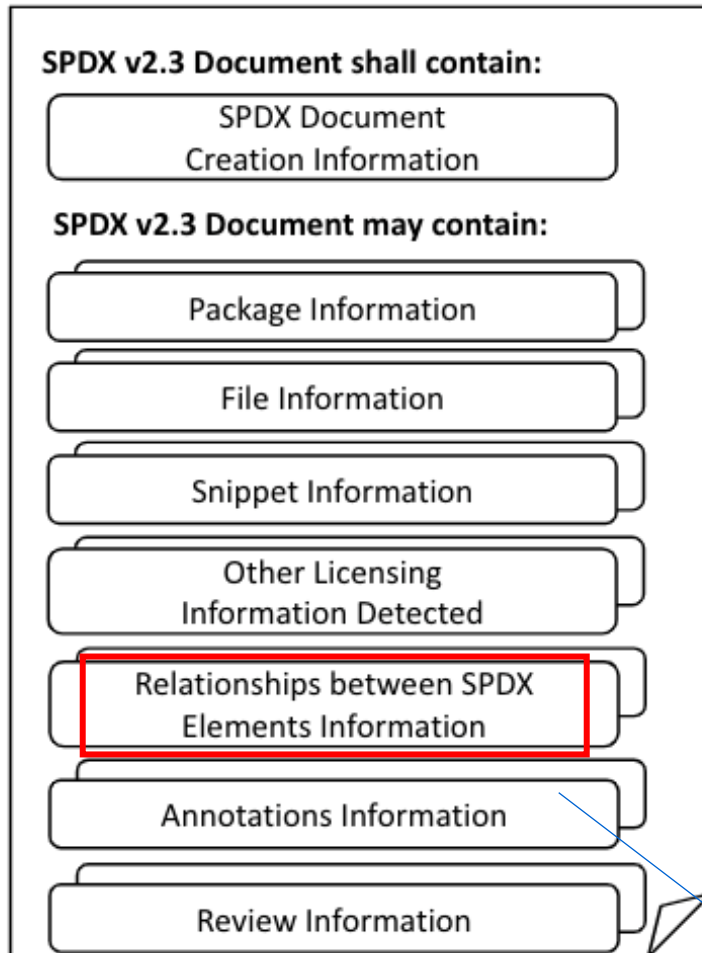


# How to use SBOM to manage dependency

- SBOM (Software Bill of Materials )
  - Formats: SPDX and CycloneDX are two widely used formats.
  - Both Formats can describe Software Dependencies



# Dependency in SPDX



## Example of SPDX elements Relationships

### Example: Between two Packages

DEPENDS_ON	Package A depends on the presence of package B in order to build and run
DEPENDENCY_OF	A is explicitly stated as a dependency of B in a machine-readable file. Use when a package manager does not define scopes.

### Example : Files relationship

DYNAMIC_LINK	An APPLICATION file 'myapp' dynamically links to BINARY file zlib.so.
STATIC_LINK	An APPLICATION file 'myapp' statically links to BINARY zlib.a.

Represent a relationship between two different Files, between a Package and a File

\* <https://spdx.github.io/spdx-spec/v2.3/relationships-between-SPDX-elements/>

# Dependency in Cyclone DX

CycloneDX provides the ability to describe components and their dependency on other components.

<https://cyclonedx.org/use-cases/#dependency-graph>

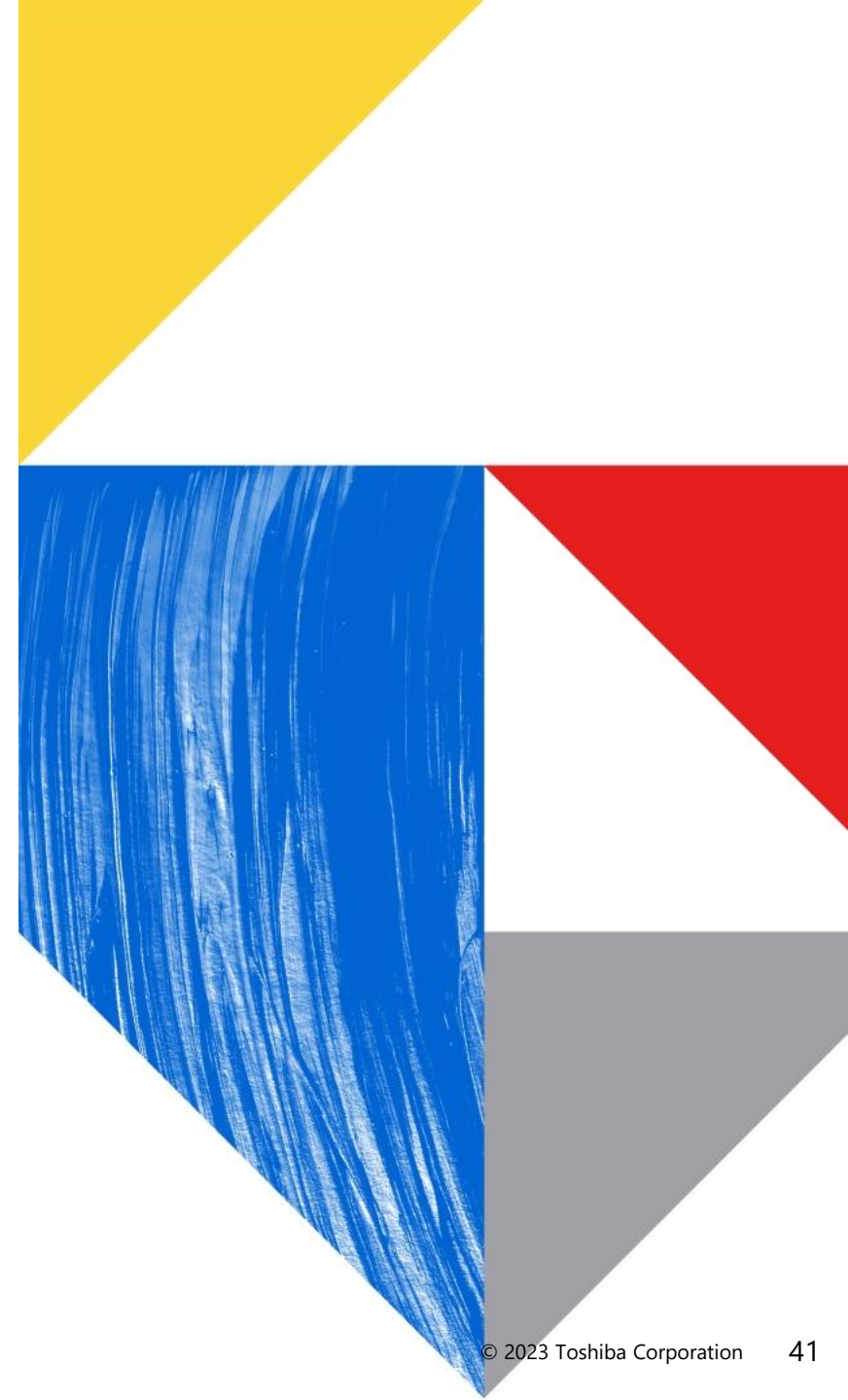
```
],  
  "dependencies": [  
    {  
      "ref": "acme-app",  
      "dependsOn": [  
        "pkg:maven/org.acme/web-framework@1.0.0",  
        "pkg:maven/org.acme/persistence@3.1.0"  
      ]  
    }  
  ]  
}
```

CycloneDX uses P-URL to denote dependencies



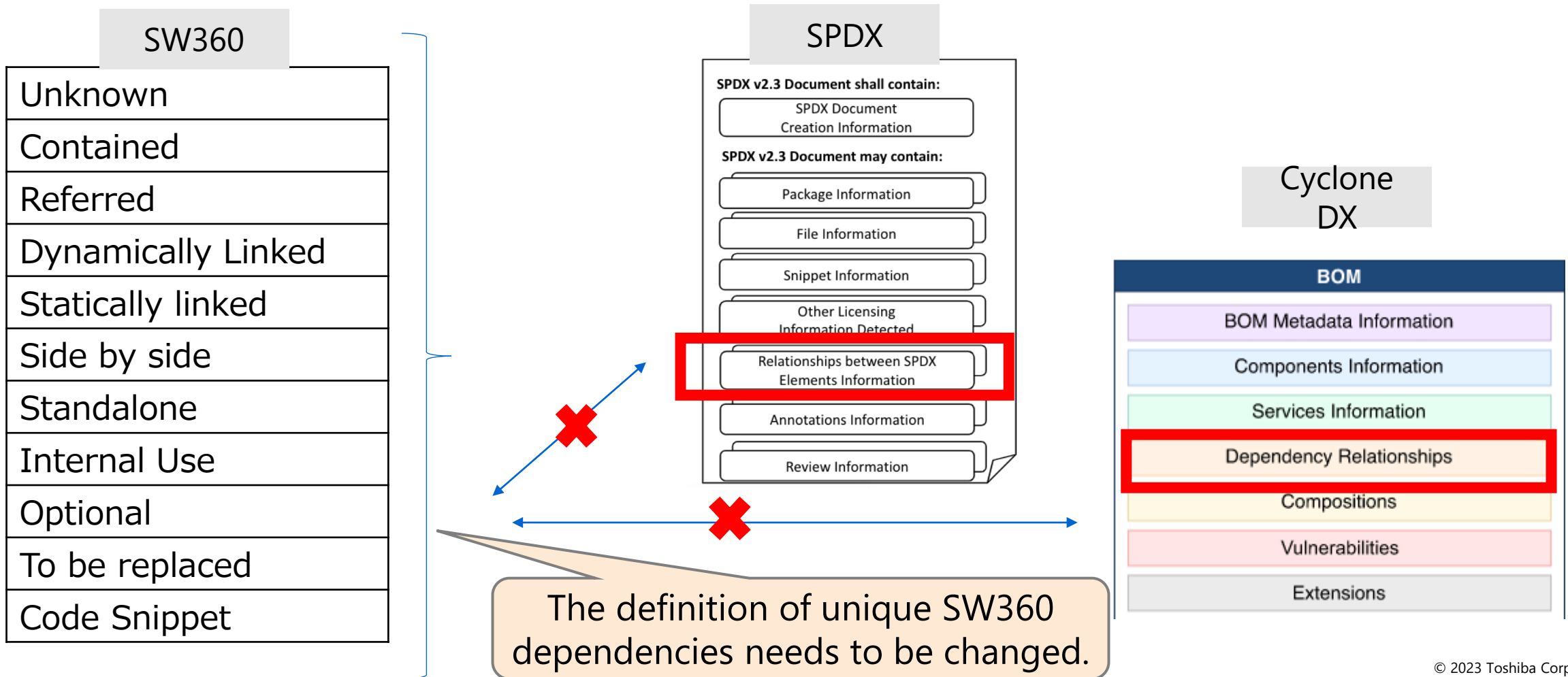
# 06

## Future Work for SBOM standards



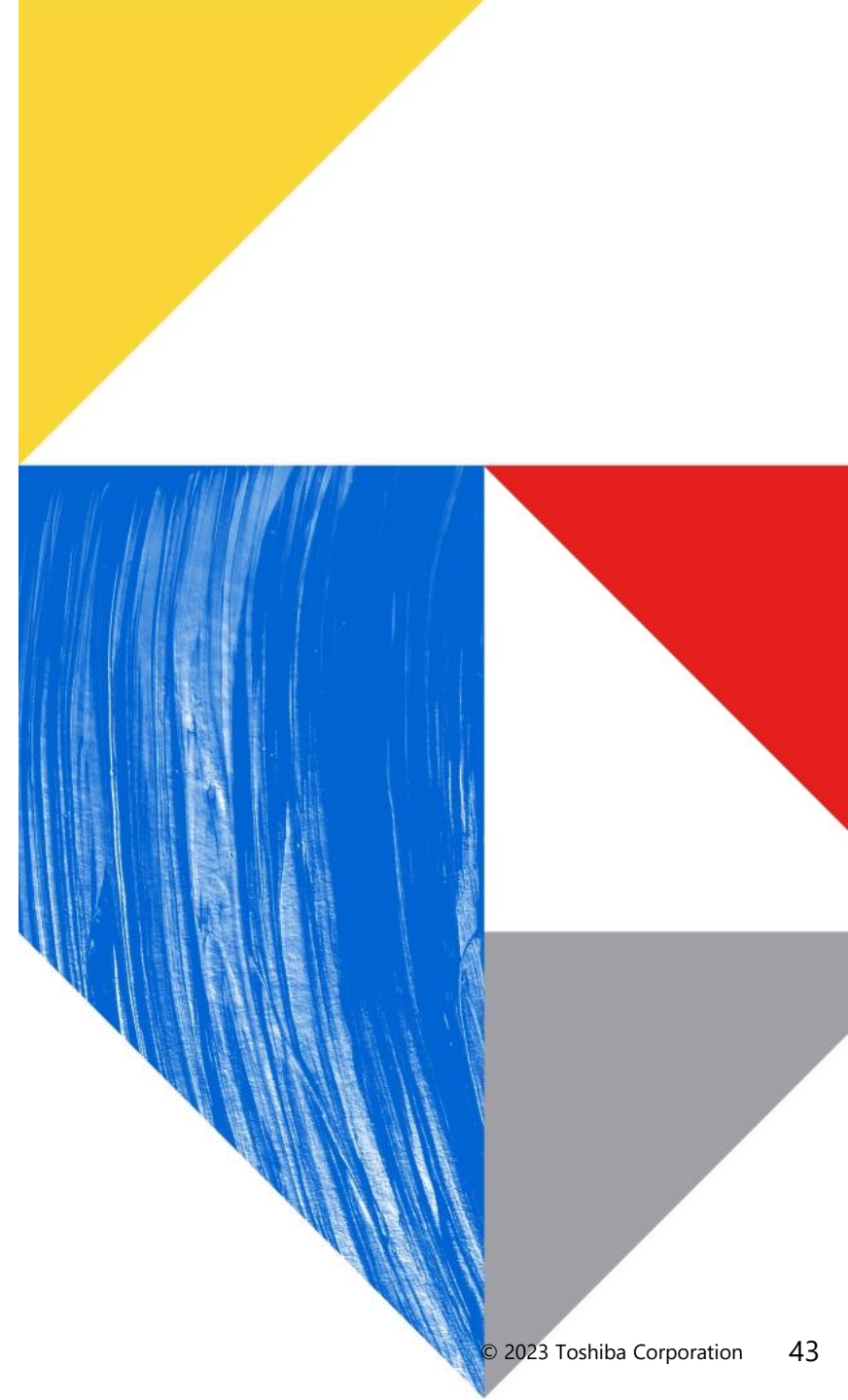
# Future Work

The relationship types defined in SW360 are different from the important SBOM relationship types.



# 07

## Summary

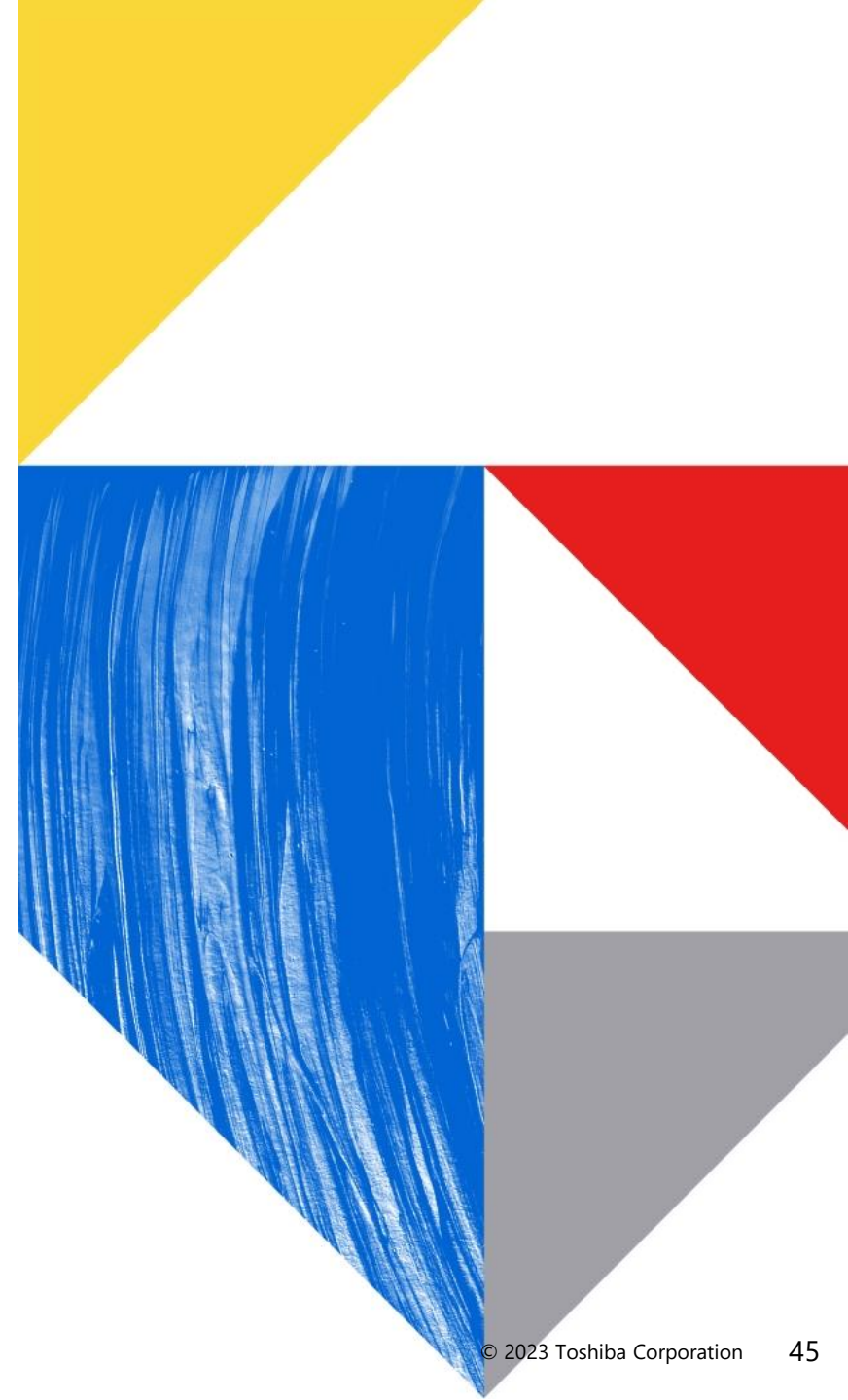


# Summary

- SW360 can manage internal software information.
- Registration of dependency is important for license and Security management.
- Registration of dependency information software in SW360 was not flexible.
- We developed a function to register different dependency information for each project to be registered.
- The definition of relevance between software is unique to SW360.
- In the future, it will be adapted to the common SBOM definition.

# Appendix

DEMO



**Committed to People,  
Committed to the Future.**



**TOSHIBA**