DepClean: Automatically revealing bloated software dependencies in Maven projects

César Soto Valero, Nicolas Harrand,
Martin Monperrus, and Benoit Baudry

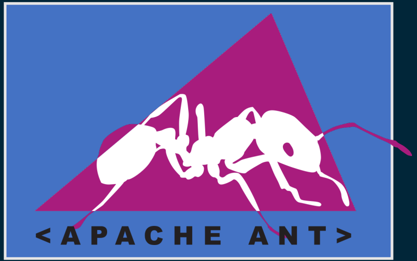
KTH
VETENSKAP
OCH KONST

CASTOR
Software Research Centre

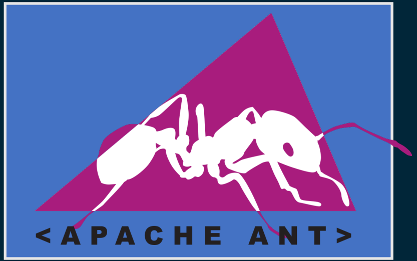
WASP
WALLENBERG AI,
AUTONOMOUS SYSTEMS
AND SOFTWARE PROGRAM

Dependency managers in Java

Dependency managers in Java



Dependency managers in Java



build.xml

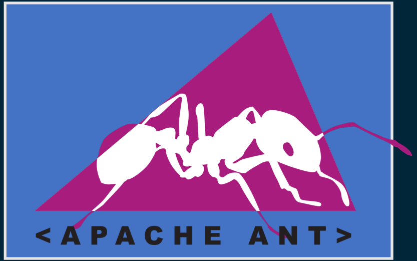
Dependency managers in Java

<APACHE ANT>

build.xml

maven

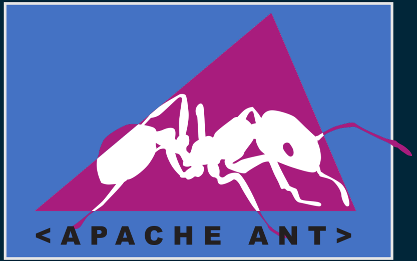
Dependency managers in Java



build.xml

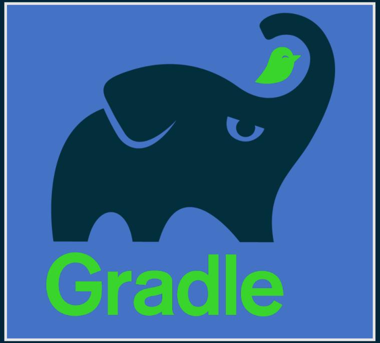
pom.xml

Dependency managers in Java

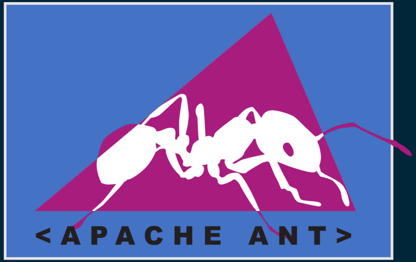


build.xml

pom.xml

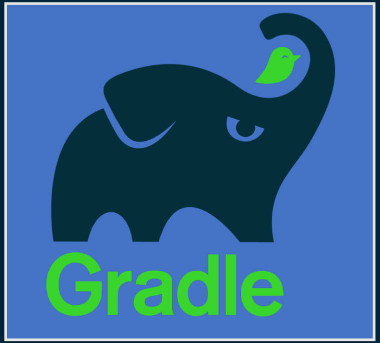


Dependency managers in Java



build.xml

pom.xml



build.gradle

What does Maven offer?

What does Maven offer?

- Build projects (compile, test, deploy)

What does Maven offer?

- Build projects (compile, test, deploy)
    - Resolve dependencies automatically

What does Maven offer?

- Build projects (compile, test, deploy)
  - Resolve dependencies automatically
  - Execute tests, add documentation, manage resources, etc.

What does Maven offer?

- Build projects (compile, test, deploy)
  - Resolve dependencies automatically
  - Execute tests, add documentation, manage resources, etc.
  - Release software artifacts to Maven Central

What does Maven offer?

- Build projects (compile, test, deploy)
  - Resolve dependencies automatically
  - Execute tests, add documentation, manage resources, etc.
  - Release software artifacts to Maven Central
- Guarantee reproducible builds (*pom.xml* file)

What does Maven offer?

- Build projects (compile, test, deploy)
  - Resolve dependencies automatically
  - Execute tests, add documentation, manage resources, etc.
  - Release software artifacts to Maven Central
- Guarantee reproducible builds (*pom.xml* file)
  - Provide consistent project structure

What does Maven offer?

- Build projects (compile, test, deploy)
  - Resolve dependencies automatically
  - Execute tests, add documentation, manage resources, etc.
  - Release software artifacts to Maven Central
- Guarantee reproducible builds (*pom.xml* file)
  - Provide consistent project structure
  - Allow us to use customized plugins

What does Maven offer?

- Build projects (compile, test, deploy)
  - Resolve dependencies automatically
  - Execute tests, add documentation, manage resources, etc.
  - Release software artifacts to Maven Central
- Guarantee reproducible builds (*pom.xml* file)
  - Provide consistent project structure
  - Allow us to use customized plugins
  - Analyze the dependencies in our projects

What does Maven offer?

- Build projects (compile, test, deploy)
  - Resolve dependencies automatically
  - Execute tests, add documentation, manage resources, etc.
  - Release software artifacts to Maven Central
- Guarantee reproducible builds (*pom.xml* file)
  - Provide consistent project structure
  - Allow us to use customized plugins
  - Analyze the dependencies in our projects

How many dependencies do we actually use?

Dependency tree

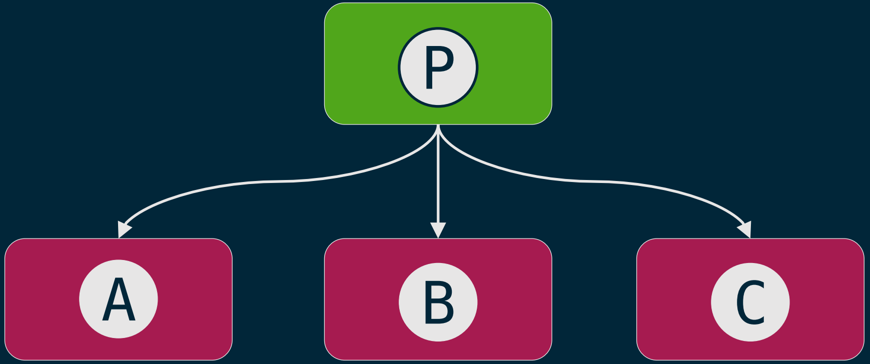
Dependency tree

Dependency tree

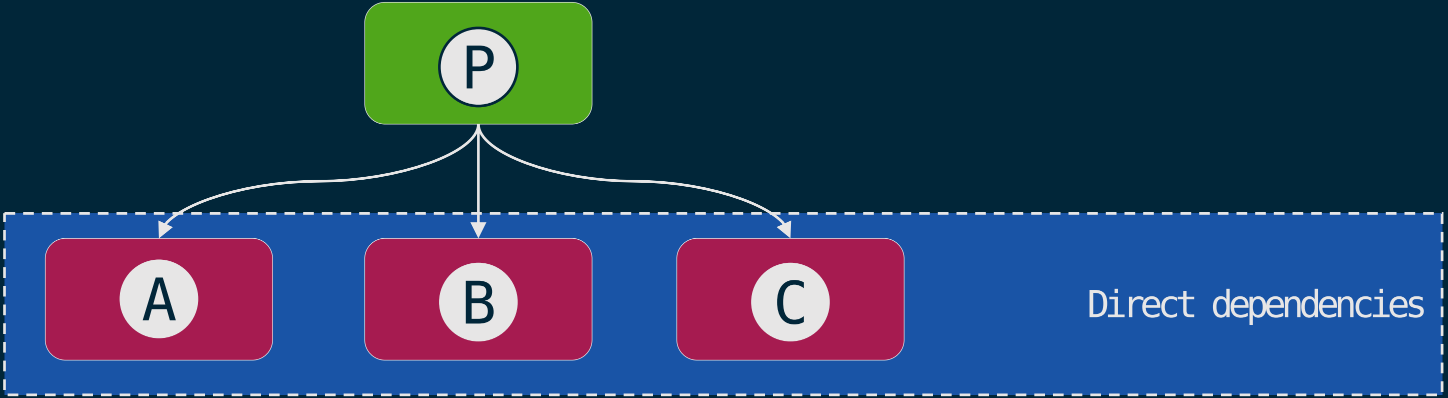
```
<dependency>
    <groupId>org.A</groupId>
    <artifactId>A</artifactId>
</dependency>
<dependency>
    <groupId>org.B</groupId>
    <artifactId>B</artifactId>
</dependency>
<dependency>
    <groupId>org.C</groupId>
    <artifactId>C</artifactId>
</dependency>
```

Dependency tree

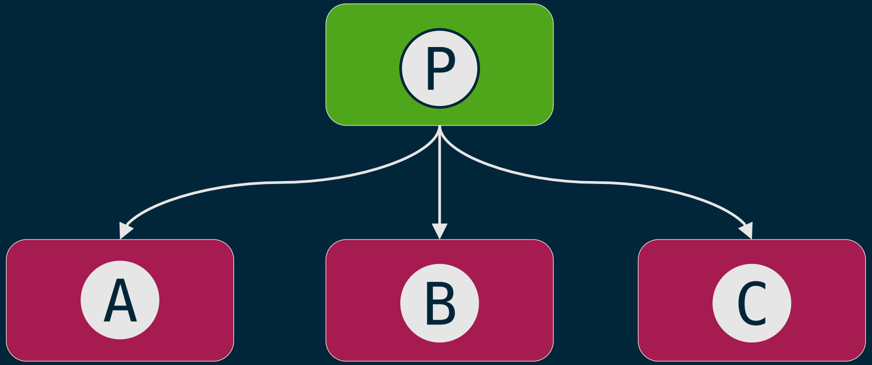
Dependency tree



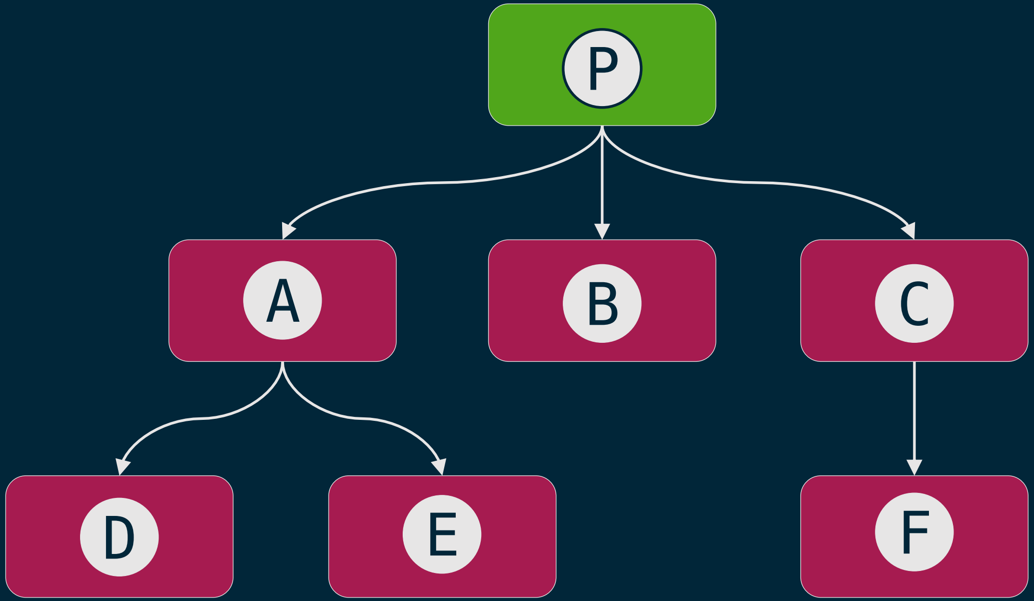
Dependency tree



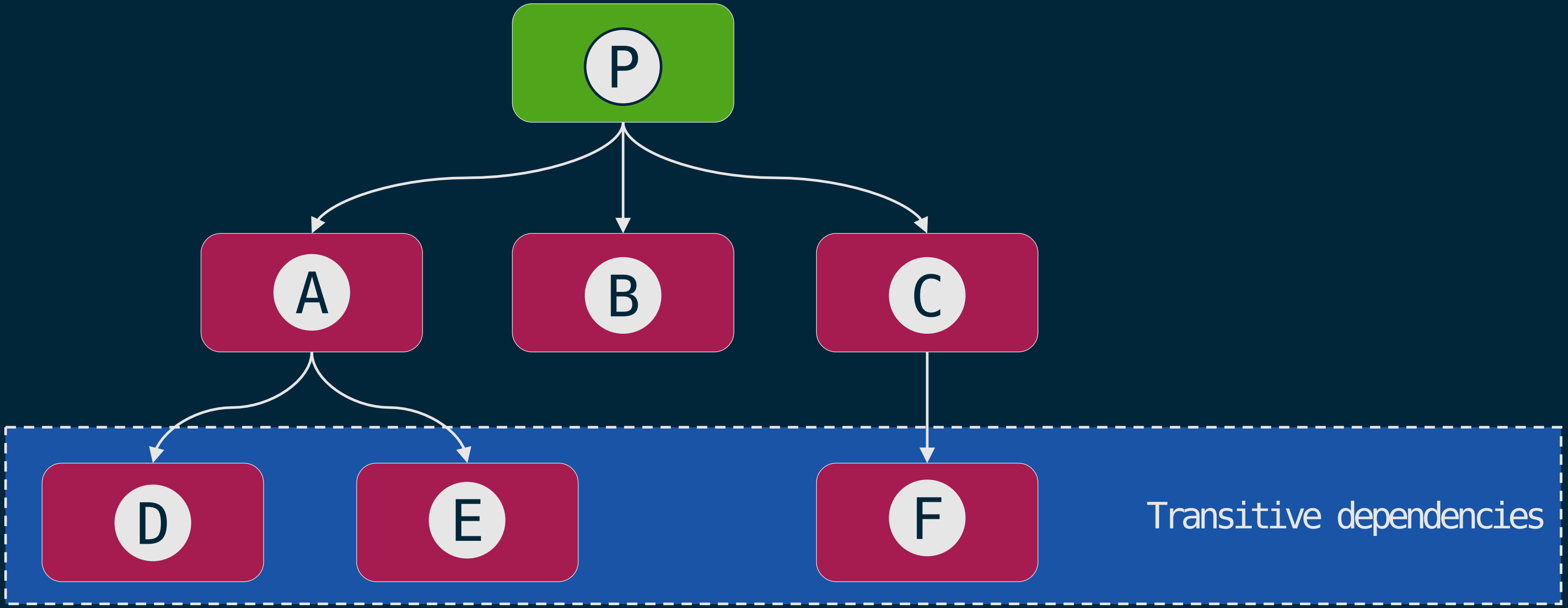
Dependency tree



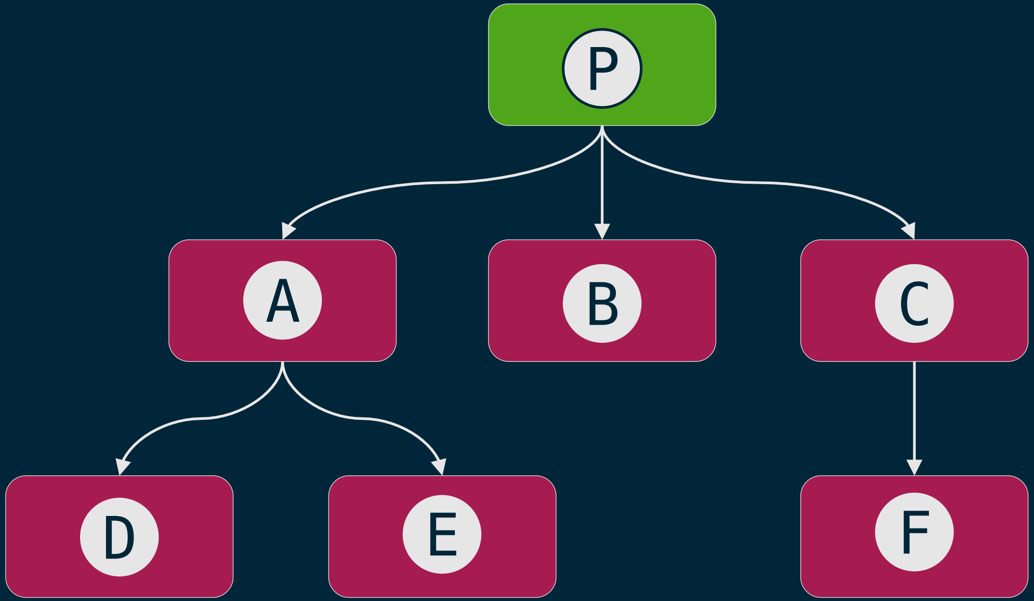
Dependency tree



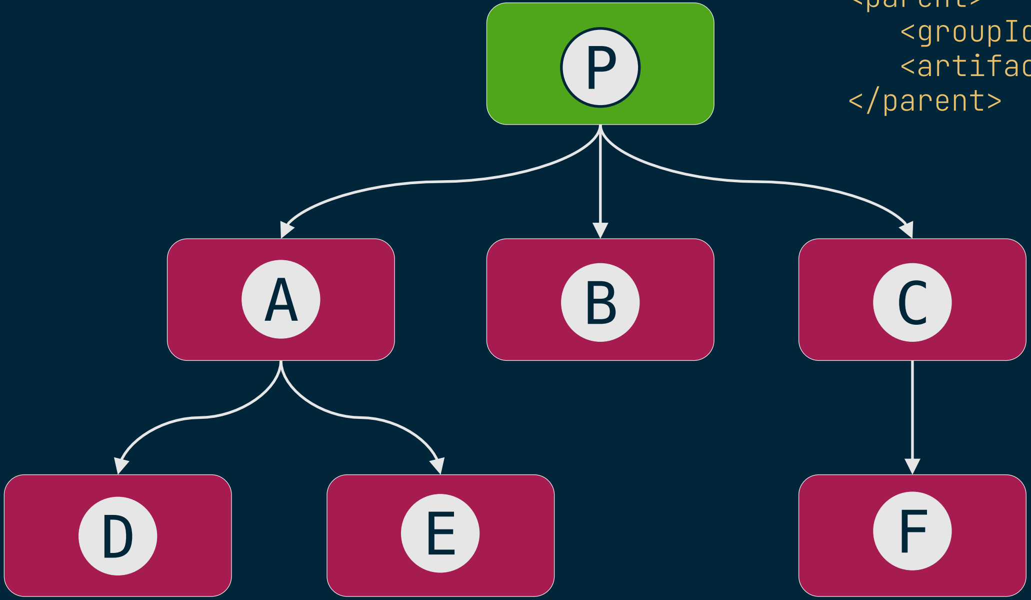
Dependency tree



Dependency tree

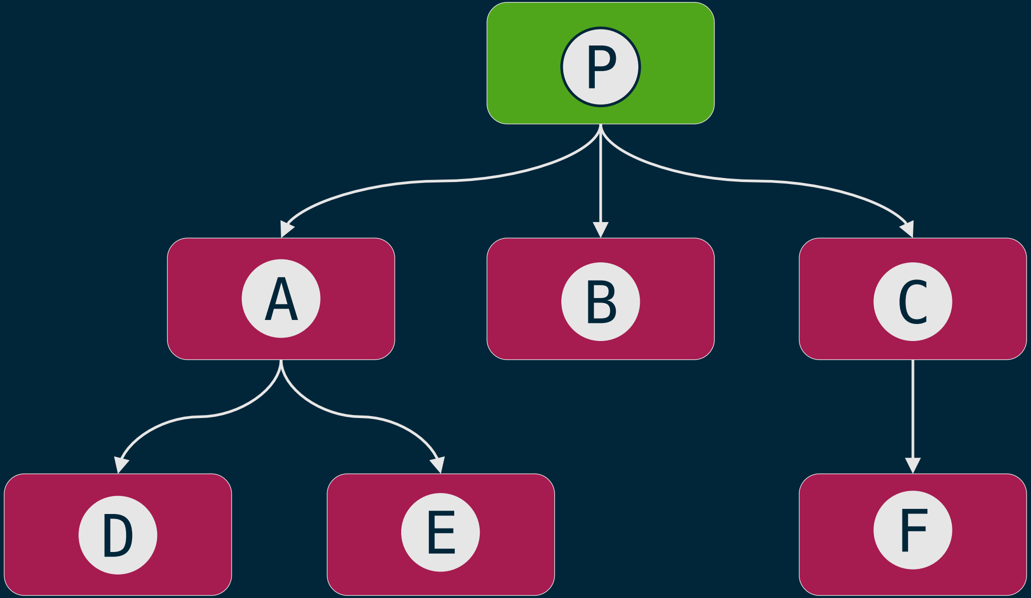


Dependency tree

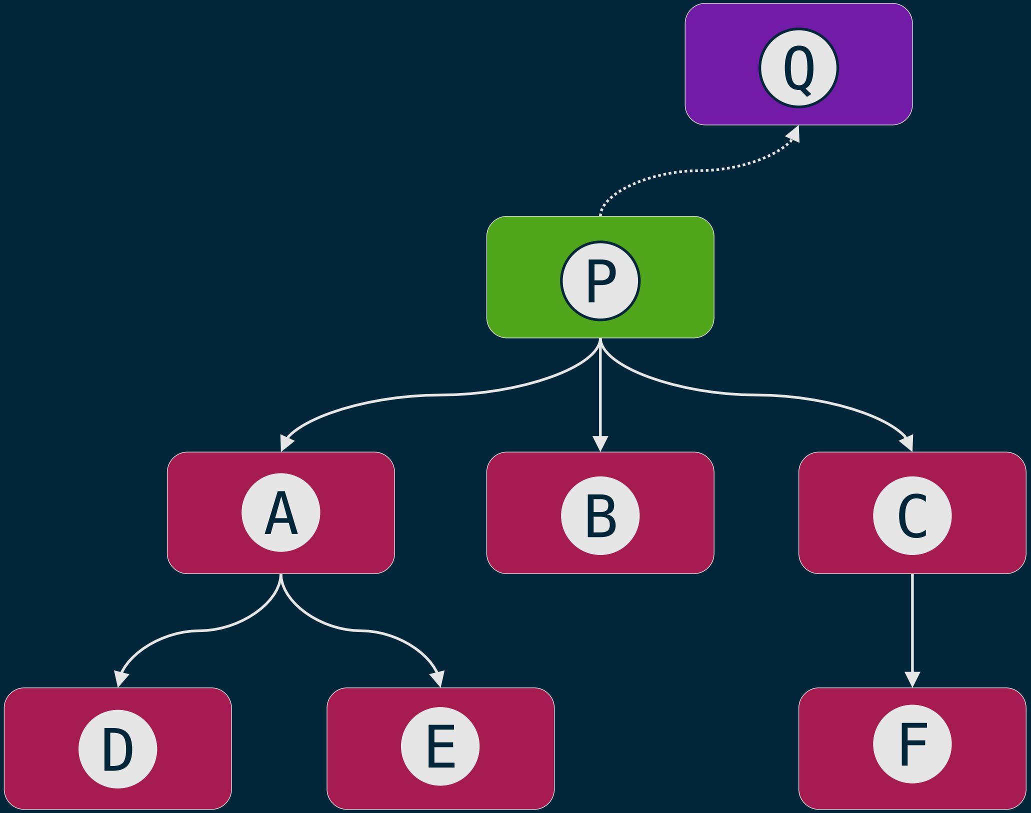


```
<parent>
   <groupId>org.Q</groupId>
   <artifactId>Q</artifactId>
</parent>
```

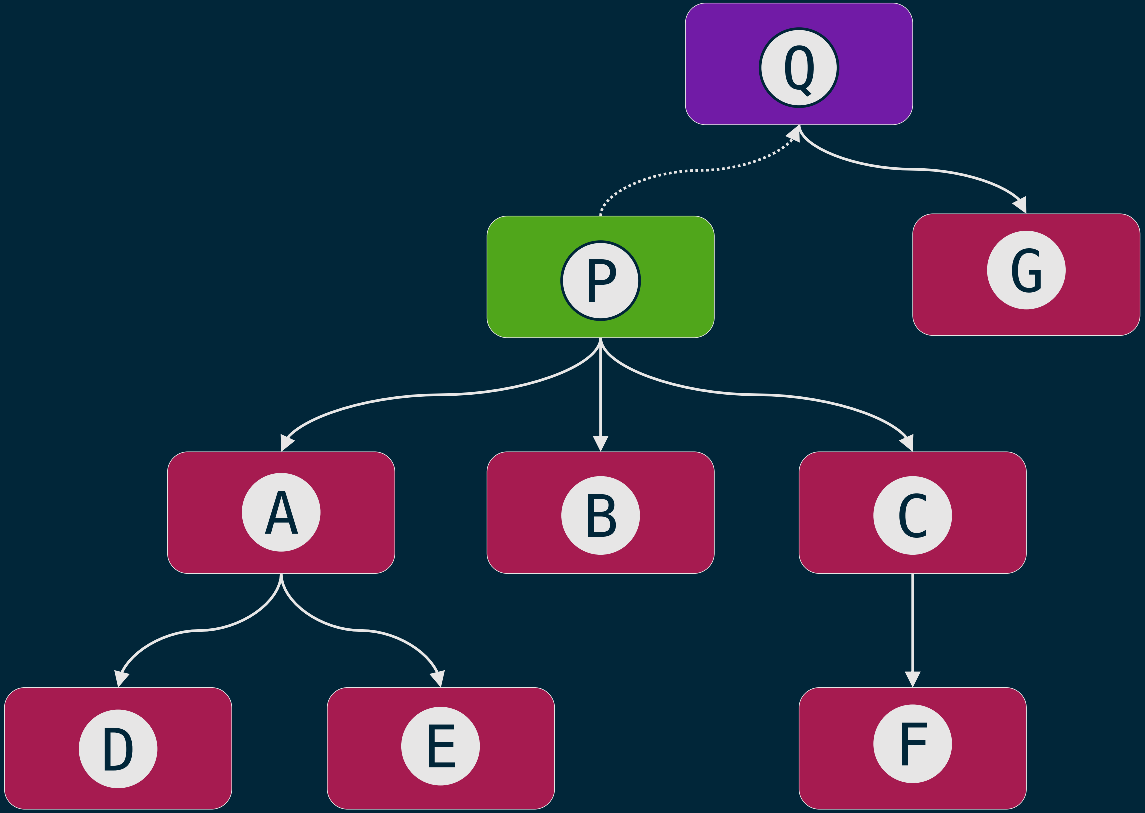
Dependency tree



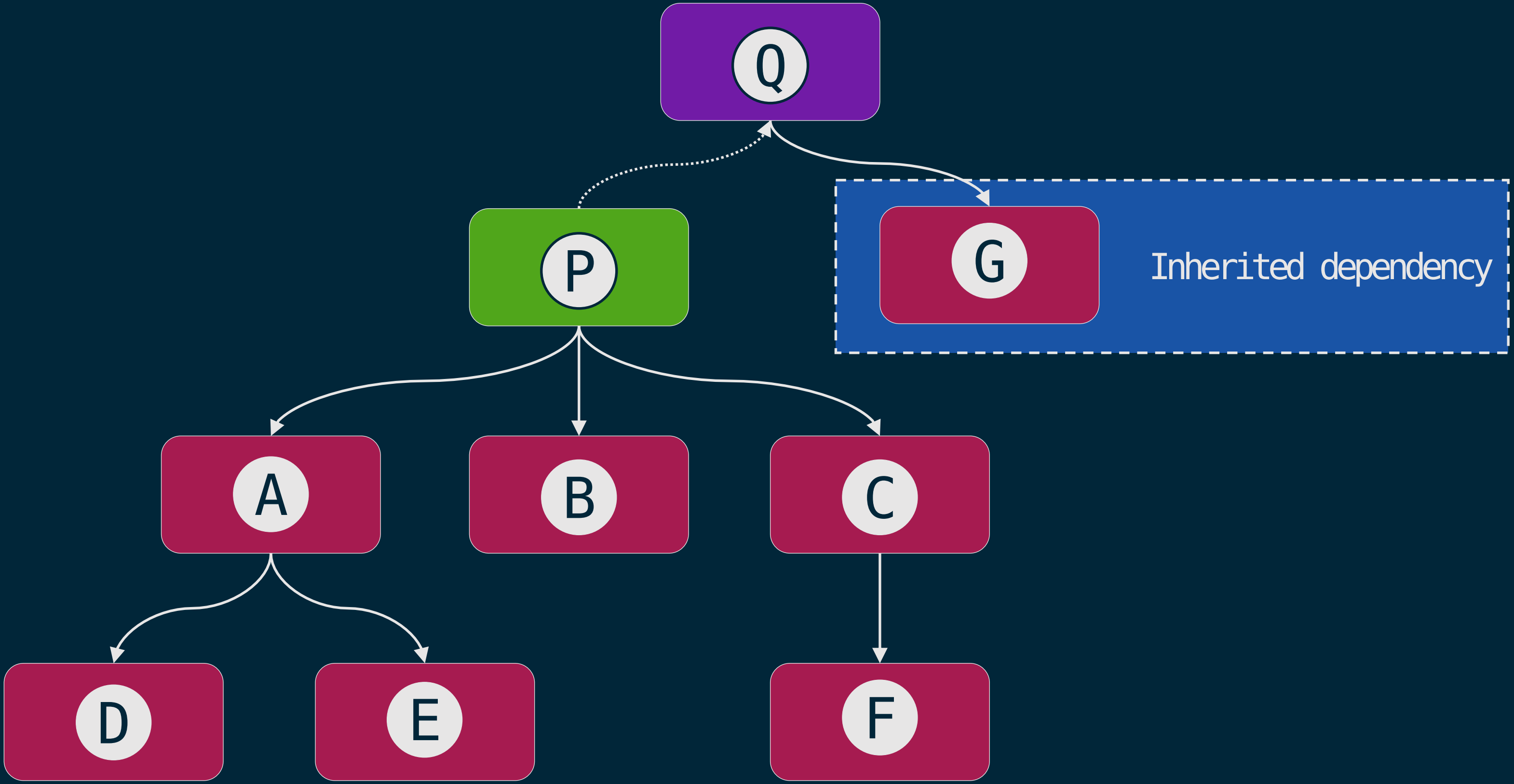
Dependency tree



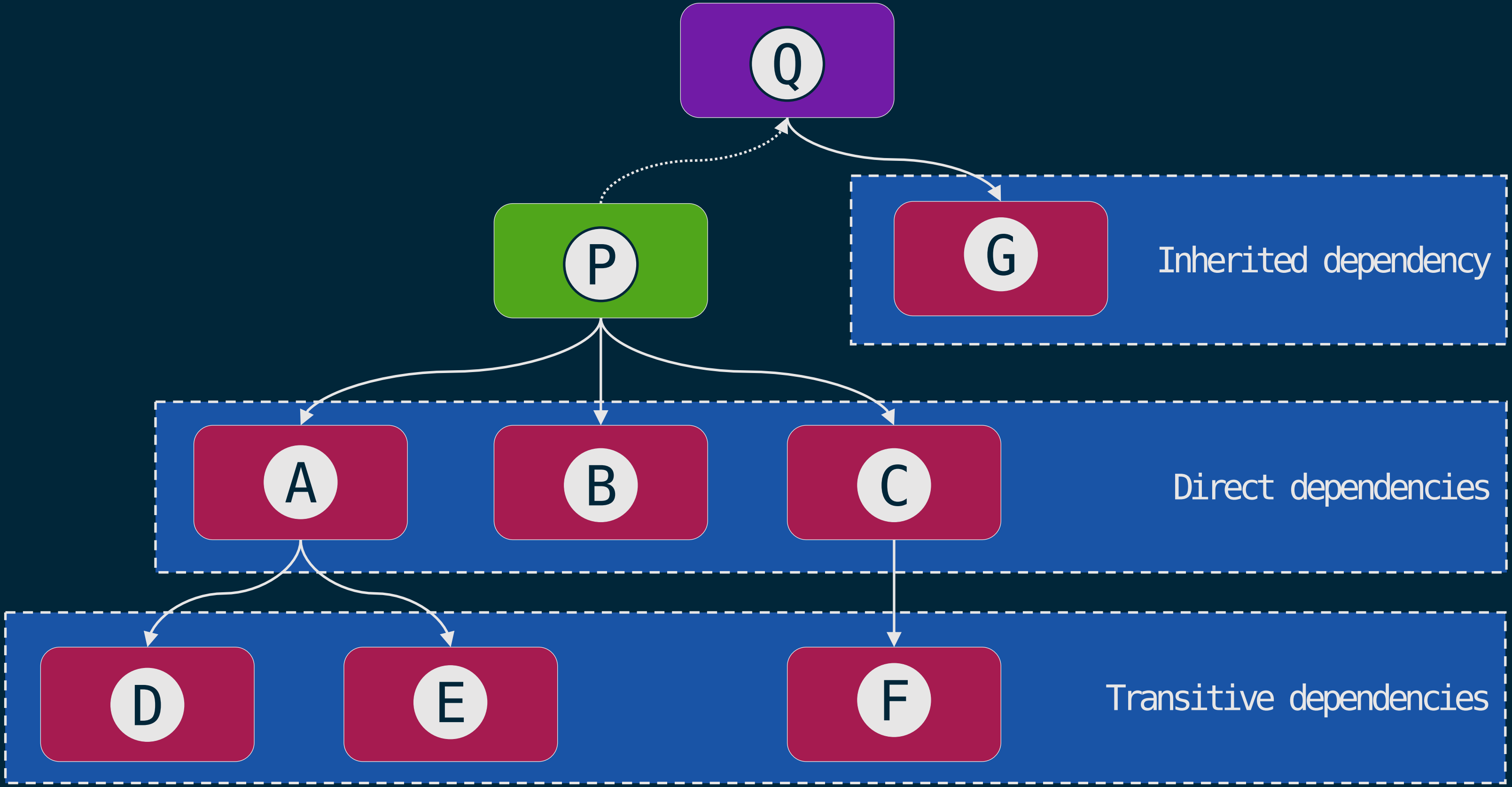
Dependency tree



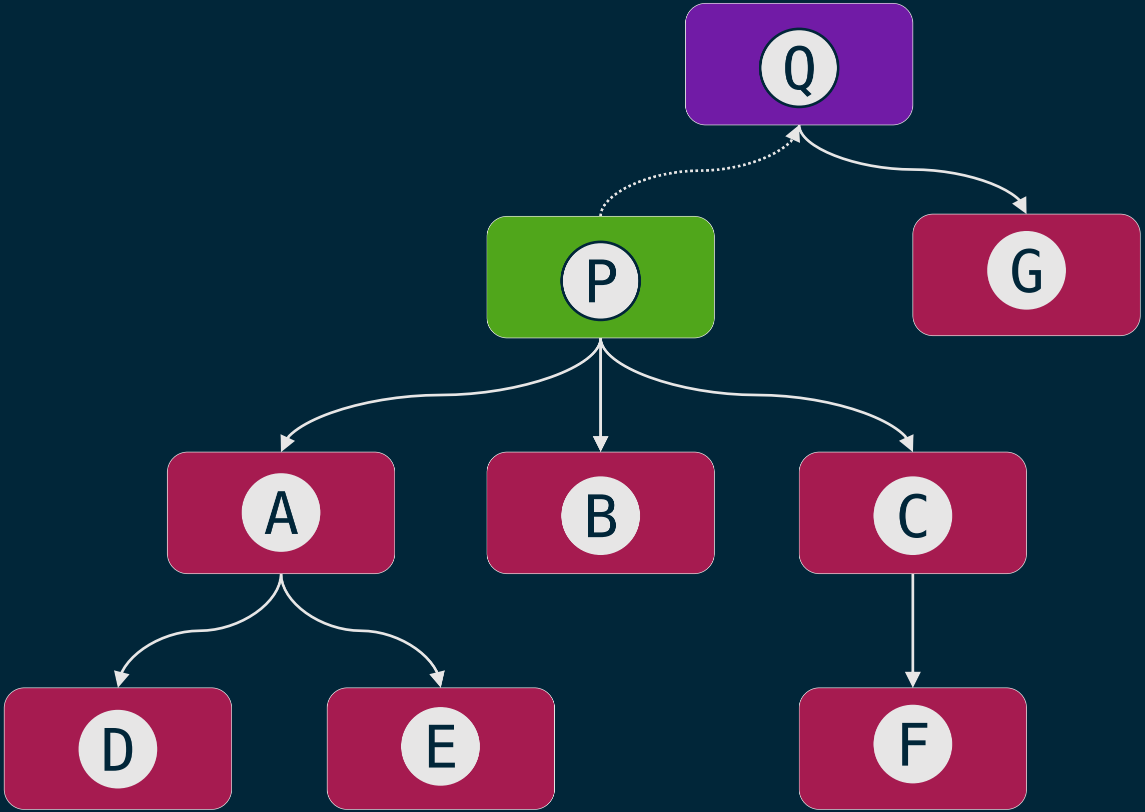
Dependency tree



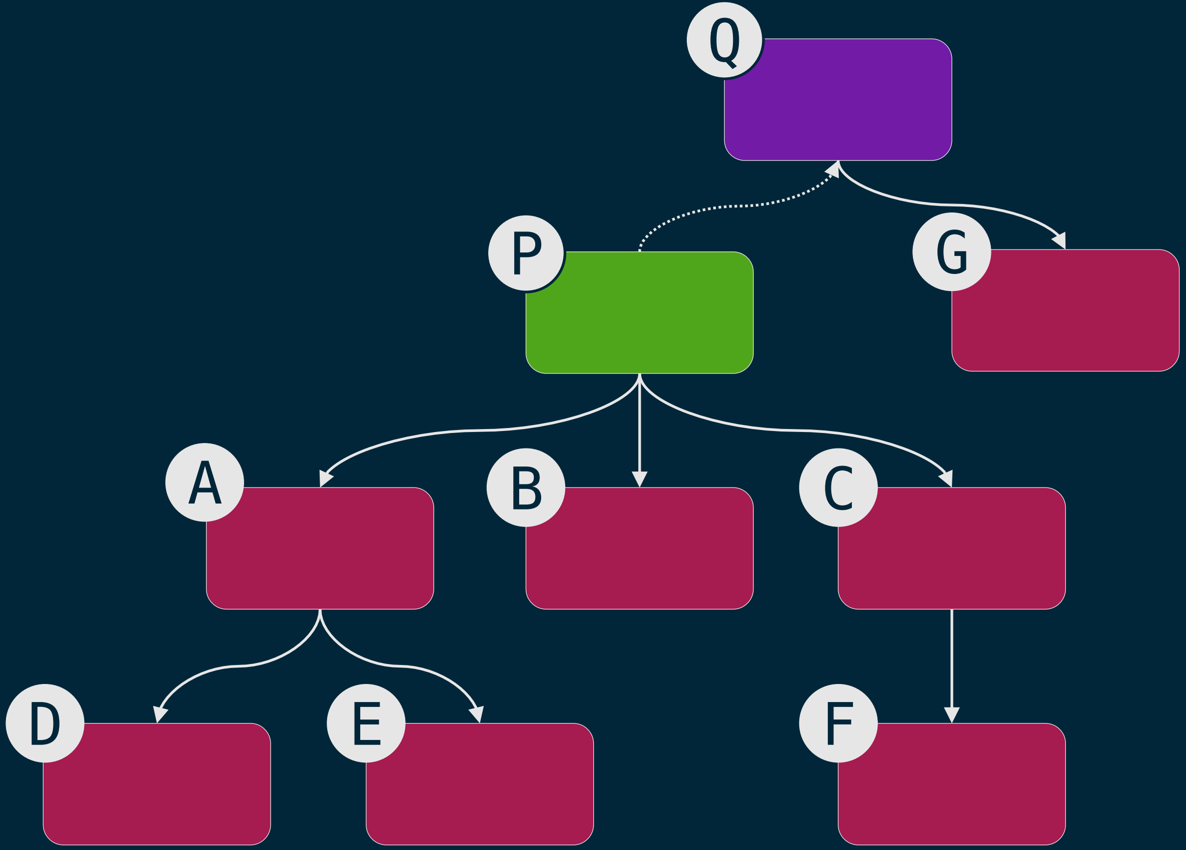
Dependency tree



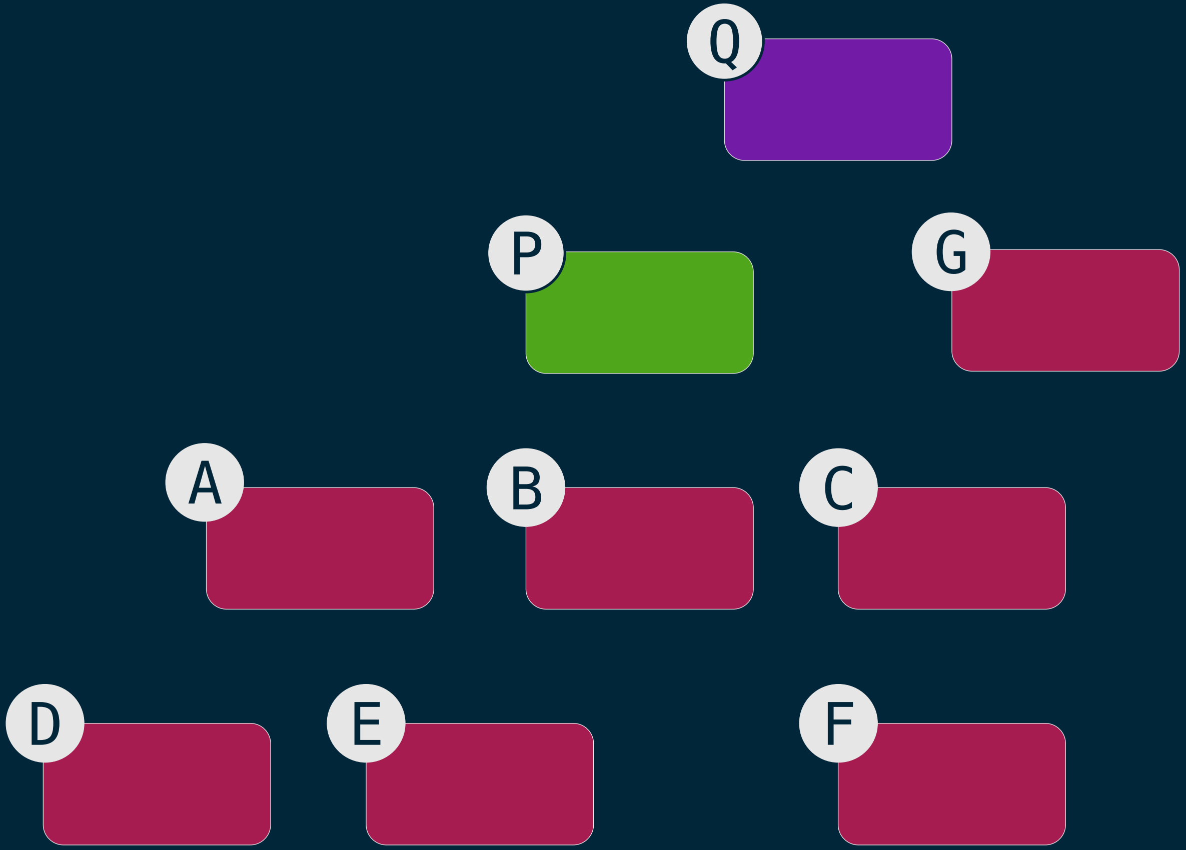
Dependency tree



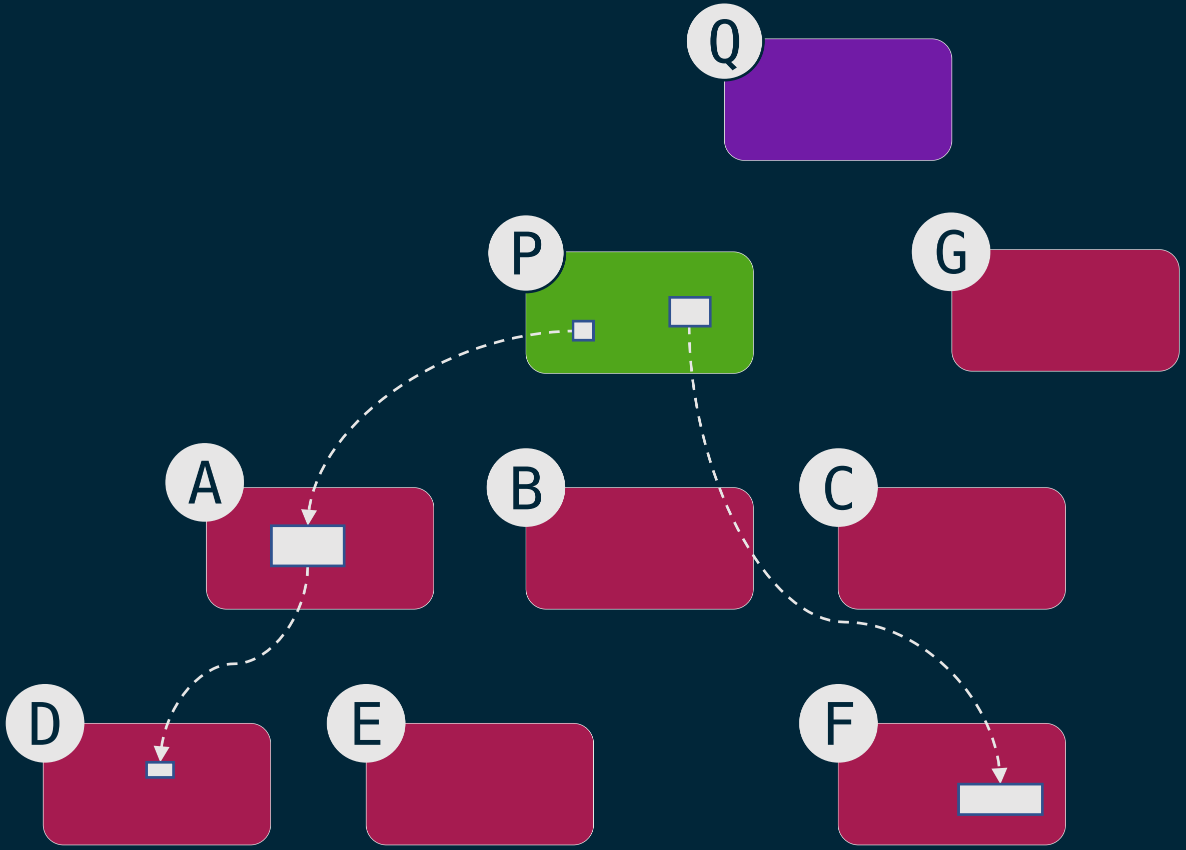
Dependency tree



Bytecode calls



Bytecode calls



DepClean

DepClean

- Detect and report bloated dependencies

DepClean

- Detect and report bloated dependencies
  - In the context of an artifact

DepClean

- Detect and report bloated dependencies
  - In the context of an artifact
  - On the whole dependency tree

DepClean

- Detect and report bloated dependencies
  - In the context of an artifact
  - On the whole dependency tree
- Automatic generation of a debloated *pom.xml* file

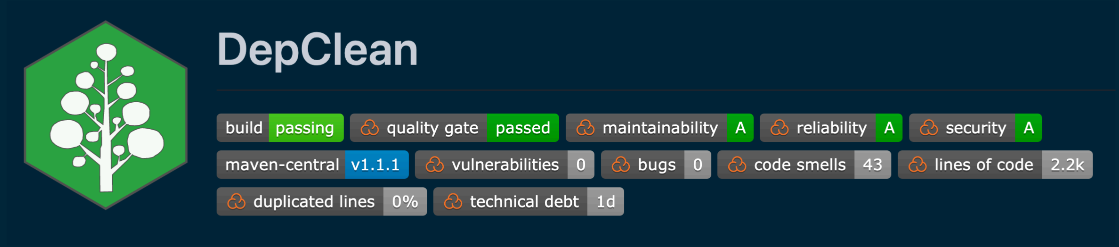
DepClean

- Detect and report bloated dependencies
  - In the context of an artifact
  - On the whole dependency tree
- Automatic generation of a debloated *pom.xml* file
- Open source

DepClean

- Detect and report bloated dependencies
  - In the context of an artifact
  - On the whole dependency tree
- Automatic generation of a debloated *pom.xml* file
- Open source

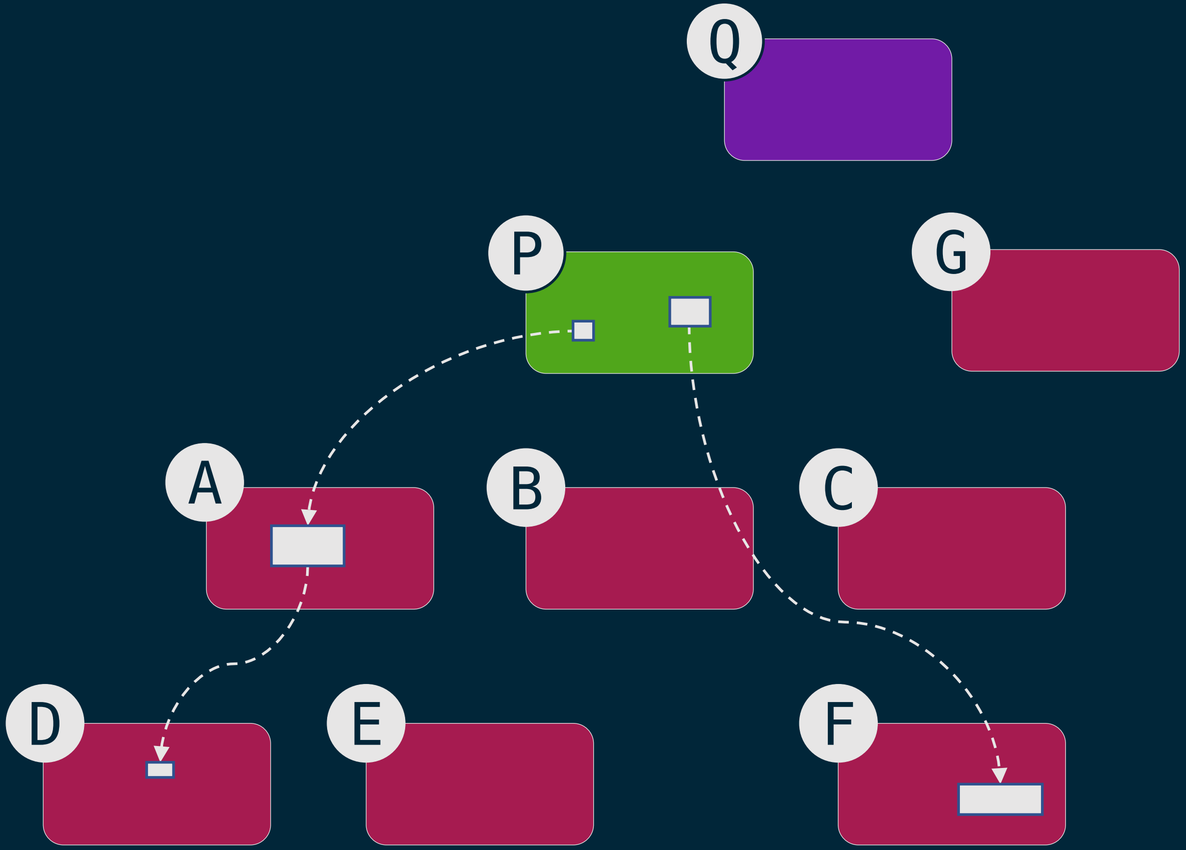
https://github.com/castor-software/depclean



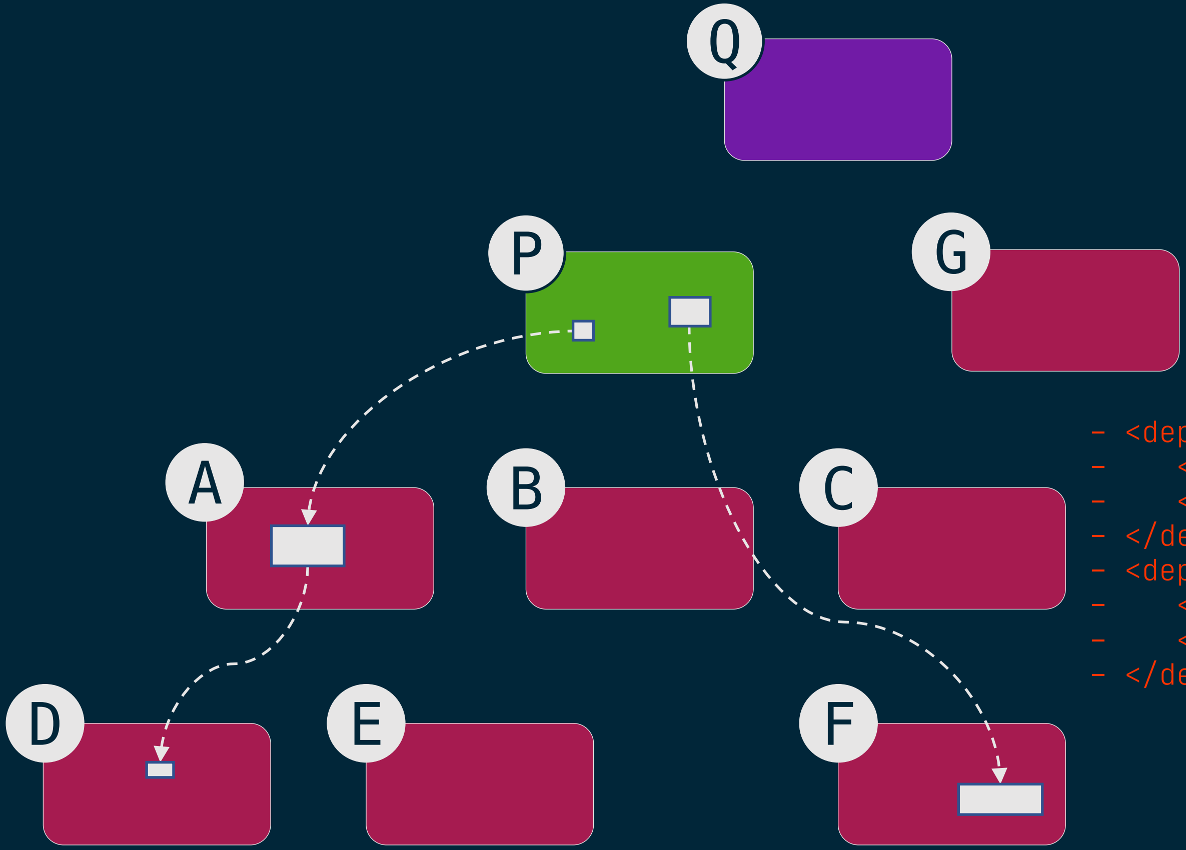
DepClean

build passing | quality gate passed | maintainability A | reliability A | security A
maven-central v1.1.1 | vulnerabilities 0 | bugs 0 | code smells 43 | lines of code 2.2k
duplicated lines 0% | technical debt 1d

Debloat direct dependencies

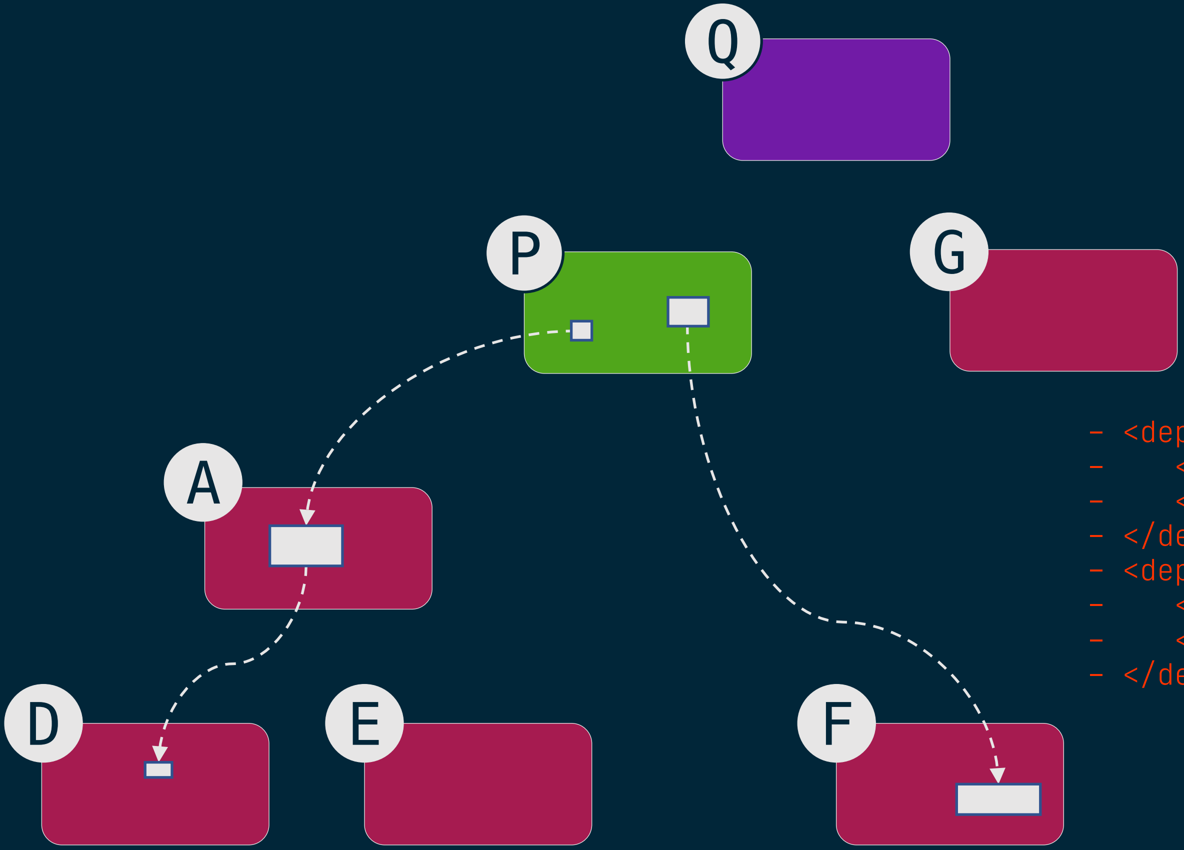


Debloat direct dependencies



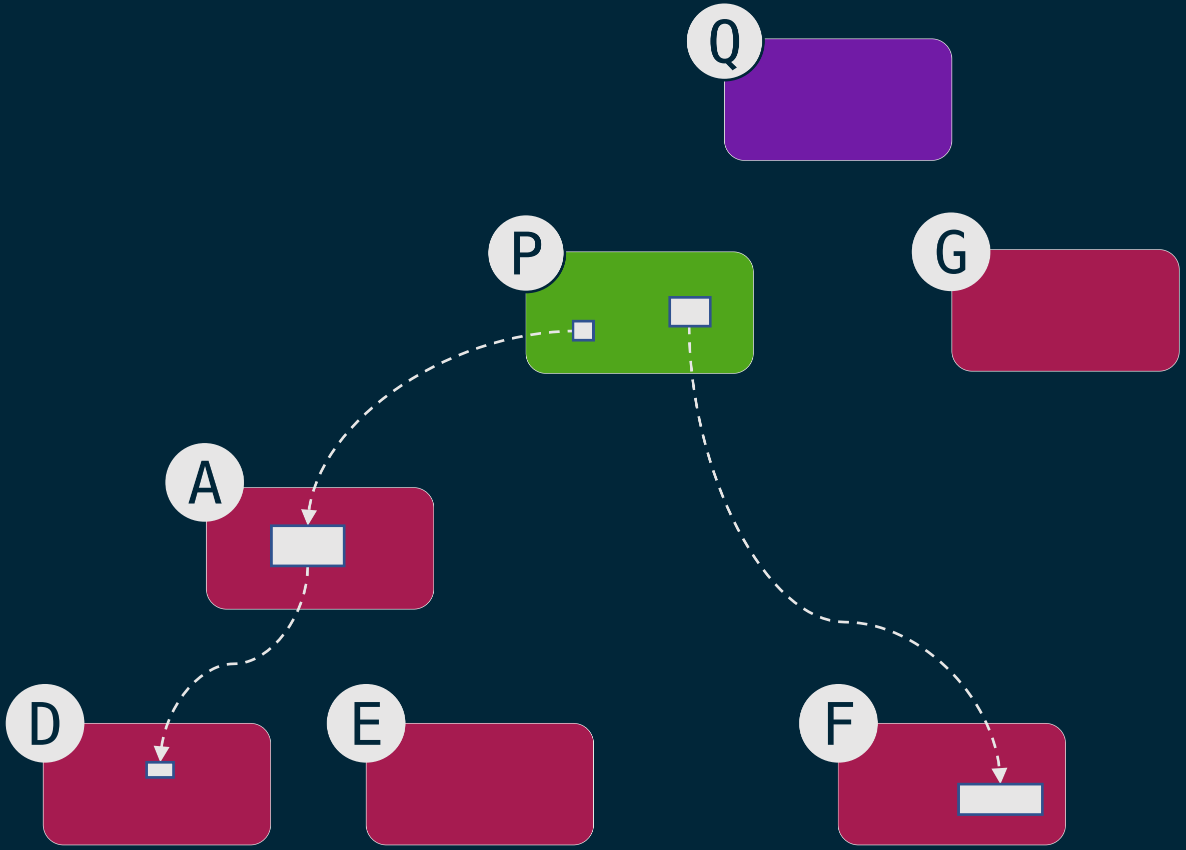
```
- <dependency>
-     <groupId>org.B</groupId>
-     <artifactId>B</artifactId>
- </dependency>
- <dependency>
-     <groupId>org.C</groupId>
-     <artifactId>C</artifactId>
- </dependency>
```

Debloat direct dependencies

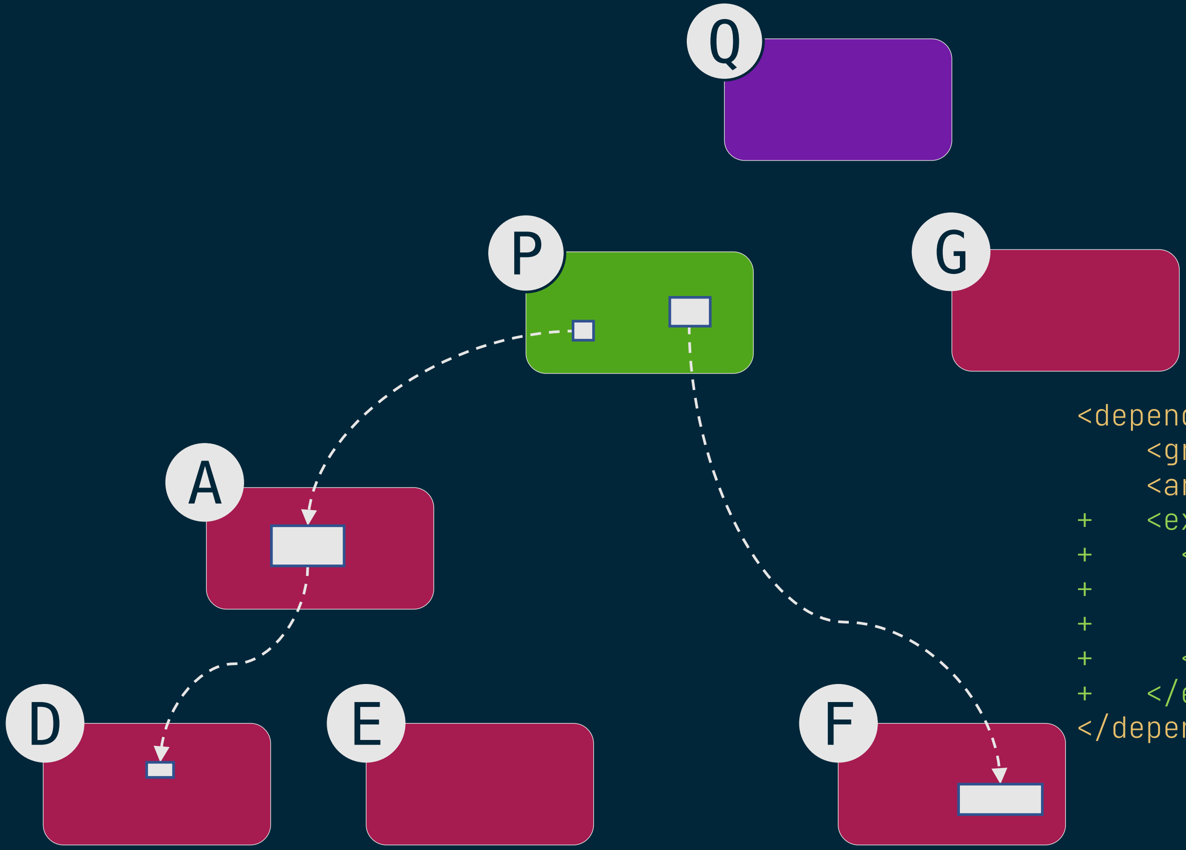


```
- <dependency>
-     <groupId>org.B</groupId>
-     <artifactId>B</artifactId>
- </dependency>
- <dependency>
-     <groupId>org.C</groupId>
-     <artifactId>C</artifactId>
- </dependency>
```

Debloat transitive dependencies

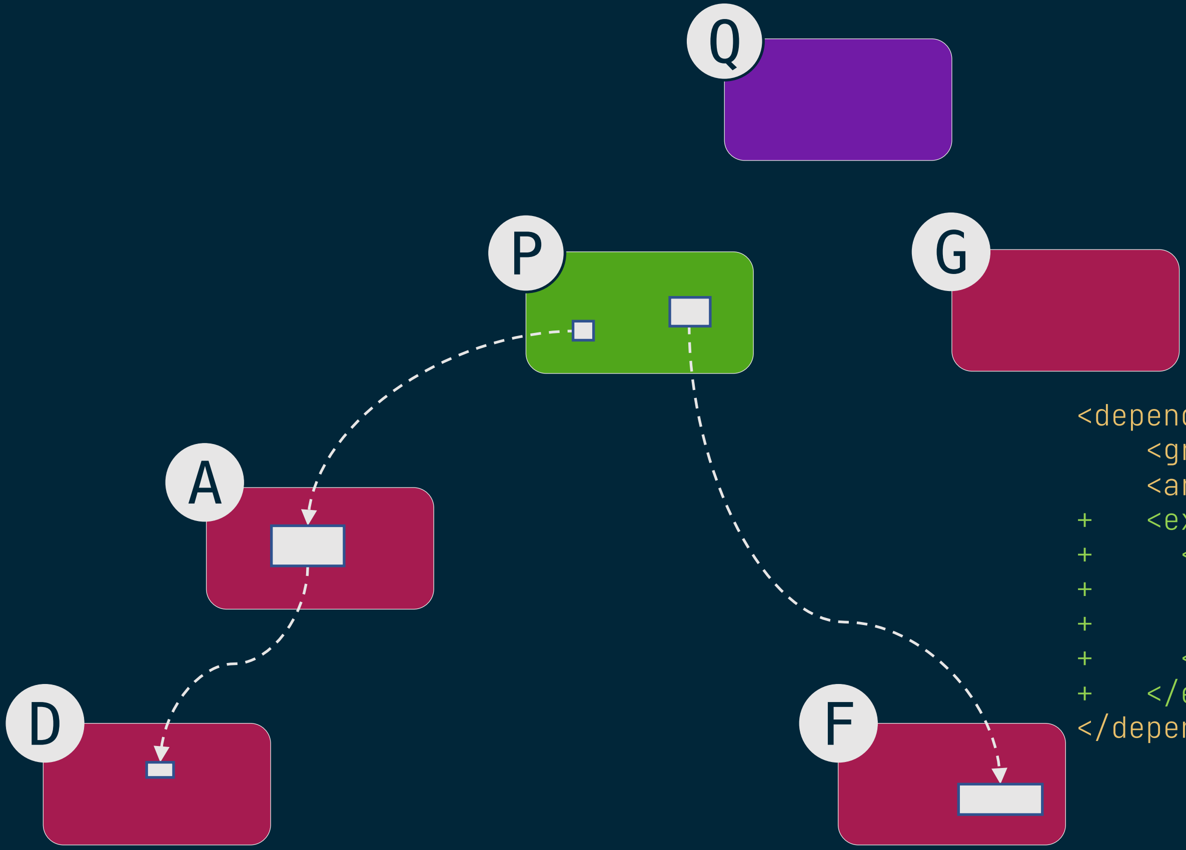


Debloat transitive dependencies



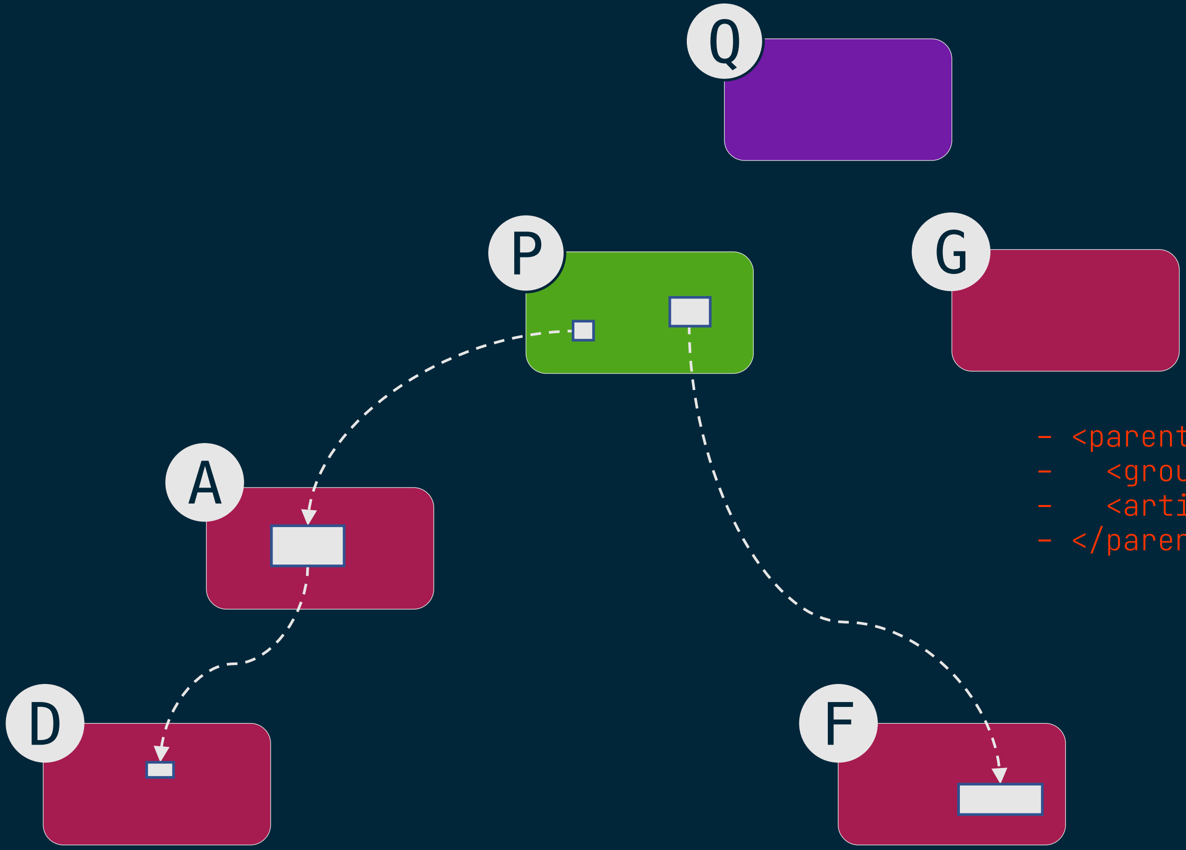
```
<dependency>
    <groupId>org.A</groupId>
    <artifactId>A</artifactId>
+   <exclusions>
+     <exclusion>
+       <groupId>org.E</groupId>
+       <artifactId>E</artifactId>
+     </exclusion>
+   </exclusions>
</dependency>
```

Debloat transitive dependencies



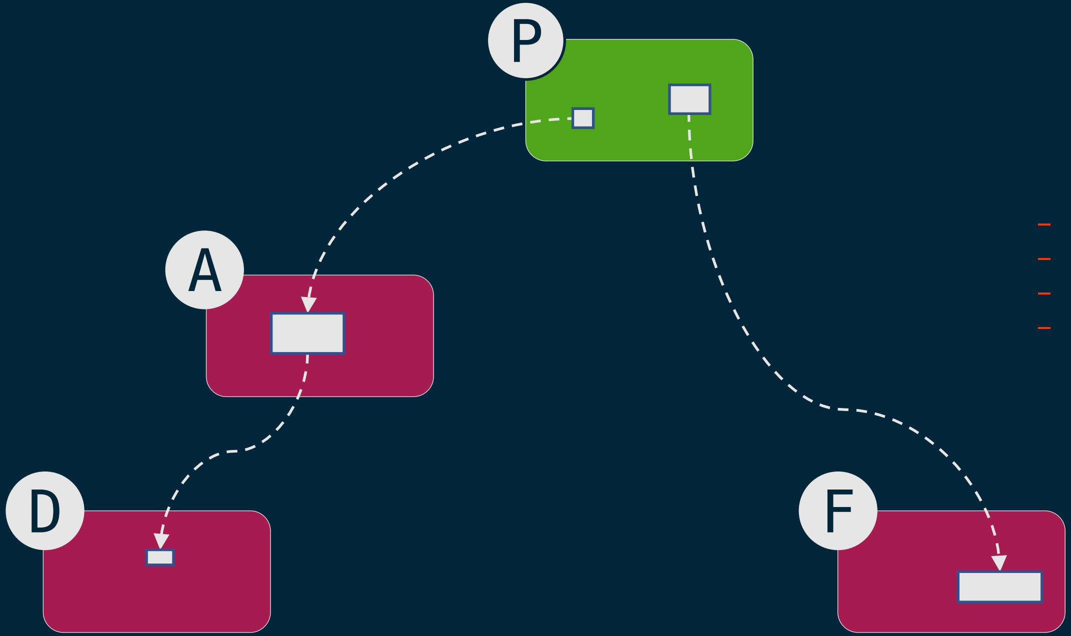
```
<dependency>
    <groupId>org.A</groupId>
    <artifactId>A</artifactId>
+   <exclusions>
+     <exclusion>
+       <groupId>org.E</groupId>
+       <artifactId>E</artifactId>
+     </exclusion>
+   </exclusions>
</dependency>
```

Debloat inherited dependencies



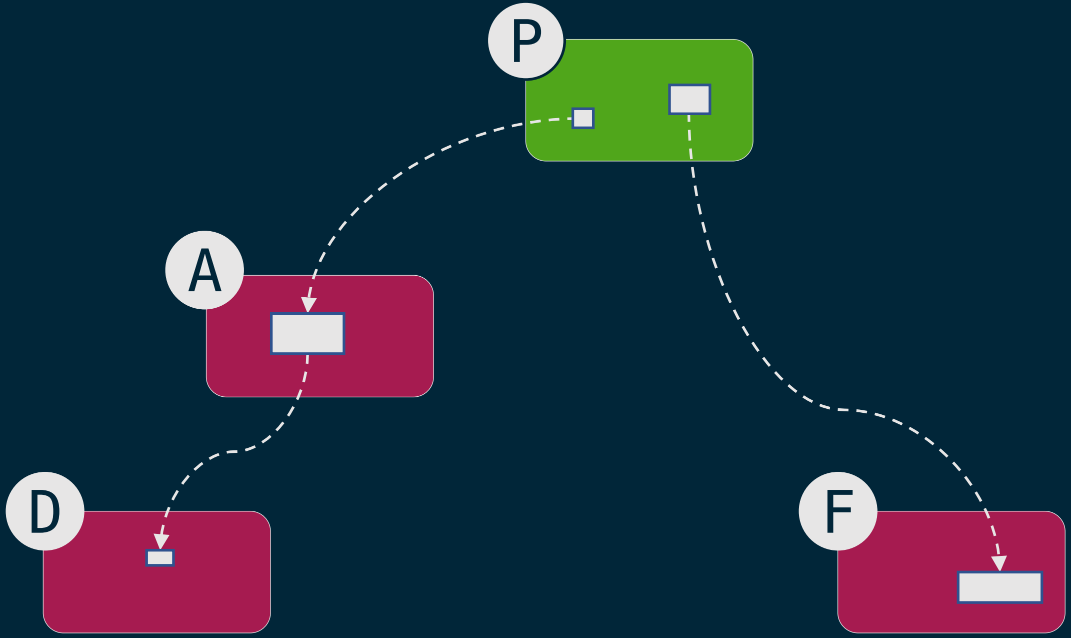
```
- <parent>
-   <groupId>org.Q</groupId>
-   <artifactId>Q</artifactId>
- </parent>
```

Debloat inherited dependencies

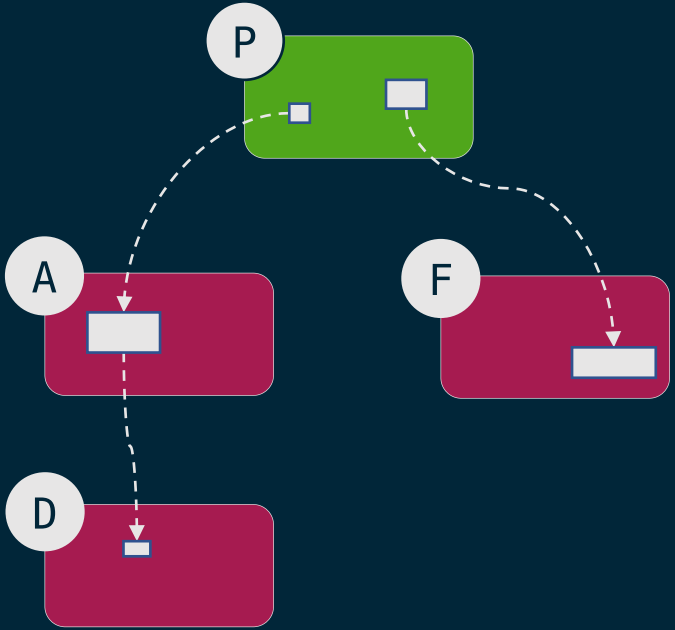


```
- <parent>
-   <groupId>org.Q</groupId>
-   <artifactId>Q</artifactId>
- </parent>
```

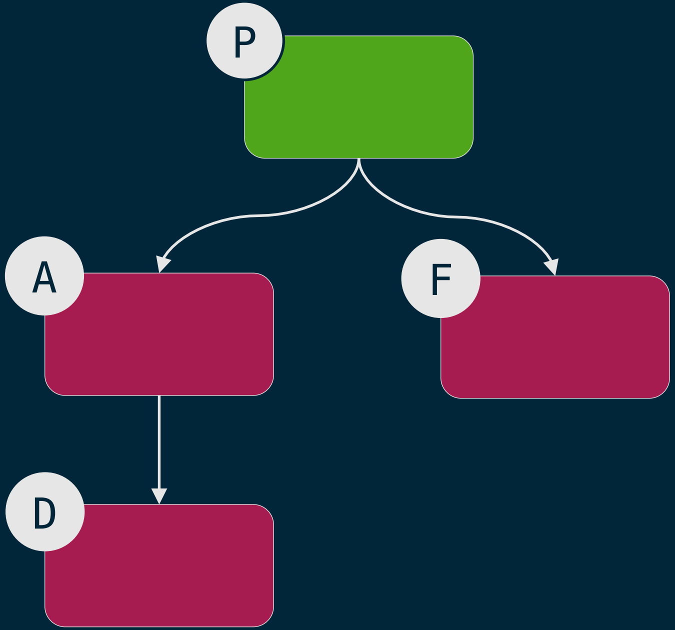
Debloat inherited dependencies



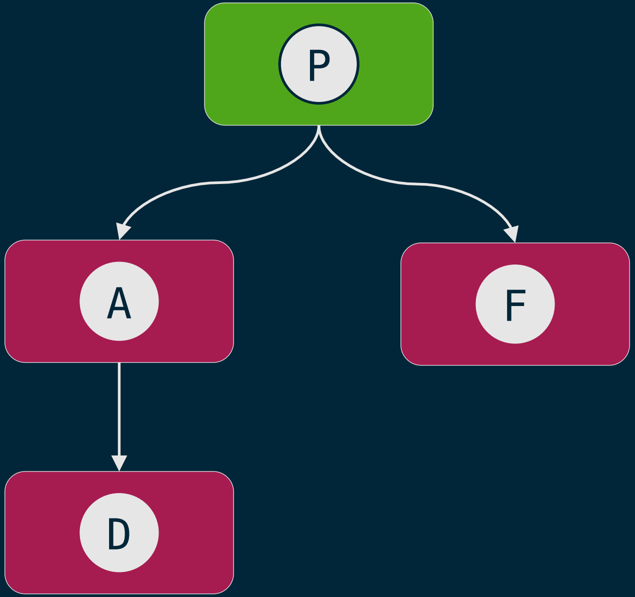
Debloat analysis result



Debloated dependency tree

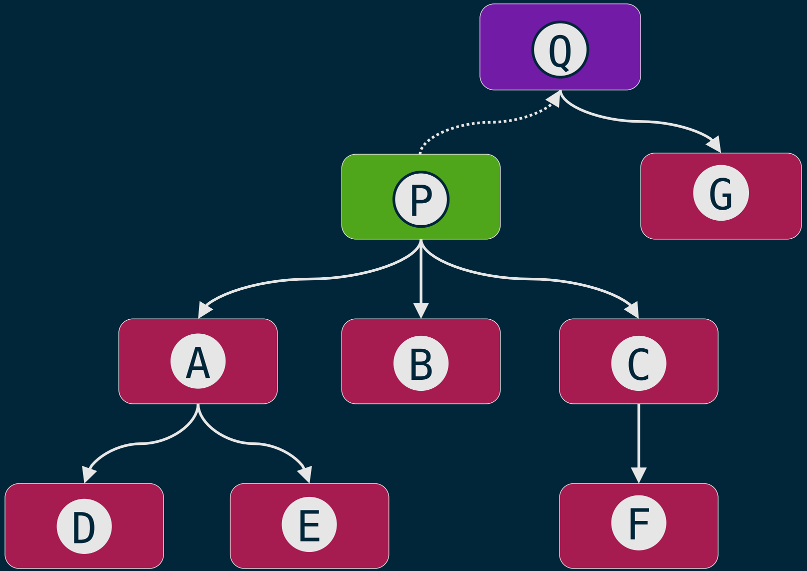


Debloated dependency tree

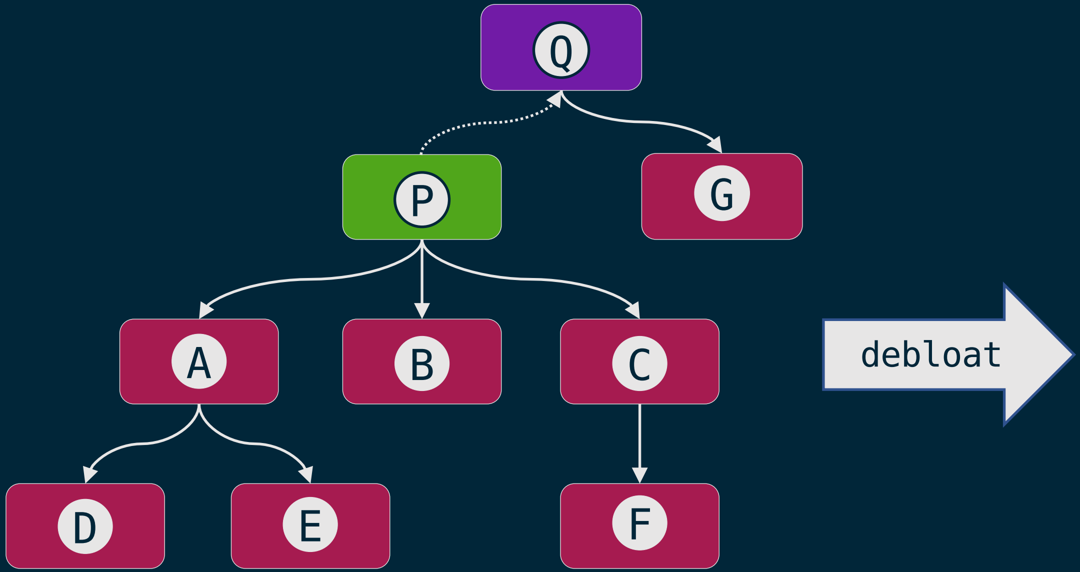


Objective

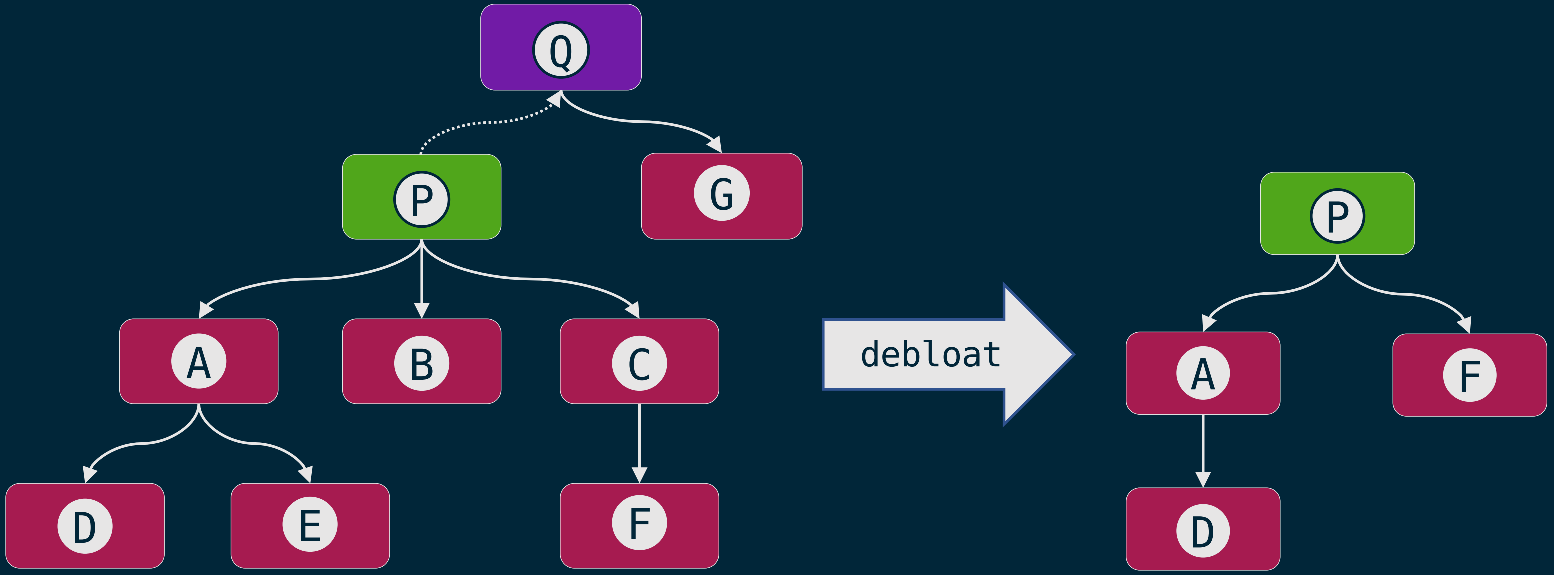
Objective



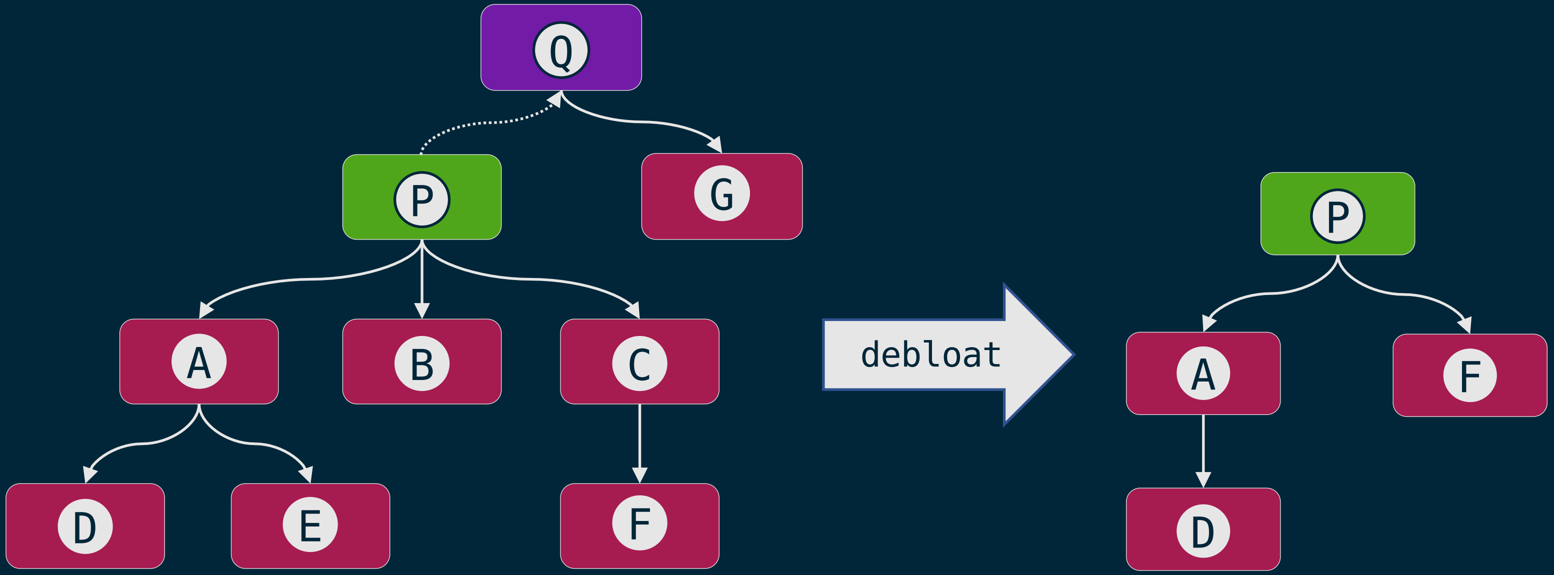
Objective



Objective



Objective



The objective of DepClean is to reduce the size of the dependency tree by debloating unused dependencies, automatically.

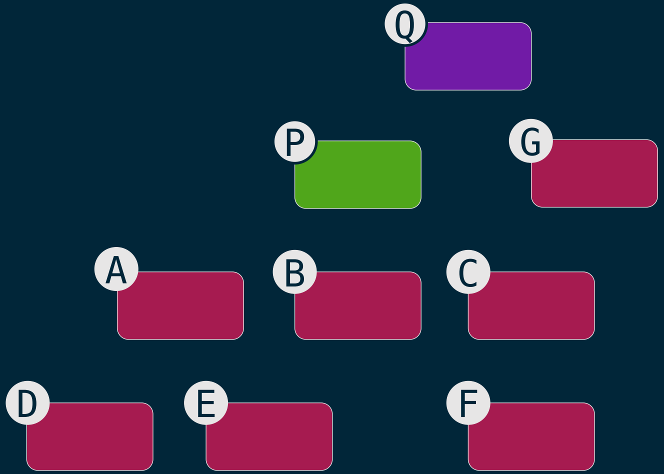
DepClean

DepClean

- Based on static analysis

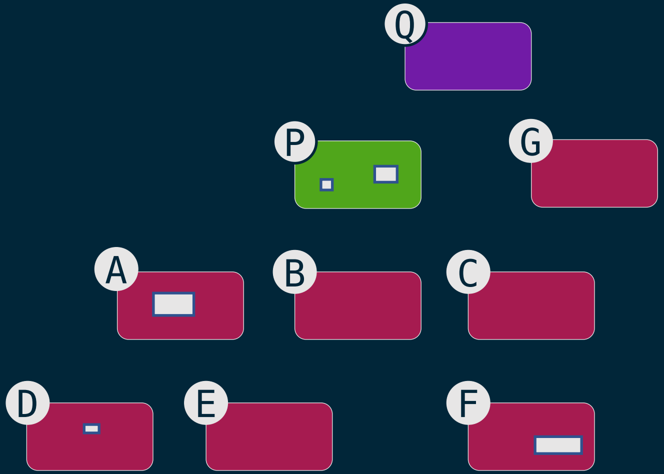
DepClean

- Based on static analysis



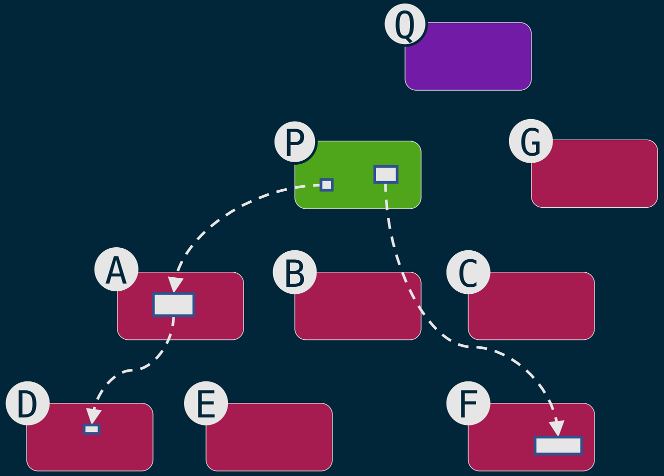
DepClean

- Based on static analysis



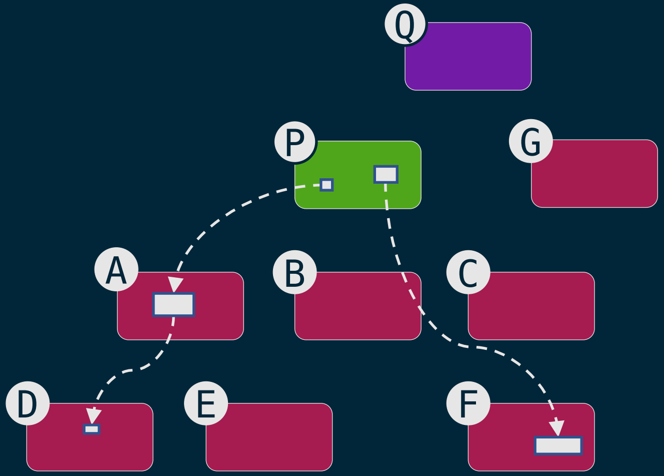
DepClean

- Based on static analysis



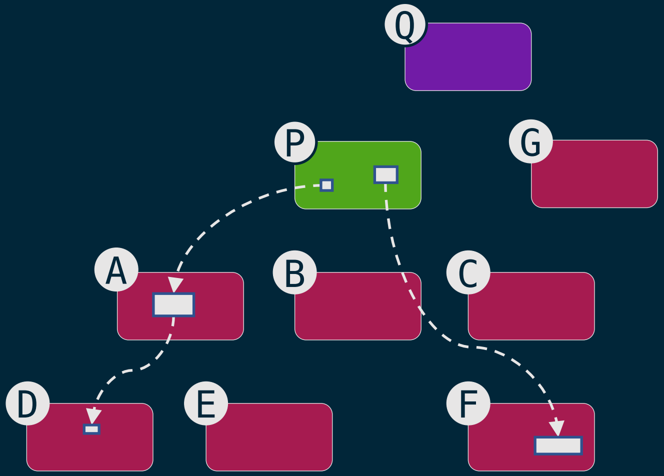
DepClean

- Based on static analysis
- Available as a Maven goal with various configurations



DepClean

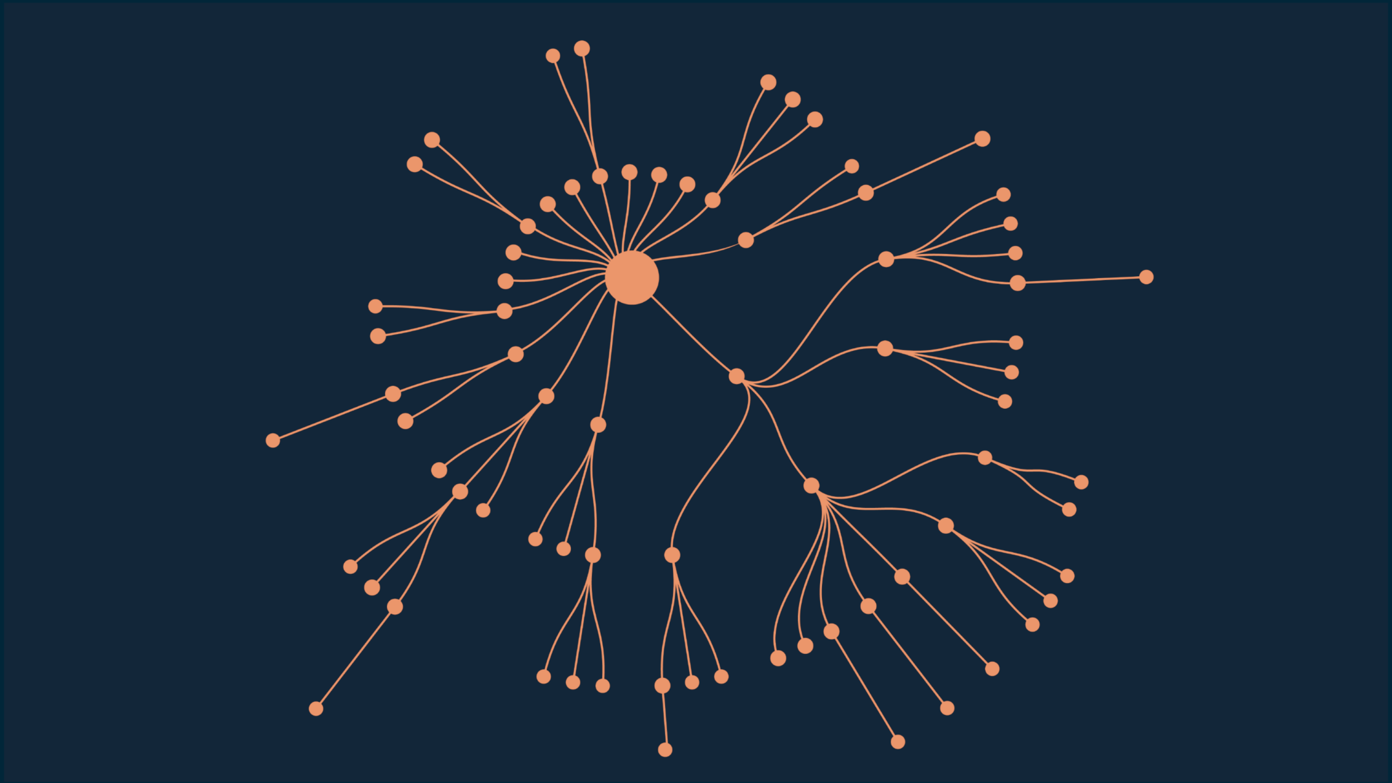
- Based on static analysis
- Available as a Maven goal with various configurations



```xml
<plugin>
    <groupId>se.kth.castor</groupId>
    <artifactId>depclean-maven-plugin</artifactId>
    <version>1.1.1</version>
    <executions>
        <execution>
            <goals>
                <goal>depclean</goal>
            </goals>
        </execution>
    </executions>
</plugin>
```

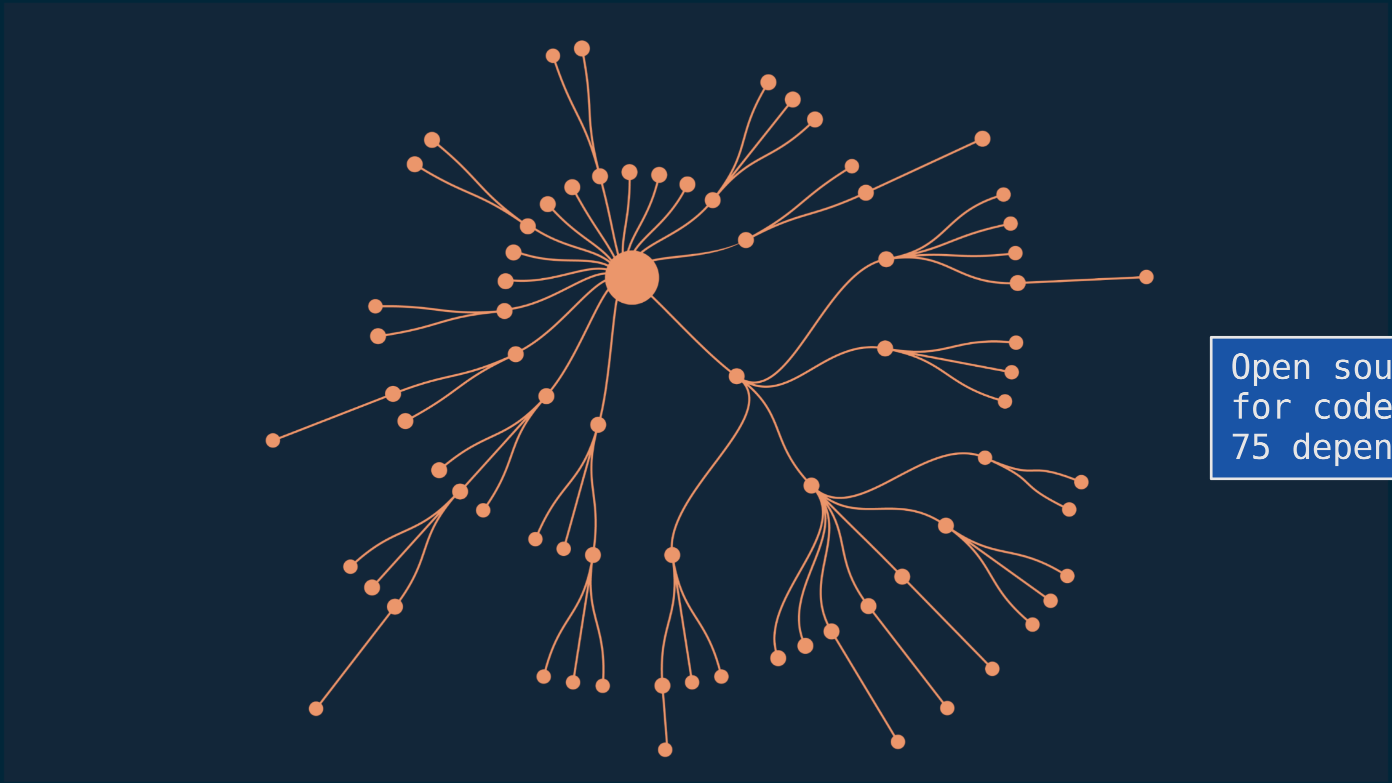
How much dependency bloat exists out there?

Example: Spoon library



https://github.com/INRIA/spoon

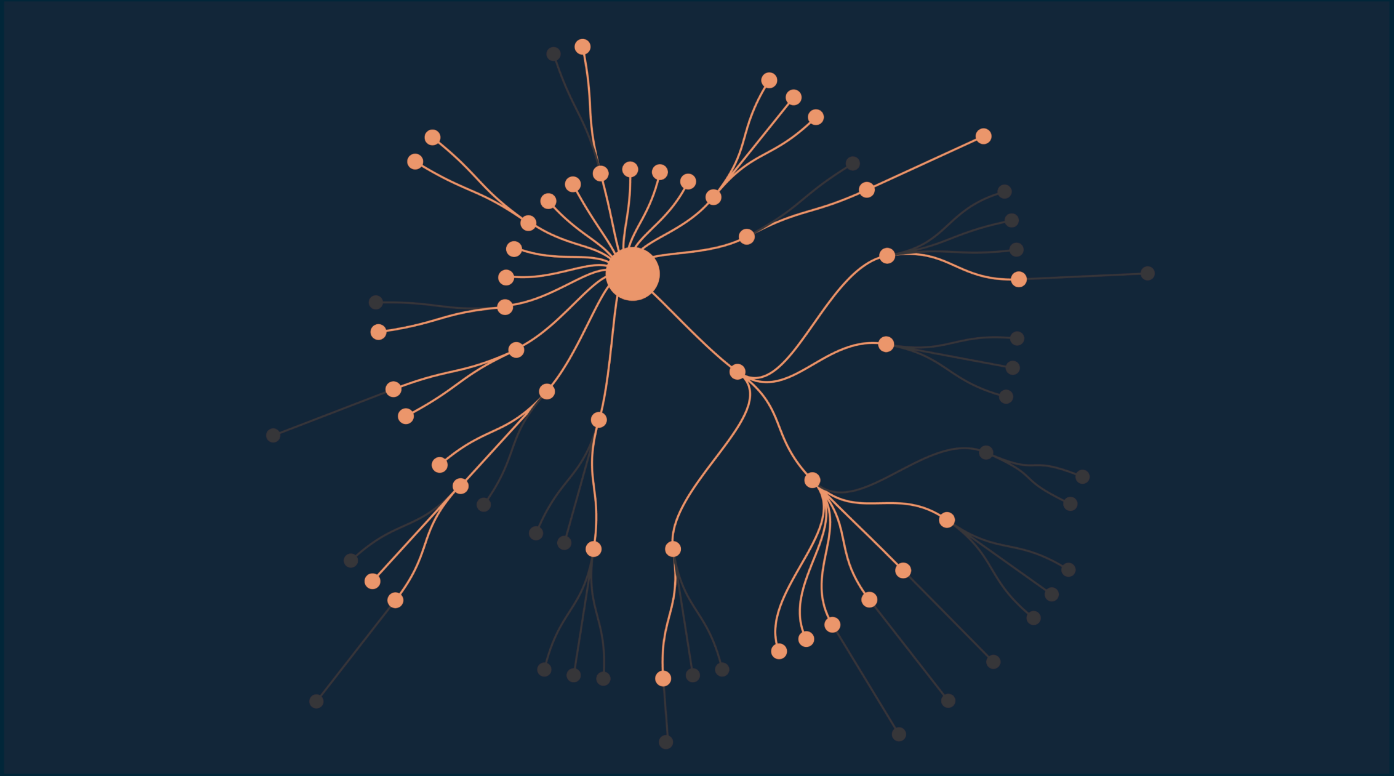
Example: Spoon library



Open source library for code analysis, 75 dependencies

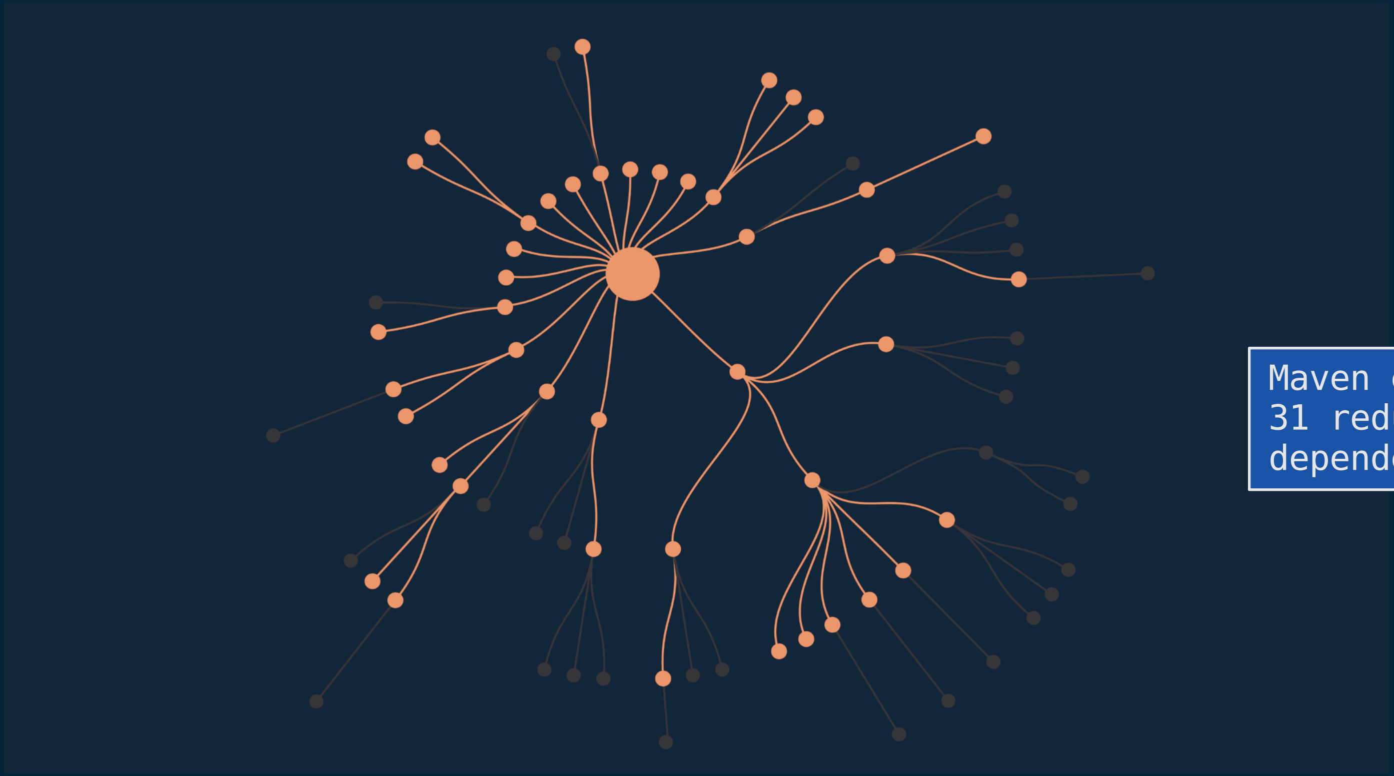
https://github.com/INRIA/spoon

Regular Maven analysis



https://github.com/INRIA/spoon

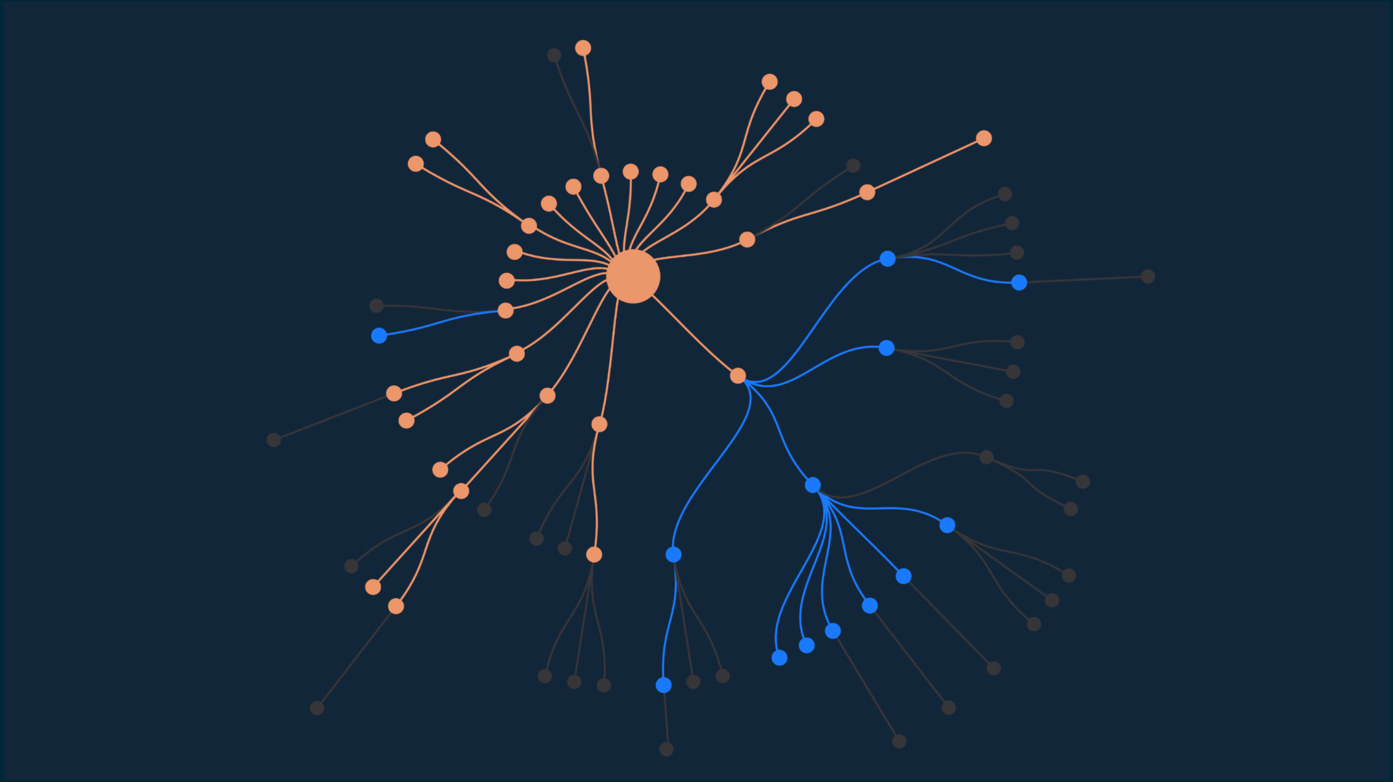
Regular Maven analysis



Maven excludes 31 redundant dependencies

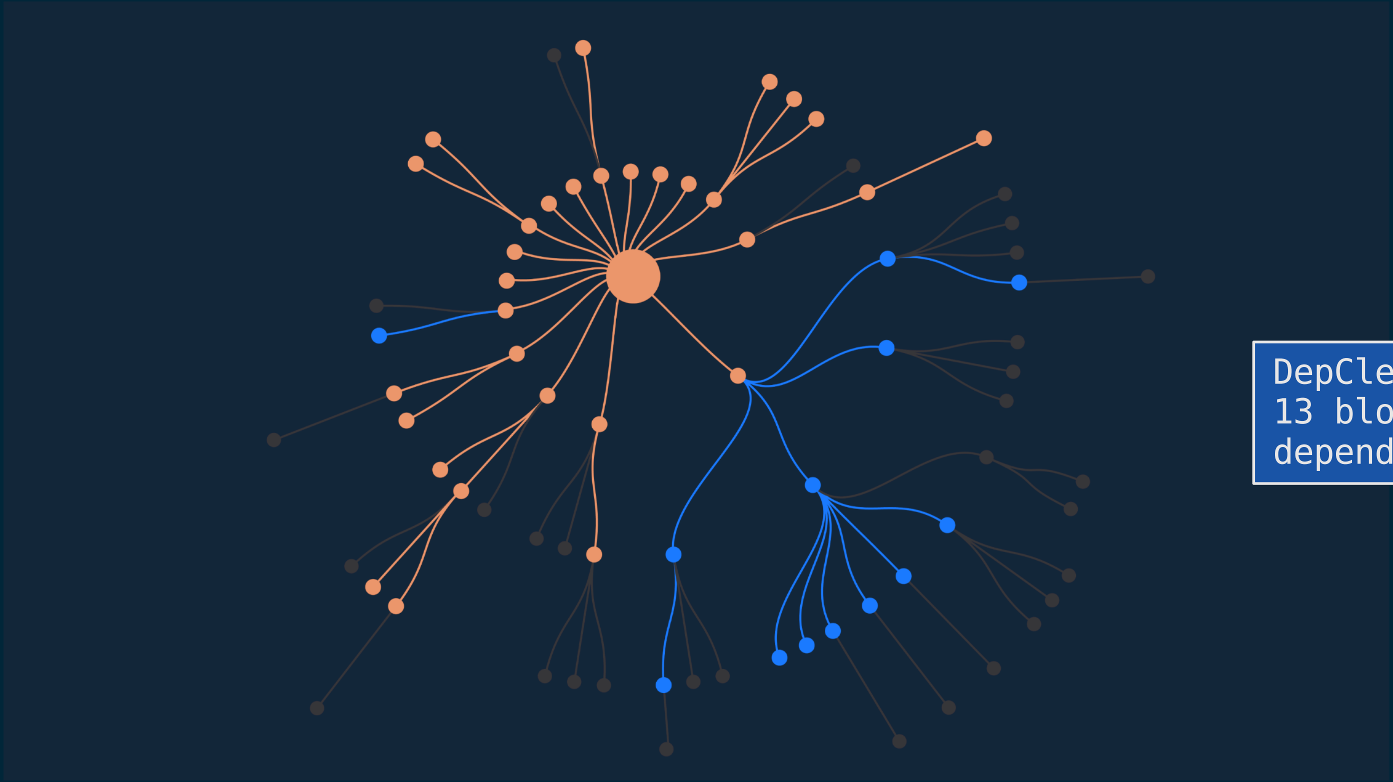
https://github.com/INRIA/spoon

DepClean novel analysis



https://github.com/INRIA/spoon

DepClean novel analysis

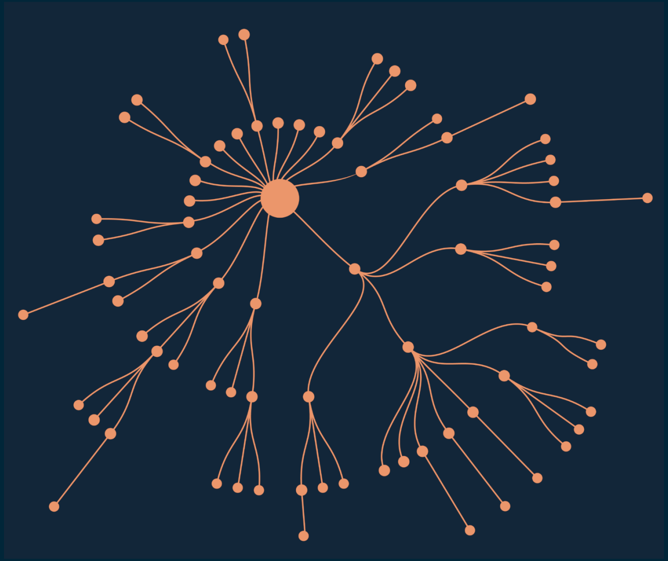


DepClean detects 13 bloated dependencies

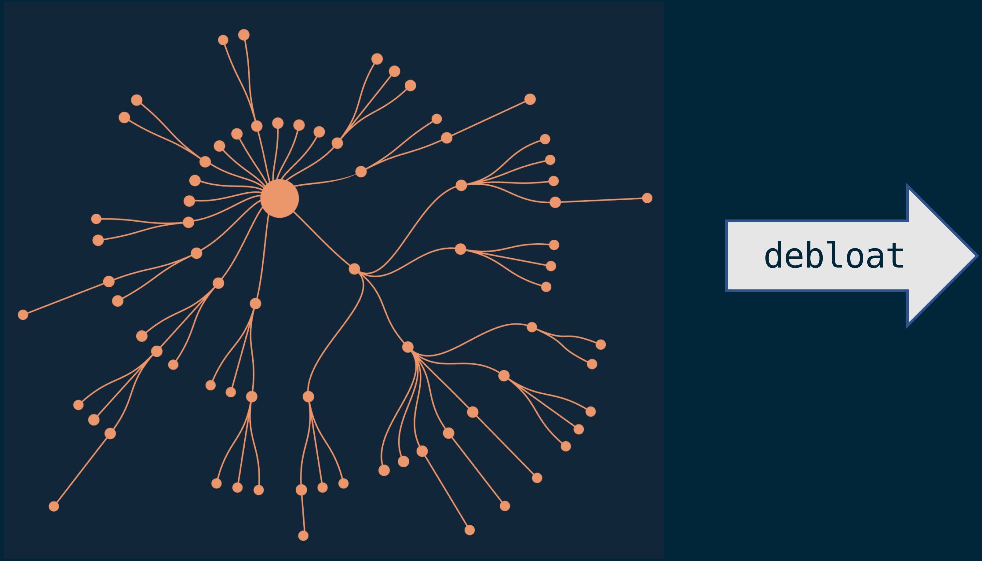
https://github.com/INRIA/spoon

Debloated Spoon library

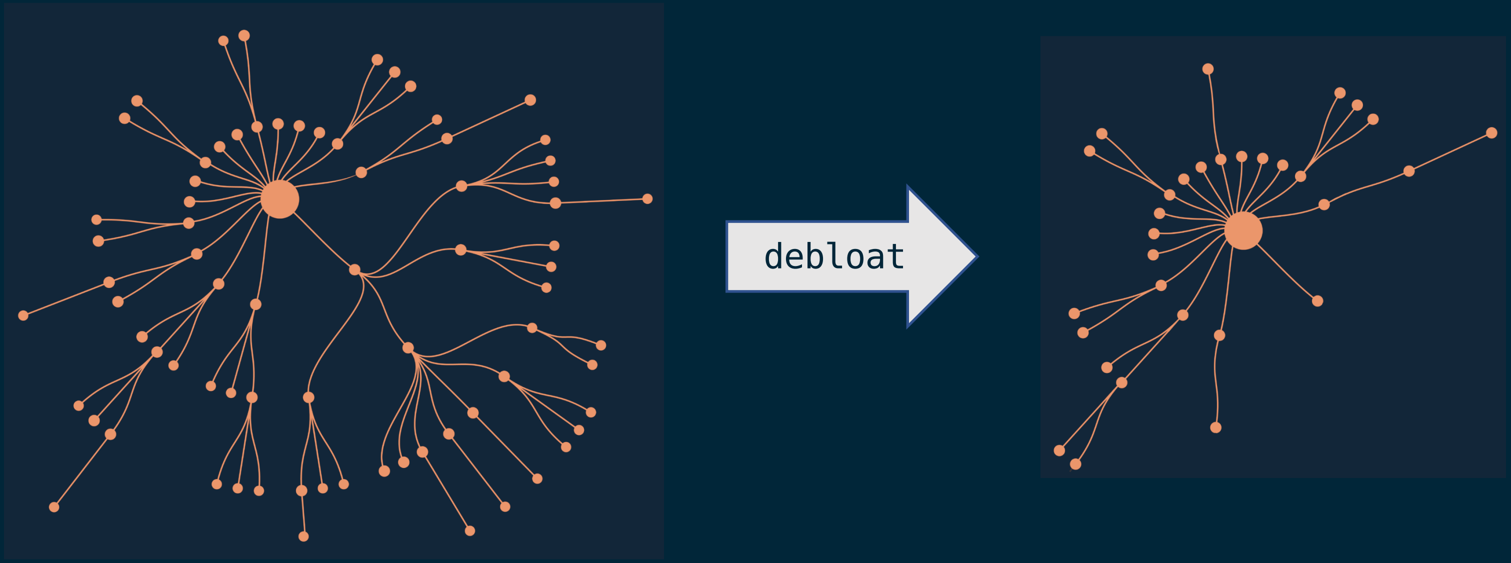
Debloated Spoon library



Debloated Spoon library

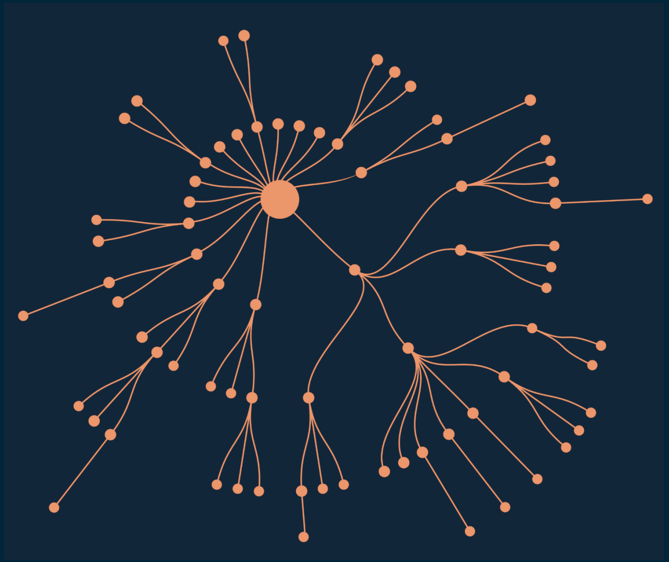


Debloated Spoon library

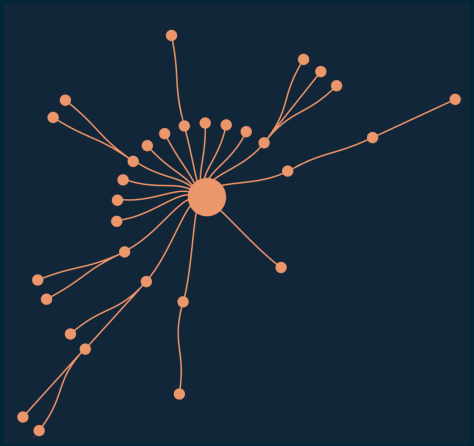


debloat

Debloated Spoon library



debloat



	JAR Size (MB)	#Classes
Before	16.2	7 425
After	12.7	5 593
Reduction (%)	27.6%	24.7%

The Maven ecosystem is big

“The Maven Dependency Graph: a Temporal Graph-based Representation of Maven Central” (published in MSR’19)

The Maven ecosystem is big

3.6 million artifacts in 2019

“The Maven Dependency Graph: a Temporal Graph-based Representation of Maven Central” (published in MSR’19)

Large scale empirical study

“A Comprehensive Study of Bloated Dependencies in the Maven Ecosystem” (accepted for publication in EMSE 2021)

Large scale empirical study

- 9K Maven artifacts

“A Comprehensive Study of Bloated Dependencies in the Maven Ecosystem” (accepted for publication in EMSE 2021)

Large scale empirical study

- 9K Maven artifacts
  - Diverse

“A Comprehensive Study of Bloated Dependencies in the Maven Ecosystem” (accepted for publication in EMSE 2021)

Large scale empirical study

- 9K Maven artifacts
  - Diverse
  - Reused

“A Comprehensive Study of Bloated Dependencies in the Maven Ecosystem” (accepted for publication in EMSE 2021)

Large scale empirical study

- 9K Maven artifacts
  - Diverse
  - Reused
  - Complex

“A Comprehensive Study of Bloated Dependencies in the Maven Ecosystem” (accepted for publication in EMSE 2021)

Large scale empirical study

- 9K Maven artifacts
  - Diverse
  - Reused
  - Complex
- 723K dependency relationships

“A Comprehensive Study of Bloated Dependencies in the Maven Ecosystem” (accepted for publication in EMSE 2021)

Large scale empirical study

- 9K Maven artifacts
  - Diverse
  - Reused
  - Complex
- 723K dependency relationships
  - 45K direct (6%)

“A Comprehensive Study of Bloated Dependencies in the Maven Ecosystem” (accepted for publication in EMSE 2021)

Large scale empirical study

- 9K Maven artifacts
  - Diverse
  - Reused
  - Complex
- 723K dependency relationships
  - 45K direct (6%)
  - 180K inherited (25%)

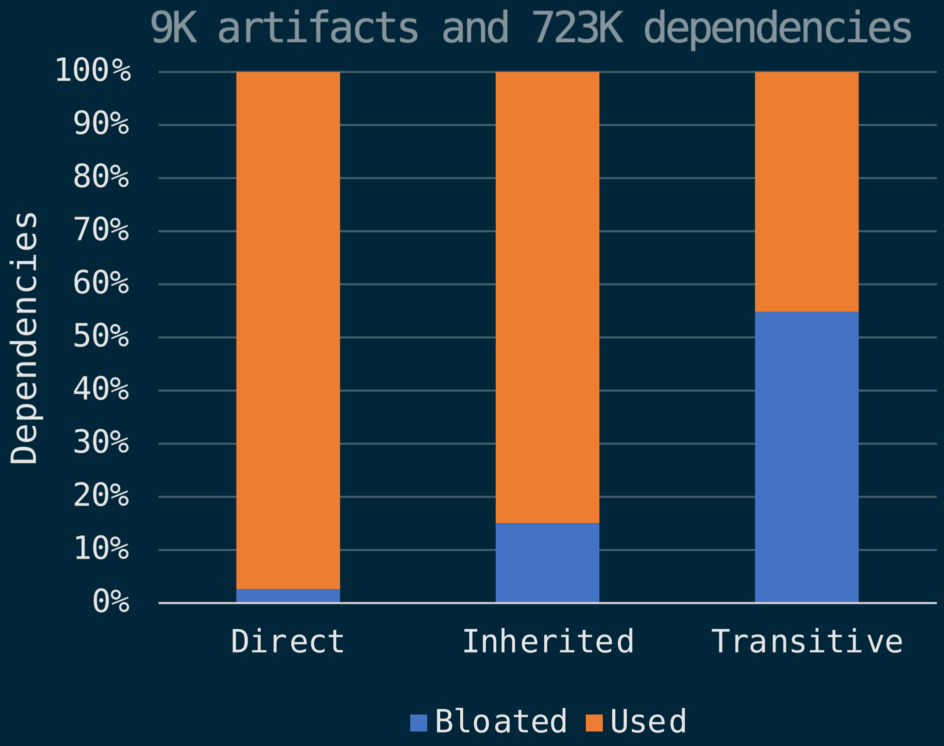
"A Comprehensive Study of Bloated Dependencies in the Maven Ecosystem" (accepted for publication in EMSE 2021)

Large scale empirical study

- 9K Maven artifacts
  - Diverse
  - Reused
  - Complex
- 723K dependency relationships
  - 45K direct (6%)
  - 180K inherited (25%)
  - 498K transitive (69%)

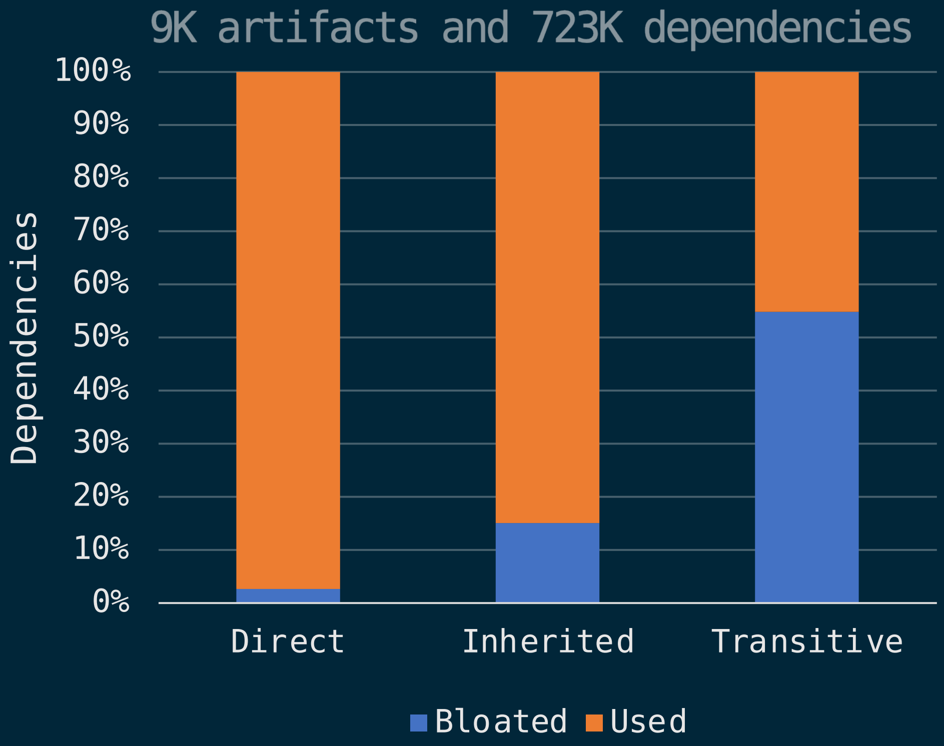
"A Comprehensive Study of Bloated Dependencies in the Maven Ecosystem" (accepted for publication in EMSE 2021)

Results



“A Comprehensive Study of Bloated Dependencies in the Maven Ecosystem” (accepted for publication in EMSE 2021)

Results



- 2.7% of direct dependencies are bloated
- 15.1% of inherited dependencies are bloated
- 57% of transitive dependencies are bloated

“A Comprehensive Study of Bloated Dependencies in the Maven Ecosystem” (accepted for publication in EMSE 2021)

How much dependency bloat exists out there?

“A Comprehensive Study of Bloated Dependencies in the Maven Ecosystem” (accepted for publication in EMSE 2021)

How much dependency bloat exists out there?

- 75% of all the dependency relationships are bloated

“A Comprehensive Study of Bloated Dependencies in the Maven Ecosystem” (accepted for publication in EMSE 2021)

How much dependency bloat exists out there?

- 75% of all the dependency relationships are bloated
- 3472 (36%) artifacts have at least one bloated direct dependency declared in the pom

“A Comprehensive Study of Bloated Dependencies in the Maven Ecosystem” (accepted for publication in EMSE 2021)

How much dependency bloat exists out there?

- 75% of all the dependency relationships are bloated
- 3472 (36%) artifacts have at least one bloated direct dependency declared in the pom
- 8305 (86%) artifacts have at least one bloated transitive dependency

“A Comprehensive Study of Bloated Dependencies in the Maven Ecosystem” (accepted for publication in EMSE 2021)

Do developers care about bloated dependencies?

User study

“A Comprehensive Study of Bloated Dependencies in the Maven Ecosystem” (accepted for publication in EMSE 2021)

User study

- 30 software projects

“A Comprehensive Study of Bloated Dependencies in the Maven Ecosystem” (accepted for publication in EMSE 2021)

User study

- 30 software projects
  - Open source

“A Comprehensive Study of Bloated Dependencies in the Maven Ecosystem” (accepted for publication in EMSE 2021)

User study

- 30 software projects
  - Open source
  - Active

“A Comprehensive Study of Bloated Dependencies in the Maven Ecosystem” (accepted for publication in EMSE 2021)

User study

- 30 software projects
  - Open source
  - Active
  - Popular

“A Comprehensive Study of Bloated Dependencies in the Maven Ecosystem” (accepted for publication in EMSE 2021)

User study

- 30 software projects
  - Open source
  - Active
  - Popular
  - Build succesfuly with Maven

“A Comprehensive Study of Bloated Dependencies in the Maven Ecosystem” (accepted for publication in EMSE 2021)

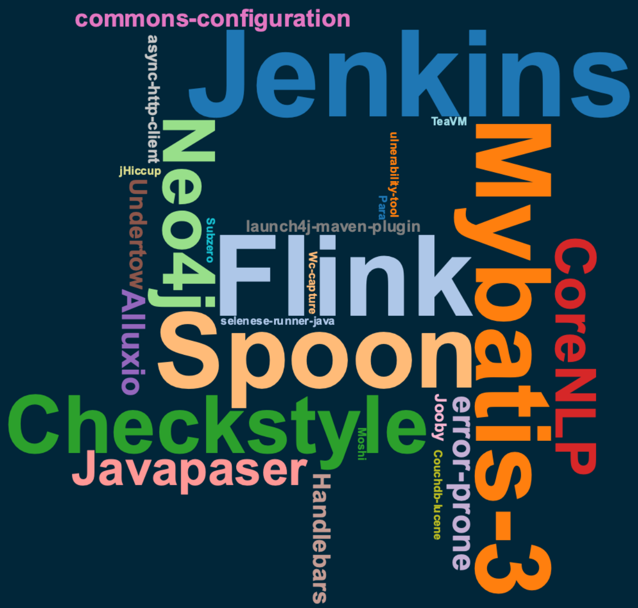
User study

- 30 software projects
  - Open source
  - Active
  - Popular
  - Build succesfuly with Maven
  - Contain dependencies

“A Comprehensive Study of Bloated Dependencies in the Maven Ecosystem” (accepted for publication in EMSE 2021)

User study

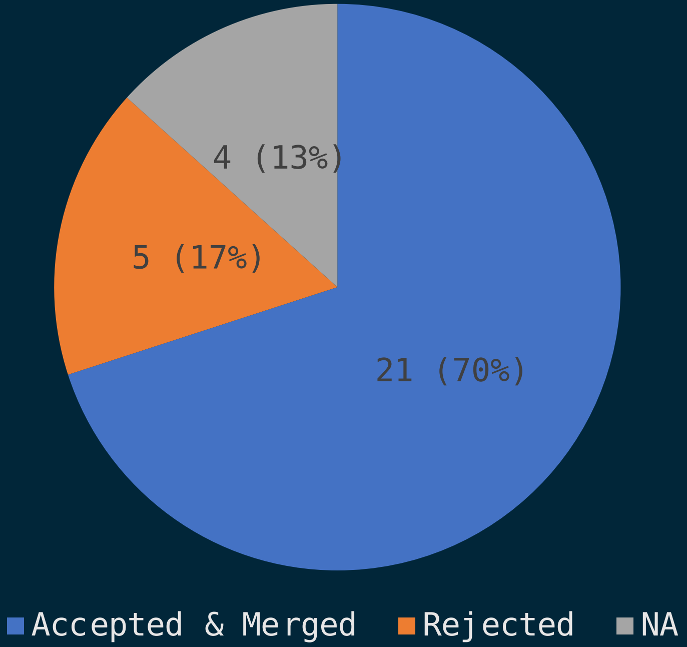
- 30 software projects
  - Open source
  - Active
  - Popular
  - Build succesfuly with Maven
  - Contain dependencies



“A Comprehensive Study of Bloated Dependencies in the Maven Ecosystem” (accepted for publication in EMSE 2021)

Results

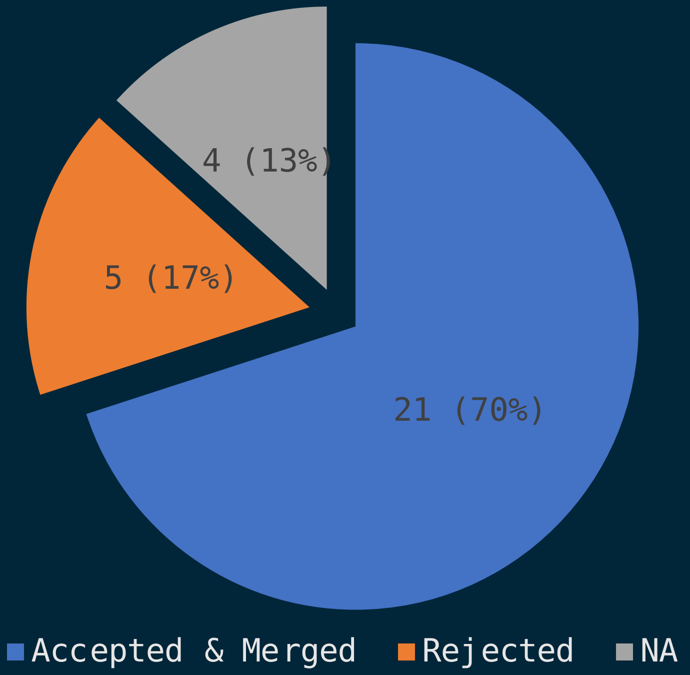
30 pull requests in 30 notable open source projects



“A Comprehensive Study of Bloated Dependencies in the Maven Ecosystem” (accepted for publication in EMSE 2021)

Results

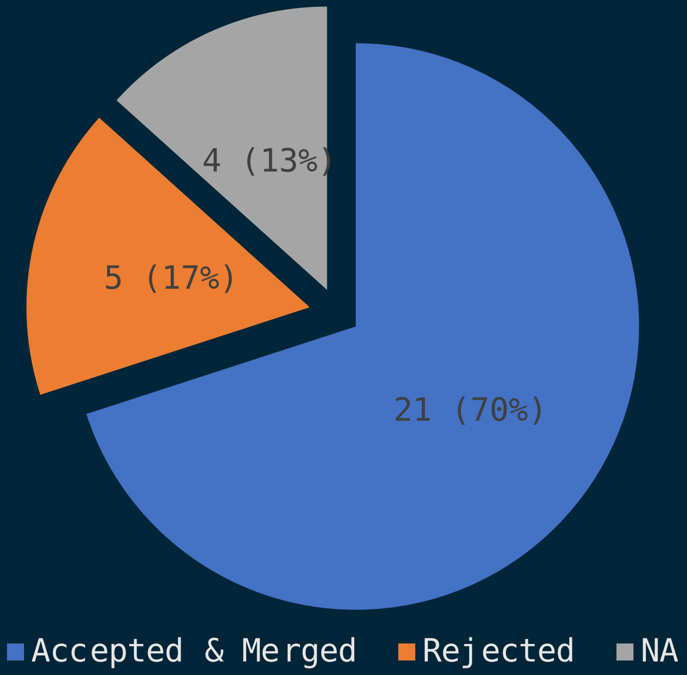
30 PRs in 30 notable open source projects



“A Comprehensive Study of Bloated Dependencies in the Maven Ecosystem” (accepted for publication in EMSE 2021)

Results

30 PRs in 30 notable open source projects



Removed 140 bloated dependencies in 21 projects thanks to DepClean

“A Comprehensive Study of Bloated Dependencies in the Maven Ecosystem” (accepted for publication in EMSE 2021)

Example: Jenkins

Bloated dependencies detected by DepClean:

Example: Jenkins

Bloated dependencies detected by DepClean:

- jenkins-core

Example: Jenkins

Bloated dependencies detected by DepClean:

- jenkins-core
  - org.jvnet.hudson:jtidy (direct)

Example: Jenkins

Bloated dependencies detected by DepClean:

- jenkins-core
  - org.jvnet.hudson:jtidy (direct)
  - org.jenkins-ci:constant-pool-scanner (transitive)
  - net.i2p.crypto:eddsa (transitive)

Example: Jenkins

Bloated dependencies detected by DepClean:

- jenkins-core
  - org.jvnet.hudson:jtidy (direct)
  - org.jenkins-ci:constant-pool-scanner (transitive)
  - net.i2p.crypto:eddsa (transitive)
- jenkins-cli

Example: Jenkins

Bloated dependencies detected by DepClean:

- jenkins-core
  - org.jvnet.hudson:jtidy (direct)
  - org.jenkins-ci:constant-pool-scanner (transitive)
  - net.i2p.crypto:eddsa (transitive)
- jenkins-cli
  - commons-codec (direct)

Example: Jenkins

Bloated dependencies detected by DepClean:

- jenkins-core
  - org.jvnet.hudson:jtidy (direct)
  - org.jenkins-ci:constant-pool-scanner (transitive)
  - net.i2p.crypto:eddsa (transitive)
- jenkins-cli
  - commons-codec (direct)

https://github.com/jenkinsci/jenkins/pull/4378

Developers’ comments

jenkins-core

Developers’ comments

jenkins-core

“Past experiences removing unused dependencies have consistently shown that some code will have depended on that inclusion and will be broken by it.”

Developers’ comments

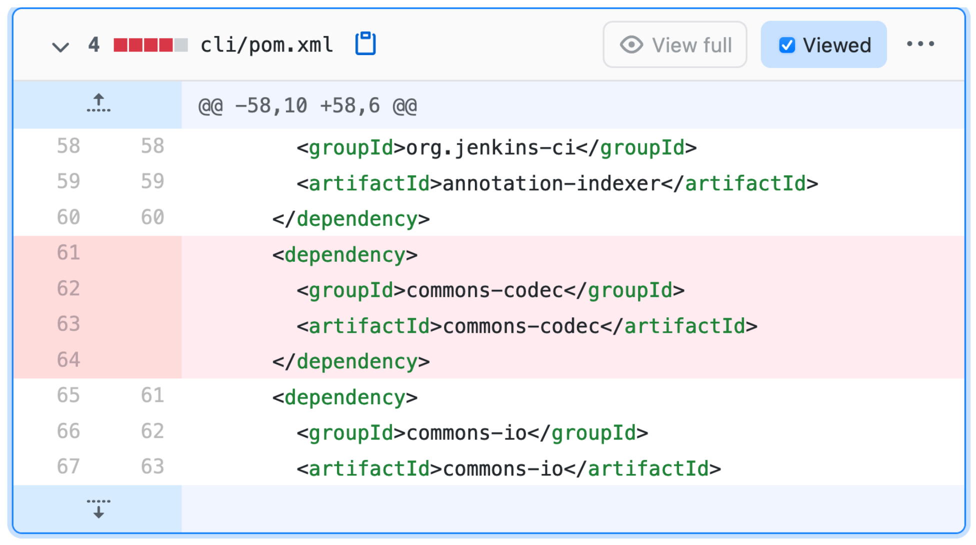
jenkins-core

“Past experiences removing unused dependencies have consistently shown that some code will have depended on that inclusion and will be broken by it.”

“We're not precluded from removing an unused dependency, but I think that the project values compatibility more than removal of unused dependencies, so we need to be careful that removal of an unused dependency does not cause a more severe problem than it solves.”

Code change

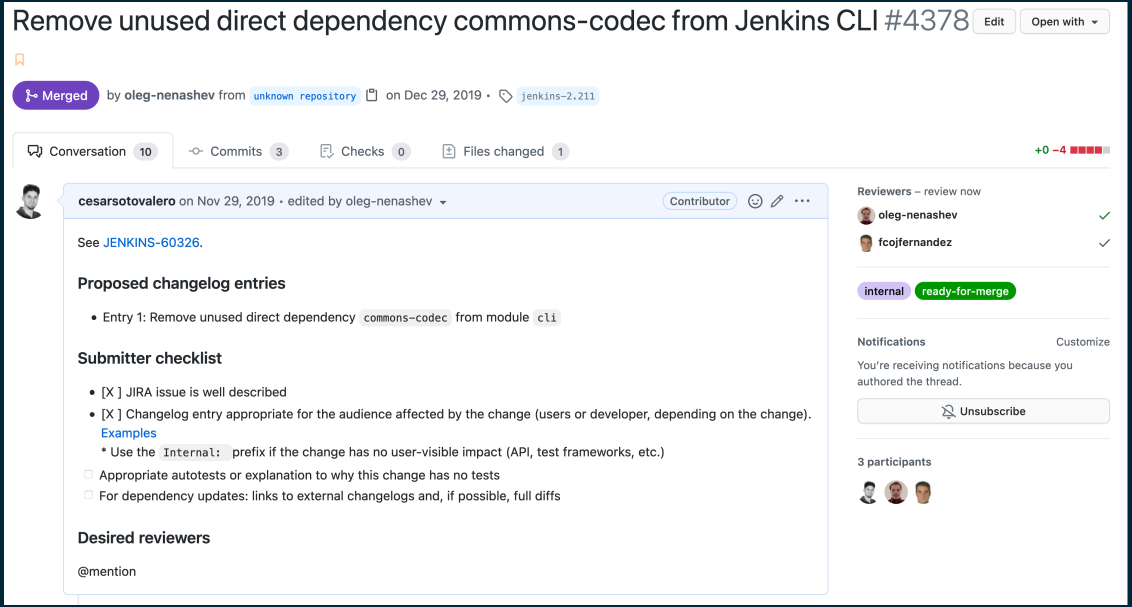
jenkins-cli



```
@@ -58,10 +58,6 @@
            <groupId>org.jenkins-ci</groupId>
            <artifactId>annotation-indexer</artifactId>
          </dependency>
          <dependency>
            <groupId>commons-codec</groupId>
            <artifactId>commons-codec</artifactId>
          </dependency>
          <dependency>
            <groupId>commons-io</groupId>
            <artifactId>commons-io</artifactId>
```

https://github.com/jenkinsci/jenkins/pull/4378

Merged pull request



Remove unused direct dependency commons-codec from Jenkins CLI #4378

Merged by **oleg-nenashev** from `unknown repository` on Dec 29, 2019 • `jenkins-2.211`

cesarsotovalero on Nov 29, 2019 • edited by oleg-nenashev

See JENKINS-60326.

Proposed changelog entries

- Entry 1: Remove unused direct dependency `commons-codec` from module `cli`

Submitter checklist

- [X] JIRA issue is well described
- [X] Changelog entry appropriate for the audience affected by the change (users or developer, depending on the change). Examples
  * Use the `Internal:` prefix if the change has no user-visible impact (API, test frameworks, etc.)

□ Appropriate autotests or explanation to why this change has no tests

□ For dependency updates: links to external changelogs and, if possible, full diffs

Desired reviewers

@mention

Reviewers – review now

- oleg-nenashev ✓
- fcojfernandez ✓

Notifications Customize

You're receiving notifications because you authored the thread.

3 participants

https://github.com/jenkinsci/jenkins/pull/4378

Conclusion

Conclusion

- There is a lot of code bloat

Conclusion

- There is a lot of code bloat
  - Caused by the induced transitive dependencies

Conclusion

- There is a lot of code bloat
  - Caused by the induced transitive dependencies
  - Caused by the heritage mechanism of multi-module projects

Conclusion

- There is a lot of code bloat
  - Caused by the induced transitive dependencies
  - Caused by the heritage mechanism of multi-module projects
  - Caused by software development practices

Conclusion

- There is a lot of code bloat
  - Caused by the induced transitive dependencies
  - Caused by the heritage mechanism of multi-module projects
  - Caused by software development practices
- Software developers care

Conclusion

- There is a lot of code bloat
  - Caused by the induced transitive dependencies
  - Caused by the heritage mechanism of multi-module projects
  - Caused by software development practices
- Software developers care
  - For security

Conclusion

- There is a lot of code bloat
  - Caused by the induced transitive dependencies
  - Caused by the heritage mechanism of multi-module projects
  - Caused by software development practices
- Software developers care
  - For security
  - For performance

Conclusion

- There is a lot of code bloat
  - Caused by the induced transitive dependencies
  - Caused by the heritage mechanism of multi-module projects
  - Caused by software development practices
- Software developers care
  - For security
  - For performance
- DepClean

Conclusion

- There is a lot of code bloat
  - Caused by the induced transitive dependencies
  - Caused by the heritage mechanism of multi-module projects
  - Caused by software development practices
- Software developers care
  - For security
  - For performance
- DepClean
  - Automatically detects and removes bloated dependencies

Demo time!

DepClean in action

Thanks!