**Dependency Metrics**

I agree with my teammate about these metrics set. These metrics set are very detailed metrics on dependencies. It provides information about cyclic dependencies and transitive dependencies of classes, as well as dependents on each class. This metric explicitly excludes infrastructure-focused dependencies like databases and operating systems, which will be developed as a distinct metric.

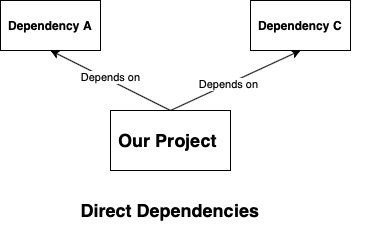
**Objectives**

The Upstream Code Dependency metric is aimed at understanding the code-based dependencies which are required to build, test, or run a piece of software. The Upstream Code Dependency metric can help identify what projects, libraries, or versions my project directly or transitively depends on.

**Metrics Overall**

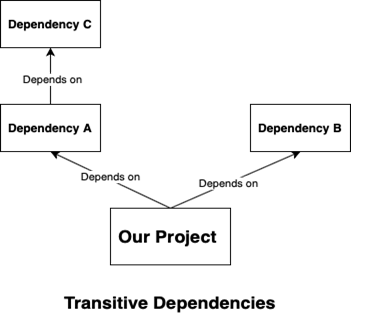
**🡪 Direct Dependencies (Dcy and Dcp)**

First-order dependencies, as declared in the source code and/or package manager configuration.



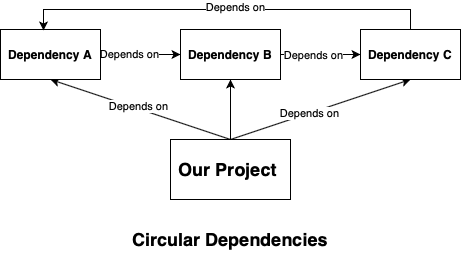
**🡪 Transitive Dependencies (Dcy\* and Dcp\*)**

Dependencies beyond first-order dependencies are also referred to as nested or second-order dependencies. For example, project A under evaluation is dependent on project B and project B is dependent on Project C. For project A, project C is a transitive dependency.



**🡪 Cyclic Dependencies (Cyclic)**

Dependencies where if traced eventually lead back to themselves. In systems that allow circular dependencies, we assume that a given dependency is only counted once in this case.



**Conclusion**

These metrics are based on the calculations of the number of immediate (or not – as transitive) subclasses subordinate to a class in the hierarchy.