**Code Metrics**

**Dependency metrics**

Dependency metrics are quantitative measures. They are used in software engineering to analyze the dependencies between various components, modules, classes, or packages within a system.

Design Patterns like Factory or Adapter can be useful for high dependencies to help with coupling (interconnectedness/interdependences between components/classes)

A screenshot of a computer

Description automatically generated

**Description:**

* **Cyclic (Number of cyclic dependency) –** are situations where there is interdependency of two or more components/modules/packages in a system that form a closed loop. Does not apply, usually to interfaces.
* **Dcy (number of dependency)** – is the count of other classes/components that a particular class/module/component depends on. Measures interconnections with other parts of the software. High dependencies lead to high coupling. Does not apply, usually to interfaces but the class that implements the interface might have dependencies.
* **Dcy\* (number of transitive dependency)** – is the count of classes/components that a particular class/module/package indirectly depends on. In other words, the class is dependent of another indirectly, through an intermediate class Does not apply, usually to interfaces but the class that implements the interface might have dependencies
* **Dpt (number of dependents)** – is the count of other classes/components that directly depend on a specific class/component/package. Denotes how many other parts of the code are dependent on the functionality provided by that class. If we consider a class that implements a certain interface, then that class is dependent on that interface.
* **Dpt\* (number of transitive dependents)** – is the count of other classes/components that indirectly depend on the specific class/component/package in question. In other words, it measures how many other parts of the code are indirectly dependent on the functionalities provided by that class. If we consider a class that implements a certain interface, then that class is dependent on that interface.
* **PDcy (number of package dependencies)** – is the count of packages/modules that the class/component/package directly depends on.
* **PDpt (number of dependent packages)** – is the count of packages/modules that depend on or have a direct dependency on the specific class/component/package in question. In other words, it measures how many other packages rely on the functionality provided by that class/component/package. This metric helps assess how widely it is used across different parts of a software.

A screenshot of a computer program

Description automatically generated

For packages we have more:

* **PDpt\* (number of transitively dependent packages)** – it’s the number of packages that are indirectly dependent on a specific package. They rely on the functionality of a certain package.

**Note:**

**Average value** – it represents the average value of the dependencies of a type ( cyclic, Dcy, PDpt, etc) in each metric (Class, Interface, Package). Having **a high average** can indicate a significant degree of interdependence between the parts of the system.

**FreeCol Dependency metrics**

**Class metrics:** A computer screen shot of a program

Description automatically generated

Classes in FreeCol have a high transitive dependency average – many classes depend indirectly on others

**Examples:**

* A screenshot of a computer

  Description automatically generated**High number of cyclic dependencies (Cyclic Collum)** – it means that multiple classes depend on each other in a circular/cyclical manner: component A depends on component B; B depends on component C and C depends on A.

* A screenshot of a computer

  Description automatically generated**High number of dependencies (Dcy Collum)** – a high number of dependencies indicates high coupling, which can be a code smell. It means that a class relies on many other parts of the code.
* A screenshot of a computer

  Description automatically generated**High number of dependent packages (PDpt Collum)** – means that the class is being used extensively by various parts of the system

**Interface metrics:** A screenshot of a computer

Description automatically generated

Interfaces in FreeCol have a high transitive dependent average; means that many other parts of the system depends indirectly on these interfaces. This is likely due to the classes that are implemented by the interfaces.

**Examples:**

* A screenshot of a computer

  Description automatically generated**High number of transitive dependencies (Dcy\* Collum)** – it means that the interface is indirectly connected to many other components.

* A screenshot of a computer

  Description automatically generated**High number of transitive dependents (Dpt\* Collum)** – indicates that the interface indirectly affects a large portion of the software system.

**Package metrics:**

A screenshot of a computer program

Description automatically generated

Packages in FreeCol have a high transitively dependent package average. In other words, many packages rely indirectly on other packages

**Examples:**

* **High number of package dependents (PDpt Collum)** – it signifies that a certain package plays a critical role in the software. In other words, many other parts of the code rely on functionalities provided by the package. Utility libraries are used very common because they have general-purposes functions/classes/methods

A screenshot of a computer

Description automatically generated

* **High number of transitively dependent packages (PDpt\* Collum)** – packages that are indirectly relied on.

A screenshot of a computer

Description automatically generated

58427 - Liliane Correia