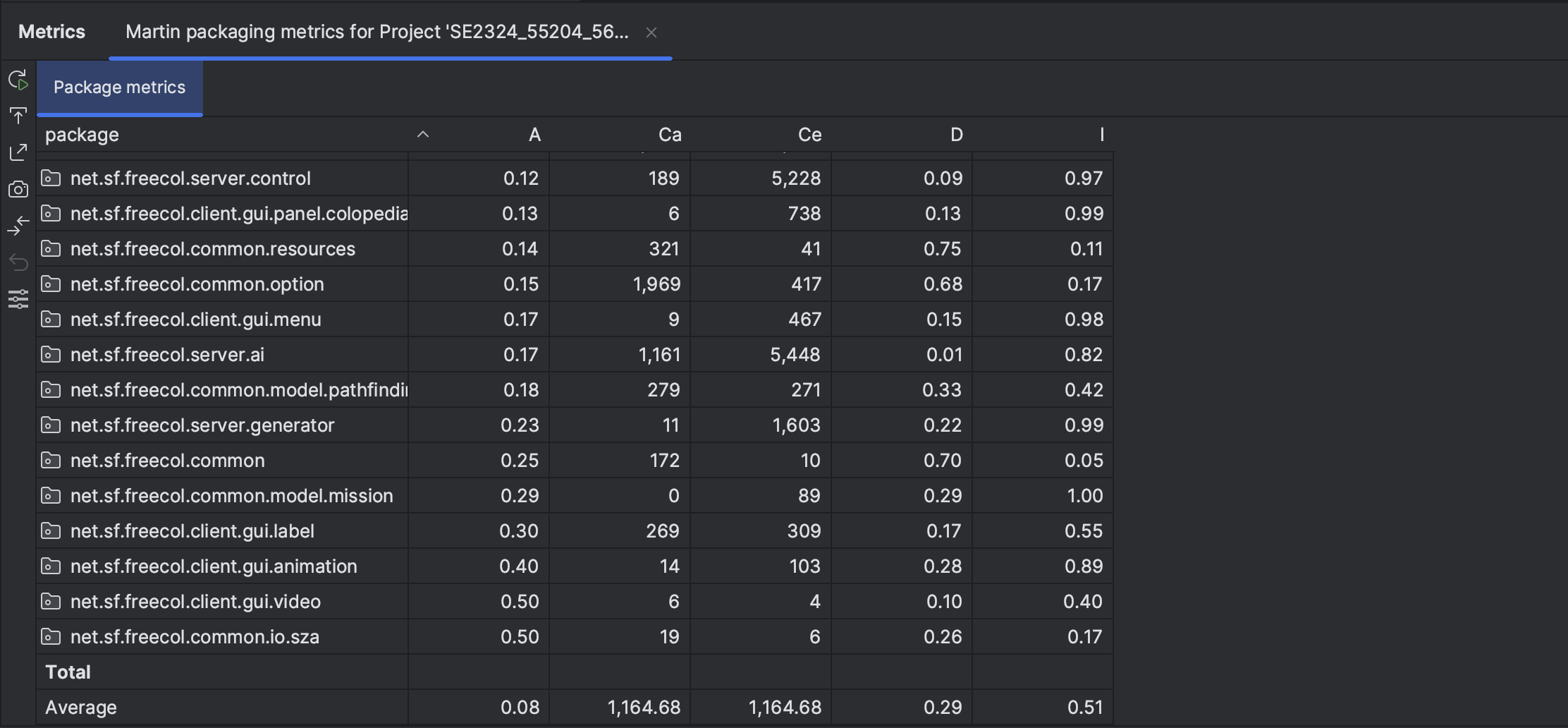
Wilker Martins -58535

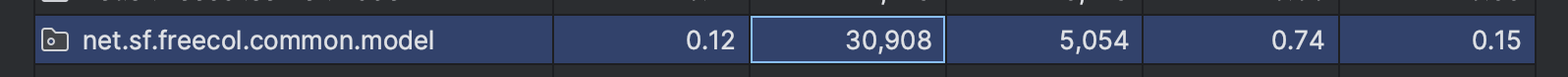
Martin Packaging Metrics

**Project metrics:**



* **A (Abstractness)**: Abstractness is a metric used in software design and architecture to measure the degree to which a module or component is abstract or high-level. Modules with high abstractness contain more general, reusable code, while those with low abstractness are more concrete and specific to a particular implementation. High abstractness is often associated with better design and maintainability.
* **Ca (Afferent Couplings)**: Afferent couplings, also known as incoming dependencies, measure the number of other modules that depend on a particular module. High afferent couplings indicate that many other modules rely on the module in question, which can lead to increased complexity and potential challenges during maintenance.
* **Ce (Efferent Couplings)**: Efferent couplings, or outgoing dependencies, measure the number of other modules that a particular module depends on. High efferent couplings suggest that a module relies on many other modules, which can make it less cohesive and more tightly coupled with the system.
* **D (Distance from the Main Sequence)**: The distance from the main sequence is a metric used to assess the balance between abstractness (A) and instability (I) in software design. It helps determine whether a module is well-placed in terms of its level of abstraction and its dependencies on other modules. Modules that are too far from the main sequence are often considered problematic in terms of design quality.
* **I (Instability)**: Instability is a metric that quantifies the balance between a module's incoming dependencies (afferent couplings, Ca) and outgoing dependencies (efferent couplings, Ce). It helps to evaluate whether a module is more stable (few dependencies and many dependents) or more volatile (many dependencies and few dependents). An ideal module should strike a balance between these two factors.

**Example of a method with high Afferent Couplings**

An elevated number of Afferent Couplings (Ca) can be a source of code smells because it signifies that a particular module or component is heavily depended upon by other modules in the codebase

**Example of a method with high Efferent Couplings**

A high number of Efferent Couplings indicates that a module is heavily dependent on other modules, potentially leading to code smells related to low cohesion, a ripple effect of changes, dependency management issues, and reduced encapsulation.

**Example of a method with high Distance from the main Sequence**

A significant distance from the main sequence suggests an imbalance between abstractness and instability, which can lead to code smells associated with overly abstract modules, fragility, complex dependencies, and violations of design principles