**NAME: PABLO BUSTILLOS**

**NRC: 7490**

**MODULARITY, ABSTRACTION, AND ENCAPSULATION AS A STEP TOWARD FOUNDATIONAL MULTI-PARADIGM SOFTWARE ENGINEERINGSOFTWARE ENGINEERING GOALS**



Software engineers aim to build quality products on time and within budget

**Some Desirable Qualities:**

* Understandability
* Testability
* Maintainability
* Efficiency
* Reliability
* Security
* Extensibility
* Openness
* Interoperability
* Reusability

**THREE CORE PRINCIPLES**

**Modularity:**

Modularity exists in a software system when it is comprised of loosely coupled and cohesive components that isolate each significant or changeable design decision in one component and ensure that related ideas are as close as possible. Modularity can improve understandability, testability maintainability, reliability, security, extensibility, and reuse. It can also help with collaboration during the software development process by outlining loosely coupled work units

**Abstraction:**

For each component, there is an explicit and clear declaration of the component’s accessible features or functionality. Depending on the paradigm and programming language, this declaration may be part of the source code, meta data, or documentation. The exposed features and functionality should be no more and no less than what other components may need or depend on. Adherence to the abstraction principle can improve understandability, testability, maintainability, and reusability. It can also allow developers to follow modularity more effectively, because it will bring to light weakness with localization of design decisions, unnecessary coupling, and low cohesion.

**Encapsulation:**

Ensure that the private implementation details of a component are insulated so they cannot be accessed or modified by other components. Doing so will lead to better testability, maintainability, and reliability. It will also help with a clear separation of concerns and avoid accidental coupling.

OVERVIEW OF THE SOLID PRINCIPLES

• SOLID is a mnemonic acronym for five principles

• **S**ingle Responsibility Principle

• **O**pen/Closed Principle

• **L**iskov Substitution Principle

• **I**nterface Segregation Principle

• **D**ependency Inversion Principle

• Some argue that these are the ”first five” principles

• That claim has not been justified or widely accepted

• However, whether they are the “first five” principles is not very important

• Following these principles can help ensure quality software, primarily from a developers’ perspective