🡪 **Requirement Analysis** 🡪 **Design** 🡪 **Implementation**

I) **Requirement Analysis**

A) Input: Vague Client Request

B) ToDo: Use Cases (With client and programmer)

1) What client would do to do each thing

2) Initiate by user ---▷ interaction between user and system ---▷ Goal

a) System is general/blackbox (Nothing specific)

3) May have variations (exceptions)\*

C) Output: Requirement Specification (Functional / Non Functional)

II) **Design**

A): Input: Requirement Specification

B) ToDo: Noun/Verb Analysis, CRC, Use Cases

1) Identify Classes/ Interfaces/Responsibilities/ Relationshsips

a) Relationships

i) IS-A (Generalization) (implements) (specific)

~ General ---> Specific Relationship

~ Inheritance (Person<Class> ◁-- Student<Class>)

~ <Interface> (Shape) Abstract type (Non-primitive) (Values and operations)

- DrawableShape<Interface> --▷ Shape<Interface>

ii) HAS-A (Instance variables) (specific)

~ Aggregation (Fluid) City ◇--- Houses

~ Composition (Lifetime) ◆**---** Airplane/Wings

- Stronger goes than weaker goes

iii) USES (Dependency)

~ A depends on B:

- A uses B (Parameter/Local Variable, Return Type)

~ Scanner class, temporary Airplane - - - -▷ Scanner

0 iv) Association (Not really sure).

b) Collaborators

i) Can collaborators do a responsibility by itself

-ie passcode in mailbox

- Identifies relationships

C) Classes

i) Name, Responsiblities, Colaborators

~Don't give a class too many responsibilities

-ie manage passcode, manage greeting, manage messages

2) Noun/Verb Analysis

C) Output: UML Diagrams/ Javadoc

1) UML

A) Class Diagram: Static relationships among classes

B) Sequence Diagram: Dynamic interactions among objec

C) State Diagram: State change of an objet

III) **Implementation**

A) Input: UML Diagram/ Javadocs

B) ToDo: Java

C) Output: PG