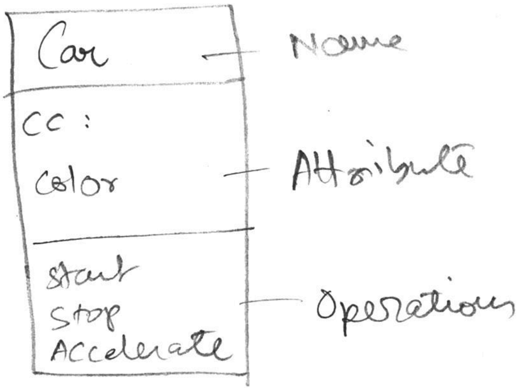
①

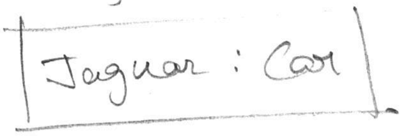


Encapsulation is a method by which Abraction is achieved

<< Stereotype >>
cannot be instantiated but only implemen

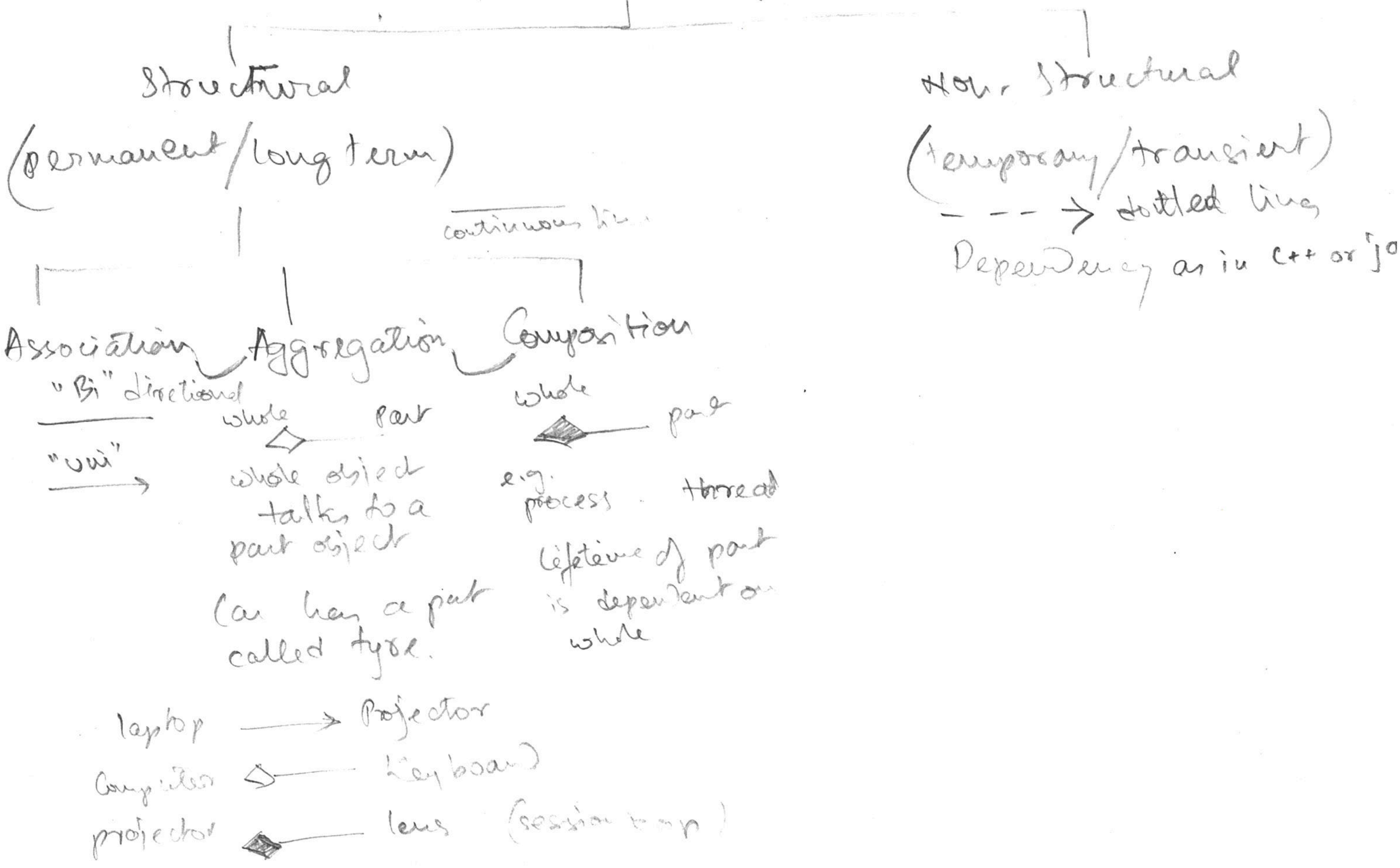
Pre-defined Stereotype
<< interface >>
<< boundary >>
<< control >>
<< entity >>
<< superInterface >> e.g. Universal Remote
User-defined.

Object : Class



subsystem is more or less same as Component
↓ Design
↓ Implementation

Relationship



Dependency as in C++ or Ja

whole object talks to a part object

Car has a part called tyre.

e.g. process thread

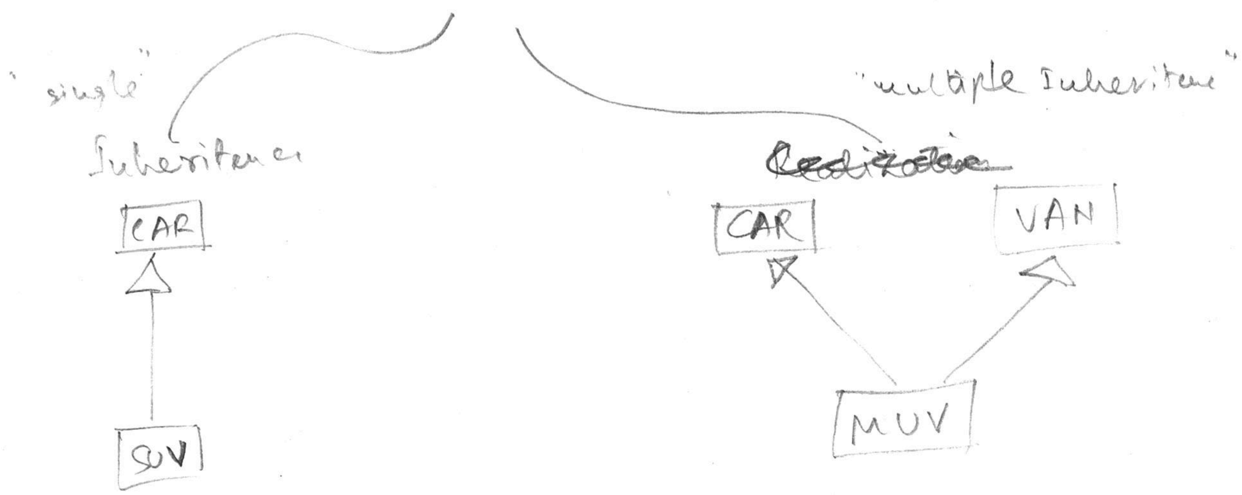
Lifetime of part is dependent on whole

laptop → Projector

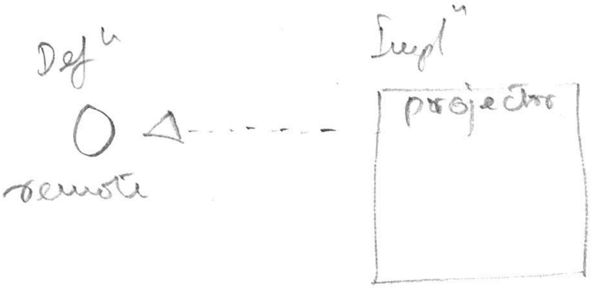
Computer ◇— Keyboard

projector ◆— lens (session temp)

Generalization

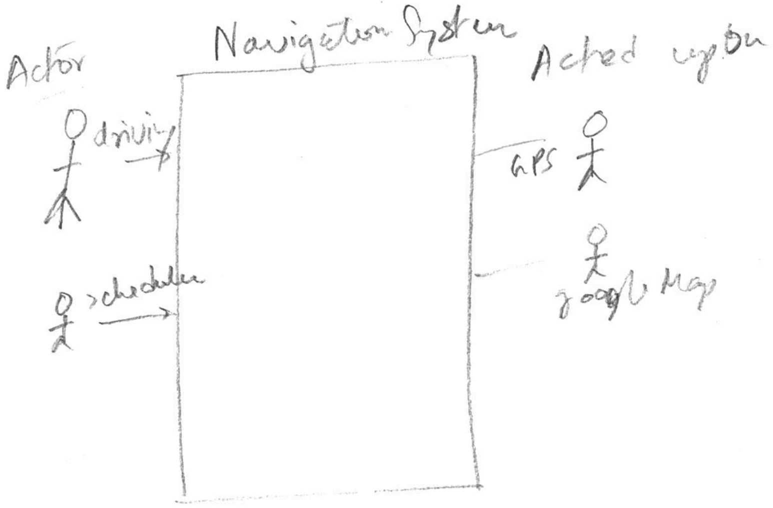


SUV has all properties of car + something more



Actor

Usecase



③ ~~Design~~ Analysis Technique

(a) Identify the Actors and the use cases

boundary condition (interfaces)

controllers

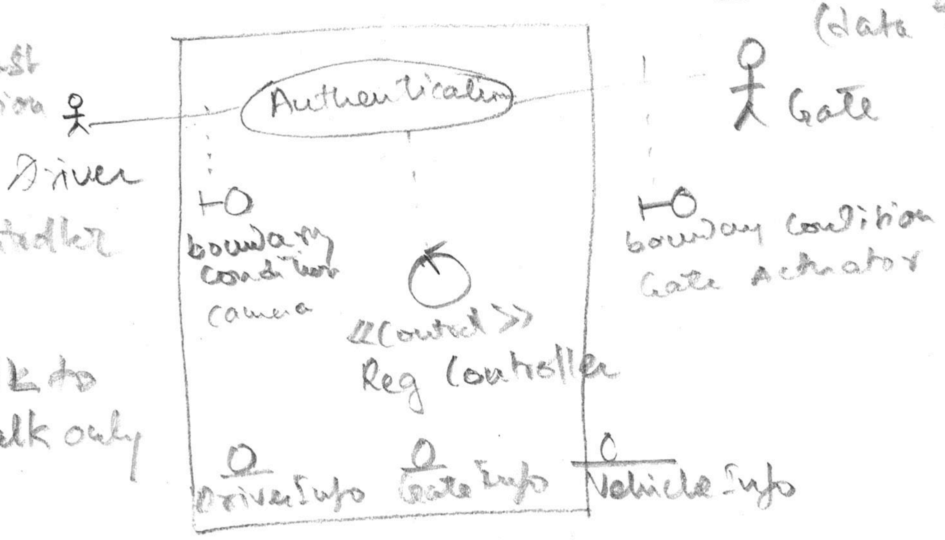
Entity (data to work on)

Gate

There will be at least one boundary condition per actor

There will be one controller per use case

entities dont talk to boundary. They talk only to control



(b) Class Diagram for Authentication Usecase

<<boundary>>
sensor

<<control>>
controller

<<entity>>

<<entity>>
License Info.

Multiplicity

V,

(c) ~~Flow~~ Sequence diagram