Software Design & Analysis

LECTURE-04

Complexity of software

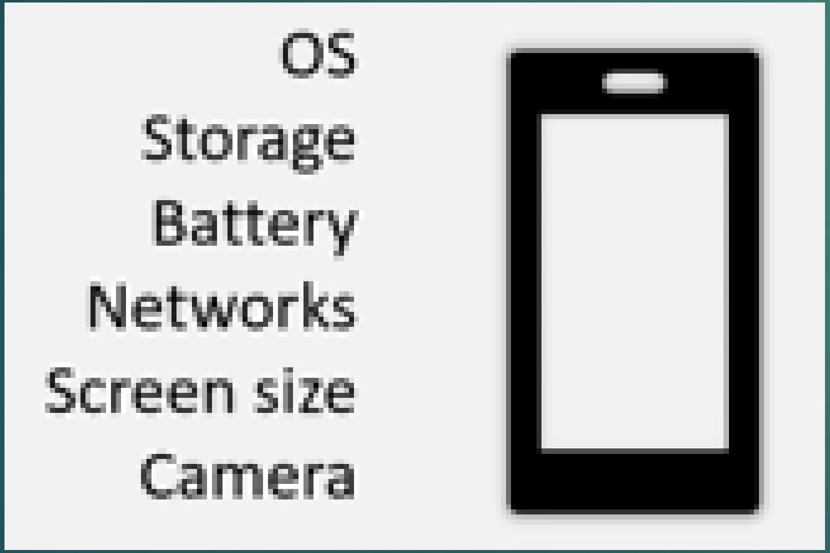
- Two ways
  - Algorithmic Decomposition
  - Object Oriented Decomposition
    - Its superior to algorithmic decomposition because
      - Its complexity is resolved through ways like abstraction, generalization etc.
      - That allow us to view the system as a set of autonomous cooperative objects

Object Oriented Decomposition

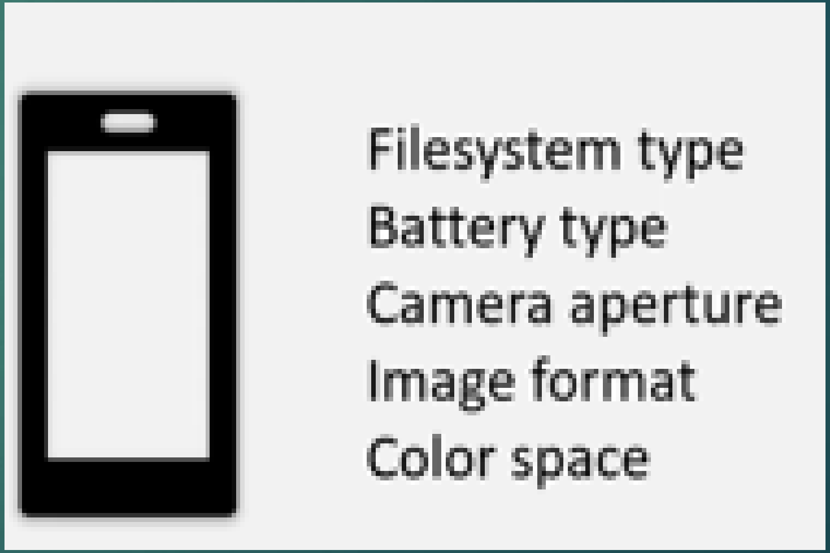
- Humans can also abstract complexity and focus only on necessary details
  - Unwanted details are ignored
  - Deal with relevant details only
  - Leads to simple entities in problem domain

For example

- When we go to buy a cellphone in market, we look at some major details



Look into these details



Ignore these details

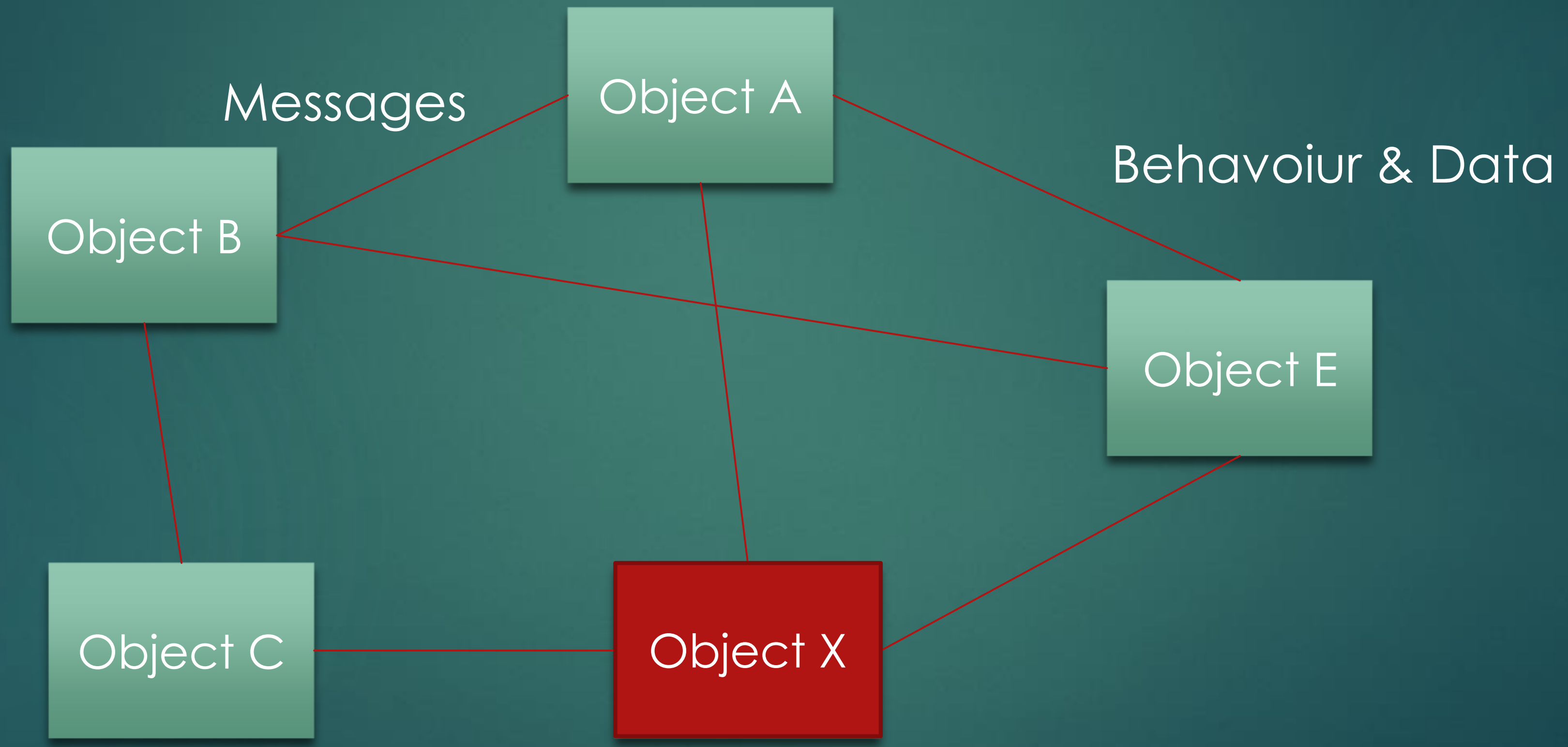
Object Oriented Decomposition

- Recognize relations between these entities
  - Some entities may be generalized
- Understand how they interact and coordinate
- The coordination give rise to the functionality of the system

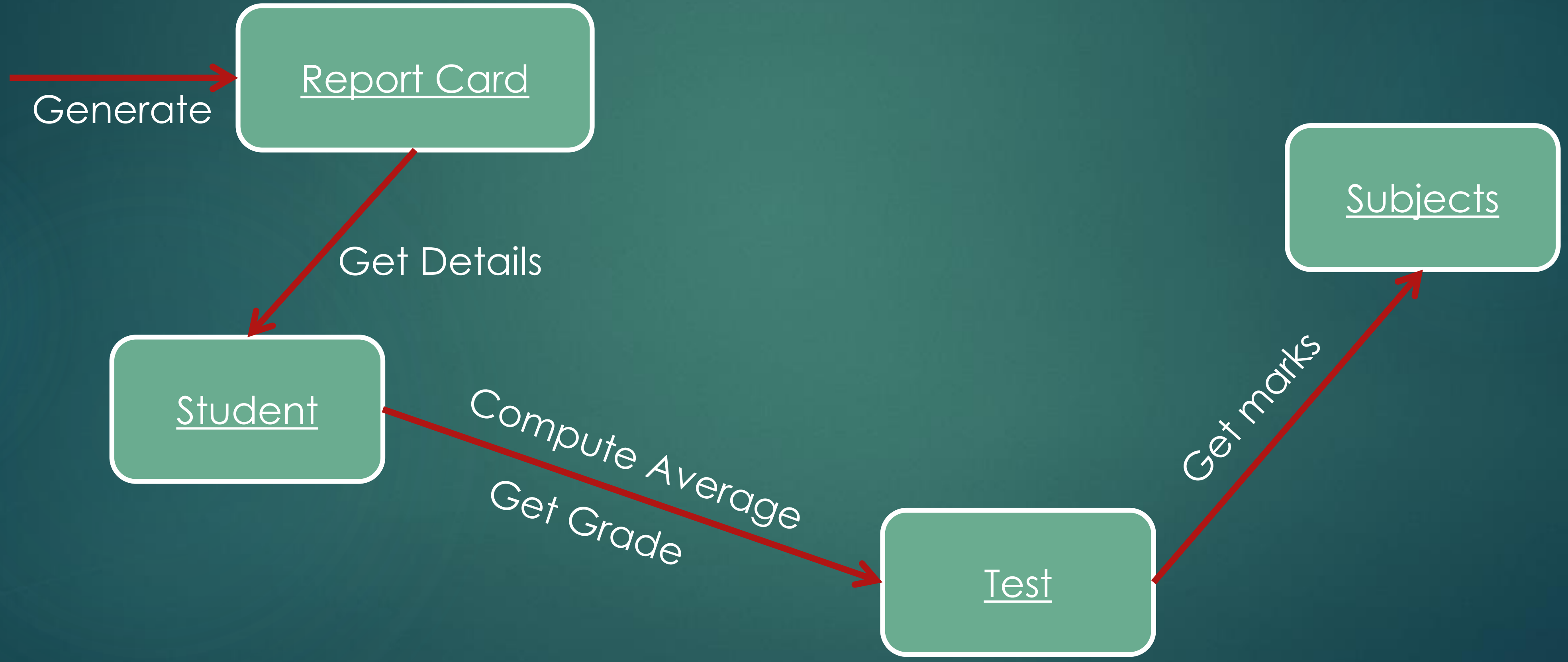
Object Oriented Decomposition

- Uses bottom-up design
- The system is decompose as a set of autonomous, but cooperative agents
- These agents are objects and represent key abstraction in the problem domain
  - Designed with details
- Each object has its own behavior
- Models some object in the real world
- The objects coordinate with each other
  - Coordination gives rise to functionality of the system
  - Send messages to each other

Object Oriented Decomposition



Object Oriented Decomposition



Advantages

- Follow separation of concerns
  - Each entity has its own responsibility
  - Addresses the complexity through organization
- System is made of objects that represent real-life entities
- Maps closely to real-world problem
- Data has high importance
  - Part of the object, not visible externally
- Enables generalization of objects
  - Promotes reuse of common functionality
  - Leads to smaller system
- Evolve incrementally over a period of time

Summary

- OO decomposition views the system as a set of autonomous objects
- Object collaborate with each other
- Data has prime importance and is hidden
- Maps closely to real-world problem domain

Object Definition

- An object represents an individual, identifiable item, unit or entity, either real or abstract, with a well defined role in the problem domain *

-Smith and Tockey

Object

- Model some part of reality
- Exists in space and time
- Can be invented as outcome of a design process
- Has well-defined behavior and a definite purpose
- Collaborate with each other objects to provide a higher-level behavior

Characteristic

- An object has the following characteristics:
  - State
  - Identity
  - Behavior
- The structure and behavior of similar objects are specified in a class
  - Object is an instance of such class
  - A class is the blueprint for objects
- Forms the building block of an application

State

- Properties and their values constitute the state of an object
- Appear as attributes of an object
  - Field, member variables , etc.
- Can be of two parts
  - Static - fundamental attributes that don’t change
  - Dynamic - attributes that change as a result of some operation performed on the project

Example- Speaker System

Static	Dynamic
Power	Volume
No. of speakers(2,2.1,4.1,etc)	treble
Connectivity	Bass
Color	
Brand	
Serial no.	

Example – Cell phone

Static	Dynamic
CPU	State (standby, in-call)
Storage Size	Remaining storage
Display Size	Battery level
Supported Bands	Signal strength
OS	
IMEI Number	

identity

- Trait that makes an object unique and gives it individuality
  - Represented through one or more attributes of the project
- Address of an object can be used to represent its identity in some cases
- Helps identify objects in a system
  - Speaker System – Serial no
  - Cellphone- IMEI no
  - Person- CNIC

Behavior

- Response of the object during interaction
- Part of the responsibility of an object
- Arises due to binding between attributes and operations
  - Operations will internally change the state of the object
  - This concludes into the behavior of the object
- Appear as operations(member function, methods)
- Examples
  - Speaker system – *ChangeVolume, ChangeBass, ChangeTreble, EqualizeLoudness*
  - Cellphone- *SendText, Dial, AcceptCall, Connect*

Examples

- Bank Account
  - State
    - (value of the attributes -> name of the account holder, account number, the balance)
  - static and dynamic attributes?
    - (static: account holder name and account number) (dynamic: balance)
  - Identity?
    - (account number)
  - Behavior
    - (the operation to be performed e.g. withdraw, deposit , check balance etc.)

More examples

- Credit card
- Employee
- File on hard drive

Summary

- Object is an entity, real or abstract, identifiable, in some problem domain
- Has the following characteristics
  - State [properties and its value]
  - Identity [uniquely identify an object in a system]
  - Behaviour [response when any operations are invoked on it]
- Their blueprint is represented by a class
- Objects do not exist in isolation
- Interdependent, coordinate