Chapter 9 : Project Management. Date 16-2 2024

1. Management Activities. (Slide 6, 7)
2. Risk management & the Risk Management Process. (8 - 21)
3. Team work & the Effectiveness of a Team (34 - 37)
4. People Management Factors. (25)

1. MANAGEMENT ACTIVITIES:

- **Project Planning:** Project managers are responsible for planning, estimating and scheduling the project and assigning tasks to the team.
- **Reporting:** Project managers are responsible for reporting progress of the project to the customer and to the managers of company developing the software.
- **Risk Management:** Project managers are responsible for the assesment of risks that may affect the project, monitor them & take necessary actions when problem arises.
- **People Management:** Project managers are responsible for choosing the right team for development, lead them, which leads to effective performance.
- **Proposal Writing:** This is the first stage of software project, involves writing a proposal to win a contract. It describes objectives of project & how it will be carried out.

2. RISK MANAGEMENT AND RISK MANAGEMENT PROCESS:

Risk Management: is concerned with identifying the risks & making plans to minimise its effect on the project.

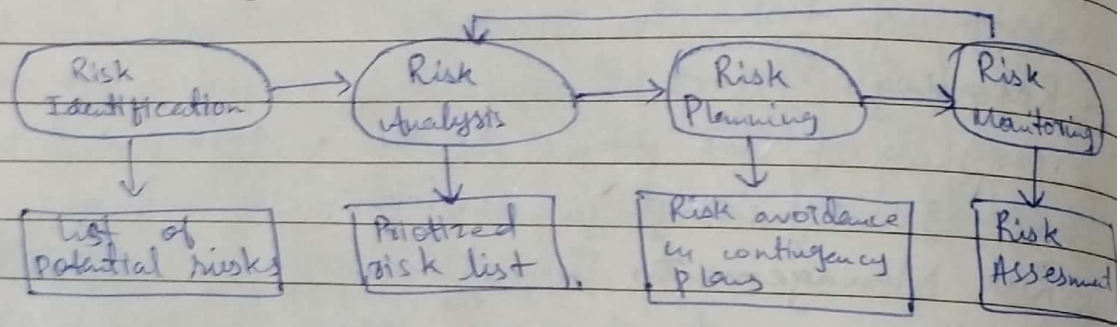
- A risk is a probability that some adverse circumstances will occur.
- A risk can affect schedule & resources & overall performance of the project.

Example of Risks:

1. Staff turnover: It affects the project, since the staff

working, will leave the project before it is finished
2. Requirements change: it affects project or product. There are more changes in requirements than expected.
3. Management change: It affects project, a change in organizational management with different priorities
4. Hardware unavailability: It affects on project, since the required hardware is not delivered timely.

Risk Management Process:



Risk Identification:

Common risks to identify in a project

① Technology risk: ① Transactions throughput less than system's database is not as expected. ② Reusable software components contains defects so they can't be reused
② People risk: ① Difficulty in recruitment of skilled staff. ② Key staff are ill when important.
③ Organizational risk: ① organization is changed, resulting in different priorities. ② Organizational financial problems results in reduced budget.
④ Tools risk: ① inefficient code generated by the tools ② Failure in integration of software tools.
⑤ Requirements risk: ① Fault in requirements gathering ② requirements change require major rework to proje
⑥ Estimation risk: ① Time required to develop soft is underestimated. ② Rate of defect repair is unde

Date 20

Risk analysis: Assessing the probability & seriousness of risk.
Probability → very low, low, moderate, high / very high.
Consequences / effects → catastrophic, serious, tolerable or insignificant

Risk types & examples:.

	Risk	Probability	Effect
1.	Organization financial problem	Low	Catastrophic
2.	Difficulty in skilled staff recruitment	High	Catastrophic.
3.	Reusable software components contain defects	Moderate	Serious.
4.	Requirements change require major rework	Moderate	Serious

Risk Planning:
For each risks, develop a strategy to mitigate the risk.
Avoidance strategies: probability to reduce the risks
Minimization strategies: impact of risk on project is reduced.
Contingency ~~strategies~~ Plans: plans to deal with risks.

	Risk	Strategy.
•	Organizational financial problems	Prepare a brief document to the senior management narrating the importance of project & how the reduced cost is affecting the effectiveness.
•	Staff recruitments problems	Alert customers about the potential difficulties & try to buy-in components.
•	Reusable components contain defects	Replace the defective components with bought-in reliable components.
•	~~Recruitment~~ changes Requirement	Apply traceability information to asses requirements change impact.

Date 20

Risk Monitoring:

- asses each identified risk regularly to identify whether it is affecting more or less.
- asses the change in effects by the risk.
- Theg key risks should be discussed at management progress meetings.
- In this phase, we can move to analysis phase to apply strategies to mitigate the risk.

3. TEAMWORK AND EFFECTIVENESS OF A TEAM:

Teamwork:

- Software engineering is a group activity and the person alone cannot complete it as per the schedule.
- Team spirit in good group is cohesive. People involved in project are motivated by the team's success & their personal goals.
- Group interaction is a key of good performance
- Flexibility in group composition, managers must do their best when composing a group.

Effectiveness of a Team:

- People in the group. There must be a balance of people in the group. Since the software development is diverse, a group may have programmers, testers etc.
- The group organization. A well organized group with the individuals contributing best of their abilities
- Technical & Managerial communications. It is essential for communication b/w team & stakeholders.

4. PEOPLE MANAGEMENT FACTORS:

1. Consistency. No favouism or discrimination b/w team members. Treatment should be consistent among everyone.

Date 20

2. Respect: Everyone must respect each other skills & abilities
3. Inclusion: Everyone's views should be count. Involve all team members.
4. Honesty: You should be honest about all the goods & bads going on in the project.

CHAPTER 8 : SOFTWARE TESTING

17-2-24

1. Software Inspections & Advantages Of inspection (12, 13)
2. Development Testing & Unit Testing (17), (18)
3. Automated Testing & Partition Testing (22), (26)
4. Interface Testing & System Testing. (35), (38)

1. SOFTWARE INSPECTIONS:

- These involve people examining the source representation (the system) with the aim of discovering defects & anomalies
- Inspection can be used before implementation, hence it doesn't require execution.
- It may be applied to any representation of the system such as requirements, design, test data etc.
- They have been shown to be an effective technique for discovering program errors.

Advantages of Inspection:

- During testing errors can mask (hide) other errors. As inspection is static process, no concern with errors interactions
- Incomplete versions of the system can be ~~tested~~ inspected without additional cost. You need to develop special tests to test the available parts of the system.
- With anomalies in the program, inspection can also search for quality attributes such as compliance with standards portability & maintainability.

Date ____ 20

2. Development Testing:

- Development testing includes all testing activities that are carried out be the development team.
- Unit testing, where individual ~~components~~ program units or objects classes are tested. It focuses on testing the functionality of objects or methods.
- Component testing: Several individual units are integrated and tested as a component. It should focus on testing component interfaces
- System testing: some or all components are integrated and tested as a whole. It should focus on testing component interactions.

Unit Testing:

- Testing individual components in isolation.
- A defect testing process.
- A unit may be individual functions or methods within an object.
- A unit may be object classes with several attributes or methods.
- A unit may be composite components with defined interfaces used to access their functionality.

3. Automated Testing:

- Unit testing is automated using a program so that tests are run or checked without manual intervention.
- A test automation framework (such as JUnit) to write and run ~~test~~ program tests.
- Unit testing frameworks provide generic test classes that can be extended to make specific tests
- The tests can be run or can be reported using some GUI, on the success of the results.

Date ____ 20 ____
M T W T F S S

PARTITION TESTING:

- Input data & output results are often fall into different classes where all members of a class are related.
- Each of these classes make an equivalence partition or domain where the program behaves in an equivalent way for each class member.
- Test cases should be chosen from each partition
- Equivalence partition is, for inputs like we have divided number of input values equally corresponding the output values.

Less than 4	Between 4 & 10	more Than 10

4. INTERFACE TESTING:

- The objective in interface testing is to detect faults due to interface errors or invalid assumptions about interfaces.
- Interface types:
  1. Parameter interfaces: data passed from one procedure to another.
  2. Shared memory interfaces: Block of memory is shared blw procedures or functions
  3. Procedural interfaces: Subsystems encapsulates a set of procedures to be called by other subsystems.
  4. Message-passing interfaces: Subsystems request services from other subsystems.
- Design tests for parameters to a called procedure to test ranges of inputs & for null values
- Use stress testing in message passing interfaces.

SYSTEM TESTING:

- System testing during development involves integrating components to create a system version & testing it
- The focus is to test the interaction b/w components.

Date 20

- System testing checks the compatibility of the components & they are interacting correctly & right data is transfering across the interfaces.
- System testing test the emergent behaviour (when it'll be in production) of a system.

CHAPTER 7 DESIGN AND IMPLEMENTATION

(15-16)

1. Object Class identification & Approaches to identification
2. Design models & examples of design models. (20,12)
3. Development platform tools (46)
4. Configuration management & configuration management activities. (43, 44).

1. OBJECT CLASS IDENTIFICATION & APPROACHES TO IDENTIFICATION

Object Class Identification:

- Identifying object classes is often considered a difficult part in object oriented design.
- There is no magic formula for identification. It requires skills & experience and domain knowledge of system designers.
- It is an iterative process. You are unlikely to get it right first time.

Approaches to Identification:

- Grammatical approach based on natural language description of system (Hood OOD method).
- Identify the objects based on tangible things in application domain.
- Behavioural approach: identify objects based on their behaviours.
- Scenario-based analysis. Identify objects, attributes

Date ____ 20____
M T W T F S S

of methods in a scenario.

2. DESIGN MODELS AND EXAMPES OF DESIGN MODELS.

Design models:

- Design models show the objects, its classes of the relationship b/w these entities.
- Static models describe static nature of the system in terms of objects classes & relationships.
- Dynamic models describe the dynamic interactions b/w objects.

Examples of Design Models:

- Subsystem models that show logical groupings of objects into coherent subsystems.
- Sequence models that shows the sequence of object interactions.
- State machine models that shows how individual objects changes their state in response to events.
- There are more models e.g. use-case models, aggregation models, generalisation models etc.

3. DEVELOPMENT PLATFORM TOOLS: A software development platform should provide.

- An integrated compiler & syntax-directed editing system that allows to create, edit & compile code.
- A language debugging system
- Graphical editing tools, such as for UML models.
- Testing tools, such as JUnit, that is an automated testing tool, to automatically run tests on system's versions.

4. CON

4. Configuration Management and its Activities

Configuration management:

- Crucial process of managing changing software system.
- Aim of configuration management is to support system integration process, in order to enable developers to access code & documents in a controlled way & can find out all the changes to update the system.

Configuration management activities:

- Version management: to provide support to keep track of different versions of ~~systems~~ components.
- System integration: to provide support to developers to help them define what components of a version are used to create each version.
- Problem tracking: to provide support to users to report bugs & problems. Developers to see who is working to fix the problems.

Chapter 6 Architectural Design (4,11,12)

1. Architectural design & architectural design decisions
2. Architecture & System characteristics (14)
3. Architecture views (15)
4. Use of application architectures. (37)

1 Architectural Design:

- Design process for identifying the subsystems making up a system & framework for the subsystem control & communication is architectural design.
- An early stage of system design process
- Represent the link b/w specification & design processes.
- Often carried out in parallel with specification activities.
- Identifying major system components & communications.

Date ____ 20 ____

Architectural design decisions.

- Architectual design is a creative process so it differs depending on type of system.
- There are number of common decision for all systems & they have an impact on non-functional characteristics of the system.

1. Is there a generic application architecture that can be reused?
2. What architectural styles are appropriate?
3. How will the system be decomposed into modules?
4. How should the architecture be documented?

2. ARCHITECTURE & SYSTEM CHARACTERISTICS.

1. Performance: localise critical operations & minimise communication. and use large components.
2. Security: Use a layered architecture & encapsulate critical assets in inner layers.
3. Availability: Include redundant components & mechanisms for fault tolerance.
4. Maintainability: Use fine grain, replaceable components.

3. ARCHITECTURAL VIEWS:

- What views or prespectives are useful when designing & documenting a system's architecture?
- What notations should be used for describing architectual models?
- Each architectual model show one view of architecture.
- For both design & documentation, you usually need to to present multiple views of software architecture.

Date

4. USE OF APPLICATION ARCHITECTURES:
Models of an application architecture can be used
- as a starting point for the architectural design p
- as a design checklist
- as a way of organising the work for developm
- as a means of assessing components for reuse

CHAPTER 4 REQUIREMENTS ENGINEERING

1. What is a Requirement & Types of Requirements (4,16)
2. Metrics for specifying non-functional requirements (2
3. The structure of a requirements document. (31, 32)
4. Form-based specifications & Tabular Specifications (43,

1. REQUIREMENT:

- It may range from a high-level abstract statement o a service or a system constraint or a detailed mathematical functional specification
- Requirements may serve a dual function, may be the basis for bid for a contract or the contract itself, both of them can be requirements

Types of requirements:

① User requirements: Statements in natural language & the diagrams of the services to be provided & its operational constraints, written for customers

② System requirements: A well structured & detailed description of system functions, services & operational constraints. It defines what should be implemented

2. METRICS FOR SPECIFYING NON-FUNCTIONAL REQUIREMENTS

Date ____ 20__
M T W T F S S

②	Size	MegaByte, No. of ROM chips.
①	Ease of use	Training Time, no. of help frames.
③	Robustness (strength)	Time to restart after failure, % of events causing failure,

3. The Structure Of A Requirements Document

Chapter	Description
1. Introduction	It should describe the need for the system, functions & explain how it works with other systems. It should also describe the value it adds to the business.
2. Glossary	Define the technical terms used in the document.
3. User Requirements Definition	Describe the services provided to the user. Non-functional user requirements. Use natural language & diagrams to make it understandable to customers. Specify the standards of the product.
4. System Requirements Specifications	Describe functional & non-functional requirements in more detail. Interfaces for the system may be defined.

4. Form-Based And Tabular Specifications.

Form-based specifications:

- Definition of the function or entity.
- Description of inputs where they come from & outputs where they go to.
- Pre & post conditions of the function (if appropriate)
- The side effects (if any) of the function.

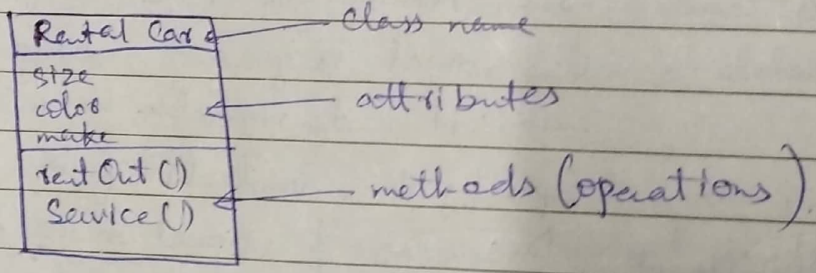
Date ____ 20

Tabular Specifications:

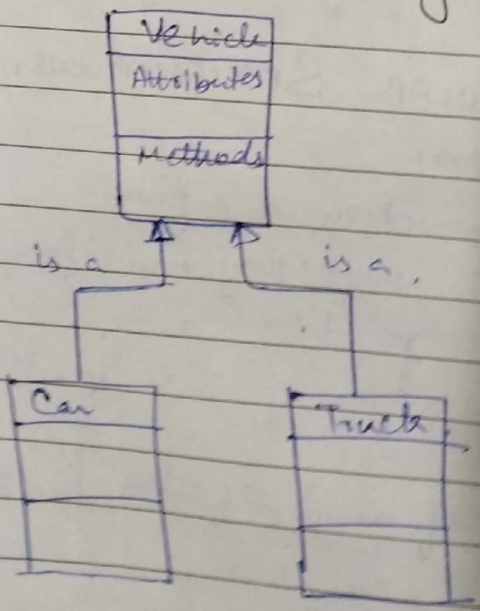
- Used to supplement natural language.
- useful when you have to define multiple alternative course of actions.
- For example, to calculate grade of a course for a student, we'll define different criterias for gpa in the course.
- Example, the insulin pump system bases its action for dosage on blood sugar level to calculate insulin requirements.

CHAPTER 5: OBJECT ORIENTED SYSTEMS ANALYSIS AND DESIGN USING UML:

① Example of UML Class.



② Class Diagram showing Inheritance.



Date ____ 20__

CRC Cards & Object Think.

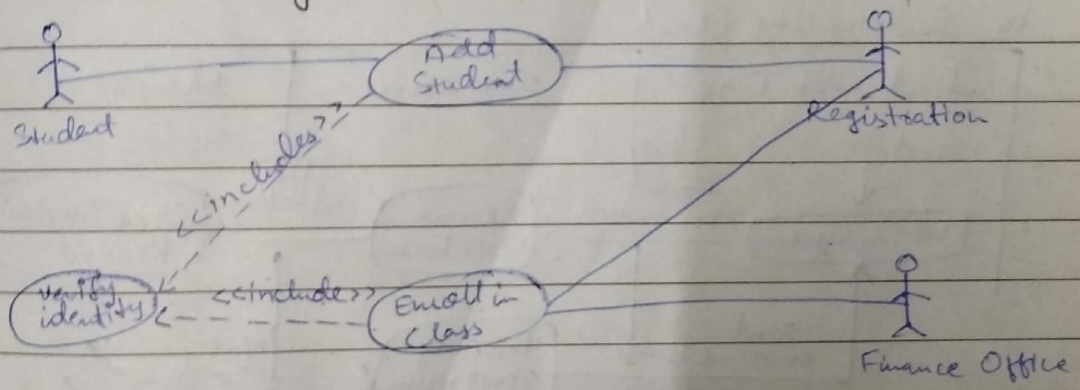
Class Responsibility Collaborator.

Classname			
Superclasses			
Subclasses			
Responsibilities	Collaborators	Object Think	Propaty
Methods description	All related Classes (inherit, aggregation etc)		all attributes

Example.

Classname : Department			
~~Superclass:~~			
Subclass			
Responsibilities	Collaborators	Object Think	Propaty
Add a new dept	course	I know my dept name	dept name
Provide dept info		I know who my HOD is.	hod

Use Case Diagram Example of a Student Enrollment



Add Student → a use case.

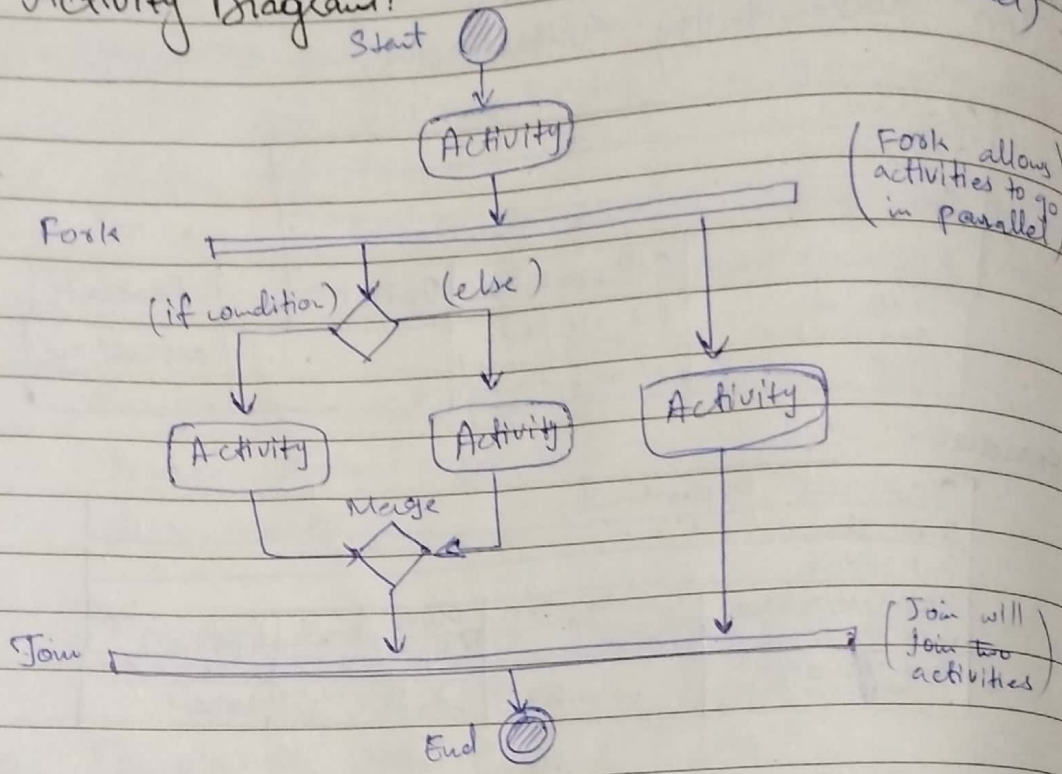
.---> → dotted arrow shows a use case involve anyother use case/method.

<<include>> → shows another method is required

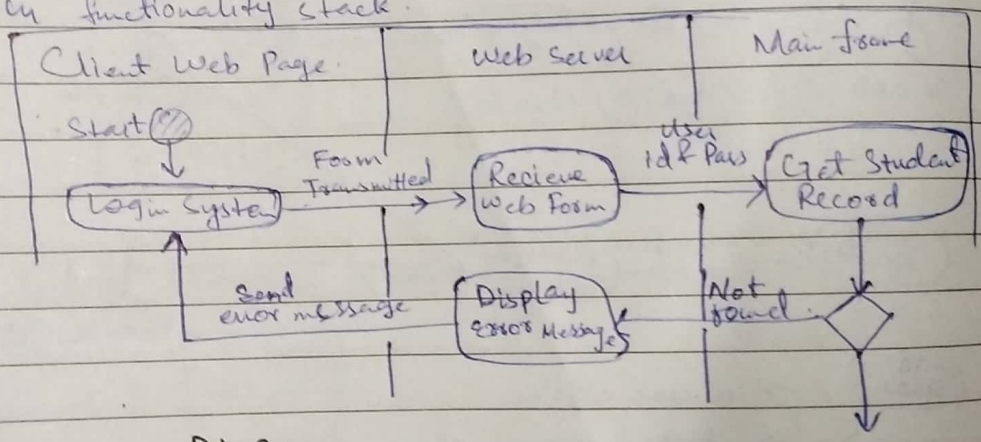
<<extend>> → shows another method is not essential but it extends this functionality.

Date ____ 20

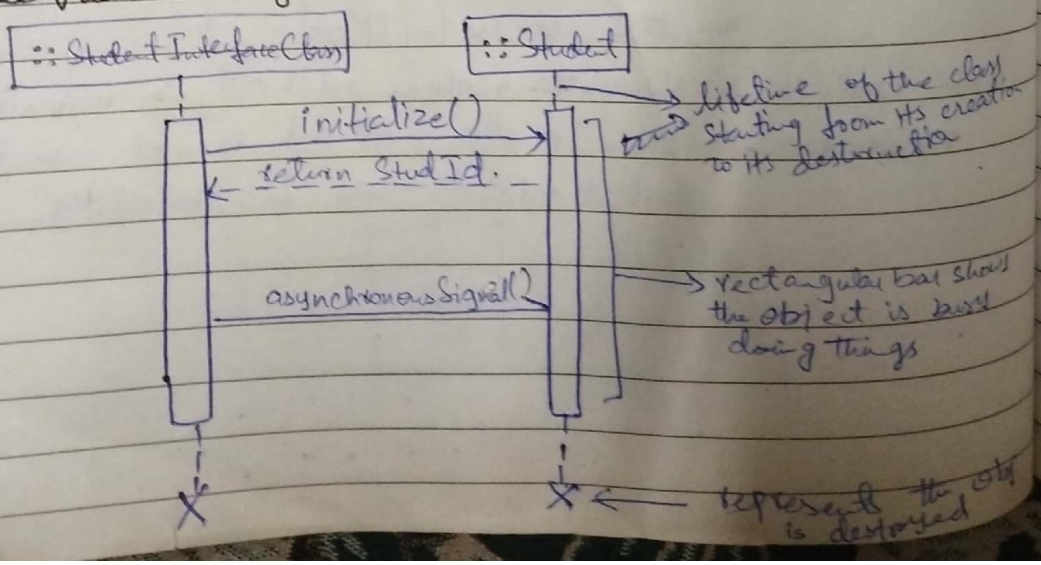
Activity Diagram: Sequence of activities (sequential or Parallel)



Swimlanes: Activity digrams are splitted by colors. by functionality stack.

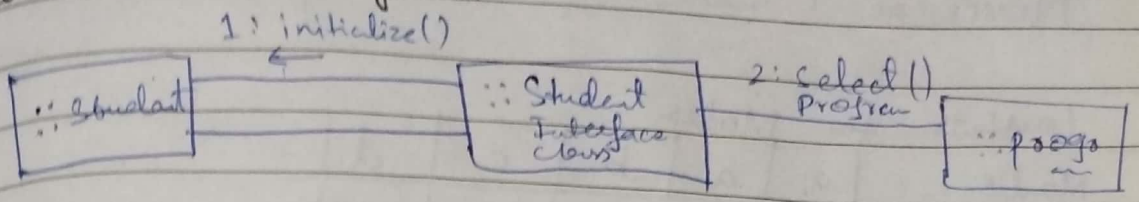


Sequence Diagram



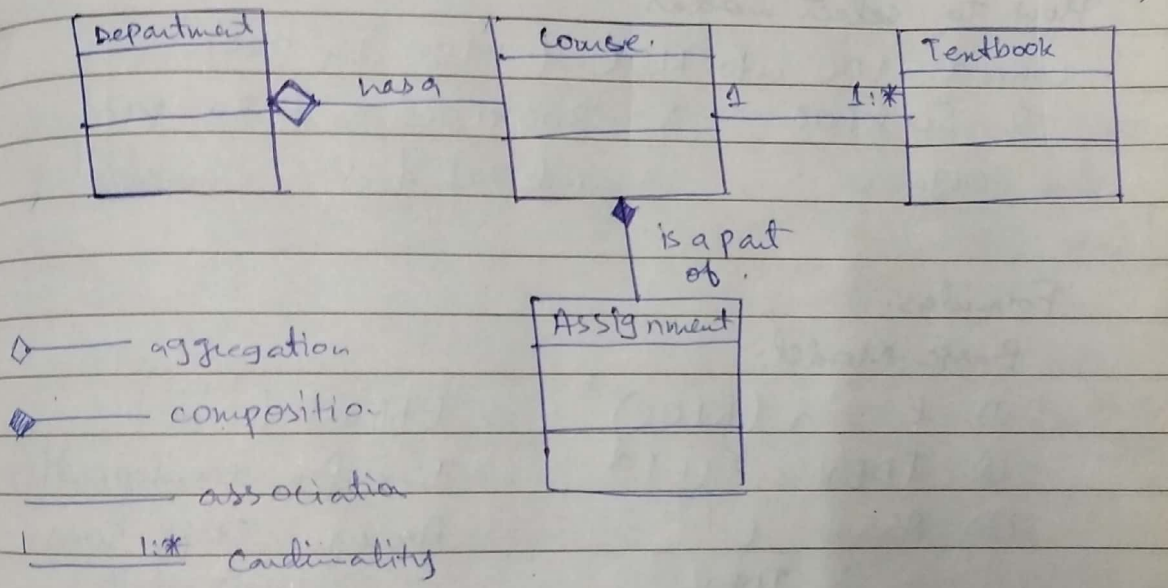
Date ____ 20 ____

Communication Diagram:

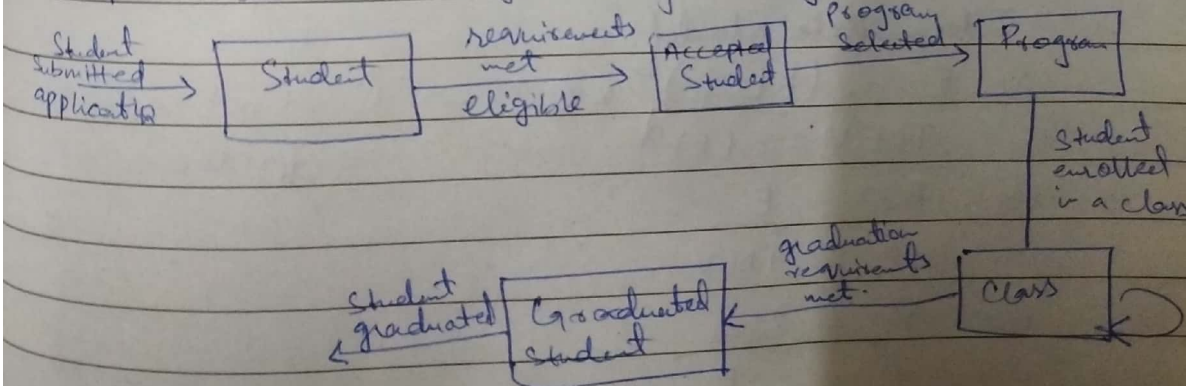


It show the same information as sequence diagram but emphasize the organization of objects rather than the ordering.

Class Diagram (with composition, aggregation & association)



Statechart diagrams. Shows the lifecycle of an object. Objects are created, go through changes & are deleted.



Date ____ 20

NUMERICAL : COCOMO (Constructive Cost Model)

Constants for Modes.

Modes.	a_i	a	b	c	d
organic	3.2	2.4	1.05	2.5	0.38
semi-detached	3.0	3.0	1.12	2.5	0.35
embedded	2.8	3.6	1.20	2.5	0.32
		Constants for Effort		Constants for TDEV.	

How to select modes

Convert LOC into KLOC. (÷ LOC by 1000)

2 - 50 KLOC — organic

50 - 300 KLOC — semi-detached

> 300 KLOC — embedded.

Formulas.

Basic Model:

① $E = a(KLOC)^b$ => Effort (Unit: SM)

② $TDEV = c(E)^d$ => Time for development (Unit: Month)

③ $ASS = \frac{E}{TDEV}$ => Average Staff Size. (unit: staff)

④ $P = \frac{Size}{E}$ => Productivity. (unit: KLOC/SM).

Intermediate Model:

① $E = a_i(KLOC)^b \times C$ → C = Cost adjustment factor.

② $TDEV = c(E)^d$

③ $ASS = \frac{E}{TDEV}$

④ $P = \frac{Size}{E}$

Date ____ 20__
M T W T F S S

Example : Suppose Size 200 KLOC, use basic model

Organic	Semidetached	Embedded
$E = a(KLOC)^b$	$E = a(KLOC)^b$	$E = a(KLOC)^b$
$E = 2.4(200)^{1.05}$	$E = 3.0(200)^{1.12}$	$E = 3.6(200)^{1.20}$
$E = 625.594$	$E = 1133.117$	$E = 2077.48$
E = 626 SM	**E = 1133 SM**	**E = 2077 SM**
$TDEV = c(E)^d$	$TDEV = c(E)^d$	$TDEV = c(E)^d$
$TDEV = 2.5(626)^{0.38}$	$TDEV = 2.5(6133)^{0.35}$	$TDEV = 2.5(2077)^{0.32}$
$TDEV = 28.88$	$TDEV = 29.303$	$TDEV = 28.80$
TDEV = 29 months	**TDEV = 29 months**	**TDEV = 29 months**
$ASS = \frac{E}{TDEV}$	$ASS = \frac{E}{TDEV}$	$ASS = \frac{E}{TDEV}$
$ASS = 626/29$	$ASS = 1133/29$	$ASS = 2077/29$
$ASS = 21.58$	$ASS = 39.06$	$ASS = 71.62$
ASS = 22 staff	**ASS = 39 staff**	**ASS = 72 staff**
$P = Size/E$	$P = Size/E$	$P = Size/E$
$P = 200/626$	$P = 200/1133$	$P = 200/2077$
P = 0.319 KLOC/SM	**P = 0.176 KLOC/SM**	**P = 0.096 KLOC/SM**

Ex: Suppose 30000 LOC, use intermediate model. C = 1.17

Organic	Semidetached	Embedded
$E = 3.2(30)^{1.05} \times 1.17$	$E = 3.0(30)^{1.12} \times 1.17$	$E = 2.8(30)^{1.20} \times 1.17$
$E = 135.14$	$E = 158.37$	$E = 194.03$
E = 135 SM	**E = 158 SM**	**E = 194 SM**
$TDEV = 2.5(133)^{0.38}$	$TDEV = 2.5(158)^{0.35}$	$TDEV = 2.5(194)^{0.32}$
$TDEV = 16.03$	$TDEV = 14.70$	$TDEV = 13.49$
TDEV = 16 months	**TDEV = 15 months**	**TDEV = 13 months**
$ASS = 133/16$	$ASS = 158/15$	$ASS = 194/13$
$ASS = 8.3125$	$ASS = 10.53$	$ASS = 14.92$
ASS = 8 staff	**ASS = 11 staff**	**ASS = 15 staff**
$P = 30/133$	$P = 11/158$	$P = 15/194$
P = 0.22 KLOC/SM	**P = 0.069 KLOC/SM**	**P = 0.077 KLOC/SM**