Software Engineering Syllabus Discussion | Imp. for Competitive, College/University & Placements

1) Introduction
2) Software Development Life cycle **
3) Requirement analysis (SRS)
4) Software Project Management * (COCOMO)
5) Software Design (Coupling, Cohesion, UML, DFD, Class diagram)
6) Coding and Testing **
7) Maintenance
8) Quality Mgmt, Reuse

13:11 / 14:20

Software Engineering Definition and Evolution

* It is systematic, disciplined, cost-effective techniques for software development.
* Engineering approach to develop a software.

Evolution

1945 65 -> Origin

1965 -> Crisis

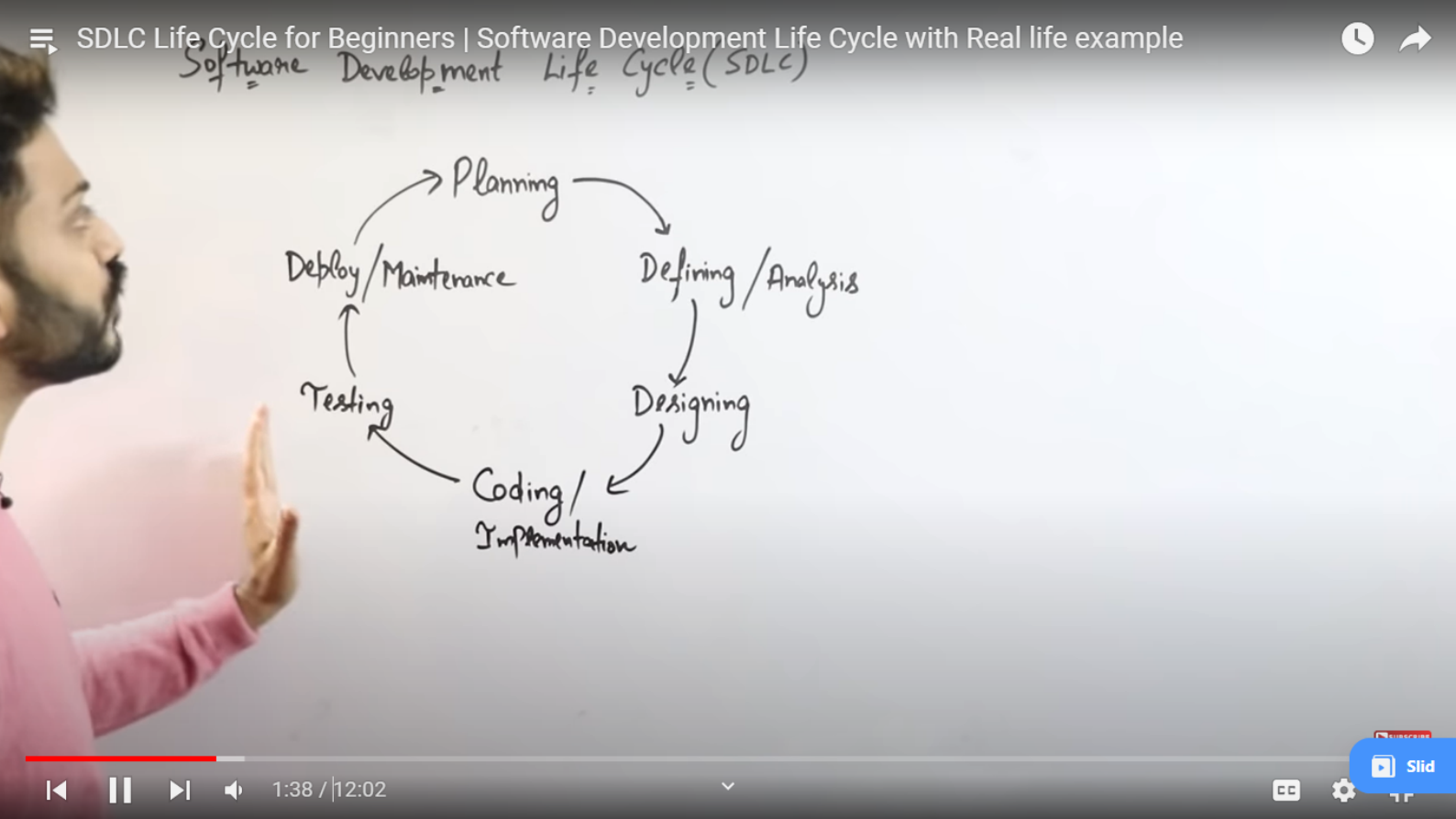
199 00 -> Internet

200 10 -> Light weight

2 Till -> AI, ML, DL

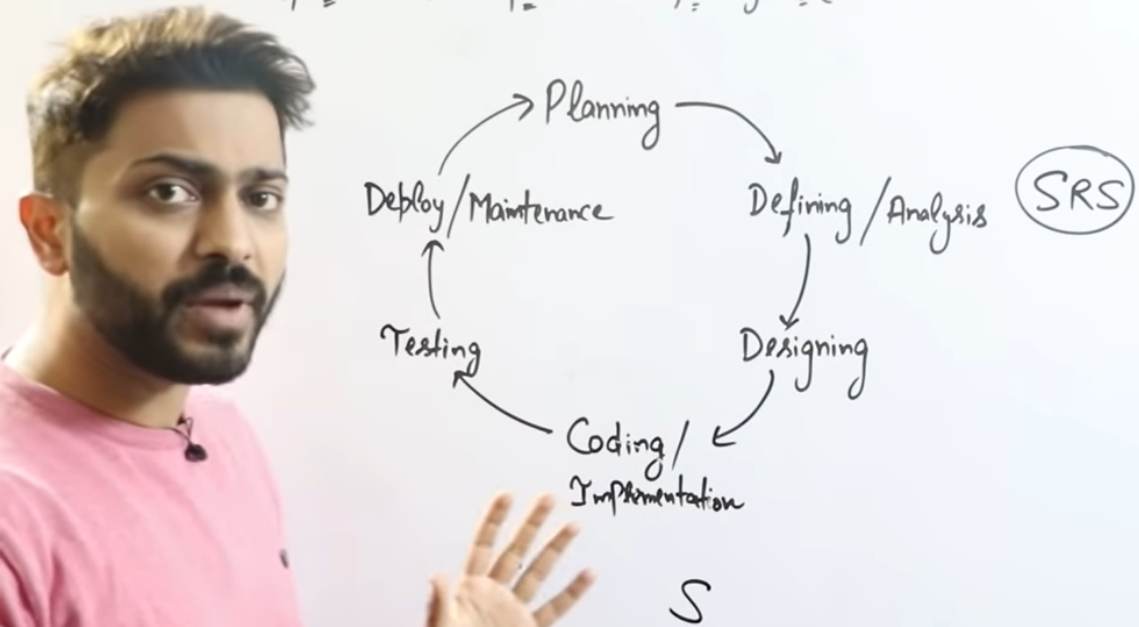
Slid

SDLC Life Cycle for Beginners | Software Development Life Cycle with Real life example



1:38 / 12:02

Software Development Life Cycle (SDLC)



Slid

Classic Waterfall model in Software Engineering

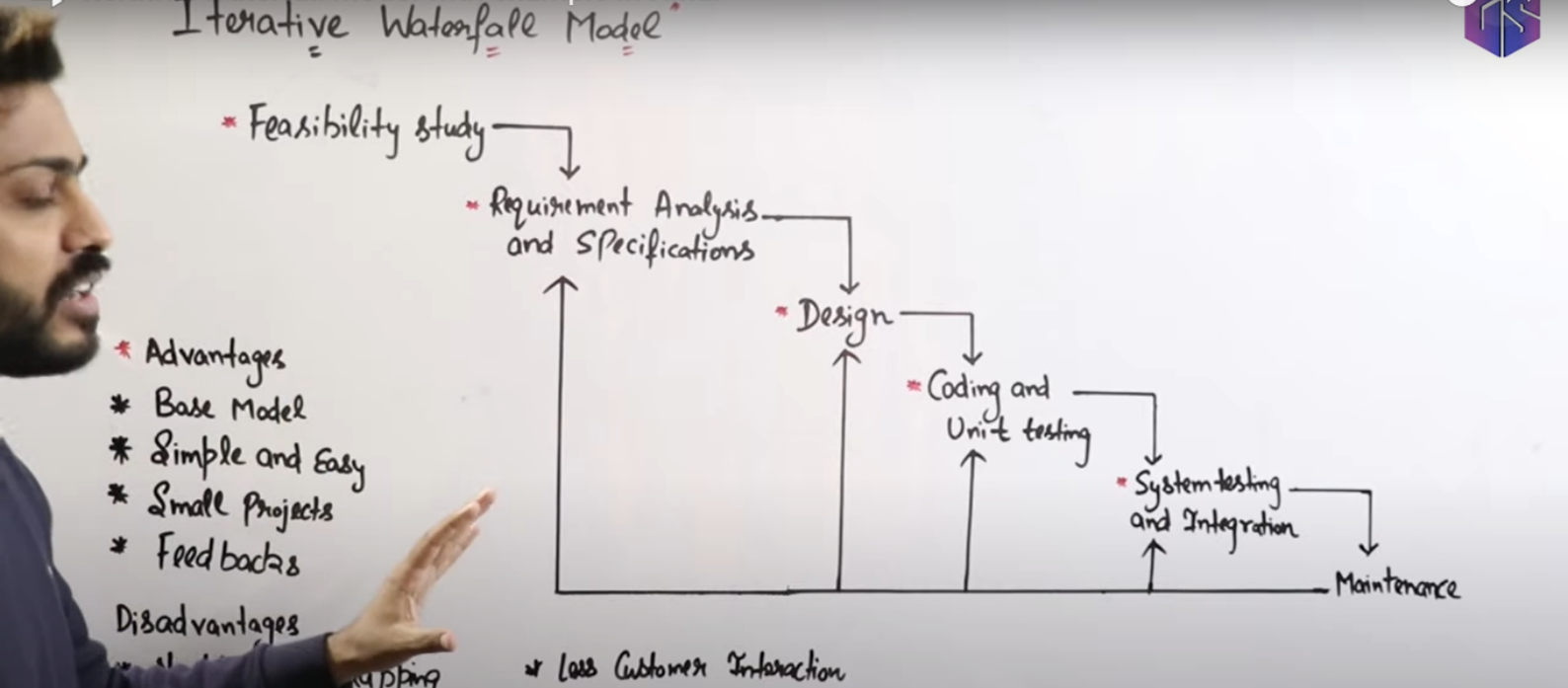
"Classical Waterfall Model"

- Feasibility study
- Requirement Analysis and Specifications
- Design
- Coding and Unit testing
- System testing and Integration
- Maintenance

Advantages	Disadvantages
* Base Model * Simple and Easy * Small Projects	* No feedback * No Experiment * No Parallelism * High Risk * 60% Efforts Maintenance

3:37 / 8:02
Slid

Iterative Waterfall Model



0:20 / 4:36
Slid

Iterative Waterfall Model with example in Hindi

Iterative Waterfall Model

- Feasibility study
- Requirement Analysis and Specifications
- Design
- Coding and Unit testing

- Advantages
- Base Model
- Simple and Easy
- Small Projects
- Feedbacks

Disadvantages

- No phase overlapping
- No Intermidiate Delivery
- Rigid (No changes)
- Less Customer Interaction

2:23 / 4:36
Slid

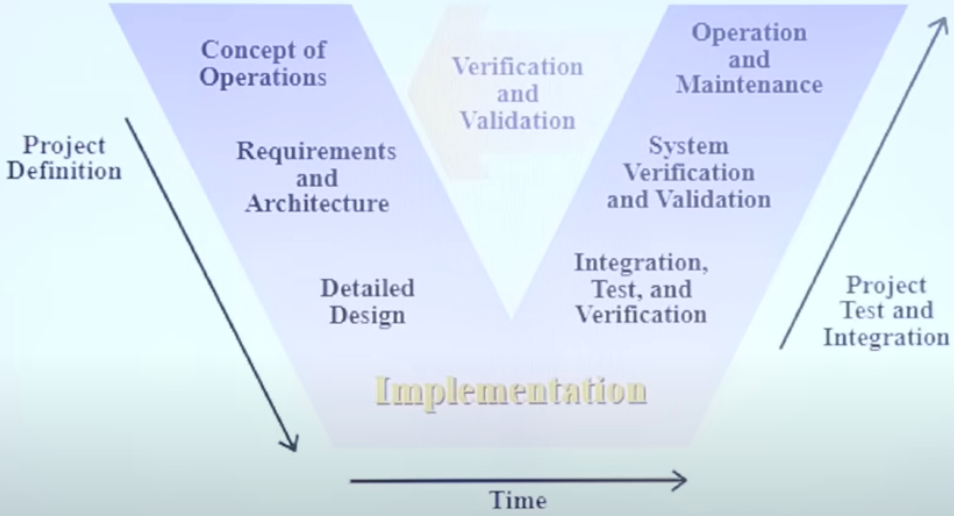
V Shaped Model with examples | SDLC | Software Engineering

V-Shaped Model

- Also known as Verification & Validation Model
- Extension of Waterfall model.
- Testing is associated with every phase of lifecycle.
- Verification Phase(Requirement analysis, System design, Architecture design, Module design)
- Validation Phase(Unit testing, Integration, System, Acceptance Testing)

3:12 / 7:15

V Shaped Model with examples | SDLC | Software Engineering



3:53 / 7:15

V Shaped Model with examples | SDLC | Software Engineering

Advantages

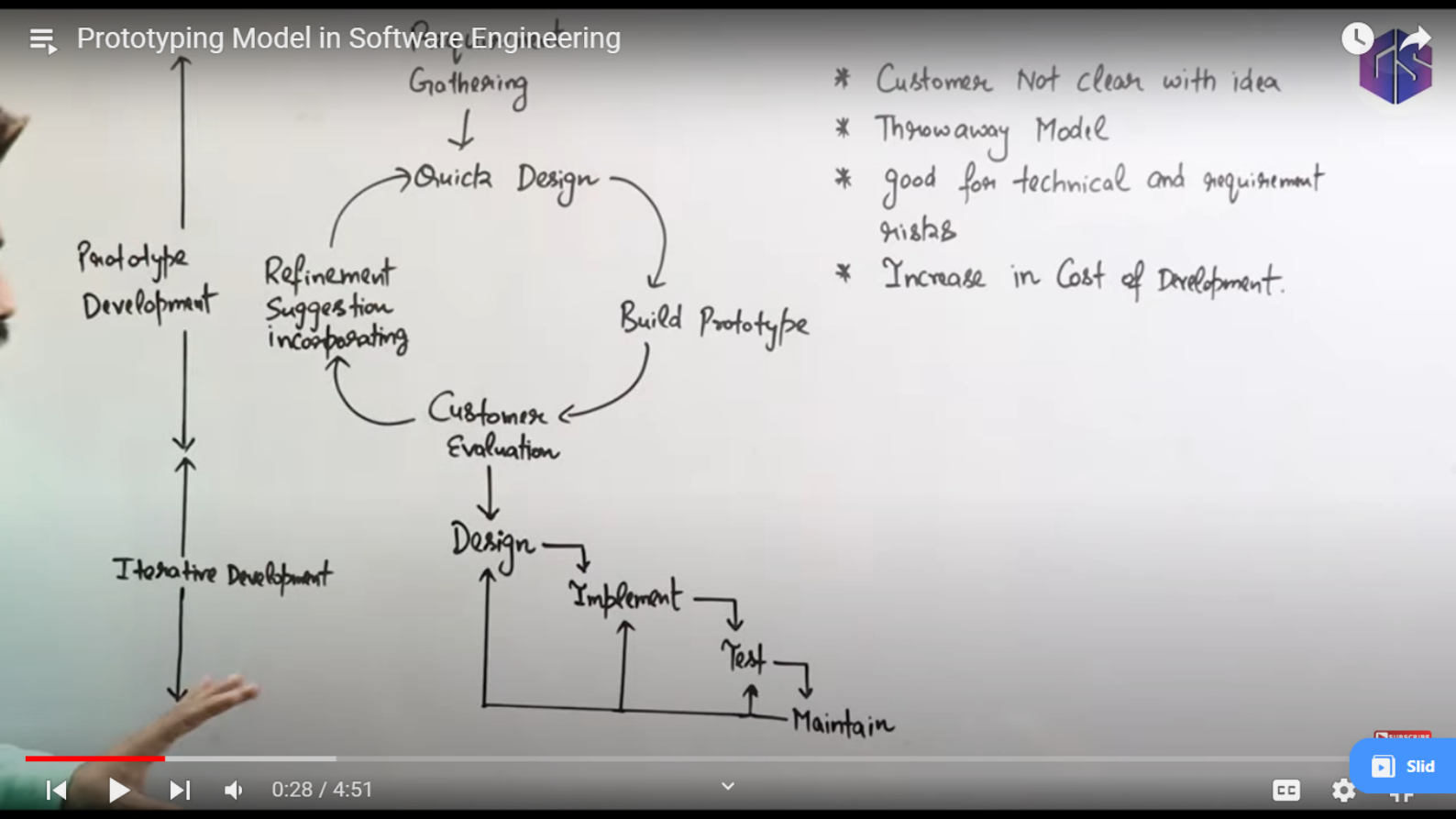
- Time saving
- Good understanding of project in the beginning.
- Every component must be testable.
- Progress can be tracked easily.
- Proactive defect tracking

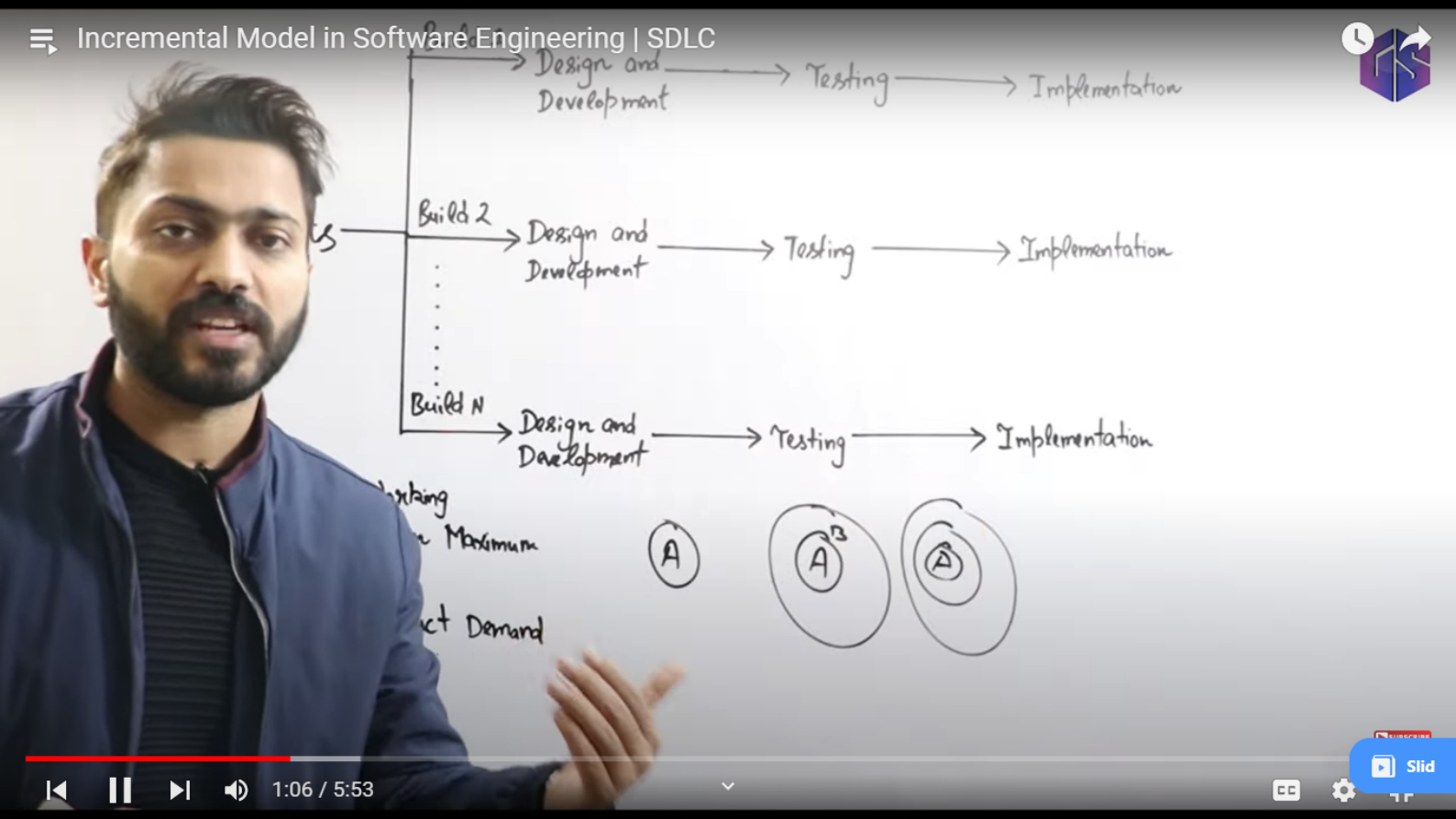
Disadvantages

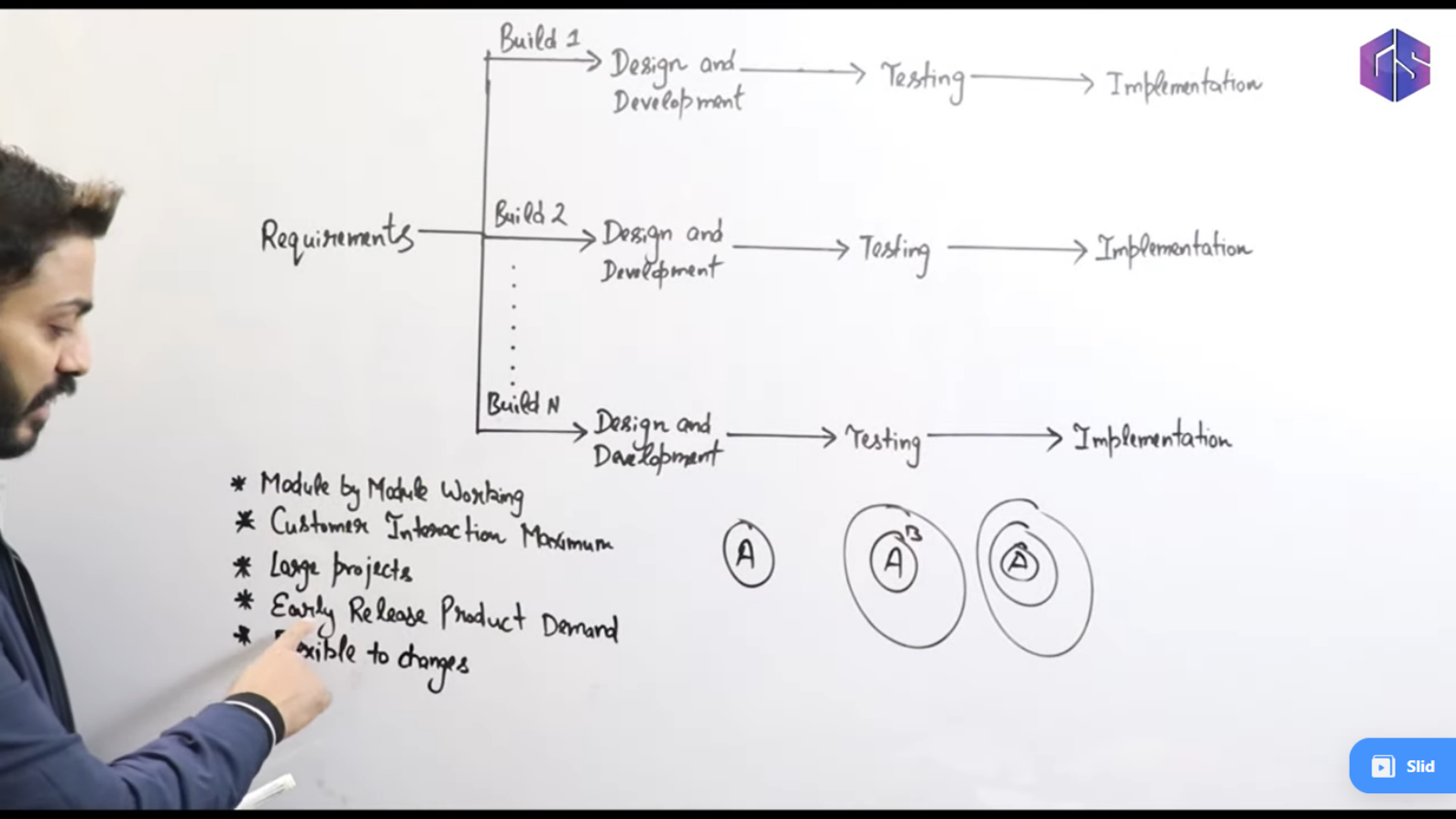
- No feedback so less scope of changes
- Risk analysis not done
- Not good for big or object-oriented projects

4:49 / 7:15

Prototyping Model in Software Engineering

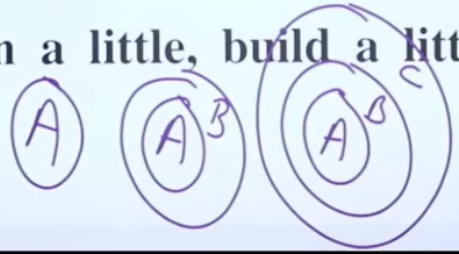



Incremental Model in Software Engineering | SDLC
Design and Development
Testing
Implementation
Build 2
Design and Development
Testing
Implementation
Build N
Design and Development
Testing
Implementation
Maximum
Demand
A
B
C
1:06 / 5:53
Slid


Build 1
Design and Development
Testing
Implementation
Requirements
Build 2
Design and Development
Testing
Implementation
Build N
Design and Development
Testing
Implementation
* Module by Module Working
* Customer Interaction Maximum
* Large projects
* Early Release Product Demand
* Flexible to changes
A
A
A

Evolutionary model

- Evolutionary model is a combination of Iterative and Incremental model of software development life cycle.
- Incremental model first implement a few basic features and deliver to the customer. Then build the next part and deliver it again and repeat this step until the desired system is fully realized. No long-term plans are made.
- Iterative model main advantage is its feedback process in every phase.
- Also known as "Design a little, build a little, test a little, deploy a little model".



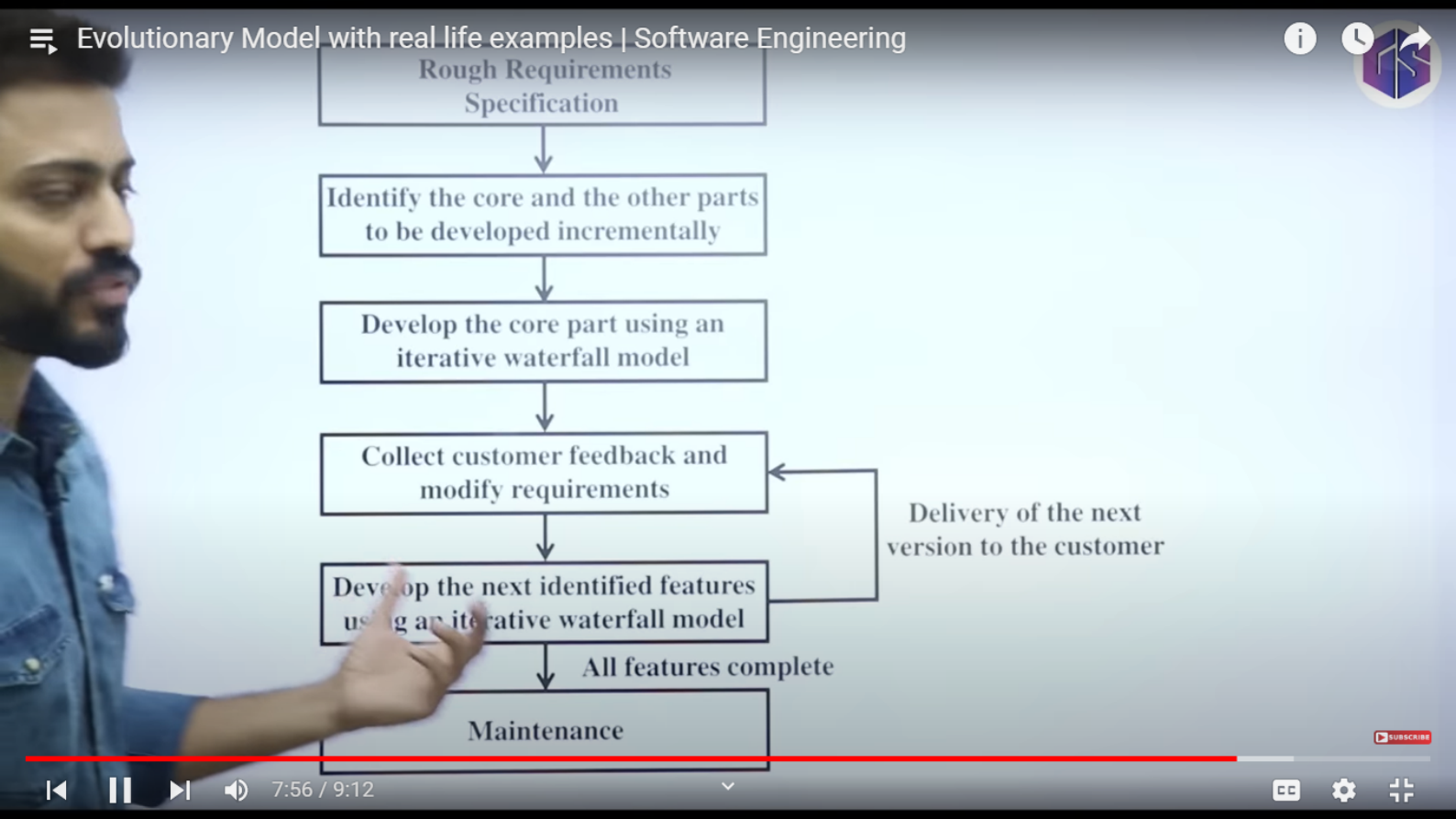
Advantages

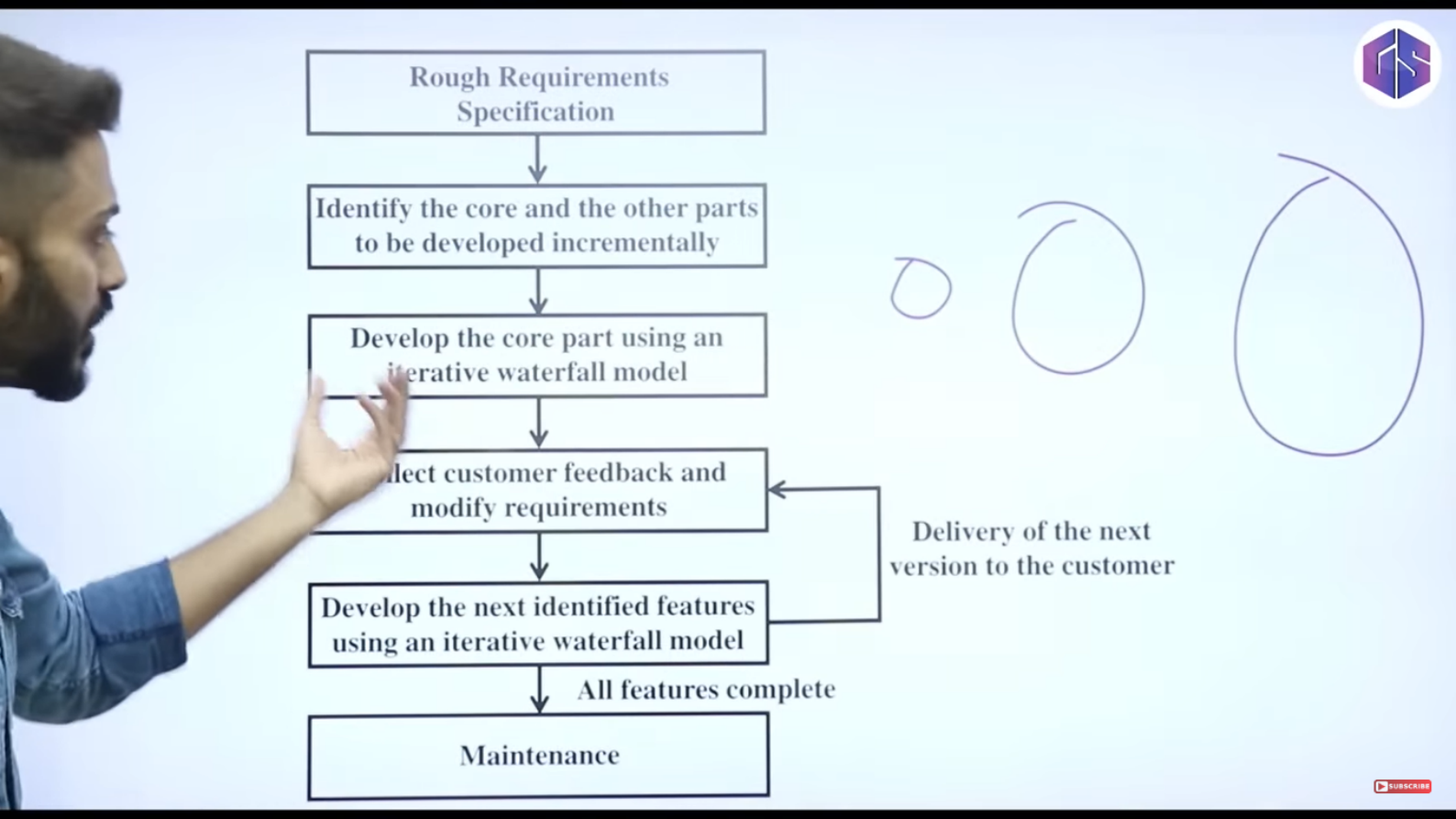
- Customer requirements are clearly specified.
- Risk analysis is better.
- It supports changing environment.
- Initial operating time is less.
- Better suited for large mission-critical projects.

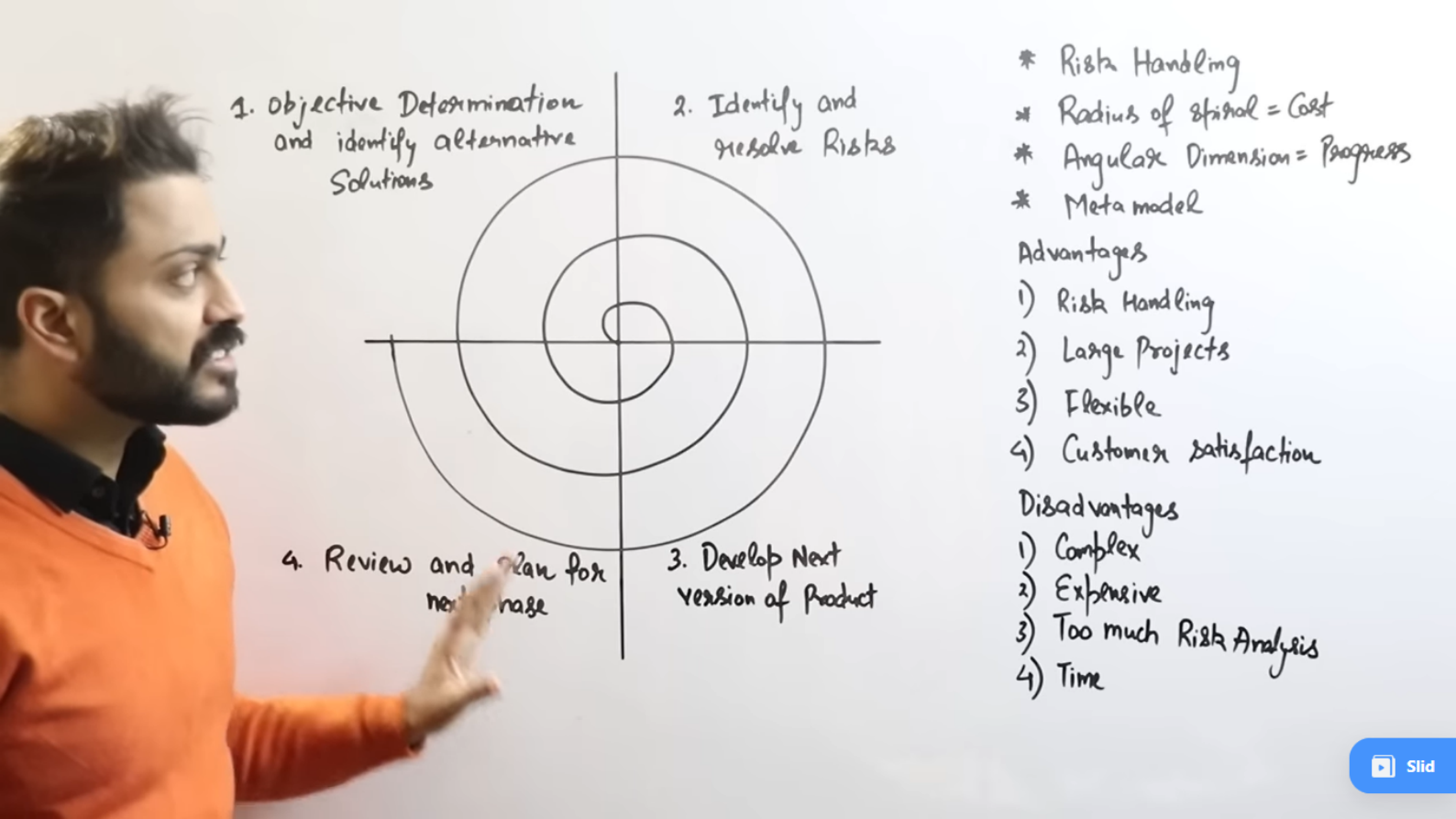
Disadvantages

- Not suitable for smaller projects.
- Cost
- Highly skilled resources are required.

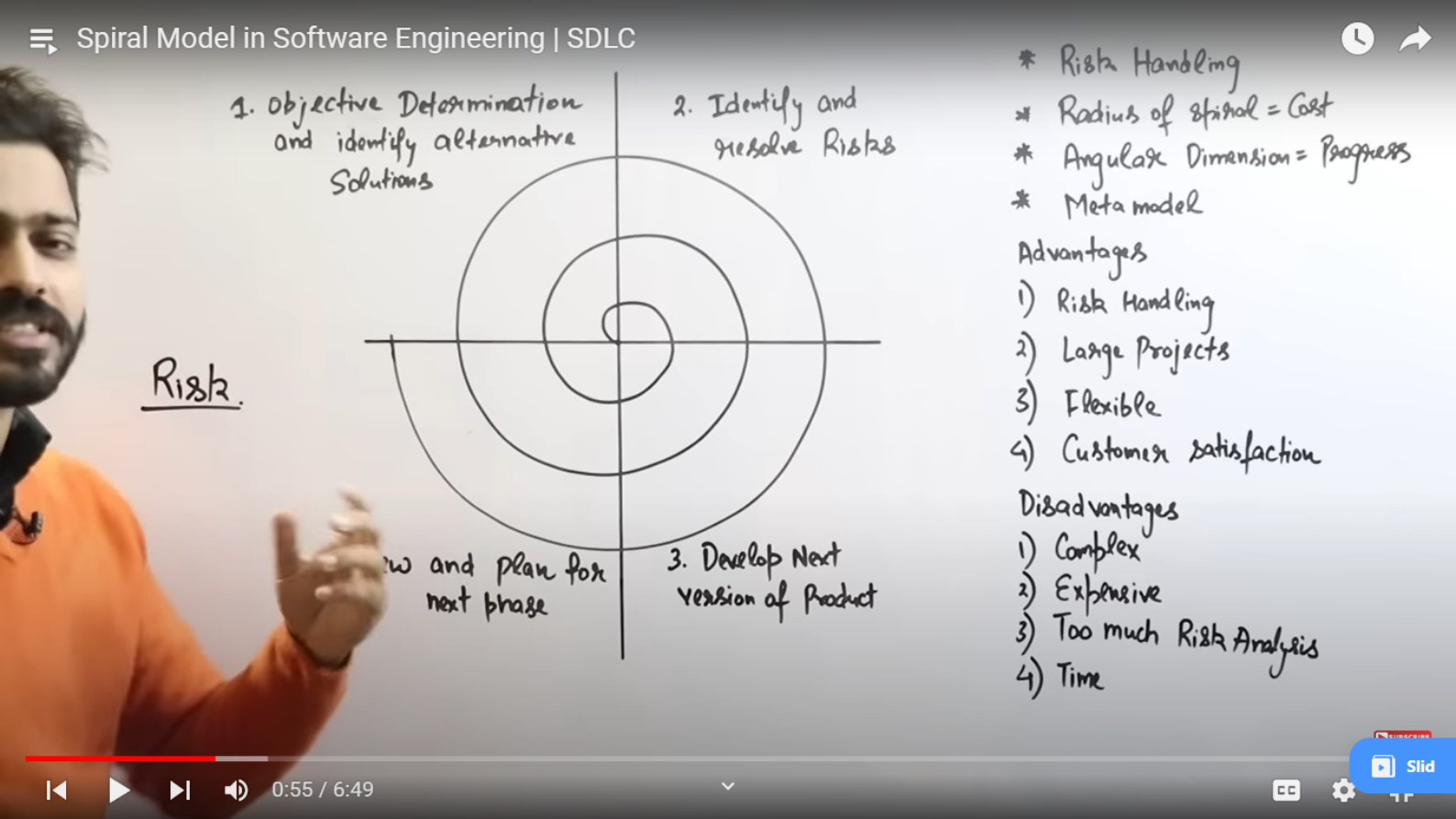
SUBSCRIBE


Evolutionary Model with real life examples | Software Engineering
Rough Requirements Specification
Identify the core and the other parts to be developed incrementally
Develop the core part using an iterative waterfall model
Collect customer feedback and modify requirements
Delivery of the next version to the customer
Develop the next identified features using an iterative waterfall model
All features complete
Maintenance
7:56 / 9:12

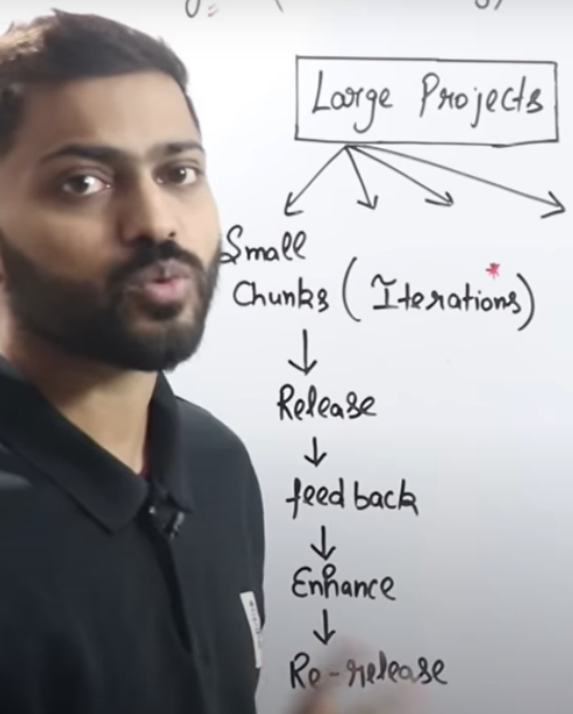

Rough Requirements Specification
Identify the core and the other parts to be developed incrementally
Develop the core part using an iterative waterfall model
Collect customer feedback and modify requirements
Develop the next identified features using an iterative waterfall model
Delivery of the next version to the customer
All features complete
Maintenance


1. Objective Determination and identify alternative Solutions
2. Identify and resolve Risks
3. Develop Next version of Product
4. Review and Plan for next phase
* Risk Handling
* Radius of spiral = Cost
* Angular Dimension = Progress
* Meta model
Advantages
1) Risk Handling
2) Large Projects
3) Flexible
4) Customer satisfaction
Disadvantages
1) Complex
2) Expensive
3) Too much Risk Analysis
4) Time
Slid

Spiral Model in Software Engineering | SDLC



"Agile" (Move Quickly)



Advantages:
1) Frequent Delivery
2) Face to face communication with client
3) Changes
4) Time

Disadvantage:
1) Less documentation
2) Maintanance Problem

SUBSCRIBED

3:06 / 8:01

Slid

SCRUM Model in Software Engineering | Agile Technology

SCRUM

- One of the most popular agile methodology.
- Scrum is a lightweight, iterative and incremental framework.
- Scrum breaks down the development phases into stages or cycles called "sprints".
- The development time for each sprint is maximized and dedicated, thereby managing only one sprint at a time.
- Scrum Team has scrum master and product owner with constant communications on the daily basis.
- Keywords: Backlog, Sprint, Daily Scrum, Scrum master, Product owner.

1:35 / 7:58
Slid

SCRUM Model in Software Engineering | Agile Technology

Advantages

- Freedom & Adaption
- High-quality, low-risk product.
- Reduce the development time up to 40%
- Scrum customer satisfaction is very important.
- Reviewing the current sprint before moving to new one.

Disadvantages:

- More efficient for small team size.
- No changes in the sprint.

5:47 / 7:58
Slid

Comparison of All SDLC Models | Waterfall, Iterative, Prototype, Spiral, Increment, RAD, Agile etc.

Classical Waterfall	Iterative Waterfall	Prototype Model	Incremental Model	Evolutionary Model	RAD Model	Spiral Model	Agile Model
Basic, Rigid, Inflexible, Not for Real Project	Basic, Problem is well understood	User Requirement Not clear, Costly, No Early lock on Requirements → High User Involvement → Reusability	Module by Module Delivery, Easy to test and debug	Large Projects	Time and Cost Constraint, User at all levels → Reusability	Risk, Not for small Projects, → No Early lock on Requirements, → Less Experience Can work	Flexible, Advanced, Parallel, Process divided into sprints

1:05 / 9:26

Software Requirements | Requirement Engineering | Feasibility Study, Elicitation, SRS, Validat...

Software Requirements

- It is the description of features and functionalities of the target system.
- It is the description of what the system should do.
- Requirements engineering (RE) refers to the process of defining, documenting, and maintaining requirements in the engineering design process.

It is a four step process, which includes –

1. Feasibility Study
2. Requirement Gathering/Elicitation
3. Software Requirement Specification
4. Software Requirement Validation

2:44 / 10:17

Software Requirements | Requirement Engineering | Feasibility Study, Elicitation, SRS, Validat...

Tool support for Requirements Engineering

- Observation reports (user observation)
- Questionnaires (interviews, surveys and polls),
- Use cases
- User stories
- Requirement workshops
- Mind mapping
- Role-playing
- Prototyping

8:43 / 10:17

Functional vs Non-functional Requirements | Requirement Engineering | Software Engineering

Functional vs Non-Functional Requirements

- Requirements, which are related to functional/Working aspect of software fall into this category.
- Non-Functional Requirements are expected characteristics of target software. (Security, Storage, Configuration, Performance, Cost, Interoperability, Flexibility, Disaster recovery, Accessibility)

0:11 / 5:02
Slid

Software Requirements Specification(SRS)

- SRS is a description of a software system to be developed.
- It lays out functional and non-functional requirements of the software to be developed.
- It may include a set of use cases that describe user interactions that the software must provide to the user for perfect interaction.

0:13 / 9:35 • Introduction ›
Slid

SRS Structure

1. Introduction
    - 1.1 Purpose
    - 1.2 Intended Audience
    - 1.3 Scope
    - 1.4 Definitions
    - 1.5 References
2. Overall Description
    - 2.1 User Interfaces
    - 2.2 System Interfaces
    - 2.3 Constraints, assumptions and dependencies
    - 2.4 User Characteristics

3:01 / 9:35 • SRS Structure ›
Slid

3. System Features and Requirements

- 3.1 Functional Requirements
- 3.2 Use Cases
- 3.3 External Interface Requirements
- 3.4 Logical database requirement
- 3.5 Nonfunctional Requirements

4. Deliver for Approval

6:41 / 9:35 • SRS Structure ›
Slid

3. System Features and Requirements

3.1 Functional Requirements

3.2 Use Cases

3.3 External Interface Requirements

3.4 Logical database requirement

3.5 Nonfunctional Requirements

4. Deliver for Approval

Safety

Security

Availability

8:57 / 9:35 • System Features and Requirements ›

Slid

User Requirements with real life examples | User Requirement Specification | Software Engineering

User Requirements

- Easy & Simple to Operate
- Quick Response
- effectively handling operational errors
- Customer support

0:24 / 6:39
Slid

User Requirement Specification

- The user requirement(s) document (URD) or user requirement specification (URS) is a document usually used in software engineering that specifies what the user expects the software to be able to do.
- It is a contractual agreement.

Menu Exit Share List Selection Write Eraser More Add Previous 10/10 Page Next SUBSCRIBE

What is DFD? How to design DFD, Symbols, examples full explanation

Data Flow Diagrams

- A *graphical tool*, useful for communicating with users, managers, and other personnel.
- Useful for analyzing existing as well as proposed systems.
- Focus on the *movement* of data between external entities and processes, and between processes and data stores.

A relatively simple technique to learn and use.

Type here to search
ENG
23:28
24-04-2023
Slid

Data Flow Diagrams

Bubble Chart.

- A *graphical tool*, useful for communicating with users, managers, and other personnel.
- Useful for analyzing existing as well as proposed systems.
- Focus on the *movement* of data between external entities and processes, and between processes and data stores.
- A relatively simple technique to learn and use.

Slid

Why DFD ?

- Provides an overview of-
  - What data a system processes
  - What transformations are performed
  - What data are stored
  - What results are produced
  - Graphical nature makes it a good communication tool between-
  - User and analyst
  - Analyst and System designer

Slid

DFD elements

- Source/Sinks (External entities)
- Data flows
- Processes
- Data Stores

Slid

What is DFD? How to design DFD, Symbols, examples full explanation

DFD elements

User,

- Source/Sinks (External entities)
- Data flows
- Processes
- Data Stores

DB

5:10 / 8:15

What is DFD? How to design DFD, Symbols, examples full explanation

External Entities

- A Rectangle represents an external entity
- They either supply or receive data
- They do not process data

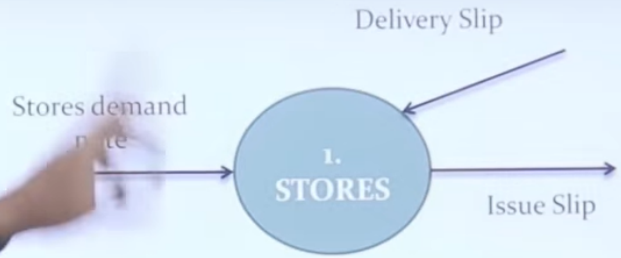
Source – Entity that supplies data to the system.

Sink – Entity that receives data from the system.

5:29 / 8:15

What is DFD? How to design DFD, Symbols, examples full explanation

Processes

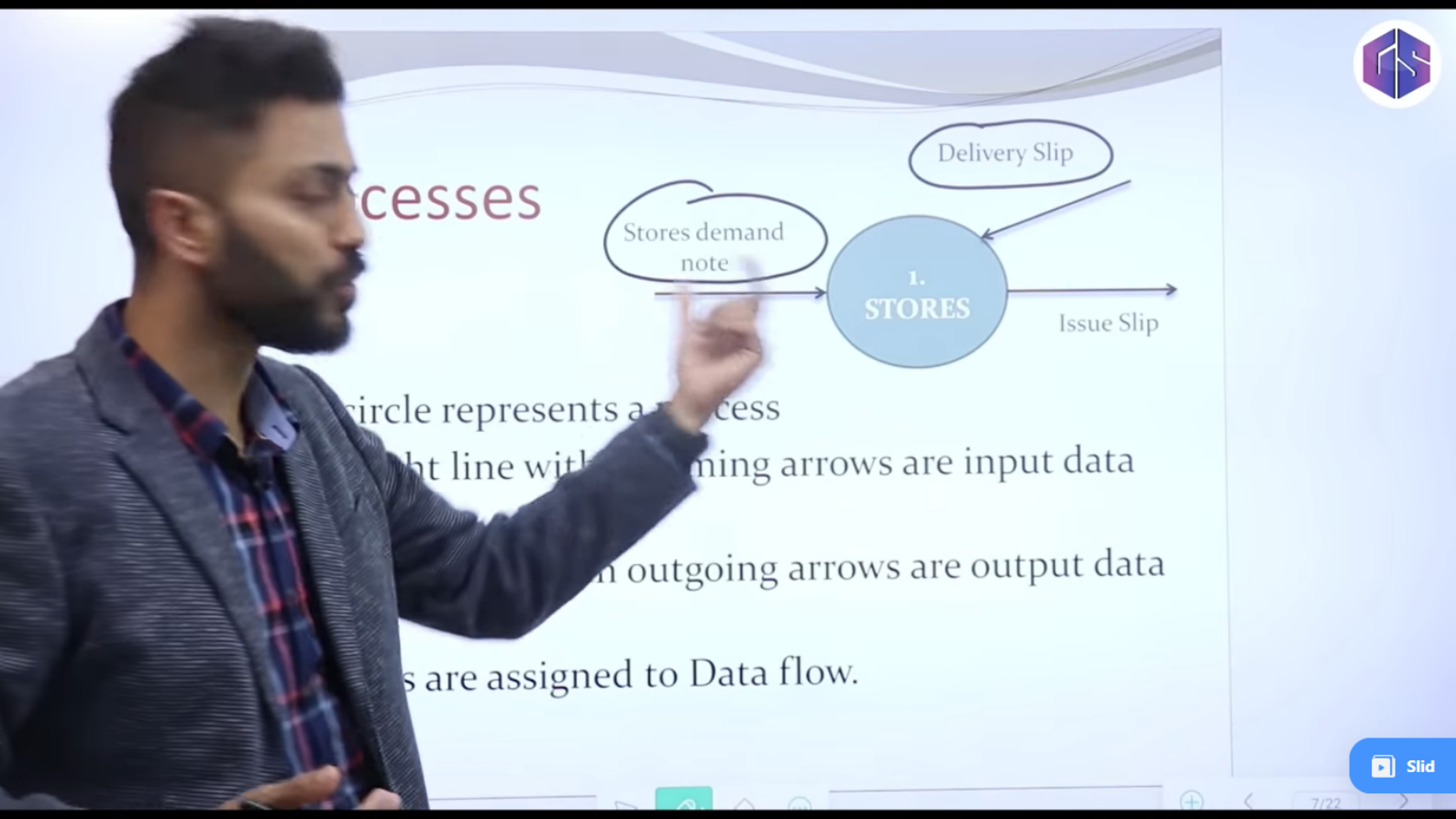


ents a process

ght line with incoming arrows are input data flows

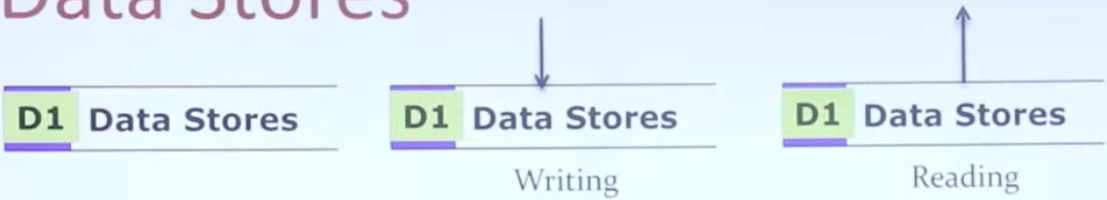
- Straight lines with outgoing arrows are output data flows
- Labels are assigned to Data flow.

5:44 / 8:15


cesses
Delivery Slip
Stores demand note
1. STORES
Issue Slip
circle represents a cess
ht line wit ming arrows are input data
n outgoing arrows are output data
s are assigned to Data flow.
Slid

What is DFD? How to design DFD, Symbols, examples full explanation

Data Stores



- A Data Store is a repository of data
- Data can be written into the data store. This is depicted by an incoming arrow
- Data can be read from a data store. This is depicted by an outgoing arrow
- External entity cannot read or write to the data store

5:57 / 8:15
Slid

What is DFD? How to design DFD, Symbols, examples full explanation

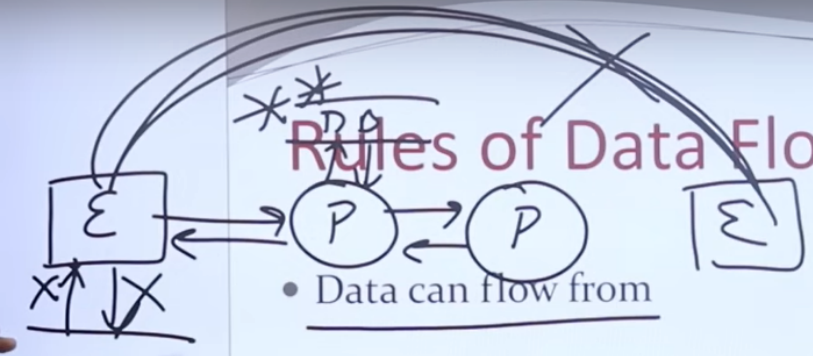
Rules of Data Flow

- Data can flow from
  - External entity to process
  - Process to external entity
  - Process to store and back
  - Process to process
- Data cannot flow from
  - External entity to external entity
  - External entity to store
  - Store to external entity
  - Store to store

6:13 / 8:15

What is DFD? How to design DFD, Symbols, examples full explanation

Rules of Data Flow

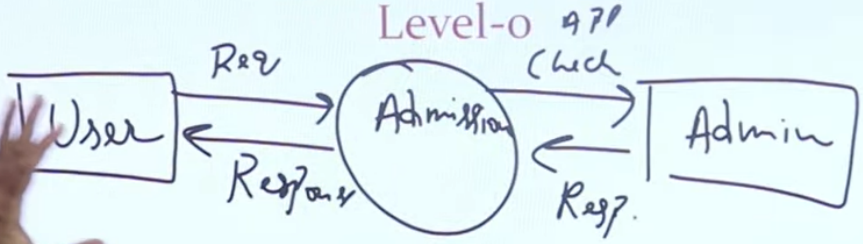


- Data can flow from
  - External entity to process
  - Process to external entity
  - Process to store and back
  - Process to process
- Data cannot flow from
  - External entity to external entity
  - External entity to store
  - Store to external entity
  - Store to store

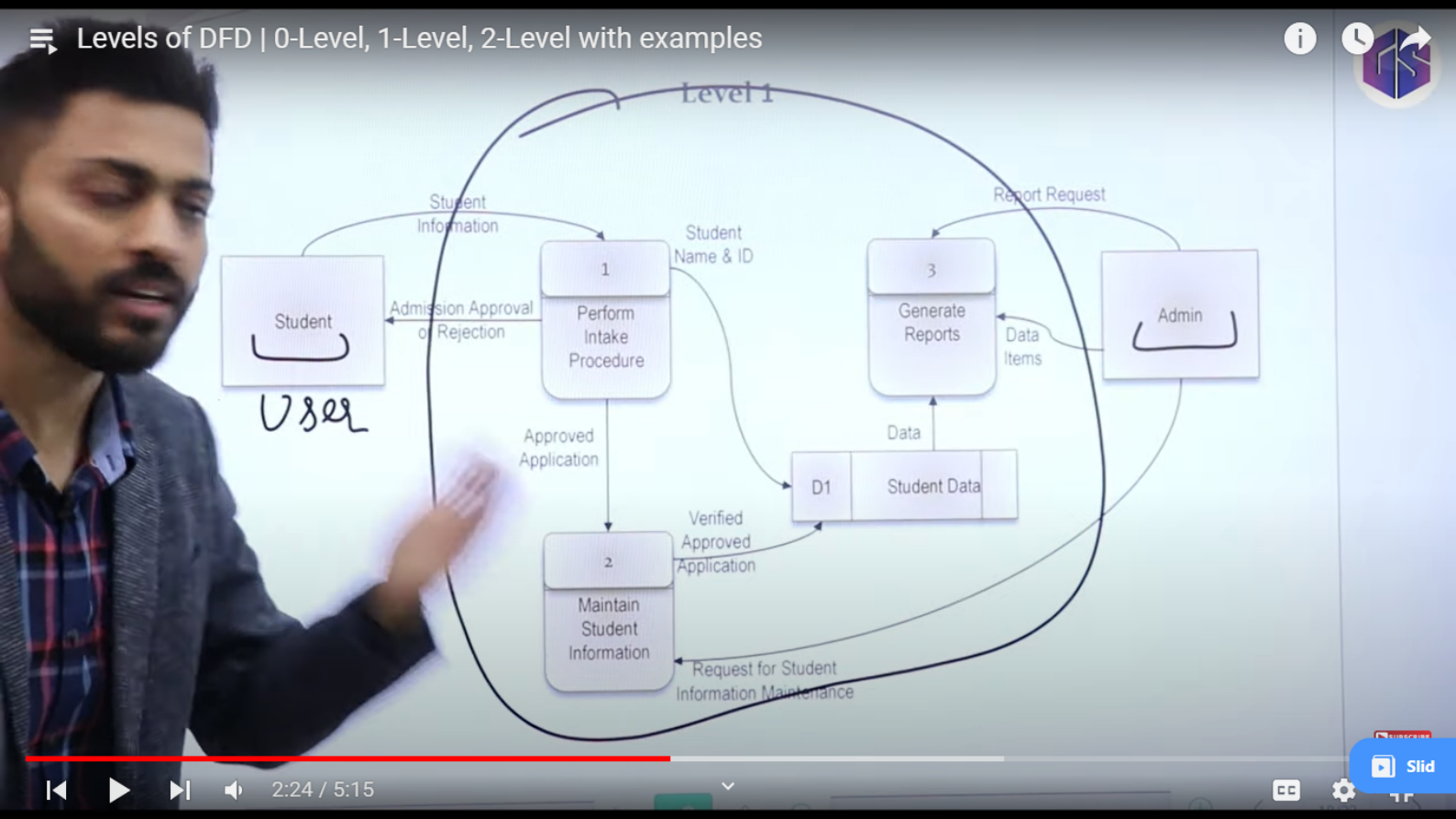
7:57 / 8:15
Slid

DFD for University Admission System

Level-o

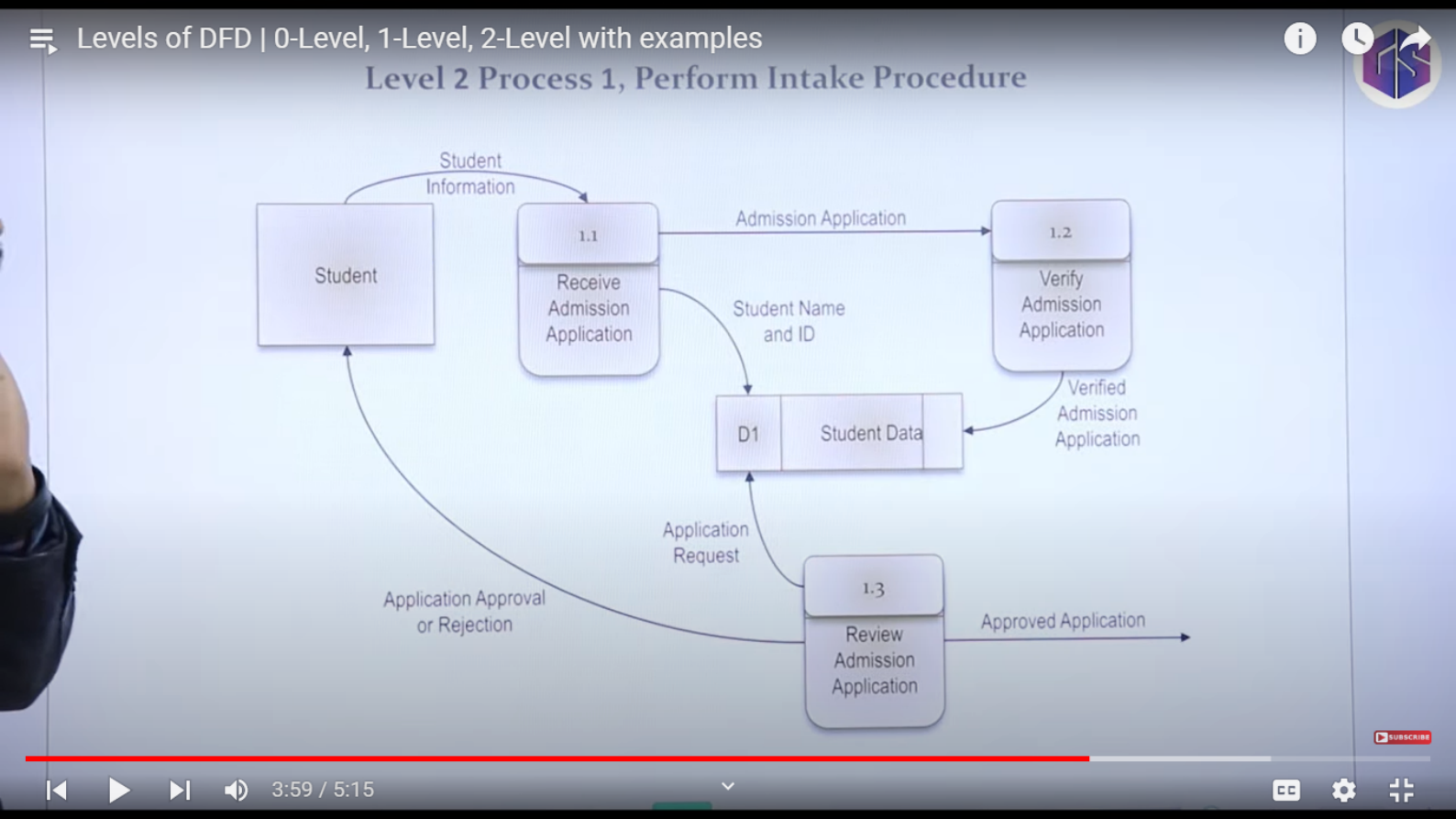


Levels of DFD | 0-Level, 1-Level, 2-Level with examples



2:24 / 5:15

Levels of DFD | 0-Level, 1-Level, 2-Level with examples



Logical vs Physical DFD with example | How Question asked from DFD?

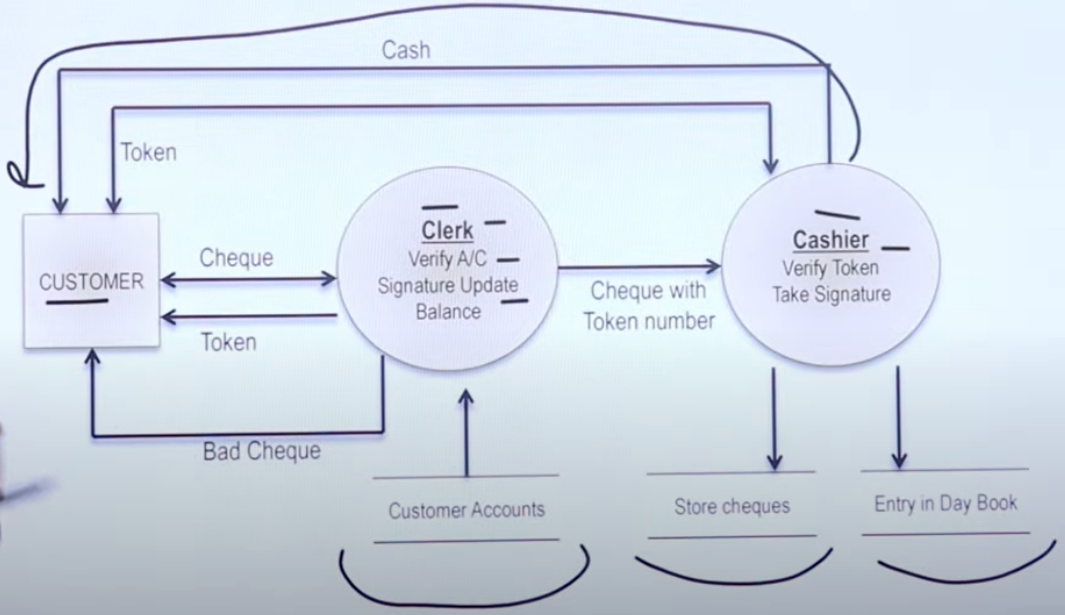
Logical and Physical DFD

- DFDs considered so far are called logical DFDs
- A physical DFD is similar to a document flow diagram
- It specifies who does the operations specified by the logical DFD
- Physical DFD may depict physical movements of the goods
- Physical DFDs can be drawn during fact gathering phase of a life cycle

0:08 / 3:11
Slid

Logical vs Physical DFD with example | How Question asked from DFD?

Physical DFD for Cheque Encashment



1:30 / 3:11
Slid

Logical vs Physical DFD with example | How Question asked from DFD?

Question

Which possibility among the following is invalid in case of a Data Flow Diagram?

a) A process having in-bound data flows more than outbound data flow

b) A data flow between two processes

c) A data flow between two data stores

d) A data store having more than one in-bound data flows

2:35 / 3:11
Slid

Software Design Approaches

Function Oriented Design

- System is designed from a functional viewpoint.
- Top-down decomposition
- Divide & Conquer approach
- DFD is used

Object Oriented Design

- System is viewed as a collection of objects (i.e., entities)
- Bottom-up approach
- UML is used

Slid

Software Design Approaches

MS

Return

Student, faculty, Book

Fun ted Design

- Syste ed from a funct point.
- T

Obj ent Oriented Design

- Syst s viewed as a tion of objects (i.e., tities)
- Bottom-up approach
- UML is used

Clean Ruler Shape Form Mind map Note Move Grid Undo Add Previous 3/3 Page Next SUBSCRIBE

SPM(Software Project Management) with real life examples

SPM (Software Project Management)

- Software project management is an art and science of planning and leading software projects.
- Main goal is to enable a group of developers to work effectively towards successful completion of project.
- Project manager is an administrative leader of the team.
- Various factors make this job very complex (e.g. Changeability, Complexity, Uniqueness, Possibility of multiple solutions etc.)

5 seconds
2:50 / 10:37
Slid

SPM(Software Project Management) with real life examples

Job responsibilities of Project Manage

- Planning
- Organizing
- Staffing
- Directing
- Monitoring
- Controlling
- Innovating
- Representing

5:06 / 10:37
Slid

SPM(Software Project Management) with real life examples

Skills required for Project manager

- Managerial skills
- Technical skills
- Problem solving skills
- Coping skills
- Conceptual skills
- Leadership skills
- Communication skills

7:00 / 10:37
Slid

SPM(Software Project Management) with real life examples

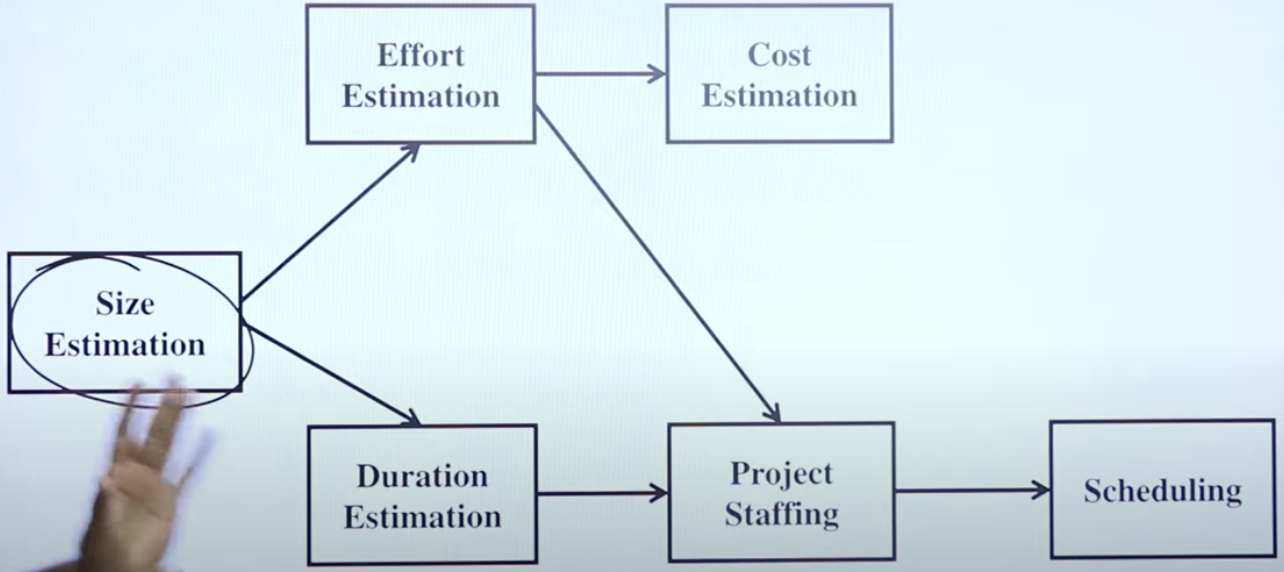
Project Planning

- Estimation(Cost, Duration, Effort)
- Staffing(Staff organization, Staff plans)
- Scheduling manpower & other resources
- Risk Management
- Miscellaneous Plans (Quality assurance plan, configuration Installation plans)

7:48 / 10:37
Slid

SPM(Software Project Management) with real life examples

Precedence Ordering



9:11 / 10:37
Slid

Risk Identification | Reactive vs Proactive Risk Management |Types of Risks with real life exa...

Risk Management

- A risk is "an uncertain event or condition that, if it occurs, has a positive or negative effect on a project's objectives.
- A risk management plan is a document that a project manager prepares to foresee risks, estimate impacts, and define responses to risks.

3:01 / 9:30
Slid

Risk Identification | Reactive vs Proactive Risk Management |Types of Risks with real life exa...

Reactive vs Proactive Risk Management Strategies

- Reactive risk management tries to reduce the damage of potential threats and speed an organization's recovery from them, but assumes that those threats will happen eventually.
- Proactive risk management identifies threats and aims to prevent those events from ever happening in the first place.

5:20 / 9:30
Slid

Types of Risks

- **Business Risk:** Building a product that no one wants or loosing budgetary commitments.
- **Technical Risk:** Concerned with quality, design, implementation, interface, maintenance problems.
- **Project Risk:** Concerned with schedule, cost, resources, customer-related issues.

Slid

Risk As ent

RM < RA — I, A, P
RC

- It is a pr nk the risks in terms of their damage
- Determin rage probability of occurrence value for each risk.
- Dete for e omponent based on i lue)
- ed by multiplying the y and Severity values together.

Risk Assessment

RM < RA — I, A, P
RC

- It is a process to rank the risks in terms of their damage
- Determine the average probability of occurrence value for each risk.
- Determine the impact for each component based on impact assessment matrix.(Severity Value)
- Risk Assessment values are determined by multiplying the scores for the Probability and Severity values together.

1:47 / 6:10

Risk Assessment with examples | Risk Management | Software Engineering

Risk Assessment

RM < RA — I, A, P
RC

- It is a process to rank the risks in terms of their damage
- Determ[illegible] the average probability of occurrence value for each ri[illegible]
- Dete[illegible]ne the impact for each component based on im[illegible]t assessment matrix.(Severity Value)
- [illegible]k Assessment values are determined by multiplying the [illegible]cores for the Probability and Severity values together.

1:50 / 6:10

Risk Control vs Risk Mitigation with example | Software
Suggested: Risk Identification | Reactive vs Proactive Risk Manag...

Risk Control vs Risk Mitigation

- Risk control is basically the specific actions to reduce a risk event's probability of happening. For example, Correct inspection and maintenance of the car reduce the likelihood of mechanical failures.
- Whereas Risk mitigation is the set of actions to reduce the impact of a Risk Event. For example, Airbags are used to reduce the impact of an accident on the passengers and driver.

0:23 / 5:29
Slid

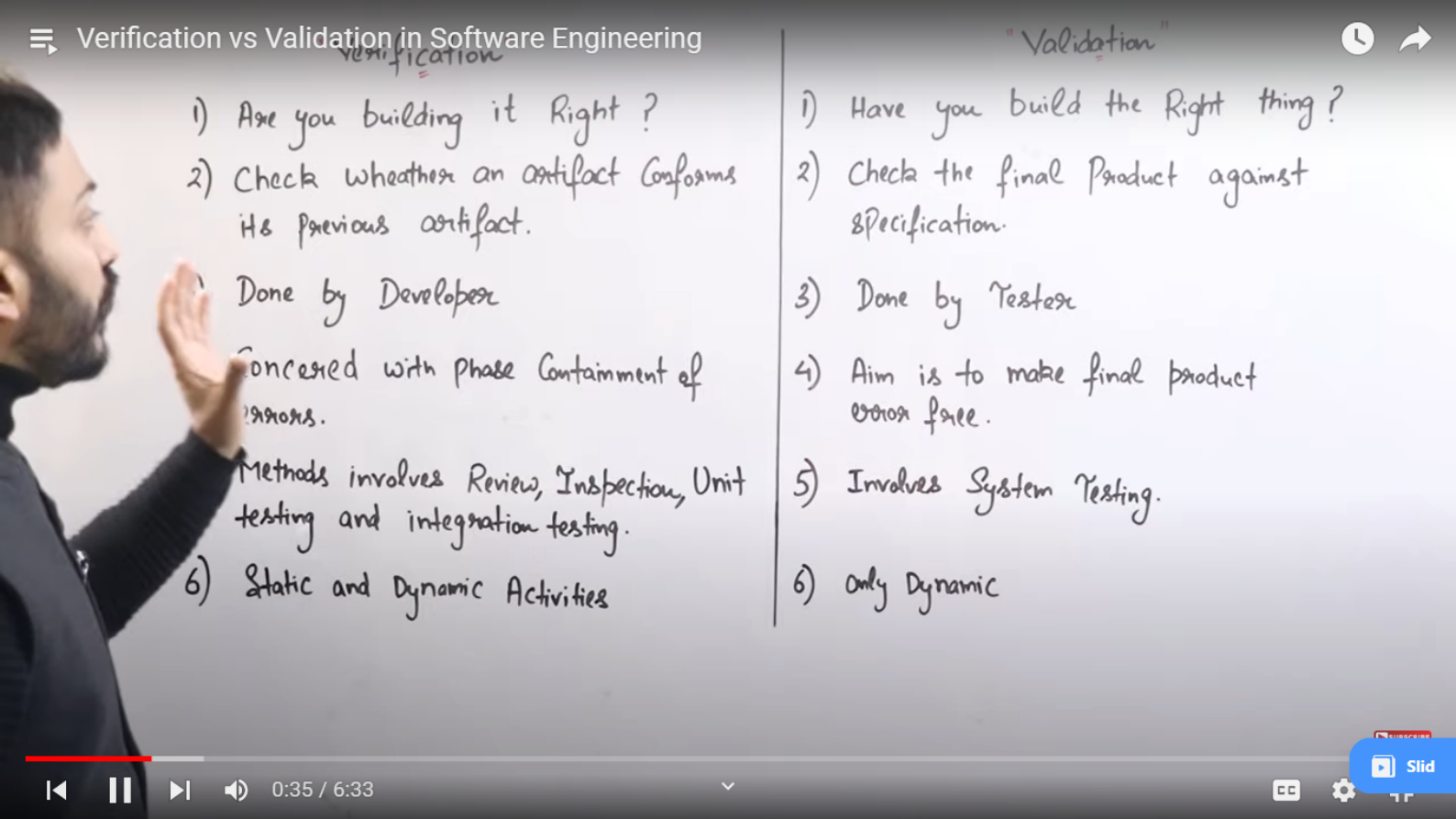
Risk Control vs Risk Mitigation with example | Software engineering

Risk Mitigation & Control Strategies

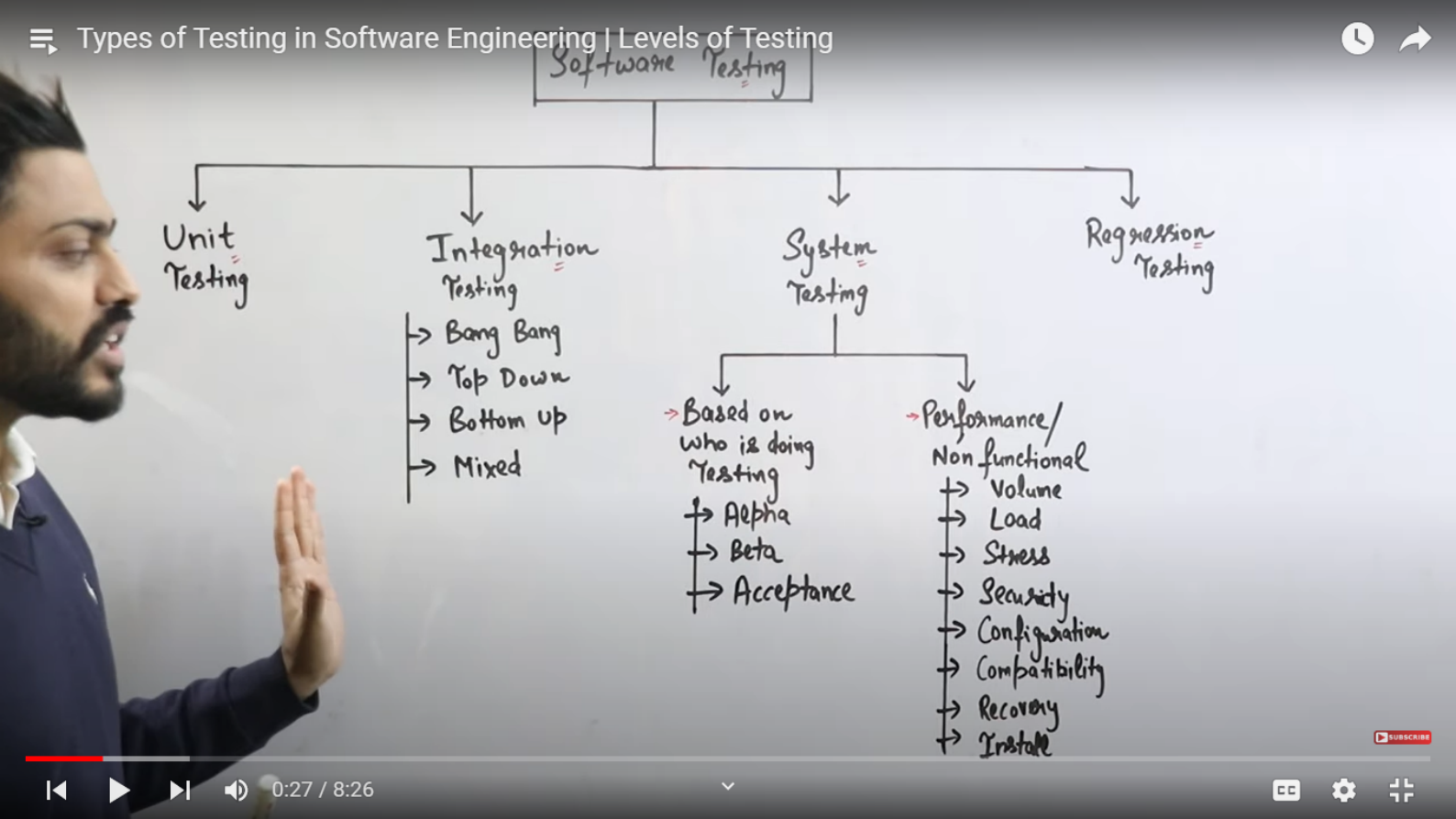
- Risk Avoidance
- Risk Transfer
- Risk Reduction
- Risk Monitoring

3:06 / 5:29
Slid

Verification vs Validation in Software Engineering



Types of Testing in Software Engineering | Levels of Testing



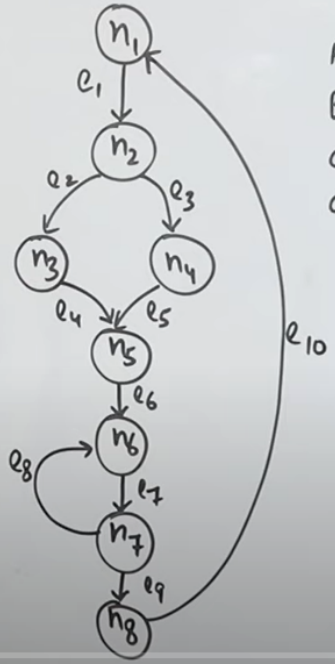
0:27 / 8:26

Que. Which of the following is white box testing technique?

a) Path coverage] white box
b) Equivalence Class testing
c) Cause effect graphing
d) State based testing

] Black box

Que. Find the cyclomatic Complexity of following flow graph of software code?



A) 10
B) 4
C) 8
D) 2

1:40 / 11:03
Slid

Cohesion and Coupling in Software Engineering

Introduction

Design phase transforms SRS document:

into a form easily implementable in some programming language.

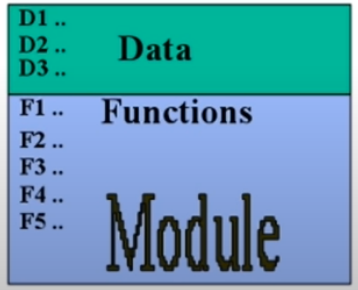
0:18 / 12:46

Cohesion and Coupling in Software Engineering

Introduction

A module consists of:

1. several functions
2. associated data structures.



0:59 / 12:46

Modularity

Modularity is a fundamental attributes of any good design.

Decomposition of a problem cleanly into modules:

1. Modules are almost independent of each other
2. divide and conquer principle.

1:58 / 12:46

Cohesion and Coupling in Software Engineering

Modularity

If modules are independent:

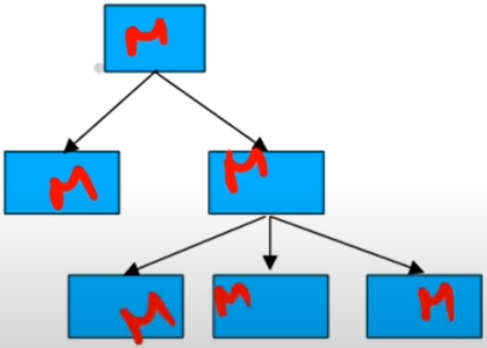
- modules can be understood separately.
- reduces the complexity greatly.

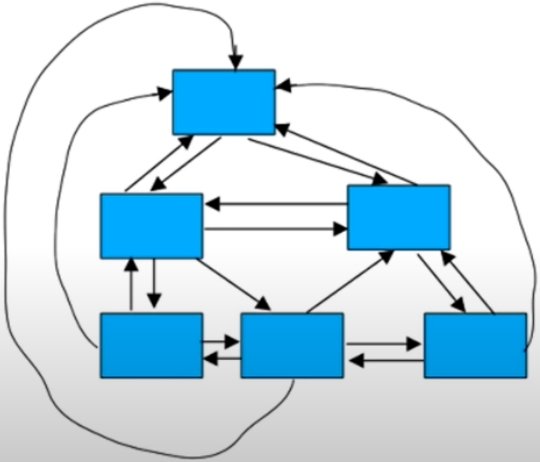
To understand why this is so,

- remember that it is very difficult to break a bunch of sticks but very easy to break the sticks individually.

3:03 / 12:46

Example of Cleanly and Non-cleanly Decomposed Modules





3:18 / 12:46
Slid

Cohesion and Coupling in Software Engineering

Modularity

Cohesion → 1 Mon

In technical terms, modules should display:

1. high cohesion
2. low coupling.

4:01 / 12:46
Slid

Cohesion and Coupling in Software Engineering

Modularity

Cohesion → 1 Module. function.

In technical terms, modules should display:

1. high cohesion
2. low coupling.

4:22 / 12:46

Cohesion and Coupling

Cohesion is a measure of:

- functional strength of a module.
- A cohesive module performs a single task or function.

Coupling between two modules:

- a measure of the degree of interdependence or interaction between the two modules.

Cohesion and Coupling in Software Engineering

Cohesion and Coupling

A module having high cohesion and low coupling:

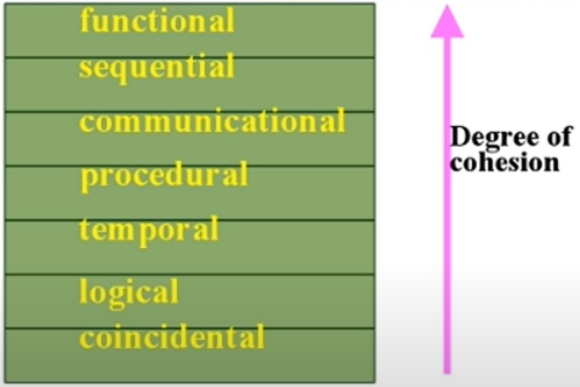
functionally independent of other modules:

A functionally independent module has minimal interaction with other modules.

5:20 / 12:46

Cohesion and Coupling in Software Engineering

Classification of Cohesiveness



5:23 / 12:46

Logical cohesion

All elements of the module perform similar operations:
e.g. error handling, data input, data output, etc.

When logically categorized elements are put together into a module, it is called logical cohesion.

5:37 / 12:46

Coupling

Coupling indicates:

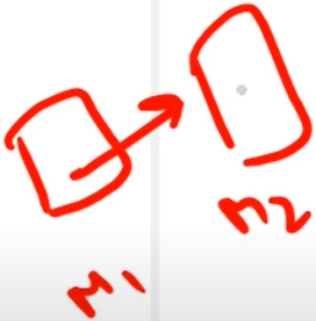
- how closely two modules interact or how interdependent they are.

- The degree of coupling between two modules depends on their interface complexity.

8:34 / 12:46

Cohesion and Coupling in Software Engineering

Classes of coupling



data
stamp
control
common
content

Degree of coupling

8:47 / 12:46

Characteristics of Module Structure

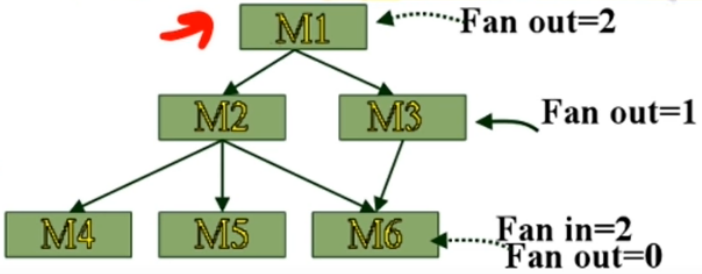
Fan-in:

indicates how many modules directly invoke a given module.

High fan-in represents code reuse and is in general encouraged.

Cohesion and Coupling in Software Engineering

Module Structure



Indegree.
=
Outdegree.

11:29 / 12:46

Cohesion and Coupling in Software Engineering

Characteristics of Module Structure

Depth:
number of levels of control

Width:
overall span of control.

Fan-out:
a measure of the number of modules directly controlled by given module.

12:21 / 12:46

Unit Testing with examples in Software Engineering

Unit Testing

- Unit testing is the process of checking small pieces of code to ensure that the individual parts of a program work properly on their own.
- Unit tests are used to test individual blocks (units) of functionality.
- Unit Testing is done by developers.

3:09 / 3:50
Slid

Integration Testing

- Integration testing is conducted to evaluate the compliance of a system or component with specified functional requirements.
- It occurs after unit testing and before system testing.
- Types of Integration Testing
  1) Big-bang
  2) Mixed (Sandwich)
  3) Top-down
  4) Bottom-up.

0:21 / 5:36
Slid

System Testing with examples | Software Engineering

System Testing

- System Testing is a level of testing that validates the complete and fully integrated software product.
- The purpose of a system test is to evaluate the end-to-end system specifications.
- System Testing is a black-box testing.
- System testing categories based on: Who is Doing the Testing?
- System testing categories based on: Functional/Non-Functional Requirements?

0:26 / 6:57
Slid

System Testing with examples | Software Engineering

System Testing

Unit → Int. → System

m Testing is a level of testing that validates the ete and fully integrated software product.

urpose of a system is to evaluate the end-to-end stem specifications.

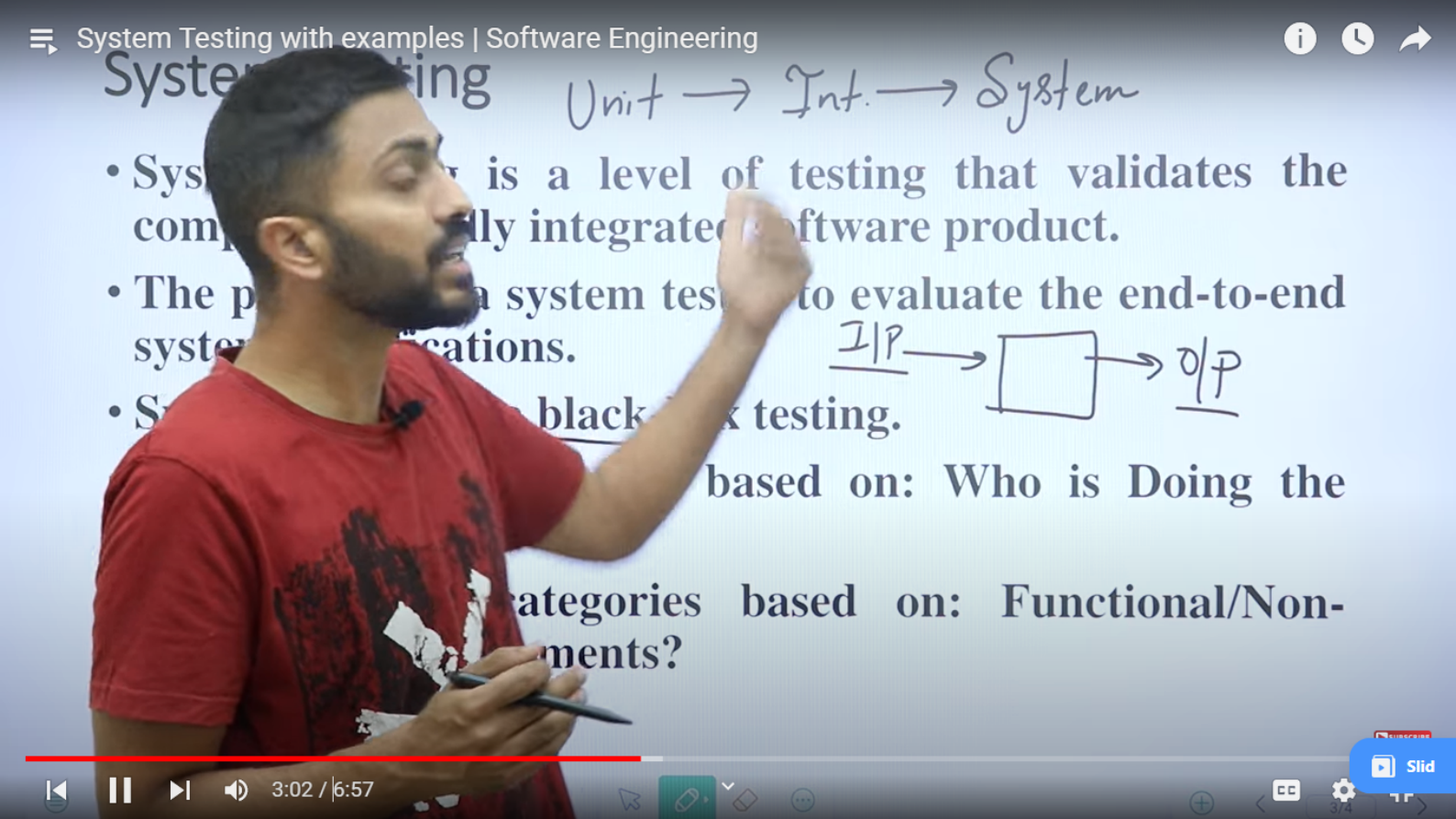
em Testing is a bla box testing.

ies based on: Who is Doing the

esting categories based on: Functional/Non-onal Requirements?

1:11 / 6:57

Slid


System Testing with examples | Software Engineering
Unit → Int. → System
is a level of testing that validates the
ftware product.
I/P → O/P
black testing.
based on: Who is Doing the
ategories based on: Functional/Non-
ments?
3:02 / 6:57
Slid

Types of System Testing | Software Engineering

System Testing

- System testing is testing conducted on a complete integrated system to evaluate the system's compliance with its specified requirements.
- System Testing is basically performed by a testing team that is independent of the development team that helps to test the quality of the system impartial. It has both functional and non-functional testing. System Testing is black-box testing.

0:04 / 7:01
Slid

Suggested: Software Engineering

Type System Testing

- **Perfor sting:** Performance Testing is a type of software testing ried out to test the speed, scalability, stability, and reliabili ftware product or application.
- **Load Te** Load Testing is a type of software testing which is carrie rmine the behavior of a system or software product

ting is a t of software testing performed he syste nder varying loads.

lity T ting is a type of software testing the performance of a software its capability to scale up or scale er loads.

Slid

Types of System Testing | Software Engineering

Types of System Testing

- **Performance Testing:** Performance Testing is a type of software testi g that is carried out to test the speed, scalability, stability, and reli ty of the software product or application.
- **Lo Testing:** Load Testing is a type of software testing which is ca d out to determine the behavior of a system or software product u extreme load.
- **ss Testing:** Stress Testing is a type of software testing performed heck the robustness of the system under varying loads.
- **Scalability Testing:** Scalability Testing is a type of software testing which is carried out to check the performance of a software application or system in terms of its capability to scale up or scale down the number of user request loads.

4:24 / 7:01
Slid

White Box vs Black Box Testing | Software Engineering

White-box testing	Black box testing
The developers can perform white box testing.	The test engineers perform the black box testing.
what the software is supposed to do, also aware of how it does it.	what the software is supposed to do but is not aware of how it does it.
To perform WBT, we should have an understanding of the programming languages.	To perform BBT, there is no need to have an understanding of the programming languages.
In this, we will look into the source code and test the logic of the code.	In this, we will verify the functionality of the application based on the requirement specification.
In this, the developer should know about the internal design of the code.	In this, there is no need to know about the internal design of the code.
Test design techniques:Control flow testing, Data flow testing, Branch testing, Statement coverage, Decision coverage, Path testing.	Test design techniques: Decision table testing, All-pairs testing, Equivalence partitioning, Boundary value analysis, Cause–effect graph
Can be applied mainly at unit testing level but now in integration, system level also.	Can be applied virtually to every level of software testing: unit, integration, system and acceptance

0:25 / 5:45

White box Testing with example | Software Engineering

White-box testing

- White-box testing (also known as clear box testing, glass box testing, transparent box testing, and structural testing) is a method of software testing that tests internal structures or workings of an application.
- White-box testing can be applied at the unit, integration and system levels of the software testing process.
- Here are some of the top white box testing tools to use: Veracode, CppUnit, Nunit, RCUNIT etc.

0:23 / 4:28
Slid

White box Testing with example | Software Engineering

White-box test design techniques

- Control flow testing
- Data flow testing
- Branch testing
- Statement coverage
- Decision coverage
- Path testing

2:47 / 4:28
Slid

Statement Coverage Technique | White Box Testing | Software Engineering

Statement coverage technique

Statement coverage technique is used to design white box test cases.

- This technique involves execution of all statements of the source code at least once.
- It is used to calculate the total number of executed statements in the source code out of total statements present in the source code.
- This technique covers dead code, unused code and branches.

0:06 / 7:23
Slid

Statement Coverage Technique | White Box Testing | Software Engineering
Statement cover chnique
• Statement coverage te s used to design white box test cases.
• This technique involve n of all statements of the source code at least once
• It is used to c of executed statements in the source co sent in the source code.
• This techniqu de and branches.
0:25 / 7:23
Slid

Statement Coverage Technique | White Box Testing | Software Engineering

```
1. main()
2. {
3. int n1, n2, n3;
4. if (n1 >= n2 && n1 >= n3)
5.     printf("%d  n1 is the largest number.", n1);
6. else if (n2 >= n1 && n2 >= n3)
       printf("%d  n2 is the largest number.", n2);

       printf("%d n3 is the largest number.", n3);
10. }
```

n_1, n_2, n_3

30, 20, 10

$30 > = 20$, $30 > 10$ 10

T

3:03 / 7:23

Slid

Condition Coverage in White Box Testing | Software Engineering

Condition Coverage technique

- It is used to cover all conditions.
- Condition coverage is also known as Predicate Coverage in which each one of the Boolean expression have been evaluated to both TRUE and FALSE.

0:47 / 6:59
Slid

Condition Coverage in White Box Testing | Software Engineering

```
read a, b, c;
if(a==0|| b==0)
{
   print 1;
}
else
{
   if(  &&  d==0)
   {
      nt 2;
   }
}
```

1:12 / 6:59

Slid

Data Flow Testing Technique in White Box Testing | Software Engineering

Data Flow Testing

- It is one of the white-box testing techniques.
- Dataflow Testing focuses on two points:
  1) In which statement the variables are defined.
  2) In which statement the variables are used.
- It designs the test cases that cover control flow paths around variable definitions and their uses in the modules.

0:34 / 5:28
Slid

Data Flow Testing Technique in White Box Testing | Software Engineering

Example

```
1. read a, b, c;
2. if(a>b)
      x = a+1
4     print x;
   lse
      x = b-1
      print z;
```

	Define	Use
a		
b		
c		
x		
z		

2:30 / 5:28
Slid

Example

30 20 10

1. read a, b, c;

30 20

2. if(a>b)

3. x = a+1

4. print x;

else

5. x = b-1

6. print z;

	Define	Use
a	1	2, 3
b	1	2, 5
C	1	NA
X	3, 5	4
Z	NA	6

4:11 / 5:28

Slid

Data Flow Testing Technique in White Box Testing | Software Engineering

Advantages of Data Flow Testing

- A variable that is declared but never used within the program.
- A variable that is used but never declared.
- A variable that is defined multiple times before it is used.
- Deallocating a variable before it is used.

4:23 / 5:28
Slid

Boundary Value Testing | Black Testing | Software Engineering with Real Life examples

Boundary Value Testing

- The black box testing techniques are helpful for detecting any errors or threats that happened at the boundary values of valid or invalid partitions rather than focusing on the center of the input data.

0:05 / 6:12

Boundary Value Testing | Black Testing | Software Engineering with Real Life examples

Test Case Number #1

Let us assume a test case that takes the speed of a car from 40 to 80 to get the best fuel efficiency.

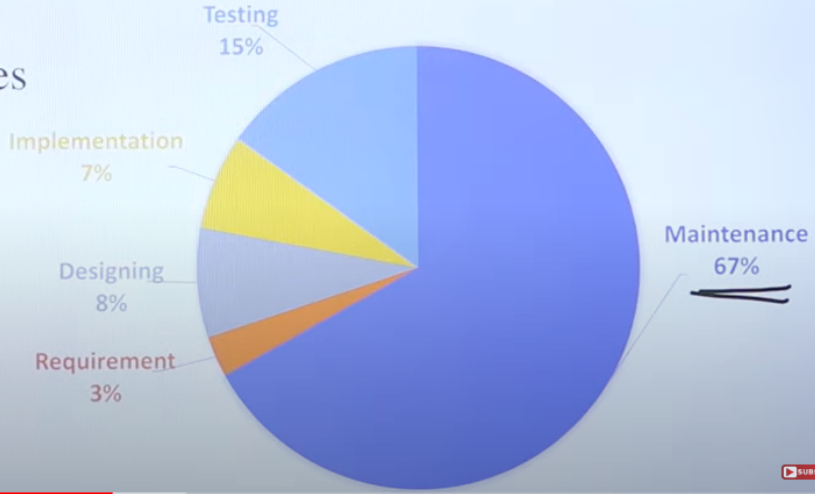
Boundary Value Test Case		
Invalid Test Case (Min Value – 1)	Invalid Test Case (Min, +Min, Max, –Max)	Invalid Test Case (Max Value + 1)
39	40,41,79,80	81

1:55 / 6:12

Perfective, Preventive, Adaptive, Corrective Maintenance in Software Engineering

Software Maintenance

- Its primary goal is to modify and update software applications after delivery to correct errors and improve performance.
- Software Maintenance is an inclusive activity that includes:
  - Error corrections
  - Deletion of obsolete capabilities
  - Enhancement of capabilities
  - Optimization

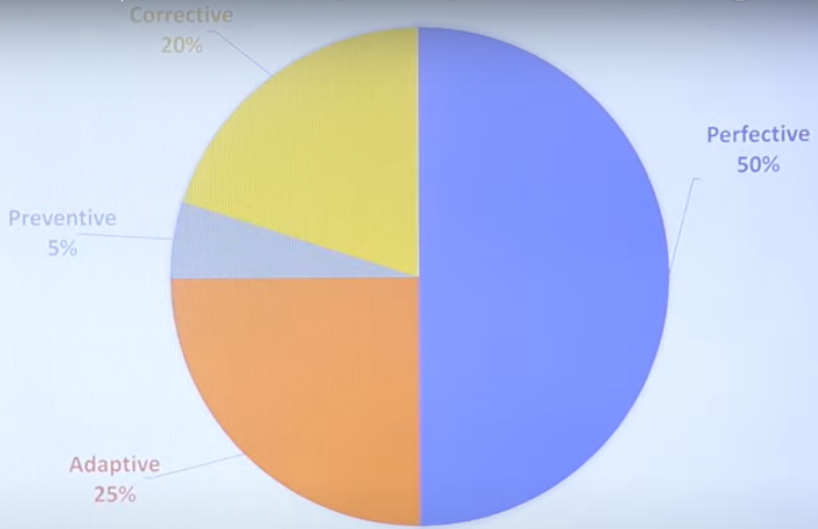


4:48 / 9:11

Perfective, Preventive, Adaptive, Corrective Maintenance in Software Engineering

Types

1. Corrective Maintenance
2. Adaptive Maintenance
3. Preventive Maintenance
4. Perfective Maintenance

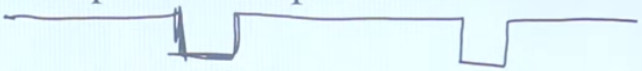


6:13 / 9:11

MTBF vs MTTR | Mean Time Between Failure | Mean Time to Repair

Maintenance Metrics

1. Mean Time Between Failure (MTBF): It is the average time available for a system or component to perform its normal operations between failures.



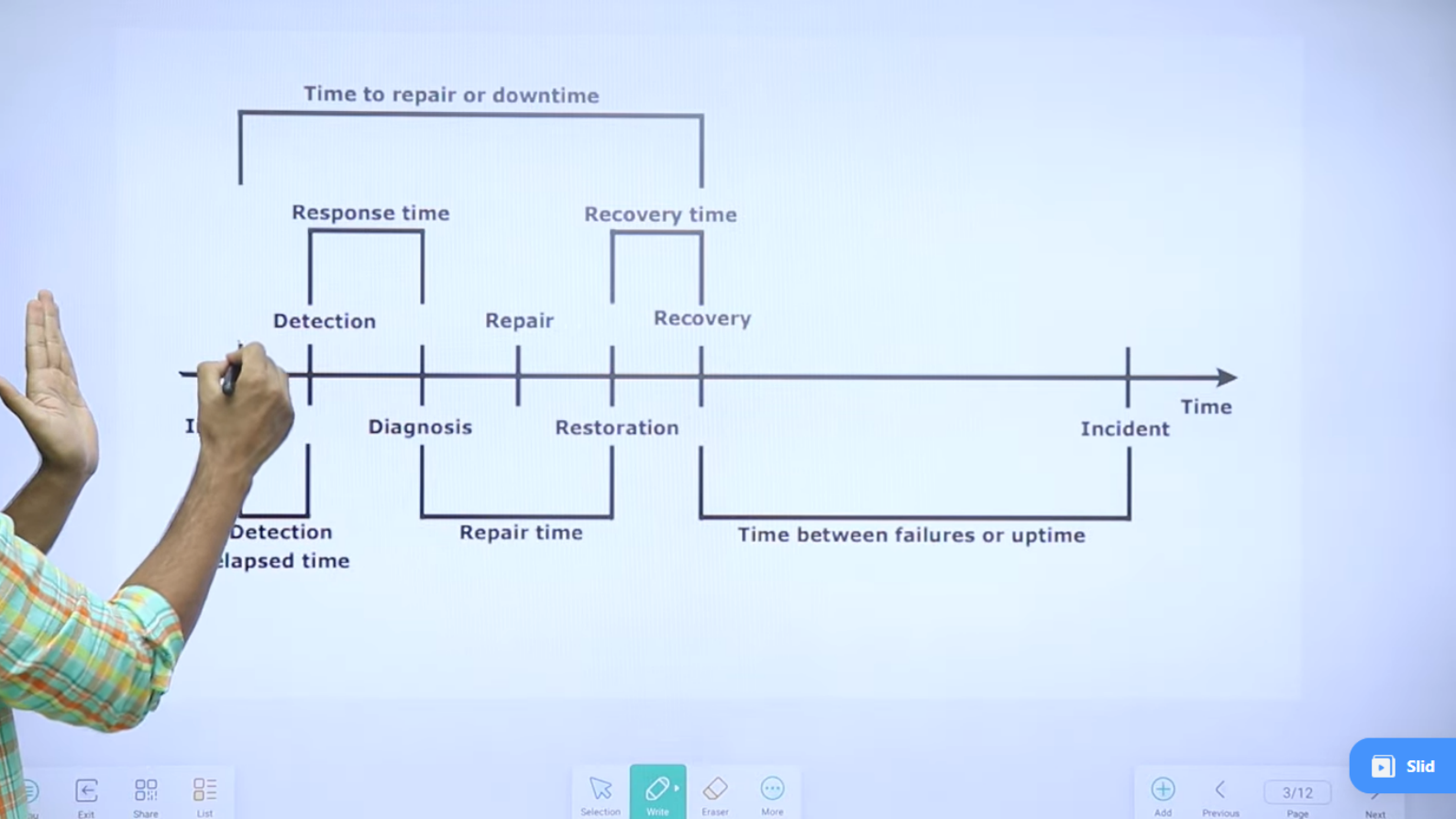
- MTBF= (SUM of operational time/total number of failures)

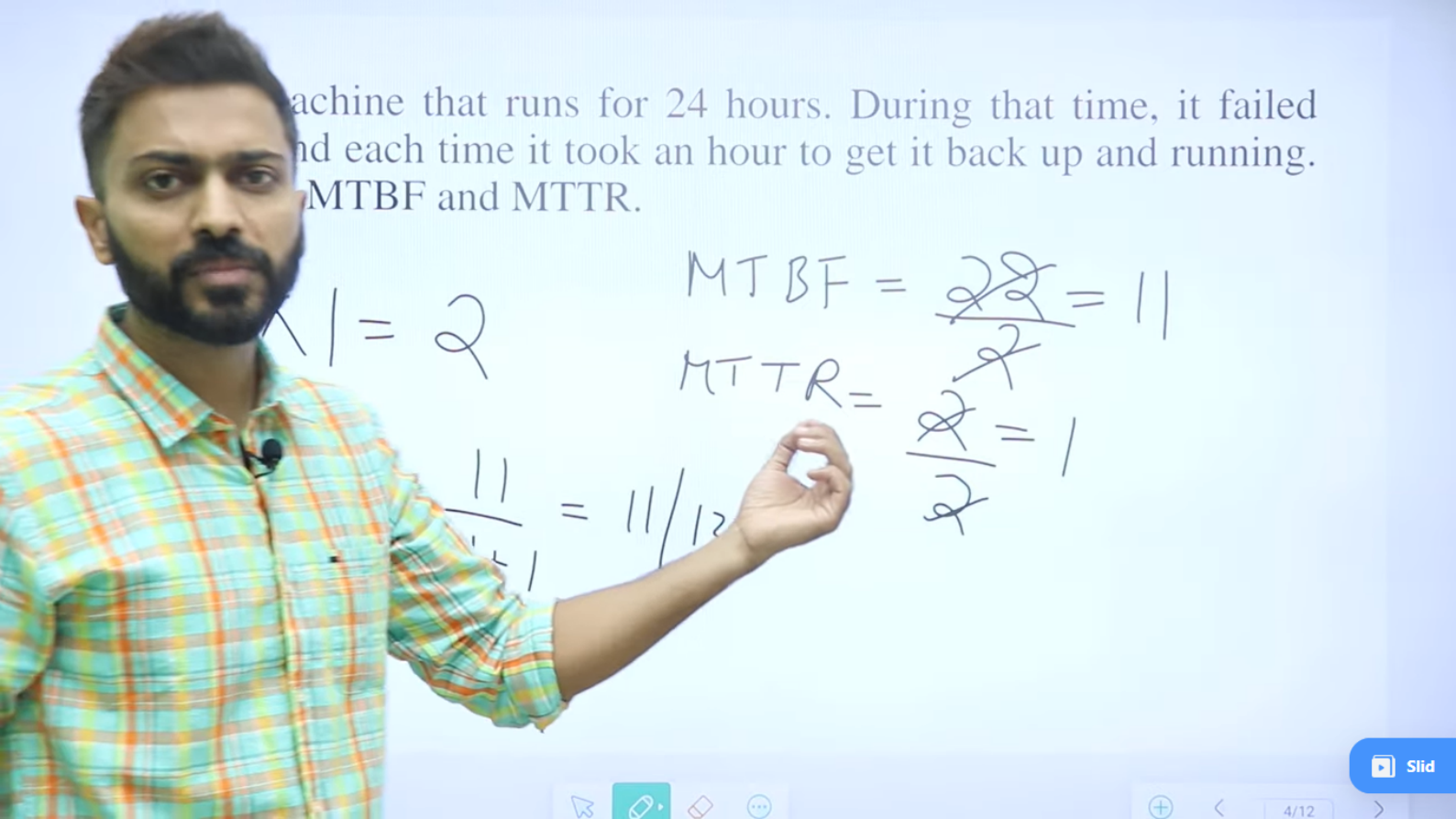
2. Mean Time To Repair (MTTR): It is the average time required to repair a failed component.

- MTTR= (SUM of downtime periods/ total number of failures)

Availab = MTBF/(MTBF + MTTR)

1:51 / 5:47
Slid


Time to repair or downtime
Response time
Recovery time
Detection
Repair
Recovery
Time
Diagnosis
Restoration
Incident
Detection
elapsed time
Repair time
Time between failures or uptime
Exit
Share
List
Selection
Write
Eraser
More
Add
Previous
3/12
Page
Next
Slid


achine that runs for 24 hours. During that time, it failed
nd each time it took an hour to get it back up and running.
MTBF and MTTR.
MTBF = 22/2 = 11
MTTR = 2/2 = 1
11/12
4/12
Slid

Reverse Engineering with Real life examples | Software Engineering

Reverse Engineering/ Backward Engineering

- Software reverse engineering can help to improve the understanding of the underlying source code for the maintenance and improvement of the software.
- In some cases the goal of the reverse engineering process can simply be a redocumentation of legacy systems.

Code → Module Specification → Design → Requirement Specifications

1:12 / 6:52

Usage of Reverse Engineering

- Malware developers often use reverse engineering techniques to find vulnerabilities in an operating system to build a computer virus that can exploit the system vulnerabilities.
- Reverse engineering is also being used in cryptanalysis to find vulnerabilities in substitution cipher, symmetric-key algorithm or public-key cryptography.

SUBSCRIBE

Regression Testing with Real Life Examples | Software Engineering

Regression Testing

Regression testing is a type of software testing that verifies that changes made to the system, such as bug fixes or new features, do not impact previously working functionality.

1:42 / 4:32
Selection Write Eraser Clean Ruler Shape Form Mind map Note Move Grid Undo Add 2/3
Slid

There are several types of regression testing, including:

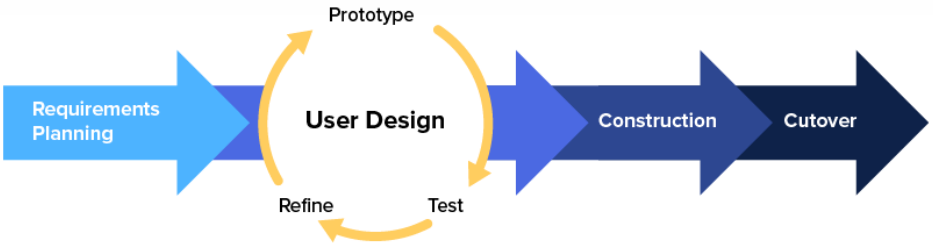
Full Regression Testing: Testing the entire application from start to finish after changes have been made.

Partial Regression Testing: Testing only those parts of the application that were affected by the changes.

3:30 / 4:32
Selection Write Eraser Clean Ruler Shape Mind map Note Move Grid Undo Add
3/3
Slid

Rapid Application Development (New Tab
https://kissflow.com/application-development/rad/rapid-application-development/
Code Faster with AI... CSS reference - CSS... HTML Color Picker:... Learn JavaScript & j... 12 Tips For Naming... html - How to keep... Settings - Passwords How to Get Startup...

Rapid Application Development (RAD)



Type here to search
ENG 07:14 25-04-2023

customer feedback	Cannot work with large teams
Reviews are quick	Needs highly skilled developers
Development time is drastically reduced	Needs user requirements throughout the life cycle of the product
More productivity with fewer people	Only suitable for projects which have a small development time
The time between prototypes and iterations is short	More complex to manage when compared to other models
Integration isn't a problem, since it integrates from project inception	Only systems which can be modularised can be developed using Rapid application development.

Advantages and disadvantages of the Rapid Application Development (RAD) model

With these steps, it can seem like application development's a great idea for all projects, but that's a stretch. RAD software is great for small teams and quick projects. But it's not a solution to everything. Here are a few advantages and disadvantages of using rapid application development.

Advantages of RAD model	Disadvantages of RAD model
Requirements can be changed at any time	Needs strong team collaboration
Encourages and priorities customer feedback	Cannot work with large teams
Reviews are quick	Needs highly skilled developers
Development time is drastically reduced	Needs user requirements throughout the life cycle of the product