FPT Education

FPT UNIVERSITY

SOFTWARE ENGINEERING (SWE201C)

SOFTWARE DEVELOPMENT LIFECYCLE SPEC

COURSES SUMMARY

C1 - SW Dev Processes & Methodologies

- After completing this course, a learner will be able to
  - Apply core software engineering practices at conceptual level for a given problem.
  - Compare and contrast traditional, agile, and lean development methodologies at high level. These include Waterfall, Rational Unified Process, V model, Incremental, Spiral models and overview of agile mindset
  - Propose a methodology best suited for a given situation
- Course Modules
  - W1. SW Dev Process Part 1
  - W2. SW Dev Process Part 2
  - W3. SW Dev Models: Traditional Models
  - W4. SW Dev Models: Agile & Lean

C1 - SW Dev Processes & Methodologies

W1. Software Development Processes 1/2

- Learning Objectives
  - Interpret the given situation and recommend the missing engineering practices causing that situation
  - Select the appropriate engineering practices for a given situation
  - Describe the key engineering practices and their purpose
- What software development looks like
- Requirements
  - Why do we need requirements?
  - Requirements vs Specification: Problems vs Solution
  - Non-functional Requirements
  - WRSPM (World | Requirement | Specification | Program | Machine)
- Architecture
  - Software Architecture: Definition
  - Software Architecture: Models
  - Software Architecture: Process

C1 - SW Dev Processes & Methodologies

W2. Software Development Processes 2/2

- Learning Objectives
  - Describe the key engineering practices and their purpose
  - Select the appropriate engineering practices for a given situation
  - Interpret the given situation and recommend the missing engineering practices causing that situation
- Design
  - Software Design: Introduction
  - Software Design: Modularity – Coupling, Cohesion, Information Hiding, Data Encapsulation
- Implementation
  - Implementation
  - Deployment
  - Deployment: Rollback
  - Deployment: Cutover Strategies
- Test & Verification: strategies, perspectives

C1 - SW Dev Processes & Methodologies

W3. Sw Development Models - Traditional

- Learning Objectives
  - Apply a selected traditional software development methodology for a given situation
  - Describe the key characteristics; pros and cons; and recommended use of traditional software development models
  - Compare and contrast waterfall, incremental and iterative models
- Waterfall Models
  - Software Development Models
  - Waterfall Model: V-Model, Sashimi Model
- Incremental Models
- Iterative Models: Unified Process, Spiral Model
- Applying traditional software development models
  - Phase Gates / Stage Gates
  - Applying Software Development Models

- Learning Objectives
  - Apply a selected agile software development methodology for a given situation
  - Recommend a software development methodology best suited for a given situation
  - Compare and contrast traditional, agile, and lean development methodologies at high level.
- Agile Fundamentals
- Agile Frameworks
  - Agile Frameworks
  - Scrum
  - Kanban
  - Agile and Lean Summary
  - Lean Startup

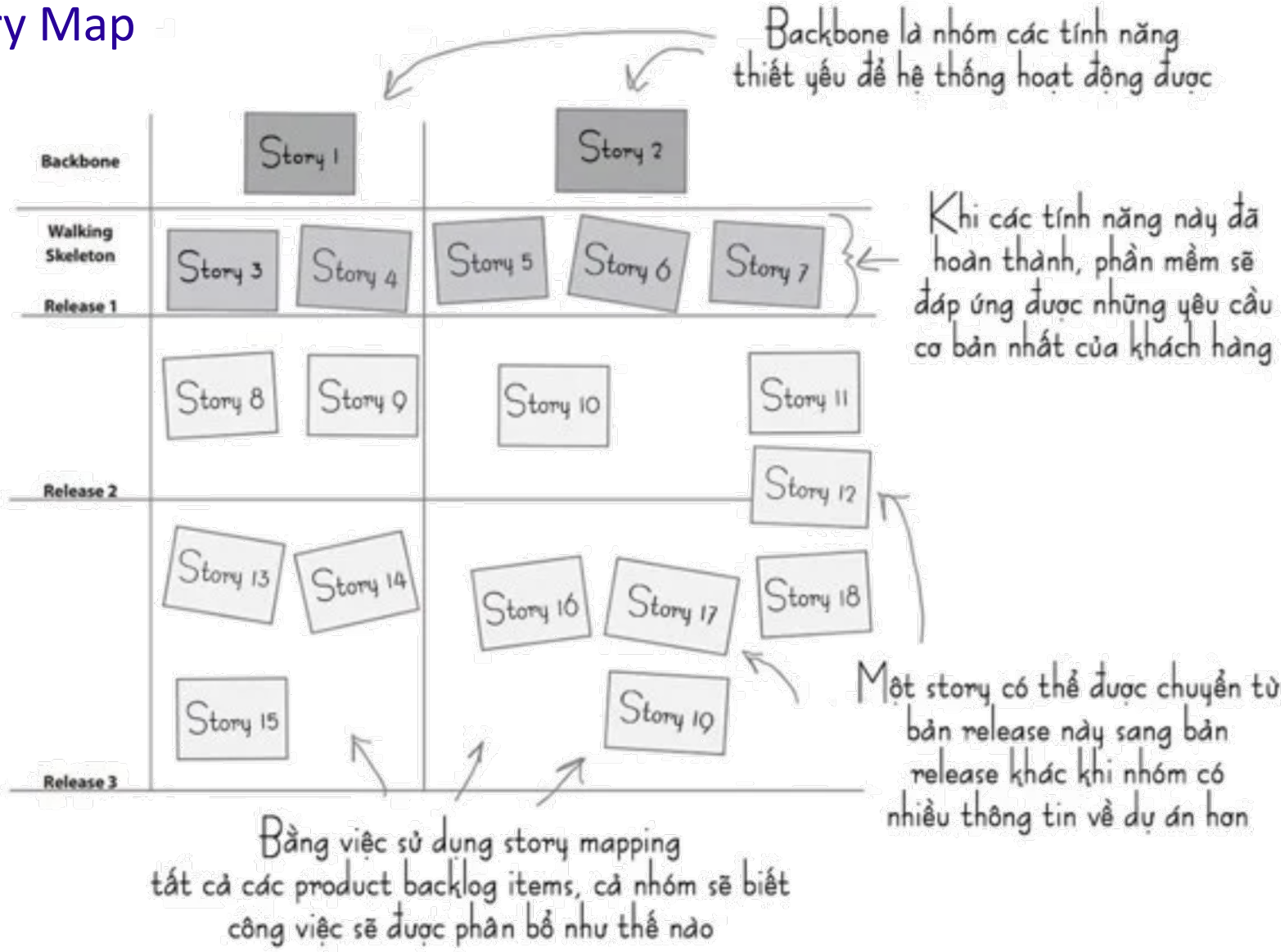
C2 - Agile Software Development

- After completing this course, you will be able to :
  - Demonstrate the ability to participate effectively in agile practices/process for software development.
  - Explain the purpose behind common agile practices.
  - Ability to apply agile principles and values to a given situation.
  - Ability to identify and address most common problems encountered in adopting Agile methods.
- Course Modules
  - W1. Agile Fundamentals
  - W2. Requirements & Planning
  - W3. Scrum
  - W4. XP & Course Wrap-up

- Learning Objectives
  - Explain Agile Values and Principles
  - Analyze a given situation and classify if it aligns with Agile principles or not
  - Describe pros and cons, usage and application of Agile Methods
  - Explain how agile mindset can be applied to build software
- Course Outline
  - What software development looks like?
  - Sw Dev Models: predictive, adaptive, iterative, incremental
  - Agile Values & Principles
  - Applying Agile Mindset

- Learning Objectives
  - Demonstrate the ability to gather user needs and requirements using agile techniques like story mapping
  - Demonstrate the ability to plan and track a release on an agile project
  - Ability to write good user stories
  - Critique a user story and make recommendation for correction
  - Demonstrate the ability to participate in agile estimation process
- Module Outline
  - User Stories and Requirements Gathering
  - Agile Estimation and Planning
    - Agile Estimation and Planning
    - Estimation Styles and Process
    - Velocity
    - Release Planning
    - Release Tracking

- Story Map


Backbone là nhóm các tính năng thiết yếu để hệ thống hoạt động được
Backbone
Story 1
Story 2
Walking Skeleton
Release 1
Story 3
Story 4
Story 5
Story 6
Story 7
Khi các tính năng này đã hoàn thành, phần mềm sẽ đáp ứng được những yêu cầu cơ bản nhất của khách hàng
Story 8
Story 9
Story 10
Story 11
Release 2
Story 12
Story 13
Story 14
Story 16
Story 17
Story 18
Story 15
Story 19
Release 3
Một story có thể được chuyển từ bản release này sang bản release khác khi nhóm có nhiều thông tin về dự án hơn
Bằng việc sử dụng story mapping
tất cả các product backlog items, cả nhóm sẽ biết
công việc sẽ được phân bổ như thế nào

- Learning Objectives
  - Demonstrate the ability to participate in practices/process of building software using Scrum
  - Explain how Scrum aligns with Agile Values and Principles
  - Demonstrate the ability to participate in Sprint Planning and Tracking activity
  - Analyze a given situation and make a case to use Agile
- Module Outline
  - XP Overview
  - Sprint Planning and Tracking
  - Sprint Review, Retrospective and Sprint Execution

C2 - Agile Software Development

W4. XP (Extreme Programming)

- Learning Objectives
  - Demonstrate the ability to participate in practices/process of building software using XP
  - Explain the difference between Scrum and XP
  - Demonstrate ability to apply agile mindset and key agile techniques to a given situation
- Module Outline
  - XP Overview
  - XP Values
  - XP Practices
  - XP Process Model

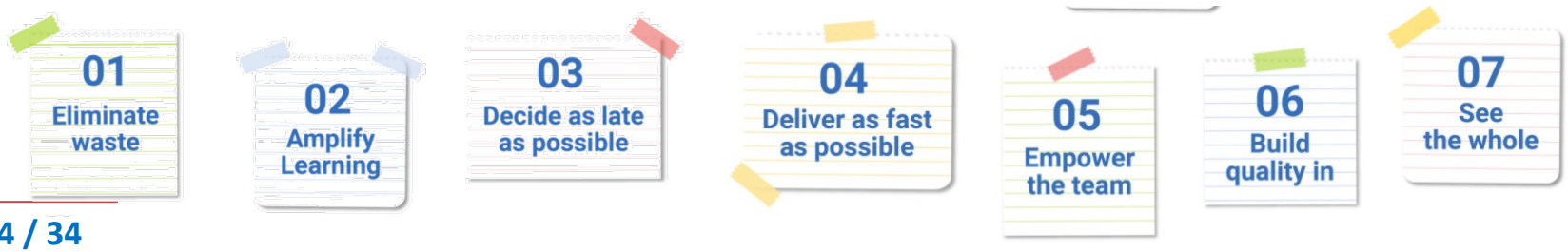
C3 - Lean Software Development

- Course Modules
  - W1. Lean Fundamentals
  - W2. Kanban, Value Steam Mapping & Kaizen
  - W3. Lean Startup
  - W4. Design Thinking

C3 - Lean Software Development

W1. Lean Fundamentals

- Learning Objectives
  - Explain the core principles of Lean
  - Explain the lean principles for software development
  - Explain the various tools, techniques and practices for lean software principles
- Introduction to Lean
  - What do we mean by Lean?
  - Five Principles of Lean Manufacturing
  - Lean for software development
- Lean Principles for software development



C3 - Lean Software Development

W2. Kanban, Value Steam Mapping & Kaizen

- Learning Objectives
  - Demonstrate the ability to participate in practices/process of building software using Kanban
  - Apply Value Stream Mapping concepts on a given situation
  - Apply Lean tools and practices to improve software development processes
- Module Contents
  - Kanban for Software Development
  - Lean Metrics : Tracking Flow Based Methods like Kanban
  - Value Stream Mapping
  - Kaizen
    - Intro to Kaizen
    - The 5 Whys

- Learning Objectives
  - Explain the principles of Lean Startup
  - Apply the Lean Startup Concepts to a given business case
  - Create meaningful metrics to validate assumptions behind a business case
- Module Contents
  - Lean Startup Principles
  - Lean Startup Principles
  - Innovation Accounting

- Learning Objectives
  - Apply the Design Thinking concept for product development
  - Analyze the given situation and Identify key assumptions behind the given business case
  - Identify MVP Tests to validate key assumptions behind the given business case
- Module Contents
  - Design Thinking
  - Lean Startup with Design Thinking
  - Identifying and Classifying Assumptions
  - Prototype and Test

C4 - Engineering Practices for Building Quality Software

- Course Modules
  - W1.1 Introduction to Quality Software
  - W1.2 Quality in Design
  - W2. Quality in Architecture
  - W3. Quality in Implementation
  - W4. Quality in Testing & Deployment

- What is Quality Software?

The standard of something as measured against other things of a similar kind; the degree of excellence of something.

Of good quality; excellent.

- Quality throughout the engineering process
  - Design
  - Architecture & Security
  - Implementation
  - Testing & Deployment

- What is good design?
  - Quality Attributes: Performance, Security, Modifiability, Reliability, Usability
  - Software quality criteria
    - Coupling: the level of dependency between two entities
    - Cohesion: how well an entity's components relate to one another
    - Liskov's Substitution Principle
    - SOLID
    - Law of Demeter
- Quality Metrics
- Software Design Patterns

S	Single responsibility principle
O	Open/closed principle
L	Liskov substitution principle
I	Interface segregation principle
D	Dependency inversion principle

- Learning Objectives
  - Describe applications of software architecture.
  - Contrast architectural styles and evaluate their applicability to software projects.
  - Appraise software quality from multiple architectural views.
- Module Outline
  - What is Software Architecture?
  - ISO/IEC/IEEE Systems and software engineering — Architecture description
  - **Architectural Styles**
  - View, Viewpoint, and Perspective
  - Writing Scenarios
  - Security as an Architectural Concern
    - Security Perspective
    - Attack Trees
    - Security Tactics

- Learning Objectives
  - Conform software development to established coding standards.
  - Compare code style alternatives.
  - Recognize benefits of manual and tool-supported debugging approaches.
- Module Outline
  - Code Style
  - Debugging & Static Analysis
  - Comment & Self-Documentation
  - Version Control & Build Process
    - Version Control Systems
    - Git and GitHub
    - Build Process
    - Intro to Make
    - A closer look at Apache Ant
    - Gradle
    - Comparison: Ant, Maven, and Gradle

- Learning Objectives
  - Review the difficulties of selecting effective tests.
  - Explain how code coverage can be used for test selection.
  - Explain how code coverage can be used in measure testing adequacy.
- Module Outline
  - Test Selection
  - More Details of Code Coverage: Branch, Statement, Decision, FSM
  - Minimum Acceptable Code Coverage
  - Test Adequacy
  - Test-Driven Development
  - Deployment
    - Continuous Integration: Jenkins, SonarQube
    - Continuous Delivery / Continuous Deployment


Q
A