Agile development life cycle and its advantages

Life cycle: concept (define goal), planning (backlog), Iteration/development, release.

Advantages: Flexibility, Fast deployment (market benefits), less overhead, improved quality (less bugs, small releases, Continuous integration), communication between teams, Continuous Improvement

User understanding and how do we gain user understanding.

Interview users, involve users in development, prototyping, feedback

PRESONAS, SCENARIOS, STORIES

Personas

- **Definition**
- **which information should be included in a persona description**
- **the benefits of persona** –> Put the developer in the shoes of the user

Scenario

- **definition**
- **the most important elements that may be included in a scenario description**
- **how can users get involved in the development of scenarios**

User stories

- **Definition**
- **a typical form of a user story and its constituent parts**
- **How personas, scenarios and user stories can contribute to feature design.**

Software features

- **The most important characteristics of software features**
  - Independence, relevance, coherence
- **Six important factors in feature set design**
  - Functionality and simplicity
  - Familiarity and novelty
  - Automation and control
- **'feature creep'**
  - Bloated software, too many features
- **How to identify features from a scenario description**
  - Look for buzz words (need, uses, want)
- **How features can be described and defined using user stories**
- **Why user research on its own is not necessarily sufficient when designing software**

Products

Definition and roles of software architecture

The organization of components and their relationship to each other and environment, and the design of how users can interact with it.

Decision types about architectural design

- Non functional attributes and their tradeoffs
- Software reuse
- Lifetime – evolve
- Number of users – scalavlility
- Software compatablitity

Non-functional attributes and examples

- Relislience
- Availability
- Scalability
- Usability
- Security
- Maintainability
- Responsiveness

Software attribute trade-offs

System decomposition (service, component, modules)

- Service: self-contained units of functionality that perform specific business functions. service-oriented architecture (SOA) and microservices architecture.
- Components: Components are cohesive units of functionality within a system. They encapsulate a specific piece of functionality and interact with other components via interfaces.
- Modules: organizational units within a codebase, often used to group related classes, functions, and resources. They typically correspond to a single file or a group of files within a directory

Design guidelines

- Implement once
- Separation of concern
- Stable interfaces
- Minimize complexity
- High cohesion and low coupling
- Reusability

Object orientation

CLOUD-BASED SOFTWARE

Its characteristics: Scalability, Elasticity and Resilience

Virtual server and Container

SaaS: pros and cons

- **Pros**
  - Continuous cashflow
  - Low startup time
  - Server choice
  - Distributed development
- **Cons**
  - Security
  - Must be on internet.
  - Can choose when to update (benefit for company)
  - User data for users

Multi-tenant vs Multi-instance

SLA: The provider can guarantee, penalties of not providing.

MICRO-SERVICE

A software architecture style that focuses on

independence of teams

dynamic run-time scaling and failure handling

• Key technology is independent service deployment

Design questions

- How to communicate
- Failure management
- Data be distributed and shared
- What are the microservices that make the system

Challenges

- Data consistency
- Design

8. RELIABLE PROGRAMMING

The major quality attribute groups and what attributes are in each group

- Reliability: reliability, availability, security, and resilience.
- User experience: responsiveness and usability
- Maintainability.

Techniques for software reliability improvement

- Fault avoidance
- Input validation
- Failure management

Underlying causes of program errors

- Programmers does not understand problem
- Programmers does not understand technology
- Programmers makes mistakes in programs

Program complexity

- Reading
- Data
- Decision
- Structure

increased complexity leads to program errors

the three main types of program complexity

- Data
- Decision
- Structure

complexity reduction guidelines for each of program complexity types

structural complexity

- Functions should do one thing and one thing only.
- Functions should never have side effects.
- Every class should have a single responsibility.
- Minimize the depth of inheritance hierarchies.
- Avoid multiple inheritance.

Data complexity

- Define interfaces for all abstractions.
- Define abstract data types.
- Avoid using floating-point numbers.
- Never use data aliases.

Decision complexity

- Avoid deeply nested conditional statements.
- Avoid complex conditional expressions

Design patterns

Program refactoring

Code smells, definition and examples

Input validation

Number checking

The three most important categories of software failure

- Data failure
- Timing failure
- Program failure

A program ‘exception’ and ‘exception handler’

Mechanisms that can be used to help users recover work after a system failure.

- Auto saving
- Activity logging

9. TESTING

Two main cause of program bugs

- Program errors
- Understanding errors

Functional testing

- **Four types of functional testing**
  - Unit
  - Feature
  - System
  - Release
- **Unit testing, its underlined principals, and unit testing guidelines**
  -
- **Feature testing and its two types of test**
- **System testing and its four concerns**
- **Release testing**

• Concept of an end-to-end pathway

• ‘executable test’ and its commonly-used structure