Concepts of Software Engineering theory

**Requirements**

Definition and formulation of natural language requirements

Elicitation

* Interviews
* Observation
* Prototyping

Analysis

* Requirements classification
  + User (High-level) vs. System (Low-level) requirements
  + Functional requirements vs Non-functional requirements
  + Quality requirements
  + Design constraints
  + Prioritization
  + Best practices for writing requirements
  + Modelling requirements
  + Use-cases
  + Actor
  + UML Use-case diagram (one diagram can contain several use-cases)
  + UML Class diagram
  + UML Activity diagrams
* Stakeholders

Requirements Specification

* IEEE Std 830
* Features
* Agile requirements practices
* User stories
* Behavior driven development
* Formalization

Validation & verification

Software quality factors

**Design and Architecture**

System architecture (vs. Software architecture)

* Functional models
* Platform models
* Allocation

Why design an architecture? -Communication, Early Decisions, Reuse

Prototyping

* Throw away
* Evolutionary

Box-and-line diagram

Views

* Implementation (code) View
* Execution View
* Deployment View

Relation Architecture – quality factors

* Coupling – Cohesion
* Performance
* Security
* Safety
* Modifiability
* Usability
* Testability
* Business quality

Architecture styles

* Client-server, 3 variants
* Layered
* Pipe-and-filter
* SOA – Service Oriented Architecture
* Model-view-controller
* Examples of architectural styles in modern multi-tier applications

What and when to document

The Architecture notebook

UML purpose sketching, blueprinting, programming language

* class diagram: attributes, association. composition, generalization, realization, dependency
* sequence diagram
* state machine diagrams
* (activity diagrams) (don’t mix up with state diagrams)

Design patterns

* Strategy
* Observer
* Façade

**Testing and SCM**

Intention of testing

Other methods for V&V

Error, Fault. Failure

Sins of omission and sins of commission

Types of faults

Black-box testing

* Exhaustive testing
* Equivalence class testing
* Boundary value testing

White-box testing

* Control graph testing coverage

Oracle

Contents of a test-case ID, Inputs, Expected outputs

Test suite

Test table

Unit testing

* xUnit

Regression testing

Integration testing

* Stubs, Drivers
* Bottom – up
* Top-down
* Sandwich
* Big-bang

System testing

* Function testing
* Performance Testing
* Acceptance testing
  + Benchmark testing
  + Pilot testing
    - alpha test
    - beta test
  + Installation testing
  + Parallel testing

Test-Driven Development

GUI testing (know what it is)

**Configuration management**

Revision control

* Versions
* Main
* Branch

Workflows:

* Centralized
* Feature branches
* Gitflow workflow
* Decentralized

Continuous practices

* Continuous Integration
* Continuous Delivery
* Continuous Deployment

**Project management**

Project definition

SMART goals

Project stakeholders

Dependent parameters

* Calendar time
* Resources
* Features
* Quality

GANTT chart

* Task
* Phase
* Dependency
* Real time
* Slack time
* Critical path
* Milestone
* Tollgate

Effort estimation: COCOMO, Delphi, Planning poker

Buffer time

Teamwork

Risk management

* Kinds of risks
* Identification
* Analysis
  + Magnitude indicator
* Planning
  + Avoidance
  + Transfer
  + Acceptance
    - Mitigation
    - Contingency plan
* Monitoring

Roles (first lecture)

Project plan

Status report

**Processes, Life Cycles, and methods**

Process definition

Software life cycle components or phases

* Requirements
* Design
* Implementation
* Testing
* Integration
* Deployment
* Support, operation, and maintenance
* Replacement

Life cycle models

* Waterfall
* V-model
* Incremental model
* Iterative development
  + Time-boxing

Method frameworks

* Open/UP (Know what it is, a web where roles, activities and artefacts are linked. The four phases)
* Essence Kernel (know what it is, alphas, practices)
* Agile software development (important)
* Extreme programming (XP), values and practices
* Scrum, roles, artifacts, meetings
* Lean software development (important)
* Kanban, work In progress, cycle time

**Software Quality**

Fault-tolerance: Foundations and Metrics

* Core concepts of software fault tolerance
* Architectures for software fault tolerance
* Foundations of reliability /availabilitz modeling
* Direct measures
  + Time To Failure
  + Time To Repair
  + Time Between Failures
* Reliability approximation
* Failure intensity
* Availability
* Maintainability

Usability metrics

* Relevance
* Efficiency
* Attitude
* Learnability

Security

* Confidentiality
* Integrity
* Availability

ISO/IEC 25010

Software metrics, pedagogic classification, pattern of describing metrics

* Usage-based metrics
* Verification & Validation metrics
* Volume metrics
* Structural metrics
* Effort metrics

GQM (Know what it is)

**Software reviews**

Inspections

* Roles
  + Inspection leader
  + Recorder
  + Reader
  + Author
  + Inspector
* Process
  + Plan and overview
  + Individual checking
  + Inspection meeting
  + Edit and Follow-up
* Data collection

Other s/w reviews

* Management reviews
* Audits
* Technical reviews
* Walk-throughs
* Peer-review