# Lesson 1 Flashcards (Chapter 2: Software Processes)

1. Specification

* defining what the system should do

1. Design and implementation

* defining the organization of the system and implementing the system

1. Validation

* checking that the system does what the customer wants

1. Evolution

* changing the system in response to changing customer needs

1. Stage of testing
   * Component / Development testing
   * System / Release testing
   * Customer / User testing
2. Software prototype
   * A prototype is an initial version of a system used to demonstrate concepts and try out design options
3. Process metric
   * Used to measure the efficiency and effectiveness of various processes
4. Design activities
   * Architectural design
   * Database design
   * Interface design
   * Component selection and design
5. Stages of testing
   * Component
   * System
   * Acceptance
6. Reducing the costs of rework
   * Change anticipation
   * Change tolerance
7. Benefits of prototyping
   * Improved system usability.
   * A closer match to users’ real needs.
   * Improved design quality.
   * Improved maintainability.
   * Reduced development effort
8. Throw away prototypes
   * It may be impossible to tune the system to meet non-functional requirements;
   * Prototypes are normally undocumented;
   * The prototype structure is usually degraded through rapid change;
   * The prototype probably will not meet normal organizational quality standards.
9. Incremental delivery
   * the development and delivery is broken down into increments with each increment delivering part of the required functionality
   * User requirements are prioritised and the highest priority requirements are included in early increments
10. Process improvement
    * Way to enhance quality of software
    * Process maturity approach
    * Agile approach
11. Process improvement activities
    * Measure
    * Analysis
    * Change

# Lesson 2 Flashcards (Chapter 3: Agile Software Development)

1. Rapid software development

* often most important requirement of software

1. Agile development

* Program specification, design and implementation are inter-leaved
* Delivered system as a series of versions
* Frequent delivery 2 to 4 weeks
* Minimal documentation

1. Plan driven development

* Separate development stages with outputs at each stage
* Iteration occurs within activities

1. Agile development

* Specification, design, implement and testing interleaved
* Small to medium sized product
* Customer involvement

1. Aim of agile

* To reduce overhead in software process
* Reduced documentation
* Respond quickly to changing requirements

1. Extreme programming

* Incremental planning
* Small releases
* Simple design
* Test first
* Refactoring
* Pair programming
* Collective ownership
* Continuous integration
* Sustainable pace
* Customer involvement

1. User stories

* To capture requirements

1. Refactoring

* Constant code improvement to make change easier

1. Test-first
   * Write tests before code
   * Need test automation
2. Agile project management
   * Need different approach than plan driven
3. Scrum sprint

* A short (3-4 weeks) planning unit in which work to be done is assessed, features are selected for development, the software is implemented and delivered to system stakeholders.

1. Agile problems
   * Scaling up
   * Scaling out
   * Legal approach to contracts
   * Cost of maintenance

1. System issues with agile
   * System size
   * System types
   * System lifetime
   * External regulation
2. People and Teams
   * How good are programmers
   * Team organization
   * Supporting technologies
3. Organizational issues
   * Plan based culture
   * Customer access

# Lesson 3 Flashcards (Chapter 8: Testing)

1. Program testing
   * Show program does what is intended
   * Execute program using artificial data
   * Demonstrate sw meets requirements
   * Discover defects
2. Verification and validation
   * System is fit for purpose
3. Validation:

* Are we building the right product?

1. Verification:

* Are we building the product right?

1. Inspections
   * Analysis of the system to discover faults
   * Static verification
   * Code and document reviews
   * Manual process
   * Very effective to find hidden errors

1. Testing
   * Exercising and observing product behaviour
   * Dynamic verification
2. Stages of testing

* Development testing, where the system is tested to discover bugs and defects
* Release testing where the system is tested to check that it meets its requirements
* User testing where the system is tested in the user’s environment.

1. Development testing

* where the system is tested to discover bugs and defects

1. Release testing

* where the system is tested to check that it meets its requirements

1. User testing

* where the system is tested in the user’s environment.

1. Component testing

* where several individual units are integrated to create composite components. Component testing should focus on testing component interfaces.

1. System testing

* where some or all of the components in a system are integrated and the system is tested as a whole. System testing should focus on testing component interactions.

1. Unit testing

* where individual program units or object classes are tested. Unit testing should focus on testing the functionality of objects or methods.

1. [Testing](http://iansommerville.com/software-engineering-book/web/test-planning/) strategies

* Partition testing
* Guideline-based testing

1. Partition Testing

* Groups of inputs with common characteristics

1. [Guideline based testing](http://iansommerville.com/software-engineering-book/web/path-testing/)

* Based on experience in knowing types of errors that occur and where

1. Equivalence partition

* A class of inputs or outputs where it is reasonable to expect that the system will behave the same way for all members of the class. For example, all strings with less than 256 characters

1. Test-driven development process

* Identify increment of functionality required
* Design tests for this functionality and implement as executable programs.
* Run test along with other implemented tests. The test will fail.
* Implement the functionality and re-run the test. Iterate until the test works.
* Move on to implement the next chunk of functionality

1. Alpha testing

* users work with the development team to test the software as it is being developed.

1. Beta testing

* the software is released to selected users for testing before the formal system release

1. Acceptance testing

* customers test a system to check that it is ready for deployment.

1. Release testing

* the software is tested by a team different than development
* purpose is to show system meets its requirements

1. Requirements based testing

* Verify functionality of system by executing test that correspond to system requirements

# Lesson 4 Flashcards (Chapter 9: Software Evolution)

1. Evolution

* The stage in a software system’s life cycle where it is in operational use and is evolving as new requirements are proposed and implemented in the system.

1. Software change

* Key problem to organize and manage in an organization

1. Servicing

* The stage in a software system’s life cycle where the software remains useful but the only changes made are those required to keep it operational i.e. bug fixes and changes to reflect changes in the software’s environment. No new functionality is added.

1. Phase-out

* The stage in a software system’s life cycle where the software may still be used but no further changes are made to it.

1. Change
   * Driven by system evolution
   * Continues throughout system lifetime
2. Change implementation

* Iteration of design, implement and test

1. Agile and evolution

* Incremental based so evolution is no different than development

1. Legacy systems

* systems that rely on obsolete languages and technologies

1. Legacy system replacement

* Scrap system.
* Continue to Maintain system
* Transform system
* Replace system

1. System replacement analysis

* Access system quality
* Access business value

1. Maintenance costs
   * 2 to 100 times development costs
2. Complexity metrics

* determine maintainability by complexity of system components

1. Software Reengineering

* recreate new system from legacy system.

1. Refactoring

* Improving program to slow down degradation.

1. Bad smells in program code

* Duplicate code
* Long methods
* Data Clumping
* Speculative generality

# Lesson 5 Flashcards (Chapter 10: Dependable Systems)

1. Most important aspect of many systems

* dependability of the system

1. Dependable system

* Reliability
* Availability
* Security

1. Hardware failure

* Hardware fails because of design and manufacturing errors or because components have reached the end of their natural life.

1. Software failure

* Software fails due to errors in its specification, design or implementation.

1. Operational failure

* Human operators make mistakes. Now perhaps the largest single cause of system failures in socio-technical systems.

1. Redundancy

* the inclusion of spare capacity in a system that can be used in the event of failure of part of the system.

1. Diversity

* the use of different types of redundant component so that the probability of a common failure that affects all redundant components is reduced.

1. Formal methods

* Formal specification
* Specification analysis and proof
* Transformational development
* Program verification

1. Verification-based approaches

* Different representations of a software system such as a specification and a program implementing that specification are proved to be equivalent.
* This demonstrates the absence of implementation errors.

1. Refinement-based approaches

* A representation of a system is systematically transformed into another, lower-level represention e.g. a specification is transformed automatically into an implementation.
* This means that, if the transformation is correct, the representations are equivalent.

1. Causes of failures

* A Hardware
* Software
* Operational failure

1. Dependability costs

* Increase exponentially with dependability requirement

1. Holistic system design

* Interaction between all components and layers in a system

1. Dependability process

* Explicitly defined
* Repeatable

1. Formal methods

* Approaches to software development based on mathematical representation and analysis

# Lesson 6 Flashcards (Chapter 13: Security Engineering)

1. Application security

* the application is designed to resist attacks

1. Infrastructure security

* the software is configured to resist attacks

1. Security dimensions

* Confidentiality
* Integrity
* Availability

1. Three controls to enhance system security

* Vulnerability avoidance
* Attack detection and neutralization
* Exposure limitation and recovery

1. Stages of preliminary risk assessment

* Asset value assessment, Exposure assessment
* Threat identification, Attack assessment
* Control identification
* Security requirements definition

1. Operational security

* Primarily human and social issue

1. Security trade off

* More secure system, less usable

1. Protection issues in system design

* How should the system be organised so that critical assets can be protected against an external attack?

1. Distribution issues in system design.

* How should system assets be distributed so that the effects of a successful attack are minimised?

1. Design guidelines for secure systems engineering

* Base security decisions on an explicit security policy.
* Avoid a single point of failure.
* Use redundancy and diversity to reduce risk.
* Validate all inputs

1. Experience-based testing,

* where the system is analyzed against known types of attack.

1. Penetration testing

* where an external team is contracted to discover security flaws in a system.

1. Tool-based testing

* where tools are used to exhaustively test some features of a system, such as the strength of passwords.

1. Formal verification

* where a system is formally verified against a formal security specification.

1. Interception threats

* Allows attacker to gain access to an asset

1. Interruption threats

* Make part or all of a system unavailable

1. Modification threats

* Attacker tampers with a system asset

1. Fabrication threats

* Insert false information in the system

1. Security specification

* Avoid something bad happening

# Lesson 7 Flashcards (Chapter 14: Resilience Engineering)

1. Recognition strategy

* detecting the symptoms of a problem that may lead to system failure.

1. Resistance strategy

* Invoking actions that reduce the probability that a system will fail after a problem of cyberattack has been detected.

1. Recovery strategy

* Critical services are restored as quickly as possible so that the consequences of a failure or cyberattack are minimized.

1. Reinstatement strategy

* All system services are restored and the system brought back to normal operation

1. Cybersecurity

* Sociotechnical issue
* protection of citizens
* protection of businesses
* protection of critical infrastructures

1. Cybersecurity threats

* Threats to confidentiality of assets
* Threats to integrity of assets
* Threats to availability of assets

1. Two approaches to human errors

* Person approach.
* System approach.

1. Latent conditions

* vulnerabilities and weaknesses in a system that, at some stage, may contribute to system failure.

1. Active failures

* some operational event or human error that triggers a sequence of events that could lead to system failure.

1. Swiss cheese model.

* Defensive layers have vulnerabilities
* Vulnerabilities are dynamic
* Failure occurs when holes line up

1. Operational processes.

* Processes for using the system
* Important defense mechanism

1. Coping with failures

* Design operational processes to be flexible and adaptable

1. Critical service identification

* services that must always be available to system to be usable.

1. Diverse barriers in a system

* Close holes and reduce chance of holes lining up.

1. Defensive layers

* Use redundancy and diversity to create set of defensive layers.

# Lesson 8 Flashcards (Chapter 15: Software Reuse)

1. Types of reuse in software engineering

* System
* Application
* Component
* Object and function

1. Application Frameworks

* Integrated set of software artefacts that collaborate to provide a reusable architecture

1. A pattern

* Generic solution to a well-defined problem

1. Platform specialization

* Different versions for different platforms are developed.

1. Environment specialization.

* Different versions for different hardware / operation system environments.

1. Functional specialization

* Different versions for different customers

1. Process specialization

* Different versions for different business processes.

1. Levels of deployment time configuration

* Component selection
* Workflow and rule definition
* Parameter definition.

1. Application system reuse

* Product adapted for different customers without changing the source code.

1. Configurable Application systems

* Generic application designed to support a particular business type or activity.

1. COTS

* Commercial off the shelf system.

1. ERP

* Enterprise Resource Planning system

1. Integrated Application System

* Applications that include two or more applications.

1. Service oriented interfaces

* Service oriented approach means allowing access to the applications functionality through a standard service interface.

1. Application wrapping

* Wrap application in a service interface.

# Lesson 9 Flashcards (Chapter 22: Project Management)

1. Software project management ensures
   * Software is delivered on schedule, with required features
2. Software project management success criteria

* Deliver sw on time
* Deliver sw on budget
* Meet customer’s expectations
* Maintain well-functioning team

1. Factors in project management

* Company size
* Customers
* SW size
* SW types
* Organization culture
* Software development processes

1. Risk management

* Identifying risks and drawing up plans to minimise effect on a project

1. Risk classification

* Project risks
* Product risks
* Business risks

1. Risk management process

* Risk identification,
* Risk analysis,
* Risk planning,
* Risk monitoring.

1. Risk identification

* Estimation,
* Organizational,
* People,
* Technology,
* Tools

1. Risk consequences

* Catastrophic,
* Serious,
* Tolerable,
* Insignificant

1. Risk planning

* Consider each risk and develop a strategy to manage that risk

1. Risk monitoring

* Assess each identified risk regularly to access risk level

1. Poor people management

* Important contributor to project failure

1. Motivating people

* Motivation means to organize work and people to encourage people to work effectively

1. Personality types

* Task oriented
* Self-oriented
* Interaction oriented

1. Team work

* Software engineering is a group activity

1. Team effectiveness

* People in the group
* Group organization
* Technical and managerial communications

# Lesson 10 Flashcards (Chapter 23: Project Planning)

1. Three planning stages

* Proposal stage
* Project start up stage
* Periodically throughout the project

1. Proposal planning

* provide info to set price

1. Project start up

* Create plan with enough detail to make decisions about budget and staffing

1. Software pricing

* No simple relationship between development cost and price charged

1. Plan driven development

* Traditional way to manage a software development project

1. Plan driven development Pros

* Planning for and close monitoring of needed resources

1. Plan driven development Cons

* Early decisions have to be revised because of changes

1. Project scheduling

* Deciding how work in project is organized into separate tasks, when and how these will be executed

1. Scheduling Problems

* The unexpected always happens

1. Agile planning

* Planning is iterative and only the next iteration of the software is planned, often in detail.
* For scheduling, the system being developed is cut down so that the planned date is met

1. Experience-based cost estimation techniques

* Techniques where the estimate is based on a manager’s experience of past projects and the application domain.

1. Algorithmic cost modelling estimation techniques

* A formulaic approach is used to estimate the development effort required, based on attributes of the software and the development team.

1. Approaches to agile planning

* Planning in Scrum
* Project backlog
* Planning game, with user stories

1. Estimation models used in COCOMO II.

* The application composition model,
* The early design model,
* The reuse model,
* The post-architecture model.

1. Quality plan sections

* Product introduction,
* Product plans,
* Process descriptions,
* Quality goals,
* Risks and risk management.

# Lesson 11 Flashcards (Chapter 24: Quality management)

1. Product standard for quality management

* Standards that are applied to the software that is being developed and that define essential features and characteristics of the software.

1. Process standard for quality management

* Standards that define the processes to be followed during software development.

1. Quality management

* Provides an independent check on software development process,

1. Quality planning

* desired product qualities,
* quality assessment process,
* organizational standards

1. Quality is subjective

* The subjective quality of a software system is largely based on non-functional characteristics

1. Process and product quality

* Quality of a developed product is influenced by the quality of production process
* Need quality culture to get this

1. Importance of standards

* Encapsulation of best practice

1. Problems with standards

* Not up to date
* Bureaucratic
* Tedious form filling work

1. ISO 9001

* Standards that can be used as a basis for developing quality management systems

1. Reviews and inspections

* Group examines part or all of a process or system to find potential problems

1. Phases in the review process

* Pre-review activities
* Review meeting
* Post review activities

1. Quality in agile

* Shared good practice

1. Pair programming

* 2 people are responsible for code development

1. Software Metric

* Any type of measurement related to software system

1. Metrics assumptions

* Software property can be measured accurately

# Lesson 12 Flashcards (Chapter 25: Configuration Management)

1. Configuration management

* CM is essential for projects to control changes made.

1. CM activities

* Version management
* System building
* Change management
* Release management

1. Configuration management and Agile

* Cannot do Agile without CM.

1. Development system platform

* Platform where source code is created and edited.

1. Build server platform

* Platform where source code is compiled and managed.

1. Target environment platform

* Platform where the built system is executed

1. Change management process objectives

* To analyse the costs and benefits of proposed changes, approving changes that are worthwhile, and tracking which components of the system have been changed.

1. System version

* An instance of a system that differs, in some ways, from other instances.

1. System release

* A version that is released to customers.

1. Large systems and Configuration Management

* For large systems, never just one working version of the system

1. Version Management

* Keeping track of different versions of software components

1. Public repository and Private workspaces

* Project repository (Master version) and private workspace (checked out developer copy)

1. Most popular open source version management system

* Git

1. Release Management

* Version of a software system that is distributed to customers

1. Release tracking

* Need to be able to reproduce exactly software that has been delivered to a particular customer.

1. Software as a Service

* Reduces problems of release management
* Software provider is responsible for replacing existing release with a new release

# Lesson 13 Flashcards (Computer Reliability)

1. Does computer always give the correct answer

* No

1. Canada’s worst IT disaster

* Therac – 25
* 3 deaths

1. data reliability assessment

* Assessments of reliability made in the broader context of the particular characteristics of your research project and the risk associated with the possibility of using insufficiently reliable data.

1. Corroborating evidence in a data reliability assessment

* Independent evidence that supports information in a database or derived from one. Such evidence, if available, can be found in alternative databases or expert views. Corroborating evidence is unique to each review, and its strength—or persuasiveness—varies.

1. Risk level assessment of data

* Risk is the likelihood that using data of questionable reliability could have substantial negative consequences on the decisions of policymakers and others. To do a risk assessment, consider the following risk conditions, in which the data

1. Data testing

* Can be done by applying logical tests to electronic data files or paper copies of reports, looking for inconsistencies in the data.

1. Billing errors in computer systems are caused by

* Complex systems
* Operator error

1. Store price accuracy

* Up to 10% in error

1. Free and Open source software

* Freeware
* Shareware
* Open Source

1. Software copyright

* Proprietary software
* Free and Open source software

# Lesson 14 Flashcards (Ethics)

1. Two organizations certifying computing professionals in Canada

* IEEE
* CIPS

1. CIPS

* Canada’s Association of IT Professionals

1. Industry certification

* Means to show proficiency in a specific technology

1. Percentage of SW professionals in Canada with credentials in field

* 50 %

1. Software Engineering Code of Ethics and Professionals Practice

* 8 principles
* Related to the behaviour and decisions made by software engineers

1. Software Engineering Code of Ethics and Professionals Practice Principles

* Interest
* Client and Employer
* Product
* Judgement
* Management
* Profession
* Colleagues
* Self

1. Richard Machado

* Age 19
* Frist individual convicted of a federal e-mail hate crime (US)

1. Hughes Aircraft

* Not meeting contract testing requirements

1. Carnivore

* Wiretap for internet traffic placed directly on ISP
* Frist individual convicted of a federal e-mail hate crime (US)

1. Privacy Issues around consumer lists

* Can sell agitate information about customer lists, but not personal information