L02: Introduction to Software Engineering

1. software : Computer programs and associated documentation. developed for a particular customer or general market.

Generic products

Stand-alone systems, for any customer. The developer owns the specification, makes decisions on change.
ex) PC software, CAD software, software for specific markets

Customized products

commissioned by a specific customer to meet their own needs. customers own ~...
ex) embedded control systems, air traffic control software, traffic monitoring systems.

2. attributes of good software: functionality and performance, maintainable, dependable, usable.

Maintainability: can evolve to meet the changing needs of customers.

Dependability and security: not cause physical or economic damage of system failure. Malicious users cannot access or damage the system.

Efficiency: not wasteful of system resources(memory and processor cycles, **responsiveness**, processing time, memory utilization, etc.)

Acceptability: must be understandable, usable and compatible with other systems.

3. software engineering: engineering discipline that is concerned with all aspects of software production.

SE: concerned with all aspects of software production.

E: Using theories and methods to solve problems.

software production: technical process of development, project management and the development of tools, methods etc. to support software production.

4. fundamental software engineering activities: Software specification, software development, software validation and software evolution.

Software specification, establishing requirements and the constraints on the system.

Software development, software is designed and programmed.

Software validation, software is checked to ensure that it is what the customer requires.

Software evolution, modified to reflect changing customer and market requirements.

5. Difference between SE and CS?: CS focuses on theory and fundamentals; SE is concerned with the practicalities of developing and delivering useful software

6. difference between SE and system engineering?: System engineering is concerned with all aspects of computer-based systems development (hardware, software, process engineering)/ SE is a more general process.

7. key challenges of SE?: Coping with increasing diversity, demands for reduced delivery times and developing trustworthy software.

8. costs of SE: 60% of development costs, 40% are testing costs. For custom software, evolution costs > development costs.

9. best SE techniques and methods: different techniques are appropriate for different types of system. no method is better than another.

Batch processing systems : process data in large batches.

Entertainment systems : personal use, intended to entertain the user.

Systems for modeling and simulation : model physical processes or situations, include many, separate, interacting objects.

Data collection systems : collect data using a set of sensors and send them to other systems for processing.

Systems of systems : composed of a number of other software systems.

10. differences from web to SE: availability of software services, possibility of developing highly distributed service-based systems. advances in programming languages and software reuse.

Cloud computing : approach to the provision of computer services where applications run remotely on the ‘cloud’. Users do not buy software but pay according to use.

Software reuse: approach for constructing web-based systems. assemble them from pre-existing software components and systems.

Incremental and agile development: should be developed and delivered incrementally. it's impractical to specify all the requirements in advance.
Service-oriented systems: software components are stand-alone web services.
Rich interfaces technologies - Angular, React, AJAX and HTML5
Ethics: Software engineers have responsibilities to the engineering profession and society. They should not simply be concerned with technical issues. Professional societies publish codes of conduct standards.

L03-04: Software Processes

software process: structured set of activities required to develop a software system. (Specification, Design and implementation, Validation, Evolution)
Plan-driven processes: all of the process activities are planned in advance and progress is measured against this plan.
agile processes: planning is incremental and it is easier to change the process to reflect changing customer requirements.
software process model: abstract representation of a process. +Products, Roles, Pre- and post-conditions
The waterfall model(SPM)

Maybe a plan-driven or agile model. Incremental development. Specification, development and validation are interleaved. using reusable/configurable components.
phases: Requirements analysis and definition/System and software design/Implementation and unit testing/Integration and system testing/Operation and maintenance
Bad: difficulty of change after the process. has to be complete before moving onto the next phase. Inflexible , difficult to accept customer requirements.
-> appropriate when the requirements are well-understood and less changes / large systems' engineering projects where a system is developed at several sites. (Embedded System, Critical System,Large software systems)

Incremental development(SMP)

Good: reduce cost of changing. less work has to be redone. EZ to get customer feedback. Customers can check and comment on demonstrations. rapid delivery and deployment.
Bad: The process is invisible. not cost-effective to produce documents of every version of the system for managers. System structure degrades as new increments are added. changement corrupts its structure(if no refectoring). difficult and costly to incorporate further software changes.

Integration and configuration(Reuse-oriented)(SMP)

reuse software /for systems are integrated from existing components or application systems. (=COTS (Commercial-off-the-shelf) systems). maybe plan driven or agile.
reusable software: Stand-alone application systems (COTS), Collections of objects (.NET or J2EE), Web services (service standards and remote invocation)
Good: Reduced costs and risks,Faster delivery and deployment.
Bad: may not meet real needs of users, Loss of control over evolution of reused system elements.

Process activities: interleaved sequences of technical, collaborative and managerial activities with the overall goal of specifying, designing, implementing and testing a software system. *waterfall model-in sequence, incremental -interleaved.*
(Software specification) Requirements engineering process:
Requirements elicitation and analysis- What do the system stakeholders require or expect from the system?
Requirements specification- Defining the requirements in detail
Requirements validation- Checking the validity of the requirements
Software design and implementation: The process of converting the system specification into an executable system. closely related and may be interleaved.
Software design: Design a software structure that realizes the specification;
Implementation: Translate this structure into an executable program;
Design activities

Architectural design: identify the overall structure of the system, the principal components (subsystems or modules), their relationships and how they are distributed.
Database design: design the system data structures and how these are to be represented in a database.
Interface design: define the interfaces between system components.
Component selection and design: search for reusable components. If unavailable, you design how it will operate.

System implementation

Programming is an individual activity with no standard process.
Debugging is the activity of finding program faults and correcting these faults.
Software validation: show that a system conforms to its specification and meets the requirements of the system customer. checking and reviewing, testing.
Verification: Are we building the product RIGHT? (Design follows specification??)
Validation: Are we building the RIGHT product? (works as intended?)
System testing : executing the system with test cases that are derived from the specification of the real data to be processed by the system. most commonly used V & V activity.

Testing stages

Component testing: Individual components are tested independently;
System testing: Testing of the system as a whole.
Customer testing:Testing with customer data to check that the system meets the customer's needs.
Software evolution: business circumstances change, software also evolves and changes.

Reducing the costs of rework

Change anticipation: anticipate possible changes before significant rework is required.
Change avoidance: how to avoid future changes?? (prototype system)
Change tolerance: designed so that changes can be accommodated at relatively low cost.
Software prototyping: initial version of a system to demonstrate and try out. developed quickly to check the customer's requirements and the feasibility of design decisions. can be used in requirements engineering process, design processes, testing process.
Benefits: Improved system usability, A closer match to users' real needs, Improved design quality, Improved maintainability, Reduced development effort.
Prototype development: based on rapid prototyping languages or tools, leaving out functionality(no error checking and recovery, Focus on functional-no reliability and security)
Throw-away prototypes: discarded after development. cuz of undocumented. structure is degraded through rapid change, not meeting normal organizational quality standards.
Incremental delivery: requirements broken down into increments and prioritized, each increment delivering part of the required functionality. Once started, the requirements are frozen -> App, web, plug-in-based applications
Incremental development: Develop the increments, evaluate it before start next increment;
Incremental delivery: Deploy an increment to end-users; realistic evaluation of practical use.
Good: system functionality is available earlier, Early increments act as a prototype, highest priority system can receive the most testing.
Bad: Problematic when a new system replaces an existing system, cannot provide a set of basic facilities, Uncertainty(requirements are not defined in detail until an increment is to be implemented), organizations want complete specification and deliverables but cannot.

Process improvement: understanding existing processes and changing these processes to increase product quality and/or reduce costs and development time.
The process maturity approach: focuses on improving process and project management and introducing good software engineering practice.
The agile approach: focuses on iterative development and the reduction of overheads in the software process. rapid delivery of functionality and responsiveness to changing customer requirements.

Process improvement activities

Process measurement : measure attributes of the software process or product. forms a baseline if process improvements have been effective.
Process analysis : assess current process, identify process weaknesses and bottlenecks.
Process change : address some of the identified process weaknesses. The cycle resumes to collect data about the effectiveness of the changes.

L05: Project Management

Software project management: ensuring that on time and on schedule, accepting requirements/ cuz of budget and schedule constraints by the organization.
Success criteria: Deliver at the agreed time/ Keep overall costs within budget/meets the customer's expectations/ Maintain a coherent and well-functioning development team.
Distinct characteristics of SE projects: The product is intangible / Many software projects are 'one-off' projects / Software processes are variable and organization specific.
Factors influencing project management: Company size /Software customers Software size /Software type/Organizational culture /Software development processes
Universal management activities: Project planning / Risk management / People management / Reporting / Proposal writing

Risk management: identifying risks and making plans to minimize their effect on a project. important because of the inherent uncertainties in software development.(loosely defined requirements, requirements changes, difficulties in estimating the time and resources required, and differences in individual skills.)
Risk classification: The type of risk (technical, organizational, ..)
what is affected by the risk: Project risks ->schedule or resources; Product risks->quality or performance of the software; Business risks->organization(ex. introducing a new project by a competitor);
The risk management process
Risk identification: Identify project, product and business risks; team of individual
checklist: Technology risks./ Organizational risks. /People risks. /Requirements risks. /Estimation risks.
Risk analysis: Assess probability and seriousness of each risk.
Risk planning: Draw up plans/develop a strategy to avoid or minimize/manage the effects of the risk;
Avoidance strategies-The probability that the risk will arise is reduced;
Minimization strategies-The impact of the risk on the project or product will be reduced;
Contingency plans-If the risk arises, contingency plans are plans to deal with that risk;
What-if questions-What if engineers are ill(Reorganize team)/budget cuts of 20%(briefing)/OSS sucks/software components go out(Replace) …
Risk monitoring: Monitor the risks throughout the project; Assess each identified risks regularly to check probable, the effects of the risk have changed, at management progress meetings.

People management factors
Consistency: comparable way without favorites or discrimination.
Respect: different skills should be respected.
Inclusion: Involve all members and consider people's views.
Honesty: always be honest about what is going well and what is going badly in a project.
Motivate people (Physiologic, Safety, Social, Esteem needs)
Teamwork: best size for SE is 4 to 6 members. (not more than 12 members), should consider motivation, Technical knowledge and ability, Personality
small groups-> minimize communication problems
cohesive group->Members think group>individuals, Loyal, establish its own quality standard. (consensus), Knowledge is shared, Refactoring and improvement is encouraged.
Group communications affected by Group size/structure/composition/ Physical environment/ Available communication channels