Chapter 1 Introduction

Software engineering is concerned with theories, methods and tools for professional software development, which is also concerned with cost-effective software development. Failure to use software engineering methods often causes to more expensive and less reliable than it should be.

Software: Computer programs and associated documentation, developed for a particular customer or a general market.

Good software should deliver the required functionality and performance to the user and should be maintainable, dependable and usable.

Software engineering is an engineering discipline that is concerned with all aspects of software production.

Difference between software engineering and computer science: Computer science focuses on theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software.

Difference between software engineering and system engineering: System engineering is concerned with all aspects of computer-based systems development including hardware, software and process engineering. Software engineering is part of this more general process.

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

Ethical principles: PUBLIC, CLIENT AND EMPLOYER, PRODUCT, JUDGMENT, MANAGEMENT, PROFESSION, COLLEAGUES, SELF.

Web software engineering: software reuse is the dominant approach for constructing web-based systems. Incremental and agile development.

Ethical behavior is more than simply upholding the law but involves following a set of principles that are morally correct.

Competence: engineers should not misrepresent their level of competence. They should not knowingly accept work which is out with their competence.

Engineering discipline: Using appropriate theories and methods to solve problems bearing in mind organizational and financial constraints.

Essential attributes of good software: Maintainability, Dependability (reliability, safety and security; should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system), Efficiency (responsiveness, processing time, memory utilization), Acceptability(understandable, usable and compatible with other systems that they use).

Software specification: customers and engineers define the software that is to be produced and the constraints on its operation. Development: where the software is designed and programmed. Validation: where the software is checked to ensure that it is what the customer requires. Evolution: where the software is modified to reflect changing customer and market requirements.

Heterogeneity: systems are required to operate as distributed systems across networks that include different types of computer and mobile devices. Scale: software has to be developed across a very wide range of scales, from very small embedded systems in portable or wearable devices through to Internet-scale, cloud-based systems that serve a global community.

There are many different types of software system and there is no universal set of software techniques that is applicable to all of these. The software engineering methods and tools used depend on the type of application being developed, the requirements of the customer and the background of the development team.

The development of interactive web-based systems and mobile apps which require a blend of software and graphical design skills. The web has led to the availability of software services and the possibility of developing highly distributed service-based systems. Web-based systems development has led to important advances in programming languages and software reuse.

Generic products – Stand-alone system: CAD, appointments systems for dentists.

Customized products – Commissioned by a specific customer to meet their own needs: traffic monitoring systems.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Chapter 4 Requirements Engineering

RE processes: Elicitation, Analysis, Validation, Management. Its an iterative activity in which these processes are interleaved. Requirement elicitation: **Start,** User requirements elicitation, System req. elicitation; Requirements specification: **Business requirements specification,** User requirements specification, System requirements specification and modeling; Requirements validation: **Feasibility study**, Prototyping, Reviews.

Ways of writing a system requirements specification: Natural language, Structured natural language, Design description languages, Graphical notations, Mathematical specifications.

Metrics for nonfunctional requirements: Speed, Size, Ease of use, Reliability, Robustness, Portability

Requirements completeness and consistency: They should include descriptions of all facilities required. There should be no conflicts or contradictions in the descriptions of the system facilities.

System stakeholder types: End users, System managers, System owners, External Stakeholders

Agile methods: use incremental requirements engineering and may express requirements as user stories which is practical for business systems but not for critical systems.

Functional requirements: Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.

Non-functional requirements (Product requirements, Organizational requirements, External requirements) define system properties and constraints e.g. reliability, response time and storage requirements, and may be more critical than functional requirements.

Requirements Abstraction: A contract defined its needs in a sufficiently abstract way that a solution is not pre-defined. The requirements must be written so that several contractors can bid for the contract, offering different ways of meeting the client organization’s needs. Once a contract has been awarded, the contractor must write a system definition for the client in more detail so that the client understands and can validate what the software will do. Both of these documents may be called the requirements document for the system.

System requirements: A structured document setting out detailed descriptions of the system’s functions, services and operational constraints. Defines what should be implemented so may be part of a contract between client and contractor.

The process of establishing the services that a customer requires from a system and the constraints under which it operates and is developed.

The system requirements are the descriptions of the system services and constraints that are generated during the requirements engineering process may range from a high-level abstract statement of a service or of a system constraint to a detailed mathematical functional specification.

Complete: Requirements should include descriptions of all facilities required.

Consistent: Requirements should be no conflicts or contradictions in the descriptions of the system facilities.

Metrics for specifying nonfunctional requirements: Speed (Processed transactions/second; User/event response time; Screen refresh time); Size (Mbytes, Number of ROM chips); Ease of use (Training time, Number of help frames); Reliability (Mean time to failure, Probability of unavailability, Rate of failure occurrence, Availability); Robustness (Time to restart after failure; Percentage of events causing failure, Probability of data corruption on failure); Portability (percentage of target dependent statements, Number of target systems)

Ethnography (Spending time observing and analyzing how people actually work for social scientists) is effective for understanding existing processes but cannot identify new features that should be added to a system.

Scenarios should include descriptions of the starting situation, the normal flow of events, what can go wrong, the state when the scenario finishes, and information about other concurrent activities.

Requirements specification: the process of writing down the user and system requirements in a requirements document.

Graphical notations: Graphical models, supplemented by text annotations, are used to define the functional requirements for the system; UML use case and sequence diagrams are commonly used.

Natural language specification: Requirements are written as natural language sentences supplemented by diagrams and tables. Expressive, intuitive and universal, understandable for users and customers.

Structured specifications: embedded control system ; Form-based specifications: Tabular specification:

Use cases: identify the actors in an interaction and which describe the interaction itself. A set of use cases should describe all possible interactions with the system; UML sequence diagrams may be used to add detail to use-cases by showing the sequence of event processing in the system.

UML - A visual language for specifying, constructing and documenting the artifacts of systems.

The software requirements document: is the official statement of what is required of the system developers, includes both a definition of user requirements and a specification of the system requirements.

The structure of a requirements document: Preface, Introduction, Glossary, User requirements definition, System architecture, System requirements specification, System models, System evolution, Appendices, Index.

Key points: Requirements set out what the system should do and define constraints on its operation and implementation.

Requirements validation is the process of checking the requirements for validity, consistency, completeness, realism and verifiability.

Ch2 Software Process: A structured set of activities required to develop a software system. Specification, Design and implementation, Validation, Evolution.

Incremental delivery: Rather than deliver the system as a single delivery, the development and delivery is broken down into increments with each increment delivering part of the required functionality.

The waterfall model: is mostly used for large systems engineering projects where a system is developed at several sites. Plan-driven model. Separate and distinct phases of specification and development. Requirements definition 🡪 System and software design 🡪 Implementation and unit testing 🡪 Integration and system testing 🡪 Operation and maintenance

Incremental development: Specification, development and validation are interleaved. May be plan-driven or agile. Benefits: The cost of accommodating changing customer requirements is reduced. Easier to get customer feedback on the development work that has been done. More rapid delivery and deployment of useful software to the customer is possible. Problems: The process is not visible. System structure tends to degrade as new increments are added.

Integration and configuration: The system is assembled from existing configurable components. May be plan-driven or agile. Reuse is now the standard approach for building many types of business system.

Types of reusable software: Stand-alone application systems that are configured for use in a particular environment. Collections of objects that are developed as a package to be integrated with a component framework such as .NET or J2EE. Web services that are developed according to service standards and which are available for remote invocation. Key process stages: Requirements specification, Software discovery and evaluation, Requirements refinement, Application system configuration, Component adaptation and integration. Reduced costs and risks, faster delivery and deployment, but with requirements compromises and loss of control over evolution.

Design activities: Architectural design, where you identify the overall structure of the system, the principal components, their relationships and how they are distributed; Database design, Interface design, Component selection and design.

Debugging is the activity of finding program faults and correcting these faults.

Verification and validation is intended to show that a system conforms to its specification and meets the requirements of the system customer. Involves checking and review processes and system testing. Component testing 🡪 System testing 🡪 Acceptance testing.

Change anticipation, where the software process includes activities that can anticipate possible changes before significant rework is required. Change tolerance, where the process is designed so that changes can be accommodated at relatively low cost.

A prototype is an initial version of a system used to demonstrate concepts and try out design options. Improved system usability; A closer match to users’ real needs. Improved design quality. Improved maintainability. Reduced development effort. Establish prototype objectives 🡪 Define prototype functionality 🡪 Develop prototype 🡪 Evaluate prototype.

Incremental development: Develop the system in increments and evaluate each increment before proceeding to the development of the next increment; Normal approach used in agile methods; Evaluation done by user/customer proxy. Incremental delivery: Deploy an increment for use by end-users; More realistic evaluation about practical use of software; Difficult to implement for replacement systems as increments have less functionality than the system being replaced. Define outline requirements 🡪 Assign requirements to increments 🡪 Design system architecture 🡪 Develop system increment 🡪 Validate increment 🡪 Integrate increment 🡪 Validate system 🡪 Deploy increment 🡪 complete ? Final system : Develop system increment. Advantage: System functionality is available earlier; early increments act as a prototype to help elicit requirements for later increments; lower risk of overall project failure; the highest priority system services tend to receive the most testing.