Software Engineering Quiz (Chapters 1-3)

Study online at https://quizlet.com/_abroe2

Software	Computer programs and associated documentation. Software products may be developed for a particular customer or may be developed for a general market.
Attributes of a good software	Should deliver the required functionality and performance to the user and should be maintainable, dependable and usable.
Software Engineering	An engineering discipline that is concerned with all aspects of software production
Fundamental Software Software Engineering activities	Software specification, software development, software validation and software evolution
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.
Key challenges facing software engineering	Coping with increasing diversity, demands for reduced delivery times and developing trustworthy software.
Costs of software engineering	Roughly 60% of software costs are development costs, 40% are testing costs. For custom software, evolution costs often exceed development costs
What are the best software engineering techniques and methods?	While all software projects have to be professionally managed and developed, different techniques are appropriate for different types of system. For example, games should always be developed using a series of prototypes whereas safety critical control systems require a complete and analyzable specification to be developed. You can't, therefore, say that one method is better than another.
What differences has the web made to software engineering?	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.
4 Essential Attributes of Good Software	Maintainability, Dependability and Security, Efficiency, Acceptability
Maintainability	Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment.
Dependability and Security	Software dependability includes a range of characteristics including reliability, security and safety. Dependable software 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	Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilisation, etc.
Acceptability	Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable and compatible with other systems that they use.
True or False: High-level activities of specification, development, validation and evolution are part of all software processes	True
True or False: The fundamental notions of software engineering are only applicable to certain types of system development.	False. The fundamental notions of software engineering are universally applicable to all types of system development.
True or False: There are many different types of system and all have the same software engineering tools and techniques for their development.	False. There are many different types of system and each requires appropriate software engineering tools and techniques for their development.

Software Engineering Quiz (Chapters 1-3)

Study online at https://quizlet.com/_abroe2

Software Processes	A structured set of activities required to develop a software system.
Examples of General Process Models	Waterfall Model, Incremental Development, Reuse-oriented Development
Requirements Engineering	Process of developing a software specification
Design and Implementation Processes	Concerned with transforming a requirements specification into an executable software system.
Software Validation	Process of checking that the system conforms to its specification and that it meets the real needs of the users of the system
When does software evolution take place?	When you change existing software systems to meet new requirements. The software must evolve to remain useful
What activities are needed in the process to cope with change?	Prototyping and Incremental Delivery
True or False: Processes may be structured for iterative development and delivery so that changes may be made without disrupting the system as a whole	True
SEI Process Maturity Framework	Identifies maturity levels that essentially correspond to the use of good software engineering practice.
True or False: The principal approaches to process improvement are agile approaches, geared to increase process overheads, and maturity-based approaches based on slow process management and the use of popular software engineering practice	False. The principal approaches to process improvement are agile approaches, geared to reducing process overheads, and maturity-based approaches based on better process management and the use of good software engineering practice
Foundations of All Software Processes	Specification, Design, Validation, Evolution
Software Process Model	Abstract representation of a process. It presents a description of a process from some particular perspective
4 Design Activities	Architectural design, Database design, Interface design, Component selection and design
Architectural Design	where you identify the overall structure of the system, the principal components (subsystems or modules), their relationships and how they are distributed.
Database Design	²where you design the system data structures and how these are to be represented in a database.
Interface Design	where you define the interfaces between system components
Component Selection and Design	where you search for reusable components. If unavailable, you design how it will operate.
3 Testing Stages	Component Testing, System Testing, Customer Testing
5 Principles of Agile Methods	Customer involvement, Incremental delivery, People not process, Embrace change, Maintain simplicity
Incremental planning	Requirements are recorded on story cards and the stories to be included in a release are determined by the time available and their relative priority. The developers break these stories into development 'Tasks'.
Small releases	The minimal useful set of functionality that provides business value is developed first. Releases of the system are frequent and incrementally add functionality to the first release.
Simple design	Enough design is carried out to meet the current requirements and no more.
Test-first development	An automated unit test framework is used to write tests for a new piece of functionality before that functionality itself is implemented.
Refactoring	All developers are expected to refactor the code continuously as soon as possible code improvements are found. This keeps the code simple and maintainable.
Pair programming	

	Developers work in pairs, checking each other's work and providing the support to always do a good job.
Collective ownership	The pairs of developers work on all areas of the system, so that no islands of expertise develop and all the developers take responsibility for all of the code. Anyone can change anything.
Continuous integration	As soon as the work on a task is complete, it is integrated into the whole system. After any such integration, all the unit tests in the system must pass.
Sustainable pace	Large amounts of overtime are not considered acceptable as the net effect is often to reduce code quality and medium term productivity
On-site customer	A representative of the end-user of the system (the customer) should be available full time for the use of the XP team. In an extreme programming process, the customer is a member of the development team and is responsible for bringing system requirements to the team for implementation.
Development team	A self-organizing group of software developers, which should be no more than 7 people. They are responsible for developing the software and other essential project documents.
Potentially shippable product increment	The software increment that is delivered from a sprint. The idea is that this should be 'potentially shippable' which means that it is in a finished state and no further work, such as testing, is needed to incorporate it into the final product. In practice, this is not always achievable.
Product backlog	This is a list of 'to do' items which the Scrum team must tackle. They may be feature definitions for the software, software requirements, user stories or descriptions of supplementary tasks that are needed, such as architecture definition or user documentation.
Product owner	An individual (or possibly a small group) whose job is to identify product features or requirements, prioritize these for development and continuously review the product backlog to ensure that the project continues to meet critical business needs. The Product Owner can be a customer but might also be a product manager in a software company or other stakeholder representative
Scrum	A daily meeting of the Scrum team that reviews progress and prioritizes work to be done that day. Ideally, this should be a short face-to-face meeting that includes the whole team.
ScrumMaster	responsible for ensuring that the Scrum process is followed and guides the team in the effective use of Scrum. He or she is responsible for interfacing with the rest of the company and for ensuring that the Scrum team is not diverted by outside interference. The Scrum developers are adamant that the ScrumMaster should not be thought of as a project manager. Others, however, may not always find it easy to see the difference.
Sprint	A development iteration. Sprints are usually 2-4 weeks long.
Velocity	An estimate of how much product backlog effort that a team can cover in a single sprint. Understanding a team's velocity helps them estimate what can be covered in a sprint and provides a basis for measuring improving performance.
Customer Involvement	This depends on having a customer who is willing and able to spend time with the development team and who can represent all system stakeholders. Often, customer representatives have other demands on their time and cannot play a full part in the software development. Where there are external stakeholders, such as regulators, it is difficult to represent their views to the agile team.

Embrace Change	Prioritizing changes can be extremely difficult, especially in systems for which there are many stakeholders. Typically, each stakeholder gives different priorities to different changes.
Incremental delivery	Rapid iterations and short-term planning for development does not always fit in with the longer-term planning cycles of business planning and marketing. Marketing managers may need to know what product features several months in advance to prepare an effective marketing campaign.
Maintain simplicity	Under pressure from delivery schedules, team members may not have time to carry out desirable system simplifications.
People not process	Individual team members may not have suitable personalities for the intense involvement that is typical of agile methods, and therefore may not interact well with other team members.
Definition of agile methods	Incremental development methods that focus on rapid software development, frequent releases of the software, reducing process overheads by minimizing documentation and producing high-quality code
Agile development practices include	-User stories for system specification -Frequent releases of the software -Continuous software improvement -Test-first development -Customer participation in the development team
Definition of scrum	agile method that provides a project management framework
True or False: Many practical development methods are a mixture of plan-based and agile development.	True
True or False: Scaling agile methods for large systems is easy and quick	False: Scaling agile methods for large systems is difficult. Large systems need up-front design and some documentation and organizational practice may conflict with the informality of agile approaches.