**what is software engineering?**

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

소프트웨어 공학은 소프트웨어 생산의 모든 측면을 다루는 공학 분야이다.

**four fundamental activities in software porcesses**

software specification, software design and implementation, software validation, software evolution

**essential attributes of good software**

acceptability, dependability and security, efficiency, maintainability

**Generic products와 Customized products의 차이**

**Generic products**

Stand-alone systems that are marketed and sold to any customer who wishes to buy them

구매하고자 하는 모든 고객에게 마케팅 및 판매되는 독립형 시스템

**Customized products**

Software that is commissioned by a specific customer to meet their own needs

특정 고객이 자신의 요구를 충족하도록 위임한 소프트웨어

**approaches we can user to reduce the costs of rework**

**Change anticipation** : software process includes activities that can **anticipate possible changes** before significant rework is required.

prototype system may be developed to show some key features of the system to customer

**Change tolerance** : This normally involves some form of **incremental development.** proposed changes may be implemented in increments that have not yet been developed. if this is impossible then only a single increment may have be altered to incorporate the change

**the principles agile methods lead to the accelerated development and deployment of software**

individual and interactions over processes and tools

working software over comprehensive documentation

customer collaboration over contract negotiation

responding to change over following a plan

**important characteristics of XP**

**user stories for specification** : user requirements are expressed as user stories or scenarios

**refactoring** : it proposes constant code improvement to make changes easier when they have to be implemented

**test-driven development** :writing tests before code clarifies the requirements to be implemented

**pair programming :** programmers sit together at the same computer to develop the soft ware

**why it is important to make a distinction between developing the user requrements and system requirements**

**user requirement** are intended to describe the system's functions and features from a user perspective and it is essential that users understand these requirements

사용자 요구사항은 사용자 관점에서 시스템의 기능 및 기능을 설명하기 위한 것이며 사용자가 이러한 요구사항을 이해하는 것이 필수적이다.

**system requirement** are much more detailed than the user requirements and are intended to be a precise specification for the system that may be part of a system contract

시스템 요구사항은 사용자 요구사항보다 훨씬 더 상세하며 시스템 계약의 일부일 수 있는 시스템의 정확한 사양이 되기 위한 것이다.

**The waterfall model**

**plan- driven** - this is a safety-critical system so requires a lot of up-front analysis before implementation

**difficulty of accommodating change after the process is underway**

The waterfall model is mostly used **for large systems engineering projects**

**Incremental development**

**incremental development** - this is a system where requirements will change and there will be an extensive user interface components

advantages

the cost of accommodating changes to customer requirements is reduced

고객 요구사항의 변경 사항을 수용하는 비용이 절감됨

it is easier to get customer feedback on development work that has been done

완료된 개발 작업에 대한 고객 피드백을 받는 것이 더 쉽습니다.

more rapid delivery and deployment of useful software to the customer is possible

고객에게 유용한 소프트웨어를 보다 신속하게 전달하고 배포할 수 있습니다.

**Problems**

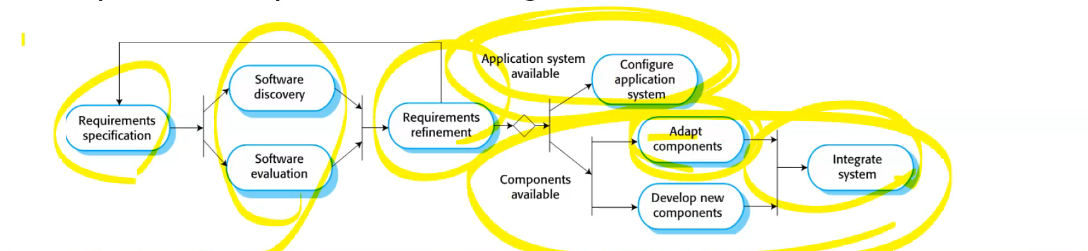
The process is not visible.

System structure tends to degrade as new increments are added*.*

**Integration and configuration**

**reuse-based approach** - software reuse where systems are integrated from existing components or application systems

* Requirements specification
* Software discovery and evaluation
* Requirements refinement
* Application system configuration
* Component adaptation and integration



**Advantages**

Reduced costs and risks as less software is developed from scratch

Faster delivery and deployment of system

**Disadvantages**

*But requirements compromises are inevitable so system may not meet real needs of users*

*Loss of control over evolution of reused system elements*

**Process activities**

**specification, development, validation and evolution**

**Design activities**

*Architectural design*

*Database design*

*Interface design*

*Component selection and design*

**Software validation**

**Verification**: Are we building the product **right**?

The software should conform to its **specification**.

**Validation**: Are we building the **right** product?

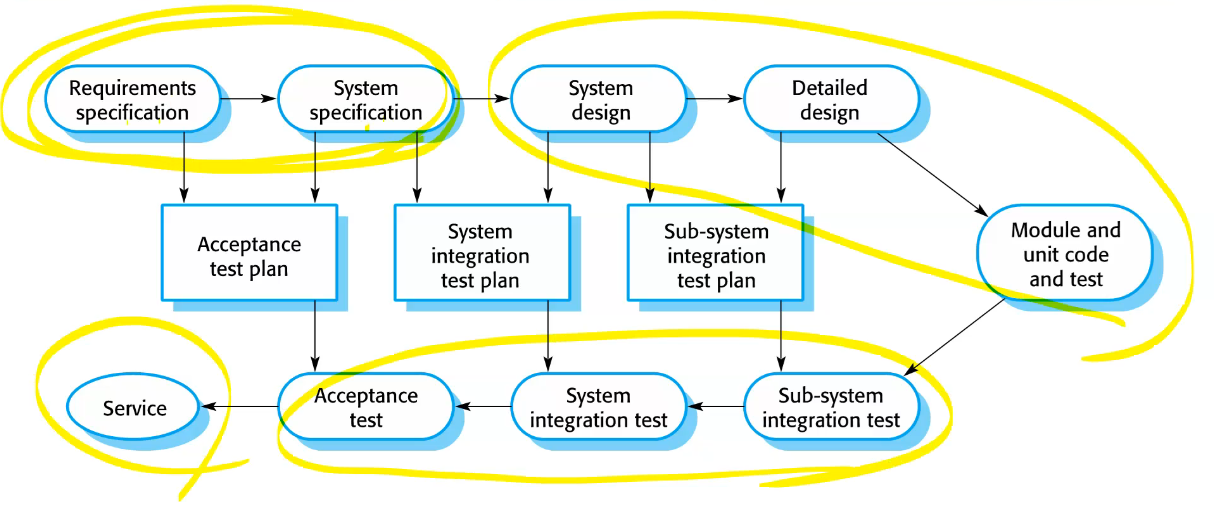
The software should do what the user really requires. **(specification+α)**

**Stages of testing**

**Component testing**

**System testing**

**Customer(acceptance) testing**

****

**Process improvement**

* ***Process measurement***
* ***Process analysis***
* ***Process change***

**Functional requirements**

statements of the services that the system must provide or are descriptions of how some computations must be carried out..

**Mentcare system: functional requirements**

* A user shall be able to search the appointments lists for all clinics.
* The system shall generate each day, for each clinic, a list of patients who are expected to attend appointments that day.

**Non-functional requirements**

often constrain the system being developed and the development process being used.

**Examples of nonfunctional requirements in the Mentcare system**

**Product requirement**

The Mentcare system shall be available to all clinics during normal working hours (Mon–Fri, 08.30–17.30). Downtime within normal working hours shall not exceed five seconds in any one day.

**Organizational requirement**

Users of the Mentcare system shall authenticate themselves using their health authority identity card.

**External requirement**

The system shall implement patient privacy provisions

**Requirements engineering processes**

* + Requirements elicitation;
  + Requirements analysis;
  + Requirements validation;
  + Requirements management.

**Requirements elicitation**

Interviews are not good for understanding domain requirements

* **Ethnography** A social scientist spends a considerable time observing and analysing how people actually work.

Requirements that are derived from cooperation and awareness of other people’s activities.

**Requirements specification**

**Natural language**

Structured natural language

Use cases

**Requirements validation**

Validity Consistency Completeness Realism Verifiability

**validation techniques**

Requirements reviews

Prototyping

Test-case generation

**Requirements management**

**Requirements management is the process of managing and controlling these changes.**

Requirements identification

A change management process

Traceability policies

Tool support

