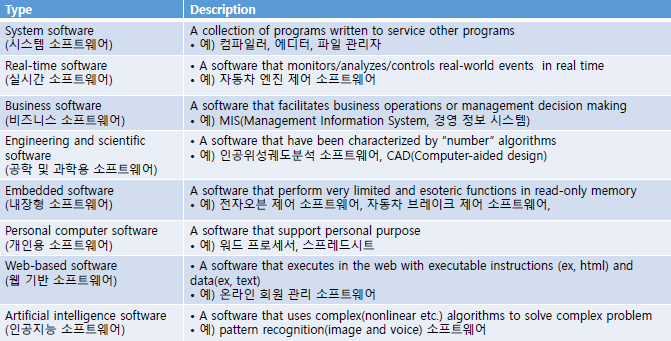
**SE Midterm**

Sehun Ahn

***1. Introduction to SE***

**1.1 What is SE?**

**SE** : Software engineering is an engineering discipline which is concerned with all aspects of software production  
=> 소프트웨어 생산의 모든 면을 고려하는 학문

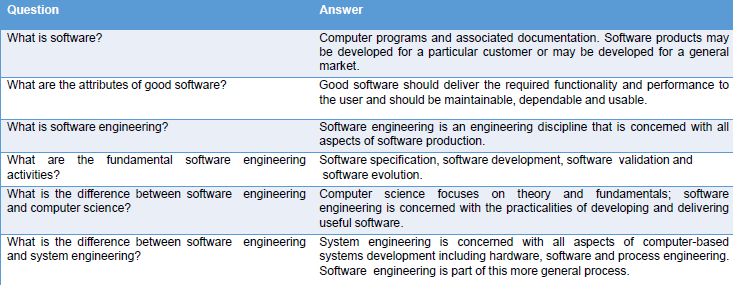
**Software Products**  


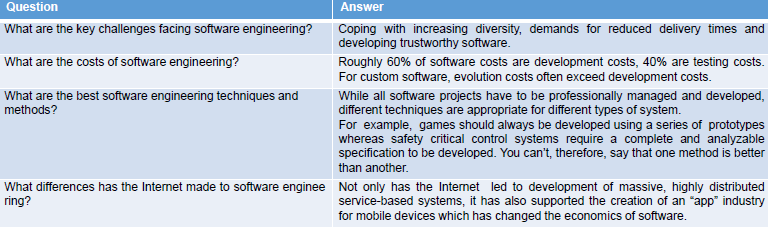
**Background**- Increasing SW needs: dependent on software  
- SW cost: maintenance cost > development cost  
=> SE is concerned with cost-effective development  
- Term: NATO conference  
poor quality, exceeding time and budget, so  
=> SW is built systematically, rigorously, measurably, on time, on budget, and within specification

**Without SE** -> Software project failure  
- Increasing system complexity  
- Software is often more expensive and less reliable

**1.2 Professional Software development**

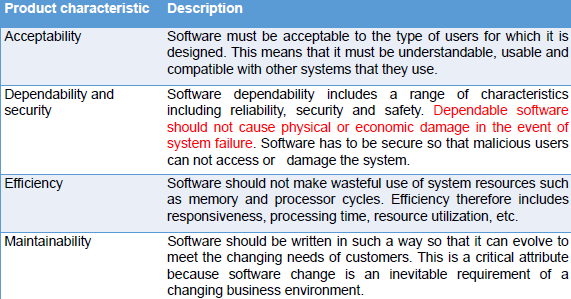
**Frequently asked questions about SE**





**Software Products**- Generic products: stand-alone system that are marketed and sold to any customer. Ex) PC  
=> The specification is owned by the **SW developer**   
- Customized products: commissioned by a specific customer to meet their own needs. Ex) traffic control  
=> The specification is owned by the **customer**

**Essential attributes of good software**



**Importance of software engineering**  
- It is usually cheaper, in the long run, to use SE methods and techniques for software systems.  
- The majority of costs are the costs of changing the software after it has gone into use

**Software Process activities**  
- Software specification: defining what the system should do  
- Software development(design & implementation): defining the organization of the system and implementing the system  
- Software validation: checking that it does what the customer wants  
- Software evolution: changing the system in response to changing customer needs

**General issues that affect software**  
- Heterogeneity: operated in different types of device  
- Business and social change  
- Security and trust  
- Scale

**Software engineering diversity**  
- the type of application, requirements, background

**Application types**  
- Stand-alone application: run on a local computer  
- Interactive transaction-based application   
- Embedded control system: control hardware  
- Batch processing systems: 일괄처리 시스템(ex성적)  
- Entertainment systems  
- Systems for modelling and simulation: ex)scientist  
- Data collection and analysis systems: ex)environment  
- Systems of systems

**Software engineering fundamentals**: Universally applicable to all types of software system- Apply to all types of SW system  
- be developed understood development process  
- Dependability and performance are important  
- Understanding the specification and requirements  
- If appropriate, reuse SW rather than write new SW

**Web-based software engineering**  
- Complex distributes systems  
- Software reuse  
- Incremental and agile development  
- Service-oriented system  
- Rich interface (by web browser)

**1.4 Case studies**

**A Personal insulin pump**  
: embedded control system  
- Safety-critical system  
- Essential high-level requirements

**A patient management system (Mentcare)**  
: information system  
- Individual care management  
- Patient monitoring  
- Administrative reporting  
=> Privacy and Safety is concerned

**A wilderness weather station**  
: data collection system

**iLearn**  
: Service-oriented system  
- Utility / Application / Configuration service  
- Integrated / Independent services

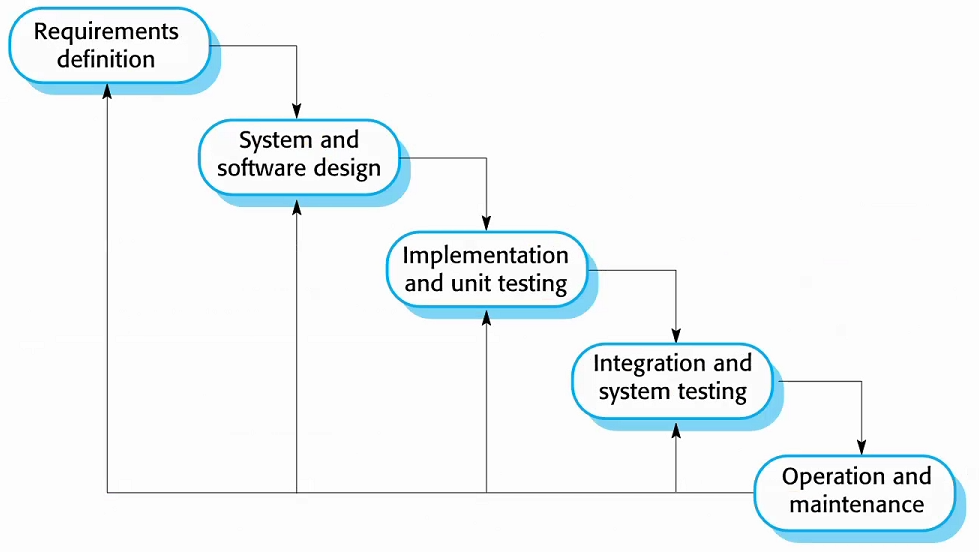
**Introduction Q & A**  
**Q1**. 소프트웨어 공학이란?  
**Q2**. What are the fundamental software engineering activities? (소프트웨어 프로세스 4단계)  
**Q3**. Generic products와 Customized products 차이  
**Q4**. What are the essential attributes of good software  
**Q5**. What are the two fundamental types of software product?  
**=>**A5. Generic products that are designed to meet the needs of many different customer  
Customized products designed to meet the specific needs of a single customer  
**Q6**. What are three key characteristics of the engineering of web-based software engineering?  
**=>**A6. Software reuse, requirements for those systems cannot be completely specified in advance, UI is constrained by the capabilities of web browser  
**=>**세훈: Software reuse, should be developed and delivered incrementally, user interfaces are constrained by the capabilities of web browser  
**Q7**. Briefly discuss why it is usually cheaper in the long run to use software engineering methods and techniques of software systems.  
**=>**세훈: The majority of costs are the costs of changing the software after it has gone into use

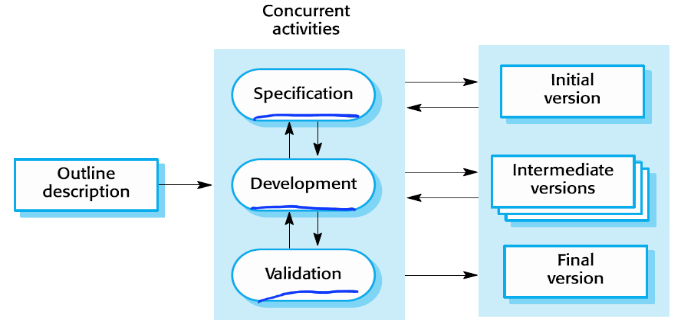
***2. Software Process***

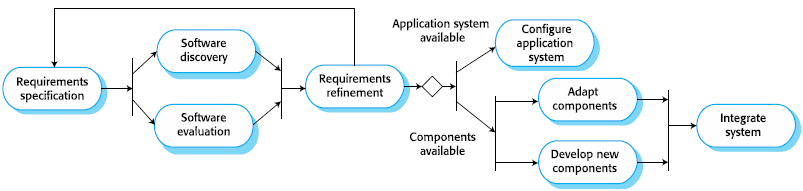
**2.1 Software process models**

**Software process descriptions**  
- Products, Roles, Pre-and Post-condition

**Plan-driven and Agile Process**  
- Plan-driven: is panned in advance, is measured  
- Agile: is incremental, is easier to change process

**The waterfall Model**  
  
pros: large system, plan-driven -> measurable  
cons: phase has to be complete before next phase  
difficult to respond to changing customer requirement  
=>plan-driven

**Incremental development**  
  
pros: Easier to get customer feedback  
Cost of accommodating changing is reduced  
More rapid delivery and deployment  
cons: System structure tends to degrade as new increments are add, Process is not visible  
=>plan-driven or agile

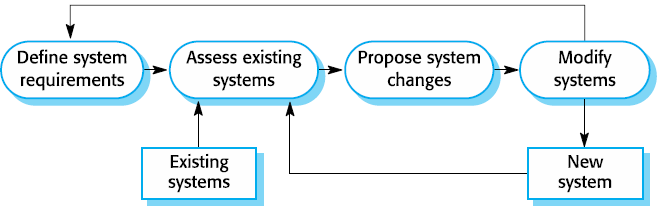
**Reuse-oriented software engineering**pros: Reduced costs and risks  
Faster delivery and deployment of system  
cons: Requirements compromises are inevitable  
Loss of control over evolution

**2.2 Process activities**

**Software specification**

**Software design and implementation**: is concerned with transforming specification into executable software system- Design activities  
: architectural, DB, interface, component design  
- System implementation  
: programming, debugging

**Software validation**  
: Checking and review processes and testing  
- Verification and Validation (V&V)  
Validation: Are we building the right product?  
=>meet the requirements (specification + α)  
Verification: Are we building the product right?  
=>conform to its specification  
- Testing stages  
component testing->system testing->customer testing

**Software evolution**  


**2.3 Coping with change**

Reason: business/platform change, new technologies  
Cost: rework + new functionality

**Reducing the costs of rework**- Change anticipation: anticipate possible change  
ex) prototype  
- Change tolerance: may be implemented in increment  
ex) incremental development

**Software prototyping**  
- an initial version used to demonstrate concepts  
- delivered quickly to check the customer requirement  
- used in requirements elicitation and validation  
- focus on functional rather than non-functional  
- Throw-away prototype: decarded after development  
pros: Improve usability/design quality/maintainability, Closer match to real need, Reduce development effort

**Incremental delivery**  
- incremental development: agile method  
pros: System functionality is available earlier,  
Early increments act as a prototype, Lower risk  
The highest priority system service=>the most testing  
cons: Hard to identify common facilities,  
Conflicts organization where the complete system specification is part of the development contract

**Software Process Q & A**  
**Q1**. Validation Verification 차이점  
**Q2**. 사례에 맞는 모델 찾기  
**Q3.** Describe the phase of the waterfall model development (5 phase)  
**Q4**. List generic software models that are used in software engineering?  
**=>**A4. Waterfall model, Incremental development, Reuse-oriented software engineering  
**Q5**. What are the benefits of incremental development, compared to the waterfall model?  
**=>**A5. Easier to get customer feedback  
Cost of accommodating changing is reduced  
More rapid delivery and deployment  
**Q6**. What are the V&V activity that is used in software validation?  
**=>**A6. V&V is intended to show that a system conforms to its specification and meets the requirements of the system customer

***3. Agile software development***

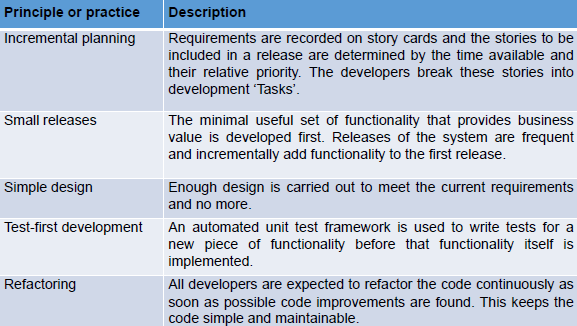
**3.2 Agile methods**

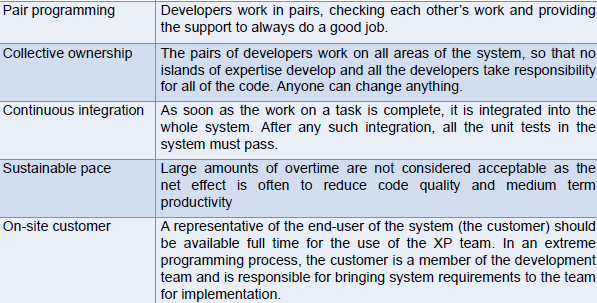
**What are the Agile methods?**  
- Focus on the code rather than the design  
- Are based on an iterative approach to development  
- Evolve SW quickly to meet changing requirements  
=>Aim: reduce overheads and to be able to respond quickly changing requirement without rework

**Agile manifesto**- Individual and interactions over process and tool  
- Working SW over comprehensive documentation  
- Customer collaboration over contract negotiation  
- Responding to change over following a plan

**The Principles of agile methods**  
- Customer involvement  
- Incremental delivery  
- People not process  
- Embrace change  
- Maintain simplicity

**3.3 Agile development techniques**

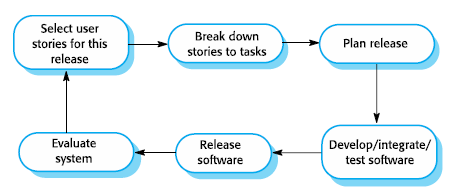
**Extreme Programming**  




**Influential XP Key practices**  
- User stories for specification  
: user is part of the XP team  
- Refactoring  
: to make changes easier when they are implemented  
- Test-first development (test-driven development)  
: Writing test before code clarifies, test is automated  
cons: Programmer may write incomplete test  
Some test can be difficult to write incrementally  
Difficult to judge the completeness of test  
- Pair programming  
: involves programmers working in pairs

**Agile software development Q & A**

**Q1**. What are 4 important characteristics of extreme programming?  
**Q2**. Plan-driven과 비교했을 때 Agile의 장점  
**Q3**. List 5 principles of agile method  
**Q4**. Explain how the principles underlying agile methods lead to the accelerated development and deployment of software  
=>A4. User stories for system specification  
Frequent releases of the software  
Continuous software improvement  
Test-first development  
Pair programming  
**Q5**. What is test-first development?  
**=>**A5. Test of the code implementing that feature are written before the code. Tests are automated and all tests are run when new increment is added.  
**Q6**. Briefly describe the extreme programming release cycle.  
**=>**A6.



***4. Requirements engineering***

**4.1 Functional and non-functional requirements**

**Requirement engineering**  
: Process of establishing the services that customer requires and the constraints under which it operates

**What is requirement?**: set out what the system should do and define constraints on its operation and implementation

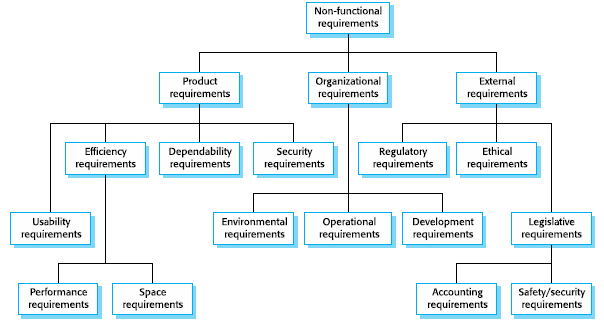
**Types of requirement**  
- User requirements: Written for customer  
- System requirements:   
Defines what should be implemented

**Functional and non-functional requirements**- Functional requirements  
: Statements of the services that the system must provide or are descriptions of how some computations must be carried out (기밀)  
- Non-functional requirements  
: Constraints on the services or functions offered  
- Domain requirements  
: Constraints on the system from the domain of operation

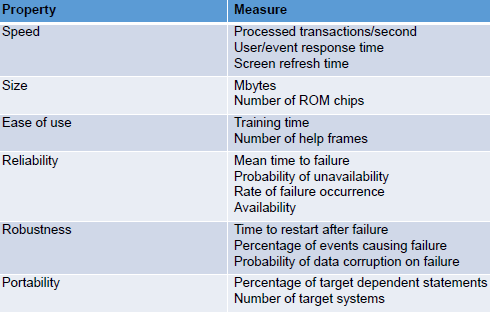
**System stakeholders**  
:end user, system manager/owner, external stakeholder

**Functional requirements**  
- describe functionality or system services  
- Depend on type of (SW, users, system)  
- may be high-level statements of system should do  
- Ambiguous requirements => imprecision(부정확)  
- Completeness & Consistency => Ideal

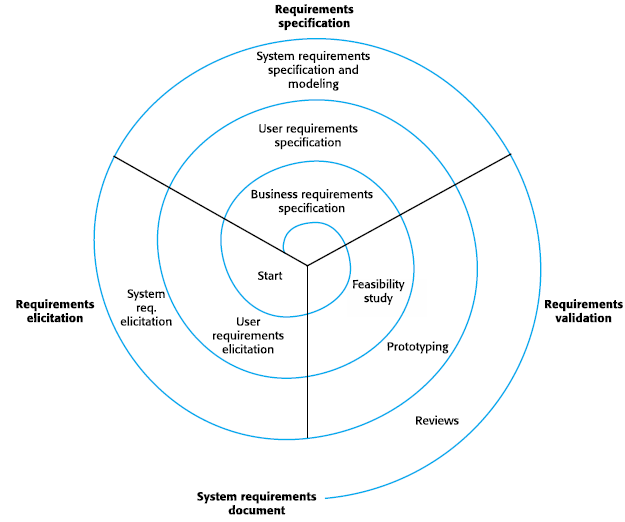
**Non-functional requirements**  
- define system properties and constraints  
ex) timing/development process constraints,   
response time, storage requirements, reliability  
I/O device capability, system representations  
=> Non-functional is more critical than functional  
- affect the overall architecture  
- generate a number of related functional requirement  
- Product/ Organizational / External / Usability(메뉴얼)  
- Goal: General intention of the user such as ease of use  
- Requirement: statement can be objectively tested



**Metrics for specifying nonfunctional requirements**



**4.2 Requirements engineering processes**

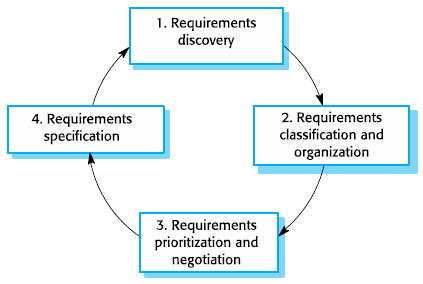
**Spiral view of the requirements engineering process**  


- iterative process that includes requirements elicitation, specification, validation

**4.3 Requirements elicitation**

- Sometimes called requirements discovery, work with system stakeholders to find out application domain  
- Problem of requirements elicitation exists

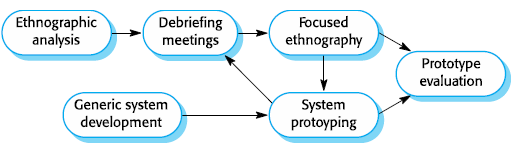
**Process activities (a spiral of activities)**



**Requirement discovery**  
- The process of gathering information  
- Interaction is with system stakeholders

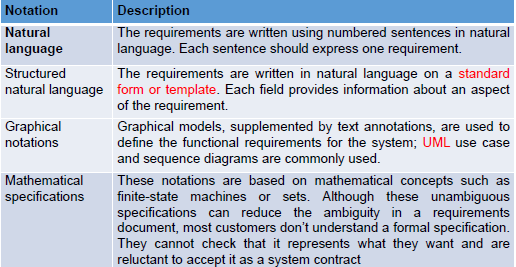
**Interviewing**  
- good for getting understanding of stakeholders  
- problem: Interviews are not good for understanding domain requirements (terminology is hard)

**Ethnography** (문화기술적 연구)  
- effective for understanding existing processes  
- cannot identify new features that should be added  
- developed in a project studying the air traffic control  
- prototype development -> ethnographic analysis  
- problem: it studies existing practices which have some historical basis which is no longer relevant



**4.4 Requirements specification**

- Process of establishing what services are required and the constraints on the system’s operation and development.  
- User requirements have to be understandable by end-users who don’t have a technical background  
- System requirements are more detailed requirement  
- In practice, requirements and design are inseparable

**Way of writing a system requirements specification**

**Natural Language specification**  
+ expressive, intuitive, universal, understood by user  
- lack of clarity, requirements confusion/amalgamation  
- Shall for mandatory, Should for desirable

**Structured specification**  
- requirements are written in a standard way  
- sometimes too rigid  
Form-based specification  
- definition, description of inputs/outputs/action,   
pre and post condition, side effect  
Tabula specification (표로 나타낸 ex.수식)  
- used to supplement natural language

**Use cases**  
- kind of scenario that are included in the UML  
- describe all possible interactions with system

**The software requirements document**  
- what is required of the system developers  
- includes definition of user and system requirements  
- it should set what the system should do (not how)

(생략) **The structure of a requirements document**

**4.5 Requirements validation**

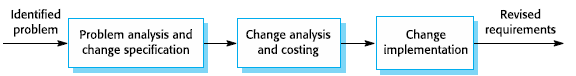
- concerned with demonstrating that the requirements define the system that the customer really wants  
- error costs are high so validation is important  
-> Fixing after delivery cost up 100 times

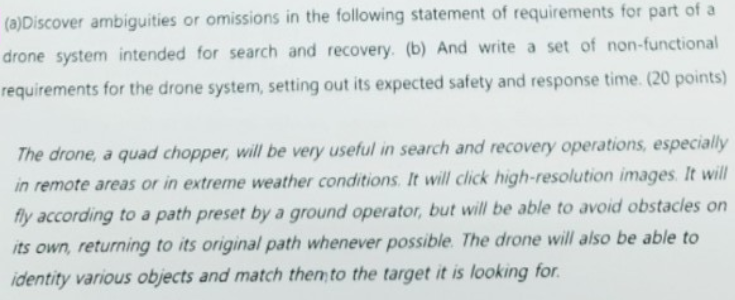
**Requirements checking**  
- Validity: Does the system provide the functions which best support the customer’s needs?  
- Consistency: Are there any requirements conflicts?  
- Completeness: Are all functions required by the customer included?  
- Realism: Can the requirements be implemented given available budget and technology?  
- Verifiability: Can the requirements be checked?

**Requirements validation techniques**  
- Requirements reviews: Systematic manual analysis  
- Prototyping: Using an executable model of the system to check requirements  
- Test-case generation: developing tests for requirements to check testability

**4.6 Requirements change**

- Environment always changes after installation

**Requirements management**  
- process of managing changing requirements  


**Requirements Engineering Q&A  
Q1**. Suggest why it is important to make a distinction between developing the user requirements and developing system requirements in the requirements engineering process.  
**=>**세훈: User requirements have to be understandable by customer who don’t have a technical background but system requirements include more technical information. Therefore, the two requirements need to be distinguished because they have different purposes.  
**Q2**.   
**=>**A2. (a) ambiguities or omission  
- Where will the picture be stored  
- What will happen when the target is found  
- What is the exact standard of extreme weather?  
- How far can drones fly?  
(b) set of non-functional requirements  
- Drone travel range  
- Response time  
- Send data at the right time  
**Q3**. What is ethnography and how is it used in requirement elicitation?  
**=>**A3. Ethnography is an observational technique where an analyst spends a period of time observing work and noting how the participants carry out their test. It is particularly useful in identifying essential cooperation in work processes.  
**Q4**. The processes used for requirements engineering very wisely depending on the application domain, the people involved and the organization developing the requirements. However, there are a number of generic activities common to all process. **Describe the generic activities in detail**.  
**=>** A4.   
requirement elicitation, specification and validation  
**Q5.** Describe the requirements engineering process and software specification.  
**=>** A5. Requirements engineering process:  
requirement elicitation, specification and validation  
Software Specification: The process of establishing what services are required and the constraints on the system’s  
**Q6**. What are user requirements and system requirements?  
**=>**A6. User requirements are statements in a language that is understandable to a user of what

services the system should provide and the constraints under which it operates.

System requirements are more detailed descriptions of the system services and constraint,

written for developers of the system.  
**Q7**. List 3 types of non-functional requirement  
**=>**A6. Product/Organizational/External requirements  
**Q8**. What checks should be applied during requirements validation?  
**=>**A8. Validity check, Consistency check, Completeness check, Realism check, Verifiability  
**Q9.** What is the distinction between functional and non-functional requirements?  
**=>**A9. Functional requirements define what the system should do. Non-functional requirements are not directly concerned with specific system functions but specify required system properties or place constraints on the system or its development process  
**Q10.** What are 6 metrics for specifying nonfunctional requirements?  
**=>**A10. Speed, Size, Ease of use, Reliability, Robustness, Portability  
**Q11**. What are 4 activities for the requirements elicitation and analysis process?  
**=>**A11. Requirements discovery, Requirements classification and organization, Prioritization and negotiation, Requirements specification  
**Q12**. List three requirements validation techniques  
**=>**A12. Requirements reviews, prototyping, test-case generation

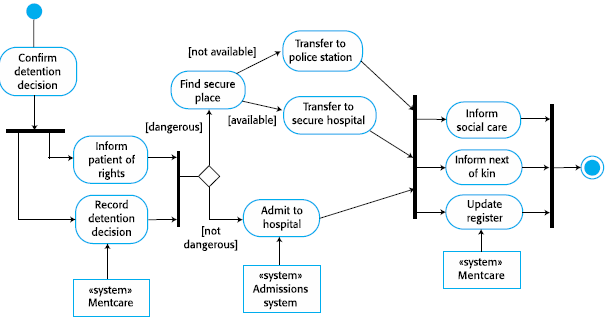
***5. System Modeling***

: is process of developing abstract models of system,  
it helps the analyst to understand and used to  
communicate with customer

Existing system->its strength and weakness  
New system->help explain the proposed requirement

**System perspective**  
external /interaction / structural/behavioral perspective

**5.1 Context model**

: illustrate the operational context of a system  
->show the other system in the environment  
=>Process model  


**System boundaries**  
: define what is inside and what is outside the system

**5.2 Interaction model**

: helps to identify user requirement

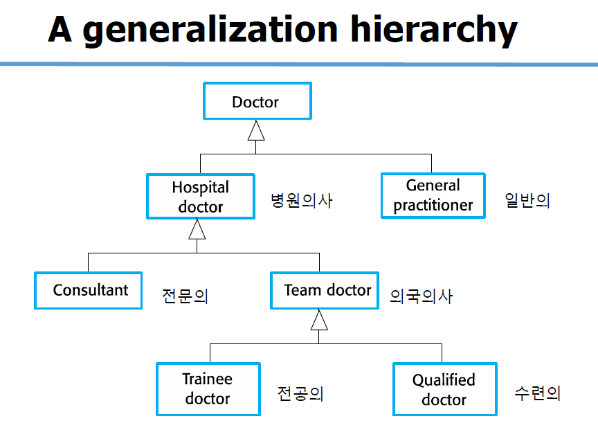
**Use case diagram**  
: describe interactions between a system and actors

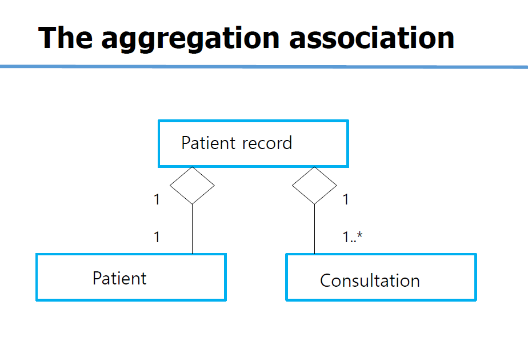
**Sequence diagram**  
: shows the sequence of interactions

**5.3 Structural model**

: display the organization of a system  
static model(structure), dynamic model(executing)

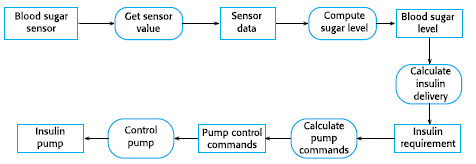
**Class diagram**  
: show the class and the associations between classes

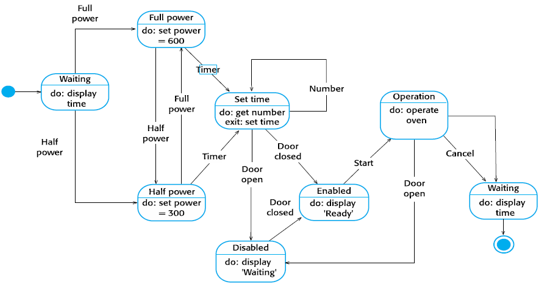
**Generalization  
**

**Aggregation**

**5.4 Behavioral model**

: show what happen / what is supposed to be happen  
=> stimulate by data / event

**Data-driven model (Activity diagram)**: the sequence of actions involved in processing data  


**Event-driven model (State diagram)**: show how a system responds to events  


**System modeling Q & A**

Q1. What do you understand by the system context and interaction model?  
=>A1. System context is a static model of the other systems in the environment  
An interaction model is a dynamic model that describes how the system interact with environment  
**Q2**. **Develop a sequence diagram** showing the interactions involved when a student registers for a course in a university. Courses may have limited enrolment, so the registration process must include checks that places are available. Assume that the student accesses an electronic course catalog to find out about available course.  
**Q3**. Based on your experience with a bank ATM, **draw an activity diagram** that models the data processing involved with a customer withdraws cash from the machine.

