คำศัพท์software engineer

- Software
  - คือ โปรแกรมคอมพิวเตอร์หรือเอกสารที่เกี่ยวข้อง
- software engineering คือ วิศวกรรมซอฟต์แวร์ เกี่ยวข้องกับการผลิต software ในทุกๆ ขั้น
- good software
  - ควรมีคุณสมบัติที่ตามลูกค้าต้องการ (attributes)
    - แก้ไขได้ (maintainable)
    - เชื่อถือได้ (dependable)
    - ใช้ได้ง่าย (usable)
- Dependability and security
  - ต้องมีความน่าเชื่อถือได้ (reliability), ความปลอดภัย (security), ปลอดภัย (safty) และความพร้อมใช้งาน (availability)
- Acceptability
  - การยอมรับ ของแนวคิดที่ต้องได้รับการยอมรับจากผู้ใช้ (ตามความต้องการผู้ใช้)
- Stakeholder
  - ผู้ที่มีส่วนเกี่ยวข้องหรือผู้มีส่วนได้เสียในทุกๆ กระบวนการผลิต software
- Concurrency
  - การทำสองสิ่งพร้อมๆ กันได้
- Portability
  - ความสามารถในการเคลื่อนย้ายหรือพกพาได้
- Modifility
  - สามารถปรับปรุง เปลี่ยนแปลงได้ง่าย
- Feasibility
  - คือ ความเป็นไปได้
- Ubiquitous
  - การเข้าถึงข้อมูลได้ในทุกที่ ทุกเวลา

Software Development

content หลักภายในบท

ย่อยลงมาจากสีเหลือง

บทที่ 1 Introduction

- อารมณ์แบบ software แฝงอยู่ในทุกๆ อย่าง run โลกนี้
- There are no notation, methods or techniques for software engineering
  - different type of software require different approches
- Software failures
  - increasing system complexity
    - ยิ่ง complex ยิ่งต้องการ technique ที่ซับซ้อนมากขึ้นกว่าเดิม
  - failure to use software engineering methods
    - บางบริษัทก็พัฒนางานของตัวเองโดยไม่ได้ใช้ วิธีการเหล่านี้
      - ทำให้ software แพง (เวลา, ความยืดหยุ่น, ค่าใช้จ่าย) และไม่สามารถไว้วางใจได้
- Professional Software Development
  - What is software?
    - computer program and associated documentation
  - What are good artibute of good software?
    - deliver the require functionallity and performance → up to the application
    - dependable, maintainable and usable
  - What is software engineer?
    - engineer discipline, concerend with all aspect of software production
  - What is the best software engineer techniques and methods?
    - different type of techniques are appropriate for different type of system
    - ไม่มีวิธีที่เหมาะสมกับทุกอย่าง
  - There are 2 kinds of software product
    - 1. generics product
      - sold on the open market
    - 2. customized
      - made for particular customer

  → เดี๋ยวนี้ software ส่วนใหญ่ base on generic product หมด
    - เช่น ERP

↳ software engineering
  ↳ an engineer discipline that is concerned with all aspect of software production from the early stages of system specification through to maintaining the system
    ↳ acceptability (ยอมรับได้)
      ↳ design ต้องเข้าใจและใช้งานได้ง่าย { understandable, usable, and compatible }
    ↳ Dependability and security (น่าเชื่อถือ)
      ↳ reliability, security, and safety
    ↳ Efficiency (ประสิทธิภาพ)
      ↳ should not make wasteful of system resources { memory, cpu }
    ↳ Maintainability (ดูแลรักษา)
      ↳ software should be written in such a way that it can evolve to meet the changing need of customers.
  ↳ All aspects of software production
    ↳ ระบบ, ขั้นตอน, เครื่องมือ
  ↳ Engineer is about getting the result of the required quality within schedule and budget.
  ↳ Software Engineer is important for 2 reasons
    ↳ ① we need to be able to product reliable and trustworty system
      ② failure to use software engineer method lead to higher costs
  ↳ software process : sequence of activities, leads to the production of a software product
    ↳ ① software specification การกำหนดของ software
      ② software development การพัฒนา software
      ③ software validation การตรวจสอบว่าตรงตามความต้องการของลูกค้า
      ④ software evolution การพัฒนา software ให้ปรับเปลี่ยนตามความต้องการของ customer
  ↳ four related issued that affect many different types of software
    ↳ ① Heterogeneity (ความหลากหลาย ทั้งทาง sw, hw, os)
      ↳ the challenge here is to develop techniques for building dependable software that is flexible enough to cope with this heterogeneity

② business and social change

↳ need to be able to change their existing software, and deploy new one

③ security and trust

④ Scale software

↳ Software engineer diversity

↳ การใช้งาน techniques ต่างๆ ขึ้นอยู่กับ คน, ชนิดงาน, ความชำนาญ

↳ SEMAT : fundamental meta-process ; instantiated to create different kind of process.

↳ มีหลายชนิดของ application มากมายบนโลกนี้ , แต่ละชนิดก็ต้องการ technique ที่แตกต่างกัน

↳ มีพื้นฐานคล้ายกัน ① they should be developed using a managed and understood dev-process

② Dependability and performance : Software should behave as aspect

③ Understand software specification and requirements

④ You should make effective use of existing resource

↳ Software engineering ethics

↳ You must also behave in an ethical and morally responsible : ก็ตาม sense อ่ะ

↳ มีพฤติกรรมที่เราไม่ต้องทำก็ได้เพราะมันจะไม่ต้องอยู่ในกฎหมาย

↳ confidentiality : the state of being to keep the secret

competence : the ability to do something successfully ; don't misrepresent your level

intellectual property : copy right law

computer misuse : การใช้ตัวมันในการที่ผิดต่างๆ

↳ UML (unified modeling language)

↳ activity model that illustrates how the software transform an input to a sequence of command.

↳ Key point

↳ software engineer is an engineering discipline that is concerned with all aspects of software production

↳ software is not only the program but also included all eletronic documentation, required by system users, quality assurance staff and developer

↳ คุณสมบัติที่สำคัญ : maintainability (สามารถปรับปรุงตาม requirement ได้)

dependability and security (ปลอดภัย, เชื่อถือได้)

efficiency (ใช้ resource ในระบบอย่างมีประสิทธิภาพ)
acceptability (understandable, usable, compatible

↳ software process : the high level activities of : specification
development
validation
evolution

↳ each types of software requires appropriate software engineering techniques

↳ fundamental ideas of software engineering : software process,
software dependability and security,
requirement engineering,
software reuse

↳ don't forget about the ethical terms.

บทที่ 2 Software process

↳ set of related activities that leads to the production of a software-system. (have many type)

↳ although there are many different software processes, they all must include is some form

↳ ① software specification : the functionality of the software and constraints on its operation must be defined

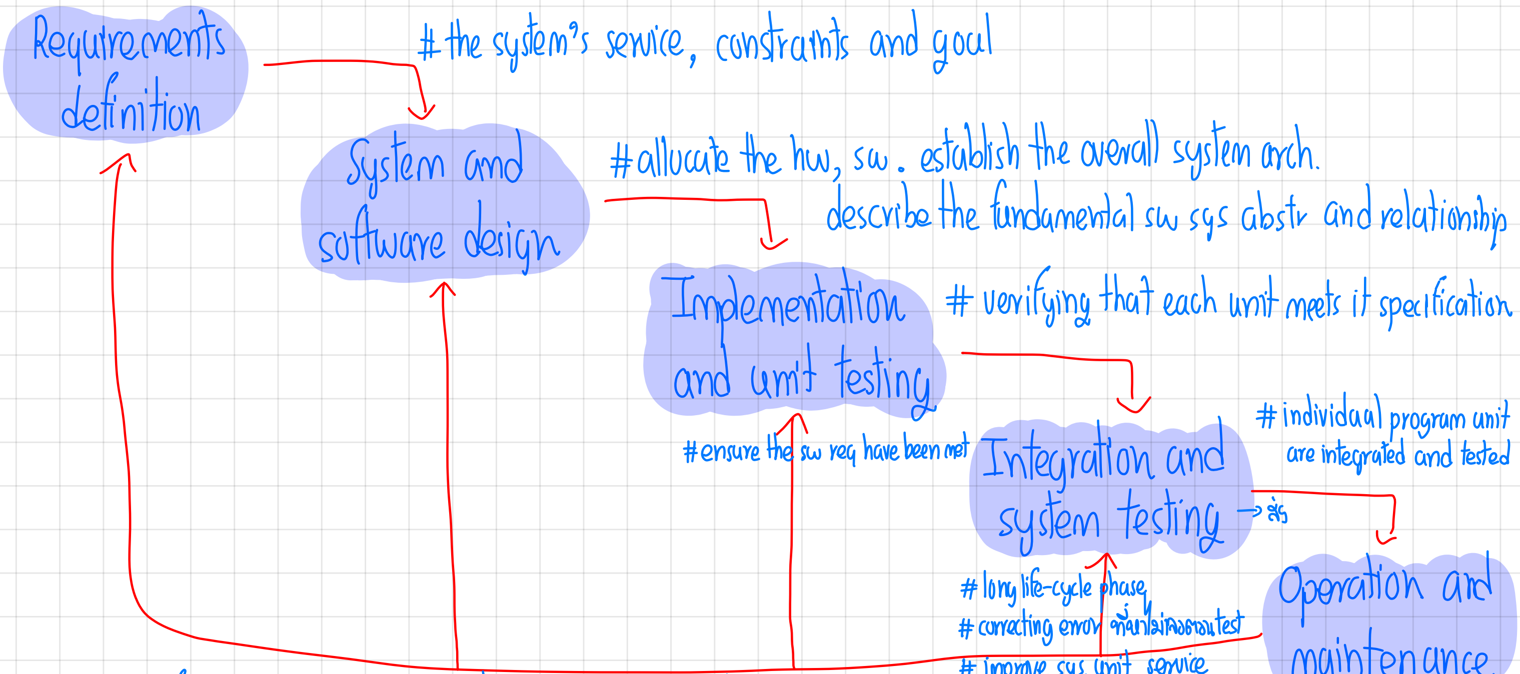
② software development : the software to meet the specification must be produced

③ software validation : the software must be validated to ensure it does what the customer want

④ software evolution : the software must evolve to meet changing customer needs.

↳ When desciping process, it is also important to describe who is involved, what is produced and condition that influence the sequence of activities

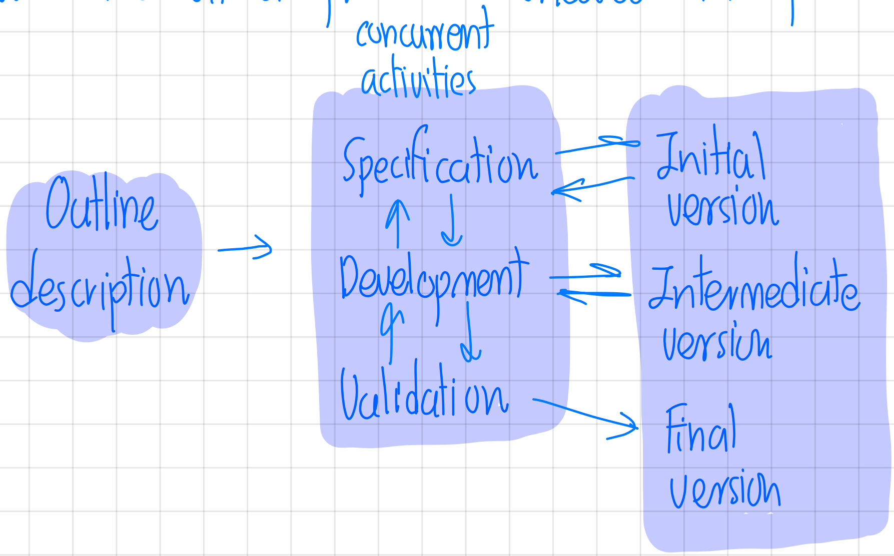
- ↳ Products or deliverables : outcomes of a process activity
  - ↳ architectual design --outcome--> software architecture
- ↳ Role : the responsibilities of the people involves in process
- ↳ Pre-post-conditions : conditions that must hold before and after a process activity
  - ↳ precondition → meet the requirement
  - ↳ postcondition → after activity is finished
- ↳ For safty-critical system, a very structure development process is required
  business system, rapidly change requirements, more flexible, agile is better
- ↳ Software process models
  - ↳ = software develop life cycle (SDLC)
  - ↳ the right process depends on requirements, where the software will be used, type of sw.
    - ↳ safety-critical software : waterfall model
      normal software : incremental development
      business software : configuring and integration
  - ↳ The waterfall model



- ↳ the following phase should not start until the previous phase has finished
  - ↳ see something?
    - ↳ if the req is expensive → ขอปรับ, ลด → รอลูกค้า ok → เสียเวลา
    - ↳ ถ้า requirement ไม่ชัดก็ไม่สามารถเริ่ม process ได้

↳ in reality, software has to be flexible, accomodate change
↳ waterfall : embedded systems → sw interface with hw
→ inflexibility of hw
critical systems → need safety and security
large software system → dev by several partner companies
↳ ไม่เหมาะกับงานที่มีความเปลี่ยนแปลงอย่างรวดเร็ว (เหมาะกับ agile มากกว่า)

↳ **Incremental development**
↳ idea of developing an initial implementation, getting feedback from users and other and evolving the software through several version until the required system has been devel
↳ ทุกขั้นตอนของ fund. software processes interleaved with rapid feedback across activities



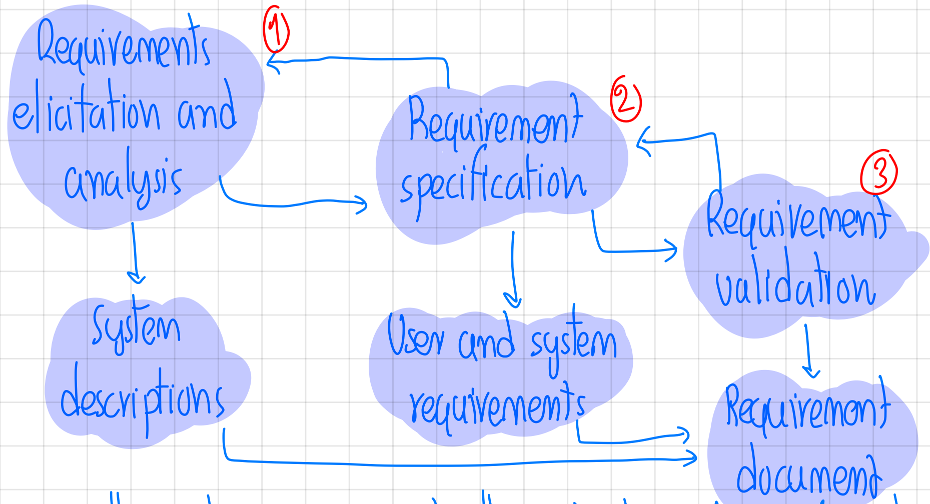
↳ if agile approach is adoped, the early increments are identified, but the development of later increments depends on progress and customer priorities
↳ major advantages over waterfall model :
↳ cost of implementing requirements change is reduced
↳ customer ให้ฟีดback จากที่ได้เห็นจริงๆ ไม่ต้องรอใช้ doc เขาไม่ชอบอ่าน, อ่านแล้วเบื่อ
↳ early delivery and deployment of useful sw to customer
↳ even if all of functionallity has not been included → ได้ลองก่อนแล้วค่อยๆตาม function
↳ big problem → the process is not visible → if dev quickly, not cost effective to produced document
↳ need regular deliverable to measure process

↳ the system structure tends to degrade as new increments are added
  ↳ regular change lead to messy code as new functionality is added
    ↳ ป้องกันการ regradation → you should regularly refactor the software
↳ incremental development become particular acute for large, cumplex, long-lifetime system
  ↳ large framework need a stable framework or architecture → ต้องมีการวาง plan ดีๆ

↳ Integration and configuration
  ↳ software reuse → หา code ที่คล้ายๆ กันแล้วไม่ต้องเขียนใหม่เองทั้งหมด
    ↳ เอาของเก่ามาใช้, ปรับใช้ รวมเข้ากับของใหม่
  ↳ 3 types of software components are frequently reused
    ↳ ① stand-alone application → เอาของเก่าทั้งก้อนมาปรับใช้ใหม่
      ② collections of objects → เอาส่วนประกอบของระบบเก่ามาใช้งาน
      ③ web service → เอา service เก่าทั้ง service มาปรับใช้
↳ the stages in this process are
  ↳ ① requirement specification : don't have to be elaborated in detail but should include brief descriptions of essential requirements and desirable system features.
    ② software discovery and evaluation
      ↳ candidate components and system are evaluated if they meet requirement, sutable for use in system
    ③ requirement refinement
      ↳ requirements are refined (กลั่นกรอง) ด้วย component แบบไหนที่เราเอามาใช้กับระบบนี้ได้
      ↳ the requirements are modified to reflect available components, the system is re-defined
      ↳ where modifications are impossible, the component analysis activity may be reenter to search for alternative solutions.
    ④ Application system configuration
      ↳ if the application meet the requirements, it may then be configured for use to create new system
    ⑤ Component adaption and integration
      ↳ If there is no off-the-shelf system, individual reusable components may be modified and new components developed. the integrated to create the system

- reuse-oriented software engineering, based around configuration and integration
  - ลดการผลิต software, ค่าใช้จ่าย, ความเสี่ยง ทำให้ส่งงานไวขึ้นด้วย

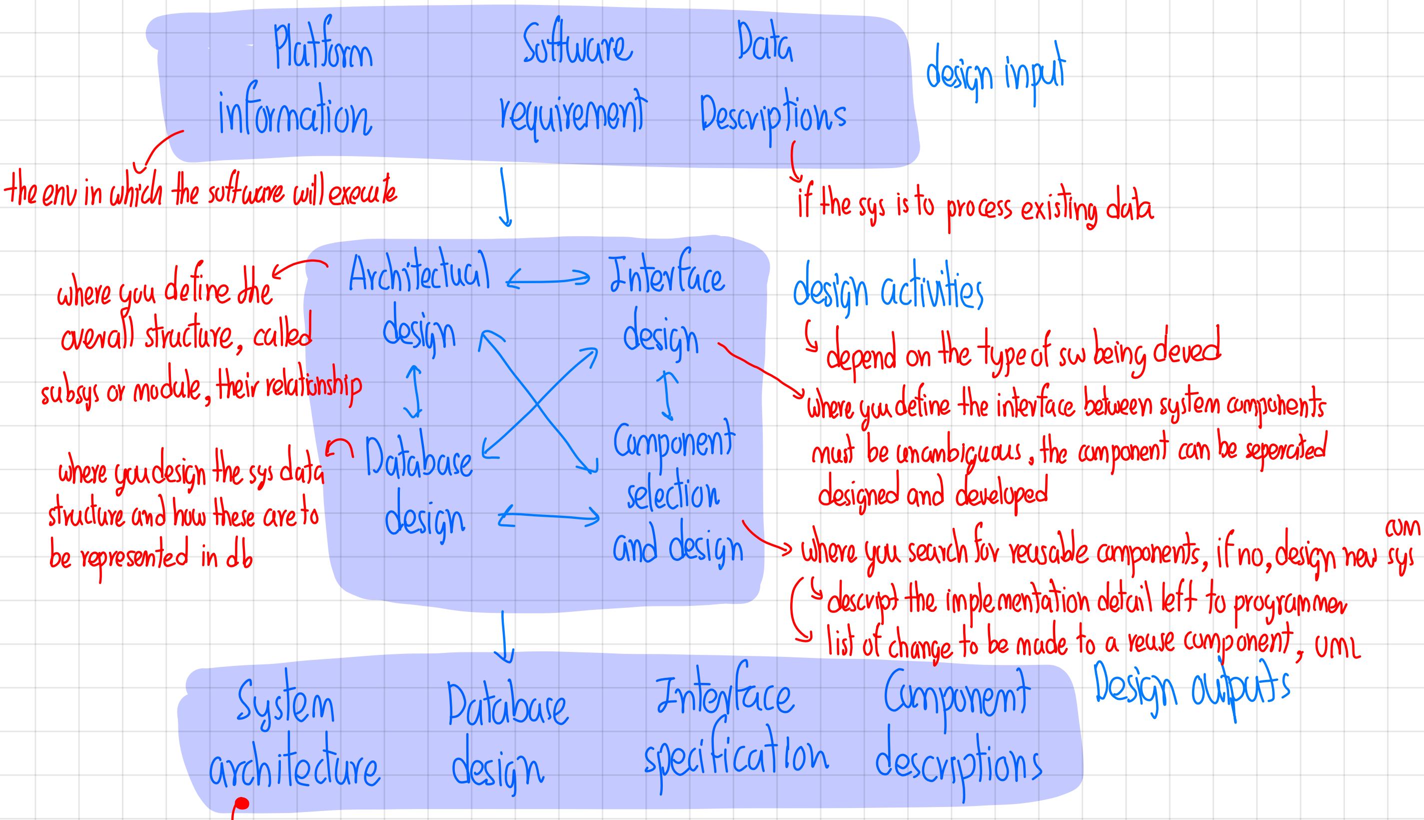
- Process activities
  - Software specification, requirement engineer (→ ก่อนทำขั้นตอนนี้ company จะหา feasibility ของความต้องการของตลาด)
    - the process of understanding and defining what services are required from the system and identify constraints on the os.
    - very critical → mistake leads to later problems in system designed and implementation
    - the requirement engineer process aim to produce an agreed requirements doc.
      - หาสิ่งที่ผู้ว่าจ้าง (stakeholder) require
      - end-users and customers need high-level of the requirement; system developers need a more detailed system specification



- ① deriving the system requirements through observation of existing system, discussions with potential user and procurer, task analysis...
- ② translating the information gathered during requirements analysis, into set of requirement
  - มีประเภท 2 รูปแบบ user requirement: abstract statements of the sys requirement
    system requirement: more detailed description of the functionality to be provided
- ③ check the requirement for realism, consistency (ความสอดคล้อง), completeness
  during the process, error in req(n) are inevitably discovered → must be modified

↳ ใน agile, requirement specification is not seperat activity ; part of the sys development
  ↳ requirements are informally specified for each increment
    ↳ การ elicitation (การดึงออกมา) requirements จาก user ที่ทำงานร่วมเดียวกัน

↳ **Software design and implementation**
  ↳ process of developing an executable system → sometime involved sw designed, programing
  ↳ if agile approach, design and implementation are interleaved
    ↳ no formal-designed doc ; the sw is still designed ; in-formally way
  ↳ sw design = description of the software to be implement, data model and structure, the interface between the sys component, algorithm used.
    ↳ ไม่ได้ design ทีเดียว ทำเป็น stage ๆ ค่อยๆเพิ่ม detail กลับไปแก้ของอันเก่า

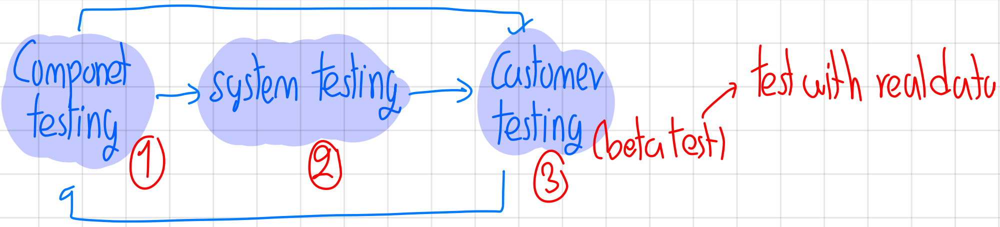


detailed design documents setting out precise and accurate description of the system
  ↳ upto the method and model you use
    ↳ if agile are used, the output of design process may be a code in the program

↳ บางคนก็เริ่มที่ component ที่เข้าใจก่อน บางคนก็ที่ไม่ค่อยเข้าใจก่อน หรือ ส่วนที่เข้าใจก็จะทำเสร็จ
↳ normally programmer carry out some testing code they have developed, often reveal bug, which must be removed finding and fixing "debugging"
  ↳ testing establishes the existence of defects (bug)
    debugging is concerned with locating and correcting the defects

Software validation

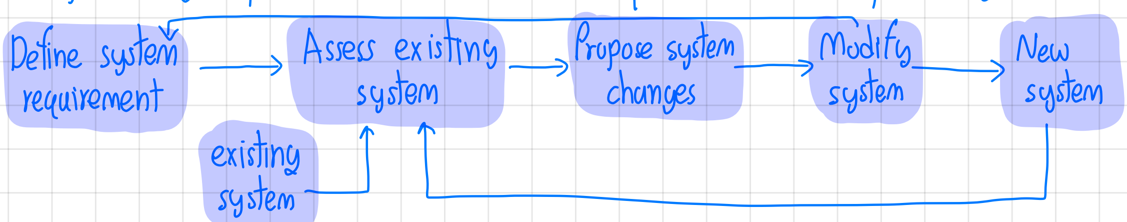
- show tha a system both conforms to its specification and meet the expectations of system customer
- Program testing → ใช้โปรแกรมรันโดยข้อมูลจำลอง → เป็นหลักการของการทำ validation



- System component are individually tested, then integrated system is tested
- ① Component testing
  - testing by the people developing the system, independently with out other component
  - component may be function, object class
- ② System testing
  - components are integrated to create complete system
  - concerned with finding errors that result from unanticipated interaction, showing system meets it functional and non-functional req(n)
    - หรือ test ตั้งแต่เป็น subsystem ก่อนเอามาประกอบรวมกัน
- ③ Customer testing
  - may reveal error and omission in the system req(n)
  - show how well the software product meets the customer's needs.
- ถ้าเป็น increment develop → each increment should be tested as it is developed
- ถ้าเป็นแบบ beta ได้ไหม → ให้ลูกค้าช่วยดูได้ใช้ระหว่างพัฒนาเพื่อ report bug → beta test

Software evolution

- ความยืดหยุ่นของ software เป็นเหตุผลหลักของการที่เราจะนำไปใช้ reuse
- hw change is very expensive, software is cheaper than corresponding changes to hw



↳ Coping with change

↳ whatever software process model is used, it is essential that it can accommodate changes to the software being developed.

↳ change add to the cost because the completed component has to be redone may be design the new system

↳ 2 related approach may be used to reduce the cost of rework

↳ ① Change anticipation

↳ ...

↳ prototype system may be developed to show some key features of sys customers

② Change tolerance

↳ where the process and software are designed so that changes can be easily made to the system

↳ 2 ways coping with change

↳ ① system prototyping → ...

↳ in requirement engineering, ... valid requirement

in system design process, explore software solution

↳ They may get new idea for requirement, strength and weakness in sw

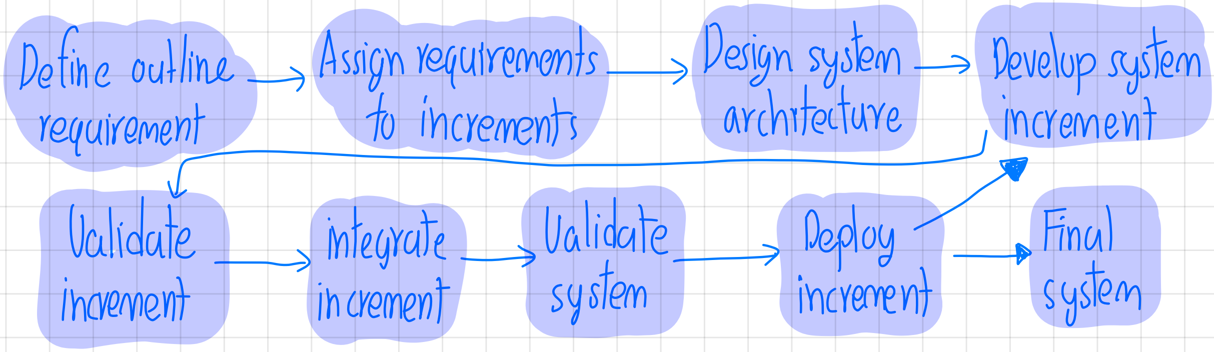
↳ The prototype is developed, it may reveal error and omission in sys reqm

↳ some function ... prototype → modified reqm

↳ check the feasibility of proposed design

↳ the next step is to decide what to put into, to reduce cost and accelerate the delivery schedule, you may leave some functionality out of prototype relax non-functional reqm)

↳ prototype evolution → a general problem with prototyping is that user may not use it in the same way as they use the final system



↳ ② Incremental delivery

↳ each increment providing a subset of system functionality

↳ allocation of services to increment depend on the services priority → higher first

↳ การวิเคราะห์ requirement เพื่อทำการสร้าง increment เป็นเรื่องปกติ
แต่ถ้าเปลี่ยน requirement ในระหว่างสร้าง increment เสียแรง!

↳ advantage :) increment → ให้ user ลองใช้, เป็น part of the real system → cus ไม่ต้อง เรียนรู้ใหม่

:) cus do not have to wait until the entire system is delivered

:) easy to change into the system

:) highest priority services are delivered first and later increment then integrated
↳ no software failure in the most important part

↳ disadvantage :) user ชอบ old version มากกว่า ทั้งๆที่ new version ยังไม่เสร็จ
ในกรณีที่แบบ มีเปลี่ยนทั้ง database, user interface

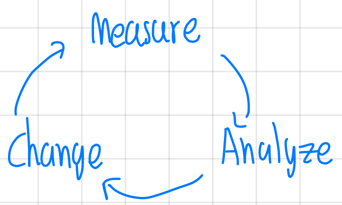
ในระบบใหญ่ๆ จะ ถ้า requirement ไม่เคลียร์?

:) hard to identify the common facilities that are all need by increments where the requirement are not defined in detail.

:) ไม่มี system specification จนกว่า increment สุดท้ายจะออกมา ยากต่อการเขียนสัญญา กับหน่วยงานขนาดใหญ่ หรือ รัฐบาล

↳ Process improvement

↳



① Process measurement

↳ You measure one or more attributes of software process or product

② Process analysis

↳ assess the process weakness and bottleneck are identified

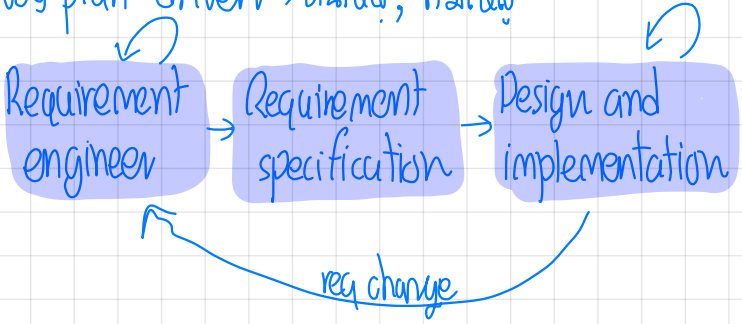
③ Process change
↳ address some of identified process weakness

↳ key point
↳ software process are the activities involved in producing a software system. (abstract)
↳ general model → waterfall model, incremental development, reusable component configuration and integration
↳ requirement engineer is the process of developing a software specification
↳ design and implementation processes are concerned with transforming a requirement spec(n) into an executable software system
↳ software validation is the process of checking that the system conforms to its spec(n) and meet the real needs of users
↳ software evolution : take place when you change existing software system to meet new req(n)
↳ Processes should include activities to cope with change. This may involved a prototype phase ; avoid poor decisions on req(n) and design. Change may be made with out disrupting the system as a whole.
↳ Process improvement is the process of improving existing software process to improve sw quality, lower development costs, reduce development time. cyclic process above

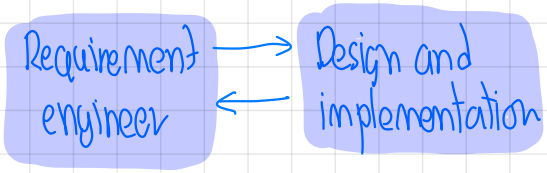
บทที่ 3 Agile software development in increment development

↳ The process of specification, design and implementation are interleaved.
↳ The system is developed in a series of increment ; customer give opinion on it system specification

ของ plan-driven → ตายตัว, ชัดเจน



ของ agile approach



requirement and design are dev together

- Agile method
  - the philosophy behind
    - customer involvement: customer should be closely involved throughout the development process
    - embrace change: expect the system requ(n) to change; accomodate the change
    - incremental delivery:
    - maintain simplicity: eliminate complexity for the system
    - People, not process: team member should be dev their way of working without process
- Agile development technique
  - User stories ← XP นิยมใช้เลย
    - story แบบที่เป็นภาษาคนที่ customer เขาต้องการ นำไปสร้าง story card มีหน้าที่ briefly descrip a story that encapsulate the customer needs.
      - ข้อเสียแบบ ถ้าผู้ใช้เพิ่มเข้าไปเยอะขึ้น ก็ยาก...
      - ทำการ break down story card ลงไปเป็น task อีกที แล้วทำอันที่สำคัญที่สุดก่อน เพื่อให้ user บอกความต้องการที่แท้จริง
  - Refactoring
    - You should design it for change. you should anticipate future changes to software and design.
    - suggest code being dev should be constantly refactored even there is no immediate need
    - fundamental problem of increment development is that local change tend to degrade the software structure
      - เราเลยต้องทำการ refactor กันบ่อยๆ
  - Test-first development → ทำ test card → input, test, output
    - one of different between increment dev and plan-driven dev is the way that system is tested
      - increment dev → no system specification that can be tested by external test team
        - มีแค่การ test แบบ informal
      - plan-driven → มีแผนมาเลยเขียน test ก่อนก่อนเขียนโปรแกรม → ทำให้นักพัฒนารู้ว่าใช้ test-case นี้ได้เลย

↳ require there to be clear relationship between system requirements and the code implementing the corresponding requirements.

↳ **Pair programing**

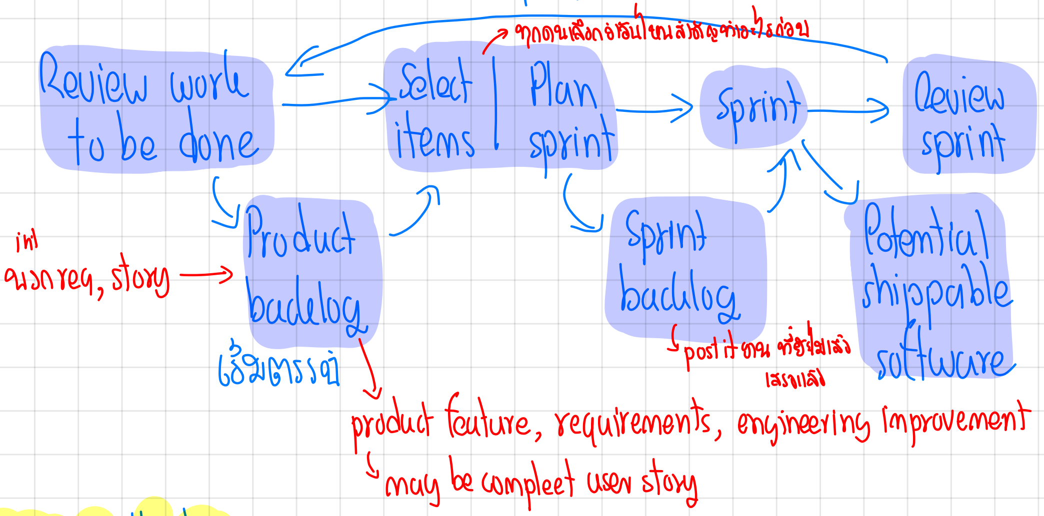
↳ ทำงาน 2 คนกับ code ส่วนเดียวกัน

↳ **Agile project management**

↳ ถึงแม้จะส่งเป็น increment ก็ต้องมีการจัดการ เรื่องของเวลา และการส่งมอบ software อยู่ดี

↳ ทำให้เกิด scrum agile method

↳ แต่มันไม่ใช่ method สำหรับการจัดการ project นะ



↳ **Scaling agile methods**

↳ agile made for small-medium team with small-medium sized system

↳ **Practical problem with agile methods**

↳ for large, long-lifetime system agile approch present a number of problem

↳ informality of agile deevelopment is incomplete with legal approch to contract definition

↳ appropriate for new software dev, not maintenance

↳ large companies maintain their existing software system

↳ decide for small co-located team

↳ เมื่อไป maintain ต่อ : lack of documentation

: keeping the customer involve

: development team continuty

lack of document : agile says "waste time for writing document"

↳ lack of high-quality and readable code

keeping the customers involved : if customers are busy, ? . what about the maintain

↳ เขาจะสนใจอยู่หรอ

↳ Agile and plan-driven methods

- ↳ a fundamental of scailing agile methods is to integrate them with plan-driven approach
- ↳ ดูที่ keypoint กัน (ของระบบ)
  - ↳ ระบบใหญ่แค่ไหน : เล็ก-กลางใช้ agile, ใหญ่ plan-driven
    เป็นระบบแบบไหน : เน้นการทำงานเรียลไทม์ ไป plan-driven
    ระยะเวลาการทำงานนานแค่ไหน : long-lifetime → need more documentation
    short → likely argue doc(n), may be doc(n) not update

↳ Agile methods for large system

- ↳ large system :> large system are usually system of system
  :> are browfield (พื้นที่ที่เคยใช้ในอุตสาหกรรมที่สามารถปลูกบ้านได้แล้ว)
  they include and interact with number of existing system
  :> concerned with system configuration ถ้าเป็น agile จะเจอการรวม system ลงทำงานร่วมกัน
  :> ถ้าคนที่รู้เกี่ยวกับระบบทำการลาออกไป
- ↳ the scrum team model is maintained
  - ↳ ① role replication
    - ↳ each team has a product owner. อาจจะมีหัวหน้าใหญ่ที่รวมหมดอีกทีก็ได้
  - ② project architects
    - ↳ each team choose product architect, เหมือนเลเยอร์ทั้งระบบ? หรือของส่วนตัวเองอย่างเดียว
  - ③ Release alignment
    - ↳ the date product release from each team are aligned
  - ④ Scrum of scrum
    - ↳ where representatives from each team meet to discuss the process, identify problem and plan the work to be done

↳ Agile methods across organizations

- ↳ It's very hard to introduce agile methods into large companies for a number of reason
  - ↳ Projectmanager who don't have exp on agile may be reluctant to accept risk of new approa

↳ they don't know how it affect on particular project
↳ large organization often have quality procedures and standard that all work have to follow
↳ agile method seem to work well if team members have a relatively high skill level
↳ there may be cultural resistance to agile method → องค์กรยังยึดติดกับวิธีการแบบเดิมๆ
↳ Change management and testing procedure not compatible with agile
↳ process of controlling change to system
↳ All change has to be approved before they are made
↳ สำหรับ agile อาจต้องมีการ refactor
↳ ไม่สามารถทำ external test ได้ → agile เน้นการเขียน requirement

↳ key point
↳ agile method are iterative development methods that focus on reducing process overheads and documentation and on incremental software delivery. They involve customer representatives directly in development process
↳ The decision on whether to use an agile or a plan-driven approach to development should depend on the type of software being developed, the capabilities of dev team and the culture of the company developing system. In pratice, mix type are used
↳ agile development pratices include requirement expressed as user stories, pair programing, refactoring, continuous integration, and test-first development
↳ Scrum is an agile method that provide a framework for organizing agile project. It is centered around the set of sprint, fixed time period. Planning is based on prioritizing a backlog of work and selecting the highest priority tasks for a sprint
↳ To scale agile methods, some plan-based practices have to be integrated with agile prac. (include up-front requirements, multiple customer representative, more documentation, common tooling across project team, and the alignment of release across team.

Scrum

- การออกแบบ, design
- ขั้นตอน : sprint cycle → burn down chart, meeting,

บทที่ 4 requirement engineering first state of software engineering proces

↳ to make clear seperation between these different levels of description.

↳ user requirement = high level abstract requirement → ใช้อธิบายให้ระบบทำงานอย่างไร

system requirement = to mean the detail description of what system should do

↳ function, operating, service

↳ user ต้องใช้งานระบบ lazada → system req(n) → ระบบ generate ขึ้นมา

↳ provide more specific information about the service and function of system, is to be implemented

↳ Functional and non-functional requirements

↳ Functional requirements

↳ describe what the system should do, write in natural language → user can understand

should describe the system functions (input, output) and exception detail

↳ may be writen at different levels of detail

↳ Imprecision in the requirement specification can lead to dispute between customer and developer → delay system delivery and increase cost

↳ should be both complete and consistent → should not be contradictory

↳ all service and information required by user should be define

↳ Non-functional requirement

↳ are requirement that are not directly concerned with specific service delivered by the sys.

↳ usually specify or constrain characteristics of the system as a whole

↳ The implements may be spread throughout the system.

↳ Non-functional requirement may affect overall architecture of a system rather than individual component.

↳ ข้อกำหนดเกี่ยวกับ budget ของลูกค้า

↳ Product requirement : runtime behavior of the software, security, reliability

↳ Organizational requirements: programing language, environmental reqcn

↳ External requirements: ethical, legislative reqcn

ถ้าตามโจทย์สไลด์ของอาจารย์, product reqcn: mentcare ต้องเปิดให้บริการล่วงหน้าจาก clinic ตามเวลาทำการ

organization reqcn: user ต้องสามารถยืนยันตัวตนด้วยบัตร หรือ บัตรอื่นๆ

external reqcn: implement patient privacy provision.

↳ say nothing about the functionality; clearly identify the constraint

เราอาจจะจำกัดการเข้าไปยังส่วนการอบรมการใช้ software นี้

↳ ; may interact with other functional or non-fun

หลักๆ reliability, safety, confidentialy

↳ การยืนยันตัวตน ก็คือที่ counter จะมีที่รับบัตร แล้วต้องยื่นบัตรยืนยันผล-

↳ Requirement engineering process

↳ elicitation and analysis → convert to standard form (specification) → checking reqcn that actually define the system, customer wants (validation)

↳ Requirement elicitation

↳ to understand how stakeholder might use a new system software to help their work

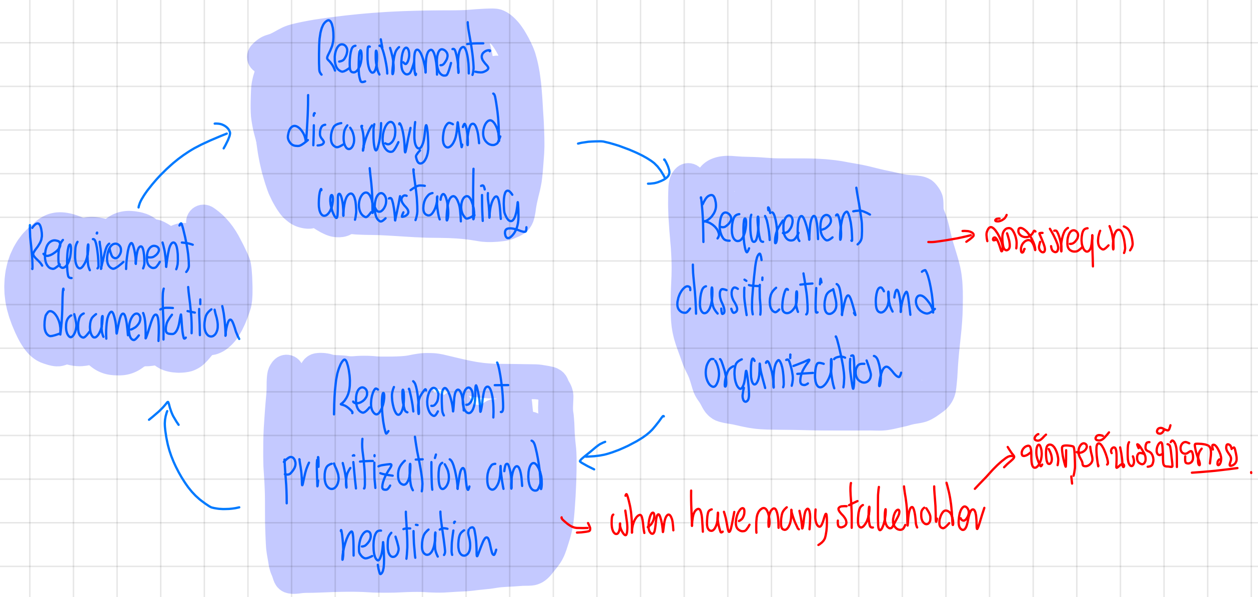
↳ ยาก: they make unrealistic demand 'cause don't know what is feasible.

: express reqcn in their own term

: different stakeholder → different reqcn

: Political factor → เหมือนเขาให้เงินทำงานที่เขาได้ผลประโยชน์ฝ่ายเดียวจากตัว software

: the economic and business environment may change



↳ **technique** : interview, observation
↳ ถามเขาว่าแต่ละขั้นทำอะไร, ระบบเก่าเป็นยังไง, ถ้าระบบใหม่ควรมีอะไรไหม แต่ข้อเสียคือขึ้นอยู่กับคนและองค์กร
↳ reqm derived from the way people actually work.

↳ **Stories and scenarios**
↳ it is easier to relate to real-life example than abstraction description

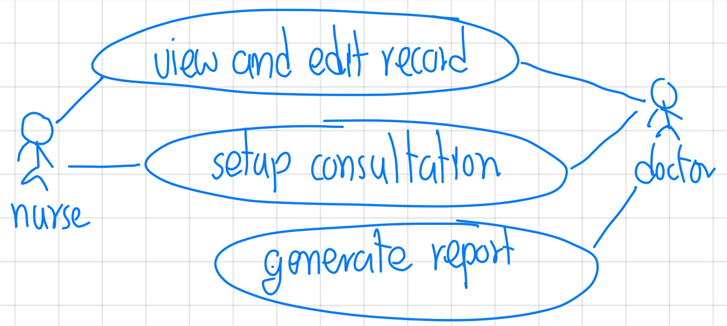
↳ Requirement specification

↳ process of writing down the system and user requirement document
- user ↳ write in natural language
- system ↳ write in natural language in other from
  - ↳ natural language sentences
  - ↳ structured natural languge
  - ↳ graphical notations
  - ↳ mathematical specifications

↳ **natural language specification**
↳ potential vague and ambiguous ไม่ได้ใช้คำที่เห็นความแตกต่างอย่างชัดเจน เช่น ต้องทำ หรือ ควรทำ

↳ **structured specifications**
↳ standard form (template) → อธิบายเป็น function → แล้วอธิบาย description, input, output, action, require, condition
(require, condition: pair)

↳ use case



UML way

↳ the software requirements document
↳ ในส่วนของ document นั้น สามารถให้กับหลายบุคคลใช้รายงานร่วมกันในจุดประสงค์ต่างๆ
↳ เพื่อการเขียน test, พัฒนาต่อ, ทำความเข้าใจ

Requirement validation

- the process checking that requirements define the system that the customer really want
  - error in requirement may lead to cost
- เราต้องเช็คว่าการเขียน requirement ตรงของ customer ไหม, ไม่มีอะไรขัดแย้ง, ความสมบูรณ์ของทุก function, สามารถทำได้ในงบแค่นี้ไหม, ทำ test ตาม case ว่า function ทำงานถูกไหม
  - เขียน test ก่อนทำการเขียนโปรแกรม อยู่ใน test-driven development

Requirement change

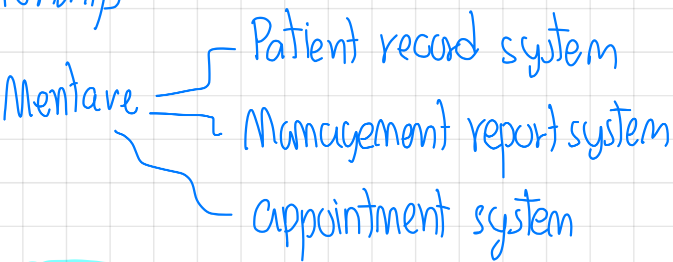
- the business and technical envrm change, budget, have a diverse stakeholder
- **Requirement management planning**
  - requirement identification, change management process, relation between each requirement and system design, tool support
- **Requirement change management**
  - problem analysis and change specification, change analysis and costing, change implementation

key point

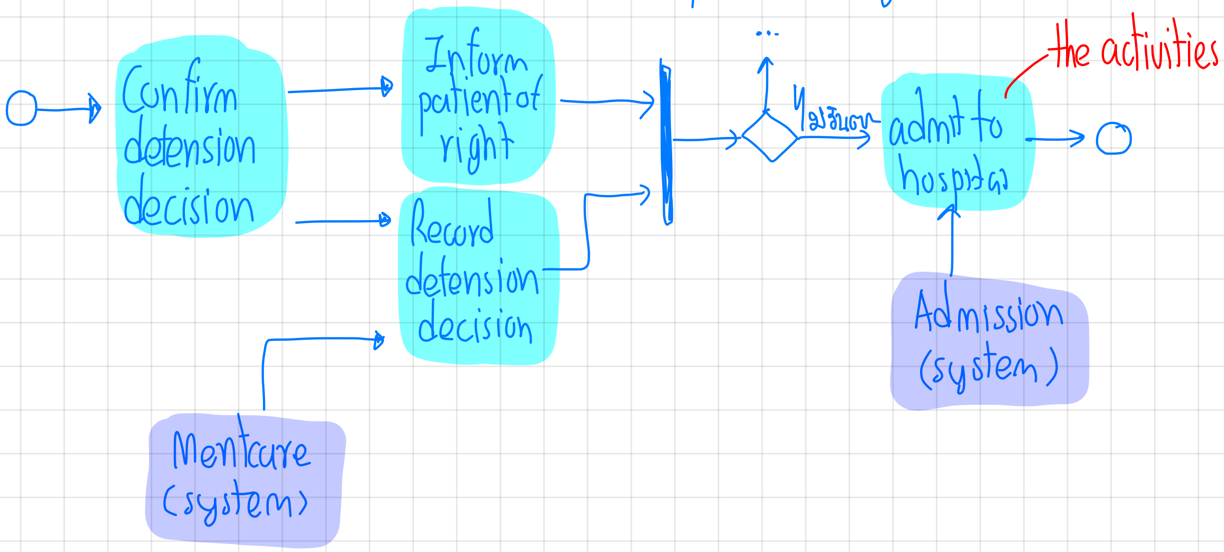
- reqrm for software system → what system should do, define constraints on its operation
- functional reqrm → service that system must provide
- Non-functional reqrm → often constrain the system being developed
- reqrm engineer process → reqrm elicitation, specfication, validation, management
- reqrm elicitation is → reqrm discovery, classification, organization, negotiation, doc
- reqrm specfication → formally documenting the user and system requirement and creating software requirement document
- reqrm validation → checking the requirement for validity, consistency, completness, realism and verifiability
- reqrm management → the process of managing and controlling these change.

บทที่ 5 system modeling

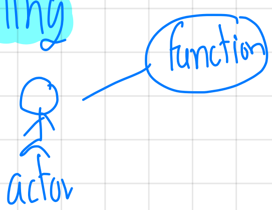
- developing abstract model of a system.
  - help carify what the system does , เวลาทำก็ไม่ต้องตามกฎของแต่ละ model ทั้งหมด
- Context models
  - at the early stage in specification → you should decide on the system boundaries
    - the functionality
  - Context models normally show the environement include several other ___ system
    - not show the type of relationship



UML มันเหมือนใน BPMN เลย

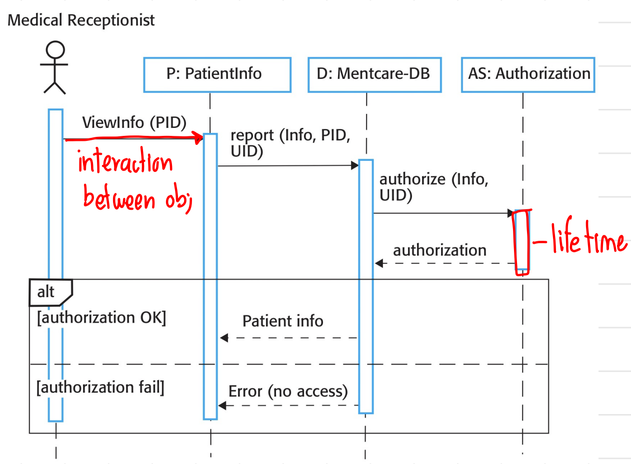


- Interaction model
  - Use case modeling



; who involved

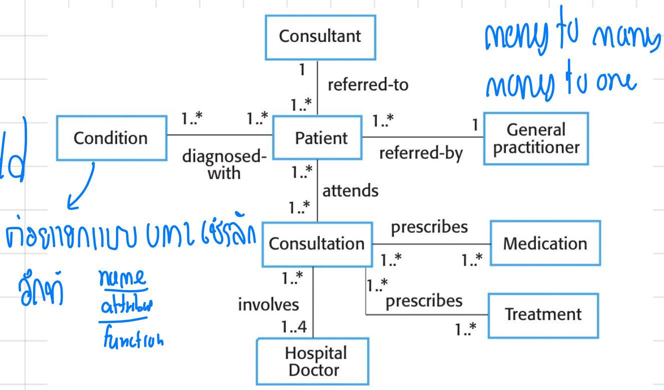
- Sequence diagram
  - actor and object and object and object



↳ Structure model

↳ Class diagram

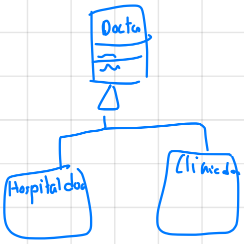
↳ object represent something in real world



many to many
many to one

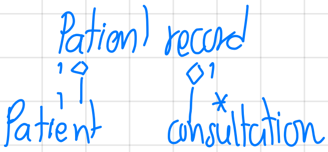
↳ Generalization

↳ หลักการของ OOP ตามที่เคยเจอ



↳ Aggregation

↳ ข้อมูลจากหลาย database รวม



; object 1 อย่างประกอบขึ้นมาจากหลายส่วน

↳ Behavior model

↳ what happen or suppose to happen when a system responds to a stimulus from its env.

↳ Data-driven modeling

↳ involved in processing input data and generating an associated output.

↳ tracking how data associated with process

↳ DFD *

↳ Event-driven modeling

↳ how the system system responds to external and internal event

↳ key point

↳ a model is an abstraction view of system, ignore some system details

↳ to show the system's context, interact, structure, behavior

↳ Context model show how the system that is being modeled is positioned in an environment with other system and process → help define the boundaries of the system

↳ Use case diagram and sequence diagram are used to describe the interaction between user and system

add more information, showing interactions between sys obj

between user and system

↳ Structural models show the organization and architecture of a system. Class diagram are used to define the static structured of classed in a system and their association

↳ Behavior model are used to describe the dynamic behavior of an executing system

↳ may be perspective of the data processed by the system or by the events stimulate response from the system

↳ Activity diagram may be used to model the processing of data, each act each proces,

↳ state diagram used to model the system behavior in response to internal or external even

↳ Model-driven engineering, is an approach to software development in which a system is represented as a set of model that can be automatically tranformed to executable code