Final Review

อ. ปรัชญ์ ปิยะวงศ์วิศาล

Pratch Piyawongwisal

Final Exam Review

- Topics
  - Scrum
  - Trello (Scrum Board)
  - Requirement Engineering
  - System Modeling
  - Software Design
  - Code Craftmanship
  - Lab: Ionic

Scrum - Summary

- เป็นกระบวนการทาง **Agile**
- ทำงานเป็น **sprint** ๆ ละ **1-4** สัปดาห์
- ทีมประกอบด้วย **3** หน้าที่หลักๆ คือ
  - **Product Owner** — จัดทำ **requirement (user stories)** ตามความต้องการของลูกค้า (stakeholder)
  - **Scrum Master** — ควบคุม ติดตาม ให้กระบวนการ **Scrum** ดำเนินไปด้วยความเรียบร้อย
  - **Dev Team** — ทีมทำงาน เช่น **Engineers/Designers/QA, etc.**

Scrum - Summary

- สิ่งที่ต้องจัดทำขึ้น (artifact)

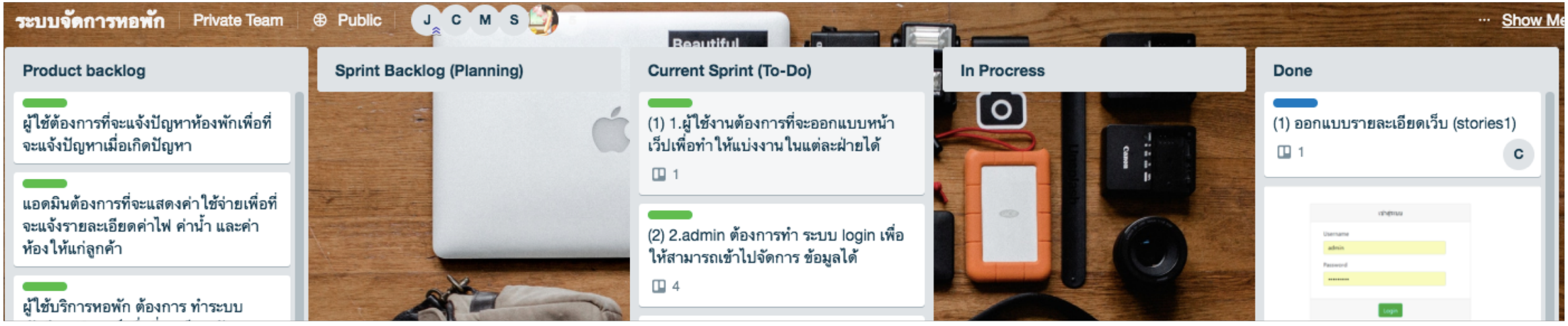
  Scrum Board (อาจใช้กระดาน หรือ Trello)
  - product backlog — รายการ user stories แบบกว้าง ๆ
  - sprint backlog — รายการ task ที่ต้องทำใน sprint
  - burndown chart — กราฟแสดงความเร็วในการทำ task เสร็จใน sprint
- กิจกรรมหลัก
  - sprint planning — คัดเลือก user stories มาแตกเป็น task เกิดเป็น sprint backlog
  - sprint — ทำงานทุก task ใน sprint backlog ให้เสร็จ ให้พอมีอะไรนำส่งลูกค้า
  - daily scrum — ทุกวันต้องพูดคุยถึงสิ่งที่ทำไป, สิ่งที่จะทำ, ปัญหาอุปสรรค
  - sprint review — ส่งงานให้ลูกค้า เก็บ feedback
  - sprint retrospective — ทบทวนกระบวนการ ปรับปรุง backlog ตาม feedback ที่ได้มา

Scrum Board on Trello

- มีหลัก ๆ 5 คอลัมน์ ได้แก่
  - Product Backlog — รายการ user story ทั้งหมด ที่ product owner สร้างขึ้น
  - Sprint Backlog — user story ที่ถูกเลือกมาทำใน sprint ปัจจุบัน
  - To-Do — task ย่อยที่จะต้องทำ อาจมีค่า story point

  } ทำในช่วง Sprint Planning

  - In Progress — task ที่กำลังทำ (กำหนดผู้รับผิดชอบ)
  - Done — task ที่เสร็จแล้ว



Scrum Quiz

1. วิธี Scrum เป็นกระบวนการในการพัฒนาซอฟต์แวร์แบบใด
2. วิธี Scrum จะมีการแบ่งหน้าที่ออกเป็น 3 หน้าที่หลัก ๆ ได้แก่อะไรบ้าง
3. จงยกตัวอย่าง artifact ของวิธี Scrum มา 2 อย่าง
4. กิจกรรมแรกสุดที่ต้องทำก่อนเริ่ม sprint เรียกว่าอะไร ต้องทำอะไร
5. เมื่อเริ่ม sprint แล้ว ทุก ๆ วันก่อนเริ่มงานจะต้องทำอะไร
6. ในกิจกรรมข้อ 5 ผู้เข้าร่วมจะต้องตอบคำถาม 3 ข้อหลักๆ ได้แก่อะไรบ้าง
7. หลังจบ sprint แล้ว จะต้องมีการประชุม 2 ครั้ง เรียกว่าอะไร
8. ใน scrum ใครมีหน้าที่คิดรายการ user story ใน product backlog
9. ใน scrum ใครมีหน้าที่แนะนำและชี้แจงกิจกรรม meeting ต่างๆ และเป็นผู้นำให้เกิดการพูดคุยแลกเปลี่ยนใน meeting เหล่านั้น

Example Scrum Worksheet

www.glurgeek.com/education/software-engineering-course/

ชื่อ Senior Project (ภาษาไทย) .....

ชื่อ Senior Project (ภาษาอังกฤษ) .....

ขั้นตอนที่ 1 Product Backlog

เขียน Requirements ของระบบอย่างน้อย 4 ความต้องการ ลงในตาราง

ID	Requirements	User Stories
1		As a I can So that
2		As a I can So that
3		As a I can So that
4		As a I can So that

ขั้นตอนที่ 2 Sprint Planning

ให้กำหนด Priority และ Story Point ลงในตาราง (ID คือหมายเลขเดียวกับขั้นตอนที่ 1)

ID	Priority	Story Point

ขั้นตอนที่ 3 Sprint Backlog

เขียนการจัดลำดับของ Sprint ที่ได้จากขั้นตอน Sprint Planning ลงในตาราง

(Sprint ที่ n คือ รอบของการทำ Sprint และ ID คือหมายเลขเดียวกับขั้นตอนที่ 1)

Sprint ที่ n	ID	Task To Do	วันส่งมอบงาน

ขั้นตอนที่ 4 Daily Scrum

ให้อธิบายการทำงานของขั้นตอนนี้ พร้อมทั้งยกตัวอย่างสถานการณ์จำลองของ Project ที่ทำ

.....

.....

ขั้นตอนที่ 5 Sprint Review and Retrospective

ให้อธิบายการทำงานของขั้นตอนนี้ พร้อมทั้งยกตัวอย่างสถานการณ์จำลองของ Project ที่ทำ

.....

.....

ขั้นตอนที่ 6 Estimation (10 คะแนน)

เขียน Burndown Chart เพื่อประเมินการทำงานทั้งกระบวนการในการพัฒนาซอฟต์แวร์ที่วางแผนไว้ตั้งแต่วันเริ่มต้นทำจนถึงวันที่ส่งมอบงานครบ ซึ่งกราฟที่ได้จะต้องมีความสอดคล้องกับข้อมูลในขั้นตอนที่ 2 และ 3

(กำหนดให้ แกน x คือ เวลา และ แกน y คือ จำนวน Story Points ของ Release ที่ยังทำไม่เสร็จ ณ เวลาใดๆ)

Example Scrum Worksheet

www.glurgeek.com/education/software-engineering-course/

ชื่อ Senior Project (ภาษาอังกฤษ) A Support Tool for Teaching

ขั้นตอนที่ 1 Product Backlog

เขียน Requirements ของระบบอย่างน้อย 4 ความต้องการ ลงในตาราง

ID	Requirements	User Stories
1	ซอฟต์แวร์มีการ Check ชื่อนักศึกษาที่เข้าห้องเรียน นักศึกษาที่ส่งการบ้าน ผ่าน Mobile Device	**As a** ผู้ใช้ **I can** ตรวจสอบชื่อ การส่งการบ้านได้ **So that** ให้ทราบสถานะของนักศึกษา
2	มีการระบบการ Chat หรือการส่ง Messages หรือ ไฟล์งาน ระหว่างกัน ผ่าน Mobile Device	**As a** ผู้ใช้ **I can** ส่งข้อความสื่อสารระหว่างกัน **So that** ส่งงานหรือตอบคำถามระหว่างเรียนได้
3	ในแต่ละบทเรียนสามารถทำ Quiz ผ่านและประมวลผลคะแนนผ่าน Mobile Device	**As a** ผู้ใช้ **I can** ทำ Quiz และดูคะแนน **So that** สะดวกในการทำและตรวจสอบคะแนน
4	สร้างเกมหรือกิจกรรม Workshop ให้ผู้เรียนมีส่วนร่วมผ่าน Mobile Device	**As a** ผู้ใช้ **I can** เล่นเกมหรือทำกิจกรรมระหว่างเรียน **So that** เพิ่มความสนใจในการเรียนของนักศึกษา

Example Scrum Worksheet

www.glurgeek.com/education/software-engineering-course/

ขั้นตอนที่ 2 Sprint Planning

ให้กำหนด Priority และ Story Point ลงในตาราง (ID คือหมายเลขเดียวกับขั้นตอนที่ 1)

ID	Priority	Story Point
1	1	20
2	3	40
3	2	40
4	4	80
5	5	30

Example Scrum Worksheet

www.glurgeek.com/education/software-engineering-course/

ขั้นตอนที่ 3 Sprint Backlog

เขียนการจัดลำดับของ Sprint ที่ได้จากขั้นตอน Sprint Planning ลงในตาราง

(Sprint ที่ n คือ รอบของการทำ Sprint และ ID คือหมายเลขเดียวกับขั้นตอนที่ 1)

Sprint ที่ n	ID	Task To Do	วันส่งมอบงาน
1	1,5	ทำระบบ login, กำหนด Security Policy, ทำ DB, ทำระบบให้ Security และตรวจสอบความถูกต้อง	3 Weeks
2	2	ทำระบบส่งข้อความ ส่งรูป ส่งเสียง	4 Weeks
3	3	ทำแบบทดสอบในแต่ละการเรียน	7 Weeks
4	4	ทำเกมและกิจกรรมให้เข้ากับบทเรียน	10 Weeks

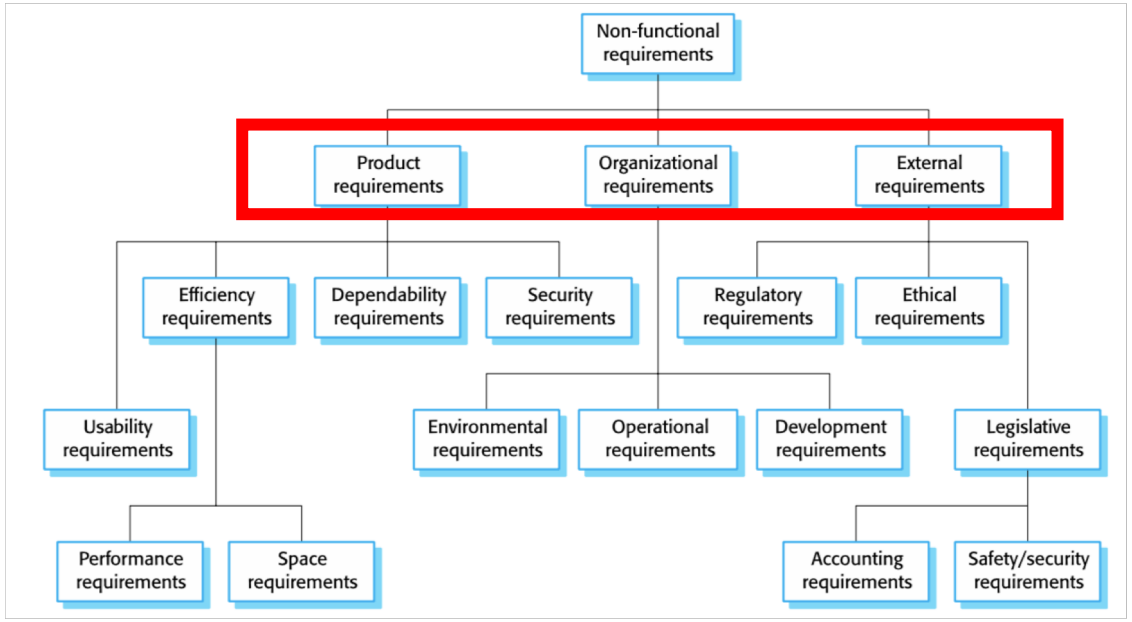
Requirement Engineering

- กระบวนการระบุข้อกำหนด (requirement) คือคำอธิบายเกี่ยวกับ service ต่าง ๆ ที่ผู้ใช้ต้องการจากระบบ รวมถึงขีดจำกัด (constraint) ต่าง ๆ ของระบบ
- ประเภทของ requirement โดยแบ่งตาม target audience
  - User requirements — สำหรับลูกค้า, คร่าว ๆ เน้นเข้าใจง่าย, ใช้ภาษาพูด + แผนภาพ
  - System requirements — สำหรับ dev, ลงรายละเอียด, ใช้ภาษาทางเทคนิค
- ประเภทของ requirement โดยแบ่งตามสิ่งที่ระบุ
  - Functional requirements — ฟังก์ชันการทำงานของระบบ เช่น user ต้องสามารถค้นหาหนังสือได้
  - Non-Functional requirements — ข้อจำกัด (constraint) ต่าง ๆ เช่น response time
  - โดยส่วนมาก Non-Functional จะสำคัญกว่า Functional

Non-functional Requirement

- มี 3 ประเภทย่อยดังแผนภาพ
- เขียน **non-functional req.** ไม่ให้กำกวมโดยใช้คำศัพท์เชิงปริมาณที่ตรวจวัดได้จริง เช่น
  - กำกวม: แอพใช้งานง่าย
  - ชัดเจน: จนท.สามารถใช้ทุกฟังก์ชันเป็นหลังผ่านการอบรมเพียง 4 ชั่วโมง
    หากจนท.ที่ผ่านการอบรมแล้วใช้งานระบบ จะไม่เกิด error ขึ้นเกิน 2 ครั้งต่อชั่วโมงโดยเฉลี่ย

Property	Measure
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



Requirement Engineering Process

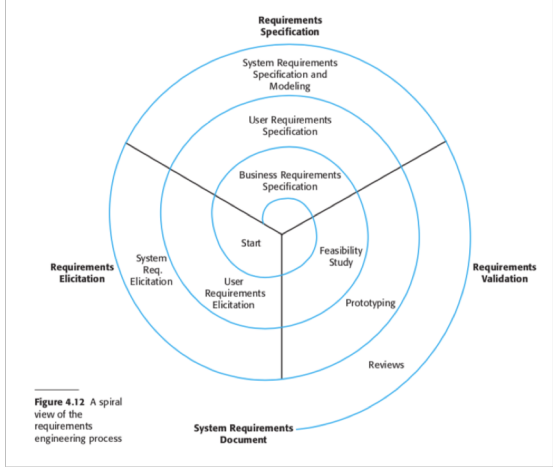


Figure 4.12 A spiral view of the requirements engineering process

- **Requirement elicitation**
  - ทำการ interview สอบถามความต้องการจาก stakeholder โดยใช้ user stories, use case เป็นเครื่องมือ
- **Requirement specification**
  - วิเคราะห์แล้วลงมือเขียนทั้ง User requirement และ System requirement
- **Requirement validation**
  - ตรวจสอบว่า requirement ตรงความต้องการลูกค้าจริง, ไม่มี conflict ใด ๆ ระหว่างแต่ละ requirement, มีความเป็นไปได้, สามารถทดสอบได้
- **Requirement management**
  - ติดตาม requirement ที่ทำสำเร็จ, การเปลี่ยนแปลงของ requirement

ลองคิดดูว่าใน **scrum** เราได้ทำกระบวนการเหล่านี้อย่างไร โดยใช้ **artifact** ใดบ้าง

Requirement Engineering Quiz

1. กระบวนการ Requirement Engineering มีความสำคัญต่อผู้ที่มีบทบาทใดในวิธีการ Scrum มากที่สุด
2. การเขียน Requirement ในรูป "As a… I want… so that…" ถือว่าเป็น Requirement แบบใด User Requirement หรือ System Requirement
3. ข้อจำกัดด้าน response time ของเซิฟเวอร์ ถือว่าเป็น requirement แบบ functional หรือ non-functional
4. การเก็บ requirement ด้วยวิธีสัมภาษณ์ ถือว่าอยู่ในขั้นตอนใด (elicitation/specification/validation/management)
5. ในวิธีการ scrum การนำ feedback ของลูกค้ามาปรับแต่ง Product Backlog ในกิจกรรม Retrospective Meeting ถือว่าเป็นขั้นตอนใด (elicitation/specification/validation/management)

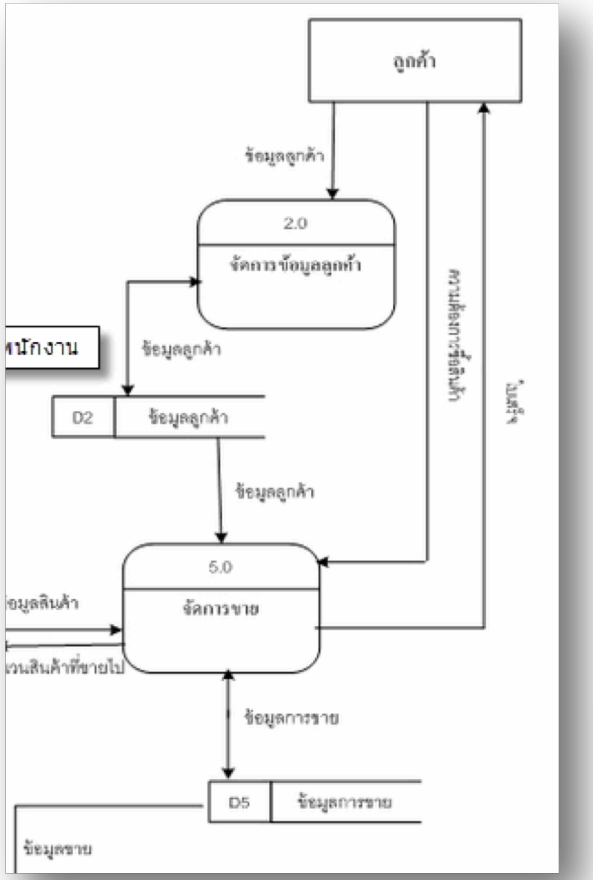
System Modeling

- การสร้างแบบจำลองเพื่ออธิบายฟังก์ชันการทำงาน (functionality) หรือองค์ประกอบ (component) ต่าง ๆ ของระบบ
- ทำในช่วง design phase
- มักใช้แผนภาพต่าง ๆ เป็นเครื่องมือช่วยในการออกแบบซอฟต์แวร์
- แบบจำลองสามารถอธิบายระบบได้จาก 4 ด้านหลักๆ ดังนี้
  - External จำลองบริบท (context) หรือสิ่งที่อยู่นอกระบบ
  - Interaction จำลองการปฏิสัมพันธ์ระหว่างส่วนต่างๆ ในระบบ
  - Structural จำลองโครงสร้างของข้อมูลในระบบ
  - Behavioral จำลองพฤติกรรมการทำงานของระบบ

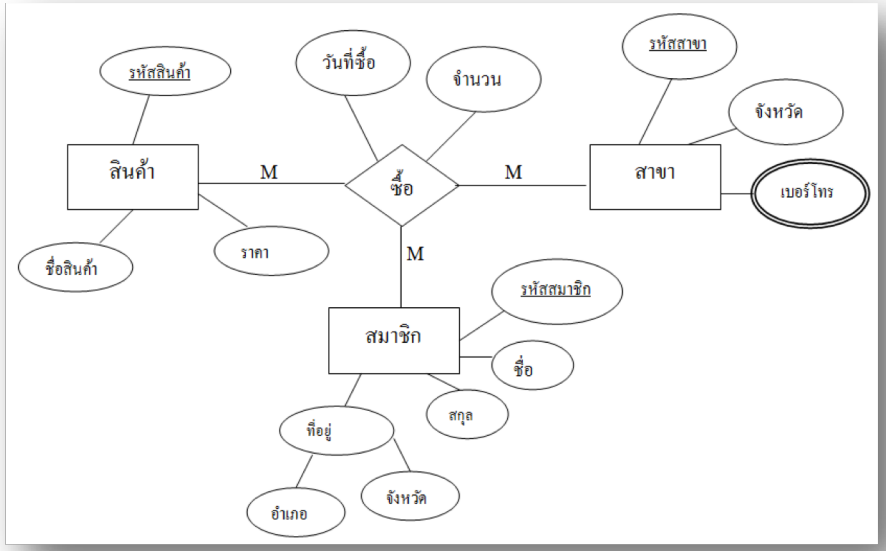
System Modeling

- แผนภาพที่ควรรู้จัก
  - Data Flow Diagram — กระแสข้อมูลระหว่าง process ต่าง ๆ
  - ER Diagram — ฐานข้อมูล (ชื่อตาราง, ชื่อคอลัมน์)
  - UML Class Diagram — คลาสต่าง ๆ ในโปรแกรมเชิงวัตถุ
  - UML Use Case Diagram — requirement ของลูกค้า
  - UML Sequence Diagram — ลำดับการทำงานของระบบ

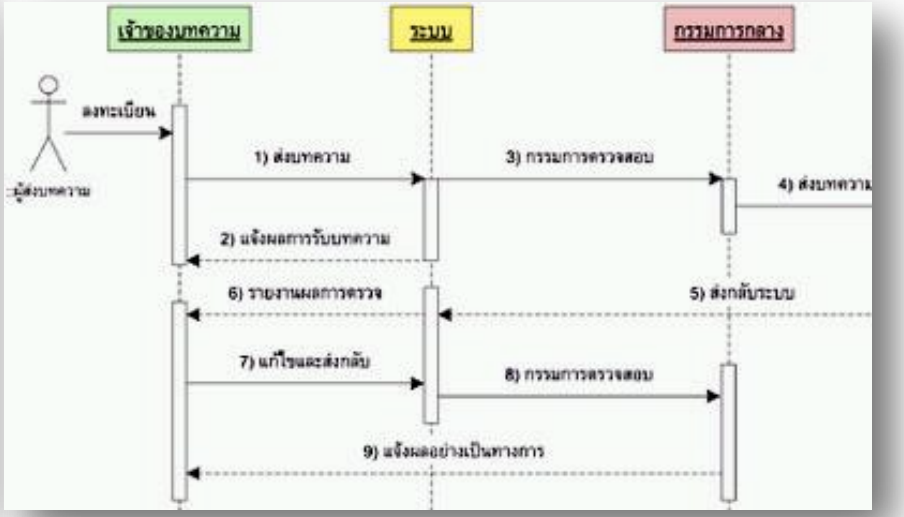
Data Flow Diagram



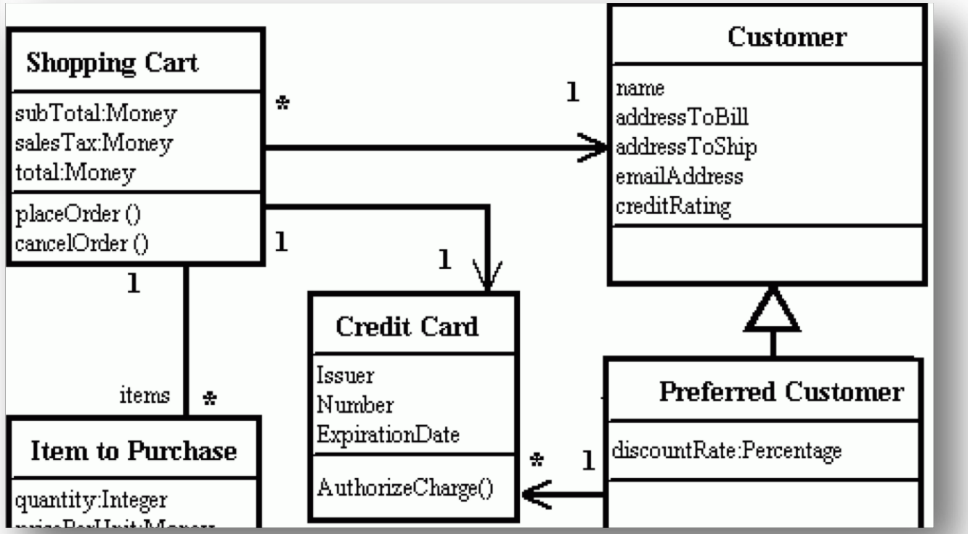
ER Diagram



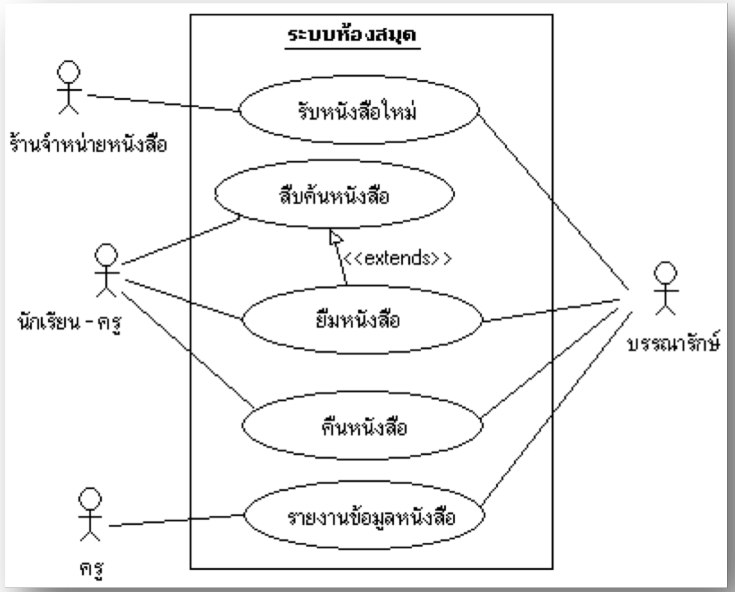
UML Sequence Diagram



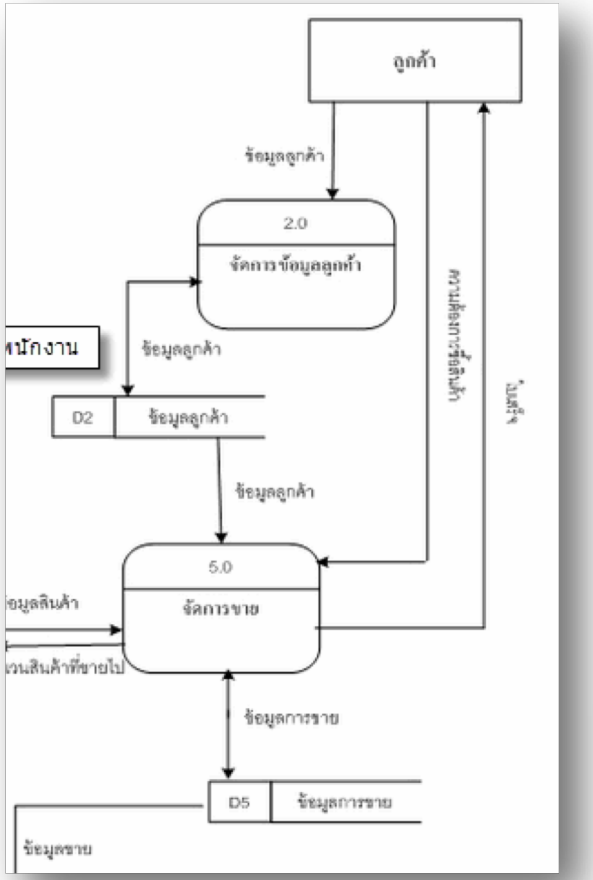
UML Class Diagram



UML Use Case Diagram



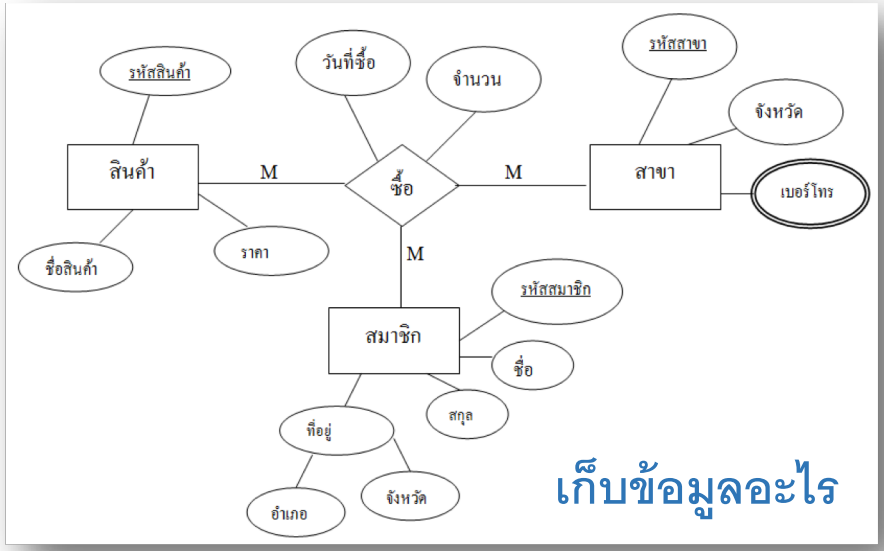
Data Flow Diagram



ข้อมูลไหลไป

process ต่าง ๆ อย่างไร

ER Diagram

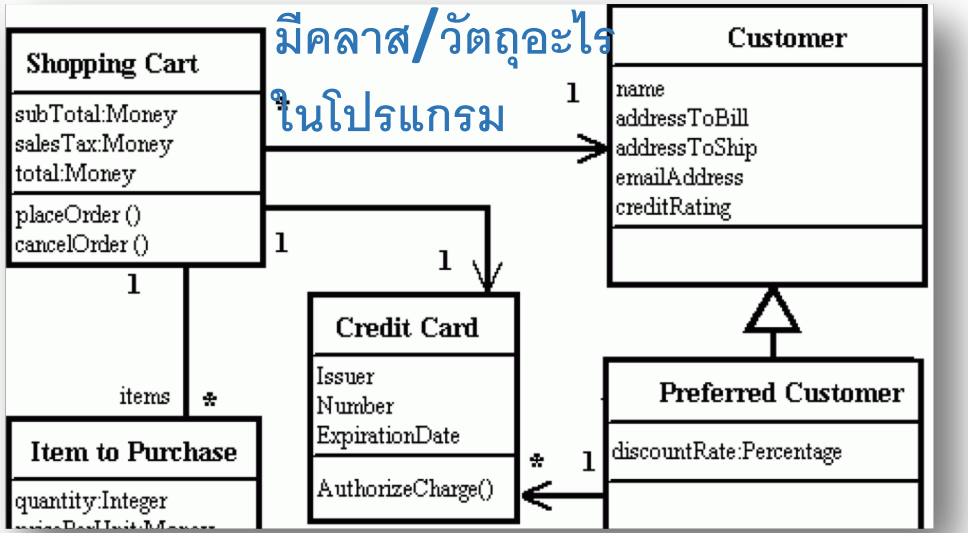


เก็บข้อมูลอะไร

UML Sequence Diagram

มีลำดับการทำงานอย่างไร

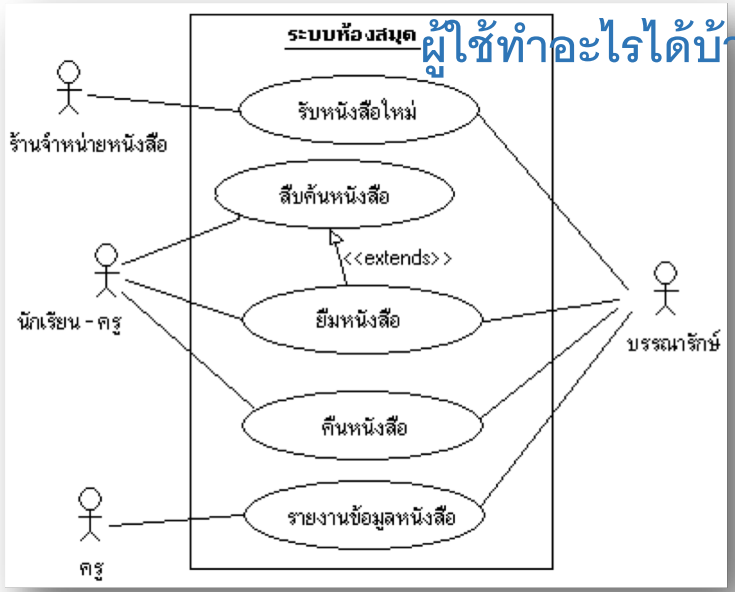
UML Class Diagram



มีคลาส/วัตถุอะไร

ในโปรแกรม

UML Use Case Diagram



ผู้ใช้ทำอะไรได้บ้าง

Software Design

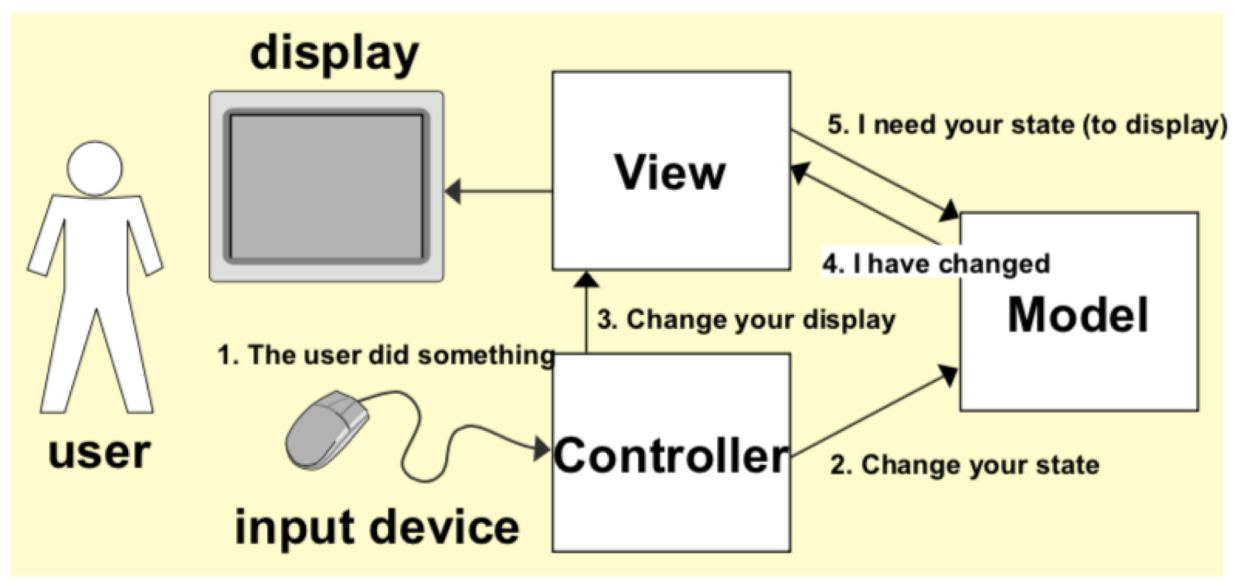
- เป็นกระบวนการในการนิยาม architecture, components, interfaces ของระบบ
- ประกอบด้วย 2 ขั้นตอนหลักๆ ได้แก่
  - Software Architectural Design
    เป็นการออกแบบโครงสร้างของระบบในระดับ top-level โดยจะตอบคำถามที่ว่า เราควรแตกระบบออกเป็น component ย่อยๆ ได้อย่างไร?
  - Software Detailed Design
    เป็นการออกแบบรายละเอียดภายในแต่ละ component ให้เพียงพอต่อการนำไปสร้างขึ้นมาได้จริง ==> design patterns

Software Architecture

- A software architecture is “a description of the **subsystems and components** of a software system and the **relationship** between them”
- ตัวอย่าง Software architecture ที่ควรรู้จัก
  - Pipe-and-Filter
  - Client-Server
  - Model-View-Controller

Model-View-Controller (MVC)

- ยึดหลัก Separation of Concerns แบ่งภาระหน้าที่ให้แต่ละส่วนทำงานไปโดยอิสระจากกันได้
- แบ่งระบบออกเป็น 3 ส่วน
  - Model ส่วนเก็บและจัดการข้อมูล เช่น โค้ดที่ใช้ติดต่อกับ Firebase, MariaDB
  - View ส่วนจัดวางรูปแบบการแสดงผล เช่น ไฟล์ .page.html ใน ionic
  - Controller ส่วนติดต่อกับผู้ใช้และ logic ควบคุม Model, View เช่น ไฟล์ .page.ts ใน Ionic
- นิยมใช้ในการเขียน Web/Mobile App
- ตัวอย่าง framework แบบ MVC
  - Ionic, Angular, Django, Ruby on Rails



Design Pattern

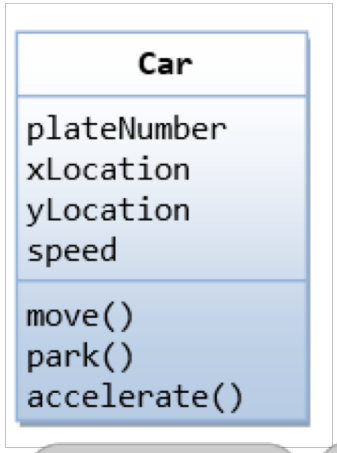
- Design Pattern is a “common design solution to a common problem in a given context”
- คือ รูปแบบการเขียนโปรแกรมที่พบเจอบ่อยๆ อาจเหมาะกับบางปัญหาตามแต่บริบท
- ตัวอย่าง Design Pattern ที่ควรรู้จัก
  - Singleton
  - Factory
  - Adapter
  - Observer

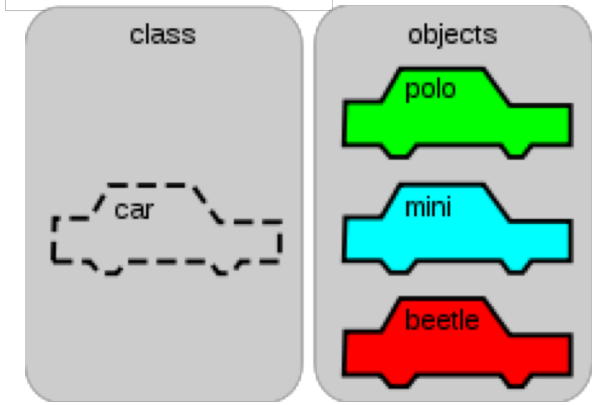
Singleton: บังคับให้แต่ละคลาสสามารถมีวัตถุได้แค่ตัวเดียวเป็นแบบ global/static

} เป็น pattern ที่เจอบ่อย แต่ต้องเรียน OOP มาก่อนจึงจะเข้าใจ
ให้จำชื่อให้ได้พอ รายละเอียดการทำงานไม่ออกสอบ

Object-Oriented Programming (OOP)

- เป็นวิธีการเขียนโปรแกรมโดยใช้ **วัตถุ** (**object**) และ **คลาส** (**class**) ในการแก้ปัญหา
- เช่น จะเขียนโปรแกรมจองเที่ยวบิน เราจะเริ่มจากคิดว่าโปรแกรมควรมีวัตถุคลาสอะไรบ้าง
  - คลาส Flight เก็บข้อมูลเที่ยวบิน
  - คลาส Passenger วัตถุที่แทนตัวผู้โดยสาร
  - คลาส Destination ข้อมูลสนามบินปลายทาง
- คลาสประกอบด้วย 2 ส่วน คือ
  - Property คือ ส่วนเก็บข้อมูล
    - เช่น Passenger ต้องมี ID, Name, Address, RewardPoints, etc.
  - Method คือ ส่วนการกระทำ
    - เช่น Passenger ต้องสามารถ BookFlight(), ChangeProfile(),.. ได้





Singleton

https://medium.com/20scoops-cnx/singleton-pattern-คืออะไร-b7b28182654f

```java
public class Singleton {
    private static Singleton instance;

    private Singleton() {
    }

    public static Singleton getInstance() {
        if (instance == null) {
            instance = new Singleton();
        }
        return instance;
    }
}
```

```java
public static void main(String[] args) {
        //init value
        UserData userData = UserData.getInstance();
        userData.setFistName("20Scoops");
        userData.setLastName("CNX");
```

ข้อดีคือ ป้องกันไม่ให้เกิดวัตถุจำนวนมากที่ทำงานซ้ำซ้อนกัน

ข้อเสียคือ เป็น global/static ถ้าเก็บข้อมูลใหญ่จะหน่วง

Code Craftsmanship

- วิธีเขียนโค้ดให้มีคุณภาพ
  - อ่านง่าย
  - ทดสอบง่าย
  - นำไปใช้และดูแลรักษาง่าย
- **Refactoring** คือการปรับโครงสร้างโค้ดให้เป็นระเบียบขึ้น โดยไม่เปลี่ยนแปลงพฤติกรรมการทำงานของโปรแกรม

```
void printOwing() {
  printBanner();
  //print details
  System.out.println ("name: " + _name);
  System.out.println ("amount      " + getOutstanding());
  }
```

Refactored to

```
void printOwing() {
  printBanner();
  printDetails(getOutstanding());
}
void printDetails (double outstanding) {
  System.out.println ("name: " + _name);
  System.out.println ("amount      " + outstanding);
}
```

Figure 1: Example of Refactoring

Basic rules

- ใช้ coding style/convention ตามโค้ดที่มีอยู่เดิม
- เขียนโค้ดให้รวบรัด แต่ก็ยังคงความอ่านง่าย
- ตั้งชื่อตัวแปรให้บ่งบอก ไม่ใช้ชื่อแปลกๆ
- พยายามไม่ให้เกิดโค้ดที่ซ้ำซ้อน
- ใส่ comment อธิบายโค้ด
  - หากนำโค้ดมาจาก stackoverflow ใส่ url ที่มาด้วย
- พยายามไม่ comment code ทิ้ง เพราะรก

Basic rules

- ใช้ coding style/convention ตามโค้ดที่มีอยู่เดิม

```
for(int i=0; i<numProduct; i++)
{
   if(isValid)
       printf("Valid");
}
```

Style แบบที่ 1

```
for(int i = 0; i < num_product; i++) {
   if(is_valid == true) {
       printf("Valid");
   }
}
```

Style แบบที่ 2

Basic rules

- เขียนโค้ดให้รวบรัด แต่ก็ยังคงความอ่านง่าย

```
if (condition == true){
 // Do something ...
}
```

```
if (condition){
 // Do something ...
}
```

รวบรัดกว่า

Basic rules

- เขียนโค้ดให้รวบรัด แต่ก็ยังคงความอ่านง่าย

ชื่อตัวแปรไม่บ่งบอก

```
public Iterator<ITweet> iterator(InputStream data, Set<String> hashTags) throws IOException {
        Iterator<ITweet> streamiter = iterator(data);
        Predicate<ITweet> hashpred = h -> h.getHashTags().containsAll(hashTags);
        TweetFilteringIterator<ITweet> filteriter = new TweetFilteringIterator<ITweet>(streamiter, hashpred);
        return filteriter;
}
```

ไม่จำเป็น

VS

```
public Iterator<ITweet> iterator(InputStream data, Set<String> hashTags) throws IOException {
        Iterator<ITweet> tweets = iterator(data);
        Predicate<ITweet> filter = tweet -> tweet.getHashTags().containsAll(hashTags);
        return new TweetFilteringIterator<ITweet>(tweets, filter);
}
```

กำลังดี

VS

```
public Iterator<ITweet> iterator(InputStream data, Set<String> hashTags) throws IOException {
        return new TweetFilteringIterator<ITweet>(iterator(data),
                    tweet -> tweet.getHashTags().containsAll(hashTags));
}
```

อาจจะรวบรัดไป อ่านยาก

Peer Review

- Code review
  - คือการให้เพื่อนร่วมงานช่วยตรวจทานโค้ดก่อนการ merge
  - นิยมทำกันทั่วไปในบริษัทซอฟต์แวร์
  - โปรแกรมเมอร์อาวุโสช่วยดูให้โปรแกรมเมอร์มือใหม่
  - โค้ดผ่านหลายตา ย่อมเจอจุดผิดได้มากขึ้น
  - ให้ feedback ที่เป็นประโยชน์ ใช้คำวิจารณ์ที่ตรงแต่ถนอมน้ำใจ
- เป็นตามค่านิยมแบบ Agile คือส่งเสริมให้เกิดการพูดคุยแลกเปลี่ยน