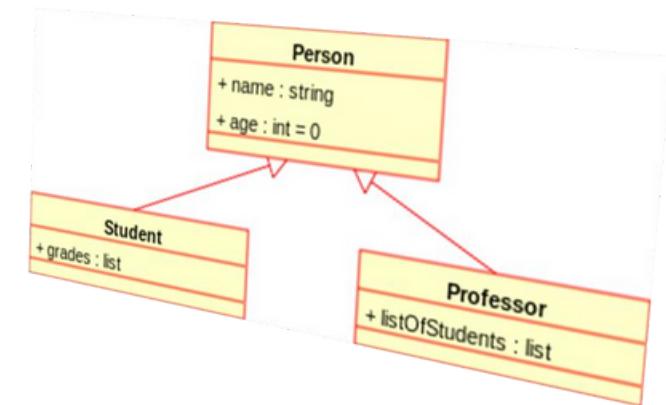


บทที่ 6

แบบจำลองระบบ (System Model)

วิชา วิศวกรรมซอฟต์แวร์ (04-06-322)



วัตถุประสงค์การเรียนรู้

-
- เพื่อให้ผู้เรียนมีความรู้ความเข้าใจเกี่ยวกับแบบจำลองระบบ (System Design) ตลอดจนเข้าใจสัญลักษณ์ และวัตถุประสงค์ของแบบจำลองระบบที่จะนำไปใช้ได้
 - เพื่อให้ผู้เรียนสามารถวิเคราะห์และออกแบบระบบด้วยแบบจำลองระบบ และสามารถนำไปประยุกต์ใช้งานร่วมกับแนวทำงกำรพัฒนาซอฟต์แวร์ที่เลือกได้อีกเช่นเดียวกัน



หัวข้อ (Agenda)

บทนำ (Overview)

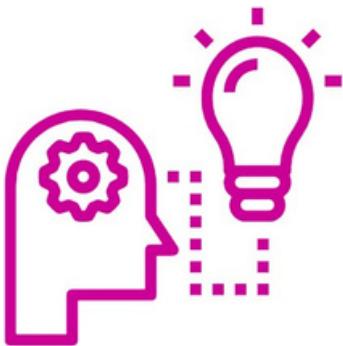
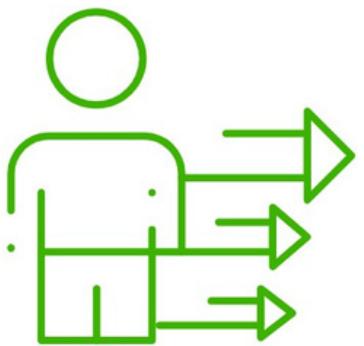
แบบจำลองระบบ (System Model)

ความสำคัญของแบบจำลอง

ประเภทแบบจำลองจรา้งแบบดำเนินการพื้นฐาน

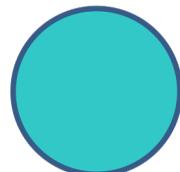
- - Structured System Approach
 - Object-oriented Approach
- สรุป (Summary)

unu የ (Overview)

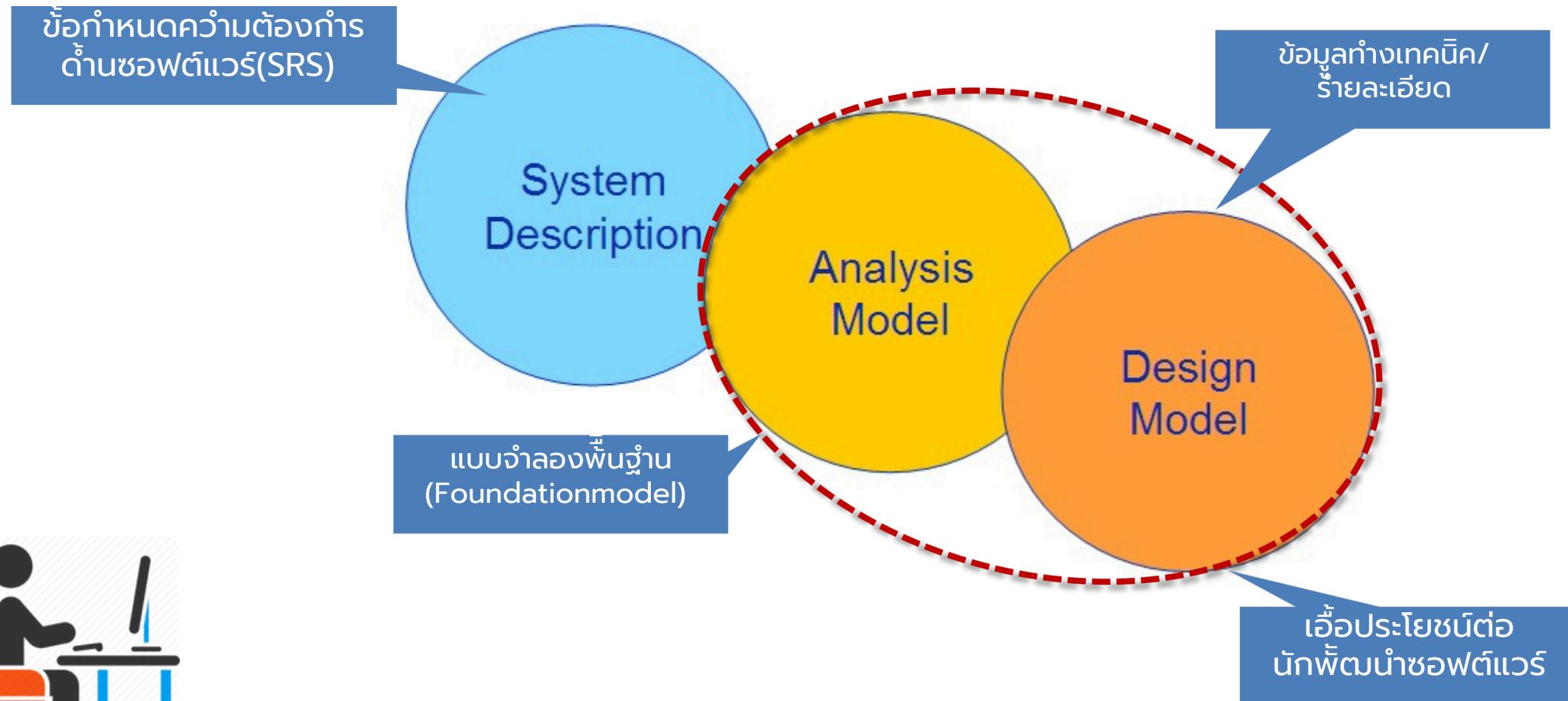


แบบจำลองระบบ (System Model)

- แบบจำลอง (Model)
 - สัญลักษณ์ที่ใช้จำลองข้อเท็จจริงต่างๆ ที่เกิดขึ้นในระบบ
 - แสดงให้เห็นในแต่ละมุมมอง
- แบบจำลองการวิเคราะห์ (Analysis Model)
 - แบบจำลองที่เขียนขึ้นจากข้อมูลกำหนดความต้องการของระบบ
 - หน้าที่การทำงานของระบบด้านต่างๆ
 - ระบบทำหน้าที่WhatและHow
- แบบจำลองระบบ (System Model)
 - แบบจำลองแนวคิด เพื่อจำลองความต้องการ
 - นักพัฒนาซอฟต์แวร์ นักวิเคราะห์ระบบและนักออกแบบระบบ



ความสัมพันธ์ระหว่างแบบจำลองการวิเคราะห์และการออกแบบ



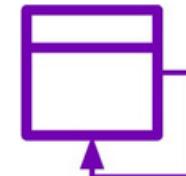
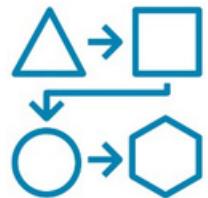
ความสำคัญของแบบจำลอง

- เพื่อให้กรับถึงความเป็นมาของระบบ และขั้นตอนในการปฏิบัติงานของระบบ
- เพื่อการออกแบบใหม่ที่ตรงตามความต้องการของผู้ใช้ ให้มากที่สุด
- แบบจำลองที่นำเสนอพิจารณาและวิเคราะห์ระบบ ดังนี้
 - Context Diagram
 - Data Flow Diagram (DFD)
 - E-R Diagram
 - System Flow Chart / Flow Chart
- ความผิดพลาดของโปรแกรมเมอร์ที่ออกแบบระบบ โดยไม่ผ่านการวิเคราะห์



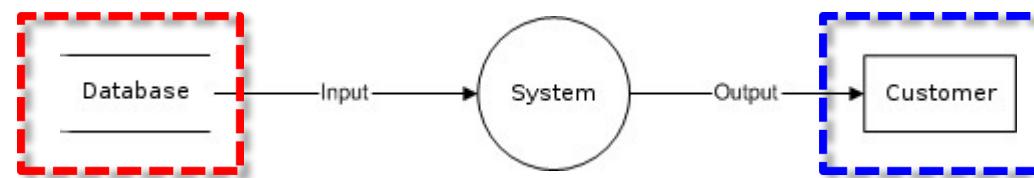
แบบจำลองระบบ (ต่อ)

- แบบจำลองตามแนวทำงเชิงโครงสร้าง (Structured Analysis)
 - แบบจำลองกระบวนการ (Process Model)
 - จำลองขั้นตอนการทำ งานของระบบ DFD
 - แบบจำลองข้อมูล (Data Model)
 - จำลองโครงสร้างข้อมูลกึ่งหมุดในระบบ E-R
- แบบจำลองตามแนวทำงวัตถุ (Object Oriented Analysis)
 - UML (Unified Modeling Language)



แผนภำพกระแสข้อมูล (Data Flow Diagram)

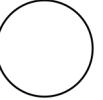
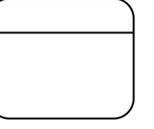
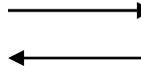
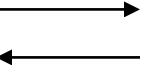
- แผนภำพที่แสดงถึงทิศทางการไหลของข้อมูลที่มีอยู่ในระบบ จำกัดกระบวนการดำเนินการหนึ่งไปอีกกระบวนการหนึ่ง หรือไปยังส่วนอื่นที่เกี่ยวข้อง
 - แหล่งจัดเก็บข้อมูล (Data Store)
 - ผู้ที่เกี่ยวข้องที่อยู่นอกระบบ (External Agent)



วัตถุประสงค์

- เป็นแผนภำพที่สรุปรูปแบบข้อมูลทั้งหมดที่ได้จำกัดสำหรับในลักษณะของรูปแบบที่เป็นโครงสร้าง
- ทрабатываที่นำข้อมูลที่ให้มาไปในรูปแบบต่างๆ (Data and Process)
- เป็นแผนภำพที่ใช้ในการพัฒนาต่อ แก้ไข ต่อนของระบบออกแบบ
- เป็นแผนภำพที่ใช้ในการอ้างอิง หรือเพื่อใช้ในการพัฒนาต่อในอนาคต
- เป็นข้อตกลงร่วมกันระหว่าง นักวิเคราะห์ ระบบและผู้ใช้งาน

สัญลักษณ์ที่ใช้ในDFD

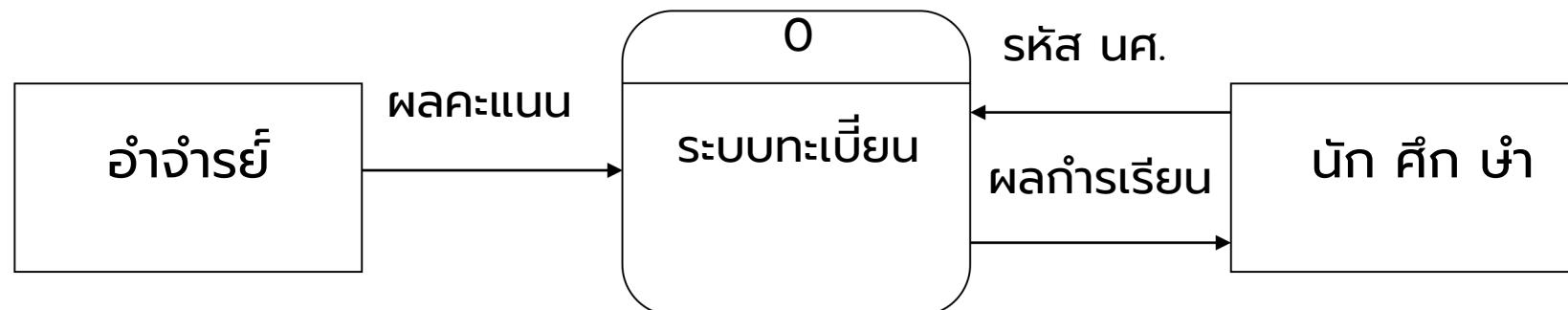
DeMarco & Yourdon	Gane & Sarson	ความหมาย
		<p>Process : ขั้นตอนการทำางานภายในระบบ</p>
		<p>Data Store : แหล่งข้อมูลสำหรับเป็นได้กับไฟล์ข้อมูลและฐานข้อมูล (File or Database)</p>
		<p>External Agent: ปัจจัย หรือ สภาพแวดล้อม ที่มีผลกระทบต่อระบบ</p>
		<p>Data Flow : เส้นทางการไหลของข้อมูล และทิศทางของข้อมูล จากขั้นตอนการทำางานหนึ่ง ไปยังอีกขั้นตอนหนึ่ง</p>

ระดับของ DFD

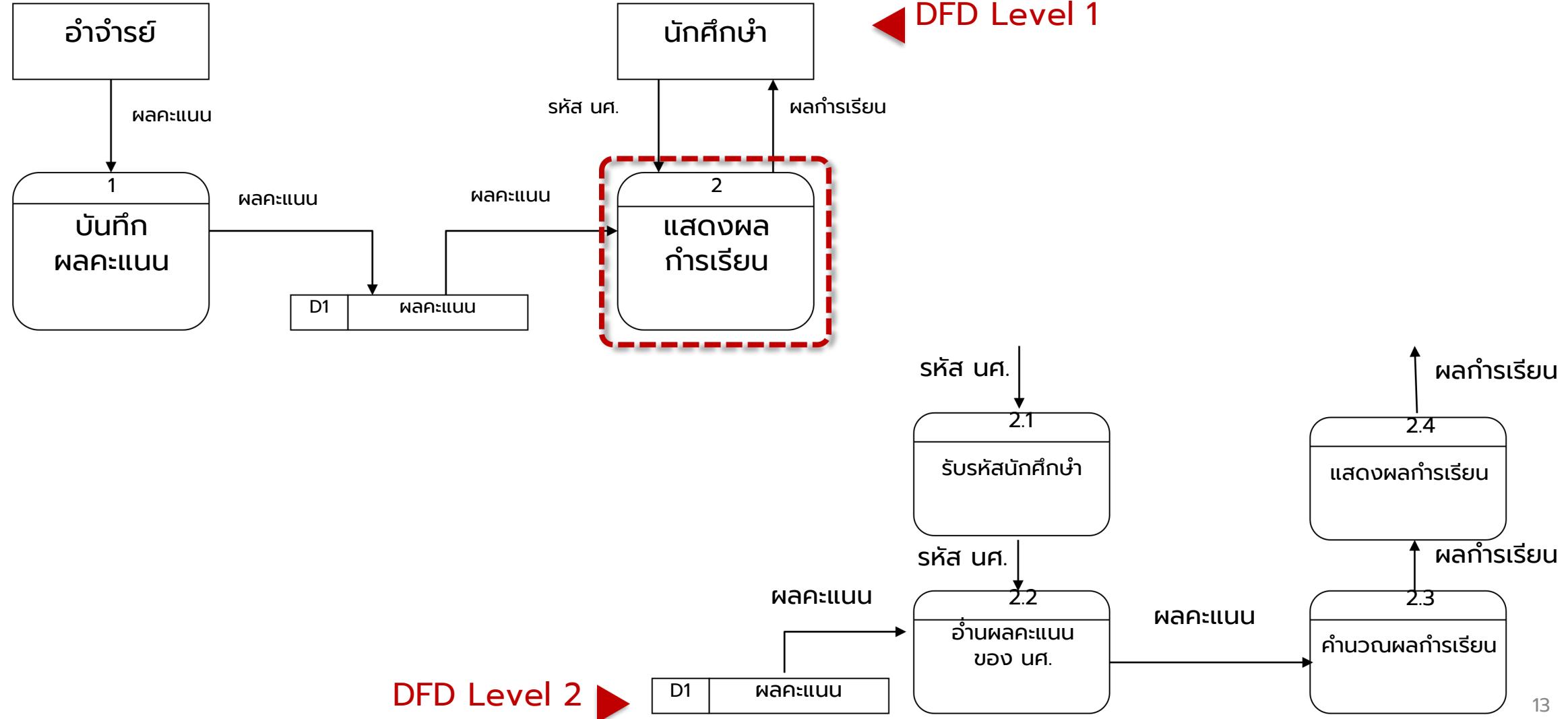
-
- กำรเขียน DFD สำมารถแยกรำยละเอียดของแพนก์พอกเป็นหลักระดับ เพื่อง่ายต่อการกำหนดขอบเขตในกำรพิจำรณำและง่ายต่อการกำค้ำมเข้าใจ
 - ระดับของ DFD ประกอบด้วยดังนี้
 - DFD Level 0 (Context Diagram)
 - DFD Level 1
 - DFD Level 2
 - DFD Level n

Context Diagram (DFD ระดับ 0)

- Context Diagram แสดงความสัมพันธ์ของระบบและสิ่งแวดล้อม โดยแสดงเฉพาะ ปฏิสัมพันธ์ระหว่าง External Entities (Actors) กับระบบเท่านั้น
 - ไม่เพียง Process เดียวคือ ระบบ
 - ไม่แสดง Data store
 - ขอบเขตของระบบพิจารณาได้จำกัดจำนวน Input/Output ก็คงหมด



ตัวอย่าง DFD ระดับ 1 และ ระดับ 2



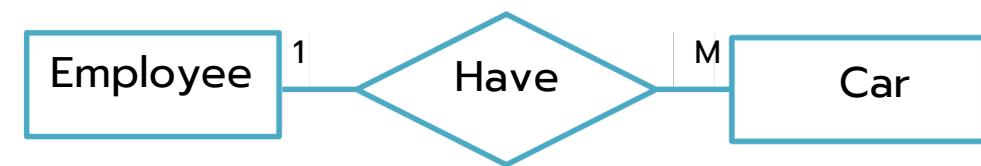
แผนกำ派แสดงความสัมพันธ์ระหว่าง Entity (Entity Relationship Diagram : ERD)

- เป็นเครื่องมือที่แสดงให้เห็นถึงความสัมพันธ์ของข้อมูลต่างๆ ที่มีต่อกันในระบบงาน
- แผนกำ派มี Cardinality เป็นสิ่งกำหนดค่าความสัมพันธ์ของ Entity ในความสัมพันธ์แต่ละลักษณะ อาทิเช่น 1:1 , 1:m และ m:n เป็นต้น ซึ่งอาจใช้สัญลักษณ์แทนได้

One-to-One Relationship



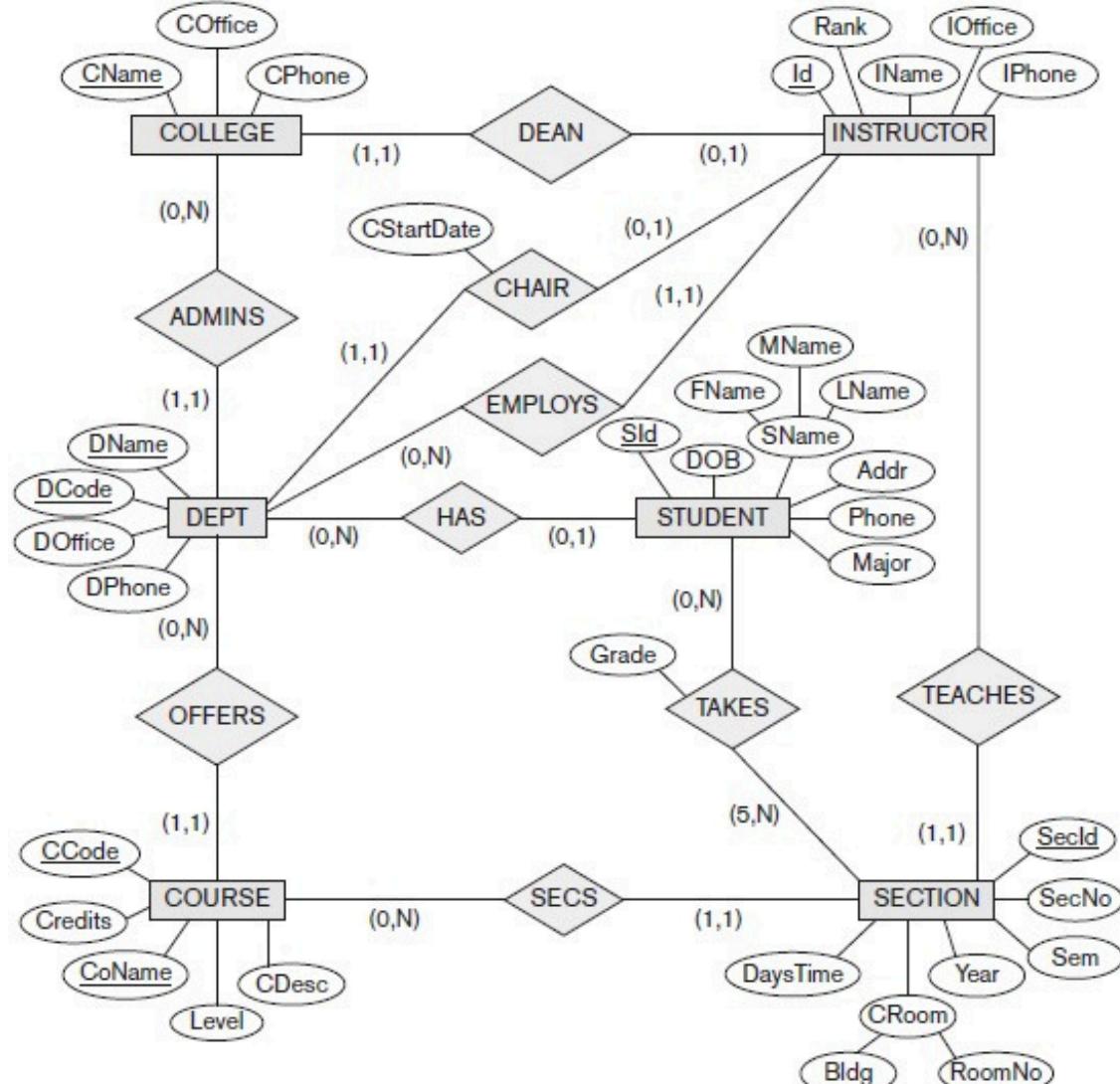
One-to-Many Relationship



Many-to-Many Relationship



ตัวอย่างแผนภาพแสดงความสัมพันธ์ของ Entity



An ER diagram for a UNIVERSITY database schema.

แบบจำลองแนวทำงานเชิงวัตถุ(Object-Oriented Approach)

- ใช้ในกระบวนการวิเคราะห์หน้าที่การทำ งานระบบ และใช้เป็นเครื่องมือสื่อสารแนวคิดในการพัฒนาระบบ
- เน้นการนิยามคลาส และลักษณะที่คลาสรаботา งานร่วมกัน
- วัตถุ (Object)ประกอบด้วยข้อมูล (Properties) และกระบวนการทำการทำงาน (Behavior)
- แบบจำลองเชิงวัตถุ UML (Unified Modeling Language) แบ่งออกเป็น 2 กลุ่ม ดังนี้
 - Structural Diagram เป็นกลุ่มแผนภาพที่แสดงให้เห็นโครงสร้างเชิงสถิติ (Static) ของระบบ
 - Behavioral Diagram เป็นกลุ่มแผนภาพให้เห็นภาพเชิงกิจกรรมของระบบ (Dynamic)

- Class Diagram
- Object Diagram
- Component Diagram
- Deployment Diagram

- Use Case Diagram
- Sequence Diagram
- Collaboration Diagram

-
-

- State Diagram
- Activity Diagram



แผนภำพคลาส (Class Diagram)

- Class

- แม่แบบที่ใช้สร้างกรณีตัวอย่าง หรือ Object ของระบบ
- โครงสร้างและพฤติกรรมที่เหมือนกัน และ จามี attribute ที่ต่างกัน
- แผนภำพคลาสสร้า งจำกมุน มองของ นักพัฒนาซอฟต์แวร์

```
//Abstract animal class
public abstract class Animal
{
    private String name; public abstract
    void speak(); public String getName() { }
    public void setName(String animalName)
    { }
    return name;
}

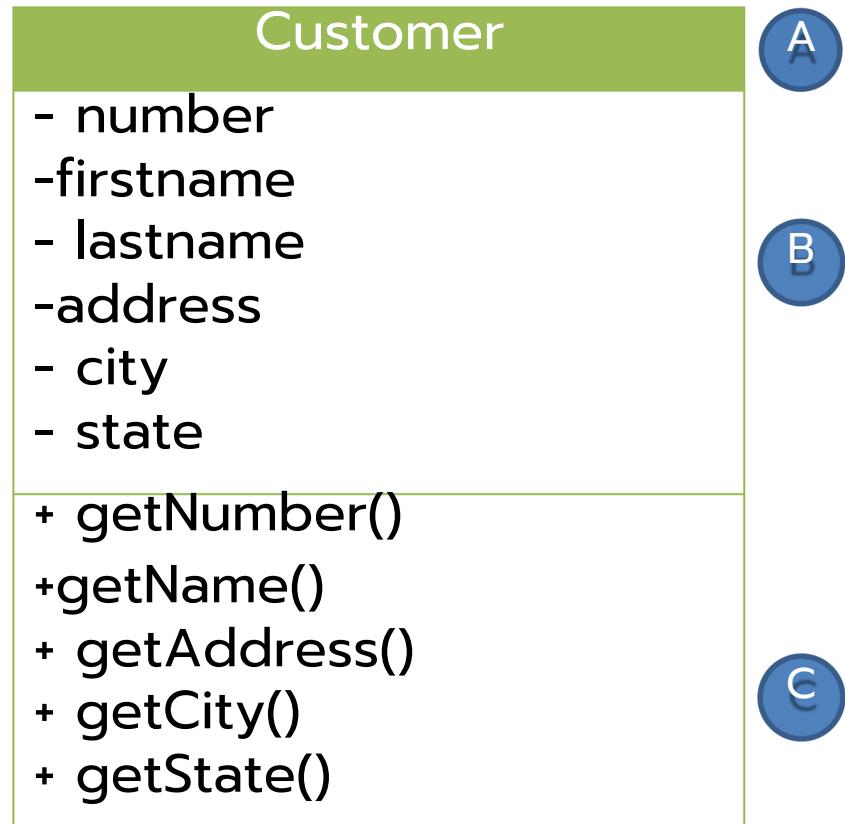
name = animalName;

}

public class Dog extends Animal
{
    public void speak()
    { }
    System.out.println("Woof!");
}
```

แผนกำ派คลาส (ต่อ)

- **Attributes**
 - ส่วนของข้อมูลหลักๆ ที่รวมกันเป็นร้ายละเอียดของคลาส
 - ประกอบด้วย ข้อมูลสำคัญที่ควบคุมมีการจัดเก็บเข้าสู่ระบบ และเป็นชนิดข้อมูลแบบพื้นฐาน
- **Operations**
 - การกระทำที่ Object สามารถทำได้ อาจมองได้เป็น
 - Method/Service/Behavior



แผนภำพคลาส (ต่อ)

สีต รี	สัญลักษณ์	การเข้าถึงข้อมูลภายนอก
Public	+	ทุกๆ Object ภายนอกในระบบสามารถเห็นหรือเข้าถึงข้อมูลในคลาสได้โดยตรง
Protected	#	Object ของคลาสที่มีคุณลักษณะเป็นคลาสสับก่อตั้ง คลาสวันสำหรับเห็นข้อมูลภายนอกในคลาส แต่ไม่สำหรับเข้าถึงข้อมูลภายนอกในคลาสได้
Private	-	Object จำกัดความสามารถ ด้วยเจตนาเพื่อใช้งานภายนอกในคลาสเท่านั้น โดยไม่ให้คลาสวันเห็นหรือเข้าถึง

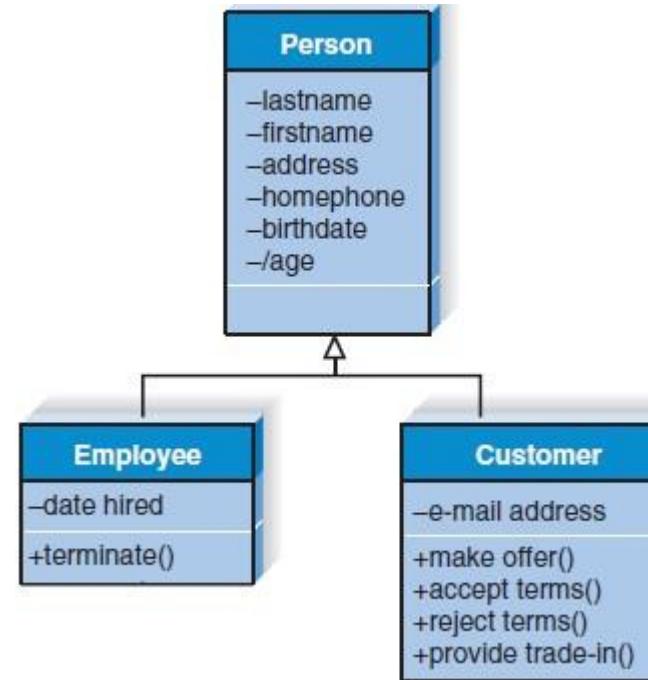
▲ Information hiding/Visibility

สัญลักษณ์และคำนิยามของแผนภำพคลาส

Term and Definition	Symbol
A class	Class name ■ Represents a kind of person, place, or thing about which the system must capture and store information. ■ Has a name typed in bold and centered in its top compartment. ■ Has a list of attributes in its middle compartment. ■ Has a list of operations in its bottom compartment. ■ Does not explicitly show operations that are available to all classes.
An attribute	Attribute name /derived attribute name +Operation name ()
A method	Operation name ()
An association	1..* verb phrase 0..1

แผนกำ派คลาส: ความสัมพันธ์ (Relationship)

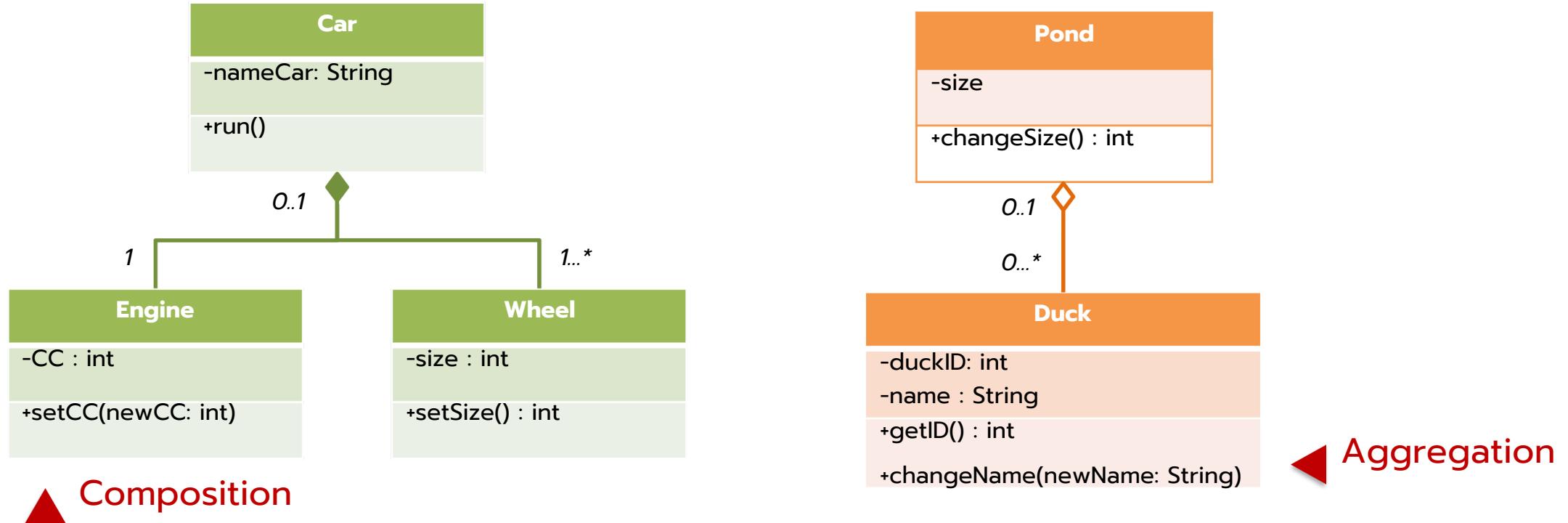
- ความสัมพันธ์แบ่งออกเป็น 5 ประเภท ดังนี้
 - Generalization
 - Composition
 - Aggregation
 - Association
 - Dependency



Generalization Relationships ➔

คลาสที่สืบทอดลักษณะประจำ หรือการดำเนินการจำกคลาสขึ้น
Superclass (1) หรือ Subclass (*)
A-kind-of

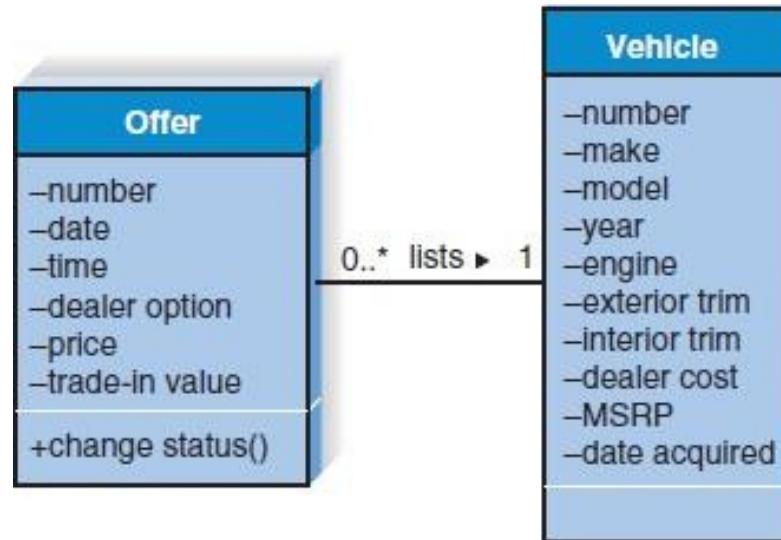
Aggregation/Composition Relationships



- เป็นความสัมพันธ์การรวมกลุ่ม/การประกอบ
- A-part-of/Has-part

Association Relationships

- เป็นความสัมพันธ์แบบก้าวไประหว่างคลาส
- ความสัมพันธ์แบบนี้อาจถูกมองในรูปของ *มีอยู่* (*has-a*) ได้
 - Object จำกัดคลาสหนึ่งมีค่าอ้างอิงไปยัง Object จำกัดคลาสอื่นได้



สัญลักษณ์และความหมายของความสัมพันธ์แบบ Association

Instance(s)	Representation of Instance(s)	Diagram Involving Instance(s)	Explanation of Diagram
Exactly one	1	<pre>graph LR; Department[Department] --- 1 Boss[Boss]</pre>	A department has one and only one boss.
Zero or more	0..*	<pre>graph LR; Employee[Employee] --- 0..* Child[Child]</pre>	An employee has zero to many children.
One or more	1..*	<pre>graph LR; Boss[Boss] --- 1..* Employee[Employee]</pre>	A boss is responsible for one or more employees.
Zero or one	0..1	<pre>graph LR; Employee[Employee] --- 0..1 Spouse[Spouse]</pre>	An employee can be married to zero or one spouse.
Specified range	2..4	<pre>graph LR; Employee[Employee] --- 2..4 Vacation[Vacation]</pre>	An employee can take between two to four vacations each year.
Multiple, disjoint ranges	1..3, 5	<pre>graph LR; Employee[Employee] --- 1..3, 5 Committee[Committee]</pre>	An employee is a member of one to three or five committees.

Dependency Relationships

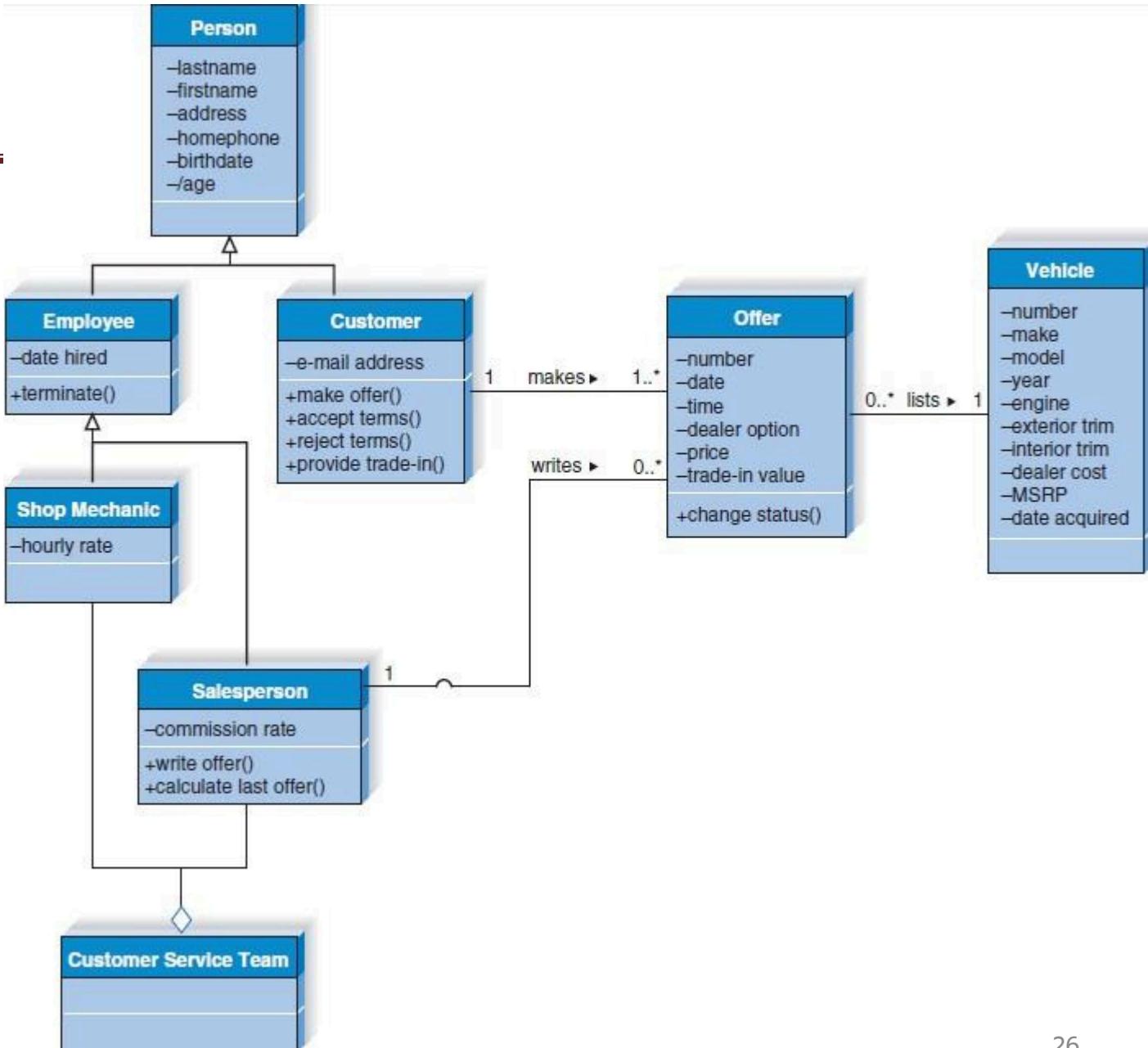
- เป็นความสัมพันธ์แบบขึ้นต่อ กัน
- การเปลี่ยนแปลงของอีก คลาสส์ ผลต่อ อีก คลาสที่ขึ้นต่อ กัน
 - ความสัมพันธ์นี้ เกิดขึ้น ในระยะเวลามั่นๆ



แนวทำงสำหรับการจัดทำแผนกำกับคลาส

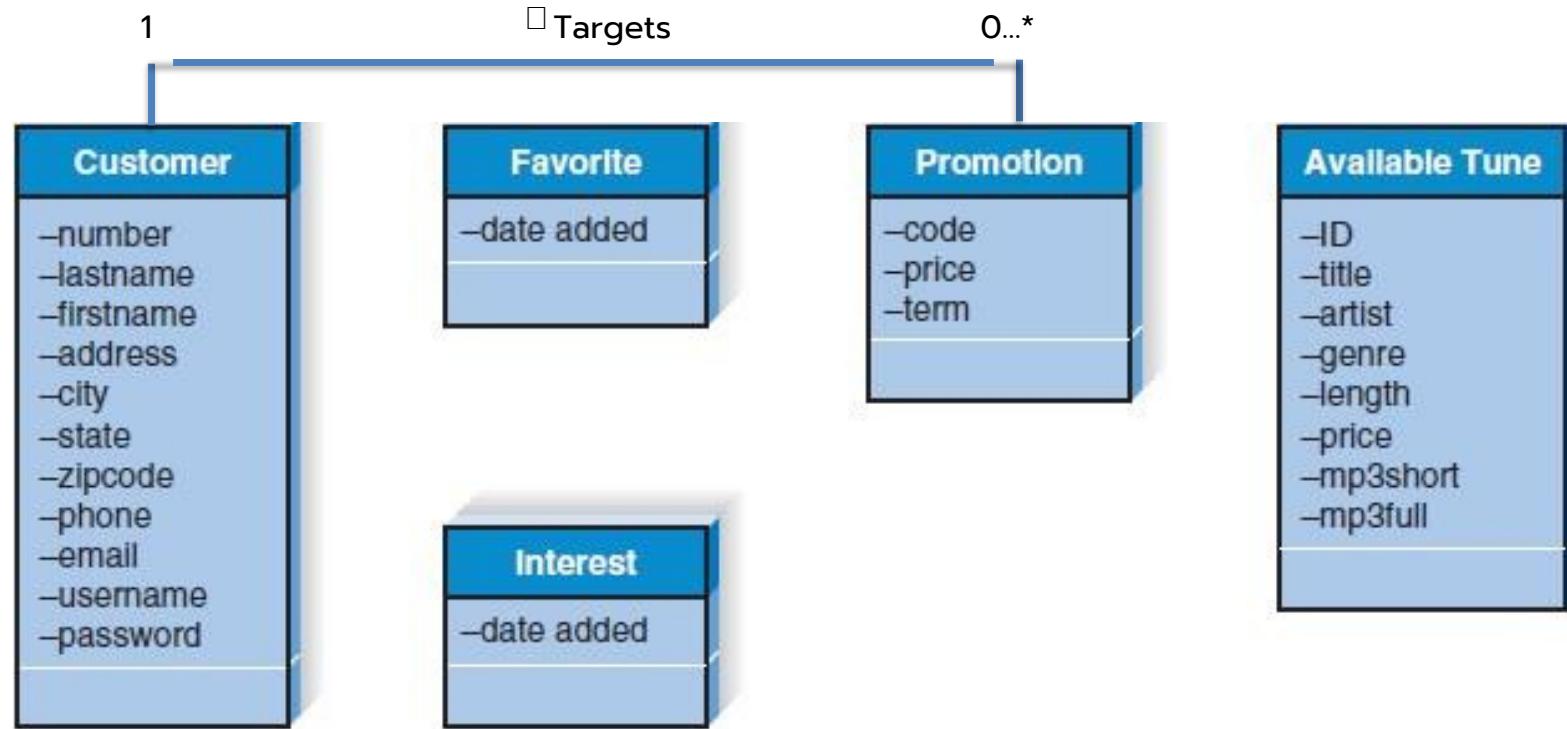
Guideline	Example
Nouns → imply objects or classes. A common or improper noun implies a class of objects. A proper noun or direct reference implies an instance of a class. A collective noun implies a class of objects made up of groups of instances of another class.	"An employee serves the customer" implies two classes of objects, employee and customer. "John addressed the issues raised by Susan" implies two instances of an object, John and Susan. "The list of students was not verified" implies that a list of students is an object that has its own attributes and methods.
Verbs → imply associations or operations. A doing verb implies an operation. A being verb implies a classification association between an object and its class. A having verb implies an aggregation or association relationship. A transitive verb implies an operation. A predicate or descriptive verb phrase implies an operation.	"Don files purchase orders" implies a "file" operation. "Joe is a dog" implies that Joe is an instance of the dog class. "The car has an engine" implies an aggregation association between car and engine. "Frank sent Donna an order" implies that Frank and Donna are instances of some class that has an operation related to sending an order. "If the two employees are in different departments, then ..." implies an operation to test whether employees are or are not in different departments.
Adjectives → imply attributes of a class. An adjective implies an attribute of an object.	"All 55-year-old customers are now eligible for the senior discount" implies that age is an attribute.
Adverbs → imply an attribute of an association or operation. An adverb implies an attribute of an association or an attribute of an operation.	"John drives very fast" implies a speed attribute associated with the driving operation.
These guidelines are based on Russell J. Abbott, "Program Design by Informal English Descriptions," <i>Communications of the ACM</i> , 26(11), 1983, pp. 882-94; Peter P-S Chen, "English Sentence Structure and Entity-Relationship Diagrams" <i>Information Sciences: An International Journal</i> , 29(2-3), 1983, pp. 127-149; and Ian Graham, <i>Migrating to Object Technology</i> , Reading, MA: Addison-Wesley, 1995.	

ตัวอย่าง Class Diagram(ต่อ)



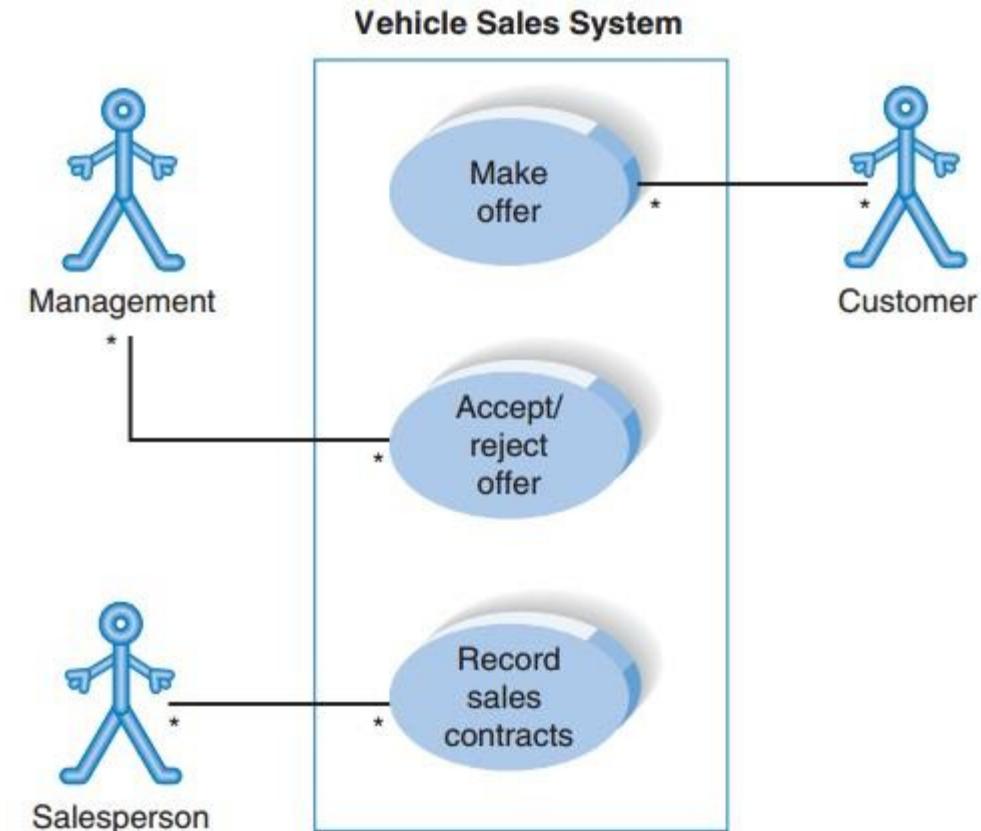
กิจกรรมที่ 6.1

จงแสดงความสัมพันธ์ของ Class Diagram แบบ Association



แบบจำลองยูสเคส (Use Case Modeling)

- **แบบจำลองยูสเคส**
 - แสดงกิจกรรมที่ผู้ใช้สำหรับระบบ
 - ระบุ
 - แบบจำลองเชิงตรรกะ ที่อธิบายกิจกรรมของระบบ
 - ยังไม่ระบุกิจกรรมจะถูกพัฒนาขึ้นได้อย่างไร/
ใช้เทคโนโลยีใด
 - ประกอบด้วย 2ขั้นตอนหลัก ดังนี้
 - เขียนคำอธิบายยูสเคส (Use Case Description)
 - แปลงคำอธิบายยูสเคสเป็นแผนภูมิ (Use Case Diagram)



คำอธิบายยูสเคส (Use Case Description)

- อธิบายถึง พัง กซ น กำรทำงำนพื้นฐำนของระบบ
- ผู้ใช้ก ำา ะไร/ระบบต้องสนองอย่างไร
- ยูสเคส 1 ไดๆ
 - อธิบายพังก์ชันกำรก ำา งำนของระบบเพียง 1พังก์ชัน
 - สร้างขึ้นจำกบุมมองของผู้ใช้ในหลาย ๆ กรณี
- ส่งที่ต้องพึงระวัง
 - ยูสเคสที่จัดทำขึ้น เกี่ยวข้องกับบทบาทของผู้ใช้
แทนที่จะ เกี่ยว วข้อ งกับ ผู้ใช้ ซึ่ง ค คลิดบุค คลหนึ่ง



องค์ประกอบของคำอธิบายยูสเคส (Elements of a Use Case Description)

ชื่อยูสเคส

(Use Case Name)

ผู้กระทำการหลัก (Primary Actor)		รหัส (UseCase ID)	
ผู้มีส่วนเกี่ยวข้องและกำรใช้ประโยชน์			
คำอธิบาย			
(Brief Description)			
เงื่อนไขก่อนกำรท ำงำນ			
(Precondition)			
สั่งกระตุ้น			
(Trigger)			
ขั้นตอนกำรท ำงำนปกติ			
(NormalFlow of Events)			
ขั้นตอนกำรท ำงำนย่อย (Sub flows)			
ขั้นตอนกำรท ำงำนทำงเลือก/พิเศษ			
(Exception Flows)			
เงื่อนไขหลังกำรท ำงำນ			
(Postcondition)			
			30

ตัวอย่าง ข้อมูล UML Use Case

รับการสั่งซื้อจากลูกค้า

ฝ่ายวางแผนและควบคุม

ชื่อユースเคส (Use Case Name)		รหัส (Use Case ID)	UDC-001 เชิงลงทะเบียน	ระดับความสำคัญ (Importance Level)	มาก
ผู้ประกอบการหลัก (Primary Actor) ผู้มีส่วนเกี่ยวข้องและกำรใช้		ประเภทยูสเคส (UseCase Type)	ประเภทยูสเคส		
ประโยชน์ คำอธิบาย	-ฝ่ายวางแผนและควบคุมกำรผลิต: ต้องกำรเพิ่มรำยกำรสั่งซื้อสินค้าจำกลูกค้า เป็นยูสเคสที่ใช้ในกำรแสดงวิธีกำรเพิ่มรำยกำรสั่งซื้อสินค้าลงยังฐานข้อมูล				
(Brief Description) เงื่อนไขก่อนกำรทำjob					
(Precondition) สั่งกระตุ้น	ต้องกำรรู้ข้อมูลผู้ใช้งานได้แก่ ชื่อผู้ใช้งานและรหัสผ่าน สำหรับตรวจสอบและดำเนินกำร				
(Trigger)	เมื่อลูกค้ามีกำรสั่งซื้อสินค้าเข้ามายังบริษัท				

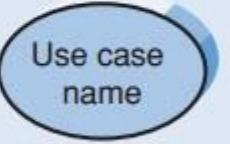
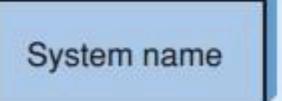
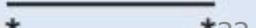


ຕัวอย่างคำอธิบายยูสเคส (ต่อ)

ขั้นตอนการกำหนดงานปกติ (Normal Flow of Events)	<ol style="list-style-type: none">ระบบแสดงระบบรับ สั่ง ชื่อผู้ใช้ ซัก รอกรหัส ใบสั่ง ชื่อระบบ กำหนดตรวจสอบใบสั่งชื่อเทียบกับฐานข้อมูลผู้ใช้ ซัก รอกรายละเอีย ดของสินค้า จำนวนที่ต้องการซื้อ ชนิด สินค้า ที่ต้องการสั่ง ชื่อ และวัน เวลาในการจัดส่งผู้ใช้ ซัก ดปุ่ม เพิ่มสินค้า ลงในรายการระบบ กำหนดบันทึกรายการสินค้าที่สั่งชื่อลงยังฐานข้อมูล
ขั้นตอนการกำหนดงานย่อย (Sub flows)	ไม่มีระบุ
ขั้นตอนการกำหนดงานกำลังเลือก/ พิเศษ (Exception Flows)	2-1: กรณีหัสกับฐานข้อมูล 2-1-1: ระบบ กำหนดรหัสที่ผู้ใช้กรอก และทำการแสดงผลให้ผู้ใช้กรอกรหัสใหม่ และแสดงสินค้าที่ได้รับ กำหนดเพิ่มแล้ว และมีรหัสตรงกันให้กรับ
เงื่อนไขหลังการกำหนด (Postcondition)	รายการสั่ง ชื่อ สินค้า ในฐานข้อมูล ถูกสร้างขึ้น

ແຜນກຳພູສເຄສ (Use Case Diagrams)

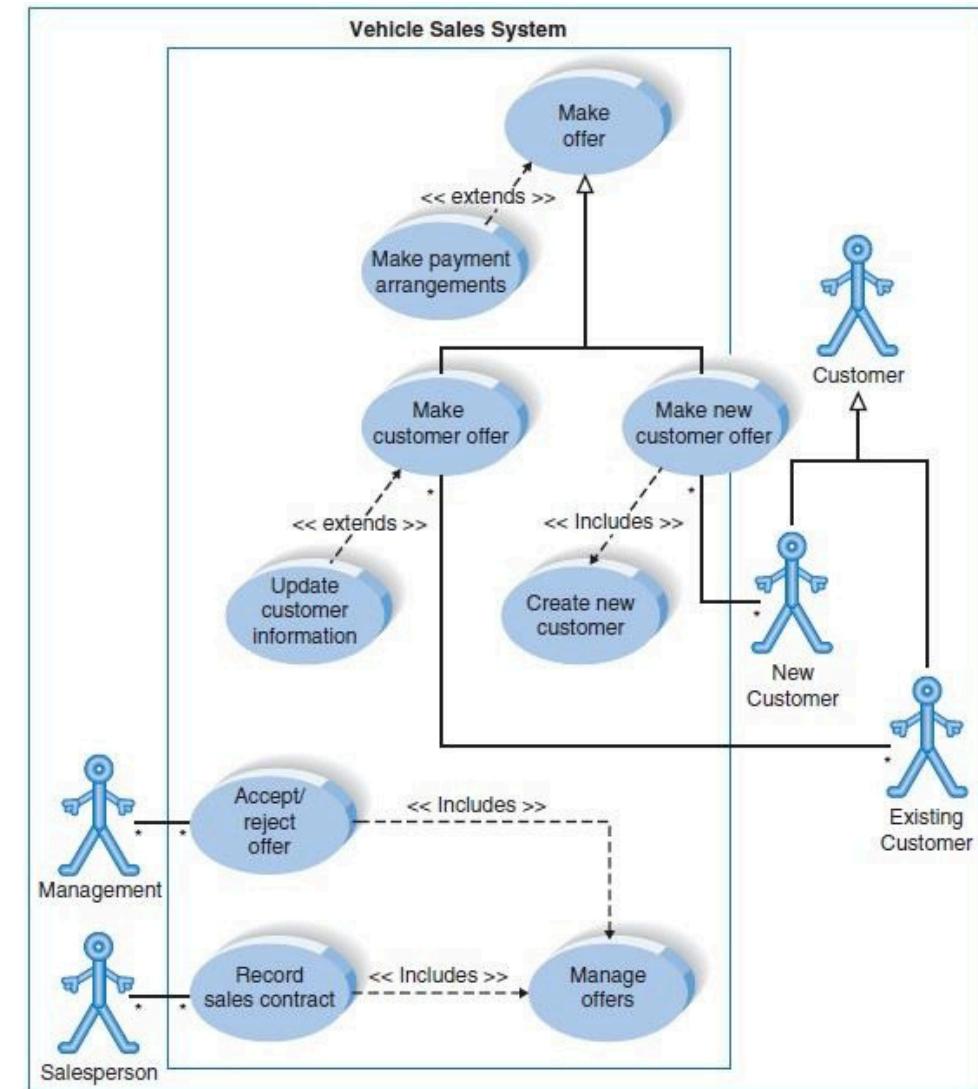
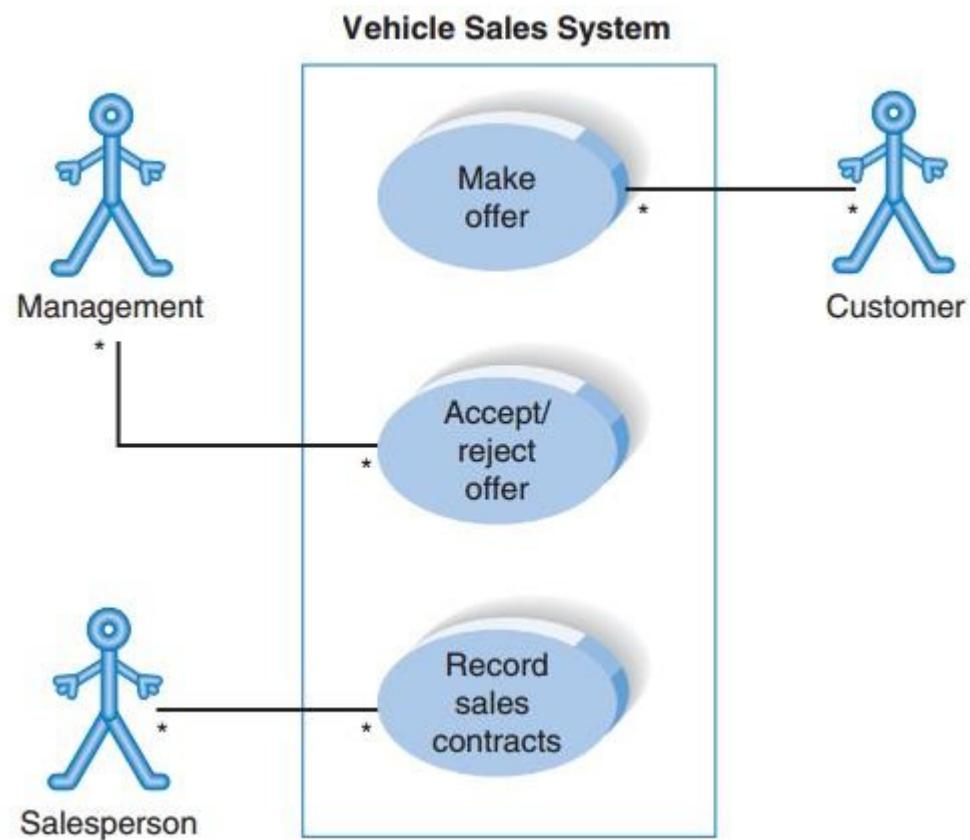
- ຂັ້ນຕອນຫລັ້ງຈຳກເບີຍນົມ ອຣີບໍາຍພູສ
- ເຄສ ສຽງພູສເຄສກັ້ນມຸດໄວ້ເປັນແຜນ ກຳພໍ ທີ່
ແສດຖານທີ່ຂ່າຍກຳນົມຫລັກຂອງຮະບບ
– ຜູ້ໃຊ້ ຊະນູ້ ແລະ ຜູ້ກໍ ເກີຍ່ ວັ້ນ ກັບ ຮະບບ
- ວິສວກຮອບພົມ/ຜູ້ໄຄຮ່າງຮະບບເຂົ້າໃຈ ກຳພຽນຂອງຮະບບ

Term and Definition	Symbol
<p>An actor</p> <ul style="list-style-type: none"> ■ Is a person or system that derives benefit from and is external to the system. ■ Is labeled with its role. ■ Can be associated with other actors by a specialization/superclass association, denoted by an arrow with a hollow arrowhead. ■ Is placed outside the system boundary. 	 Actor role name
<p>A use case</p> <ul style="list-style-type: none"> ■ Represents a major piece of system functionality. ■ Can extend another use case. ■ Can use another use case. ■ Is placed inside the system boundary. ■ Is labeled with a descriptive verb–noun phrase. 	
<p>A system boundary</p> <ul style="list-style-type: none"> ■ Includes the name of the system inside or on top. ■ Represents the scope of the system. 	
<p>An association relationship</p> <ul style="list-style-type: none"> ■ Links an actor with the use case(s) with which it interacts. 	

สัญลักษณ์และคำจำกัดความหมายของแผนภำพยูสเคส (ต่อ)

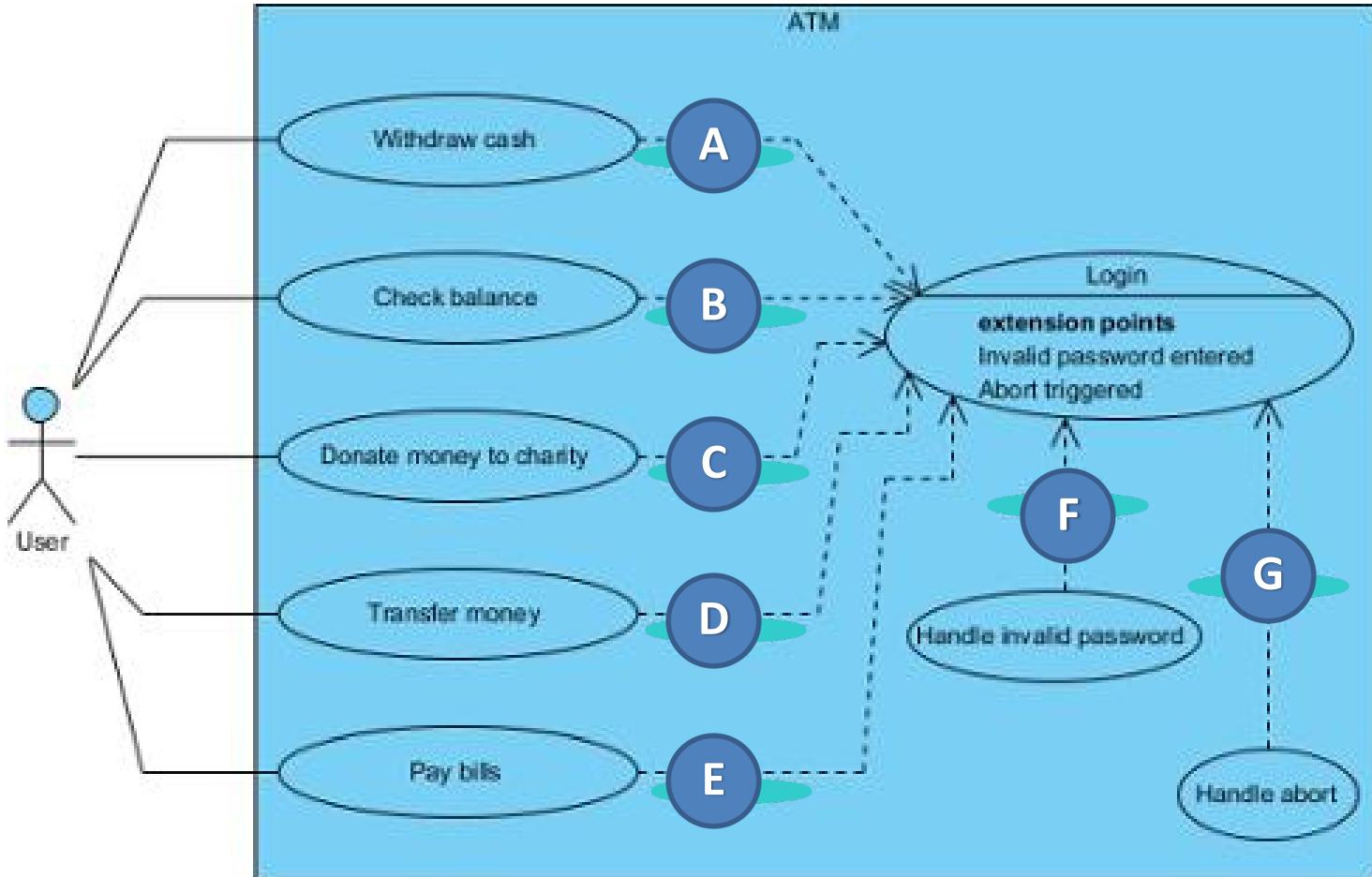
Term and Definition	Symbol
<<include>> 	An include relationships specifies how the behavior for the inclusion use case is inserted into the behavior defined for the base use case.
<<extend>> 	An extend relationships specifies how the behavior of the extension use case can be inserted into the behavior defined for the base use case.
	A generalization relationship is used to represent inheritance relationship between model elements of same type. The more specific model element share the same specification with the more general the model element but carries more details in extra.

ตัวอย่างแผนกำรพยุสคес



ตัวอย่าง ระบบ ATM

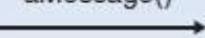
- ผู้ใช้งานต้องสอดบัตร ATM เข้าสู่เครื่องรับบัตร หากบัตรใช้งานได้จึงเข้าสู่หน้าจอ Main Menu หากใช้งานไม่ได้บัตร ATM จะถูกปล่อยคืน (Reject) ออกมานอก
- ในการนี้บัตรใช้งานได้
 - ผู้ใช้สามารถตรวจสอบยอดเงินคงเหลือในบัญชี แต่ละบัญชี
 - ผู้ใช้สามารถระบุประเภทบัญชีและจำนวนเงินที่ต้องการถอน/โอน/บริจาค/จ่ายค่าสำหรับบุปผา ถ้ามีเงินในบัญชีมากกว่าหรือเท่ากับจำนวนที่ระบุ ผู้ใช้งานจะสามารถกดทำธุรกรรมได้
- ผู้ใช้งานต้องเป็นสมาชิกกับธนาคารก่อน จึงจะสามารถใช้บริการของ ATM ได้ ส่วนหัวข้อตอนการเข้าสู่ระบบ ระบบต้องสำหรับตรวจสอบข้อมูลสมาชิก
- กรณีที่ระบบไม่สำหรับกดทำธุรกรรมนั้นแล้วเสร็จได้ระบบควรเมื่อกำจัดเงินผู้ใช้และยกเลิกการทำธุรกรรมกันที



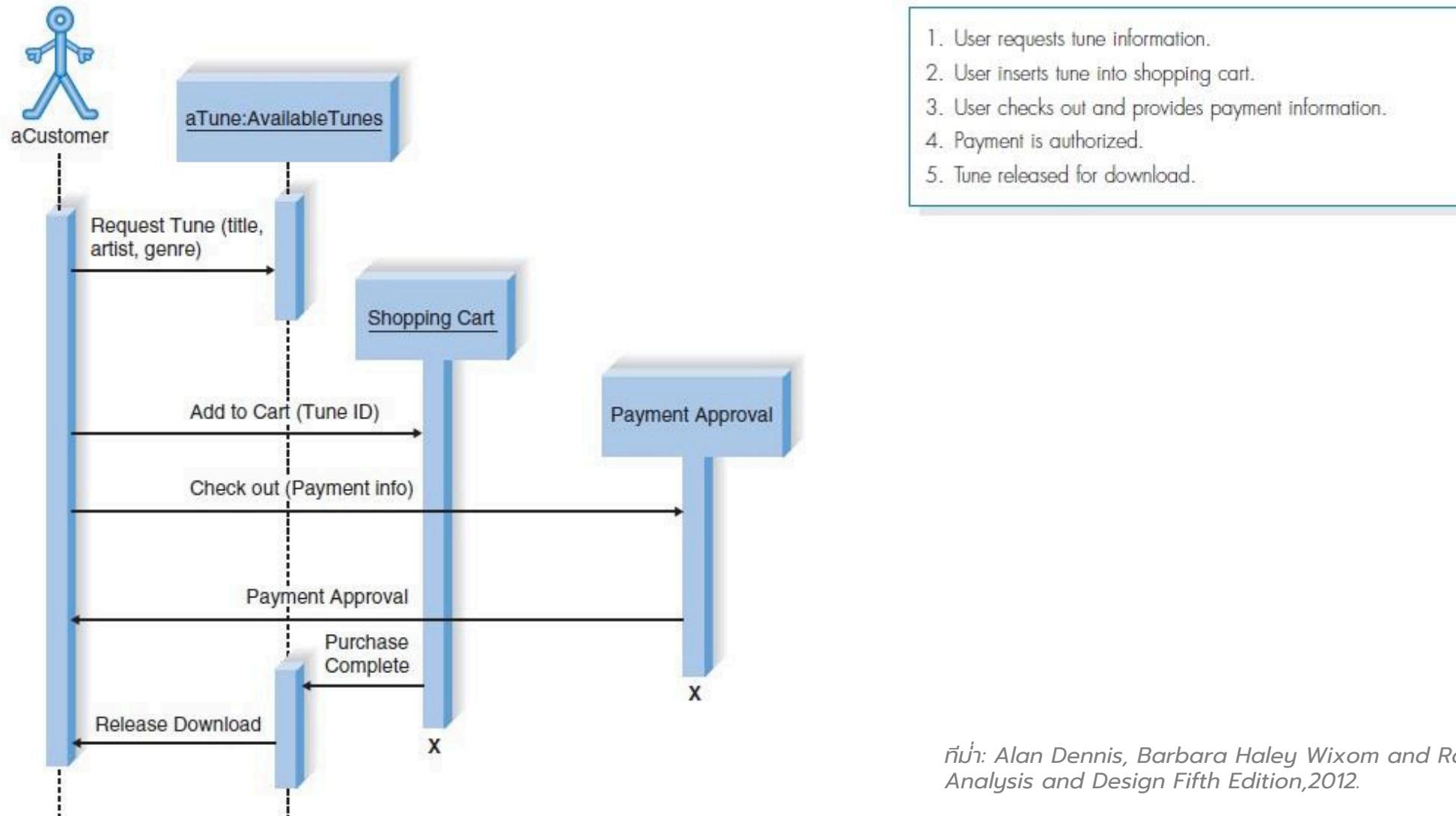
แผนภำபลำดับภารกิจ

(Sequence Diagram)

- เป็นแผนภำพที่แสดงให้เห็น
 - การติดต่อกันระหว่าง Object / พังก์ชันภารกิจ
 - การส่งผ่านข้อมูลระหว่างพังก์ชันภารกิจ
 - ลำดับภารกิจโดยทั่วไปในทุกกรณีที่เป็นไปได้ของระบบ

Term and Definition	Symbol
<p>An actor:</p> <ul style="list-style-type: none"> Is a person or system that derives benefit from and is external to the system. Participates in a sequence by sending and/or receiving messages. Is placed across the top of the diagram. 	 anActor
<p>An object:</p> <ul style="list-style-type: none"> Participates in a sequence by sending and/or receiving messages. Is placed across the top of the diagram. 	 anObject:aClass
<p>A lifeline:</p> <ul style="list-style-type: none"> Denotes the life of an object during a sequence. Contains an X at the point at which the class no longer interacts. 	
<p>A focus of control:</p> <ul style="list-style-type: none"> Is a long narrow rectangle placed atop a lifeline. Denotes when an object is sending or receiving messages. 	
<p>A message:</p> <ul style="list-style-type: none"> Conveys information from one object to another one. 	 aMessage()
<p>Object destruction:</p> <ul style="list-style-type: none"> An X is placed at the end of an object's lifeline to show that it is going out of existence. 	

ตัวอย่างแผนกำลังดับกำรทำงาน

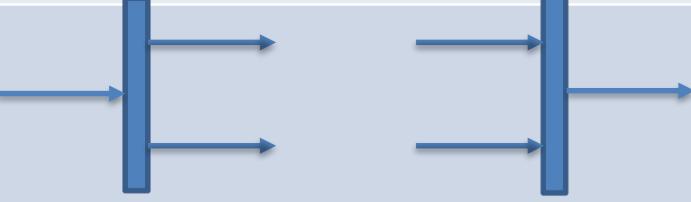
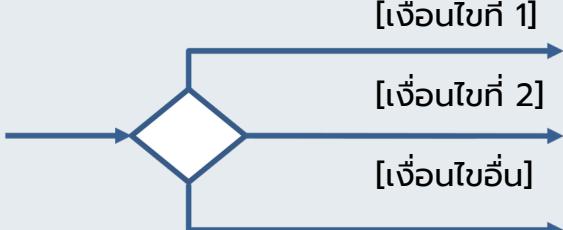


ที่มา: Alan Dennis, Barbara Haley Wixom and Roberta M, Systems Analysis and Design Fifth Edition, 2012.

แผนภำพกิจกรรม (Activity Diagram)

- แบบจำลองกระบวนการคุ้มครองเชิงธุรกิจ และแสดงให้เห็นถึง
 - กิจกรรมต่างๆ ที่เกิดขึ้นในกระบวนการคุ้มครองเชิงธุรกิจขององค์กร
 - ลำดับขั้นตอนการท่องานบุคคลที่จะเกิดขึ้นของระบบ
 - แต่ละมุมมองของผู้ดูแลระบบ
 - กระบวนการคุ้มครองที่เกิดขึ้นกับผลลัพธ์ของกิจกรรมหนึ่ง เป็นข้อมูลน้ำหน้าของกิจกรรมใด
- Activity Diagram ลักษณะเดียวกับ Flowchart (แสดงขั้นตอนการท่องานของระบบ) โดยขั้นตอนในกระบวนการคุ้มครองแต่ละขั้นตอนซึ่งเรียกว่า Activity
- วัตถุประสงค์ในการใช้ Activity Diagram
 - อธิบาย กระบวนการไหลของกระบวนการ (Workflow)
 - และแสดงขั้นตอนการทำงานของระบบ
- Activity อาจเป็นการทำงานต่างๆ ได้แก่
 - การคำนวณผลลัพธ์ของอย่างอื่น
 - การเปลี่ยนแปลงสถานะ (State) ของระบบ
 - การส่งค่ากลับคืน
 - การส่งสัญญาณ
 - กำหนดเรียกให้ Operation (Method) อื่นๆ เพื่อทำงาน
 - การสร้าง หรือ กำลังวัตถุ

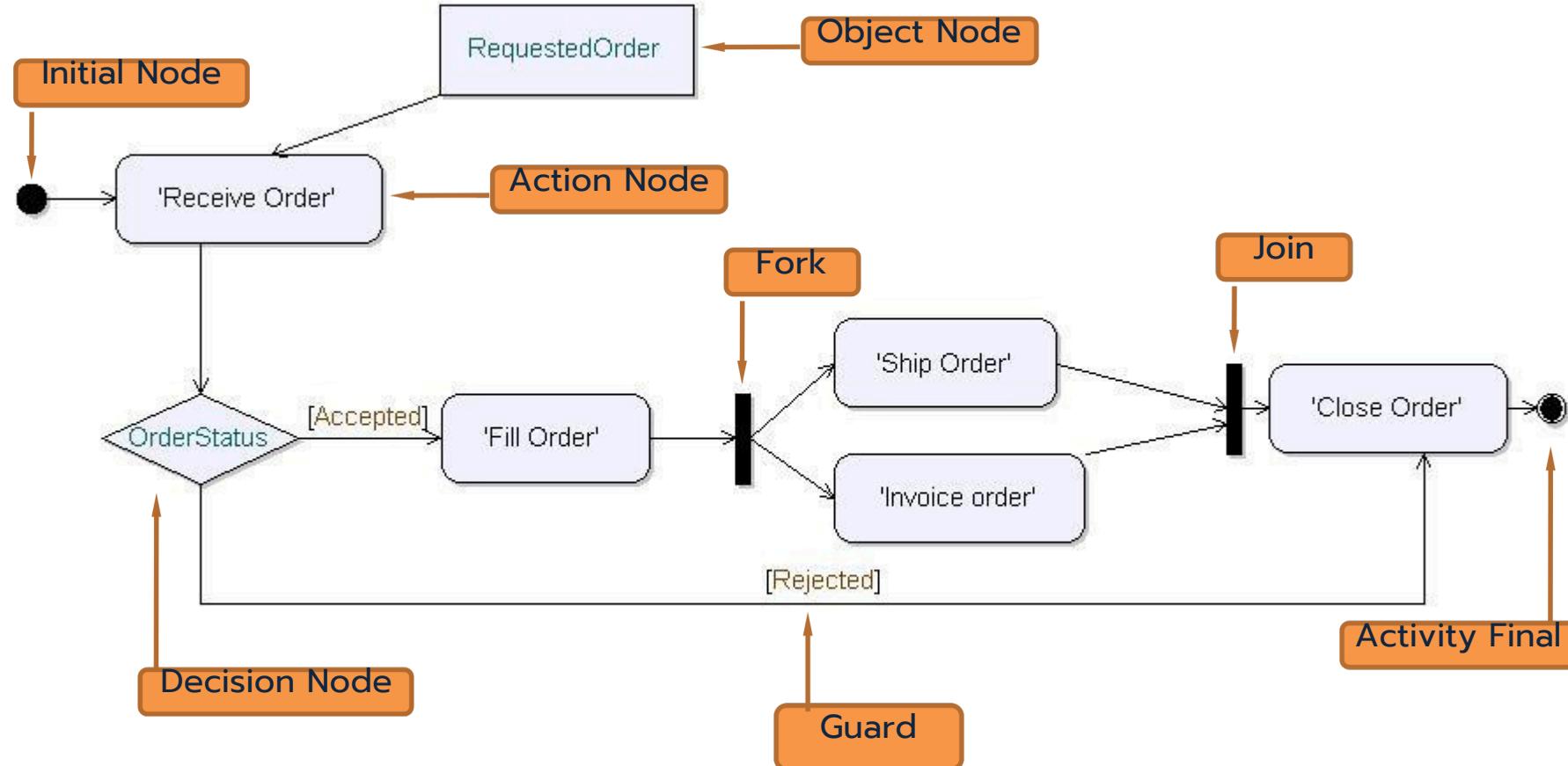
สัญลักษณ์ที่ใช้ในแผนภำพกิจกรรม

สัญลักษณ์	ความหมาย
 กิจกรรม	โน宦กิจกรรม (ActivityNode) เป็นการกำหนดกิจกรรม หรือฟังก์ชันทำการทำงาน แสดงเส้นทำงานให้ของกิจกรรม (ActionFlow) และ เชื่อมโยงจากกิจกรรมหนึ่งไปยัง กิจกรรมหนึ่ง
	โน宦แยก (ForkNode) เป็นการแยกกิจกรรมในกิจกรรมใด ๆ สำหรับแยกกิจกรรมได้มีมากกว่า 1 กิจกรรมที่สำหรับ กิจกรรมพร้อมกันได้ และ โน宦เชื่อม (Join Node) เป็นการรวมกิจกรรมเข้าด้วยกัน
	โน宦ตัดสินใจ (Decision Node) เป็นการตัดสินใจคำ กิจกรรมในกรณีมี 2 ทางเลือก ตามเงื่อนไขที่ กด ขึ้น

สัญลักษณ์ที่ใช้ในแผนภำพกิจกรรม (ต่อ)

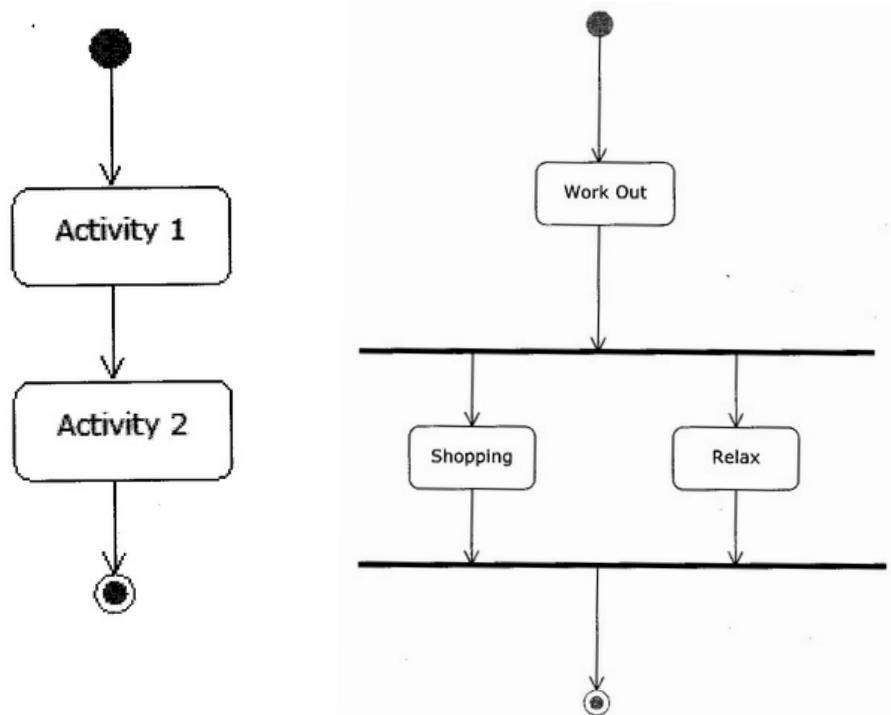
สัญลักษณ์	ความหมาย
●	โนดเริ่มต้น (Initial Node) เป็นจุดเริ่มต้น ทำการทำงานของกิจกรรม
○	โนดสิ้นสุดของกิจกรรม (Activity FinalNode) เป็นจุดสิ้นสุด ทำการทำงานของทุกกิจกรรม
⊗	โนดการไหลสุดท้าย (FlowFinal Node) เป็นการสิ้นสุดการทำงานของกิจกรรมใด ๆ โดยไม่ส่งผลกระทบกับเส้นทางการไหลอื่นของกิจกรรม
	สวีมเลนส์ (Swimlanes) หรือการแบ่งส่วน (Partition) เป็นการแบ่งแผนภำพกิจกรรมออกเป็นช่องๆ แต่ละช่องแสดงถึง Object ที่รับ ပิด ขอบกิจกรรมนั้น
	แดบซิงโครไนซ์ (Synchronization Bar) เป็นการจัดกลุ่มกิจกรรมที่มีการทำงานพร้อมกัน ในลักษณะขึ้นกันได้

สัญลักษณ์และคำจำกัดความหมายของแผนภารกิจกรรม



ลักษณะของแผนกำกิจกรรม

- Activity Diagram จะต้องผูกจุดเริ่มต้นกับจุดสิ้นสุด และในระหว่างจุดเริ่มต้นกับจุดสิ้นสุดก็จะมีขั้นตอนหรือ Activity ต่าง ๆ ของระบบ
 - การเขียน Activity Diagram โดยอ่านจากด้านบนลงล่าง



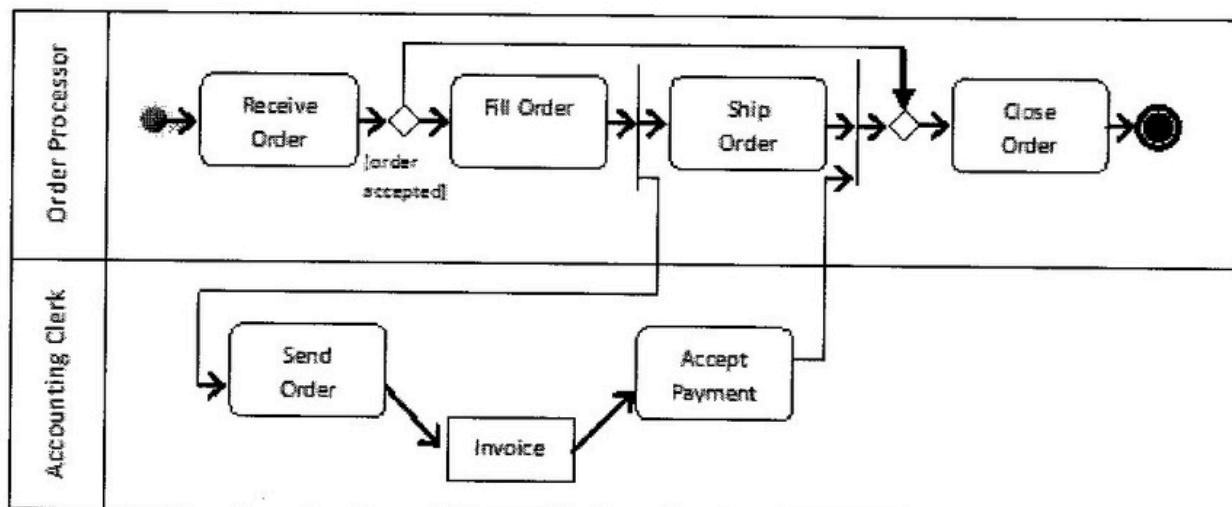
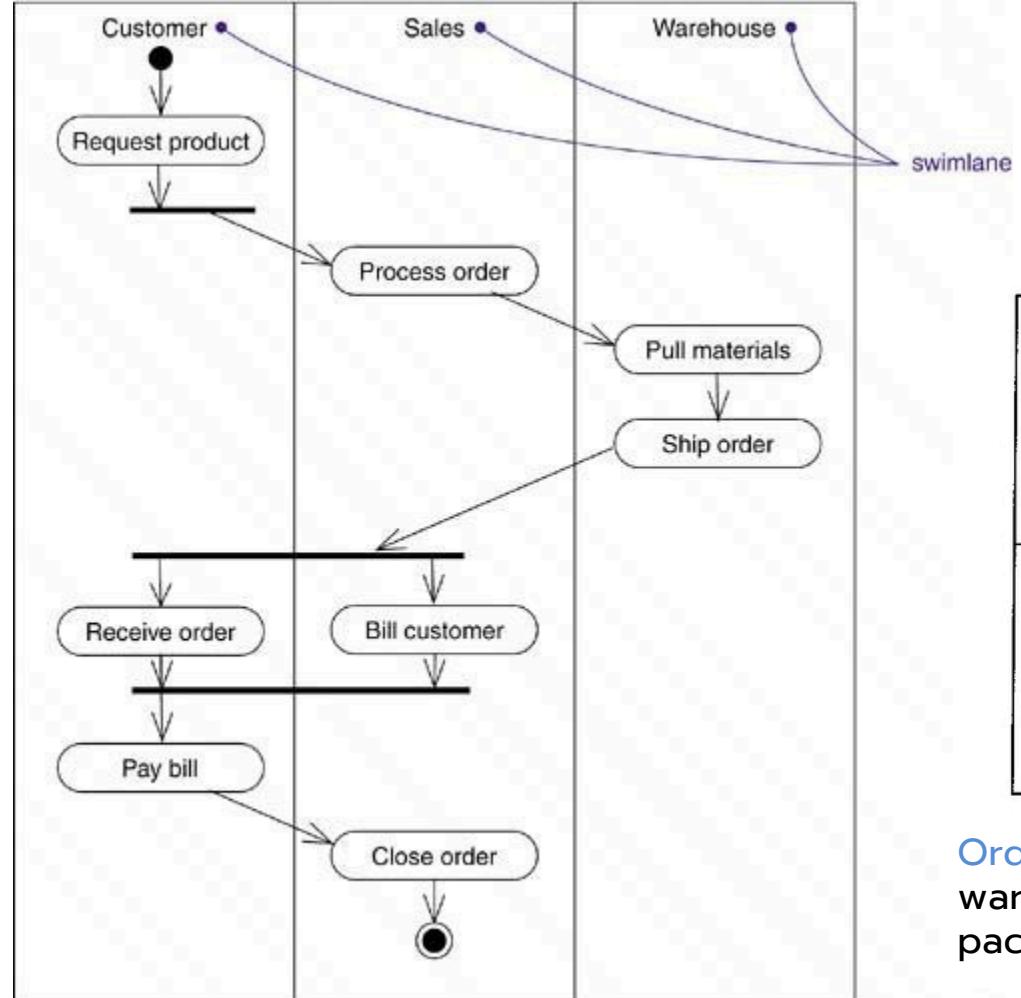
ກໍາຮກໍາງໝານແບບຂໍ້ານ

- ใช้เส้นตรงแนวโน้มเส้นหน้าที่เรียกว่า *Swim Lanes* มาเป็น สัญลักษณ์ที่ใช้จัดกลุ่มงาน
 - มีการทำงานพร้อม ๆ กัน หรือ การทำงานกิจกรรมในลักษณะคู่ขนาน

การแบ่งการทำงานให้เป็นสัดส่วน (Swimlanes)

- หลักการของกรรมการแสดงหน้าที่ จะทำโดยการแบ่งกลุ่มของกรรมการรับผิดชอบเป็นกลุ่มๆ แบ่งการทำงานให้เป็นสัดส่วน เรียกว่า *Swimlanes*
- คุณลักษณะอีกอย่างหนึ่งคือสำหรับการแสดงให้เห็นได้ว่าใครเป็นผู้มีหน้าที่รับผิดชอบในแต่ละ activity ในกระบวนการทำงานหนึ่ง ๆ
- *Swimlane*จะมีการกำหนดชื่อกำกับเอาไว้
 - กระบวนการของกรรมการสั่งซื้อสินค้า อาจแบ่งกลุ่มของคนที่มีส่วนเกี่ยวข้องเป็น 3 ส่วน ได้แก่ ก่อสั่ง ขาย และคลัง สินค้า
- การกำหนด *Swimlanes*สำหรับงานที่ในแนวตั้ง หรือ แนวอนก์ได้
- โดยแต่ละเลขจะมีชื่อผู้รับผิดชอบเพียงคนเดียว หรือ กลุ่มเดียวผู้รับผิดชอบจะทำหน้าที่ของตนที่ระบุในเลขนั้นๆ

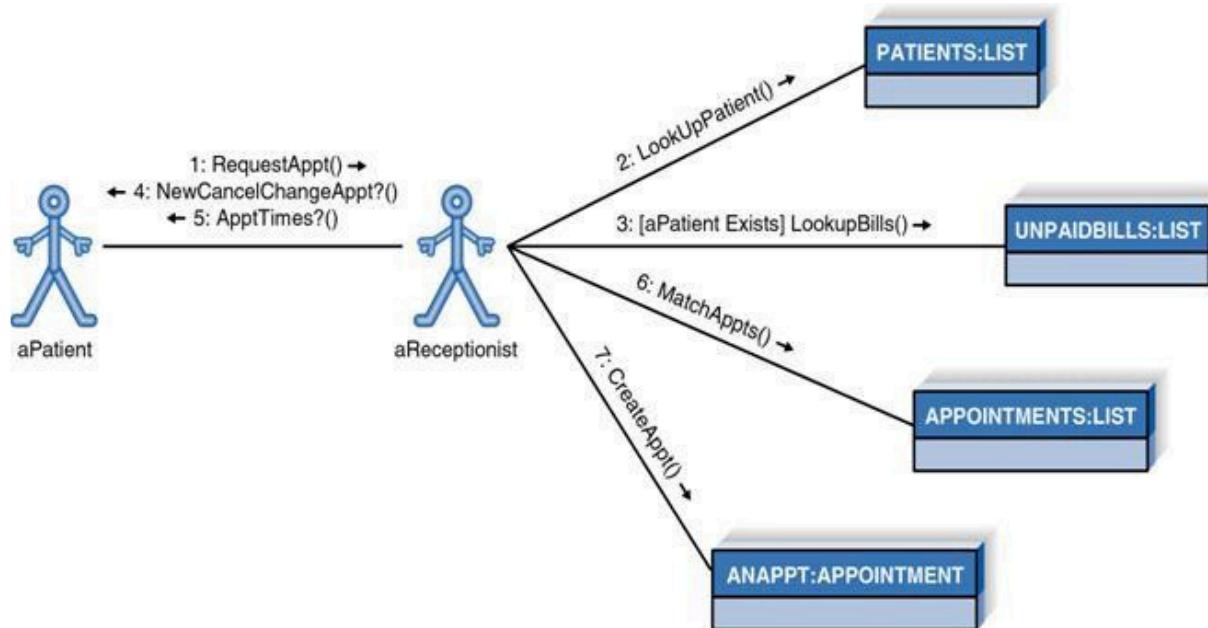
ຕົວ ອຍໍາ ກໍາຮແບ່ງ ກໍາຮກຳຈຳນໃຫ້ ປັນ ສັດ ສົວ



Order Processing -Process step where the distribution center or warehouse is responsible to *fill order*(receive and stock inventory, pick, pack and ship orders)

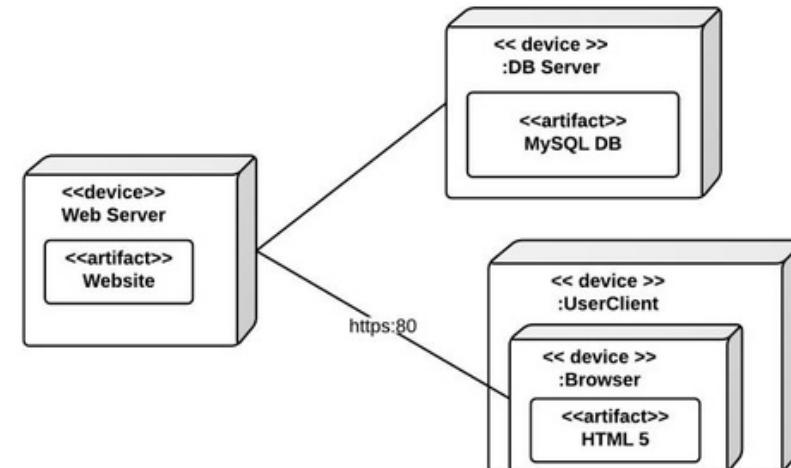
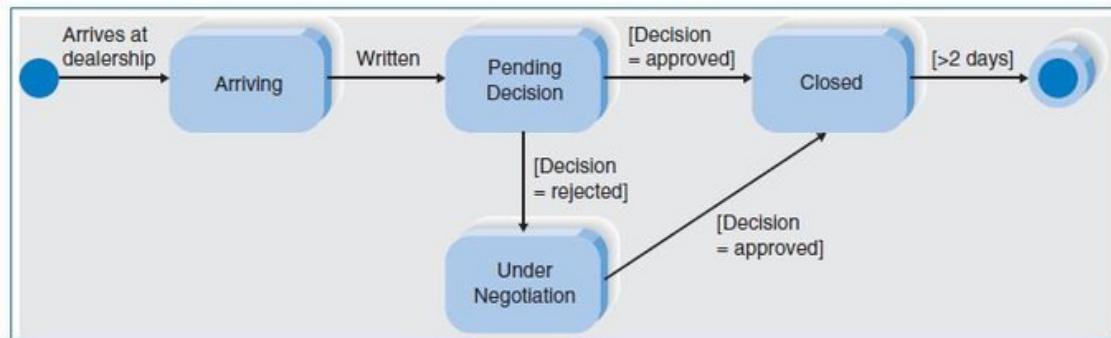
แผนภำพคอลลัมเบชัน (Collaboration Diagram)

- แผนภำพแสดงกำรปฏิสัมพันธ์ระหว่างอ็อบเจกต์ในลักษณะของกรำฟ หรือ เครื่อข่าย ซึ่ง Object จะอยู่ที่ใด ๆ ในแผนภำพก็ได้(Larman,2002)



แผนกำพสเตรทชาร์ต / แผนกำพดีพลอยเมนก

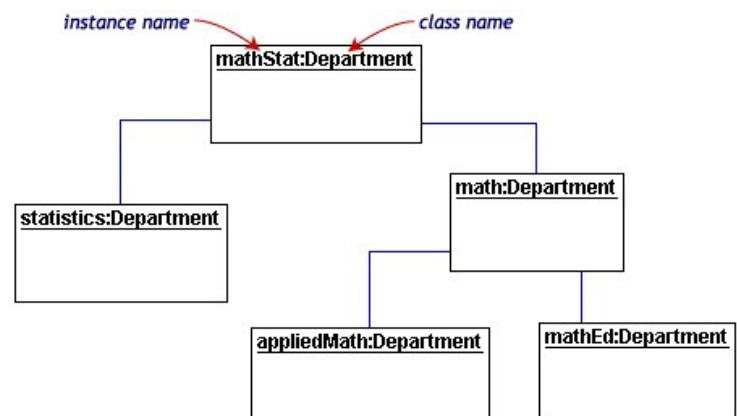
- แผนกำพสเตรทชาร์ต (State Chart Diagram)
 - แผนกำพที่แสดงเหตุการณ์ตໍ່าง ๆ ของแต่ละ State กໍມີພອກໆໄຫ້ສຳນະບອນອອບເຈັກຕໍ່ປ່ອຍແປລັງ ແລະ ພລຈຳກໍາຮຽກຮ່ວມທີ່ເກີດຂຶ້ນເນື່ອສຳນະບອນອອບເຈັກຕໍ່ນັ້ນເປັນ
- แผนกำพດີພລອຍເມນກ (Deployment Diagram)
 - ກໍາຮັດຊູສູດຳປັຕຍກຣມຂອງຮະບບໃນລັກບະລະທີ່ເປັນPhysical architecture
 - ຮະບບປະກອບດ້ວຍຄອມພິວເຕອຮີແລະອຸປກຮັນວະໄປບ້າງ



แผนกำพร้าออบเจ็กต์ / แผนกำพร้าคอมโพเนนท์

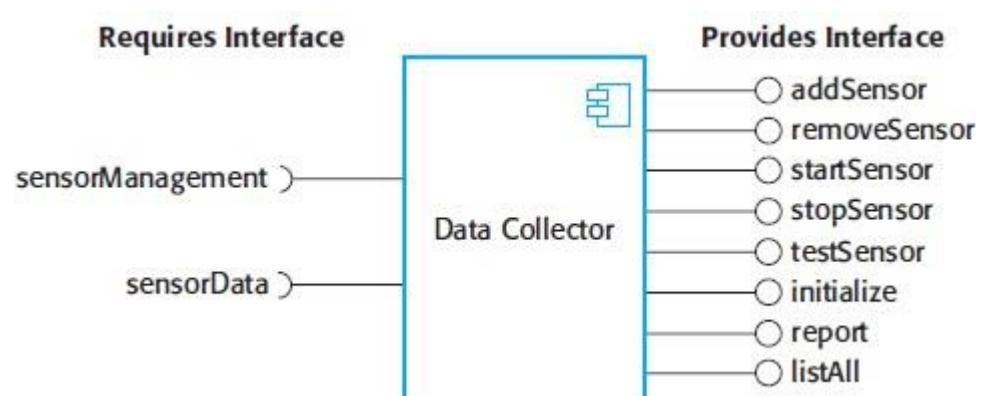
แผนกำพร้าอับเจ็กต์ (Object Diagram)

- วัตถุ คุณสมบัติ
 - เพื่อแสดงวัตถุที่ถูกสร้างขึ้นจากคลาส และจำลองคุณสมบัติที่ระบุไว้ใน Class Diagram นั้น สำหรับการใช้งานจริง ในระบบจริงหรือไม่
- สัญลักษณ์ที่ใช้จะมีลักษณะเช่นเดียวกับ Class diagram แต่กับชื่อของ Object diagram จะมีชื่อ เส้น ใต้



แผนกำพร้าคอมโพเนนท์ (Component Diagram)

- แผนภูมิที่แสดงโครงสร้างและความสัมพันธ์ระหว่างองค์ประกอบ (Components) ต่างๆ ของ Software
 - SourceCode, ExecutableProgram, Binary, Text และ User Interface



สรุป (Summary)

-
- แบบจำลองที่ใช้ในการออกแบบระบบ แบ่งเป็น 2 กลุ่ม ดังนี้
 - กลุ่ม Structural Description (Static View) ได้แก่
 - Class And Object Diagram
 - Component Diagram
 - Deployment Diagram
 - Entity Relationship Diagram ERD
 - กลุ่ม Behavioral Description (Dynamic View) ได้แก่
 - Activity Diagram
 - Collaborative Diagram
 - Data Flow Diagram
 - Flowchart and Structure Flowchart
 - Sequence Diagram
 - State Chart Diagram

กิจกรรมที่ 6.2

-
- สร้างแผนกำพ ดังนี้ แผนกำพยูสเคส (use cases) แผนกำพคลาส (class) แผนกำพล ลำดับ สำหรับ งาน (sequence) และ แผนกำพกิจกรรม (activity)
 - สำหรับน ำเสนอขั้นตอน/กระบวนการ สำหรับ ผู้ใช้บุคคล (patient) ดังนี้
 - ขั้นตอนแรก ผู้ใช้บุคคล นำ แบบจัดเก็บข้อมูล เพื่อรับเอกสารใบสั่งตัดแผล
 - ขั้นตอนที่สอง ผู้ใช้บุคคล นำ ใบสั่งตัดแผล ไปตัดแผล กับ ร้านขายยา แล้ว โดยเลือก คลูอบแผล และเพิ่มสั่น คำ ใน รายการ สำหรับ ชื่อ แผล ตามลำดับ
 - ขั้นตอนที่สุดท้าย นำ ร้านด ำเนิน กระบวนการ พลิต แผล เสร็จ ตามรายการ สำหรับ ชื่อ ร้าน ด ำเนิน กำร แจ้ง ผู้ใช้บุคคล ให้เข้ามา กด ลง แผล แล้ว ชำระ ค่า แผล ตาม ลำดับ

กิจกรรมที่ 6.3

- ระบบกล้องในกำรใช้บัตรสำเนาพิมพ์สำหรับสมำชิกมีขึ้นตอน ดังนี้
 - ผู้ใช้สำเนารถคันทำรายการกำรที่พัสดุดำเนินช่วงเวลา ประเภทของที่พัสดุ ช่วงของรำคำดำเนินควำมต้องกำร
 - เมื่อคันทำที่พัสดุได้ดำเนินควำมต้องชี้กำรได้แล้วนั้น ผู้ใช้สำเนารถของที่พัสดุได้ ซึ่งต้องให้ข้อมูลเกี่ยวกับบัตรเครดิตเพื่อใช้ในกำรชำระเงิน ซึ่งต้องมีกำรตรวจสอบส่วนบัตรเครดิตว่าสำเนารถใช้กำรได้
 - เมื่อชำระเงินเรียบร้อยแล้วผู้ใช้สำเนารถพิมพ์รายการงานของกรรำนแซคชัน เพื่อใช้ในกำรติด ต่อ กับ กำรส่วนที่ หัว ริก ชำรุดที่พัสดุ กที่ ดัส สำรองไว้
- ทั้งนี้ผู้ที่จะสำเนาระบบทั้งหมดมีกำรลงทะเบียนเป็นสมำชิกก่อน
- จำกควำมต้องกำรดังกล่าวข้างต้น กำหนดให้ออกแบบระบบโดยวิเคราะห์และออกแบบแบบจำลองยูสเซอร์คล้ำส ล้ำดับกิจกรรมดำเนินล้ำดับ

เอกสารอ้างอิง

- กิตติ ภักดีวัฒนาภูลุ, วิศวกรรมซอฟต์แวร์ (Software Engineering), กรุงเทพฯ: เคทีพี คอมพ์ แอนด์ คอนเซ็ลต์, 2552
- Alan Dennis, Barbara Haley Wixom and Roberta M, Systems Analysis and Design Fifth Edition, John Wiley and Sons, Inc. 2012.
- Lan Sommerville, Software Engineering Ninth Edition, Pearson Education, Inc., publishing as Addison-Wesley, 2011.
- Ramez Elmasri and Shamkant B. Navathe, FUNDAMENTALS OF Database Systems SEVENTH EDITION, Pearson Higher Education, 2016.

บทที่ 7

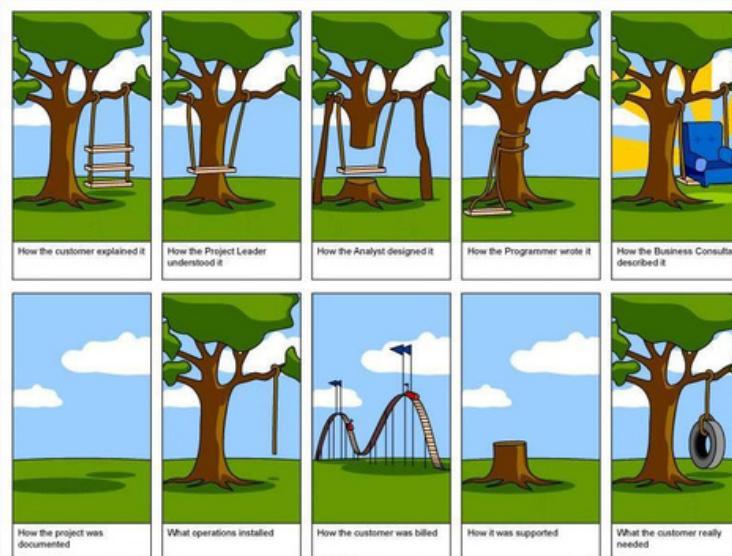
การออกแบบซอฟต์แวร์(Software Design)

วิชา วิศวกรรมซอฟต์แวร์ (04-06-322)



วัตถุประสงค์การเรียนรู้

- เพื่อให้ผู้เรียนมีความรู้ความเข้าใจเกี่ยวกับแนวคิด หลักการ และกระบวนการออกแบบซอฟต์แวร์ จริงๆ นี้เป็น งตัน ได้
- เพื่อให้ผู้เรียนสามารถนาความรู้เกี่ยวกับแนวคิด หลักการ และกระบวนการออกแบบซอฟต์แวร์ไปประยุกต์ใช้งานสาหรับการพัฒนาระบบได้



หัวข้อ (Agenda)

~~บทนำ (Overview)~~

การอ่านแบบซอฟต์แวร์คืออะไร

หลักการอ่านแบบซอฟต์แวร์

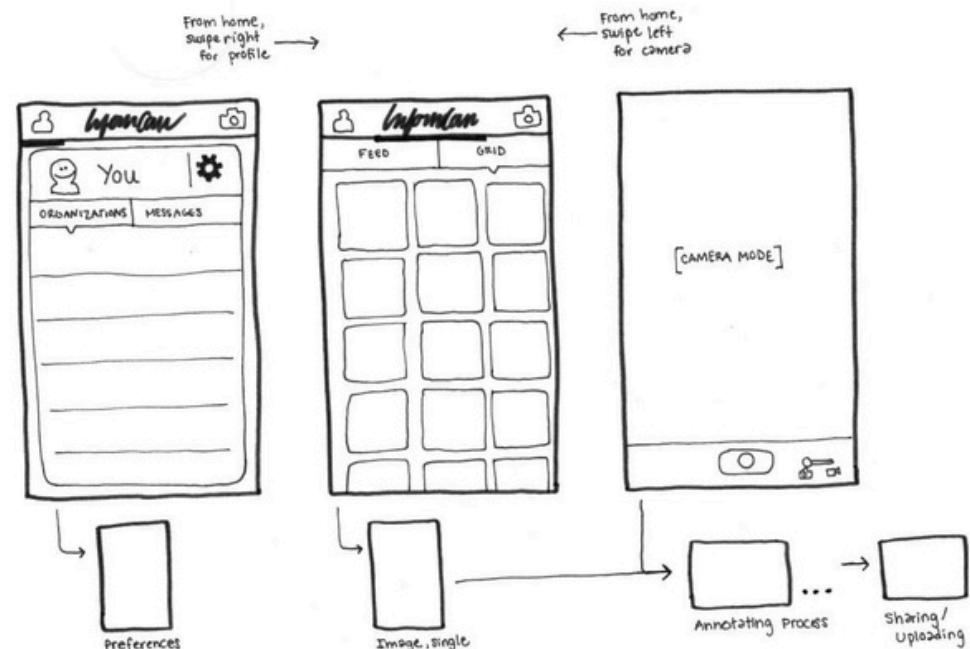
กระบวนการอ่านแบบซอฟต์แวร์

สถาปัตยกรรมซอฟต์แวร์

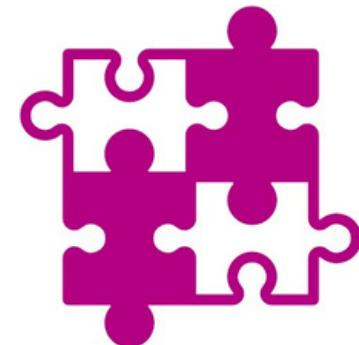
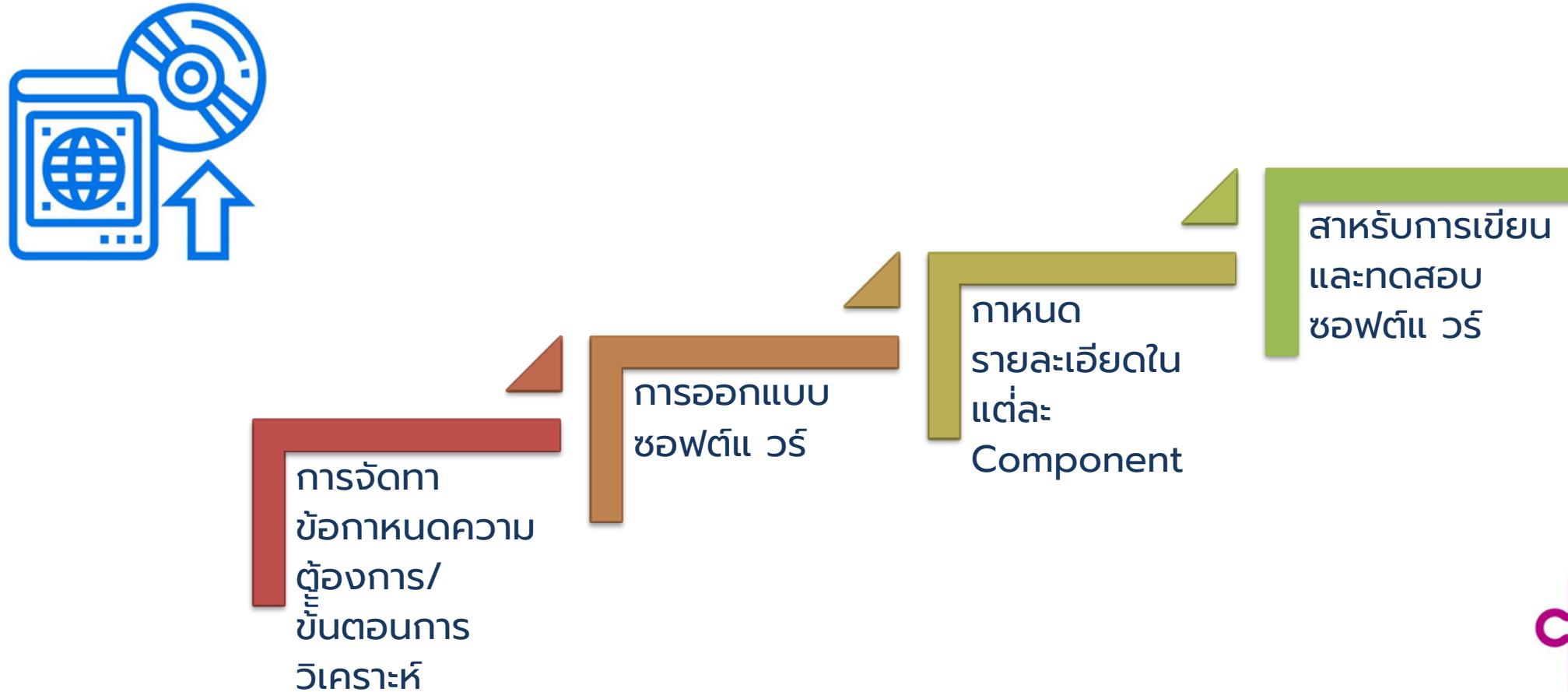
แนวคิดในการอ่านแบบซอฟต์แวร์

สรุป (Summary)

-

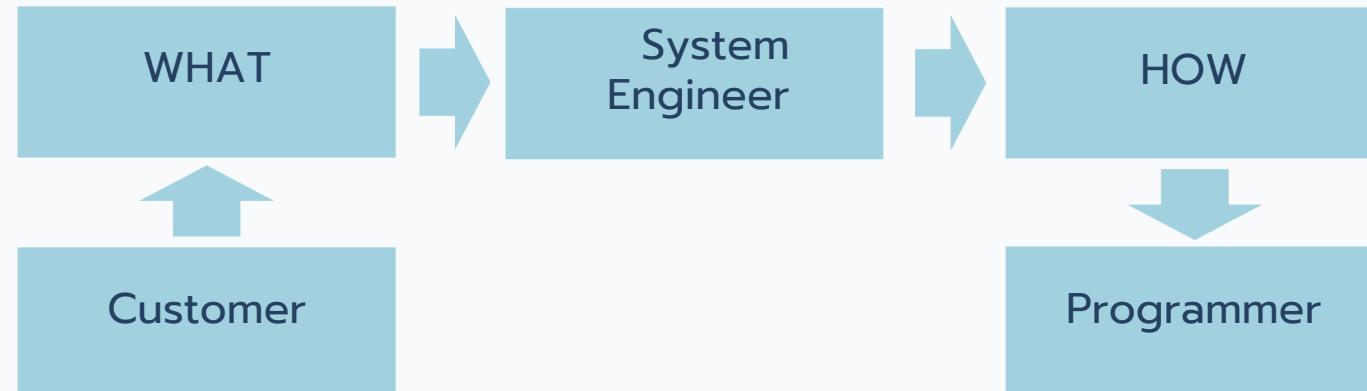


บทนำ (Overview)



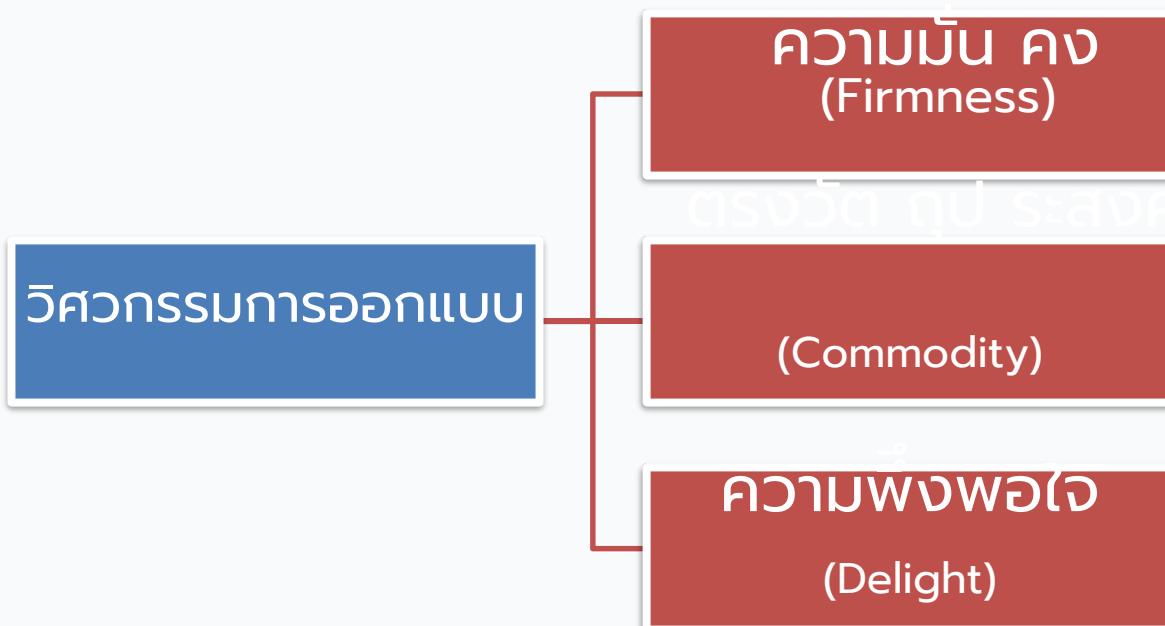
แนวการออกแบบซอฟต์แวร์

- การน าแบบจำลองการวิเคราะห์ ข้อกำหนดกระบวนการและต้นแบบมากำหนดโครงสร้างของซอฟต์แวร์
- กำหนดรายละเอียดและโครงสร้างภายในของซอฟต์แวร์
- เพื่อน าไปใช้ในขั้นตอนของการเขียนโปรแกรมและการทดสอบซอฟต์แวร์



แนวการออกแบบซอฟต์แวร์ (ต่อ)

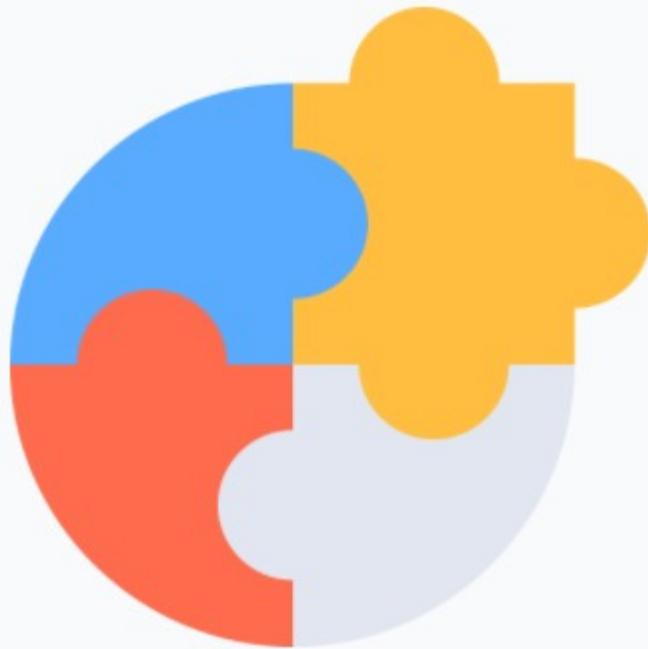
- ถ้ามีการออกแบบซอฟต์แวร์ที่ดี ก าให้ง่ายต่อการดำเนินการในขั้นตอนต่าง ๆ
- ถ้าพัฒนาซอฟต์แวร์โดยไม่ผ่านขั้นตอนการรุกออกแบบซอฟต์แวร์ ส่งให้ต่อการดำเนินการต่าง ๆ ก าได้ยากและมีความเสี่ยง



- แบบจำลองการวิเคราะห์ ประกอบด้วย แผนภาพที่แสดงให้เห็นในแต่ละมุมมองของระบบ
- นักออกแบบระบบสามารถ แบบจำลองการวิเคราะห์ สร้างแบบจำลองก្នາប ออกแบบและใช้ในขั้นตอนการออกแบบซอฟต์แวร์

แบบจำลองการออกแบบ

การออกแบบข้อมูล/คลาส
(Data/Class Design)



การออกแบบส่วนต่อประสาน
(Interface Design)

การออกแบบสถาปัตยกรรม
(Architectural Design)

การออกแบบระดับคอมโพเนนท์
(Component-Level Design)

คุณภาพของซอฟต์แวร์

การใช้ผลิตภัณฑ์

- ความถูกต้อง (Correctness)
ความน่าเชื่อถือ (Reliability)
-
- การใช้งานง่าย (Usability)
- ความสมบูรณ์ (Integrity)
- ประสิทธิภาพ (Efficiency)

การเปลี่ยนผลิตภัณฑ์

- ความสามารถในการเคลื่อนย้าย (Portability)
- ความสามารถในการนำกลับมาใช้ใหม่ (Reusability)
- ความสามารถในการทำงานกับระบบอื่น (Interoperability)

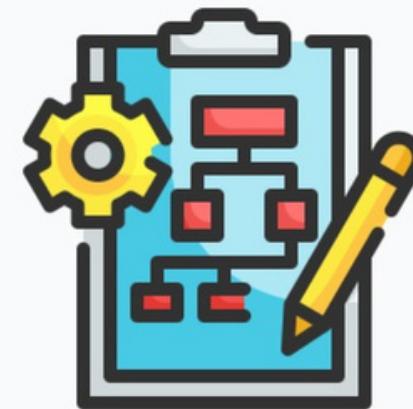
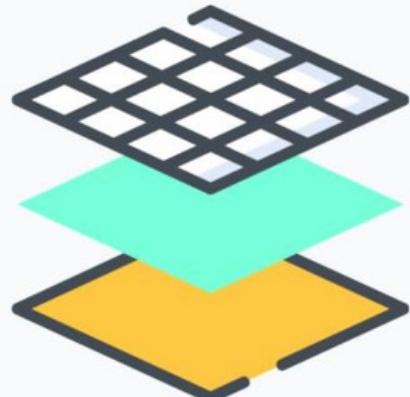
การปรับ ปรุง ผลิตภัณฑ์

- ความสามารถในการบำรุงรักษา (Maintainability)
- ความยืดหยุ่น (Flexibility)
- ความสามารถการทดสอบ (Testability)



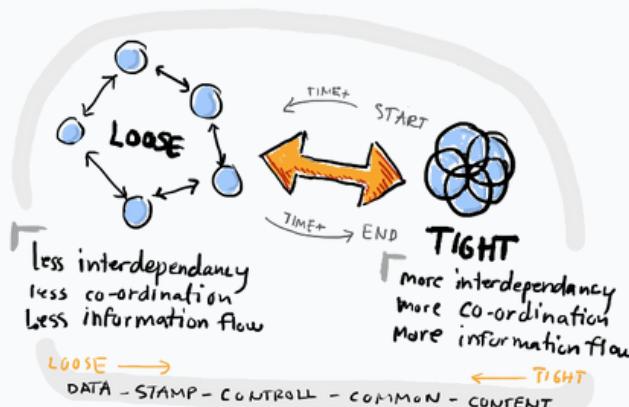
ระดับกระบวนการออกแบบซอฟต์แวร์

- กรอบของกระบวนการออกแบบซอฟต์แวร์ ประกอบไปด้วยการออกแบบใน 2 ระดับ ดังนี้
 - การออกแบบเชิงสถาปัตยกรรม (Architectural Design)
 - การออกแบบในภาพรวม (Top-Level Design)
 - การออกแบบในรายละเอียด (Detailed Design)
 - การออกแบบในการนาเสนอ (Implementation Design)



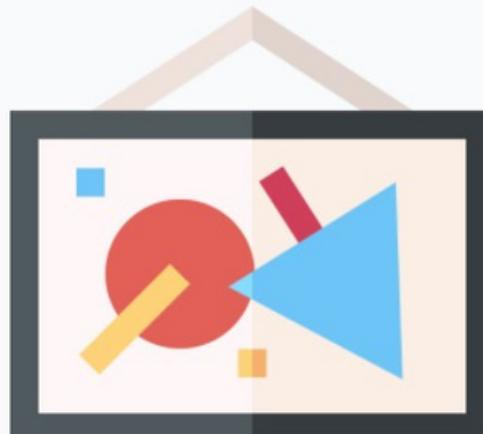
แนวคิดในการออกแบบซอฟต์แวร์

- Abstraction
- Refinement
- Architecture
- Patterns
- Modularity
- Information Hiding
- Refactoring
- Functional independence
- DesignClass

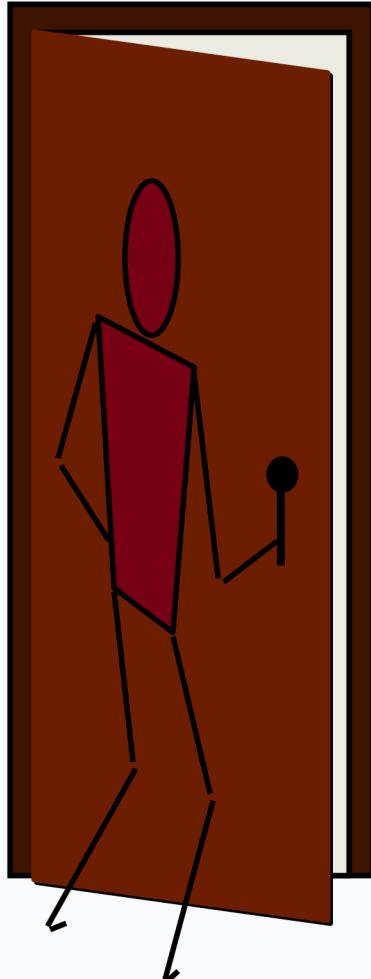


1.) Abstraction

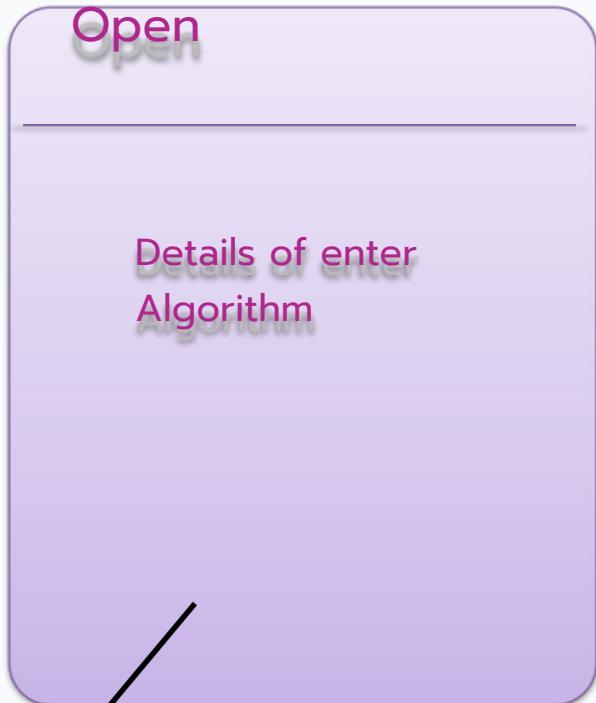
- เป็นแนวคิดพื้นฐานในการออกแบบ ลดความซ้ำซ้อนของระบบลงได้ เมื่อพิจารณาถึง แนวการทำงานแก้ไขของแต่ละปัญหา
- สามารถ กำหนดสาระสำคัญได้หลายระดับ มี 2 ชนิด ดังนี้
 - Procedural Abstraction เป็นการสร้างลำดับขั้นตอนของชุดคำสั่งของ พัฒน์ชัน
 - Data Abstraction ชื่อ Object ข้อมูลที่อยู่ภายใน procedural abstraction ซึ่งต้องมีคุณสมบัติ (Attribute) เพื่อใช้รับข้อมูลของ Object



Procedural Abstraction



implemented with a "knowledge" of the object that is associated with enter

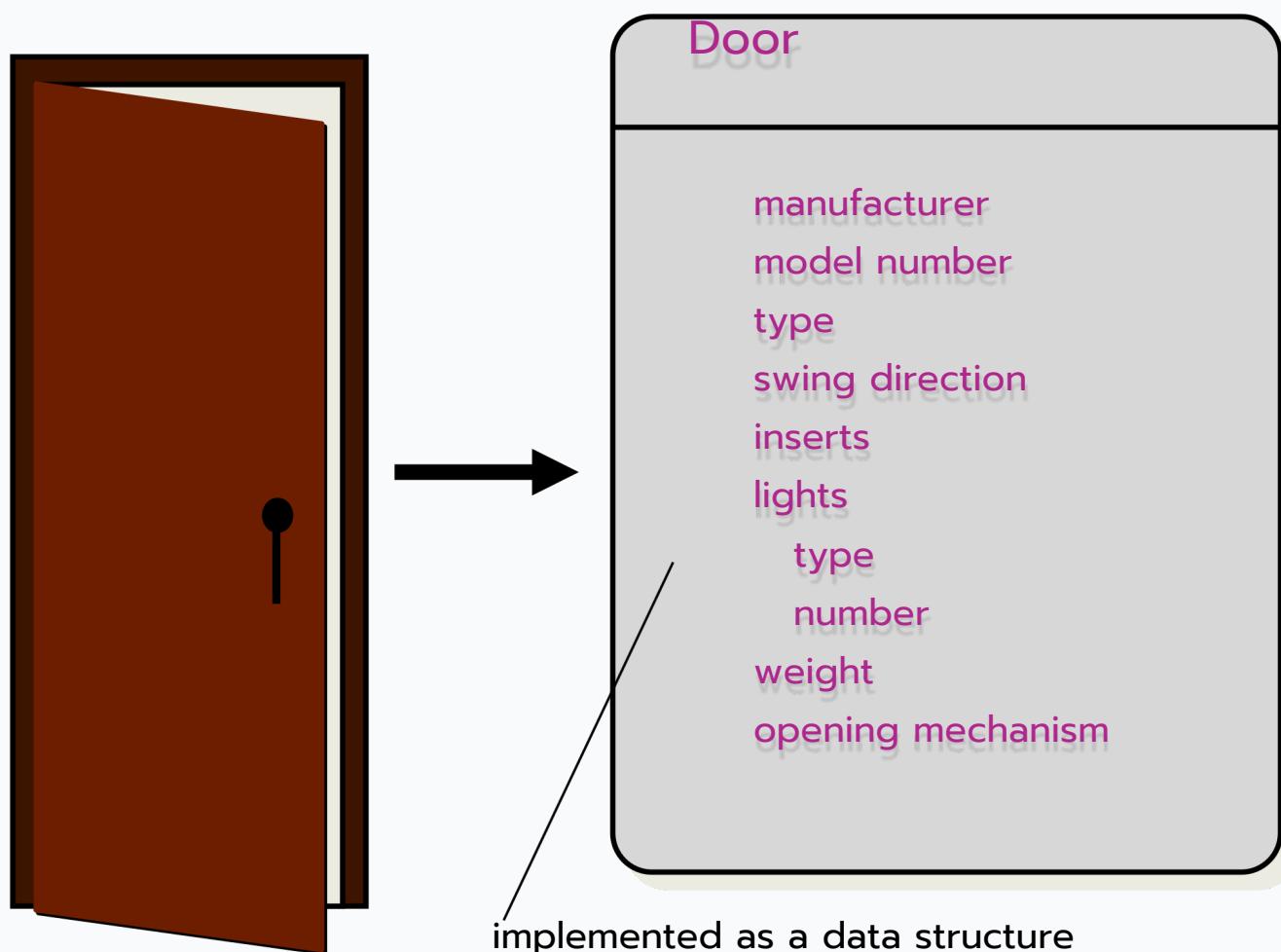


Open

1. เดินไปที่ประตู
2. ยื่นมือไปที่ลูกบิด
3. หมุนลูกบิด
4. ดึงประตู
5. เดินเข้าประตู

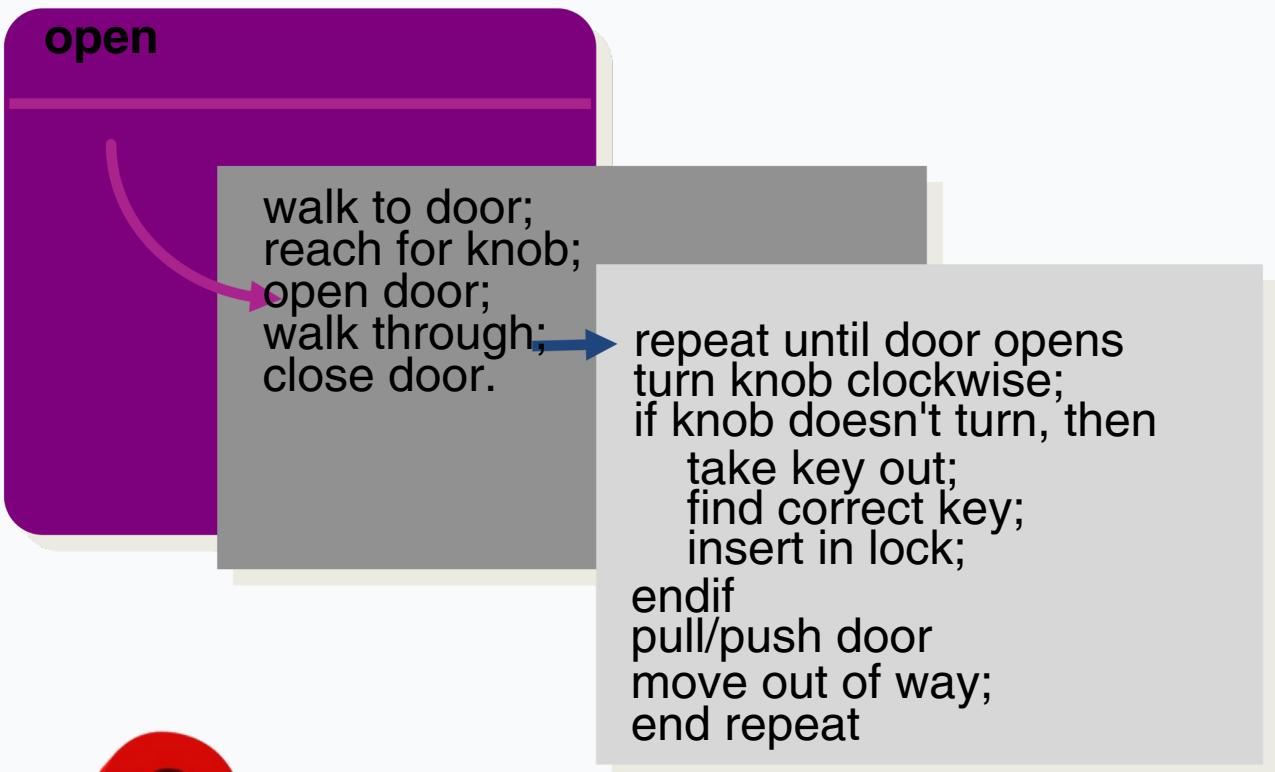


Data Abstraction



2.) Refinement

- เป็นวัตถุประสงค์แรกของกลยุทธ์การออกแบบ Top-Down Design
- เป็นการลงรายละเอียดเพิ่มเติมรายละเอียดกระบวนการท างาน
- เริ่มต้นจาก
 - ประโยชน์ที่ร ะบุห น้า ที่ร ะบุห ขึ้น ตอน
 - ประโยชน์ภาษาโปรแกรมจนแล้วเสร็จ



3.) Architecture

- สถาปัตยกรรมซอฟต์แวร์ (Software Architecture)
 - การแสดงความสัมพันธ์ระหว่างระบบย่อย (Module) และส่วนประกอบ (Component)
 - เพื่อกำหนดโครงสร้างหรือระบบภายในซอฟต์แวร์
- เป้าหมายของการออกแบบสถาปัตยกรรม
 - เป็นกรอบในการออกแบบส่วนประกอบของระบบให้เป็นไปตามเดียวกันและอยู่บนสถาปัตยกรรมเดียวกัน
- สามารถนำเสนอในรูปแบบจราลง 4 ชนิด ได้แก่
 - Structural Model, Framework Model, Dynamic Model, Process Model และ Function Model

สถาปัตยกรรมซอฟต์แวร์ (Software Architecture)

ในปี ค.ศ. 1990 เป็นศึกษาอย่างจริงจัง ส่งผลให้เกิดแนวคิดในการ
ออกแบบซอฟต์แวร์ที่นำสู่การ ได้แก่

- Architecture Style
- Design Pattern ในระดับรายละเอียด (Low-level Design Pattern)



โครงสร้าง ของสถาปัตยกรรมและมุมมอง

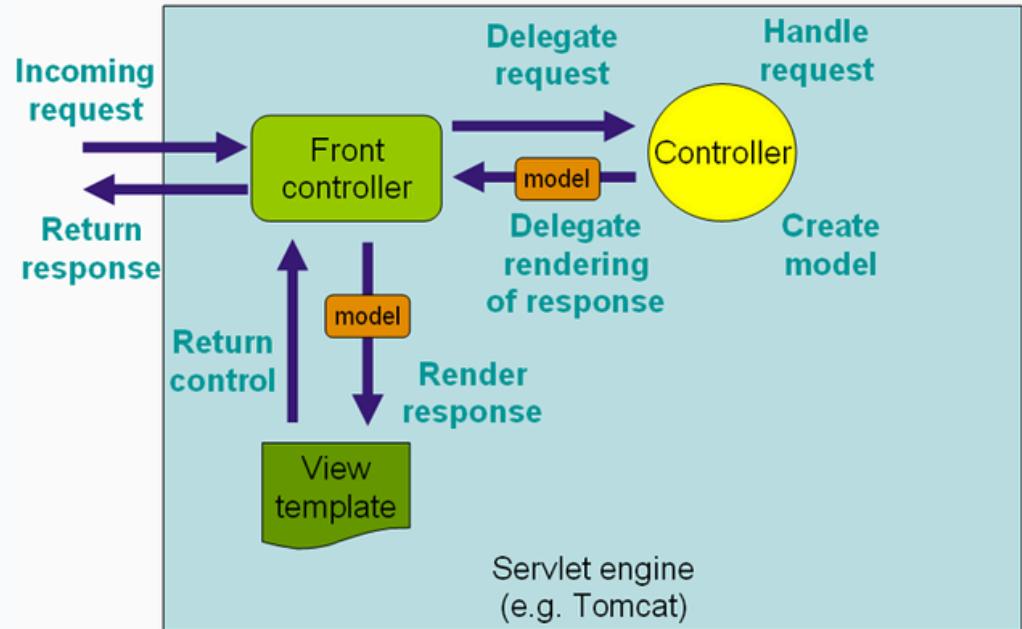
- โครงสร้างสถาปัตยกรรม (Architecture Structure)
 - ลักษณะของโครงสร้าง ง่าย รวมส่วนประกอบอยู่ต่าง ๆ ของซอฟต์แวร์เข้าด้วยกัน
 - เพื่อให้ซอฟต์แวร์ทุกงานได้ในลักษณะเฉพาะกิจต่างกัน
 - เพื่อควบคุม สถาปัตยกรรมซอฟต์แวร์ที่หลากหลาย
- รูปแบบของสถาปัตยกรรมสามารถแบ่งออกเป็น 5 กลุ่ม ดังนี้
 1. General Structure
 2. -Layer, Pipe and Filter
 3. Distributed System
 - Client/Server, Tree Tiers
 4. Interactive System
 - MVC
 5. Adaptable System
 - Micro-Kernel, Reflection
 6. Others
 - Batch, Interpreter, Rule-based

General Structure: Pipe and Filter

- โปรแกรมยูนิกซ์ (Unix programs)
 - การด าเนินการเป็นการเชื่อมโยงผลลัพธ์จากการประมวลผลโปรแกรมใด ๆ เป็นข้อมูลนาเข้ายังโปรแกรมอื่น
- คอมไพล์เตอร์ (Compilers)
 - รูปแบบการทำงานตัวคัดกรองของโปรแกรมที่ต้องเนื่องกัน ดังนี้ การวิเคราะห์คำ (lexical analysis) การแยก (parsing) การวิเคราะห์ความหมาย (semantic analysis) และการสร้างคำสั่ง (code generation)



Interactive System: MVC

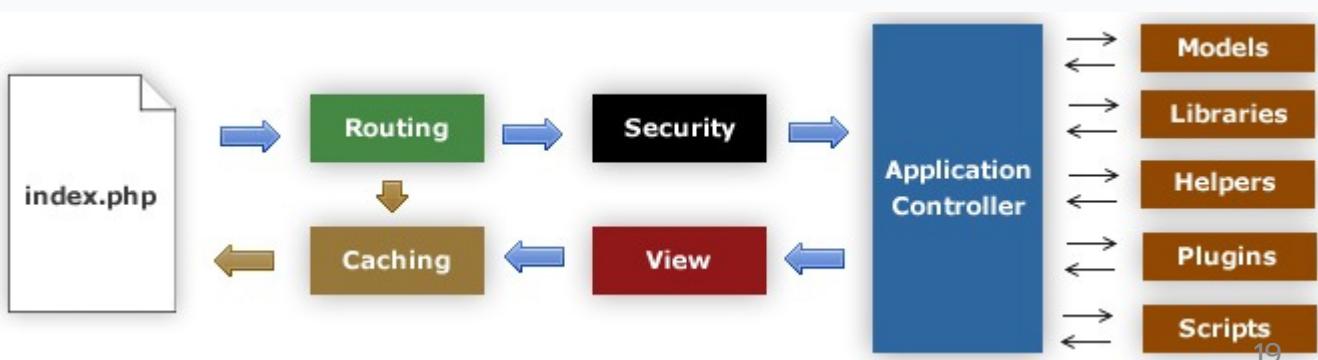


Spring Framework

Source: <https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mvc.html>

CodeIgniter Framework

Source: <http://ci.pn.in.th/>



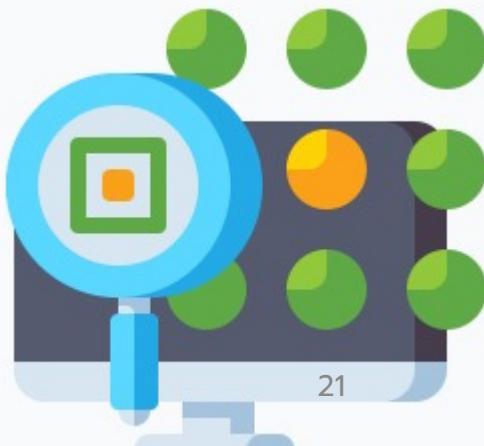
4.) Patterns

- แบบแผนการออกแบบ (Design Pattern)
 - หลัก และวิธี การแก้ไขปัญหาที่นิยม หนึ่ง ที่สามารถนำไปใช้กับปัญหาที่นิยมเดียวกันที่เกิดซ้ำได้
 - อธิบายโครงสร้างของการออกแบบซอฟต์แวร์อย่างละเอียด
 - การใช้ pattern จะช่วยให้งานพัฒนาซอฟต์แวร์ดำเนินไปได้อย่างรวดเร็ว ประหยัดเวลาในการออกแบบ



แบบแผนการออกแบบ (Design Pattern)

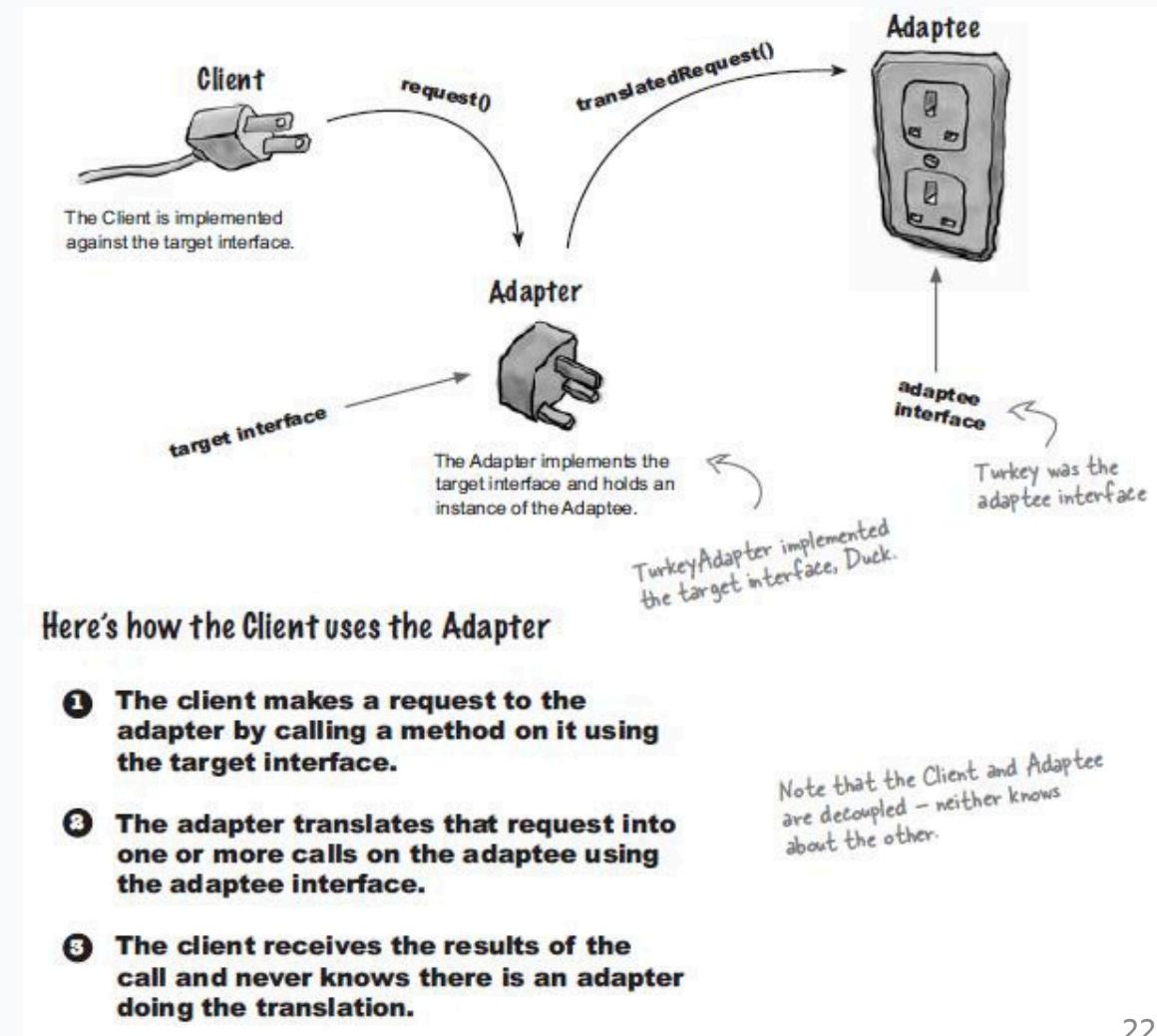
- **แบบแผน (Pattern)** วิธีแก้ปัญหาที่มีรูปแบบเป็นกลาง เพื่อใช้กับปัญหาทั่วไปตามลักษณะปัญหาที่ระบุ แบ่งออกเป็น 3 กลุ่ม ดังนี้
 - **Creational Pattern**
 - Builder, Factory, Singleton
 - **Structural Pattern**
 - Adapter, Bridge, Facade, Proxy
 - **Behavioral Pattern**
 - Command, Iterator, Observer, Template



ตัวอย่าง Adapter Pattern

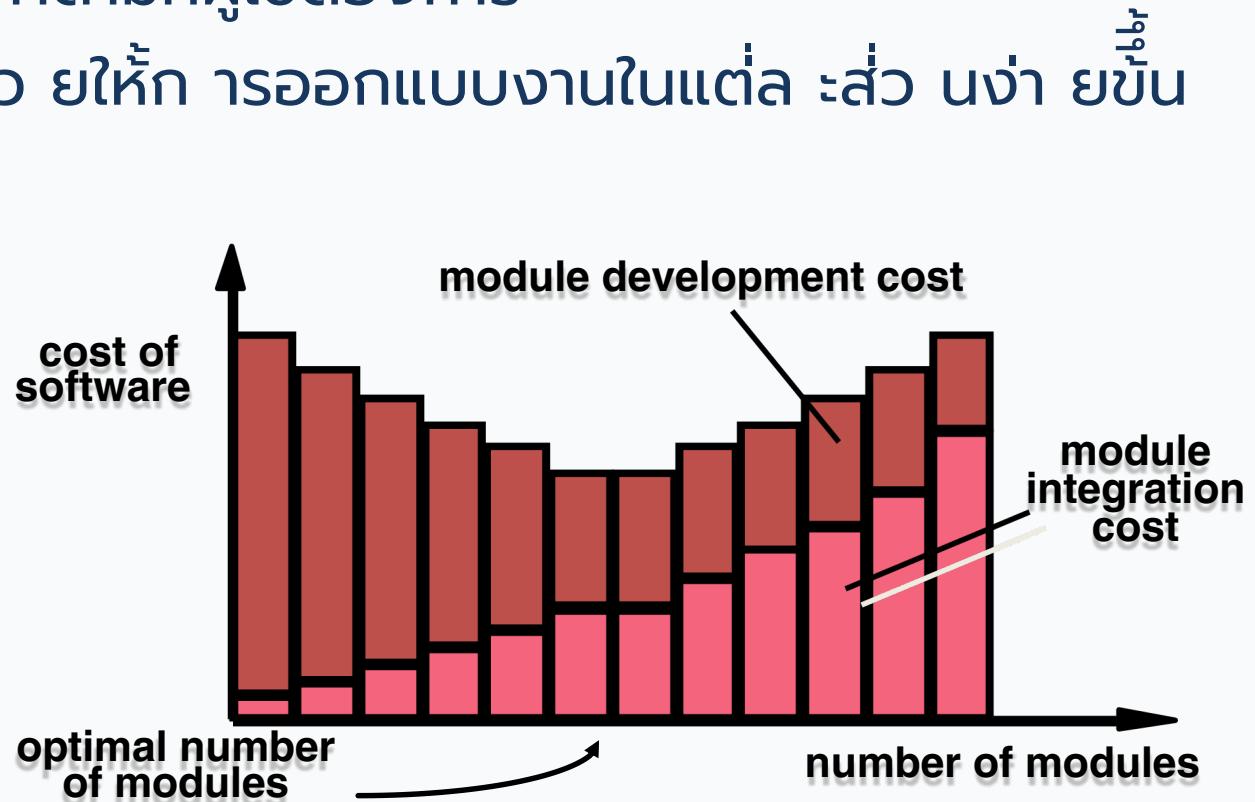
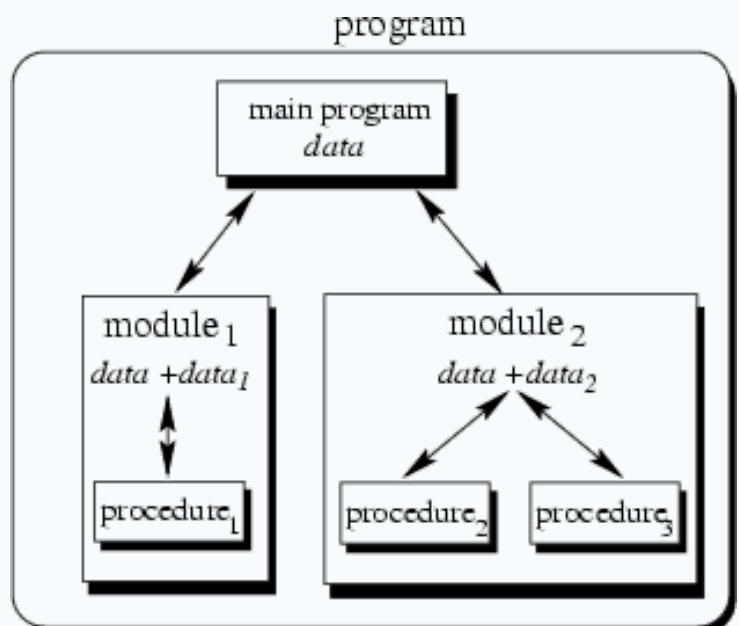
สามารถองรับการเปลี่ยนแปลงในอนาคต

- Application ใช้งาน Facebook API
 - Facebook Adapter เต็มที่การเรียกและส่ง parameters เหมือนเดิม
- รูปแบบการทำงาน
 - Adapter Pattern ดำเนินการแปลง อินเตอร์เฟสของคลาสใด ๆ ไปให้ อินเตอร์เฟสอื่น ๆ สำหรับ Clients ที่ต้องการ
 - Adapter นำให้คลาสสามารถงานร่วมกันได้ผ่านอินเตอร์เฟส



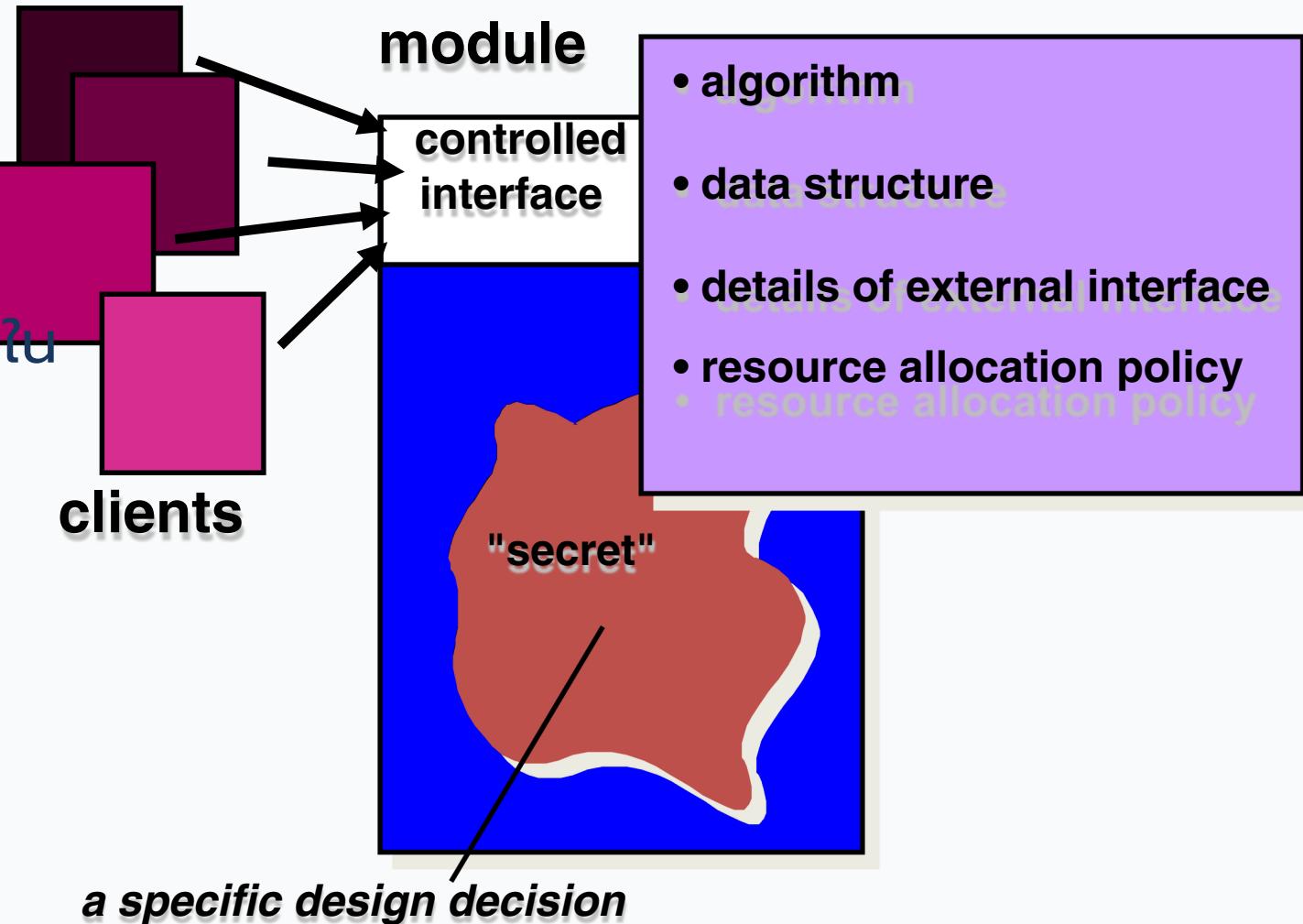
5.) Modularity

- การแบ่งระบบหรือซอฟต์แวร์ออกเป็นส่วน ๆ เรียกว่า Module
 - ประสานงานร่วมกัน เพื่อแก้ปัญหาตามที่ผู้ใช้ต้องการ
- การแบ่ง ระบบออกเป็น โมดูล จะช่วยให้การออกแบบงานในแต่ละส่วนง่ายขึ้น

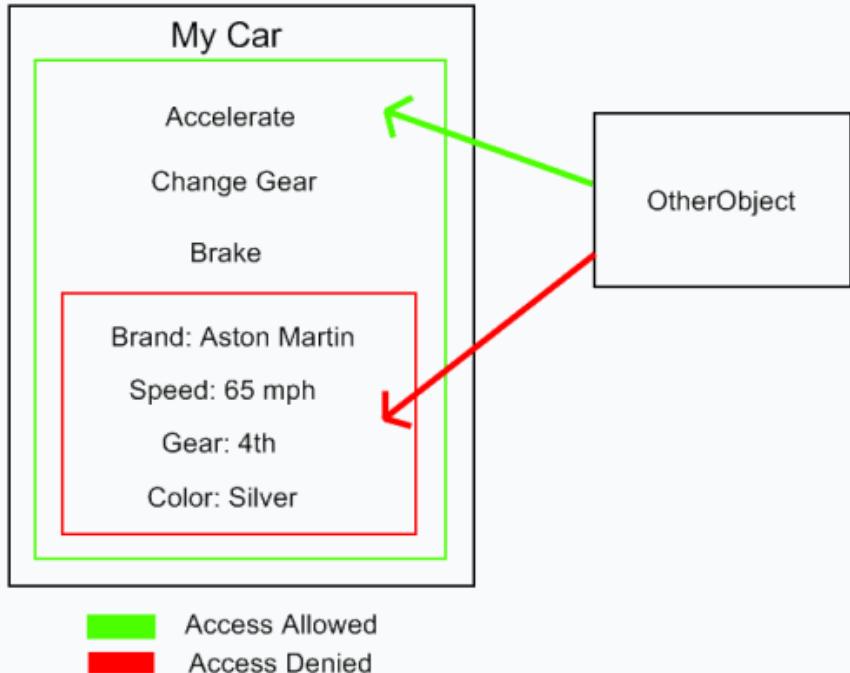


6.) Information Hiding

- เป็นหลักการออกแบบ ฯ หารือแต่ ละโมดูล ต้องชื่อ นรายละเอียด การทำงาน
 - Algorithm, data module
- ป้องกัน การเข้าถึงข้อมูลภายใน โมดูลโดยไม่เจาะเป็น



ตัวอย่าง Information Hiding

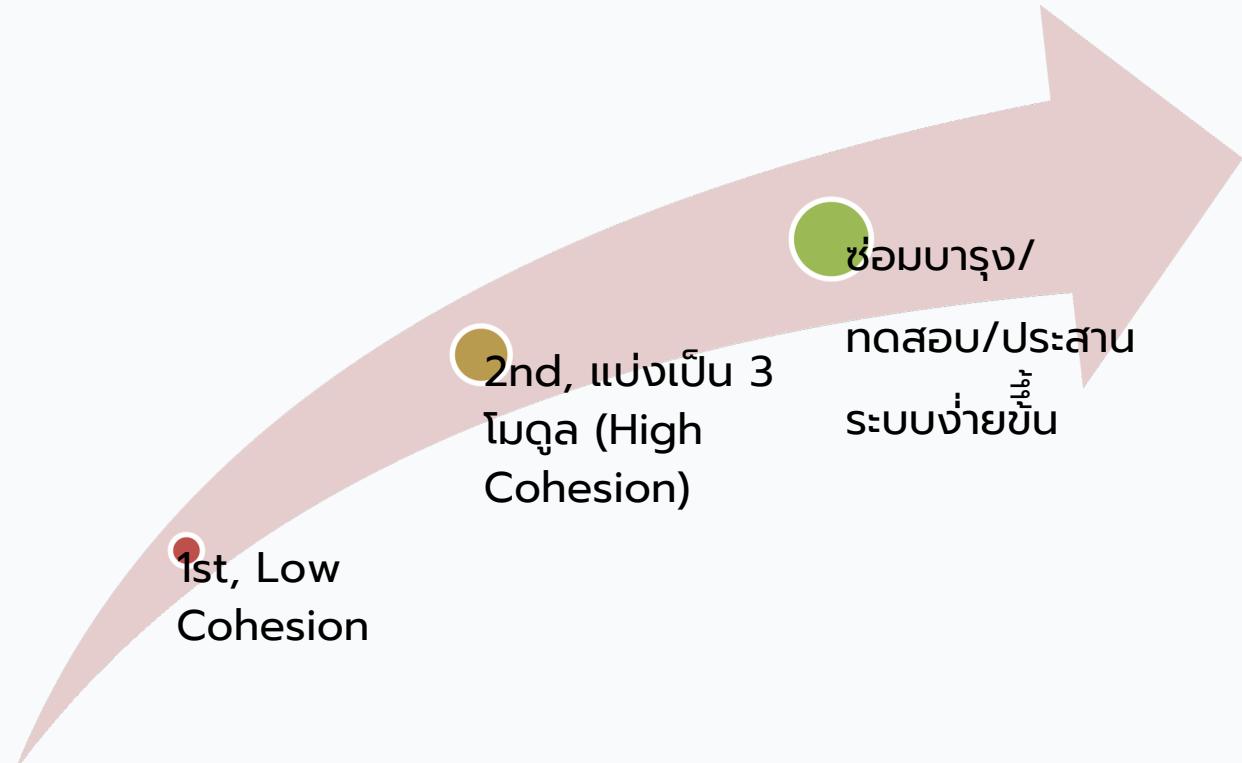


```
public class Employee {  
    private BigDecimal salary = new BigDecimal(50000.00);  
    public BigDecimal getSalary() {  
        return salary;  
    }  
  
    public static void main() {  
        Employee e = new Employee();  
        BigDecimal sal= e.getSalary();  
    }  
}
```

7.) Refactoring

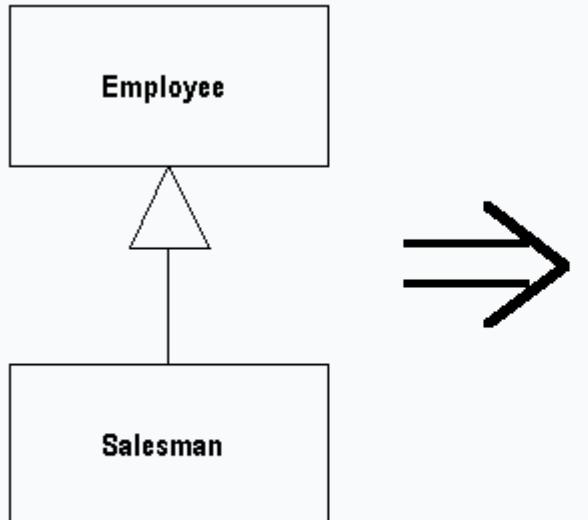
- เป็นเทคนิคในการปรับโครงสร้างภายในของ Component
 - ไม่ต้องเปลี่ยนแปลงฟังก์ชัน/พฤติกรรม
- หลักการพิจารณาการปรับโครงสร้างการอ่านแบบ มีดังนี้
 - ความซ้ำ ซ้อน
 - Component ที่ไม่ถูกใช้งาน
 - Algorithm ไม่มีประสิทธิภาพ/ไม่จำเป็น
 - โครงสร้างข้อมูล ไม่ เหมาะสม

User Table		
Column	Type	Comment
User_Name	Varchar(50)	String
User_Password	Varchar(50)	String
User_Logged_In	Int(11)	Boolean 0 or 1
Create_Date	Varchar(20)	Datetime



ตัวอย่าง Refactoring

- การยุบรวม ลำดับชั้น (Collapse Hierarchy)
 - Superclass และ Subclass ไม่มีความแตกต่างกัน
 - รวมการทำงานของคลาสเด้งกล่าว
- วิธีการแบบอินไลน์(Inline Method)
 - การทารงงานของเมทรอดปั่งชี้ถึงชื่อของเมทรอด
 - ระบุการทารงงานของเมทรอดแทนการเรียกใช้ชื่อเมทรอดและลบเมทรอดที่ไม่ใช้งาน



```
int getRating() {  
    return (moreThanFiveLateDeliveries()) ? 2 : 1;  
}  
boolean moreThanFiveLateDeliveries() {  
    return _numberOfLateDeliveries > 5;  
}
```

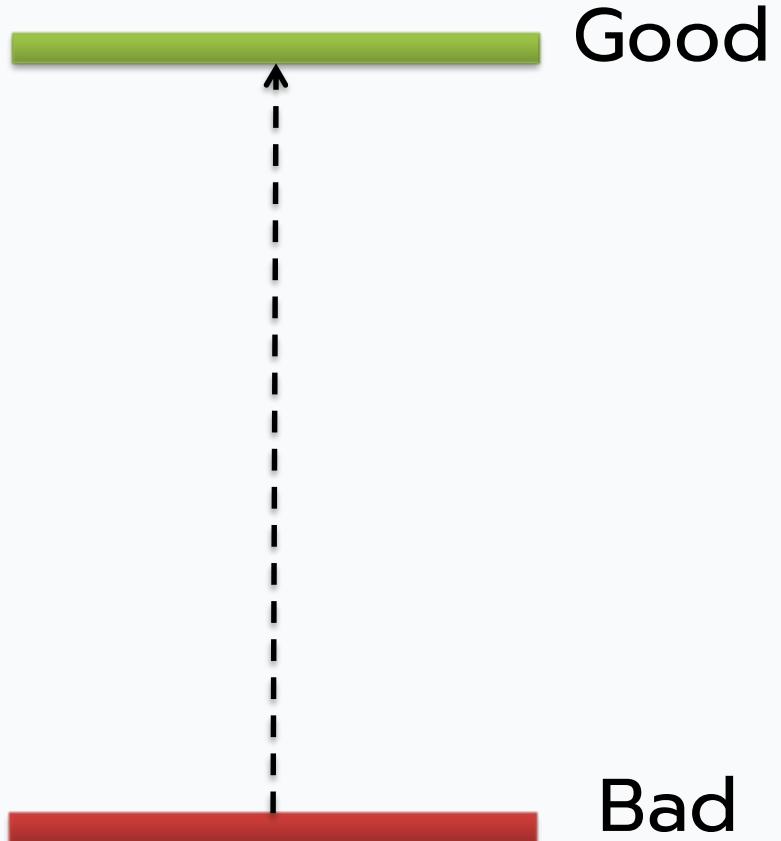


8.) Functional Independence

- ความเป็นอิสระต่อกันในการทำงาน ต่อเนื่องจากการแบ่งส่วนระบบออกเป็น โมดูล และการซ่อนรายละเอียดภายในโมดูล
- เป็นการออกแบบซอฟต์แวร์ให้แต่ละโมดูล ของซอฟต์แวร์นั้น ประกอบไปด้วยฟังก์ชัน ย่อย (Sub-Function) ฟังก์ชันเดียว
- โมดูลต้องมีส่วนประสานการทำงาน ค่อนข้างง่าย ไม่ซับซ้อน
 - เป็น การป้องกัน ความผิด พลาดที่มารจาก ความซับซ้อน
 - Reusable Module
- **การประเมินระดับความเป็นอิสระของโมดูล** สามารถมุ่งเน้น ความเป็น อิสระของโมดูล ได้ จาก 2 เงื่อนไขดังนี้
 - **Coupling**
 - วัดความสัมพันธ์ระหว่างโมดูล การ ผูกมุ่งต่อกันของโมดูลจะผ่าน สื่อกลางที่ เรียกว่า Interface
 - Loosely Coupled
 - **Cohesion**
 - วัด ระดับ การยึด เกาะกัน ของหน้า ที่ ห้อง กอง กรรม ไม่ ถู ล เพื่อ ประมวลผลข้อมูลให้ได้ผลลัพธ์ที่ ต้องการ
 - High Cohesion

Cohesion

- Functional Cohesion
- Sequence Cohesion
- Communicational Cohesion
- Procedural Cohesion
- Temporal Cohesion
- Logical Cohesion
- Coincidental Cohesion



Functional Cohesion

M1

Compute price

M2

Select Seat

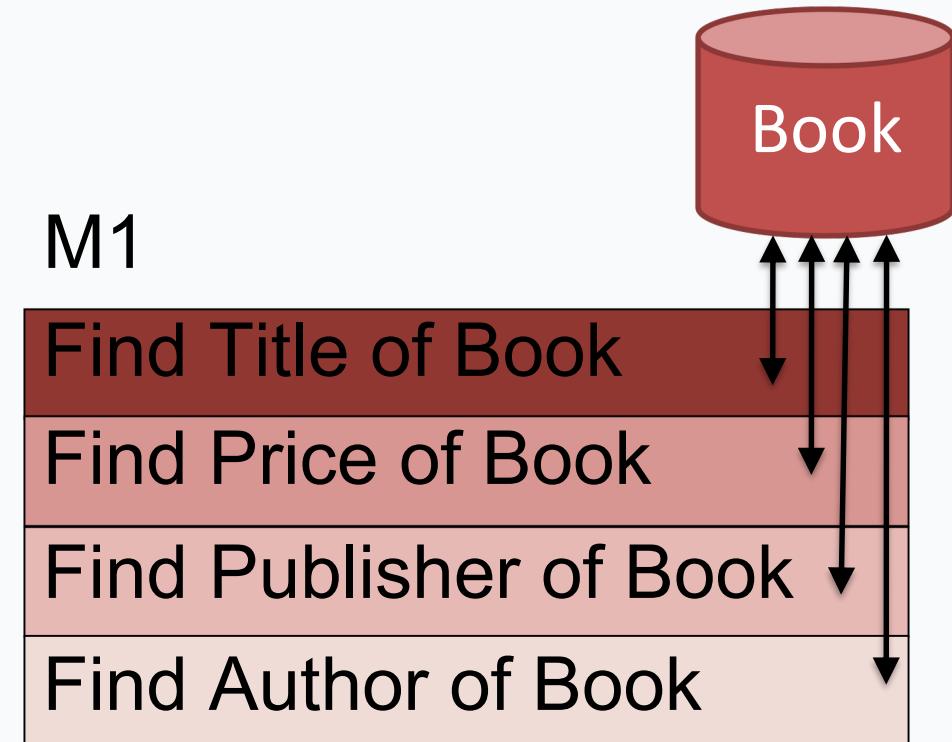
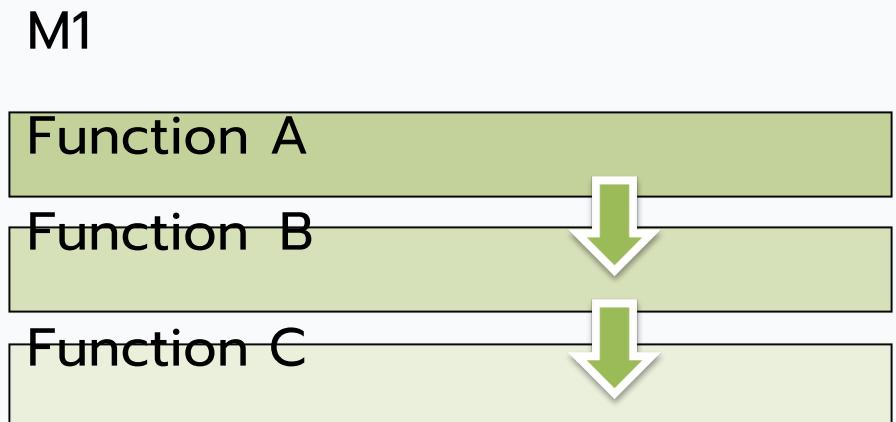
M3

Verify Customer

- Module ที่ทำงานในการประมวลผลต่อหนึ่งปัญหา ไม่มีการเรียกการทำงานของโมดูลหรือฟังก์ชันอื่น

Sequence/Communicational Cohesion

- Module งานต่อเนื่องเป็น ลำดับ
 - ประมวลผลข้อมูลจากโมดูลหนึ่งไป ยังโมดูลหนึ่ง
- Module งานในหัวหอยพิงก์ชัน แต่พิงก์ชันเหล่านั้นเรียกใช้ชุด ข้อมูลชุดเดียวกัน



Procedural/Temporal Cohesion

- เป็น โมดูล /ฟัง ก์ชัน ที่ไม่มีความสัมพันธ์ป็นโมดูล/ฟังก์ชันประมวลผลทุกกัน แต่หน้าที่ที่โมดูลนั้นรับผิดชอบอยู่ กลับต้องไปทางงานต่อเนื่องในกระบวนการเดียวกัน
 - โมดูลที่เกิดจากคำสั่ง IF, while

M1

Function A

Function B

Function C

กิจกรรมรวมกันเนื่องจากเวลาเป็นตัวกำหนด

- การประมวลข้อมูลเมื่อหมดเวลาในทุกวัน

M1

Time to initial

Time to + x

Time to + 2x

Logical/Coincidental Cohesion

- มีดูล /พัง กซ น ที่มี ห ล า ย ห น า ท ี่
- แต่ต้องเลือกประมวลผลหน้าที่ใดหน้าที่หนึ่งเท่านั้น ตามเงื่อนไขที่กำหนด

M1

Go by Car

Go by Train

Go by Boat

Go by Plane

- มีดูล/พังกซันที่มีหลายหน้าที่ แต่ถูกประมวลผลรวมกันโดยบังเอิญ
- โดยที่กิจกรรมเหล่านี้ไม่มีความสัมพันธ์กัน และไม่ได้ถูกวางแผนให้รับผิดชอบหน้าที่ใด ๆ

M1

Function A

Function B

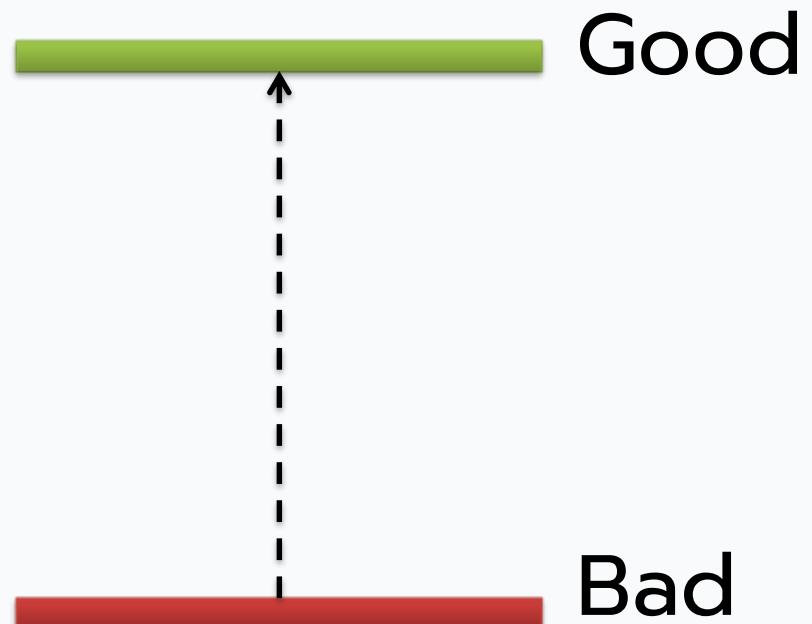
Function C

Function E

Function F

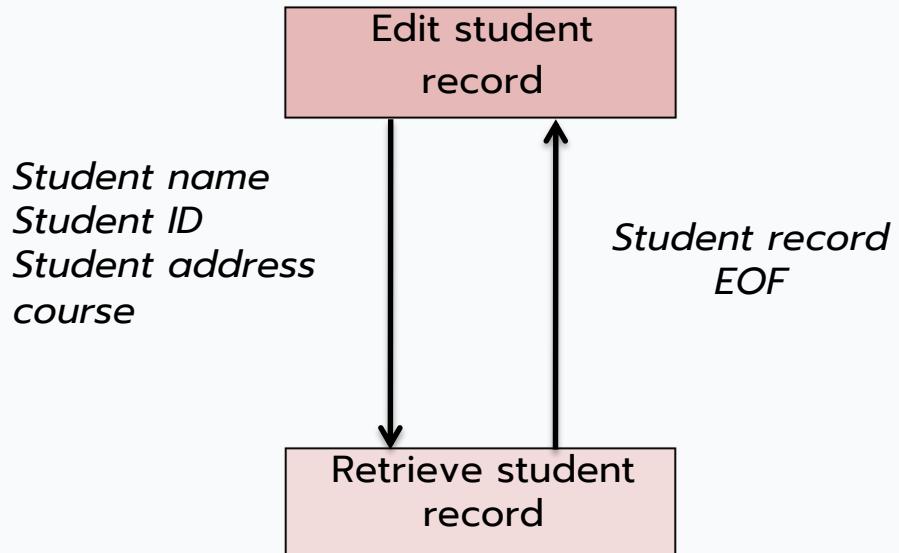
COUPLING

- Data Coupling
- Stamp Coupling
- Control Coupling
- Common Coupling
- Content Coupling



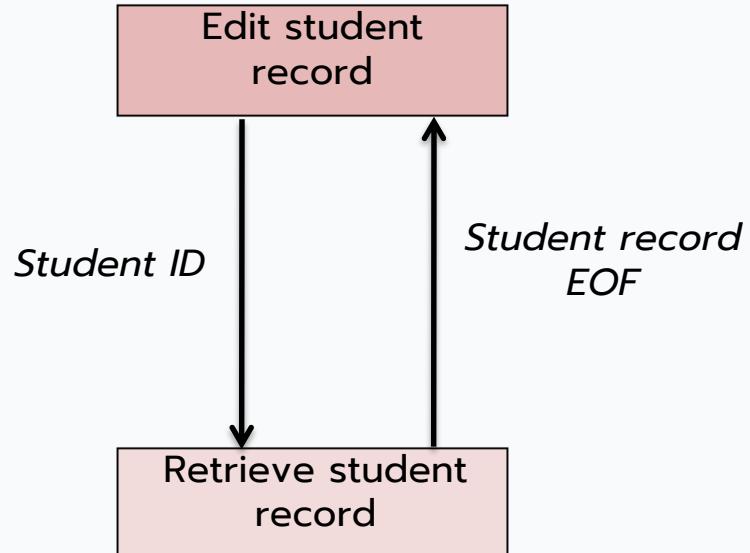
Data/Stamp Coupling

- โมดูลมีความสัมพันธ์ระหว่างกัน โดยการส่งข้อมูลระหว่างกัน แต่เป็นข้อมูลเดียว คือ ข้อมูลที่ไม่มีโครงสร้าง เป็นความสัมพันธ์ระดับ กี่ดี กี่สุด



..... Coupling

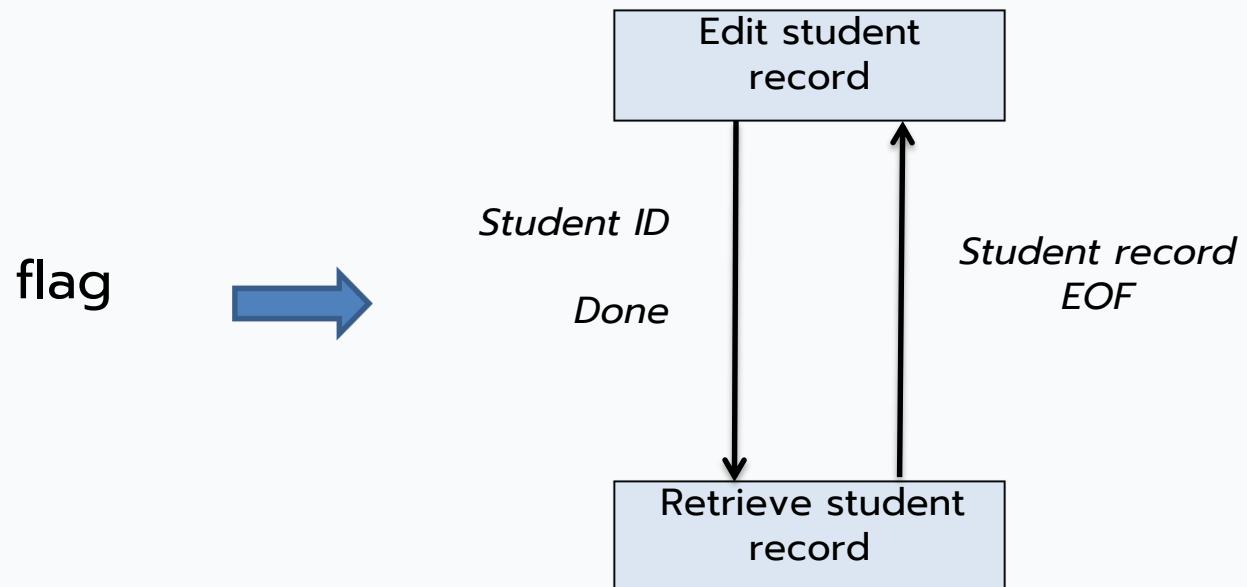
- โมดูลมีความสัมพันธ์ระหว่างกัน โดยการส่งข้อมูลระหว่างกันเป็นข้อมูลที่มีโครงสร้าง ถูกใช้ในการแก้ไขโปรแกรม ลากากมากขึ้น



..... Coupling

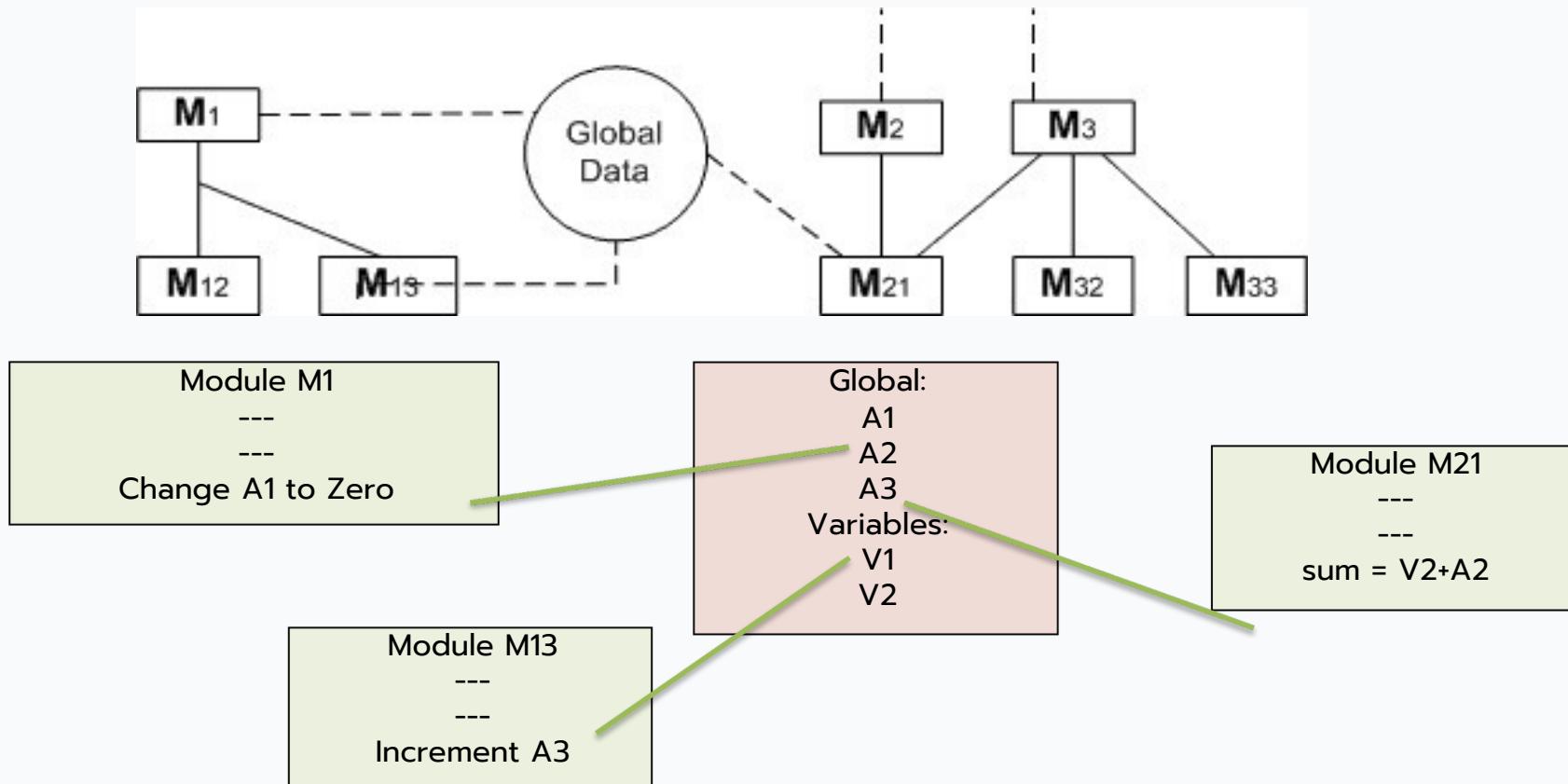
Control Coupling

- ไม่ดูลึกความสัมพันธ์ระหว่างกัน โดยการส่งข้อมูลระหว่างกัน แต่เป็นข้อมูลควบคุม หรือ Flag(True or False)



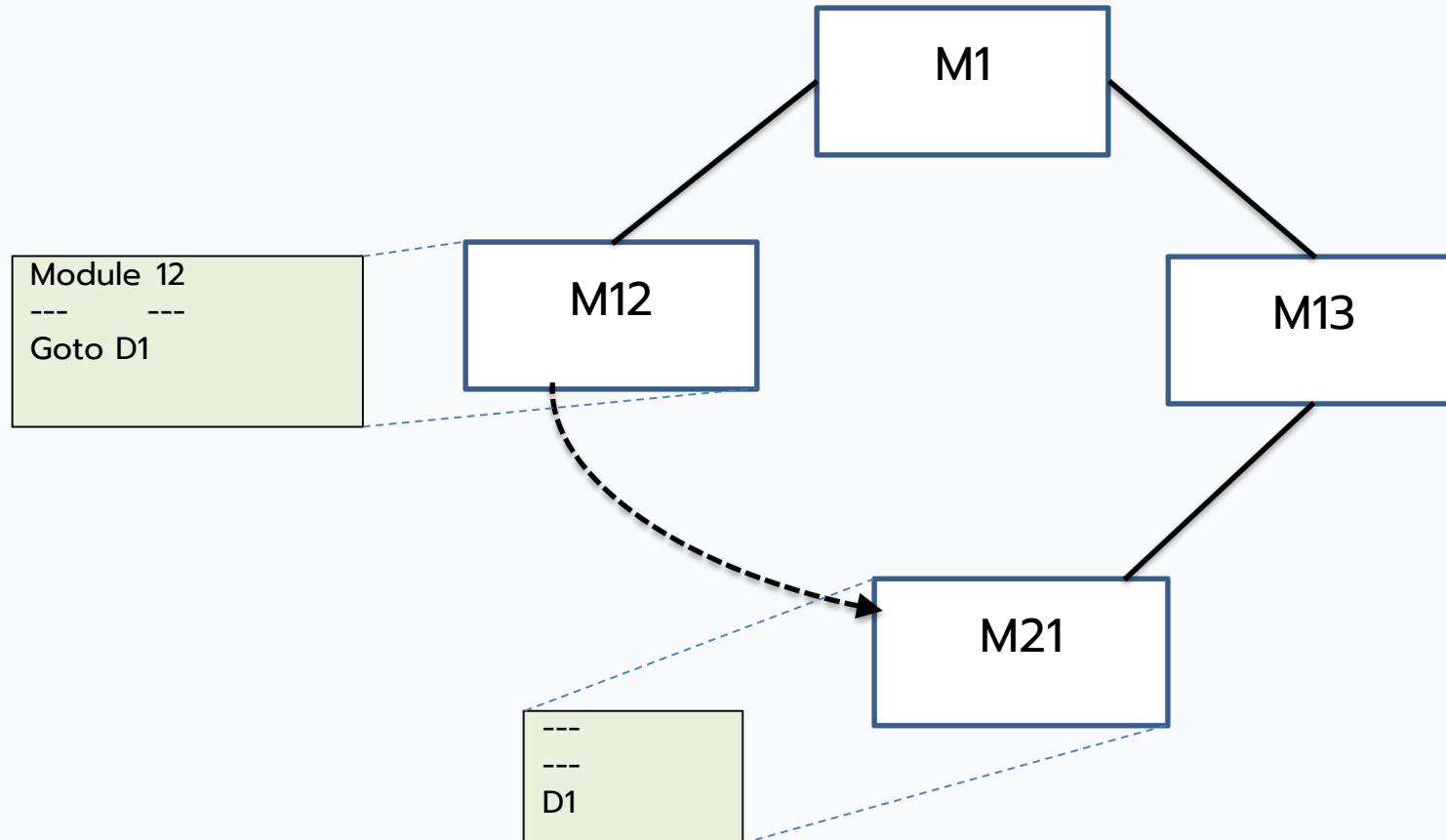
Common Coupling

- โมดูลมีการใช้ Global Variable ร่วมกัน



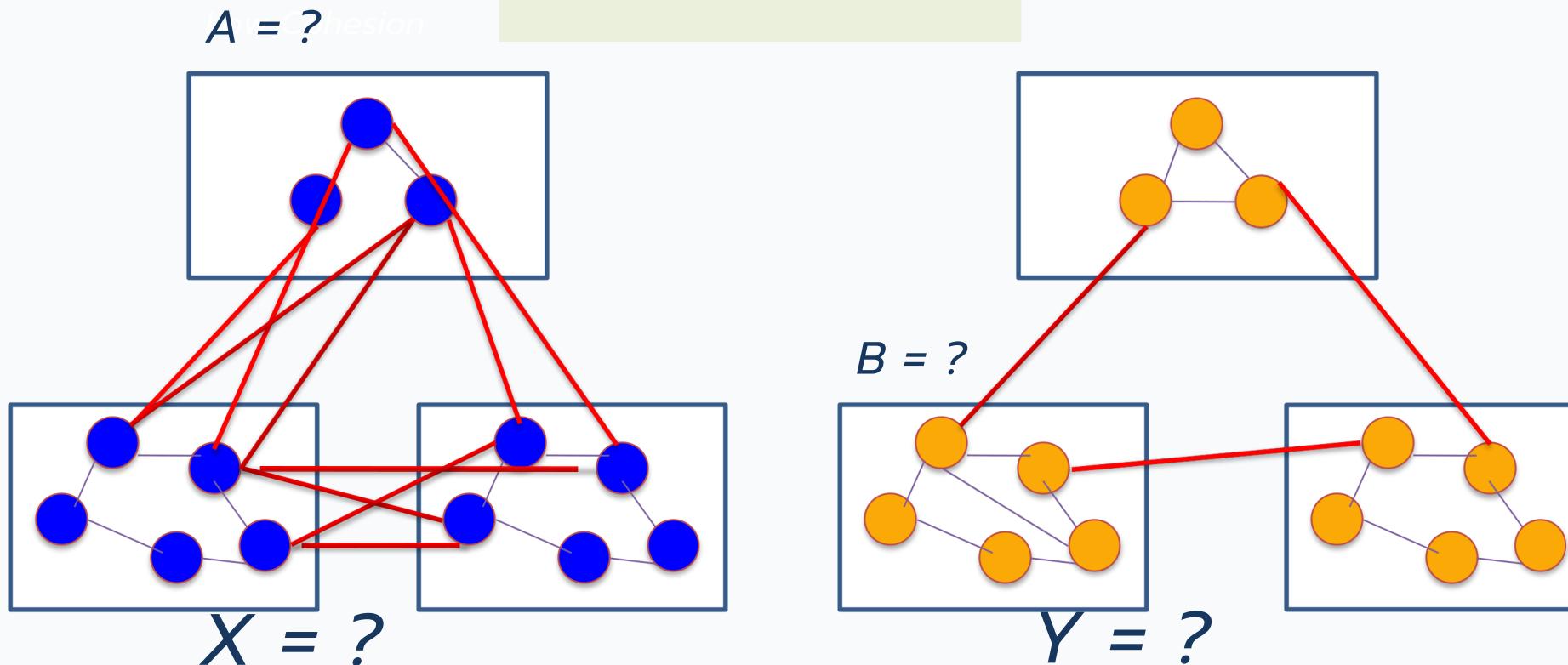
Content Coupling

- โมดูล หนึ่ง สามารถเปลี่ยนแปลงการทำงานของโมดูลอื่น ได้



เป้าหมายของการออกแบบแบบแบ่ง ส่วน

- High Cohesion
- Low Coupling



A) ? High Cohesion

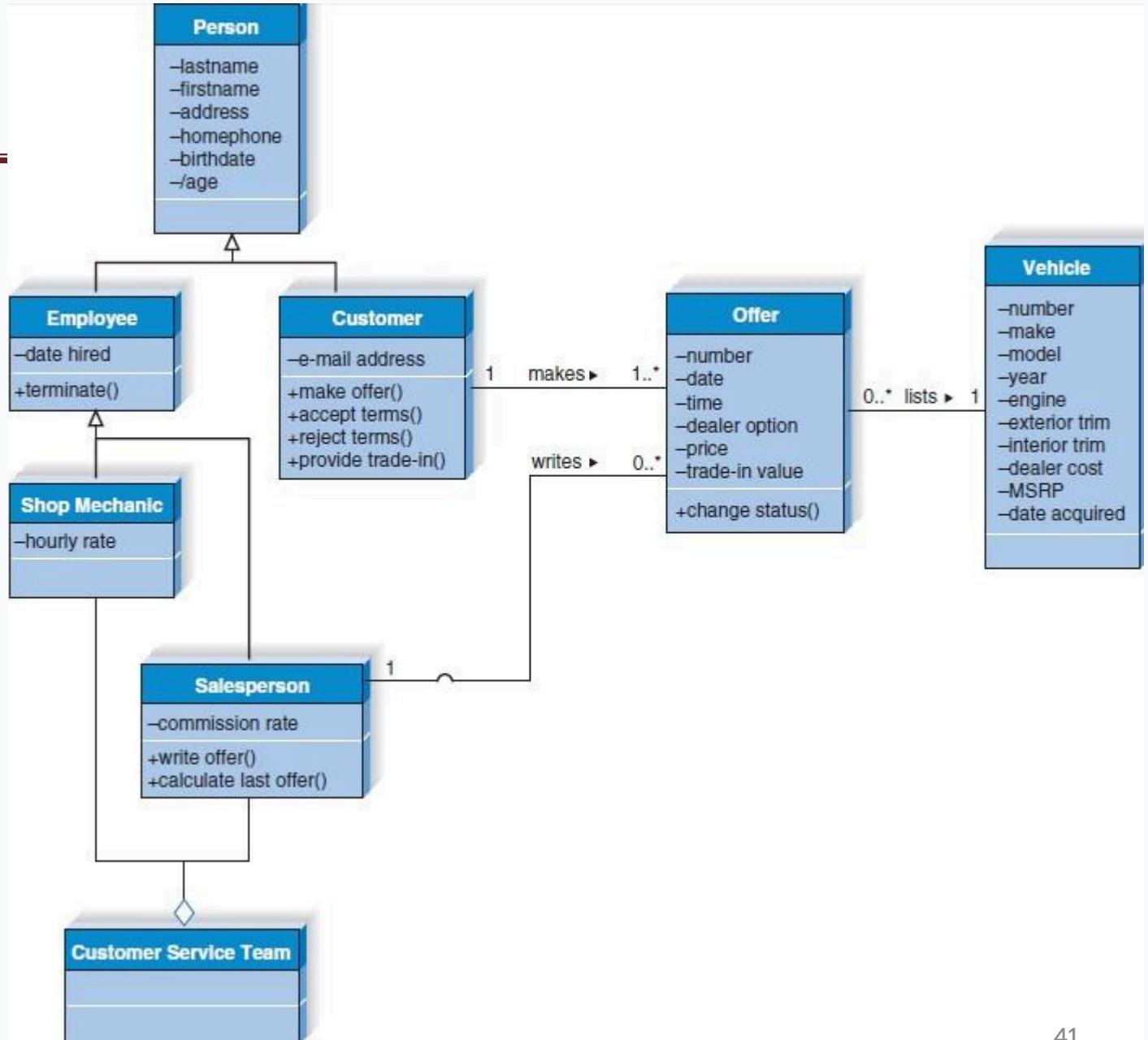
```
class Multiply {  
    int a = 5;  
    int b = 5;  
    public int mul(int a, int b)  
    {  
  
        this.a = a;  
        this.b = b;  
        return a * b;  
    }  
}  
class Display {  
  
    public static void main(String[] args)  
    {  
        Multiply m = new Multiply();  
        System.out.println(m.mul(5, 5));  
    }  
}
```

B) ? Low Coupling

```
public interface Topic { }  
class Topic1 implements Topic {  
    public void understand()  
    {  
        System.out.println("Understand");  
    }  
}  
class Topic2 implements Topic {  
    public void understand()  
    {  
        System.out.println("understand");  
    }  
}  
public class Subject {  
    public static void main(String[] args)  
    {  
        Topic t = new Topic1();  
        t.understand();  
    }  
}
```

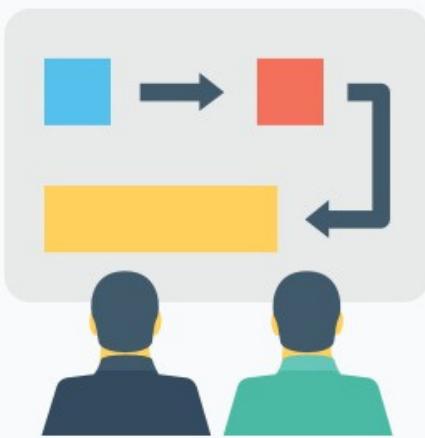
9.) Design Class

- วิเคราะห์ระบบด้วยแนวทางเชิงวัตถุ
 - ขั้นตอนการวิเคราะห์แบบจำลองคลาสระดับบน
 - ขั้นตอนการออกแบบ กำหนดรายละเอียดแบบจำลองคลาส และต่อ งสร้าง งให้บุ๊ก ถึงโครงสร้าง งภายใต้ส นับ กระบวนการธุรกิจ
- Design class 5 ชนิด ได้แก่ User Interface, Business Domain, Process, Persistent และ System Class



กลยุทธ์และเบื้องต้นในการออกแบบซอฟต์แวร์

- กลยุทธ์ในการออกแบบซอฟต์แวร์
 - เป็นเพียงหลักและแนวทางในการปฏิบัติงานแบบซอฟต์แวร์เท่านั้น ไม่ได้ระบุถึงวิธีการงานอย่างชัดเจน
- ระเบียบวิธี (Methodology)ในการออกแบบซอฟต์แวร์
 - ระบุถึงรายละเอียดของวิธีการงานอย่างชัดเจน
 - พร้อมกับเตรียมสัญลักษณ์ต่างๆ ของแบบจำลองเช่นพารามิเตอร์นั้นไว้ให้ใช้งานช่วยให้เกิดความเข้าใจในวิธีการของซอฟต์แวร์ได้ง่ายขึ้น



กลยุทธ์และระเบียบวิธีการออกแบบซอฟต์แวร์ (ต่อ)

- กลยุทธ์ทั่วไปในการออกแบบซอฟต์แวร์ (General Strategy)
- การออกแบบเชิงฟังก์ชัน (Function-Oriented Design)
- การออกแบบเชิงวัตถุ (Object-oriented Design)
- การออกแบบโดยใช้ข้อมูลเป็นศูนย์กลาง (Data-structure Centered Design)
- การออกแบบคอมโพเนนท์ (Component-base Design: CBD)



สรุป (Summary)

- การอ่านแบบซอฟต์แวร์คืออะไร
- หลักการอ่านแบบซอฟต์แวร์
- กระบวนการอ่านแบบซอฟต์แวร์
- สถาปัตยกรรมซอฟต์แวร์
- แนวคิดในการอ่านแบบซอฟต์แวร์

กิจกรรมท้ายบท

- การออกแบบซอฟต์แวร์มีความสำคัญอย่างไรต่อวิศวกรรมซอฟต์แวร์ และมีกระบวนการอะไรบ้าง จงอธิบายตามความเข้าใจ
- การออกแบบซอฟต์แวร์ที่ดีต้องคำนึงถึงอะไรบ้าง เพื่อนำไปสู่การออกแบบที่ดี
- โครงสร้างสถาปัตยกรรม MVC คืออะไร ประกอบด้วยอะไรบ้างและมีความจำเป็นอย่างไรในการพัฒนาซอฟต์แวร์
- จงอธิบายเกี่ยวกับนวัตกรรมแก้ปัญหาที่มีรูปแบบเป็นกลาง (Design Pattern) โดยใช้ Proxy Pattern ชั่งน้ำหน้าไปใช้เพื่อแก้ปัญหาอะไร มีรูปแบบการทำงานอย่างไรจงอธิบาย
- จงอธิบายเกี่ยวกับแนวคิดในการอออกแบบซอฟต์แวร์แบบ Functional Independence พร้อมกับยกตัวอย่างการนำเสนอโปรแกรมประกอบ

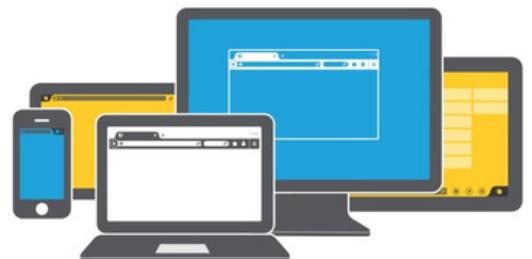
เอกสารอ้างอิง

-
- น้าฟน อ้วสวเมชน, หลักการพื้นฐานของวิศวกรรมซอฟต์แวร์ (Fundamentals of Software Engineering), กรุงเทพฯ: ชีเอ็ดยูเคชั่น, 2560.
 - กิตติ ภักดีวัฒนาภุลุ, วิศวกรรมซอฟต์แวร์ (Software Engineering), กรุงเทพฯ: เคทีพี คอมพ์ แอนด์ คอนซัลท์, 2552.
 - Lan Sommerville, Software Engineering Ninth Edition, Pearson Education, Inc., publishing as Addison-Wesley, 2011.
 - Eric Freeman, Head First Design Patterns, O'Reilly, 2004.

บทที่ 8

การออกแบบซอฟต์แวร์: การออกแบบส่วนต่อประสานผู้ใช้ (Software Design: User Interface Design)

วิชา วิศวกรรมซอฟต์แวร์ (04-06-322)



วัตถุประสงค์การเรียนรู้

-
- เพื่อให้ผู้เรียน นิมิต ความความเข้าใจเกี่ยวกับ กระบวนการออกแบบสู่ นต่อ ประสานผู้ใช้งาน
 - เพื่อให้ผู้เรียน นิมิต ความความเข้าใจเกี่ยวกับ การประเมิน คุณภาพของการออกแบบสู่ ต่อประสานในด้านต่าง ๆ



หัวข้อ (Agenda)

บทนำ (Overview)

การออกแบบส่วนประสานกับผู้ใช้ (User Interface Design)

กระบวนการออกแบบส่วนต่อประสานกับผู้ใช้

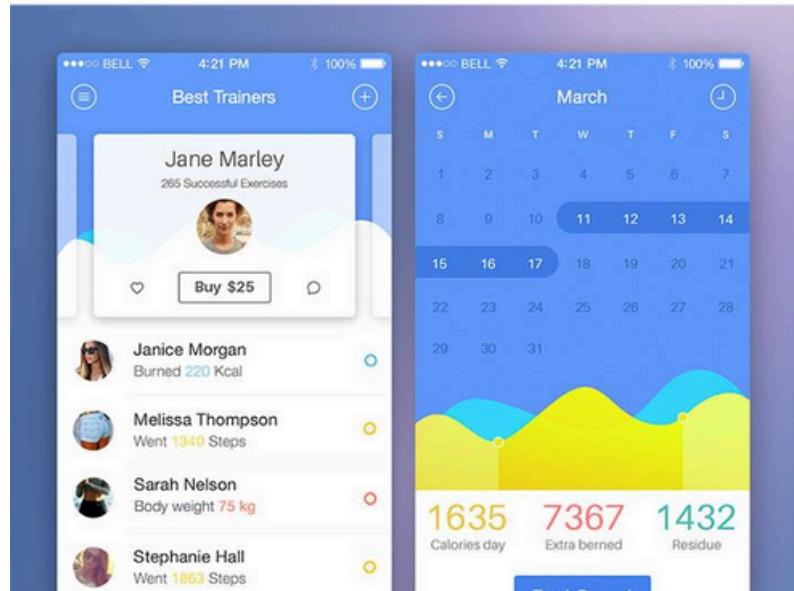
- - การวิเคราะห์และการสร้างแบบจำลองส่วนต่อประสาน (Interface Analysis and Modeling)
 - การออกแบบส่วนต่อประสาน (Interface Design)
 - การสร้างส่วนต่อประสาน (Interface Construction)
 - การตรวจสอบส่วนต่อประสาน (Interface Validation)
- แนวการทำงานเขียนโปรแกรมที่ดี
- สรุป (Summary)

บทนำ (Overview)

คอมโพเนนท์ ซอฟต์แวร์
(Software Component)

ซอฟต์แวร์กับส่วนตัวภายนอก
(Software and external agent)

ซอฟต์แวร์กับผู้ใช้งาน
(Software and User)



การออกแบบส่วนต่อประสานกับผู้ใช้ (User Interface Design)



ง่ายต่อการเรียนรู้ (Easy to learn?)



ง่ายต่อการใช้งาน (Easy to use?)



ง่ายต่อการเข้าใจ (Easy to understand?)

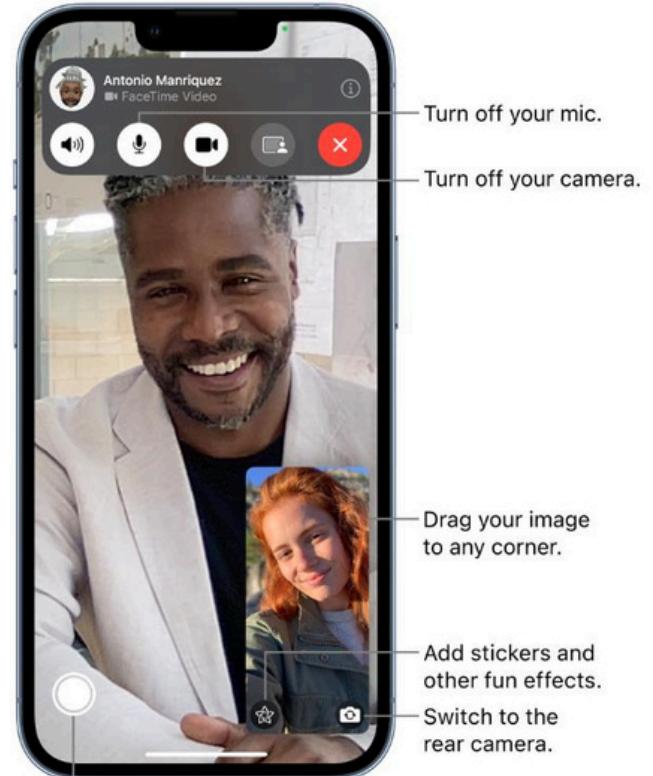


Photo Source:
<https://support.apple.com/guide/iphone/make-and-receive-calls-iph7801d5771/ios>

การออกแบบส่วนต่อประสานกับผู้ใช้ (ต่อ)

- ส่วนประสานกับผู้ใช้ (User Interface)



ส่วนติดต่อระหว่างผู้ใช้งานระบบเพื่อเตรียมสารสนเทศการทำงานและนำเสนอสารสนเทศ
เรียกว่า “การออกแบบจอกาพ (Screen Design)”

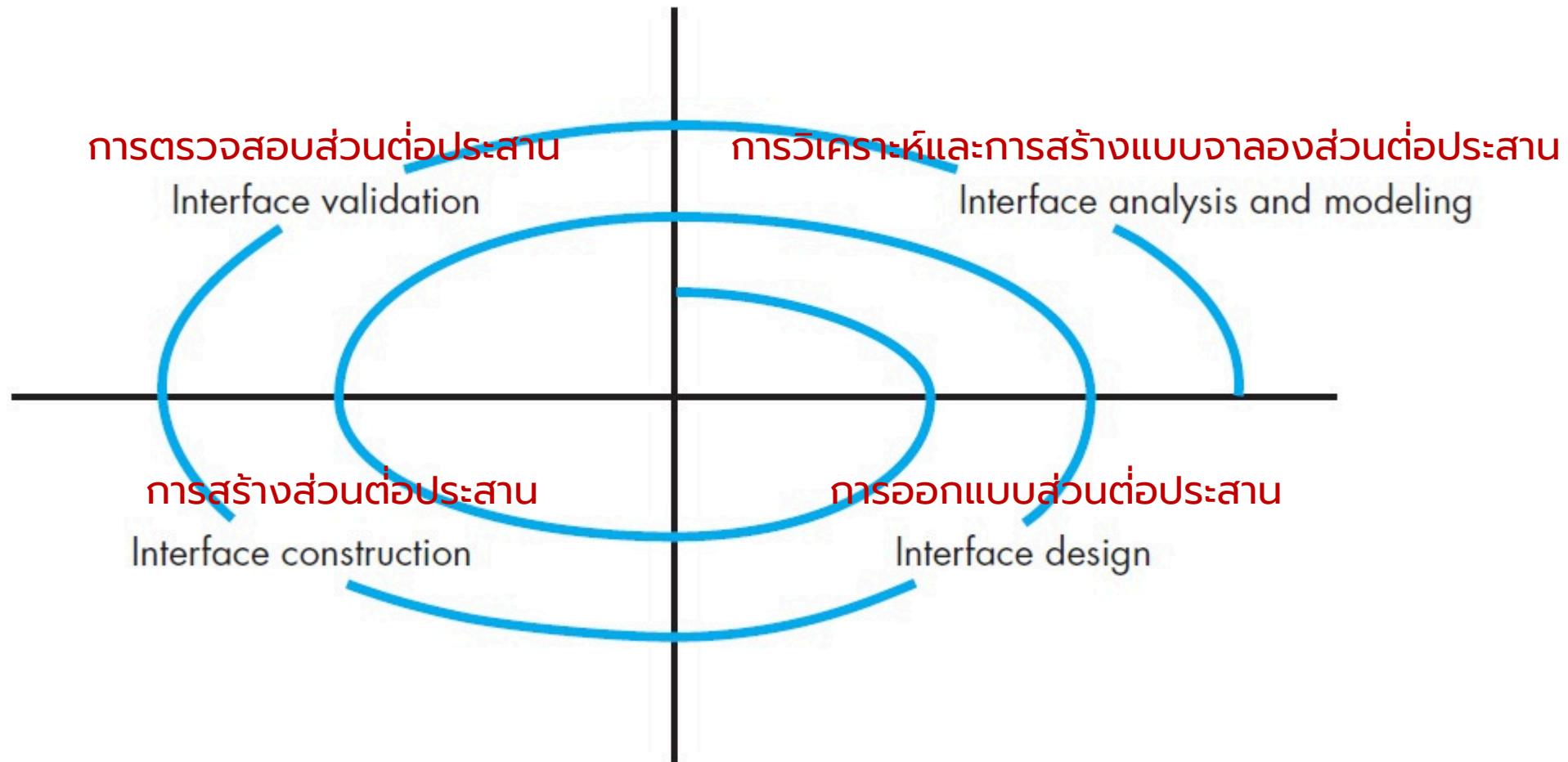


เกิดการทำงานตามวัตถุประสงค์ สามารถเป็นเครื่องปั่นชัก ารใช้ง ขพต์เวร์ เน้น
ส่วนประสานแบบกราฟฟิก (Graphic User Interface: GUI)



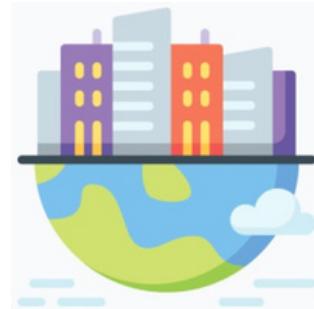
การใช้งานต้องสอดคล้อง ความแตกต่างกันของผู้ใช้งานทั้งพูติกรรมและ
ประสบการณ์การทำงาน

กระบวนการออกแบบส่วนต่อประสานกับผู้ใช้



การวิเคราะห์และการสร้างแบบจำลองส่วนต่อประสาน (Interface Analysis and Modeling)

- กระบวนการแรกของการออกแบบส่วนต่อประสาน
- ในการวิเคราะห์ส่วนต่อประสานต้องเข้าใจปัญหา ดังนี้
 - the people-เข้าใจคน บุคลากรที่ใช้งานบูรณา
 - the tasks-เข้าใจงานที่ผู้ใช้ต้องการใช้เพื่อให้ทำงานให้สำเร็จ
 - the content -เข้าใจเนื้อหาที่จะต้องนำเสนอในส่วนต่อประสาน
 - the environment - เข้าใจสิ่งแวดล้อมทั่วไป เช่น ภูมิประเทศ ภัยธรรมชาติ

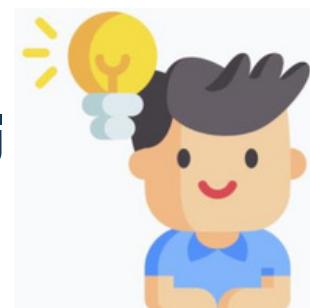


การวิเคราะห์ผู้ใช้งาน (User Analysis)

- ควรเข้าใจผู้ใช้ว่าจะใช้ระบบอย่างไร ต้องการส่วนต่อประสานลักษณะใด
- การสัมภาษณ์ ชี้แจงจะช่วยให้นักออกแบบเข้าใจว่า
 - ใครคือ ผู้ใช้ บ้าง งาน
 - แบ่งกลุ่มผู้ใช้อย่างไร
 - ผู้ใช้แต่ละกลุ่มมี特กษะและประสบการณ์ในระดับใด
 - แบบจำลองสภาพจิตใจของผู้ใช้ที่มีต่อระบบเป็นอย่างไร
 - ส่วนต่อประสานจะตอบสนองความต้องการของผู้ใช้ได้อย่างไร



ผู้ใช้ บังเอิญใหม่
(Novice User)



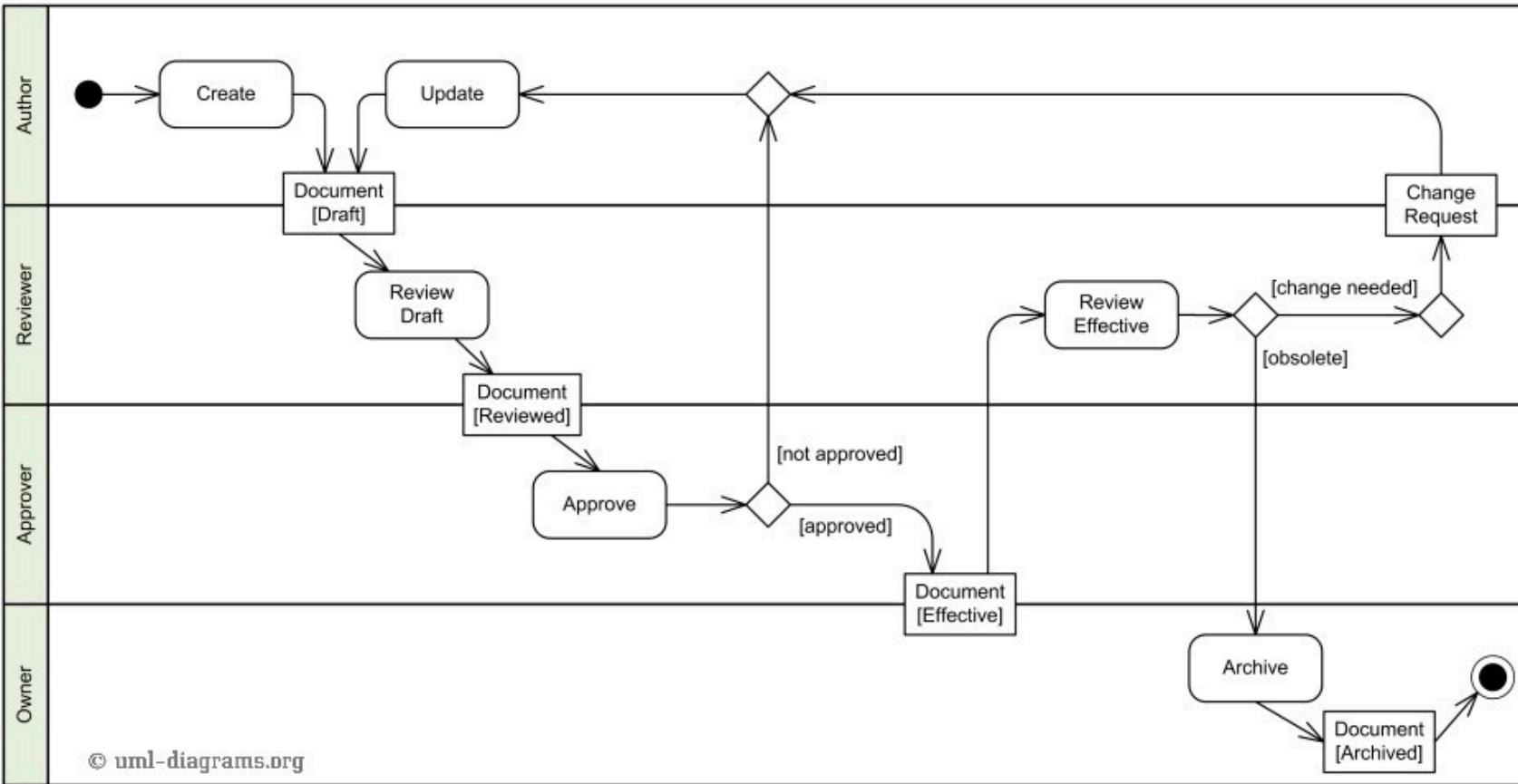
ผู้ใช้ ก็ มี พื้นฐานความรู้
(Knowledgeable
Intermittent User)



ผู้ใช้ ระดับ เชี่ยวชาญ
(Expert)

การจำลองและวิเคราะห์งานย่อ

- ผู้ออกแบบควรตอบคำถามเหล่านี้...
 - อะไรคืองานที่ผู้ใช้ต้องดำเนินการในแต่ละสถานการณ์?
 - กิจกรรมหลัก/ย่อยใดของงานที่ผู้ใช้ต้องดำเนินการ?
 - อะไรคือ ปัญหาของ โอดเมนเฉพาะที่ผู้ใช้ต้องบริหารจัดการเพื่อให้งานสามารถดำเนินการไปได้?
 - ลักษณะการดำเนินกิจกรรมของงานเป็นอย่างไร?
 - อะไรคือ ลำดับชั้นของกิจกรรม?
- เทคนิคในการวิเคราะห์ออกแบบส่วนต่อประสาน
 - สามารถนิยามการปฏิสัมพันธ์พื้นฐานด้วยแผนภาพยูสเคส
 - งานที่มี การปฏิสัมพันธ์อื่นๆ ที่ปรับแต่งด้วยการอธิบายรายละเอียดของงาน
 - ระบุออบเจกต์ ที่ส่วนต่อประสาน (classes) ด้วยรายละเอียดของอ็อตเจกต์
 - กระบวนการของงานสมบูรณ์ เมื่อมี ข้อมูลหลายฝ่าย/บทบาทเข้ามาส่วนมีเกี่ยวข้อง สามารถนิยามการวิเคราะห์การให้ผลของงาน



- แผนภาพกิจกรรม กระบวนการบริหารจัดการเอกสาร (Document Management Process)
- มีผู้เกี่ยวข้อง 4 ส่วน ดังนี้ (a)_____, (b) _____, (c) _____ และ (d)_____ นักออกแบบแบบส่วนต่อประสาน
- ควรคำนึงถึงลักษณะของผู้ใช้ และกิจกรรมงาน

การวิเคราะห์การนำเสนอเนื้อหา

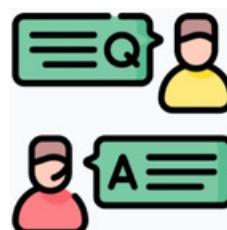
เนื้อหาที่นำเสนอ: รายงานที่เป็นตัวอักษร
รูปภาพ หรือข้อมูลเฉพาะ

- ถูกสร้างโดยส่วนประกอบของระบบที่ไม่เกี่ยวข้องกับส่วนต่อประสาน
- ดึงมาจากข้อมูลที่เก็บในฐานข้อมูล
- ส่งมาจากระบบภายนอก
- การวิเคราะห์การนำเสนอเนื้อหาที่ให้กราฟถึงเอกสารที่ต้องการและการแสดงผลที่ต้องการ รูปแบบและความสามารถของเนื้อหาจะถูกพิจารณา

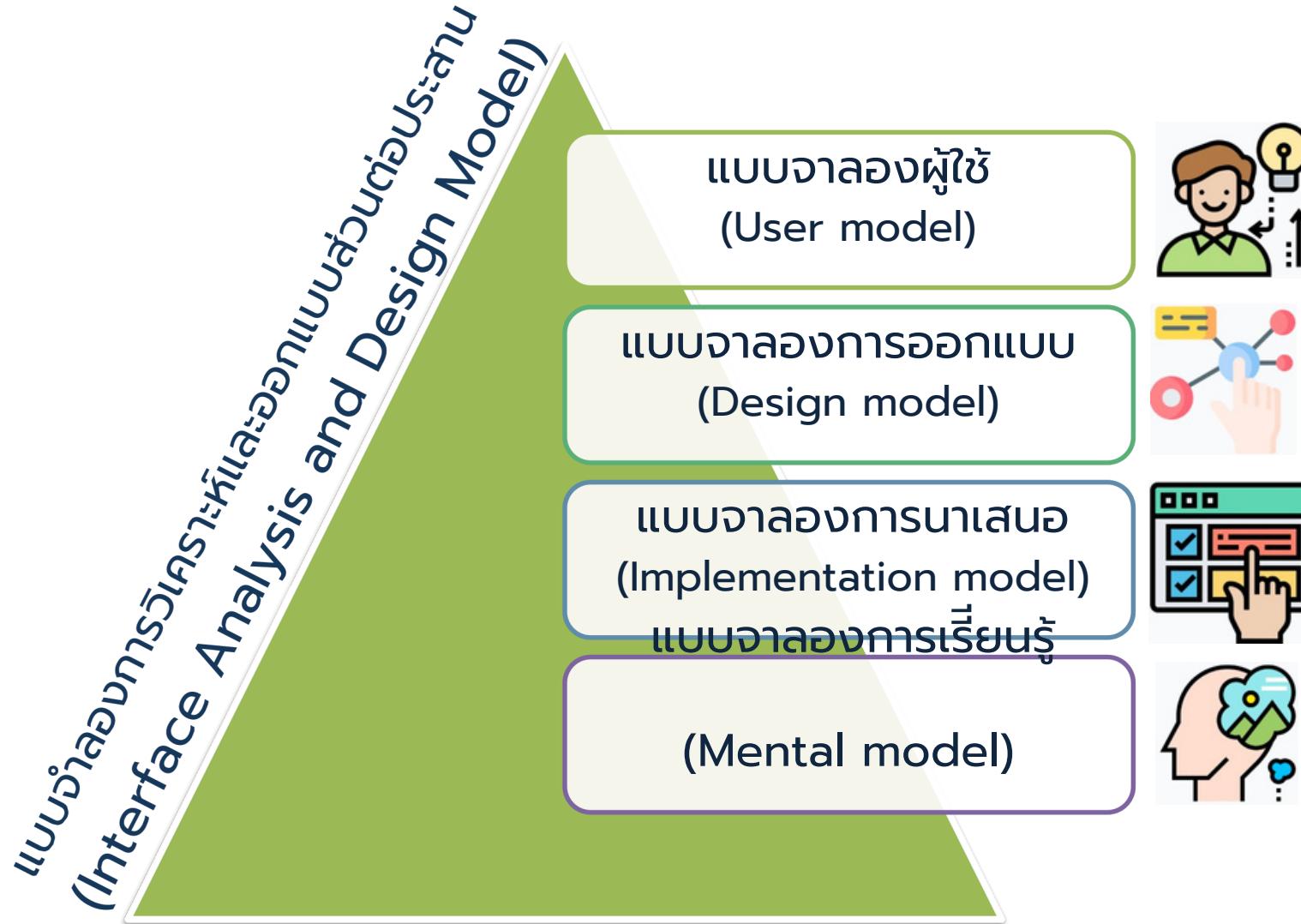


การวิเคราะห์สิ่งแวดล้อมในการทำงาน

- นักออกแบบควรคำนึงถึงสภาพแวดล้อมที่ใช้งานของระบบ ข้อจำกัดทางกายภาพที่อาจเป็นอุปสรรคในการใช้งาน
 - โรงงานเสียงดัง การใช้ลูกระโพงอาจไม่เหมาะสม
 - การใช้แม่สีคีย์บอร์ดในพื้นที่คับแคบ
- วัฒนธรรมในการทำงาน
 - ข้อมูลต้องได้รับการรับรองจากหลายฝ่าย ก่อนบันทึกหรือไม่
 - ผู้ใช้งานจะได้รับความช่วยเหลือ จากระบบอย่างไร
- นักออกแบบต้องตอบคำถามเหล่านี้ก่อนการออกแบบเสร็จสิ้น และควรเพิ่มส่วนต่อประสานกับผู้ใช้งานโดยความสะดวกด้วย



การสร้างแบบจำลองส่วนประisan กับผู้ใช้



การออกแบบส่วนต่อประสาน (Interface Design)

- 1 นิยามวัตถุและตัวดำเนินการ โดยใช้ข้อมูลจากการวิเคราะห์
Define interface objects and actions (operations)
กำหนดเหตุการณ์ที่เป็นการกระทำของผู้ใช้
- 2 Define events (user actions)
แสดงรูปภาพถึงสถานะของส่วนต่อประสานกับผู้ใช้ ที่ได้รับ ผัส
- 3 Depict each interface state
- 4 อธิบายให้ทราบความหมายของข้อมูลที่แสดง ระบุว่าผู้ใช้จะเข้าใจสถานะของระบบอย่างไร
Indicate how the user interprets the state of the system

รูปแบบการออกแบบส่วนต่อประสาน (Interface Design Pattern)



หน้า ตาสั่ว นต่อ ประสาน



การจัดการข้อมูลอย่างเหมาะสม



Page Layout และรูปแบบการ
นำเสนอ



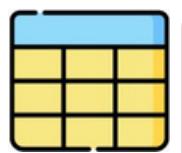
การตรวจสอบเมื่อทาง



แบบฟอร์มและข้อมูลน าเข้า



การค้นหาข้อมูล



การแสดงข้อมูลในรูปแบบตาราง



Page Element

ข้อควรคำนึงในการออกแบบ



ระยะเวลาตอบกลับ (Response time)



การช่วยเหลือ (Help facilities)



การจัดการข้อผิดพลาด (Error handling)



การกำหนดชื่อเมนูและคำสั่ง (Menu and command labeling)

การเข้าถึงแอปพลิเคชัน (Application accessibility)

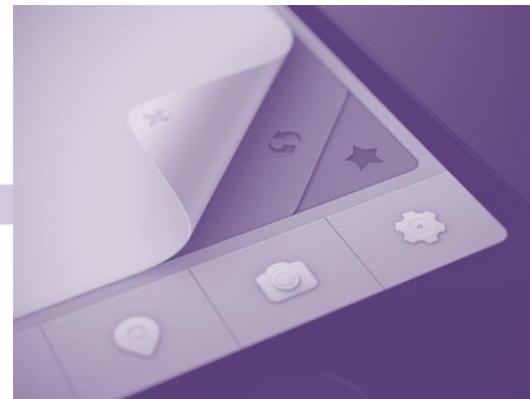
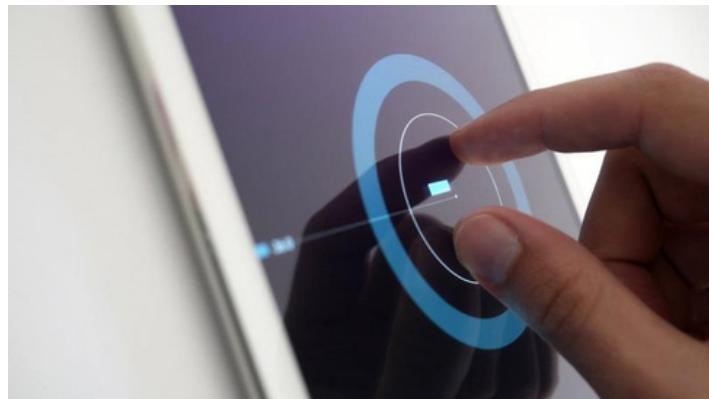


ความเป็นสากล (Internationalization)



การออกแบบส่วนประisan (ต่อ)

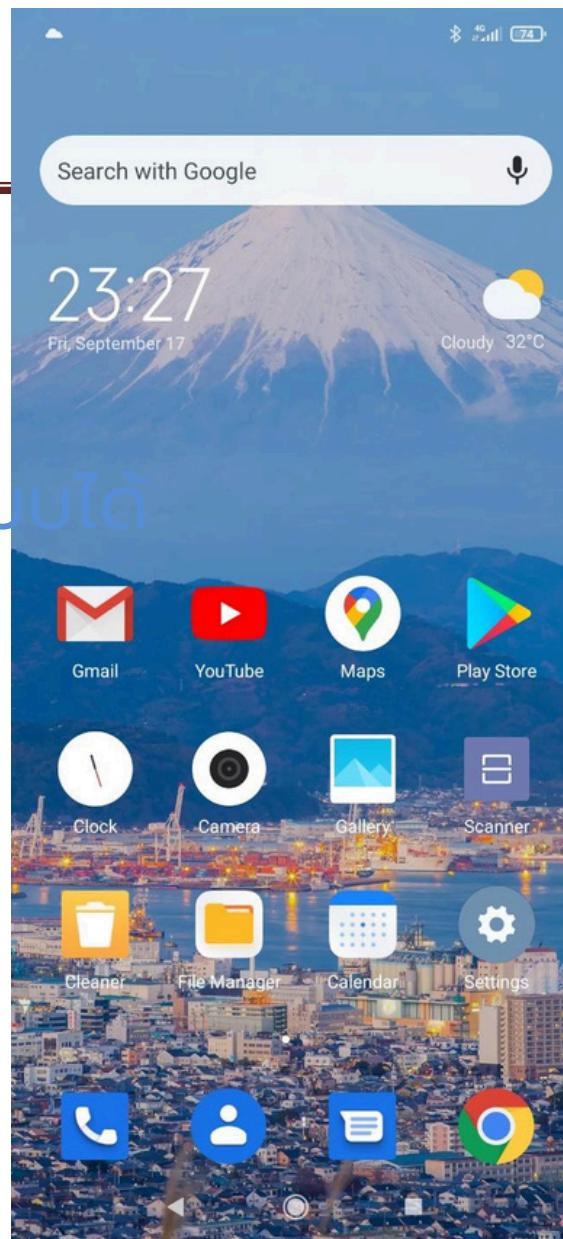
- Theo Mandel ได้บัญญัติกฎ 3ข้อในการออกแบบส่วนต่อประisan ดังนี้
 - ให้ผู้ใช้เป็นผู้ควบคุมการทำงาน -Place the user in control
 - ลดภาระการต้องจำของผู้ใช้-Reducetheuser'smemoryload
 - สร้างส่วนต่อประisanอย่างคงเส้นคงวา (สอดคล้องกัน) -Make the interface consistent



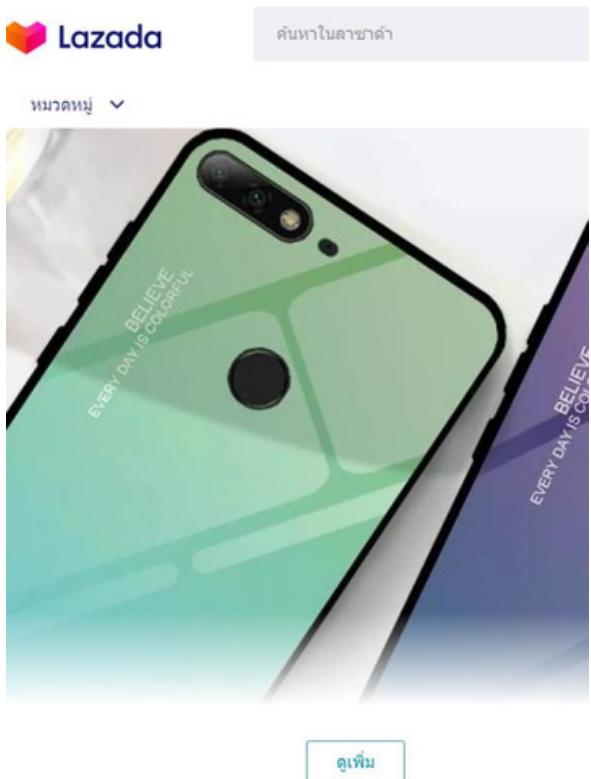
ให้ผู้ใช้เป็นผู้ควบคุมการทำงาน (Place the user in control)

หลักเกณฑ์ในการออกแบบที่ให้ผู้ใช้ควบคุม ประกอบด้วยดังนี้

- กำหนดโหมดการโต้ตอบในลักษณะที่ไม่บังคับผู้ใช้โดยไม่จำเป็นหรือในทางที่ผู้ใช้ไม่ต้องการที่จะทำ
- จัดให้มีการโต้ตอบที่ยืดหยุ่น สามารถโต้ตอบกับระบบได้มากกว่า 1 ช่องทาง
- อนุญาตให้ผู้ใช้ตัดสินใจ หยุดหรือสามารถยกเลิกได้
- ออกแบบให้การโต้ตอบ ตอบเป็นไปตามระดับความชำนาญ งาน เตรียมเครื่องมือสร้างสรรค์งานแบบอัตโนมัติให้กับผู้ใช้
- ซ่อนรายละเอียดด้านเทคนิคจากผู้ใช้ก็วุ่นไป ไม่ควรให้ผู้ใช้ติดต่อกับระบบปฏิบัติการด้วยการพิมพ์ค่าส่งโดยตรง
- การออกแบบวัตถุที่วางไว้บนจอให้เข้าถึงโดยตรง



ลดภาระการต้องจดจำของผู้ใช้ (Reduce the user's memory load)



- ซอฟต์แวร์ที่ให้ผู้ใช้รายการและอี้ดการทำงานมากเกินไป
 - เสียงต่อการเกิดความผิดพลาด
- ช่วยเตือนความจำให้ผู้ใช้
- Mandel ออกแบบหลักการที่ช่วยลดภาระการจดจำของผู้ใช้ดังนี้
 - ลดความต้องการใช้งานหน่วยความจำระยะสั้นของผู้ใช้บนหน้าจอ
 - การกำหนดค่าโดยปริยายที่มีความหมาย
 - นิยามปุ่มลัด (Shortcut) ที่เข้าใจง่าย
 - การจัดภาพของส่วนต่อประสานควรเป็นไปตามอุปกรณ์ของโลกจริง
 - เปิดเผยข่าวสารในลักษณะค่อยๆ เพิ่มพูน

ซื้อยูนิตเฉพาะของ สำหรับHuawei Y7 2018 / Huawei Y7 Prime 2018 / Huawei Y7 Pro / Huawei Honor 7c Gradient

แมร์ค

FlyGoods

เบอร์

Sports,men,mens,FOR HER

สีที่ซึ่งจะถูกต้อง

ในการซื้อ

กันดู, กันตรวจสอบ

สำหรับ

Huawei Y7 2018 / Huawei Y7 Prime 2018 /

Huawei Y7 Pro / Huawei Honor 7c

SKU

33446079_TH-647770973

ประเภทของผลิตภัณฑ์

สำหรับ

แมร์คที่ใช้งานได้

Huawei

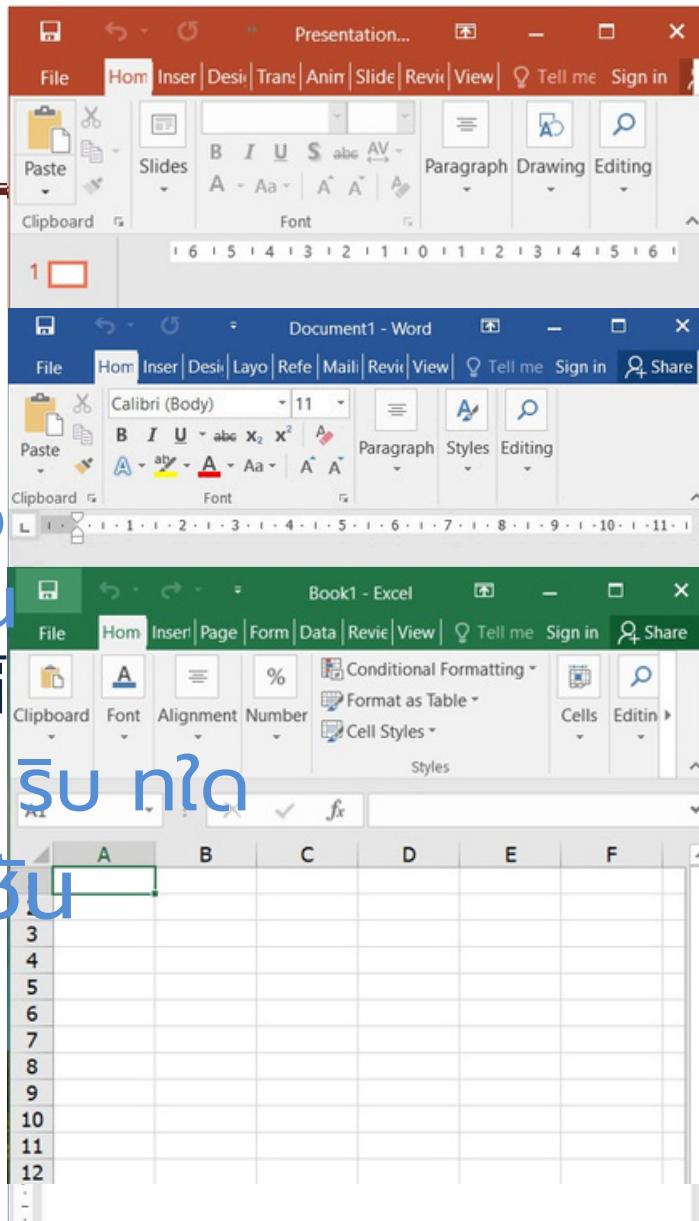
ประเภทของการรับ

สำหรับ

ไม่มีการรับคืน

ส่วนประสานต้องสอดคล้องกัน (Make the interface consistent)

- ส่วนประสานควรรับและแสดงผลในลักษณะคงเส้นคงวา
 - ข้า วสารทางภาษาจีด ระเบียบ บตามมาตรฐานการอักเบบ
เดียวกันตลอดทุกหน้าจอของระบบ
 - กลไกการท่องระบบจากงานหนึ่งสู่งานหนึ่งเป็นไปอย่างคง
เส้นคงวา สอดคล้องกันเชื่อมโยงกันเป็นลำดับขั้นตอน
- หลักการอักเบบที่ช่วยให้ส่วนต่อประสานคงเส้นคงวา มีดังนี้
 - ช่วยให้ผู้ใช้เข้าใจว่า งานปัจจุบันอยู่ภายใต้รูปแบบใด
 - คงเส้นคงวาตลอดทั้งตระกูลของแอปพลิเคชัน
 - ถ้ารูปแบบการโต้ตอบที่ผ่านมาหากให้ผู้ใช้เกิดความคาดหมาย
อย่าเปลี่ยนกฎนั้น ยกเว้นมีเหตุผลสมควร

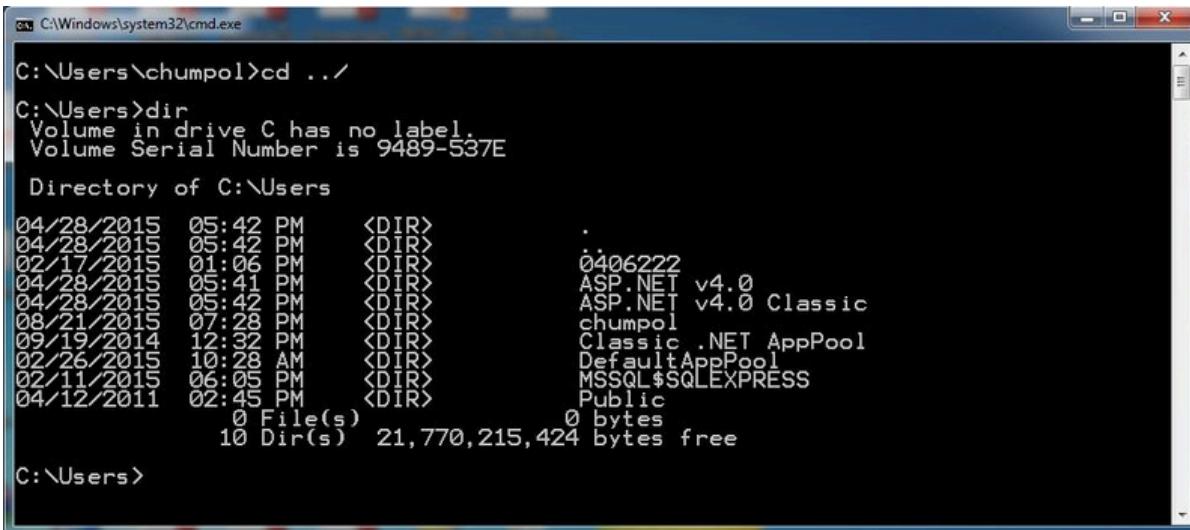


รูปแบบของ User Interfaces

- ผู้ใช้งานสามารถโต้ตอบกับระบบอย่างมีประสิทธิภาพ นิยมใช้แบบกราฟิก (Graphic User Interface: GUI) ซึ่งมีรูปแบบดังนี้
 - การโต้ตอบด้วยคำสั่ง (Command Language Interaction)
 - การโต้ตอบด้วยเมนูคำสั่ง (Menu Interaction)
 - การโต้ตอบด้วยแบบฟอร์ม (Form Interaction)
 - การโต้ตอบด้วยการทำงานเชิงวัตถุ (Object-Based Interaction)
 - การโต้ตอบด้วยภาษาธรรมชาติ (Natural Language Interaction)

การโต้ตอบด้วยคำสั่ง (Command Language Interaction)

- เป็นการโต้ตอบกับระบบโดยที่ผู้ใช้งานต้องพิมพ์คำสั่งลงในช่องป้อนคำสั่ง เพื่อกระตูนให้เกิดการท างานในระบบ
- ผู้ใช้งานต้องจำรากภาษาและกฎเกณฑ์ต่างๆ
 - ผู้ใช้ก็สามารถการใช้ระบบปฏิบัติการ DOS
- ลดความนิยมในปัจจุบัน



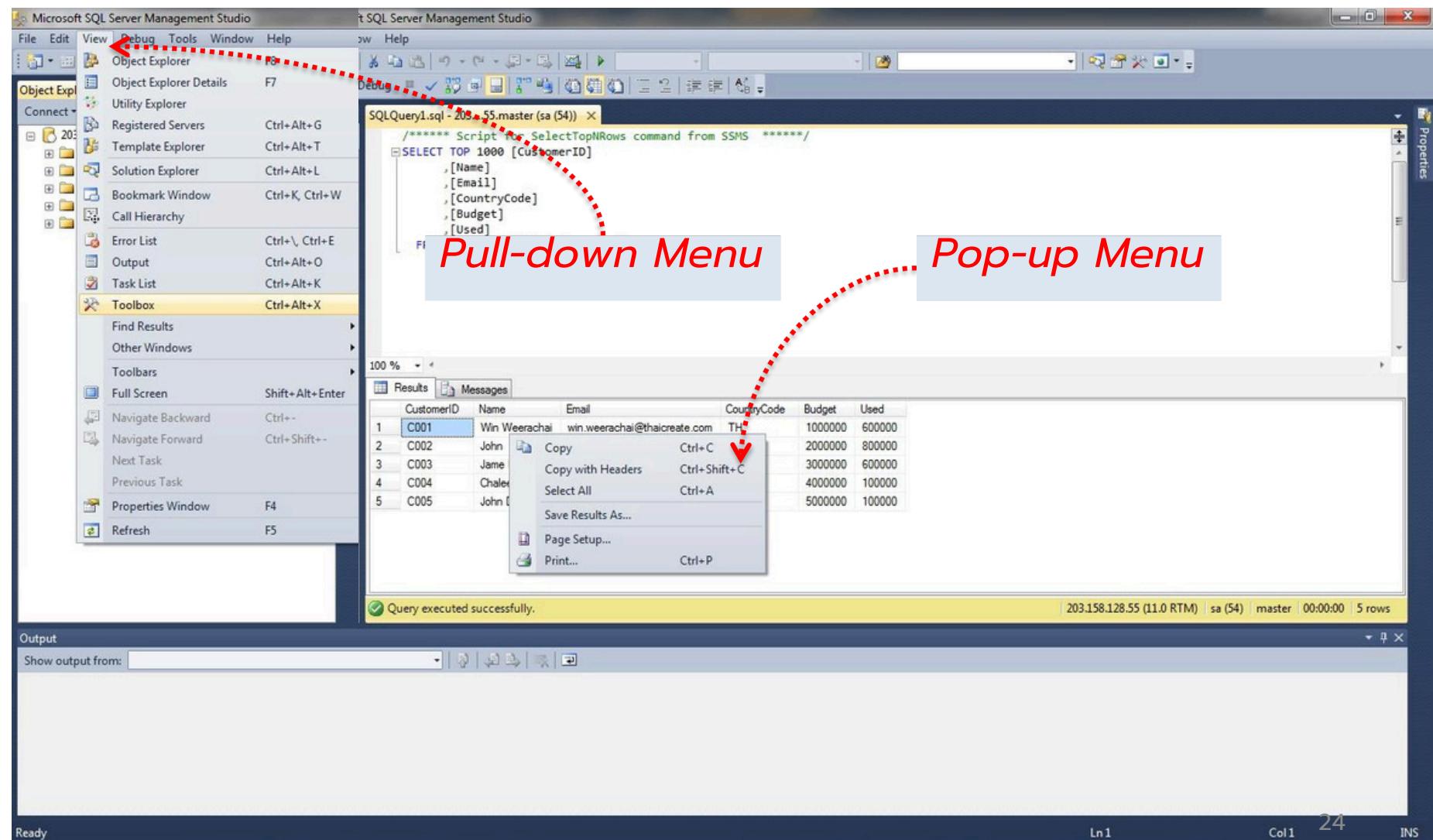
C:\Windows\system32\cmd.exe

```
C:\Users\chumpol>cd ..
C:\Users>dir
Volume in drive C has no label.
Volume Serial Number is 9489-537E
Directory of C:\Users
04/28/2015  05:42 PM    <DIR>          .
04/28/2015  05:42 PM    <DIR>          0406222
02/17/2015  01:06 PM    <DIR>
04/28/2015  05:41 PM    <DIR>          ASP.NET v4.0
04/28/2015  05:42 PM    <DIR>          ASP.NET v4.0 Classic
08/21/2015  07:28 PM    <DIR>          chumpol
09/19/2014  12:32 PM    <DIR>          Classic .NET AppPool
02/26/2015  10:28 AM    <DIR>          DefaultAppPool
02/11/2015  06:05 PM    <DIR>          MSSQL$SQLEXPRESS
04/12/2011  02:45 PM    <DIR>          Public
                           0 File(s)           0 bytes
                           10 Dir(s)  21,770,215,424 bytes free
C:\Users>
```

คำสั่ง
\$ cd../
\$ dir

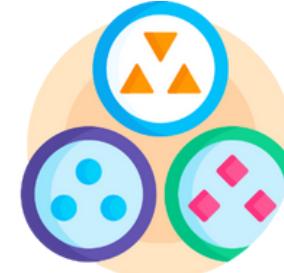
การโต้ตอบด้วยเมนูค้าง (Menu Interaction)

- เป็นการโต้ตอบกับระบบด้วยการแสดงเมนูก้าง โดยผู้ใช้ไม่จำเป็นต้องป้อนค่าสั่งเอง
- รูปแบบเมนูมีดังนี้ คือ
 - Pull-down Menu
 - Pop-up Menu



หลัก เกณฑ์ ในการออกแบบเมนูค าสั่ง

- ควรเลือกใช้ค าสั่งที่สื่อความหมายได้ชัดเจน
- ควรมีการใช้ตัว อ ก บ ร พิมพ์ใหญ่ หรือ ตัวอ ก บ ร พิมพ์เล็ก ตามความเหมาะสม
- ควรมีการจัดกลุ่มคำสั่งที่มีความเกี่ยวข้องกันไว้ในกลุ่มเดียวกัน
- ไม่ควรมีจำนวนเมนูค าสั่งมากเกินไป
- ไม่ควรมีเมนูย่อยส าหรับเมนูค าสั่งที่มีการทำงานย่อยภายในมากเกินไป
- เมื่อมีการเลือกเมนูค าสั่ง ควรออกแบบให้มีแบบสี pragmatique เมนูค าสั่งที่ถูกเลือก



การโต้ตอบด้วยแบบฟอร์ม (Form Interaction)

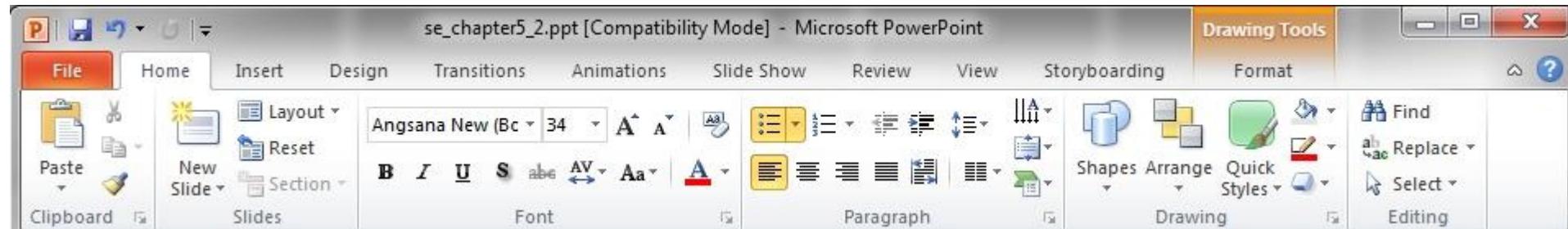
- เป็น การโต้ตอบ ของผู้ใช้ ที่จะต้องป้อนข้อมูลลงในช่องว่างที่อยู่ในแบบฟอร์มที่แสดงหน้าจอคอมพิวเตอร์
- คล้ายการกรอกแบบฟอร์มลงในกระดาษ
- ช่องของป้อนข้อมูล มีตัวชี้วัดความหมาย
- แบ่งส่วนของข้อมูลบนฟอร์มให้เหมาะสม
- ควรแสดงข้อมูลเริ่มต้นให้กับช่องป้อนข้อมูลที่ต้องใช้ข้อมูลนั้นประกอบครั้ง
- ช่องป้อนข้อมูลไม่ควรยาวมากจนเกินไป

The screenshot shows a web-based input form titled "Input Form". The form consists of several input fields and control elements:

- Name:** A text input field with the placeholder text "Type here...".
- Email:** A text input field with the placeholder text "Type here...".
- Subject:** A text input field with the placeholder text "Type here...".
- Milk:** A checked checkbox.
- Sugar:** An unchecked checkbox.
- Lemon (Disabled):** An unchecked checkbox.
- Male:** A selected radio button.
- Female:** An unselected radio button.
- Don't know (Disabled):** An unselected radio button.
- Send >**: A large green button at the bottom right.

การโต้ตอบเชิงวัตถุ (Object-Based Interaction)

- เป็นการโต้ตอบกับระบบที่ใช้สัญลักษณ์
- สัญลักษณ์เป็นตัวแทนคำสั่งที่ใช้ในการปฏิบัติงาน
- สัญลักษณ์รูปภาพแทนคำสั่งการทำงานเรียกว่าไอคอน (Icon)
- ประยุกต์พื้นที่บนหน้าจอ



การโต้ตอบด้วยภาษาธรรมชาติ (Natural Language Interaction)

- เป็น การโต้ตอบ กับ ระบบด้วย การใช้เสียง พูดของผู้ใช้ ชัด แบบ
- ใช้เสียงพูดทั้งการนาข้อมูลเข้าและออกจากระบบ

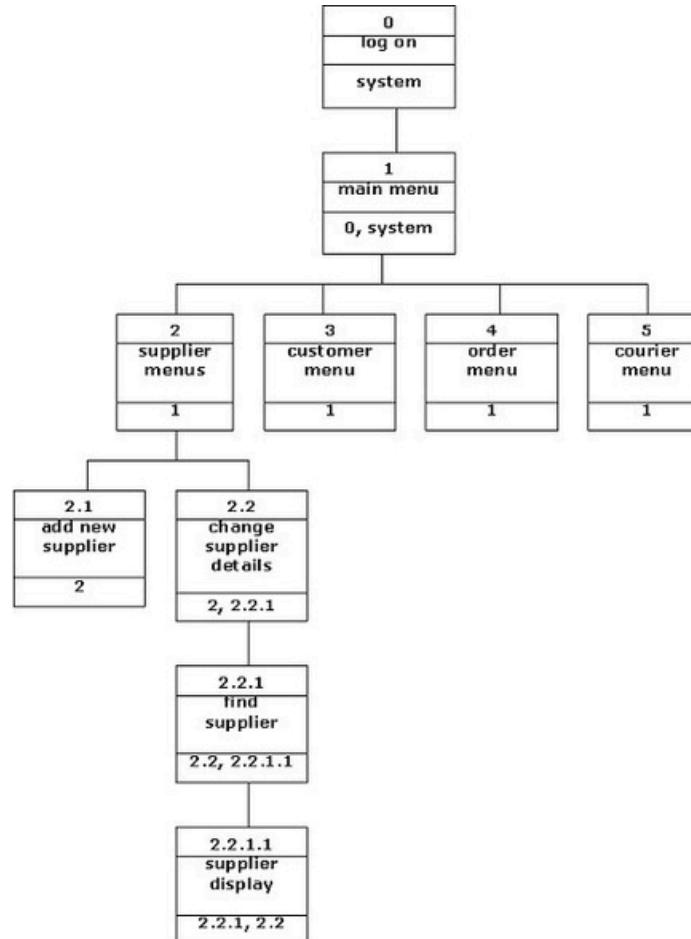
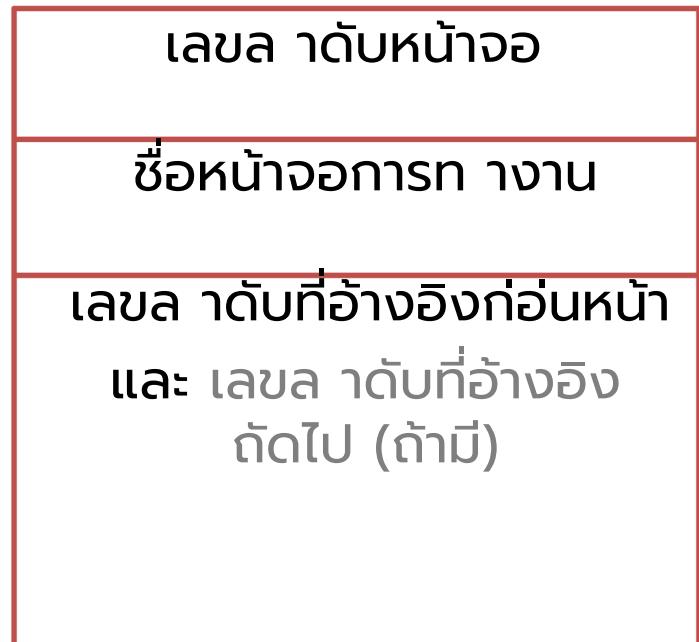


การสร้างส่วนต่อประสาน (Interface Construction)

-
- การน าส่วนต่อประสานที่ได้จากการออกแบบ เพื่อสร้างส่วนต่อประสานผู้ใช้
 - เป็นการออกแบบ ล าดับของการแสดงส่วนติดต่อกับผู้ใช้ของโปรแกรม หรือล าดับของการแสดงส่วน User Interface ทางหน้าจอคอมพิวเตอร์
 - แผนภาพแสดงล าดับการเชื่อมโยงจ�าพ (Dialogues Diagram) ประกอบไปด้วย 3ส่วน
 - ส่วนบน: เลขล าดับหน้าจอ
 - ส่วนกลาง: ชื่อหน้าจอการท างาน
 - ส่วนล่าง: เลขล าดับหน้าจอที่อ้างอิงมา ต่อไป หรือ ย้อนกลับ

Dialogues Diagram

- Dialogues Diagram เป็นแผนภาพแสดงลำดับการเชื่อมโยงของจargon

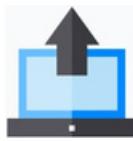


การสร้างส่วนต่อประสาน (ต่อ)

การออกแบบระบบนำทาง
(Navigation Design)



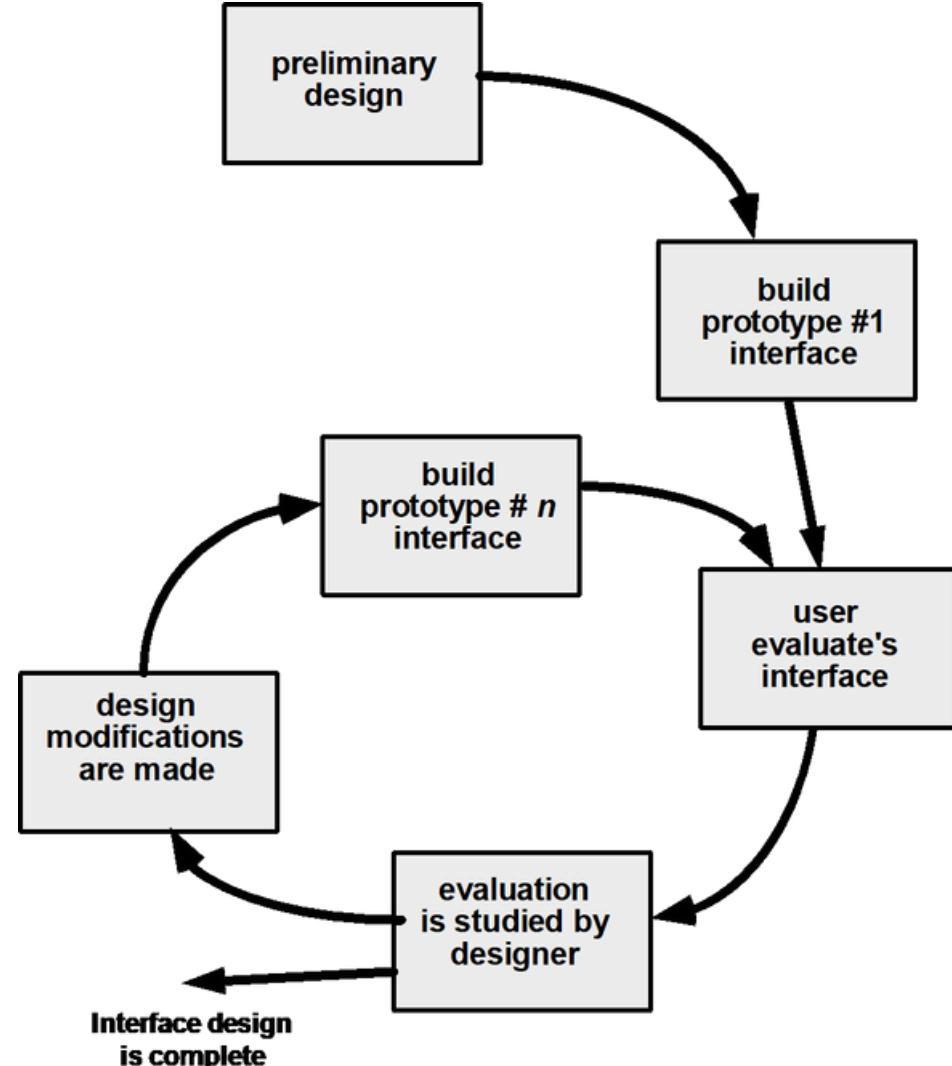
การออกแบบข้อมูลนาเข้า
(Input Design)



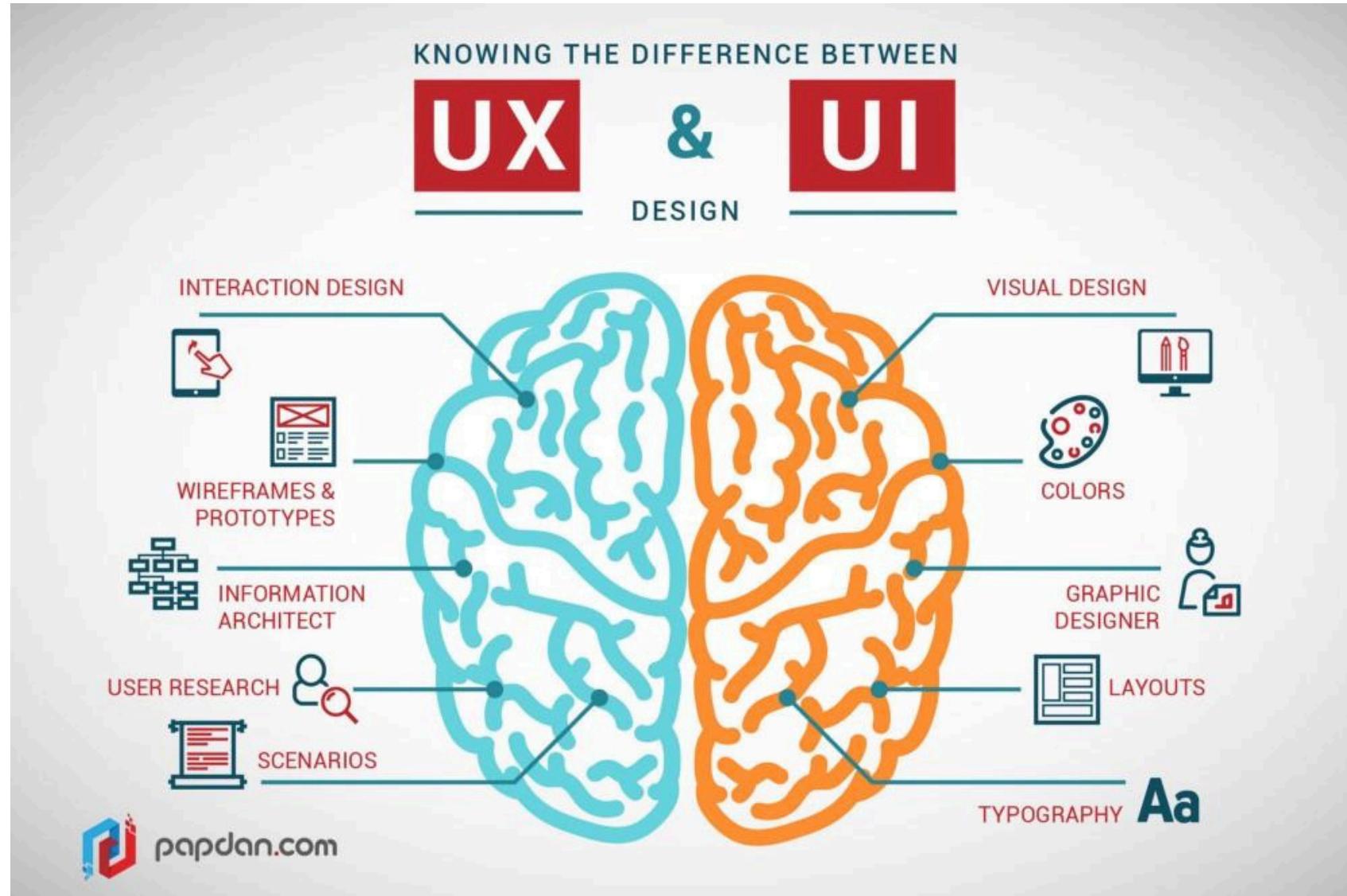
การออกแบบผลลัพธ์
(Output Design)

การตรวจสอบส่วนต่อประสาน (Interface Validation)

- หลังจากสร้างต้นแบบแล้ว จะต้องมีการ **ประเมินว่า เป็นไปตามความต้องการของผู้ใช้ หรือไม่**
- กีเมามุต้อง **เก็บข้อมูลความคิดเห็นของผู้ใช้ เพื่อนำไปปรับปรุงต้นแบบให้สมบูรณ์ที่สุด**
- คุณภาพและการประเมินคุณภาพงานออกแบบซอฟต์แวร์**
 - สามารถใช้ **เทคนิค** เดียวกันในการเก็บรวบรวมข้อมูล ของการ **วิศวกรรมความต้องการ**
 - การสัมภาษณ์
 - การสอบถาม
 - แต่ก็มีงานต้องงานข้อ มูล มาวิเคราะห์



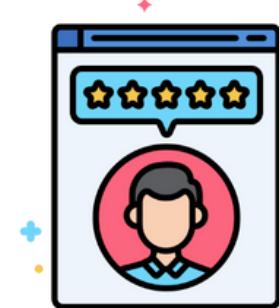
การออกแบบประสบการณ์ของผู้ใช้และการออกแบบส่วนต่อ ประสานกับผู้ใช้(UX & UI Design)



การออกแบบประสบการณ์ของผู้ใช้และการออกแบบส่วนต่อ ประสานกับผู้ใช้(ต่อ)



ความสามารถในการเรียนรู้
(Learnability)

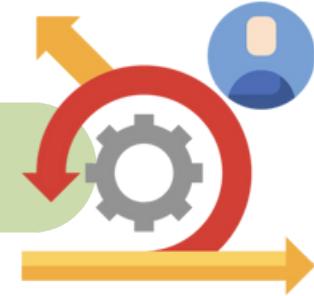


ความพึงพอใจ
(Satisfaction)



ความสามารถในการจดจำ
(Memorability)

ประสิทธิภาพ
(Efficiency)



ประสิทธิผล
(Effectiveness)



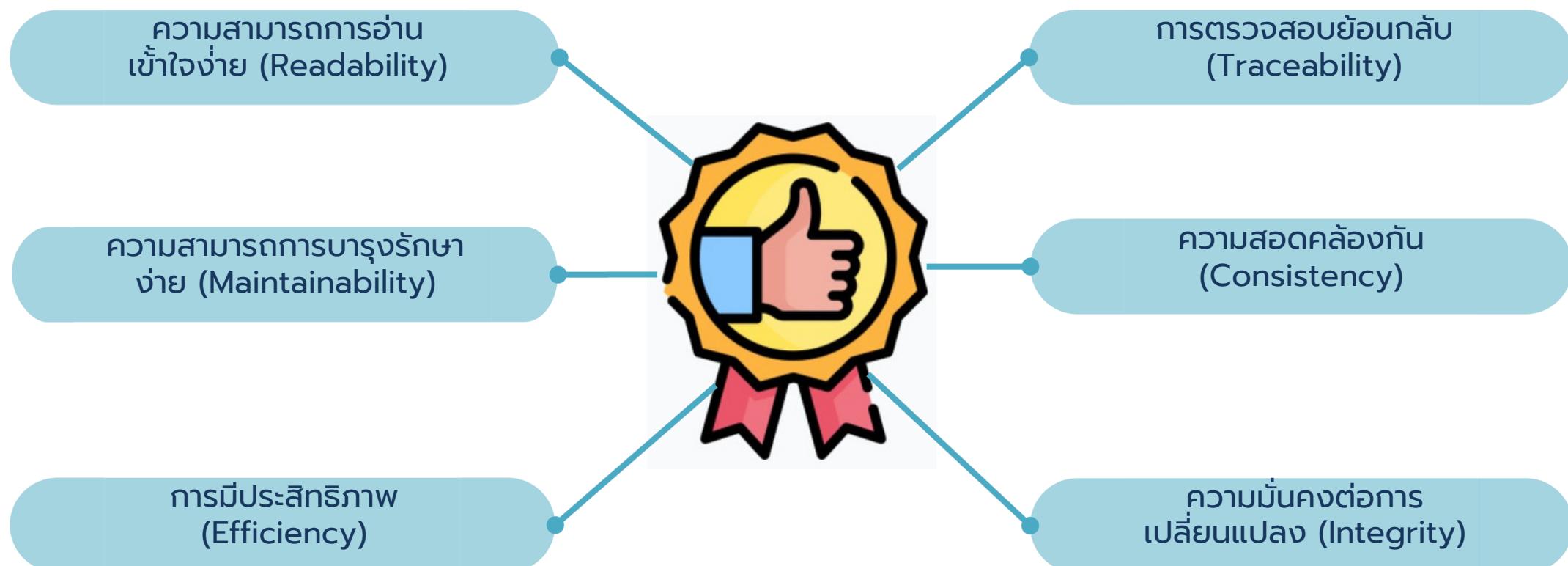
แนวการเขียนโปรแกรมที่ดี

- ทักษะและความรู้พื้นฐาน
 - เข้าใจปัญหาและวิธีแก้ปัญหา
 - ความรู้และทักษะด้านคอมพิวเตอร์
 - ความสามารถในการจัดวางโครงสร้างของโปรแกรม
 - ทักษะในการเขียนคำอธิบายโปรแกรม
 - ความสามารถในการทดสอบและแก้ไขข้อผิดพลาด
- รูปแบบของการเขียนโปรแกรมที่ดี
 - การเขียนผังงาน
 - การเลือกใช้ภาษาสั่งต่าง ๆ ภายในโปรแกรม
 - การย่อหน้าและการเว้นวรรคภายในโปรแกรม
 - การตั้งชื่อไฟล์และตัวแปร
 - เมื่อคำอธิบายโปรแกรม
 - การตรวจสอบและแก้ไขข้อผิดพลาด



แนวทางการเขียนโปรแกรมที่ดี (ต่อ)

- สามารถพิจารณาผลการทำงานของโปรแกรมตามเกณฑ์

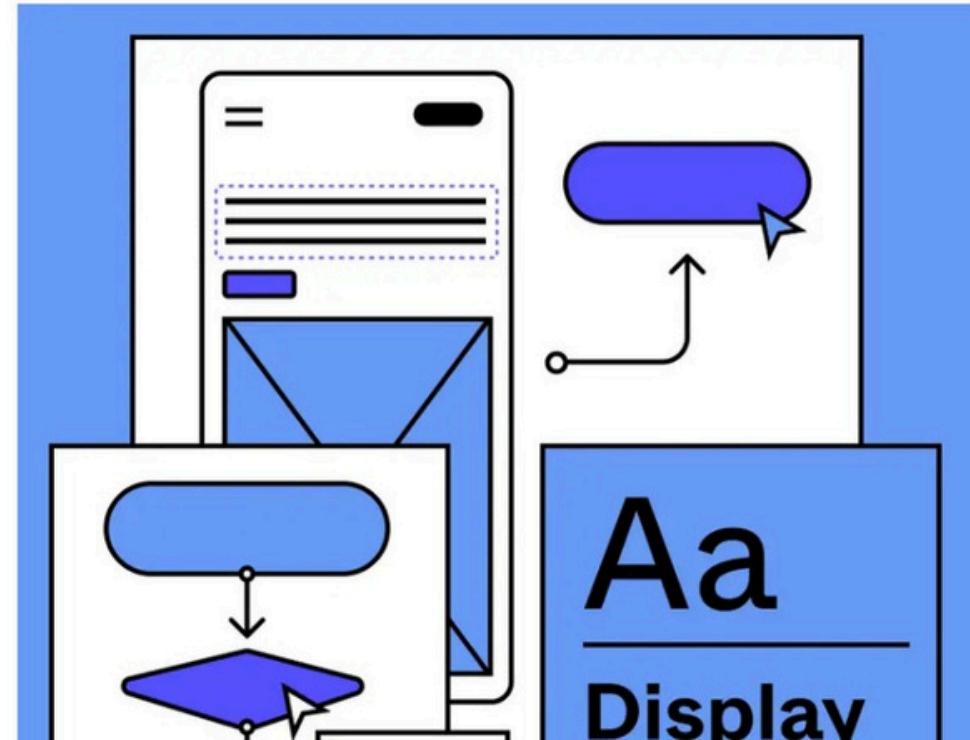


[Products ▾](#)[Enterprise ▾](#)[Pricing](#)[Resources ▾](#)[Community ▾](#)[Log in](#)[Get started for free](#)

UX DESIGN TOOL

Form the experience together

Go from idea to product faster by collaborating in a single, free, web-based design tool.



สรุป (Summary)

- การออกแบบส่วนประสานกับผู้ใช้ (User Interface Design)
- กระบวนการพัฒนาส่วนต่อประสาน
 - การวิเคราะห์และการสร้างแบบจำลองส่วนต่อประสาน (Interface Analysis and Modeling)
 - การออกแบบส่วนต่อประสาน (Interface Design)
 - การสร้างส่วนต่อประสาน (Interface Construction)
 - การตรวจสอบส่วนต่อประสาน (Interface Validation)
- แนวการเขียนโปรแกรมที่ดี

กิจกรรมท้ายบท

- จบออกความสำคัญของการออกแบบส่วนประสานมาพอสั้งเขป
- บัญญัติกฎ 3 ข้อในการออกแบบส่วนประสานของ Theo Mandel มีกี่ข้อ ประกอบด้วย อะไรไรบ้า งพร้อ มกับ อริบ ายแต่ล ะข้อ พอสั้ง เขป
- อะไรคือข้อควรค านึงในการออกแบบ จงอธิบายและยกตัวอย่างประกอบในแต่ละข้อมา พอสั้ง เขป
- รูปแบบของ User Interfaces มีกี่รูปแบบประกอบด้วยอะไรบ้างพร้อมกับอธิบายแต่ละรูปแบบมาพอสั้งเขป
- จบออกแบบที่ประเมินคุณภาพของซอฟต์แวร์ ประกอบด้วยเกณฑ์อะไรบ้างพร้อมกับ อธิบายแต่ละเกณฑ์มาพอสั้งเขป
- ก าหนดให้เคราะห์และออกแบบ ลำดับของการแสดงส่วนติดต่อกับผู้ใช้งานระบบ **กลางการใช้งาน ริก ารห้อ ชั้นพั้กการอินเทอร์เฟซ** โดยใช้แล แผนภาพแสดงลำดับ การ เชื่อมโยงจวาก (Dialogues Diagram)

เอกสารอ้างอิง

-
- น้าfon อ็คเวย์น, หลักการพื้นฐานของวิศวกรรมซอฟต์แวร์ (Fundamentals of Software Engineering), กรุงเทพฯ: ชีเอ็ดยูเคชั่น, 2560.
 - กิตติ ภัสด้วนະกุล, วิศวกรรมซอฟต์แวร์ (Software Engineering), กรุงเทพฯ: เคทีพี คอมพ์ แอนด์ คอนซัลท์, 25xx.
 - Alan Dennis, Barbara Haley Wixom and Roberta M, Systems Analysis and Design Fifth Edition, John Wiley and Sons, Inc. 2012.
 - Lan Sommerville, Software Engineering Ninth Edition, Pearson Education, Inc., publishing as Addison-Wesley, 2011.
 - Roger S. Pressman and Bruce R. Maxim, Software Engineering A Practitioner's Approach Eighth Edition, McGraw-Hill Education, 2015.