88823459
การวิเคราะห์และออกแบบระบบเชิงวัตถุ

พีระศักดิ์ เพียรประสิทธิ์

Outline

- วงจรการพัฒนาระบบ
- แบบจำลองของวงจรการพัฒนาระบบ
  - Waterfall Model
  - Spiral Model
- แผนภาพ Use Case
  - เทคนิคที่ใช้ในการค้นหา Use Case

วัตถุประสงค์การเรียนรู้

- เพื่อเรียนรู้แบบจำลองของวงจรการพัฒนาระบบ
- ข้อดี ข้อเสีย ของแบบจำลองแต่ละแบบ
- เพื่อเข้าใจแผนภาพ Use Case และวิธีการค้นหายูสเคสของระบบ

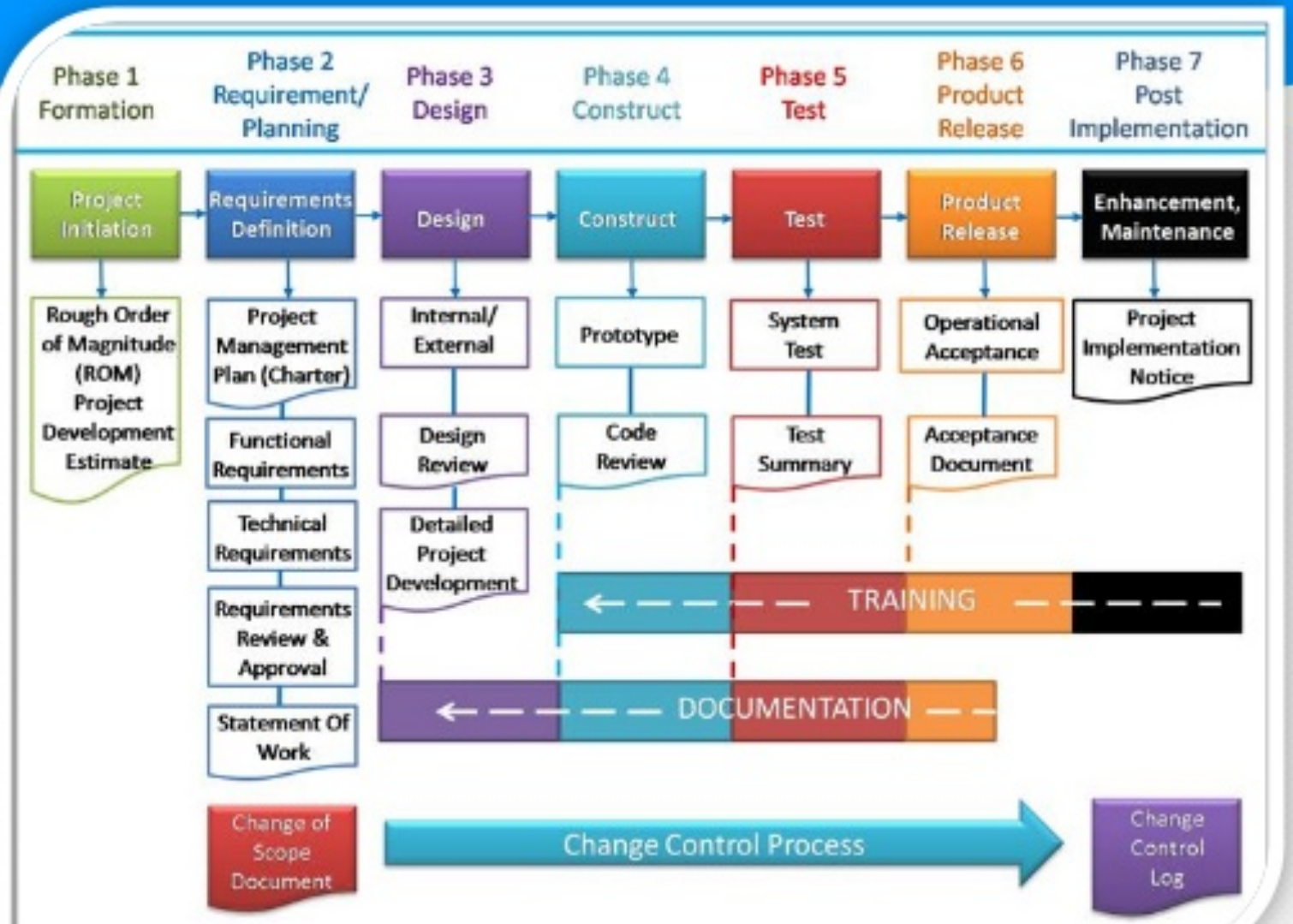
SDLC

- A framework that describes the activities performed at each stage of a software development project.
- กรอบแนวคิดที่อธิบายถึงแต่ละกิจกรรมในแต่ละขั้นตอนของโครงการพัฒนาซอฟต์แวร์

SDLC PHASES

- Requirements Gathering and Analysis
- Design
- Development
- Testing
- Maintenance

SDLC PHASES


Phase 1 Formation
Phase 2 Requirement/ Planning
Phase 3 Design
Phase 4 Construct
Phase 5 Test
Phase 6 Product Release
Phase 7 Post Implementation
Project Initiation
Requirements Definition
Design
Construct
Test
Product Release
Enhancement, Maintenance
Rough Order of Magnitude (ROM) Project Development Estimate
Project Management Plan (Charter)
Functional Requirements
Technical Requirements
Requirements Review & Approval
Statement Of Work
Internal/ External
Design Review
Detailed Project Development
Prototype
Code Review
System Test
Test Summary
Operational Acceptance
Acceptance Document
Project Implementation Notice
TRAINING
DOCUMENTATION
Change of Scope Document
Change Control Process
Change Control Log

SDLC MODELS

- To help understand and implement the SDLC phases various SDLC models have been created by software development experts, universities, and standards organizations.

Reasons for Using SDLC Models

- Provides basis for project planning, estimating & scheduling
- Provides framework for standard set of terminologies, activities & deliverables
- Provides mechanism for project tracking & control
- Increases visibility of project progress to all stakeholders

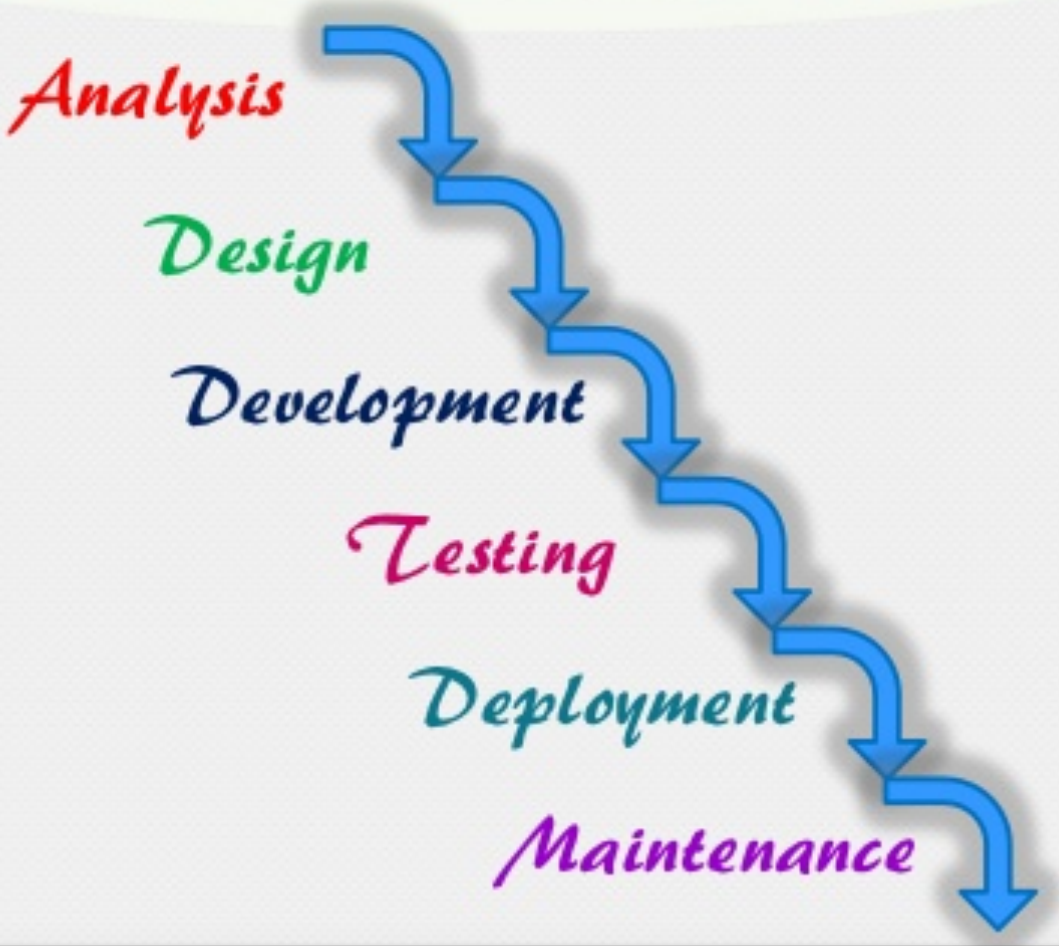
Advantages of Choosing an Appropriate SDLC

- Increased development speed
- Increased product quality
- Improved tracking & control
- Improved client relations
- Decreased project risk
- Decreased project management overhead

Common Life Cycle Models

- Waterfall
- Spiral/Iterative

Waterfall Model



Waterfall Model

- Oldest and most well-known SDLC model
- Follows a sequential step-by-step process from requirements analysis to maintenance.
- Systems that have well-defined and understood requirements are a good fit for the Waterfall Model

Waterfall Model Strengths

- Easy to understand, easy to use
- Provides structure to inexperienced staff
- Milestones are well understood
- Sets requirements stability
- Good for management control (plan, staff, track)
- Works well when quality is more important than cost or schedule

Waterfall Model Weaknesses

- All requirements must be fully specified upfront
- Deliverables created for each phase are considered frozen – inhibits flexibility
- Can give a false impression of progress
- Does not reflect problem-solving nature of software development – iterations of phases
- Integration is one big bang at the end
- Little opportunity for customer to preview the system (until it may be too late)

When to use the Waterfall Model

- Requirements are very well known
- Product definition is stable
- Technology is understood
- New version of an existing product
- Porting an existing product to a new platform.

Spiral/Iterative Model

Spiral Model

- Spiral Model is a “risk-driven” iterative model
- แบบจำลองแบบเกลียวมีแนวคิดว่า การทำงานทุกอย่างล้วนแล้วแต่มีความเสี่ยง (Risk) ทั้งสิ้น และความเสี่ยงสามารถเกิดขึ้นได้ทุกระบบตั้งแต่เริ่มต้นโครงการไปจนกระทั่งจบโครงการ
- Divides a project into iterations
  Each iteration deals with 1 or more risks
- Each iteration starts with small set of requirements and goes through development phase (except Installation and Maintenance) for those set of requirements.

Spiral Model

- Iterate until all major risks addressed and the application is ready for the Installation and Maintenance phase (production)
- Each of the iterations prior to the production version is a prototype of the application.
- Last iteration is a waterfall process

ขั้นตอนการพัฒนาซอฟต์แวร์ตามแบบจำลองแบบเกลียว (Spiral Model)

- 1. ขั้นตอนของการวางแผนการดำเนินงาน ในแต่ละวงรอบ ในที่นี้จะรวมถึงการปรับแผนให้เหมาะสมกับสถานการณ์ปัจจุบันด้วย
- 2. ขั้นตอนการจัดการความเสี่ยง ในขั้นตอนนี้จะมีการสร้างโปรแกรมต้นแบบเพื่อใช้เป็นเครื่องมือในการวิเคราะห์และกำหนดแนวทางในการจัดการความเสี่ยงที่อาจเกิดขึ้นในแต่ละวงรอบ
- 3. ขั้นตอนการพัฒนาและทดสอบระบบ เป็นการนำผลที่ได้จากขั้นตอนที่ 1 และ 2 มาพัฒนาระบบแต่ละเวอร์ชัน จากนั้นจึงทดสอบระบบตามหลักการวิศวกรรมซอฟต์แวร์
- 4. ขั้นตอนการประเมินผลระบบ จะทำโดยผู้เกี่ยวข้องทุกฝ่าย จากนั้นจึงติดตั้งเพื่อใช้งานระบบเวอร์ชันนั้นๆ

Spiral Model Strengths

- Provides early indication of insurmountable risks, without much cost
- Critical high-risk functions are developed first
- The design does not have to be perfect
- Users see the system early because of rapid prototyping tools
- Users can be closely tied to all lifecycle steps Early and frequent feedback from users

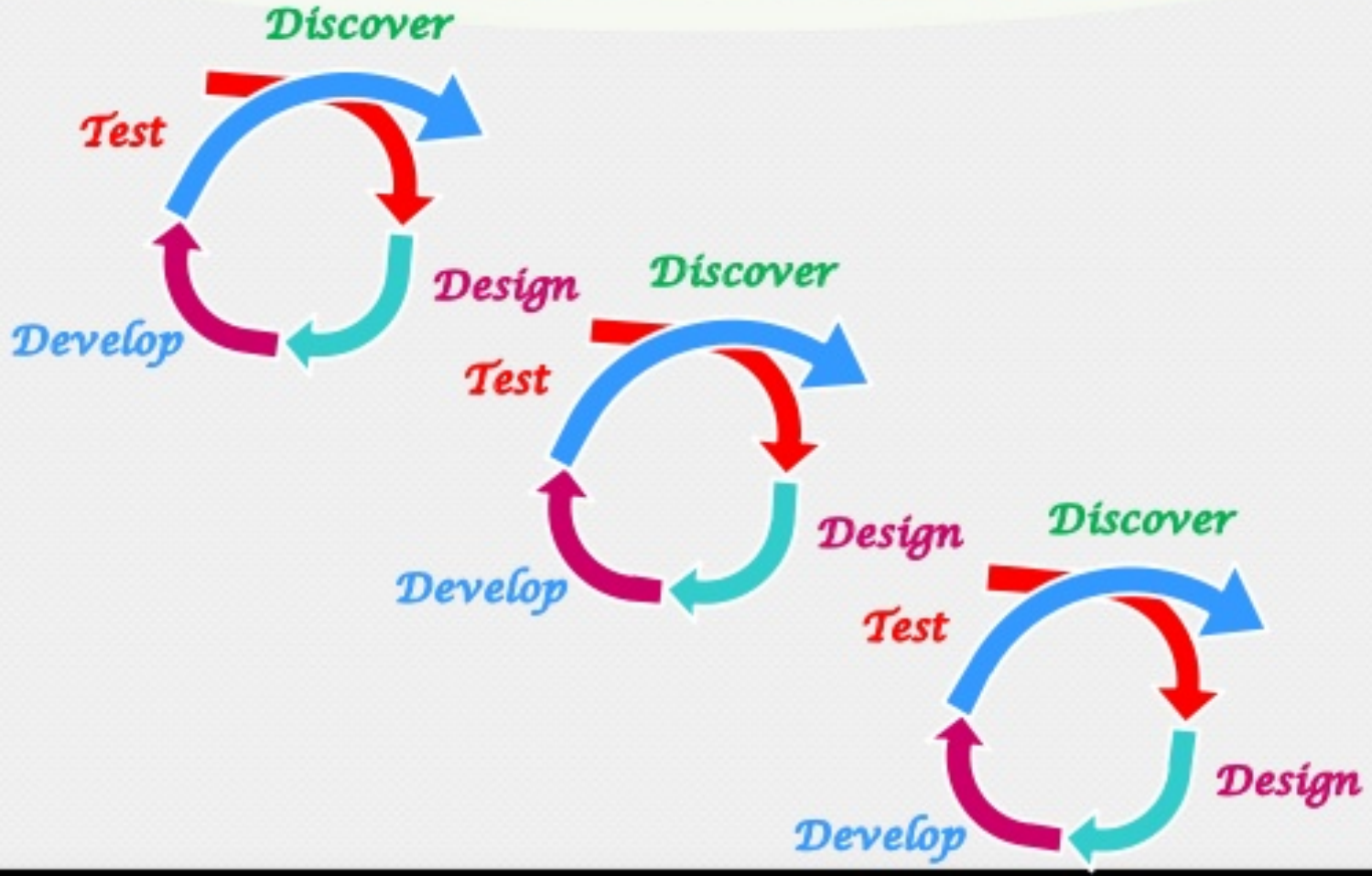
Spiral Model Weaknesses

- Time spent for evaluating risks too large for small or low-risk projects
- Time spent planning, resetting objectives, doing risk analysis and prototyping may be excessive
- The model is complex
- Risk assessment expertise is required Spiral may continue indefinitely
- May be hard to define objective, verifiable milestones that indicate readiness to proceed through the next iteration

When to use Spiral Model

- When creation of a prototype is appropriate
- When costs and risk evaluation is important For medium to high-risk projects
- Users are unsure of their needs Requirements are complex
- New product line
- Significant changes are expected (research and exploration)

Agile Model


Discover
Test
Design
Develop
Discover
Test
Design
Develop
Discover
Test
Design
Develop

ขั้นตอนการพัฒนาซอฟต์แวร์ตามแบบเอไจล์

- 1. การวางแผน เป็นการแบ่งความต้องการใช้งานออกเป็นส่วนย่อยและเรียงลำดับความสำคัญก่อน – หลัง ร่วมกับผู้ที่มีอำนาจในการตัดสินใจขององค์กร เพื่อให้สามารถส่งมอบงานได้ในระยะเวลาสั้น ๆ และมีการจัดทำแผนการดำเนินงานโครงการระยะสั้นตามวงรอบของการทำงาน
- 2. การรวบรวมข้อมูล เป็นการรวบรวมข้อมูลความต้องการใช้งานระบบที่จำเป็นต่อการพัฒนาระบบย่อยในรอบนั้น ๆ ซึ่งในบางครั้งอาจต้องอ้างอิงกลับไปยังระบบย่อยที่ได้ส่งมอบงานไปแล้วในกรณีที่มีการทำงานที่สัมพันธ์กัน
- 3. การพัฒนาระบบ เป็นการออกแบบและพัฒนาระบบย่อยในรอบนั้น ๆ จากนั้นทีมงานพัฒนาระบบจะดำนเนินการทดสอบระบบตามหลักการวิศวกรรมซอฟต์แวร์
- 4. การทดสอบระบบ เป็นการติดตั้งและทดสอบระบบร่วมกับผู้ใช้งาน กรณีที่ผู้ใช้ยอมรับระบบย่อยนั้น ๆ ก็สามารถใช้งานได้ทันที แต่ถ้ามีส่วนที่ต้องปรับแก้ก็จะต้องนำไปเพิ่มไว้ในแผนการดำเนินงานในวงรอบถัดไป

Agile

- Speed up or bypass one or more life cycle phases
- Usually less formal and reduced scope
- Used for time-critical applications
- Used in organizations that employ disciplined methods

Some Agile Methods

- Rapid Application Development (RAD)
- Scrum
- Extreme Programming (XP)
  Adaptive Software Development (ASD)
- Feature Driven Development (FDD)
- Crystal Clear
- Dynamic Software Development Method (DSDM)
- Rational Unify Process (RUP)

Agile Model Strengths

- Deliver a working product faster than conventional linear development model
- Customer feedback at every stage ensures that the end deliverable satisfies their expectations
- No guesswork between the development team and the customer, as there is face to face communication and continuous inputs from the client

Agile Model Weaknesses

- For larger projects, it is difficult to judge the efforts and the time required for the project in the SDLC.
- Since the requirements are ever changing, there is hardly any emphasis, which is laid on designing and documentation. Therefore, chances of the project going off the track easily are much more

ระเบียบวิธี Object – Oriented Analysis and Design (OOAD)

- ระเบียบวิธี Object – Oriented Analysis and Design เป็นวิธีการพัฒนาระบบที่ใช้มุมมองตามหลักการธรรมชาติว่า ทุกองค์ประกอบของระบบล้วนแล้วแต่เป็นวัตถุ (Object) โดยมีคุณสมบัติหลัก 3 ประการคือ
  - 1. ชื่อเฉพาะ (Unique Identifier) เช่น ลูกค้า ผู้ขาย พนักงาน สินค้า คลังสินค้า และรถ เป็นต้น
  - 2. คุณลักษณะ (Attribute) เป็นส่วนที่อธิบายลักษณะของวัตถุนั้น ๆ เช่น รถจะมีคุณลักษณะของยี่ห้อ รุ่น สี จำนวนที่นั่ง และหมายเลขตัวถังรถ เป็นต้น
  - 3. สถานะ (State) เช่น รถใหม่ รถอยู่ระหว่างการใช้งาน รถอยู่ระหว่างการซ่อมบำรุง และรถถูกยกเลิกการใช้งาน เป็นต้น

ระเบียบวิธี Object – Oriented Analysis and Design (OOAD) (ต่อ)

- แต่ละวัตถุจะแสดงพฤติกรรมที่เกิดจากการเรียกวิธีการ (Method/Operation) เพื่อเปลี่ยนแปลงสถานะของวัตถุ ณ ช่วงเวลาใดๆ
- วัตถุต่าง ๆ ในระบบจะทำงานสัมพันธ์กับโดยการส่งสาร (Message) ระหว่างกัน กลุ่มของวัตถุที่มีคุณลักษณะพื้นฐานเหมือนกันจะถูกรวบรวมไว้ภายในคลาส (Class) เดียวกัน
- ขั้นตอนการวิเคราะห์และออกแบบเชิงวัตถุสามารถใช้แบบจำลองเชิงวัตถุ (Object – Oriented Model) ที่เป็นมาตรฐานคือ Unified Modeling Language (UML) ซึ่งเป็นรูปแบบของภาษาที่มีสัญลักษณ์ (Notation) สำหรับสื่อความหมายเชิงวัตถุ เพื่อใช้ในการพัฒนาโปรแกรมเชิงวัตถุได้อย่างต่อเนื่องและสอดคล้องกัน

UML

- UML ประกอบด้วยแบบจำลองหลายประเภท เช่น
  - Use Case Diagram เป็นแผนภาพแสดงความสัมพันธ์ระหว่างผู้ใช้กับระบบที่จะพัฒนา หรือองค์ประกอบภายนอกและระบบ เพื่อให้เห็นขอบเขตการทำงานในภาพรวมของระบบทั้งหมด
  - Class Diagram เป็นแผนภาพแสดงองค์ประกอบของแต่ละคลาส ทั้งคุณลักษณะและวิธีการ รวมไปถึงความสัมพันธ์ระหว่างคลาสต่าง ๆ
  - Sequence Diagram เป็นแบบจำลองเชิงพฤติกรรม (Behavioral Model) ของการทำกิจกรรมหนึ่ง ๆ โดยแสดงถึงขั้นตอนการทำงานและลำดับของการสื่อสาร (Message) ระหว่างวัตถุที่โต้ตอบกัน

แบบจำลอง Use Case

- Use case คือกิจกรรมที่ระบบดำเนินการ มักอยู่ในรูปแบบการตอบสนองต่อการร้องขอของผู้ใช้งาน
- Use case ใช้ อธิบาย/กำหนด/ระบุ ความต้องการที่เป็นฟังก์ชัน
- นักวิเคราะห์จำแนกหน้าที่ของระบบ ออกเป็นกลุ่มของ use case โดยใช้ 2 เทคนิค
  - User goal technique
  - Event decomposition technique
- ชื่อของ use case ให้ใช้คำกริยานำหน้าคำนาม

User Goal Technique

- เทคนิคเป็นที่นิยมและใช้งานบ่อยครั้ง เนื่องจากง่ายและมีประสิทธิภาพ
- วิธีการ
  - ระบุประเภทของผู้ใช้งานระบบที่อาจเกิดขึ้นได้
  - สัมภาษณ์/สอบถาม ผู้ใช้งานระบบ ว่าต้องการใช้คอมพิวเตอร์ช่วยทำงานอะไรให้ผู้ใช้งาน
  - ตรวจสอบ/ปรับแต่ง การทำงานให้เข้ากับเป้าหมายของผู้ใช้งาน เช่น ต้องการจัดส่งสินค้า ตรวจสอบสถานะการขนส่ง และการส่งสินค้าคืน

User Goal Technique:

Specific Steps

- 1. ระบุผู้ใช้งานที่อาจเกิดขึ้นในระบบงาน
- 2. แบ่งประเภทผู้ใช้งานในแง่ของ บทบาทการทำงาน/หน้าที่ เช่น การจัดส่งสินค้า การตลาด การขาย
- 3. นอกจากนี้ แบ่งประเภทผู้ใช้งานตามโครงการองค์กร เช่น ระดับปฏิบัติการ ระดับบริหาร
- 4. สัมภาษณ์ ผู้ใช้งานแต่ละกลุ่ม เพื่อค้นหาเป้าหมายเฉพาะ (specific goals) เมื่อพวกเขาใช้งานระบบใหม่ (ทั้งเป้าหมายปัจจุบันและนวัตกรรมใหม่เพื่อเพิ่มมูลค่า)

User Goal Technique:

Specific Steps

- 5. สร้าง use case เบื้องต้นของแต่ละประเภทผู้ใช้งาน
- 6. ค้นหา use case ที่มีชื่อซ้ำหรือคล้ายกัน และทำการแก้ความไม่สอดคล้อง
- 7. ระบุการใช้งาน use case ของแต่ละประเภทผู้ใช้งาน ซึ่งบาง use case อาจมีการใช้งานจากหลายประเภทผู้ใช้งาน
- 8. ทบทวนรายการ use case ของแต่ละประเภทผู้ใช้งาน

Event Decomposition Technique

- เทคนิคในการค้นหา use case
  - ระบุเหตุการณ์ที่ระบบจะต้องทำการตอบสนอง
  - ในแต่ละเหตุการณ์ use case จะเป็นทำหน้าที่ในการบรรยายถึงสิ่งที่ระบบกระทำ เมื่อมีเหตุการณ์เกิดขึ้น
- เหตุการณ์ (Event) – บางสิ่งบางอย่างที่เกิดขึ้นในเวลาหรือสถานที่ที่กำหนดขึ้น
  - ระบบควรจดจำเหตุการณ์ไว้ได้

ประเภทของเหตุการณ์ (Types of Events)

- เหตุการณ์ภายนอก (External Event)
  - เหตุการณ์ที่เกิดขึ้นนอกระบบมักจะกระทำโดยตัวแทนภายนอก (external agent) หรือ Actor
- เหตุการณ์ชั่วคราว (Temporal Event)
  - เหตุการณ์ที่เกิดขึ้นเมื่อถึงเวลาที่ระบุไว้
- State Event
  - เหตุการณ์ที่เกิดขึ้นเมื่อบางสิ่งเกิดขึ้นในระบบและ ก่อให้เกิด/เรียกใช้งาน โปรเซสงานบางโปรเซส

External Event Checklist

- ตัวแทนภายนอกหรือ Actor ต้องการผลลัพธ์บางอย่างของการทำธุรกรรม (transaction) เช่น
  - ลูกค้าซื้อผลิตภัณฑ์
- ตัวแทนภายนอกหรือ Actor ต้องการข้อมูลบางอย่าง เช่น
  - ลูกค้าต้องการทราบรายละเอียดสินค้า
- เกิดการเปลี่ยนแปลงข้อมูลจากภายนอกและต้องการปรับปรุงข้อมูลในระบบ
  - ลูกค้าเปลี่ยนที่อยู่หรือเบอร์โทรศัพท์ใหม่
- ผู้บริหารต้องการข้อมูลบางอย่าง
  - ผู้จัดการฝ่ายขายต้องการปรับปรุง/แก้ไข แผนการผลิต

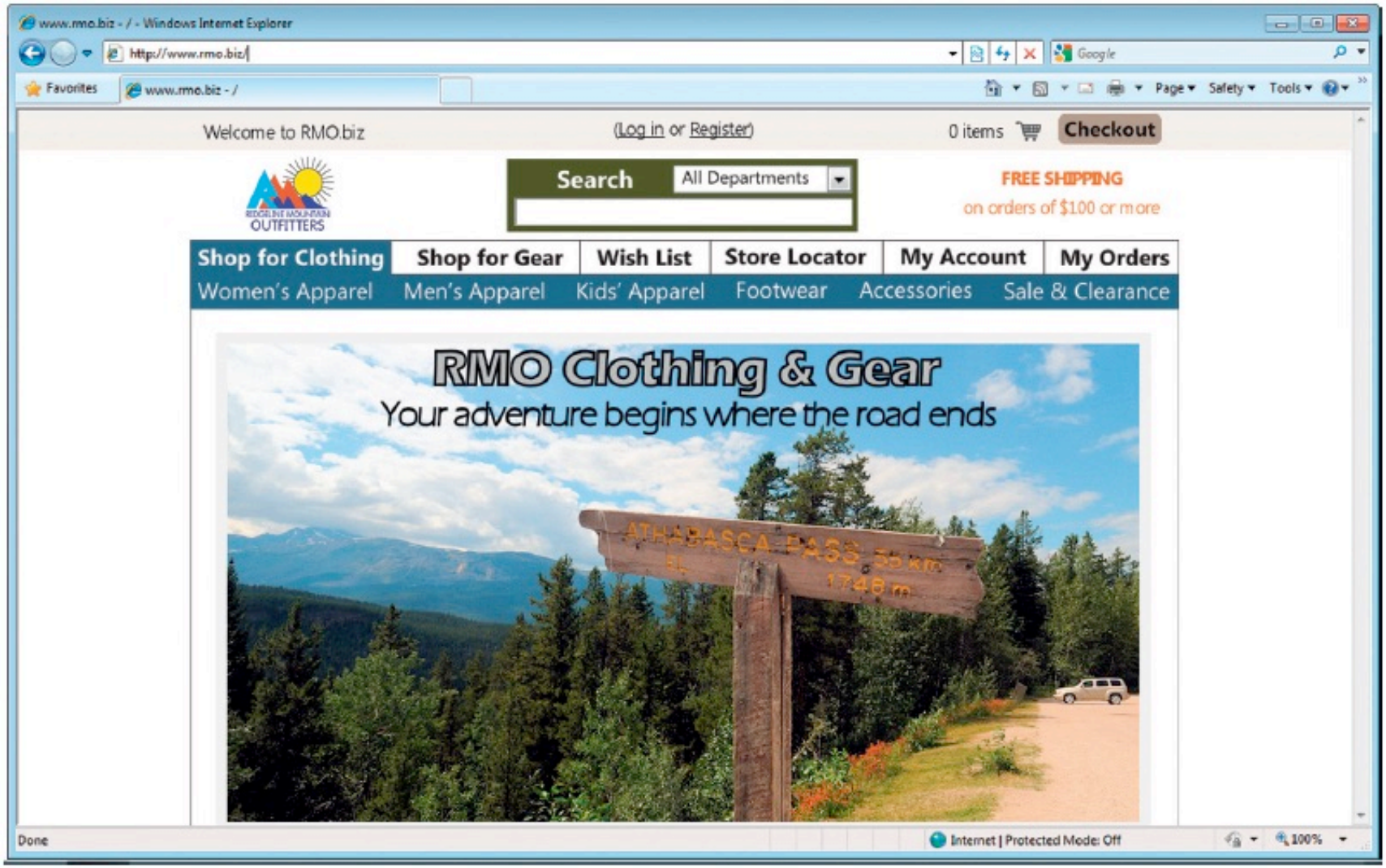
Temporal Event Checklist

- Internal outputs ที่ต้องการในบางช่วงเวลา
  - รายงานการบริหาร (Management reports) เช่น
  - รายงานการดำเนินงาน (Operational reports) เช่น รายละเอียดของการทำธุรกรรม
  - รายงานงบของบริษัท เช่น เงินเดือนของพนักงาน
- External outputs ที่ต้องการในบางช่วงเวลา
  - รายงานงบดุลของบริษัท
  - รายงานสถานะ
  - ใบแจ้งหนี้/ใบแจ้งค่าใช้จ่าย

กรณีศึกษา

- บริษัท Ridgeline Mountain Outfitters (RMO) เป็นบริษัทค้าปลีกขนาดใหญ่ จำหน่ายอุปกรณ์กลางแจ้ง อุปกรณ์กีฬา อุปกรณ์สกี จักรยาน อุปกรณ์ตั้งแคมป์ และอื่นๆ
- การสั่งซื้อ ลูกค้าสามารถสั่งซื้อได้ผ่านทางไปรษณีย์ หรือโทรศัพท์สั่งซื้อ
- มีความต้องการขยายกิจการให้ทำธุรกรรมผ่านอิเล็กทรอนิกส์เพื่อเพิ่มรายได้จากการขายสินค้า

Ridgeline Mountain Outfitters (RMO)



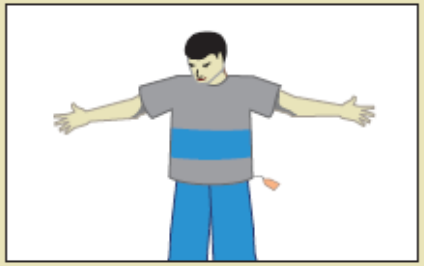
Finding the actual event that affects the system

Customer thinks about getting a new shirt

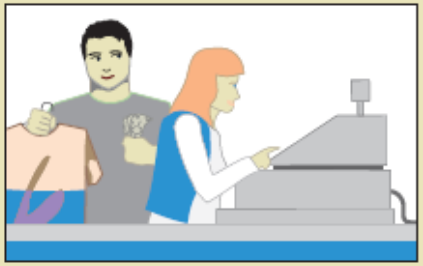
Customer drives to the mall

Customer tries on a shirt at Sears

Customer goes to Walmart



Customer tries on a shirt at Walmart



Customer buys a shirt

(the event that directly affects the system!)

Tracing a sequence of transactions resulting in many events

- การติดตามลำดับของเหตุการณ์ที่เกิดขึ้นในหลายๆ เหตุการณ์

Customer requests a catalog


Customer wants to check item availability


Customer places an order


Customer changes or cancels an order


Customer wants to check order status

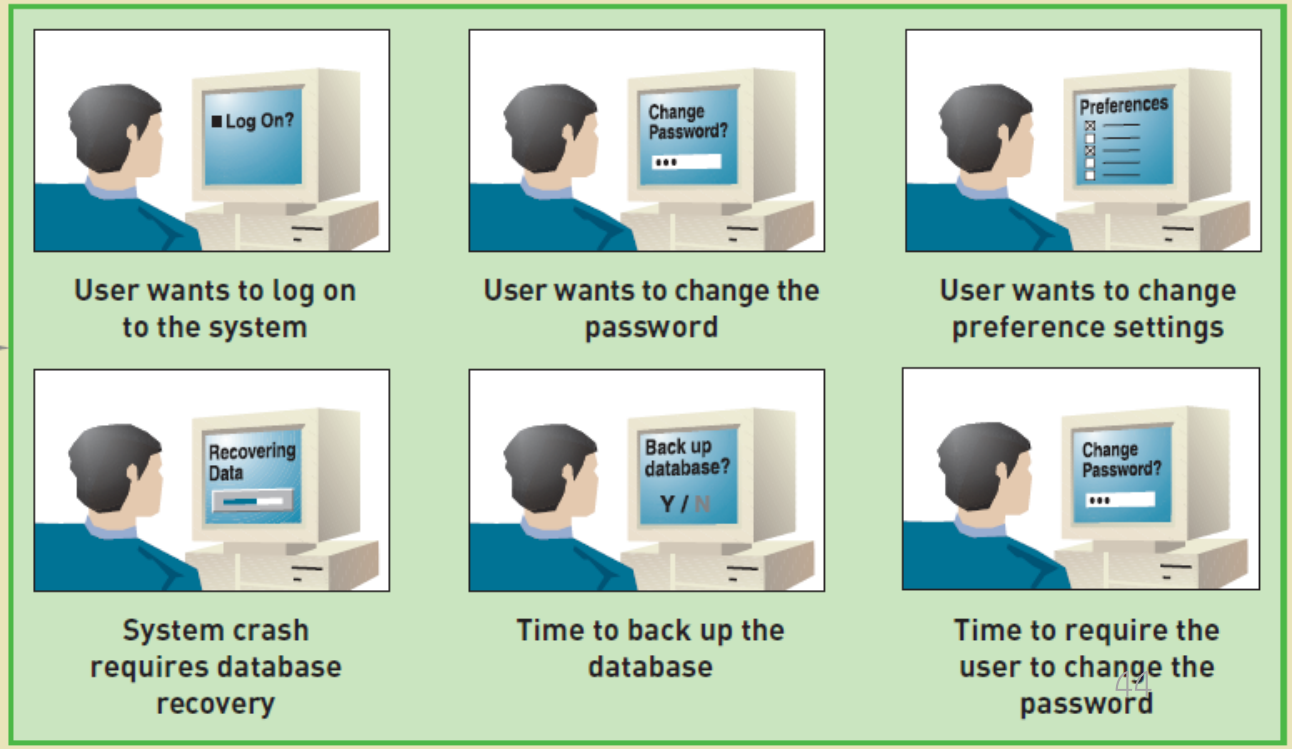
Customer updates account information

Customer returns the item

Perfect Technology Assumption

- Don’t worry about functions built into system because of limits in technology and people. Wait until design.

Don’t worry much about these until you are considering design issues



Event Decomposition Technique:

Specific Steps

- 1. พิจารณาเหตุการณ์ภายนอก (external events) ในสภาพแวดล้อมของระบบ (system environment) ที่ต้องมีการตอบสนองจากระบบโดยใช้รายการตรวจสอบ (checklist)
- 2. ในแต่ละเหตุการณ์ให้ทำการระบุชื่อของ use case ที่ระบบต้องการ
- 3. พิจารณาเหตุการณ์ชั่วคราว (temporal events) ที่ต้องมีการตอบสนองจากระบบ
- 4. ในแต่ละเหตุการณ์ชั่วคราว (temporal events) ให้ทำการระบุชื่อของ use case ที่ระบบต้องการ และเวลาที่มีการเรียกใช้งาน use case นั้น

Event Decomposition Technique:

Specific Steps (ต่อ)

- 5. พิจารณา state events ที่ระบบทำการตอบสนอง โดยเฉพาะอย่างยิ่งสำหรับระบบที่เป็น real-time ที่ internal state events เรียกใช้งาน use case
- 6. ในแต่ละ state event ให้ทำการระบุชื่อของ use case ที่ระบบต้องการ และกำหนดสถานะที่เปลี่ยนแปลง
- 7. ตรวจสอบ events และ use case ที่ได้กำหนดขึ้นมา ว่าเป็นที่ต้องการใช้งาน/จำเป็น ในระบบจริงๆ โดยไม่นับรวมเหตุการณ์ที่เกี่ยวกับกับการควบคุมระบบ เช่น login, logout, change password, backup or restore database เนื่องจากจะนำเหตุการณ์จำพวกนี้มาใส่เพิ่มทีหลัง

Use Cases and

Brief Use Case Descriptions

- Use case descriptions คือ คำอธิบายที่แสดงให้เห็นถึงขั้นตอนหลักของ use case นั้นๆ

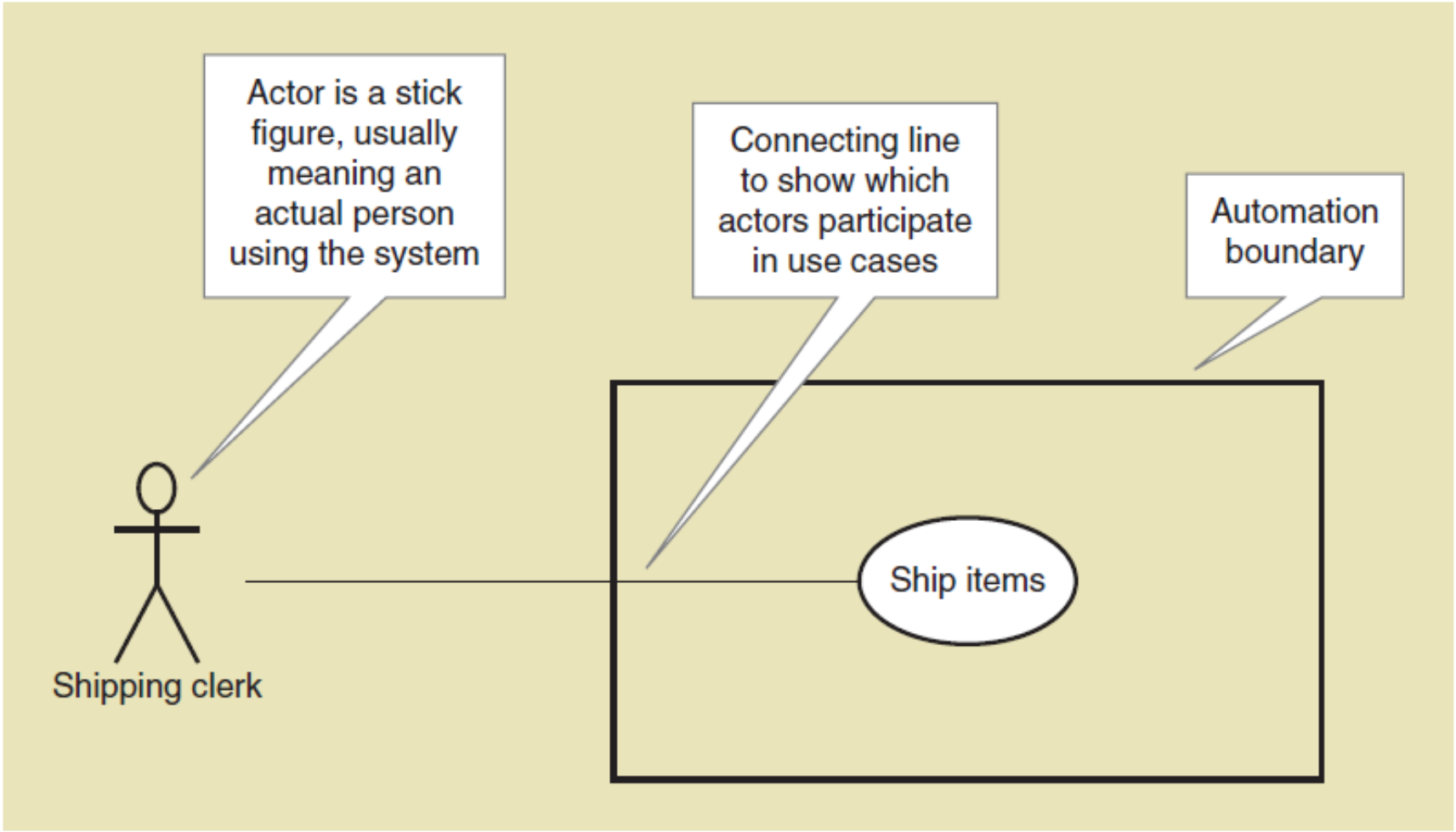
Use case	Brief use case description
Create customer account	User/actor enters new customer account data, and the system assigns account number, creates a customer record, and creates an account record.
Look up customer	User/actor enters customer account number, and the system retrieves and displays customer and account data.
Process account adjustment	User/actor enters order number, and the system retrieves customer and order data; actor enters adjustment amount, and the system creates a transaction record for the adjustment.

Use Case Diagrams

- แผนภาพ use case เป็นหนึ่งในแบบจำลอง UML ที่ใช้แสดงถึงความสัมพันธ์ระหว่าง use case และ actor
- UML is Unified Modeling Language, the standard for diagrams and terminology for developing information systems
- Actor ใน UML หมายถึงผู้ใช้งานระบบ (end user)
- Automation boundary - ขอบเขตระหว่างโปรแกรมคอมพิวเตอร์กับผู้ใช้งานแอพพลิเคชัน

Use Case Diagrams

Symbols

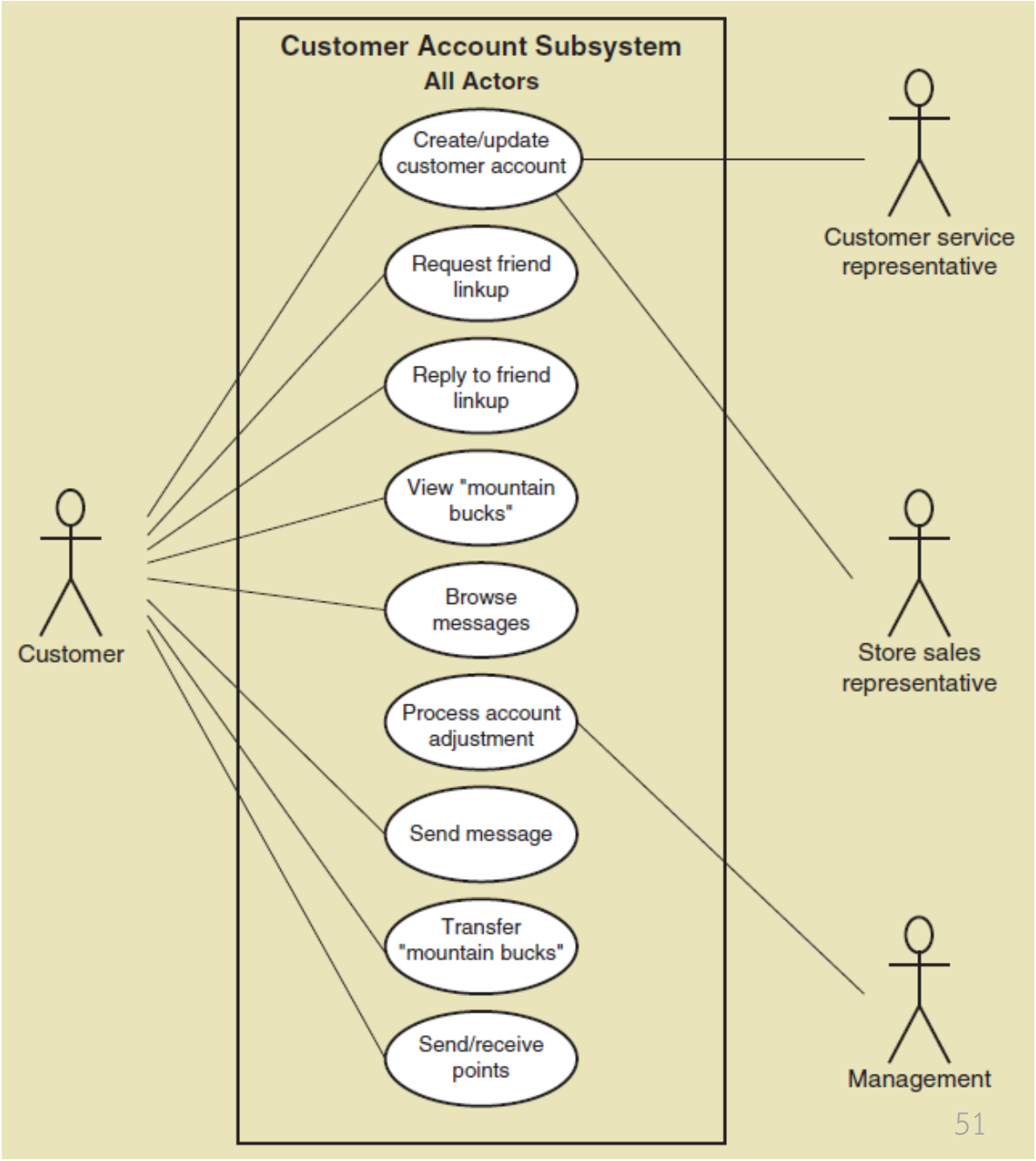


การสร้างแผนภาพยูสเคส

- แบ่งระบบย่อย เช่น ระบบจัดการลูกค้า ระบบการขาย
- แต่ละระบบย่อย
  - กำหนด Actor และ Use Case
  - กำหนดความสัมพันธ์ระหว่าง Actor และ Use Case

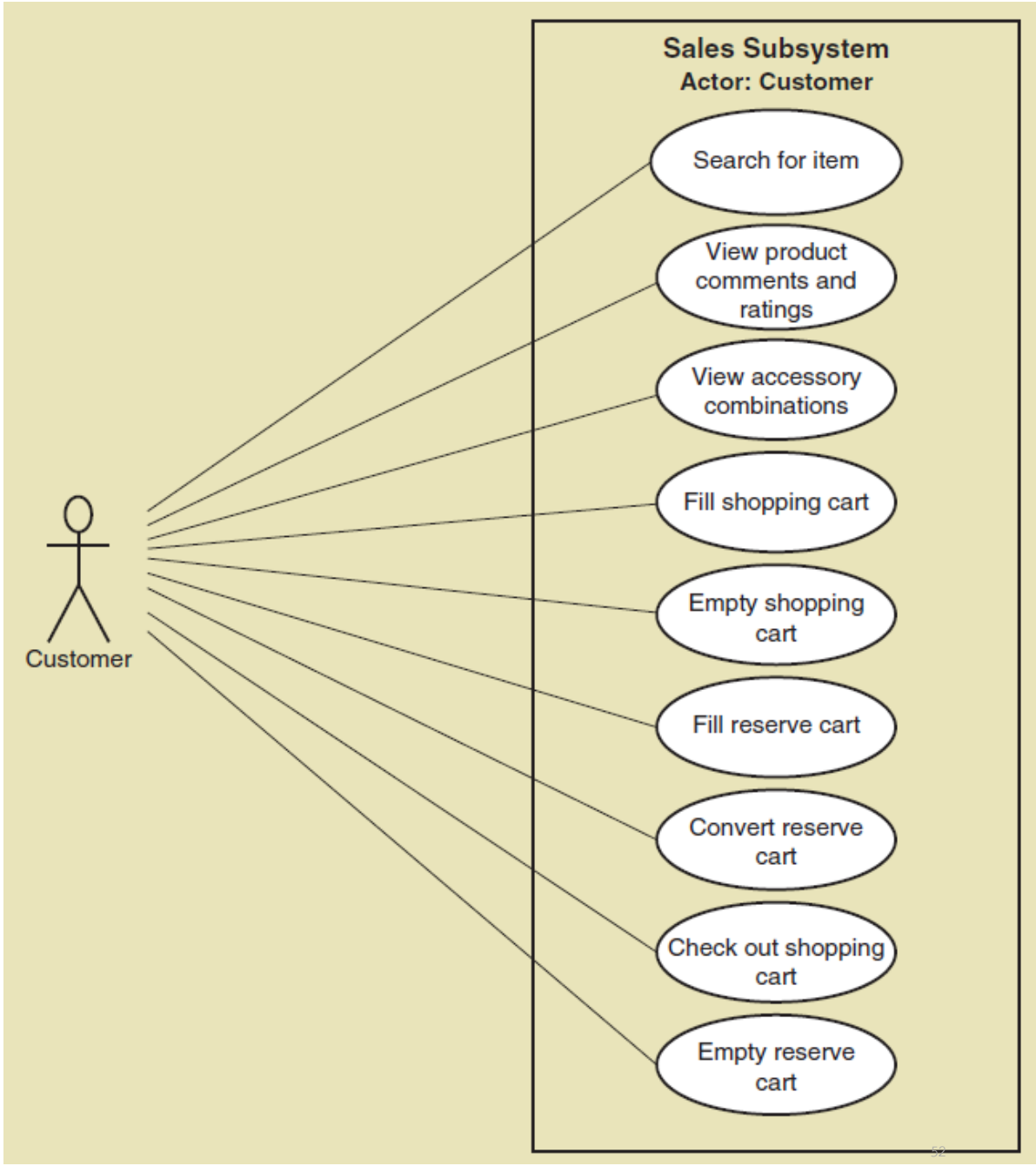
Use Case Diagram

Draw for each subsystem

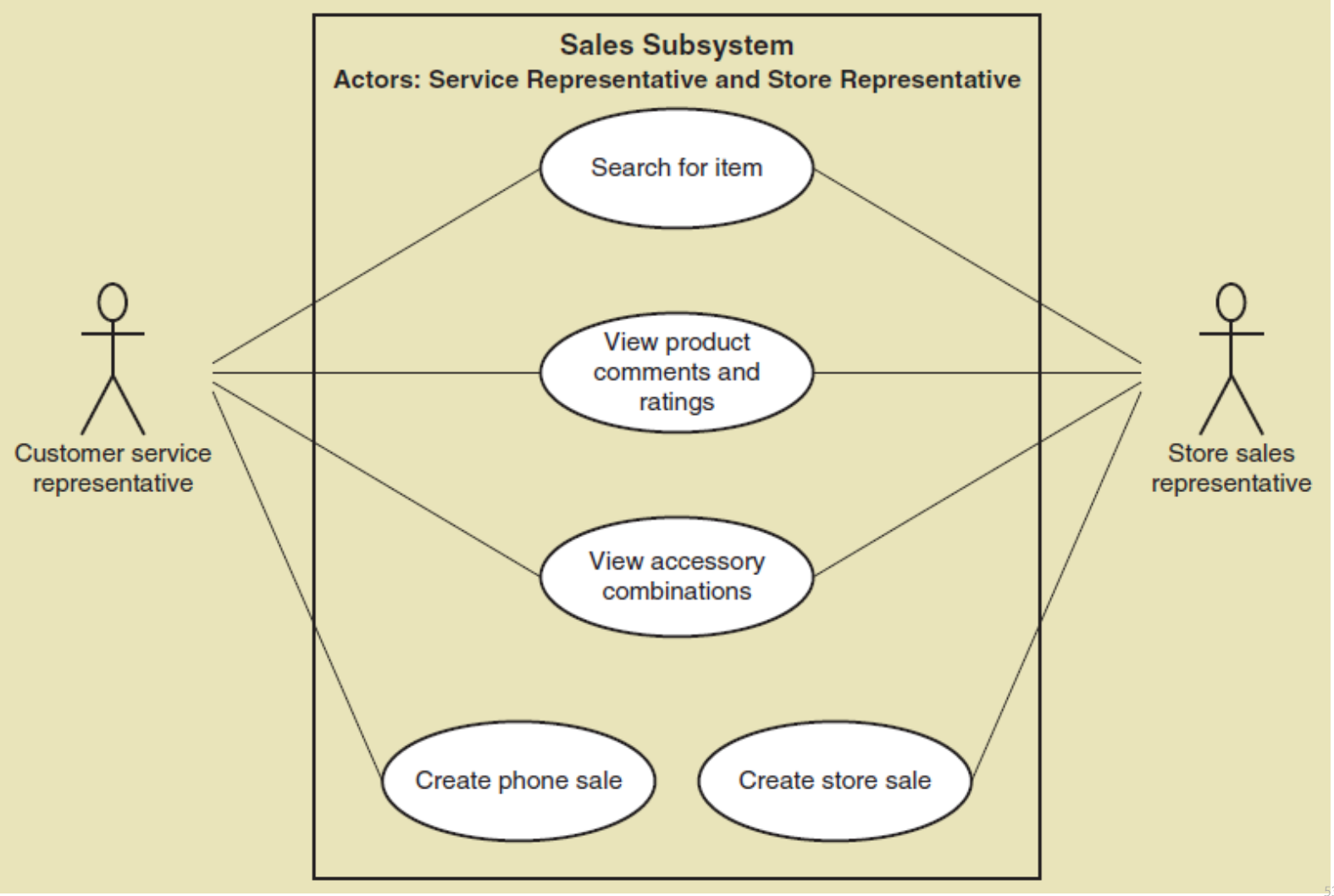


Use Case Diagram

Draw for actor such as customer

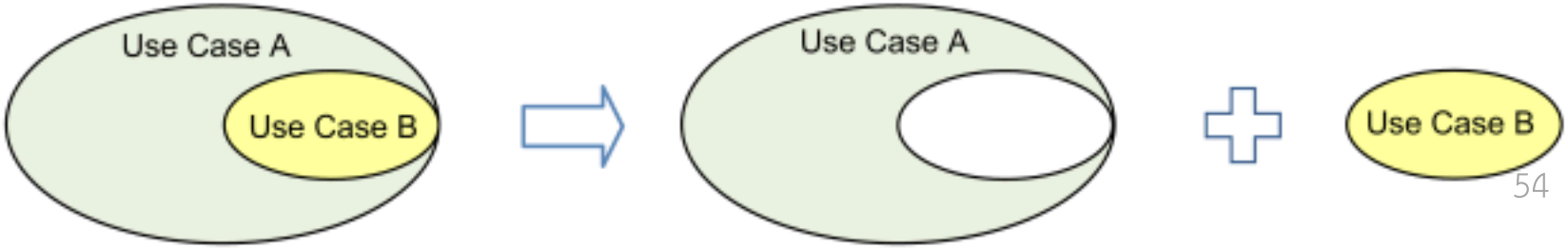


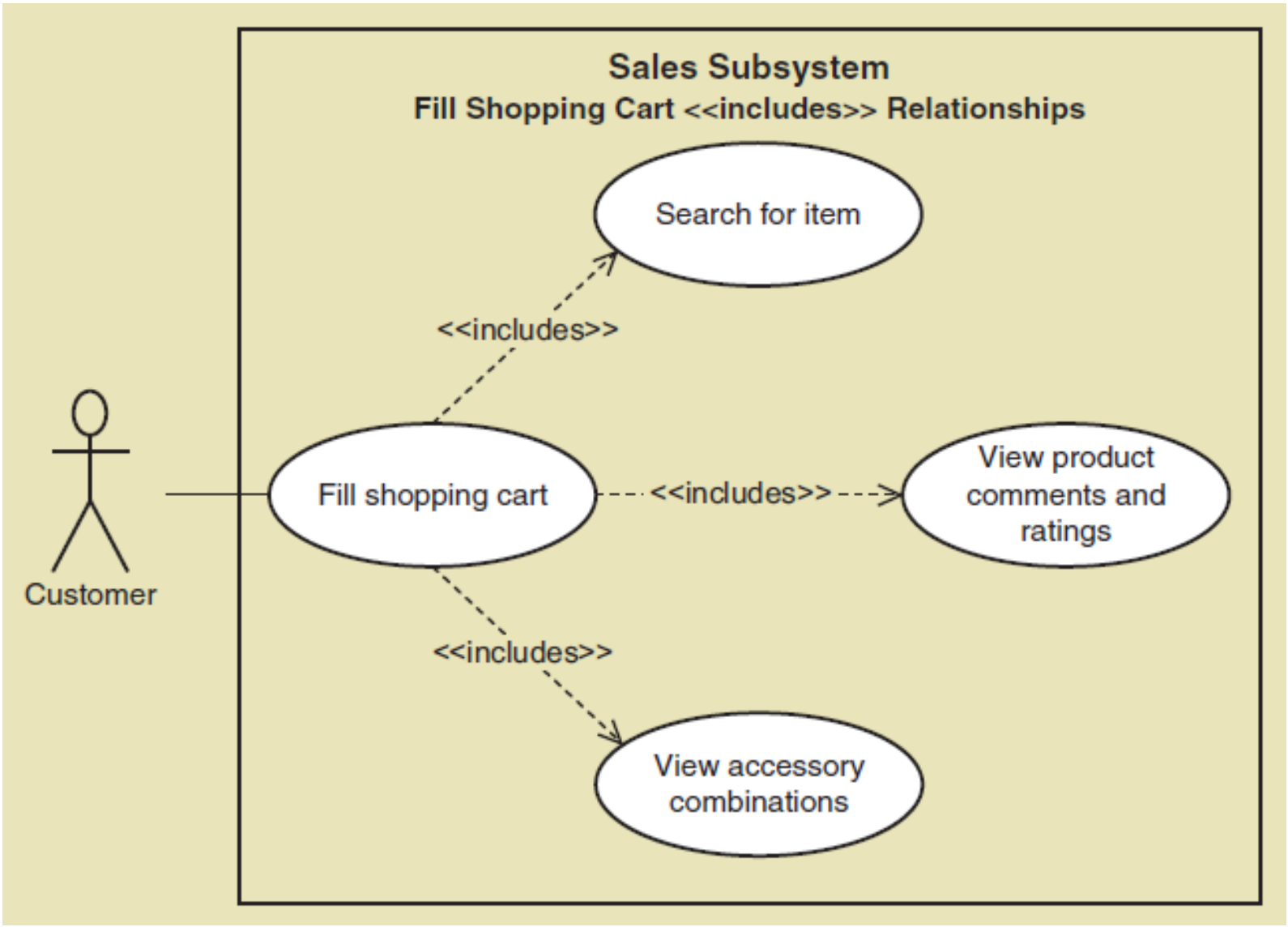
Use Case Diagrams



Use Case Diagrams : ความสัมพันธ์แบบ includes

- ความสัมพันธ์ระหว่าง use case ที่แสดงถึง use case หนึ่งที่ใช้งาน use case หนึ่ง
- คล้ายกับการเรียกโปรแกรมย่อยจากโปรแกรมหลัก
- หัวลูกศรชี้ไปที่ use case ที่เป็น use case ย่อย
- ตย. Use Case A มีความสัมพันธ์กับ Use Case B แบบ includes



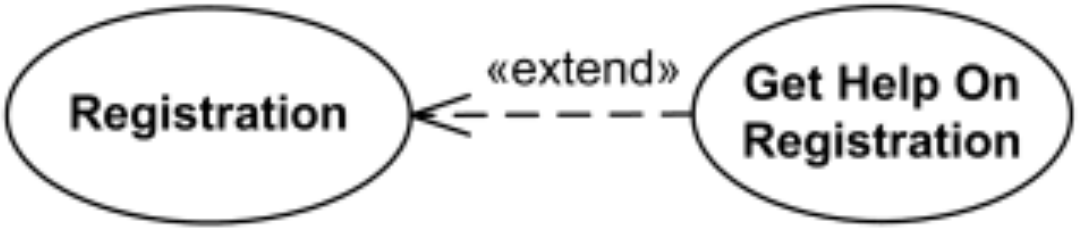

Sales Subsystem
Fill Shopping Cart <<includes>> Relationships
Search for item
<<includes>>
Fill shopping cart
<<includes>>
View product comments and ratings
Customer
<<includes>>
View accessory combinations

Use Case Diagrams : ความสัมพันธ์แบบ extend

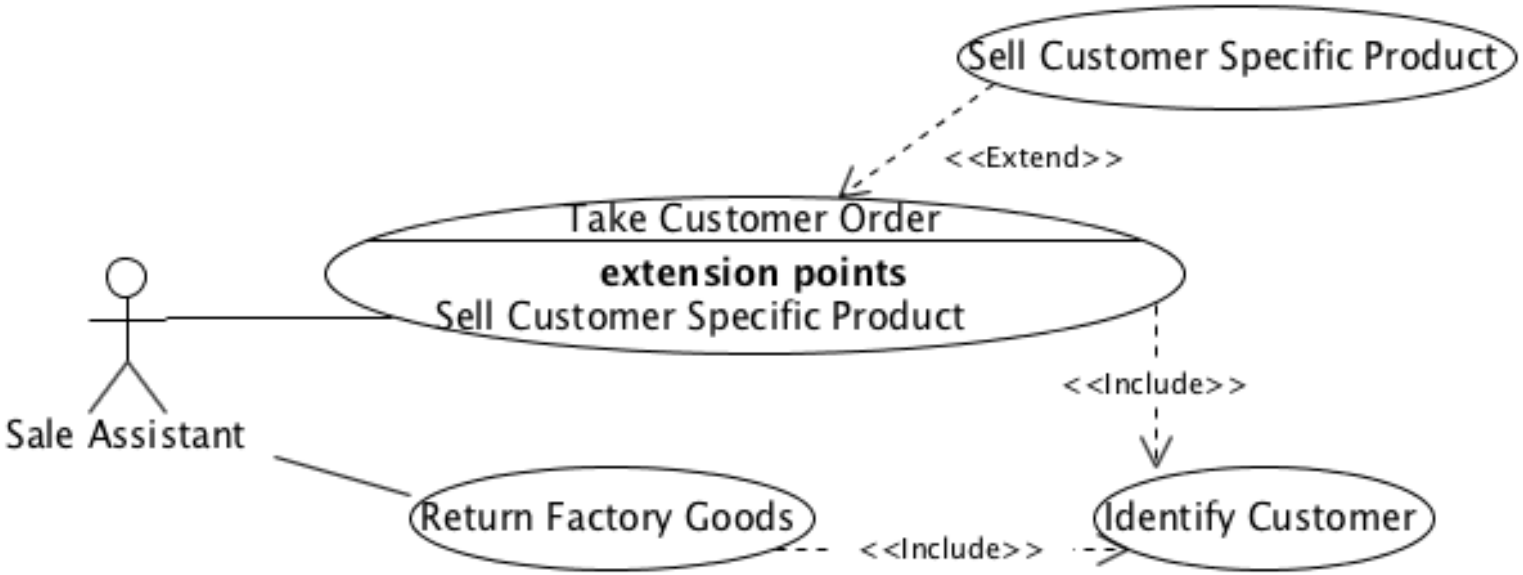
- ความสัมพันธ์ระหว่าง use case ที่แสดงถึง use case หนึ่งที่ใช้งาน use case หนึ่ง
- ยูสเคสส่วนขยาย (Extend Use Case) เป็นอิสระจากยูสเคสหลัก โดยที่ยูสเคสส่วนขยายนี้ใช้กำหนดถึงฟังก์ชันที่เป็นเสมือนส่วนเสริม (Optional) ที่อาจจะใช้งานหรือไม่ใช้ก็ได้
- โดยที่ Actor เรียกใช้งาน Use Case หลักแล้ว
- หัวลูกศรชี้ไปที่ use case ที่เป็น use case หลัก

ตัวอย่าง ความสัมพันธ์แบบ extend

- ยูสเคส “ขอความช่วยเหลือในการลงทะเบียน” เป็น extned use case



ตัวอย่าง แผนภาพยูสเคส



Use Case Diagrams:

Steps

- หาผู้มีส่วนได้ส่วนเสียหรือผู้ใช้งานทั้งหมดที่จะได้รับประโยชน์จากการมีแผนภาพ use case
- ทบทวน/ตรวจสอบ สิ่งที่ stakeholder หรือผู้ใช้งานแต่ละกลุ่มต้องการ
  - แต่ละระบบย่อย (subsystem) มีผู้ใช้งานแต่ละประเภท (type of user) มีความสัมพันธ์กับ use case ครบถ้วน สมบูรณ์แล้วหรือไม่
- เลือก use case และ actor ที่จำเป็น มาวาดแผนภาพ use case
- ตั้งชื่อ use case อย่างรอบคอบ (ให้เข้าใจได้ง่าย) เพื่อให้ stakeholder หรือคนที่เกี่ยวข้องเข้าใจได้ง่าย

Question