บทที่ 10
การออกแบบระบบเชิงวัตถุ

เกริ่นนำ

ในบทที่แล้วผู้อ่านได้เข้าใจถึงการวิเคราะห์ระบบเชิงวัตถุ(Object-Oriented Design) โดยใช้แผนภาพต่าง ๆ ของยูเอ็มแอล ซึ่งหลังจากที่ได้ทำการวิเคราะห์ความต้องการของผู้ใช้ด้วยแผนภาพยูสเคสจนได้ข้อกำหนดของความต้องการแล้วจากนั้นก็ทำการปรับปรุงแผนภาพ (Refinement Diagram) ต่าง ๆ จนครบถ้วนสมบูรณ์ ขั้นตอนต่อไปคือการออกแบบระบบเชิงวัตถุ (Object Oriented Design) โดยในบทนี้จะได้ทำความเข้าใจถึงหลักการออกแบบระบบต่าง ๆ โดยใช้ยูเอ็มแอลซึ่งมีความจำเป็นมากที่ต้องออกแบบระบบเพื่อให้มั่นใจได้ว่าระบบที่จะพัฒนานั้นถูกต้องและตรงกับความต้องการของผู้ใช้งาน โดยในขั้นตอนการออกแบบนั้นจะแบ่งออกเป็นการออกแบบแอปพลิเคชัน การออกแบบสถาปัตยกรรมของระบบ และการออกแบบระบบฐานข้อมูล ดังนั้นเมื่อมีการพัฒนาระบบจะมีความถูกต้องตรงกับที่ได้วิเคราะห์ความต้องการเอาไว้

การออกแบบระบบ (System Design) มีวัตถุประสงค์เพื่อออกแบบระบบให้เข้ากับความต้องการของระบบใหม่ตามที่ได้มีการวิเคราะห์ไว้โดยนักวิเคราะห์ระบบจะต้องออกแบบส่วนนำเข้าส่วนส่งออกฐานข้อมูล เครือข่ายระบบรักษาความปลอดภัย ส่วนต่อประสานกับผู้ใช้ (ศรีไพร ศักดิ์รุ่งพงศากุล และ เจษฎาพร ยุทธนวิบูลย์ชัย, 2549)

ความหมายการออกแบบระบบ

ความหมายของการออกแบบระบบ (System Design) หมายถึง การนำเอาความต้องการของระบบมาเป็นแบบแผนหรือเรียกว่าพิมพ์เขียวในการสร้างระบบเพื่อให้สามารถใช้งานได้จริง โดยการออกแบบระบบนั้นมีวัตถุประสงค์เพื่อออกแบบระบบให้เข้ากับความต้องการของระบบใหม่ตามที่ได้มีการวิเคราะห์ไว้ โดยนักวิเคราะห์ระบบจะต้องออกแบบส่วนนำเข้า ส่วนส่งออก ส่วนฐานข้อมูล ส่วนเครือข่ายและระบบรักษาความปลอดภัย ส่วนต่อประสานกับผู้ใช้ (ศรีไพร ศักดิ์รุ่งพงศากุล และ เจษฎาพร ยุทธนวิบูลย์ชัย, 2549)

ในขั้นตอนการออกแบบระบบเป็นการพิจารณาว่าระบบจะดำเนินการไปได้อย่างไร โดยเป็นการตัดสินใจว่าจะพัฒนาระบบใหม่ด้วยแนวทางใด เช่น พัฒนาระบบขึ้นเอง หรือจัดซื้อโปรแกรมสำเร็จรูป หรือว่าจ้างบุคคลหรือบริษัทภายนอกมาพัฒนาระบบใหม่ เป็นต้น ซึ่งเกี่ยวข้องกับการออกแบบสถาปัตยกรรมระบบโดยจะประกอบด้วยอุปกรณ์ฮาร์ดแวร์(Hardware) ซอฟต์แวร์(Software) เครือข่าย(Network) และฐานข้อมูลที่ใช้ในระบบ รวมถึงการเริ่มโปรแกรมและการออกแบบหน้าจอที่ใช้งาน ในขั้นตอนการออกแบบระบบนี้จะมุ่งเน้นถึงการดำเนินการแก้ปัญหาอย่างไร โดยการนำผลลัพธ์ของแบบจำลองเชิงตรรกะที่ได้จาก

กระบวนการวิเคราะห์ระบบ มาสร้างเป็นแบบจำลองเชิงกายภาพเพื่อนำไปใช้ในขั้นตอนการพัฒนาระบบเชิงวัตถุต่อไป (ไม่ปรากฏ, 2558)

ในระยะของการออกแบบจะให้ความสำคัญด้านองค์ประกอบอีกรูปแบบหนึ่ง คือ การออกแบบสถาปัตยกรรมระบบ (System Architecture Design) ซึ่งจะเป็นการอธิบายถึงสภาพแวดล้อมของระบบ หรือเทคนิคการทำงานของระบบ โดยผู้ทำการออกแบบจะต้องตัดสินใจเกี่ยวกับการประมวลผลว่าต้องการออกแบบในลักษณะใด โดยการออกแบบสถาปัตยกรรมระบบ เป็นการออกแบบในส่วนทั้งที่เป็นฮาร์ดแวร์และซอฟต์แวร์ โครงสร้างของระบบเครือข่าย และความเกี่ยวข้องกันภายในระบบที่พัฒนา ซึ่งขั้นตอนการออกแบบสถาปัตยกรรมระบบจัดได้ว่าเป็นขั้นตอนสำคัญในระยะการออกแบบ

ในบทที่ผ่านมาผู้อ่านได้เข้าใจถึงการวิเคราะห์ระบบเชิงวัตถุโดยใช้แผนภาพต่าง ๆ ของยูเอ็มแอล ซึ่งหลังจากที่ได้ทำการวิเคราะห์ความต้องการของผู้ใช้ด้วยยูสเคสแผนภาพจนได้ข้อกำหนดของความต้องการแล้ว จากนั้นก็ทำการปรับปรุงแผนภาพ (Refinement Diagram) ต่าง ๆ จนครบถ้วนสมบูรณ์ ขั้นตอนต่อไปคือการออกแบบระบบเชิงวัตถุ (Object Oriented Design) โดยในบทนี้จะได้ทำความเข้าใจถึงหลักการออกแบบระบบต่าง ๆ ด้วยยูเอ็มแอลซึ่งมีความจำเป็นมาก เพื่อให้มั่นใจได้ว่าระบบที่จะพัฒนานั้นถูกต้องและตรงกับความต้องการของผู้ใช้งาน โดยขั้นตอนการออกแบบนั้นจะแบ่งออกเป็นการออกแบบแอปพลิเคชัน การออกแบบสถาปัตยกรรมของระบบ และการออกแบบระบบฐานข้อมูล ดังนั้นเมื่อพัฒนาระบบจึงมั่นใจได้ว่าระบบที่พัฒนาจะมีความถูกต้องตรงกับที่ได้วิเคราะห์และออกแบบเอาไว้ สำหรับขั้นตอนการออกแบบระบบผู้พัฒนาระบบจะต้องทำการกำหนดรายละเอียดเชิงเทคนิคของระบบให้พร้อม เพื่อนำไปเขียนเป็นโปรแกรมเชิงวัตถุ (Implement) ในขั้นตอนถัดไป (กิตติพงษ์ กลมกล่อม, 2552)

การออกแบบระบบเชิงวัตถุ

เป้าหมายของการออกแบบเชิงวัตถุคือ การออกแบบคลาส โดยกำหนดคลาสและอ็อบเจกต์ เพิ่มเติมจากการวิเคราะห์ในส่วนที่จะสนับสนุนความต้องการ ตัวอย่าง เช่น ในระยะการออกแบบนั้นอาจจำเป็นต้องเพิ่มวัตถุที่เป็นส่วนติดต่อผู้ใช้ (User Interface) ในการพัฒนาระบบเชิงวัตถุนั้นมักมีการย้อนกลับไปทบทวนขั้นตอนก่อนหน้าหลายครั้งหรือพูดอีกด้านหนึ่งก็คือต้องทำสิ่งต่อไปนี้โดยเริ่มจากวิเคราะห์เชิงวัตถุ สร้างตัวแบบ ออกแบบเชิงวัตถุ และทำซ้ำในแต่ละขั้นตอนเพิ่มอีกกี่ครั้งก็ได้เท่าที่ต้องการ สิ่งที่ควรพิจารณาควบคู่ไปกับการออกแบบเชิงวัตถุ มีดังนี้

1. ออกแบบให้สามารถนำคลาสที่มีอยู่แล้วกลับมาใช้ได้มากที่สุด สร้างคลาสใหม่ให้น้อยที่สุด ต้องรู้ว่ามีคลาสอะไรบ้างที่กำหนดไว้ก่อนแล้ว

2. สร้างคลาสง่าย ๆ ไม่ซับซ้อนถึงจะมีจำนวนมากก็ตาม ซึ่งดีกว่าสร้างคลาสจำนวนน้อยแต่คลาสเหล่านั้น มีความซับซ้อนมาก

3. ออกแบบเมธอดสำหรับแต่ละคลาส

4. ในกรณีที่ยังรู้สึกว่าติดขัดในการนำเสนอสิ่งที่ได้ออกแบบไปแล้วให้ย้อนกลับไปทบทวนคลาสข้างต้นใหม่

ในการเปลี่ยนผ่านจากขั้นตอนวิเคราะห์ระบบเชิงวัตถุ (OOA) สู่ขั้นตอนการออกแบบระบบเชิงวัตถุ (OOD) นั้น การวิเคราะห์ระบบเป็นการมองหาว่าปัญหาที่ต้องการแก้ไขนั้นคืออะไร (What is the problem to be solved?) หากแต่ว่าเพียงแค่รู้ว่าปัญหาที่ต้องการแก้ไขคืออะไรนั้นคงไม่พอเพียง จึงจำเป็นที่จะต้องหาหนทางแก้ไขปัญหาหรือตอบคำถามให้ได้ว่า ปัญหานั้นจะถูกแก้ไขได้ด้วยวิธีใด ซึ่งในการวิเคราะห์และออกแบบระบบเชิงวัตถุ (Object Oriented Analysis and Design) เรียกขั้นตอนนี้ว่าขั้นตอนการออกแบบเชิงวัตถุ (OOD: Object-Oriented Design) เนื่องจากว่าปัญหาหรือระบบที่สนใจนั้นจะถูกแก้ไขในเครื่องคอมพิวเตอร์ ดังนั้นหนทางแก้ไขปัญหาก็ย่อมถูกทำขึ้นเพื่อให้สามารถนำไปใช้งานในคอมพิวเตอร์ได้จริง ๆ ซึ่ง ณ จุดนี้ทำให้เกิดข้อแตกต่างระหว่างขั้นตอนวิเคราะห์ระบบเชิงวัตถุกับขั้นตอนการออกแบบระบบเชิงวัตถุขึ้น เพราะในขั้นตอนขั้นตอนวิเคราะห์ระบบเชิงวัตถุนั้นไม่ได้คำนึงถึงเครื่องคอมพิวเตอร์เลย เพียงแต่วิเคราะห์หรือค้นหาปัญหาที่มีอยู่เท่านั้น แต่สิ่งที่ต้องทำสำหรับขั้นตอนการออกแบบระบบเชิงวัตถุคือการนำเอาผลลัพธ์ที่ได้จากวิเคราะห์ระบบเชิงวัตถุซึ่งหมายถึงแผนภาพต่าง ๆ นั้นมาทำการปรับปรุง (Refinement) เพื่อให้สามารถนำมาใช้ในเครื่องคอมพิวเตอร์ได้ง่ายขึ้น และเพิ่มเติมแผนภาพอื่น ๆ เพื่อจำลองและอธิบายภาพของแนวทางการแก้ปัญหาในคอมพิวเตอร์ต่อไป (กิตติ ภักดีวัฒนะกุล และพนิดา พานิชกุล, 2548)

หลักการออกแบบระบบเชิงวัตถุ

ในการออกแบบระบบเชิงวัตถุ (Object-Oriented Design) นั้นจะแบ่งการออกแบบออกเป็นส่วนต่าง ๆ 4 ส่วนด้วยกัน ดังนี้

1) การปรับปรุงแผนภาพต่าง ๆ ที่ได้จากขั้นตอนการวิเคราะห์ระบบเชิงวัตถุ (Object-Oriented Analysis) ให้ดีขึ้น มีประสิทธิภาพมากยิ่งขึ้น เรียกขั้นตอนนี้ว่าการทำรีไฟร์เม้นท์ (Refinement)
2) การออกแบบสถาปัตยกรรมของแอปพลิเคชัน (Application Architecture Design)
3) การออกแบบฐานข้อมูล (Persistent Data Design)
4) การออกแบบสถาปัตยกรรมของระบบ (System Architecture Design)

การปรับปรุงแผนภาพต่าง ๆ ที่ได้จากขั้นตอนการวิเคราะห์ระบบเชิงวัตถุ

โดยในขั้นตอนการออกแบบระบบเชิงวัตถุ ทีมนักพัฒนาระบบจะทำการกำหนดรายละเอียดเชิงเทคนิคของระบบให้พร้อม เพื่อนำไปเขียนเป็นโปรแกรมเชิงวัตถุ (Implement) จริงในเฟสถัดไป อาจกล่าวได้ว่าขั้นตอนนี้เป็นการค้นหาวิธีการแก้ปัญหาภายหลังจากการทำความเข้าใจปัญหา สิ่งที่ต้องเตรียมสำหรับใช้ในขั้นตอนนี้ คือ รูปแบบการวิเคราะห์จากขั้นตอนที่สอง ซึ่งอาจรวมถึงความต้องการแบบไม่เชิงฟังก์ชัน (Non-Functional) เช่น ระบบจะต้องใช้งานง่าย หมายถึงการออกแบบหน้าจอให้ดูสวยงานไม่ซับซ้อน เมนูไม่มากหรือน้อยเกิน เมนูหรือหน้าจอจะต้องเป็นธีมเดียวกันมีความสม่ำเสมอ ก็จะถูกนำมาพิจารณาในการออกแบบในขั้นตอนนี้ด้วยเช่นกัน กิจกรรมในการขั้นตอนการออกแบบระบบมีดังต่อไปนี้

1) การเพิ่มเติม คลาส หรือ แพ็กเกจ ลงไปภายในรูปแบบ

การวิเคราะห์ระบบ (ในขั้นตอนที่สอง) เช่น การเพิ่มแพ็กเกจที่เกี่ยวข้องกับฐานข้อมูล การติดต่อสื่อสารโดยแพ็กเกจ หรือ คลาสเหล่านี้จะทำงานร่วมกันกับแพ็กเกจเดิมที่มีอยู่ในรูปแบบการวิเคราะห์ระบบ นอกจากนี้ยังรวมถึงการแก้ไขปรับปรุงเพิ่มเติมแอตทริบิวต์ (Attribute) หรือฟังก์ชัน (Function) ของคลาสต่าง ๆ ในรูปแบบการวิเคราะห์ระบบด้วยเช่นกัน

2) จัดหาลำดับชั้นของคลาส (Class Hierarchy) หรือคอมโพเน้นท์ (Component) จากที่อื่นเพื่อนำมาใช้อีกครั้ง(Reusable) เพื่อช่วยลดเวลาในการพัฒนาระบบ

3) กำหนดรายละเอียดส่วนติดต่อกับผู้ใช้ของระบบ (User Interface Design)

4) หาวิธีจัดการกับข้อผิดพลาดที่อาจจะเกิดขึ้นในการใช้งานระบบ (Exception Handling)

5) ออกแบบสถาปัตยกรรมระบบ (System Architecture Design) เพื่อทำการพิจารณาที่อยู่ หรือตำแหน่งการติดตั้งของส่วนประกอบ (Component) หรือแพ็กเกจ (Package) ต่าง ๆ พิจารณาว่าคลาสใดควรอยู่ในส่วนประกอบ (Component) หรือไฟล์ใดควรติดตั้งไว้ที่ระบบคอมพิวเตอร์ส่วนใด กลุ่มของคลาสที่เกี่ยวข้องกับการคำนวณจะจัดให้อยู่ในกลุ่ม package เดียวกัน นั่นคือการสร้างแผนภาพคอมโพเน้นท์ (Component Diagram) และแผนภาพการดีพลอยด์ (Deployment Diagram) สำหรับรูปแบบของสถาปัตยกรรมที่ใช้กันโดยทั่วไป คือ สถาปัตยกรรมแบบกระจาย (Distributed System) ซึ่งได้แก่ 2-เทียร์ (2-Tier) 3-เทียร์ (3-Tier) และ มัลติเทียร์ (Multi-Tier System)

6) ออกแบบส่วนความต้องการที่ไม่เป็นฟังก์ชัน (Non-functional Requirement) เช่น กลุ่มของผู้ใช้งานมีความต้องการใช้งานระบบใหม่ร่วมกับฐานข้อมูลหรือระบบเดิมที่มีอยู่ (Legacy System Integration) เช่น ระบบจะต้องใช้งานง่าย หมายถึงการออกแบบหน้าจอให้ดูสวยงานไม่ซับซ้อน เมนูไม่มากหรือน้อยเกิน เมนูหรือหน้าจอจะต้องเป็นธีมเดียวกันมีความสม่ำเสมอ หรือการโหลดหน้าแรกของเว็บจะต้องไม่เกิน 3 วินาที

7) นอกจากนี้ การออกแบบยังรวมถึงการตัดสินใจเลือกใช้เทคโนโลยีต่าง ๆ ที่มีอยู่ให้เหมาะสมกับความต้องการและงบประมาณของผู้ใช้งานด้วยเช่นกัน เช่นการสร้างแอปพลิเคชันแบบ Cross Platform สร้างครั้งเดียวแต่ใช้ได้ทั้ง Andriod และ IoS เช่น ionic , google flutter

กล่าวโดยสรุปแผนภาพยูเอ็มแอล (UML Diagram) ที่ถูกสร้างในขั้นตอนของการออกแบบนี้ได้แก่ แผนภาพคอมโพเน้นท์ (Component Diagram) และแผนภาพดีพลอยด์ (Deployment Diagram) ในส่วนของแผนภาพคลาส (Class Diagram) และแผนภาพลำดับ (Sequence Diagram) จะถูกเพิ่มรายละเอียดเชิงเทคนิค โดยเรียกผลลัพธ์รวมของขั้นตอนนี้ว่าการออกแบบระบบเชิงวัตถุ (Object Oriented Design) ซึ่งประกอบไปด้วยแผนภาพต่าง ๆ ข้างต้น รวมถึงข้อกำหนดด้านอื่น ๆ ซึ่งระบุถึงรายละเอียดของความต้องการแบบไม่เชิงฟังก์ชัน (Non-functional) เทคนิควิธีการแก้ปัญหา และเอกสารการออกแบบ (UML Design Document) นี้จะถูกส่งต่อไปให้โปรแกรมเมอร์เพื่อนำไปพัฒนาในขั้นตอนถัดไป (กิติ ภักดีวัฒนะกุล และกิตติพงษ์ กลมกล่อม, 2544)

การออกแบบสถาปัตยกรรมของแอปพลิเคชัน

การออกแบบสถาปัตยกรรมแอปพลิเคชัน (Application Architecture Design) คือ กระบวนการที่ดึงเอาผลลัพธ์ต่าง ๆ ที่ได้ผ่านกระบวนการวิเคราะห์เชิงวัตถุมาแล้ว (Object Oriented Analysis) โดยจะได้แผนภาพต่าง ๆ มาเป็นวัตถุดิบในการออกแบบส่วนประกอบของแอปพลิเคชันที่จะมีในระบบ ซึ่งจะใช้แผนภาพคอมโพเน้นท์ (Component Diagram) เป็นเครื่องมือในการออกแบบสถาปัตยกรรมของแอปพลิเคชัน (Application Architecture) (นัฐพงศ์ ส่งเนียม, 2563) ซึ่งมีรายละเอียดดังต่อไปนี้

แผนภาพคอมโพเน้นท์

แผนภาพคอมโพเน้นท์ (Component Diagram) เป็นแผนภาพเชิงสถิตย์ที่ใช้จำลองลักษณะทางกายภาพของระบบเชิงวัตถุ (Object Oriented System) โดยจะแสดงให้เห็นถึงส่วนประกอบต่าง ๆ ของระบบรวมไปถึงความสัมพันธ์ระหว่างส่วนประกอบแต่ละตัวเพื่อให้สามารถแบ่งระบบงานที่มีขนาดใหญ่ (Integrate System) ออกเป็นระบบย่อย (Sub System) ซึ่งในแต่ละระบบย่อยจะมีส่วนโปรแกรมต่าง ๆ เป็นองค์ประกอบอยู่โดยการเขียนอธิบาย Component หรือ Relationship จะถูกเขียนอยู่ในเครื่องหมาย { } (นัฐพงศ์ ส่งเนียม, 2563)

สัญลักษณ์ต่าง ๆ ในแผนภาพคอมโพเน้นท์

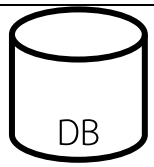
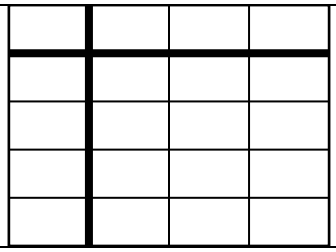
สัญลักษณ์ต่าง ๆ ในแผนภาพคอมโพเน้นท์จะมีลักษณะคล้ายคลึงกับแผนภาพคลาสและแผนภาพยูสเคสโดยแบ่งออกเป็น 2 ส่วนดังนี้

1) ส่วนของสัญลักษณ์แทนคอมโพเน้นท์

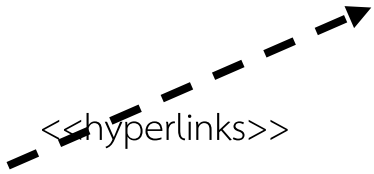
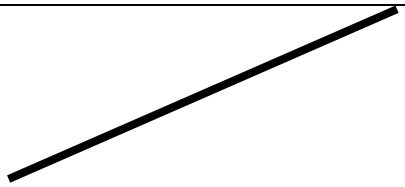
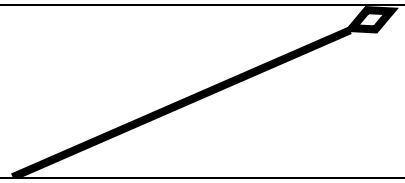
2)ส่วนของสัญลักษณ์แทนความสัมพันธ์

ตารางที่ 10.1 สัญลักษณ์แทนคอมโพเน้นท์

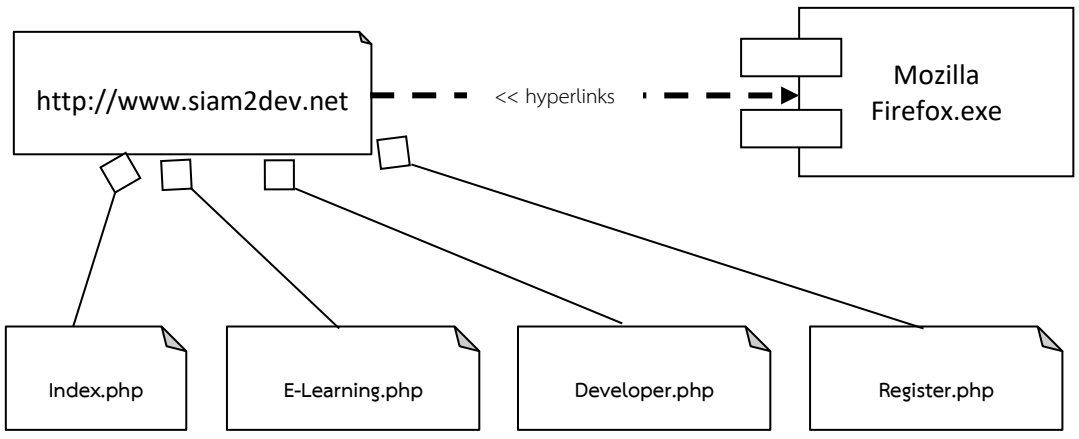
สัญลักษณ์	ชื่อ	ความหมาย
	Executable Program	โปรแกรมที่สามารถนำไปประมวลผลได้ (Executable Program) ไฟล .exe
	Page or File	ไฟล์ หรือรหัสต้นฉบับ(Source Code) เช่น MngEmployee.java ,AccountDB.java เป็นต้น

	**Database**	ฐานข้อมูล (Database)
	**Table**	ชุดข้อมูลในฐานข้อมูล (ตารางหรือรีเลชัน)

ตารางที่ 10.2 สัญลักษณ์แทนความสัมพันธ์

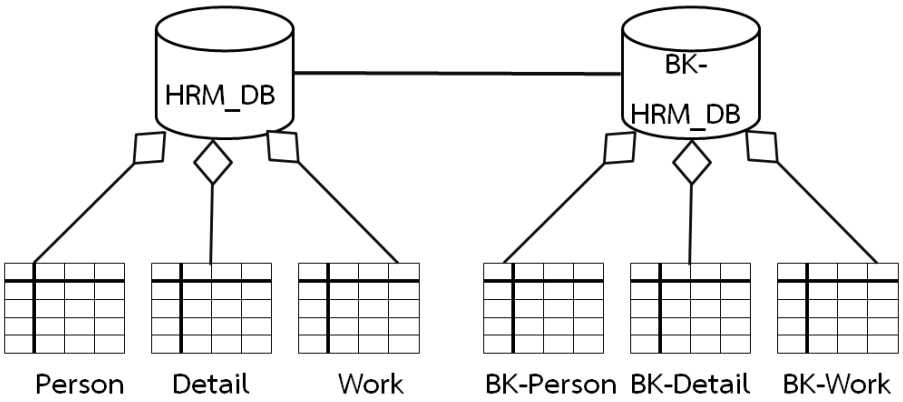
สัญลักษณ์	ชื่อ	ความหมาย
	**Dependency, Calls or Uses**	สัญลักษณ์การเรียกใช้งานหรือการขึ้นต่อกัน (Call, Uses, Dependency)
	**General Connection**	สัญลักษณ์แสดงการเชื่อมโยงหรือเชื่อมต่อ (General Connection)
	**Composition or Aggregation**	สัญลักษณ์แสดงการเป็นส่วนประกอบ (Aggregation)

ตัวอย่างที่ 10.1 ตัวอย่างแผนภาพคอมโพเน้นท์ของระบบ Web Browsing System



ภาพที่ 10.1 แสดงตัวอย่างแผนภาพคอมโพเน้นท์ของระบบ Web Browsing System

ตัวอย่างที่ 10.2 ตัวอย่างของแผนภาพคอมโพเน้นท์ของระบบฐานข้อมูลฝ่ายบุคคล

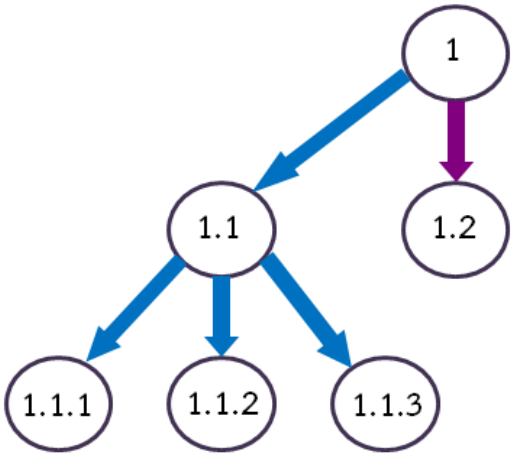


ภาพที่ 10.2 แสดงตัวอย่างของแผนภาพคอมโพเน้นท์ของระบบฐานข้อมูลฝ่ายบุคคล

จากภาพที่ 10.2 เป็นแผนภาพคอมโพเน้นท์ของ Database Subsystem ของระบบฐานข้อมูลของระบบงานบุคคลที่ประกอบด้วยฐานข้อมูลจำนวน 2 ชุด (ชุดใช้งานและชุดสำรอง) ซึ่งฐานข้อมูลทั้งสองมีลักษณะเหมือนกันเพราะต่างก็ประกอบด้วยตารางจำนวน 3 ตารางเหมือนกัน และฐานข้อมูลทั้ง 2 ชุดถูกเชื่อมโยงกันด้วยการเชื่อมต่อบางอย่าง ซึ่งอาจจะเป็นสายสื่อสารข้อมูลหรืออินเตอร์เน็ตก็ได้

การออกแบบระบบแบบแยกส่วนประกอบ

การออกแบบระบบซอฟต์แวร์ในปัจจุบันที่มีความซับซ้อนมากส่วนใหญ่มักจะใช้วิธีการทำที่เรียกว่าระบบแบบแยกส่วนประกอบ (System Decomposition) ซึ่งหมายถึงการแบ่งระบบใหญ่ออกเป็นส่วน ๆ ย่อย โดยเรียกระบบที่แยกออกมาว่า ระบบย่อย (Subsystems) และจะทำการแยกย่อยแต่ละส่วนจนกระทั่งละเอียดที่สุดซึ่งการแบ่งระบบใหญ่ออกเป็นระบบย่อย ๆ หรือที่เรียกว่าระบบแบบแยกส่วนประกอบนั้นมีด้วยกันหลายวิธี แต่ในที่นี้จะใช้การแบ่งแยกระบบแบบวิศวกรรมขุดเจาะลงไป (Drill Down Engineering) คือพยายามแบ่งระบบใหญ่ออกเป็นระบบย่อย โดยทำการขุดเจาะลงไปในระบบย่อยแต่ละส่วนไปจนกระทั่งถึงขั้นละเอียดที่สุด หรือละเอียดจนกว่าทีมงานออกแบบเห็นสมควรดังแสดงในภาพที่ 10.4



ภาพที่ 10.3 แสดงวิศวกรรมขุดเจาะลงไป (Drill Down Engineering)

โดยหลักการแบ่งระบบใหญ่ออกเป็นระบบย่อย ๆ ที่นิยมในปัจจุบันจะแบ่งระบบออกเป็น 3 ระบบย่อยด้วยกันตามหน้าที่ของแต่ละส่วนดังนี้

1) Presentation Logic Subsystem

2) Working Logic Subsystem / Business Logic Subsystem

3) Database Logic Subsystem

1. Presentation Logic Subsystem

Presentation Logic Subsystem เป็นระบบย่อย (Subsystem) ที่เกี่ยวข้องกับระบบย่อยที่ติดต่อกับผู้ใช้งานระบบโดยตรง (User Interface) เช่น การรับข้อมูลจากแป้นพิมพ์ และการแสดงผลออกมาในรูปแบบของผลลัพธ์ซึ่งจะประกอบไปด้วย

1) ส่วนที่ติดต่อกับผู้ใช้ (User Interface)

2) ส่วนของการแสดงผลลัพธ์ (Output)

3) ส่วนของการนำเข้า (Input)

2. Working Logic Subsystem หรือ Business Logic Subsystem

Working Logic Subsystem หรือ Business Logic Subsystem คือ ส่วนของระบบย่อยที่เกี่ยวข้องกับการทำงานที่เกิดขึ้นจริง ๆ ใน หน่วยประมวลผลกลาง (CPU) ของเครื่องคอมพิวเตอร์ เช่น ส่วนของการคำนวณ กลไกการดึงข้อมูลที่ได้รับมาเพื่อนำมาใช้งาน กลไกการบันทึกข้อมูล เป็นต้น

3. Database Logic Subsystem

Database Logic Subsystem เป็นระบบย่อยที่จำลองภาพของ Data Item ต่าง ๆ ที่ถูกจัดเก็บอยู่ในสื่อบันทึกข้อมูล ซึ่งสามารถอยู่ในรูปของฐานข้อมูลหรือแฟ้มข้อมูลก็ได้ซึ่งการออกแบบ Database Logic Sub

System นี้จะมีส่วนสัมพันธ์กับ Present Data Design ด้วยหลักการ Drill Down Engineering สามารถแบ่งระบบย่อยแต่ละตัวให้แยกย่อยลงไปได้อีกซึ่งระบบย่อยแต่ละตัวจะมีความละเอียดจนถึงระดับใดนั้น ขึ้นอยู่กับความเหมาะสมและความเห็นชอบของทีมงานที่พัฒนาระบบงาน หลังจากนั้นจะใช้แผนภาพคอมโพเน้นท์ (Component Diagram) เพื่อจำลองระบบย่อยแต่ละตัว

หลักการในการทำระบบแยกส่วนประกอบ (System Decoposite)

ในการทำระบบแยกส่วนประกอบ สามารถแบ่งออกเป็นขั้นตอนคร่าว ๆ ได้ดังนี้

1) การเขียนโดยทั่วไป

2) การเขียน Presentation Logic Subsystem

3) การเขียน Working Logic Subsystem

4) การเขียน Database Logic Subsystem

หลักการในการเขียนโดยทั่วไป

1) ทำจากระบบคร่าว ๆ ไปจนกระทั่งละเอียด

2) เมื่อได้ระบบย่อยแต่ละตัวแล้วให้พิจารณาคลาสที่มีอยู่และหาคลาสที่ตกหล่นหรือขาดหายไป

3) เมื่อมีการเพิ่มคลาสใหม่ให้ใส่ความสัมพันธ์ของคลาสนั้นกับคลาสอื่นที่มีอยู่แล้ว ลงในแผนภาพคลาสและแผนภาพลำดับด้วย

หลักการเขียน Presentation Logic Subsystem

ในการเขียน Presentation Logic Subsystem มีแนวทางดังนี้

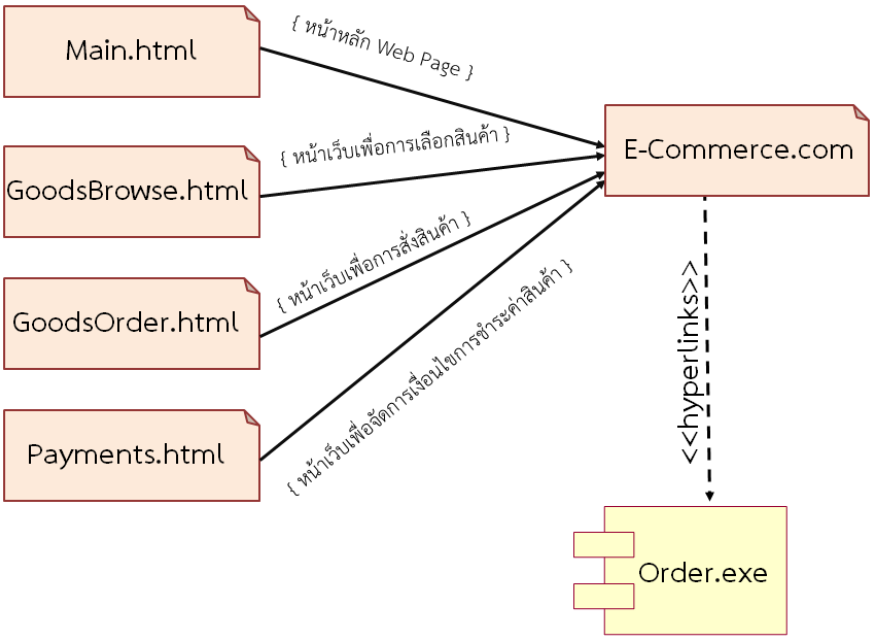
1) ให้ดึงทุก ๆ คลาสจากแผนภาพคลาสที่เป็นส่วนติดต่อผู้ใช้ (User Interface) มาใส่ลงใน Presentation Logic Subsystem

2) พยายามหา Generalized Class ของส่วนติดต่อผู้ใช้ทั้งหมด ให้ดูคลาสสืบทอดหรือรับทอดมรดก

3) ทำการเขียนแผนภาพคอมโพเน้นท์ของ Presentation Logic Subsystem

ตัวอย่างที่ 10.3 ตัวอย่างการเขียน Presentation Logic Subsystem ของระบบ E-commerce

การเขียน Presentation Logic Subsystem ของระบบ E-commerce คือ การสร้างแผนภาพคอมโพเน้นท์ของระบบย่อยต่าง ๆ ของระบบการขายสินค้าทางอินเตอร์เน็ต (E-Commerce) ซึ่งมีรายละเอียดดังภาพที่ 10.5 เน้นคลาสที่เกี่ยวข้องกับส่วนติดต่อผู้ใช้ (UI)



ภาพที่ 10.4 ตัวอย่างการเขียน Presentation Logic Subsystem ของระบบ E-commerce

จากภาพที่ 10.4 เป็นตัวอย่างของ Presentation Logic Subsystem ที่แสดงส่วนของการรับข้อมูลส่วนติดต่อผู้ใช้ และส่วนแสดงผลข้อมูล ของระบบการทำธุรกรรมบนอินเตอร์เน็ต ซึ่งประกอบไปด้วยไฟล์ Main.html จะแสดงหน้าหลักของระบบ ส่วนไฟล์ GoodBrowse.html จะแสดงรายการสินค้าตามหมวดหมู่หรือแยกตามประเภทของสินค้า ในขณะที่ไฟล์ GoodOrder.html จะแสดงรายการสั่งซื้อสินค้า และไฟล์ Payment.html แสดงรายการที่ต้องชำระเงิน เป็นต้น

หลักการเขียน Working Logic Subsystem (ระบบย่อย)

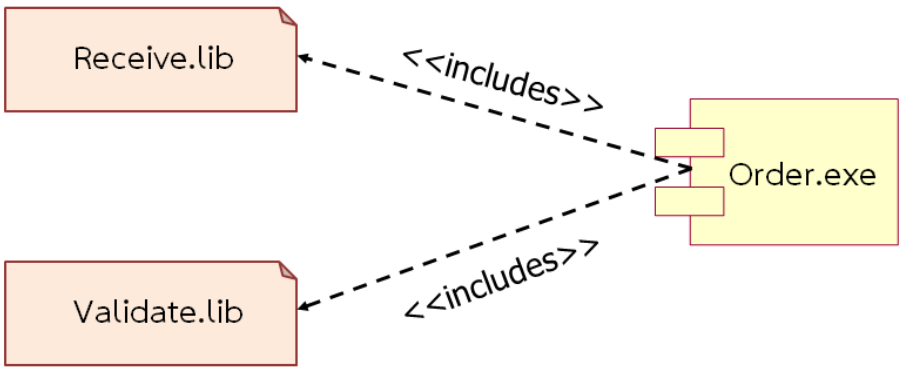
ในการเขียน Working Logic Subsystem มีแนวทางดังนี้

1) ให้ดึงทุกคลาสและความสัมพันธ์ที่มีทั้งหมดที่ไม่ใช่ส่วนติดต่อผู้ใช้ (User Interface) จากแผนภาพคลาสมาใส่ใน Working Logic Subsystem

2) ใช้หลักการวิศวกรรมขุดเจาะลงไปเพื่อแบ่งแยก Working Logic Subsystem โดยยึดหลักการที่ว่าคลาสที่มีความสัมพันธ์ต่อกันมักจะทำงานร่วมกันเสมอ

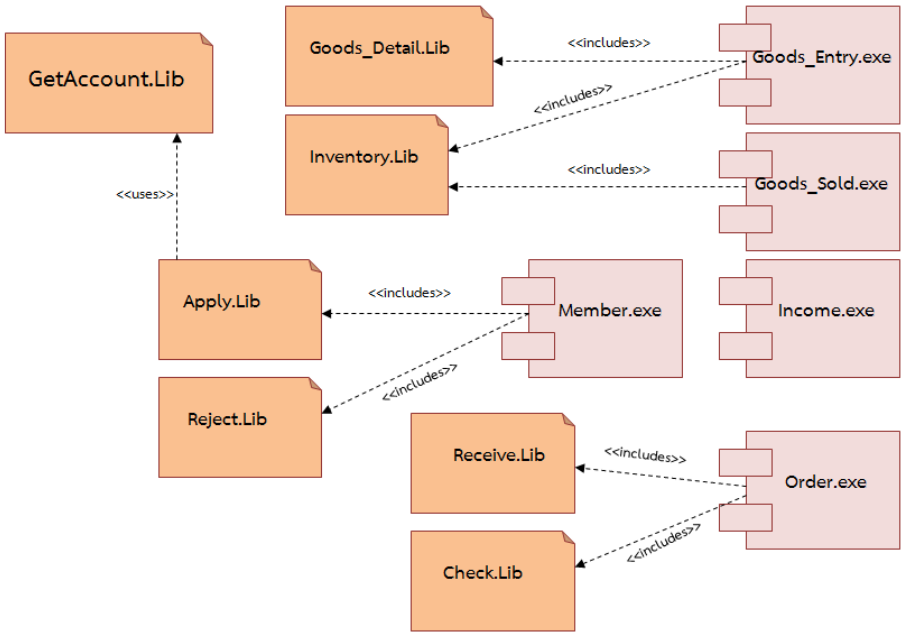
3) เขียนแผนภาพคอมโพเน้นท์ของ Working Logic Subsystem

ตัวอย่างที่ 10.4 ตัวอย่างการเขียน Working Logic Subsystem ของระบบการสั่งซื้อสินค้า



ภาพที่ 10.5 ตัวอย่างการเขียน Working Logic Subsystem ของระบบการสั่งซื้อสินค้า

จากภาพที่ 10.5 เป็นตัวอย่างการเขียน “Working Logic subsystem ซึ่งประกอบไปด้วย 2 ไฟล์ ได้แก่ Receive.lib ซึ่งเป็นไลบรารี สำหรับจัดการใบเสร็จของลูกค้า และไฟล์ Validate .lib เป็นไฟล์ไลบรารี



ภาพที่ 10.6 ตัวอย่างการเขียน Working Logic Subsystem ของระบบจัดการสินค้า

จากภาพที่ 10.6 สามารถอธิบายได้ว่า Goods_Entry.exe และ Goods_Sold.exe เป็นแอปพลิเคชันที่จัดการการสั่งสินค้าเข้าและขายสินค้าออก ซึ่งไลบรารี (Library) ที่ต้องใช้สำหรับ 2 แอปพลิเคชันนี้คือ Goods_Detail.Lib และ Inventory.Lib ที่ใช้เพื่อการจัดการรายละเอียดของสินค้าและการดูแลสินค้าคงคลังตามลำดับ ส่วน Member.exe เป็นแอปพลิเคชันที่ใช้จัดการเกี่ยวกับสมาชิกของระบบการขายสินค้า ซึ่งไลบรารีที่แอปพลิเคชันนี้ใช้งานคือ Apply.Lib และ Reject.Lib ใช้ในกรณีการสมัครสมาชิกใหม่และสมาชิกลาออกหรือออกด้วยเหตุผล Apply.Lib มีไลบรารีย่อย ชื่อ GetAccount.Lib ซึ่งใช้สำหรับการ จัดการบัญชีเงินฝากของสมาชิกที่ใช้เพื่อการจ่ายค่าสินค้า

- Income.exe เป็นแอปพลิเคชันที่ใช้สำหรับการจัดการรายได้ต่าง ๆ ขององค์กร

- Order.exe เป็นแอปพลิเคชันที่ใช้สำหรับการจัดการสั่งสินค้าจากลูกค้า โดยแอปพลิเคชันนี้ต้องใช้ไลบรารีชื่อ Receive.Lib เพื่อการรับการสั่งสินค้าจากลูกค้า และใช้ไลบรารีชื่อ Check.Lib เพื่อการตรวจสอบสินค้าที่มีอยู่ว่าพอจะให้ลูกค้าสั่งได้หรือไม่

หลักการเขียน Database Logic Subsystem

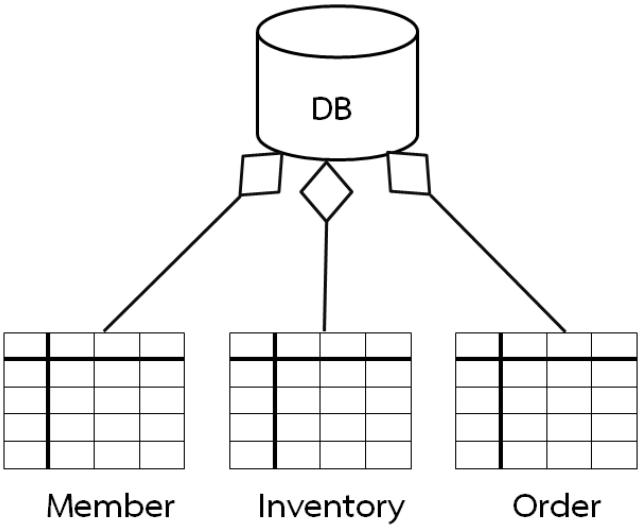
ในการเขียน Database Logic Subsystem มีหลักการหรือแนวทางดังนี้

1) คลาสที่อยู่ในระบบย่อยนี้ไม่จำเป็นต้องมีอยู่ในแผนภาพคลาสก็ได้ (หมายถึงอาจเป็นตารางที่ทางผู้พัฒนาเพิ่มเติมขึ้นมาเพื่อใช้งานทางเทคนิคบางอย่าง เช่น ตารางข้อมูล logfile ในการเก็บข้อมูลการเข้าใช้งานระบบ)

2) ตัวอย่างของ Database Component คือ ตารางต่าง ๆ ในฐานข้อมูลเชิงสัมพันธ์

3) ทุก ๆ Database Interface เป็นคลาสที่มีคุณสมบัติเหมือนกันบางประการ เช่น จะต้องมี Function Connect และ Disconnect เพื่อเข้าและออกจากระบบฐานข้อมูล

ตัวอย่างที่ 10.5 ตัวอย่างการเขียน Database Logic Subsystem ของระบบ e-Commerce



ภาพที่ 10.7 ตัวอย่างการเขียน Database Logic Subsystem ของระบบ e-Commerce

จากภาพที่ 10.7 สามารถอธิบายได้ดังนี้ คือใน Database Logic Subsystem ของระบบธุรกรรมอิเล็กทรอนิกส์ (e-Commerce) จะประกอบไปด้วย 3 ตาราง ดังนี้คือ 1. ตารางสมาชิก (Member) เป็นตารางที่เก็บข้อมูลเกี่ยวกับสมาชิกของระบบ 2. ตารางสินค้า (Inventory) เป็นตารางที่เก็บข้อมูลเกี่ยวกับสินค้าที่มีอยู่ในระบบและ 3. ตารางการสั่งซื้อ (Order) เป็นตารางที่เก็บข้อมูลเกี่ยวกับการสั่งสินค้าจากลูกค้า

การออกแบบระบบฐานข้อมูลเชิงวัตถุ

ในการพัฒนาระบบงานใด ๆ ก็ตามส่วนใหญ่แล้วมักจะมีการทำงานกับข้อมูลซึ่งเป็นสิ่งที่สำคัญ เนื่องจากแต่ละองค์กรหรือหน่วยงานจะมีข้อมูลมากมาย และสำคัญมากสำหรับแต่ละองค์กร ดังนั้นการจัดเก็บข้อมูลที่ดีจึงมีความสำคัญต่อองค์กร โดยทั่วไปแล้วข้อมูลมักจะถูกจัดเก็บในรูปแบบของไฟล์ข้อมูล หรือระบบฐานข้อมูล โดยการออกแบบฐานข้อมูล (Persistent Data Design) สำหรับระบบเชิงวัตถุมีรายละเอียดดังนี้

1) ข้อมูลแบบถาวร (Persistent Data)
2) การออกแบบฐานข้อมูล (Persistent Data Design)
3) ระบบฐานข้อมูลเชิงสัมพันธ์ (RDBMS)
4) การแปลงจากคลาสแผนภาพไปเป็นฐานข้อมูลเชิงสัมพันธ์

ความหมายของข้อมูลแบบถาวร

ข้อมูลแบบถาวร (Persistent Data) คือข้อมูลที่ยังคงอยู่แม้ว่าโปรแกรมที่สร้างข้อมูลนั้นขึ้นมาได้ถูกปิดไปแล้วก็ตาม โดยสามารถเก็บข้อมูลไว้ได้หลายรูปแบบ ได้แก่ ไฟล์ (File) ฐานข้อมูล (Database) ฐานข้อมูลเครือข่าย (Network Database) ฐานข้อมูลลำดับชั้น (Hierarchical Database) เป็นต้น

การออกแบบข้อมูลแบบถาวร (Persistent Data Design) หรือ Conceptual Database Design คือหลักการในการออกแบบฐานข้อมูลเพื่อใช้เก็บข้อมูลต่าง ๆ สามารถเก็บข้อมูลไว้ได้หลายรูปแบบ โดยในหลักการออกแบบเชิงวัตถุ (Object Oriented Design) จะไม่มีฐานข้อมูลสำหรับเก็บข้อมูลแบบถาวรโดยเฉพาะ แต่จะสามารถใช้เครื่องมือที่มีประสิทธิภาพที่ใช้กันอยู่ทั่ว ๆ มาทดแทนได้ เช่น ฐานข้อมูลเชิงสัมพันธ์ (Relational Database) ดังนั้น การใช้ฐานข้อมูลประเภทนี้ในการจัดเก็บข้อมูลของระบบ จึงจำเป็นต้องแปลงแผนภาพคลาสเป็นฐานข้อมูลเชิงสัมพันธ์โดยมีหลักการดังต่อไปนี้

ประเภทของฐานข้อมูลเชิงวัตถุ (Persistent Data Design)

สำหรับแนวคิดตามหลักการของวัตถุที่นำมาใช้กับฐานข้อมูลมี 2 แบบ คือ

1) ถ้ามีการจัดการแบบเชิงวัตถุและโปรแกรมภาษาอยู่แล้วแต่เพิ่มความสามารถด้านการจัดการฐานข้อมูลลงไปคือ OODBMS (Object Oriented Database Management System)

2) ฐานข้อมูลเชิงสัมพันธ์รองรับงานแบบออบเจ็กต์ คือ ORDBMS (Object Relational Database Management System)

ระบบฐานข้อมูลเชิงสัมพันธ์

ฐานข้อมูลเชิงสัมพันธ์ (Relational Database) คือ การเก็บข้อมูลในรูปของตารางหรือรีเลชัน (Table หรือ Relation) หลาย ๆ ตารางที่มีความสัมพันธ์กัน ในแต่ละตารางแบ่งออกเป็นแถว (Row) และในแต่ละแถว

จะแบ่งเป็นคอลัมน์ (Column) โดยโมเดลฐานข้อมูลเชิงสัมพันธ์ผู้ริเริ่มคือ Dr. Edgar F. Codd โดยกำหนดส่วนประกอบของโมเดลเชิงสัมพันธ์แบ่งเป็น 3 ส่วนได้แก่

1. ส่วนที่เกี่ยวข้องกับโครงสร้างของข้อมูล (DDL)
2. ส่วนที่เกี่ยวกับการควบคุมความถูกต้องให้กับข้อมูล (DCL)
3. ส่วนในการจัดการกับข้อมูล (DML)

เนื่องจากการออกแบบเชิงวัตถุไม่มีฐานข้อมูลสำหรับข้อมูลเชิงวัตถุโดยเฉพาะ ดังนั้นหากต้องการจัดเก็บเป็นข้อมูลโดยใช้ฐานข้อมูลเชิงสัมพันธ์ จึงต้องทำการแปลงข้อมูลจากแผนภาพคลาสไปเป็นข้อมูลเชิงสมัพันธ์ก่อน โดยมีหลักกการดังนี้

การปรับปรุง Persistent Class Diagram

หลักจากที่ได้ Persistent Class Diagram แล้ว สิ่งที่ต้องดำเนินการต่อไปคือการปรับความสัมพันธ์ระหว่าง Class ใหม่ โดยการปรับความสัมพันธ์ระหว่าง Class ที่ไม่สามารถอธิบายด้วย Abstraction (ได้แก่ Binding และ Derived) ให้อยู่ในรูปแบบที่สามารถอธิบายด้วย Abstraction แบบใดแบบหนึ่งเสมอ หากไม่มั่นใจว่า Class ทั้งสองข้าง Binding เป็นแบบ 1 : 1 หรือไม่ Binding ควรจะถูกแทนที่ด้วย One-to-Many Association โดยให้ Class ที่ได้รับการ Bind มี Multiplicity มากกว่า 1 ในขณะที่ Minimum Multiplicity ของ Class ทั้งสองด้านของ Association เป็น 0 Derive ควรถูกแทนที่ด้วย Many-to-Many Association โดย Minimum Multiplicity ของ Class ทั้งสองด้านของ Association เป็น 0 การปรับเปลี่ยนให้ Class ที่มี Attributes เป็น Class ให้กลายเป็น Class 2 Class ที่มีความสัมพันธ์กัน หรือเปลี่ยนให้เป็น Attributes ปกติที่ไม่ใช่ Class

กฎการอ้างอิง

กฎการอ้างอิง (Referential Rules) หมายถึง กฎที่กล่าวถึงการระบุความโดดเด่นของข้อมูล และการอ้างอิงความสัมพันธ์รหว่างข้อมูล ซึ่งในเบื้องต้น Referential ได้อ้างถึงคำสองคำ คือ

1) คีย์หลัก(Primary Key) คือ Column หรือกลุ่มของ Column ใน Table ซึ่งค่าของมันสามารถระบุความโดดเด่นหรือความแตกต่างของข้อมูล Row หนึ่งๆ จากข้อมูล Row อื่นๆได้
2) คีย์นอก(Foreign Key) คือ Column หรือกลุ่มของ Column ใน Table ซึ่งค่าของมันสามารถแสดงความสัมพันธ์กับ Table อื่นโดยอ้างอิงไปยัง Primary Key ของ Table นั้นได้

โดยคลาสใดคลาสหนึ่งในแผนภาพคลาส (Class Diagram) จะต้องมีความสัมพันธ์รูปแบบใดรูปแบบหนึ่งกับ คลาสอื่นอย่างน้อยหนึ่งคลาสเสมอ โดยสิ่งที่ใช้เพื่อเชื่อมความสัมพันธ์ระหว่างตาราง คือคีย์นอก (Foreign Key)

หลักการแปลงจากแผนภาพคลาสไปเป็นฐานข้อมูลเชิงสัมพันธ์

การแปลงจากคลาสแผนภาพไปเป็นฐานข้อมูลเชิงสัมพันธ์ เนื่องจากคลาส (Class) และตาราง (Table) เป็นความสัมพันธ์ในแง่ของความหมาย (Semantic Mapping) ระหว่างคลาสและตารางในเบื้องต้น หมายถึงการที่สามารถใช้ตารางหนึ่งตารางเพื่อเก็บข้อมูลของวัตถุต่าง ๆ ของหนึ่งคลาสได้โดยตรงจากคลาส เพื่อทำหน้าที่สร้างวัตถุและแถว (Object & Row) ซึ่งเป็นรายละเอียดของรายการหนึ่ง ๆ ในตาราง ดังนั้นจึงสามารถใช้แถวหรือเรคคอร์ด (Record) เพื่อเก็บรายละเอียดของวัตถุของคลาสได้ โดยในแต่ละคลาสจะประกอบไปด้วยคุณลักษณะอย่างน้อยหนึ่งตัว ในขณะที่แถวหนึ่ง ๆ จะประกอบไปด้วยคอลัมน์ (Column) อย่างน้อยหนึ่งตัวเช่นกัน ดังนั้นจึงสามารถใช้คอลัมน์เพื่อเก็บรายละเอียดของคุณลักษณะของวัตถุได้ (Booch, G. and author, 2005)

ความสัมพันธ์ระหว่างองค์ประกอบต่าง ๆ ของคลาสและตาราง

ความสัมพันธ์ระหว่างองค์ประกอบทั้งสามของคลาสและตาราง สามารถอธิบายได้ด้วยสัญลักษณ์ดังต่อไปนี้

Class → Table
Object → Row
Attribute → Column

จะเห็นได้ว่าไม่ได้กล่าวถึงฟังก์ชันหรือเมธอด (Method) แต่อย่างใด เนื่องจากฟังก์ชันเป็นส่วนหนึ่งของการกระทำหรือหน้าที่ของคลาสไม่ได้เป็นข้อมูลแต่อย่างใด ในการทำการออกแบบฐานข้อมูล (Persistent Data Design) จะมุ่งเน้นเฉพาะส่วนที่เป็นข้อมูลของคลาสหรือวัตถุซึ่งหมายถึงคุณลักษณะของวัตถุเท่านั้น

2.1 การแปลงคลาสให้เป็นตาราง

ในการแปลงคลาสเป็นตารางมีแนวทางดังนี้

1) ในการแปลงจะมุ่งเน้นพิจารณาเฉพาะแอตทริบิวต์เท่านั้น โดยไม่ต้องสนใจในส่วนของฟังก์ชัน (Function / Method)
2) กำหนดให้แอตทริบิวต์ตัวใดตัวหนึ่ง หรือกลุ่มของแอตทริบิวต์เป็นคีย์หลัก
3) สร้างตารางจากทุก ๆ แอตทริบิวต์ รวมทั้งคีย์หลัก
4) สำหรับแอตทริบิวต์ที่เป็นคีย์หลักให้กำหนดเป็น Not Null เสมอ (PK)
5) ให้พิจารณาแอตทริบิวต์ที่เหลือ ว่าเป็น Not Null หรือไม่ตามความเหมาะสม

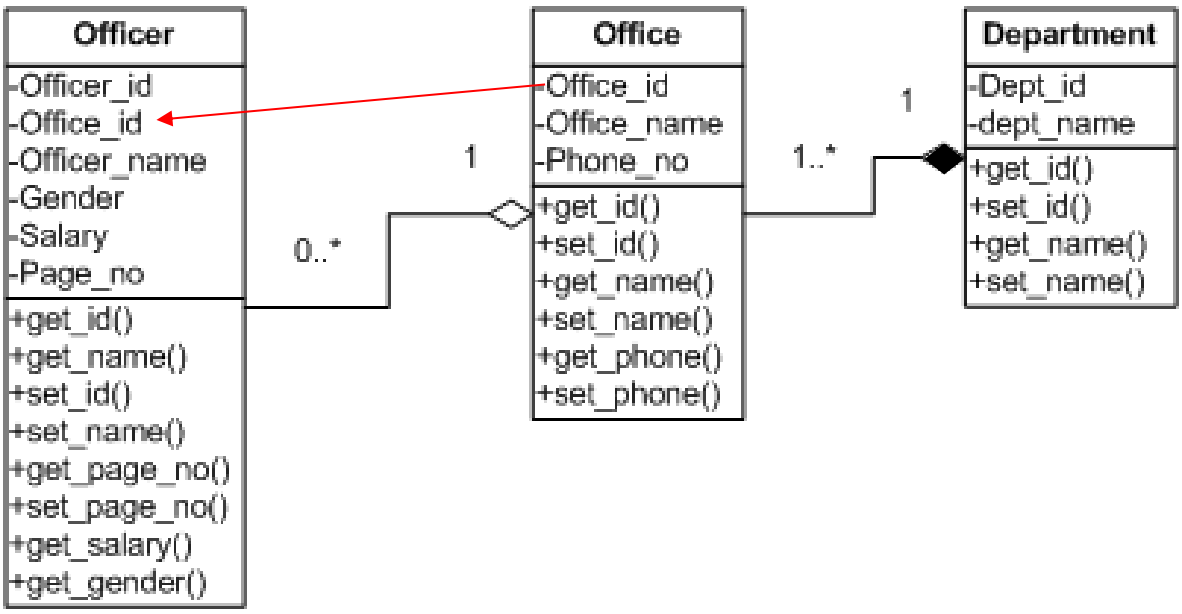
การเริ่มต้นทำข้อมูลแบบถาวร (Persistent Data Design) ต้องทำการวิเคราะห์ก่อนว่าระบบจำเป็นต้องเก็บคุณลักษณะของคลาสใดไว้บ้าง ให้คัดเลือกคุณลักษณะ (Attribute) ของคลาส (Class) ที่

จำเป็นต้องเก็บ โดยไม่จำเป็นต้องเก็บทุก ๆ คลาส (Class) และทุก ๆ แอตทริบิวต์ (Attribute) ที่ปรากฏอยู่ในแผนภาพคลาส (Class Diagram) เนื่องจากหากจะต้องเก็บข้อมูลที่ไม่ได้ใช้งานไว้ในฐานข้อมูล ซึ่งการมีข้อมูลจำนวนมากอยู่ในฐานข้อมูล ทำให้ประสิทธิภาพในการดึงข้อมูลมาใช้งาน หรือการจัดการข้อมูล ลดต่ำลงกว่าที่ควรจะเป็นและเป็นการสิ้นเปลืองเนื้อที่ในฐานข้อมูลอย่างมาก

2.2 หลักการแปลง Class ที่มีความสัมพันธ์แบบ Aggregation และ Composition เป็น Table

2.2.1 Aggregation

1) แปลง Class ทั้งสองให้เป็น Table ตามหลักการการแปลง Class ให้เป็น Table พร้อมทั้งกำหนด Primary Key ของทั้งสองให้เหมาะสม
2) วาดภาพความสัมพันธ์แบบ Association ระหว่าง Table ทั้งสอง พร้อมทั้งระบุ Minimum และ Maximum Multiplicity ให้ครบถ้วน(Table ที่มาจาก Whole Class จะมี Minimum และ Maximum Multiplicity เป็น 0 และ 1 ตามลำดับ ส่วน Part Class จะยังมี Minimum และ Maximum Multiplicity เท่าเดิม)
3) ให้นำ Primary Key ของ Table ที่มาจาก Whole Class ไปเป็น Foreign Key ใน Table ที่มาจาก Part Class โดยให้ Foreign Key นั้นมีคุณสมบัติเป็น NULL



คลาส officer เป็น Part Class (คลาสย่อย)

คลาส office จะเป็น Whole Class (คลาสหลัก)

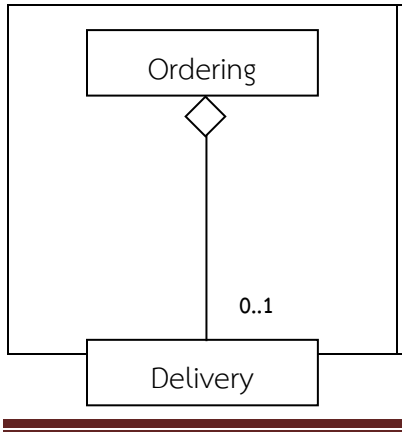
```
CREATE TABLE OFFICER
(OFFICER_ID     CHAR(10) NOT NULL,
 OFFICE_ID      CHAR(5),
```

```
  OFFICE_NAME CHAR(50) NOT NULL,
  GENDER      CHAR(1) NOT NULL,
  SALARY      INTEGER,
  PAGE_NO     CHAR(7),
  PRIMARY KEY(OFFICER_ID),
  FOREIGN KEY(OFFICE_ID) REFERENCES OFFICE
);
```

2.2.2 Composition

การแปลง Composition จะคล้ายกับการแปลง Aggregation โดยจะมีความต่างตรงที่ Composition นั้นคลาสย่อย (Part Class) จะคงอยู่ไม่ได้เลยถ้าคลาสหลัก (Whole Class) ไม่มีตัวตนอยู่ ด้วยเหตุนี้จึงต้องใช้กลไกบ้างอย่างในการจัดการกับข้อกำหนดนี้ โดยที่กลไกนั้นคือ การเพิ่มคุณสมบัติพิเศษของคีย์หลักและคีย์นอกของคลาสย่อย ดังขั้นตอนต่อไปนี้

1) แปลงคลาสทั้งสองให้เป็นตารางตามหลักการการแปลงคลาสให้เป็นตารางพร้อมทั้งกำหนดคีย์หลัก (Primary Key) ของทั้งสองให้เหมาะสม
2) วาดภาพความสัมพันธ์แบบแอสโซซิเอชัน (Association) ระหว่างตารางทั้งสอง พร้อมทั้งระบุ Minimum และ Maximum Multiplicity ให้ครบถ้วน(ตารางที่มาจากคลาสหลักจะมี Minimum และ Maximum Multiplicity เป็น 0 และ 1 ตามลำดับ ส่วนคลาสย่อยจะยังมี Minimum และ Maximum Multiplicity เท่าเดิม)
3) ให้นำคีย์หลักของตารางที่มาจากคลาสหลัก ไปเป็นคีย์นอกในตารางที่มาจากคลาสย่อย โดยให้คีย์นอกมีคุณสมบัติเป็น NOT NULL พร้อมทั้งเป็นคีย์หลักของตารางที่มาจากคลาสย่อยนั้นด้วย (ด้วยกลไกนี้จะเป็นการบังคับว่าคลาสย่อยจะมีอยู่ไม่ได้หากไม่มีคลาสหลัก ทั้งนี้เพราะหากไม่มีคลาสหลัก ก็จะไม่สามารถใส่ค่าใด ๆ ลงไปเป็นค่าของส่วนหนึ่งของคีย์หลักของคลาสย่อยได้ และเมื่อคีย์หลักของคลาสย่อยไม่ครบถ้วน ก็จะไม่สามารถมีตัวตนได้อีกต่อไป)




```
CREATE TABLE ORDERING
(
ORDER_CODE   CHAR(10)   NOT NULL,
ORDER_ADDRESS_ID CHAR(10)  NOT NULL, ORDER_DATE_TIME DATE
NOT NULL, ORDER_STAUTS   INTEGER   NOT NULL,
PRIMARY KEY(ORDER_CODE),
FOREIGN KEY(ORDER_ADDRESS_ID) REFERENCES ADDRESS
```

	); CREATE TABLE DELIVERY (DELIVERY_ID CHAR(10) NOT NULL, ORDERING_CODE CHAR(10) NOT NULL, DELIVERY_DATE_TIME DATE NOT NULL, FINISH_DATE_TIME DATE NULL, PRIMARY KEY(DELIVERY_ID, ORDER_CODE), FOREIGN KEY(ORDERING_CODE) REFERENCES ORDERING);

2.3 การแปลงแผนภาพคลาสที่เจอเนอรัลไลเซชัน

1) สร้างตารางพร้อม ๆ กับกำหนดคีย์หลักของตารางของ Super Class

2) สร้าง Table ของ Sub Class ทุกตัวที่มี โดยไม่ต้องมีคีย์หลัก

3) เปลี่ยนเส้นความสัมพันธ์แบบ Generalization ให้กลายเป็นความสัมพันธ์แบบ One-to-One โดยให้ Minimum และ Maximum Multiplicity ของ Super Class มีค่าเป็น 1 ทั้งคู่ และสำหรับ Sub Class ทุกตัวให้กำหนด Minimum Multiplicity

3) เป็น 0 และ Maximum Multiplicity เป็น 1

4) ให้สร้างคีย์หลักของ Sub Class ทุกตัวให้เหมือนกับคีย์หลักของ Super Class ทุกประการ

5) ให้ใช้คีย์หลักของ Sub Class ที่ได้เป็นคีย์หลักที่ใช้อ้างอิงไปยัง Super Class ด้วย

ตัวอย่างที่ 10.6 การแปลงคลาสนักเรียนเป็นฐานข้อมูล

- นักเรียนทุก ๆ คนจะต้องเข้าอยู่ชมรมฟุตบอลหรือเทนนิสหรืออาจจะอยู่ทั้ง 2 ชมรม
- นักเรียนทุกคนจะต้องมีคุณสมบัติเป็นนักฟุตบอลหรือนักเทนนิส อย่างใดอย่างหนึ่งหรือทั้งสองเสมอ เรียกว่า Total-Overlapping

```
CREATE TABLE STUDENT
( SID  CHAR(10)  NOT NULL,  NAME  CHAR(50)  NOT NULL,
  PRIMARY KEY(SID));
CREATE TABLE TENNIS
( SID CHAR(10)  NOT NULL,  PRIMARY KEY(SID),
  FOREIGN KEY(SID) REFERENCES STUDENT);
CREATE TABLE SOCCER
(SID   CHAR(10)   NOT NULL,  PRIMARY KEY(SID),
 FOREIGN KEY(SID) REFERENCES STUDENT);
```

- ในภาควิชาภาษาต่างประเทศ นักศึกษาทุกคนจะต้องเลือกเรียน ภาษาอังกฤษ หรือภาษาเยอรมัน เท่านั้น จะเลือกเรียนทั้งสองไม่ได้
- เรียกว่า Total-Exclusive

```
CREATE TABLE LANGUAGE
( LID  CHAR(10)   NOT NULL,  PRIMARY KEY(LID));
CREATE TABLE STUDENT
( ID    CHAR(10)  NOT NULL,  NAME  CHAR(50)  NOT NULL,  PRIMARY KEY(ID),
  FOREIGN KEY(ID) REFERENCES LANGUAGE);
CREATE TABLE ENGLISH
(LID   CHAR(10)  NOT NULL, PRIMARY KEY(LID),
 FOREIGN KEY(LID) REFERENCES LANGUAGE);
CREATE TABLE GERMAN
(LID   CHAR(10)  NOT NULL, PRIMARY KEY(LID),
 FOREIGN KEY(LID) REFERENCES LANGUAGE);
```

- นักเรียนบางคนเป็นนักกีฬา นักเรียนบางคนเป็นนักดนตรี แต่ไม่มีนักเรียนคนใดเป็นทั้งสองอย่าง แต่มีนักเรียนบางคนไม่เป็นทั้งสองอย่าง
- เรียกว่า Partial-Exclusive

```
CREATE TABLE CLUB
( CID  CHAR(10)   NOT NULL,
  PRIMARY KEY(CID));
CREATE TABLE STUDENT
( ID    CHAR(10)   NOT NULL,   CID CHAR(10),  NAME  CHAR(50)  NOT NULL,
  PRIMARY KEY(ID),   FOREIGN KEY(CID) REFERENCES CLUB);
CREATE TABLE SPORT
(SID   CHAR(10)  NOT NULL,  PRIMARY KEY(SID),
 FOREIGN KEY(SID) REFERENCES CLUB);
CREATE TABLE MISIC
(MID  CHAR(10)   NOT NULL,  PRIMARY KEY(MID),
 FOREIGN KEY(MID) REFERENCES CLUB);
```

- นักเรียนบางคนเป็นชาย นักเรียนบางคนเป็นหัวหน้าห้อง แต่หัวหน้าห้องบางคนอาจไม่ใช่นักเรียนชาย และมีนักเรียนหลาย ๆ คนที่ไม่ใช่นักเรียนชายและไม่ได้เป็นหัวหน้าห้อง

- เรียกว่า Partial-Overlapping

```
CREATE TABLE STUDENT
( ID    CHAR(10)  NOT NULL,   NAME CHAR(50)  NOT NULL,  PRIMARY KEY(ID));
CREATE TABLE MALE
( MID CHAR(10)   NOT NULL,   PRIMARY KEY(MID),
  FOREIGN KEY(MID) REFERENCES STUDENT);
CREATE TABLE HEAD
(HID      CHAR(10)          NOT NULL,  PRIMARY KEY(HID),
 FOREIGN KEY(HID) REFERENCES STUDENT);
```

2.4 การแปลงแผนภาพคลาสที่มีแอสโซซิเอชัน

2.4.1 คลาสที่ความสัมพันธ์แบบ One-to-One Association

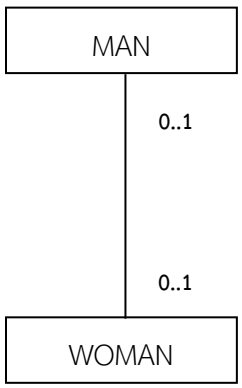
1) ออกแบบทั้งสองตารางโดยใช้หลักการตามข้อ 2.1 การแปลงคลาสให้เป็นตาราง

2) ให้เลือกเอา Primary Key ของตารางใดก็ได้ไปเป็น Foreign Key ของอีกตารางหนึ่ง

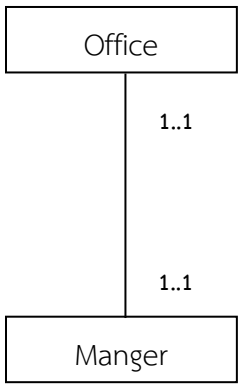
3) การใส่ Primary Key ให้พิจารณาว่า Table ที่ถูกอ้างนั้นมี Minimum Cardinality เป็นอะไร ถ้าเป็น 1..1 Foreign Key จะเป็นค่าว่างไม่ได้

ตัวอย่างที่ 10.7 การแปลงคลาสที่มีความสัมพันธ์แบบ One-to-One Association

สามารถแปลงเป็นตารางได้ดังนี้




```
CREATE TABLE WOMAN
(       WOMANID      CHAR(10) NOT NULL
        PRIMARY KEY(WOMANID)
);
CREATE TABLE MAN
(       MANID CHAR(10) NOT NULL,
        WOMANID      CHAR(10)       ,
        PRIMARY KEY(MANID),
        FOREIGN KEY (WOMANID) REFERENCES WOMAN
);
```

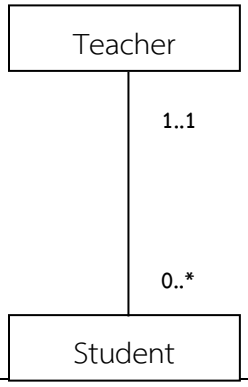



```
CREATE TABLE OFFICE
(       OFFICEID        CHAR(10) NOT NULL
        PRIMARY KEY(OFFICEID)
);
CREATE TABLE MANAGER
(       MANGID          CHAR(10) NOT NULL,
        OFFICEID        CHAR(10) NOT NULL,
        PRIMARY KEY(MANGID),
        FOREIGN KEY (OFFICEID) REFERENCES OFFICE
);
```

2.4.2 คลาสที่ความสัมพันธ์แบบ One-to-Many Association

One-To-Many Association

1) แปลง Class ทั้งสองให้เป็น Table ตามหลักการการแปลง Class ให้เป็น Table พร้อมทั้งกำหนด Primary Key ของทั้งสองให้เหมาะสม
2) วาดภาพความสัมพันธ์แบบ Association ระหว่าง Table ทั้งสอง พร้อมทั้งระบุ Minimum และ Maximum Multiplicity ให้ครบถ้วน
3) ให้นำเอา Primary Key ของ Table ที่มี Maximum Multiplicity เป็น 1 ไปเป็น Foreign Key ของ Table ที่มี Maximum Multiplicity มากกว่า 1
4) ใน Table ที่ Maximum Multiplicity ให้พิจารณาว่า Minimum Multiplicity ของ Table เป็น
   4.1) กรณีที่ Minimum Multiplicity เป็น 0 ให้กำหนดให้ Foreign Key มีคุณสมบัติเป็น NULL
   4.2) กรณีที่ Minimum Multiplicity เป็น 1 ให้กำหนดให้ Foreign Key มีคุณสมบัติเป็น NOT NULL



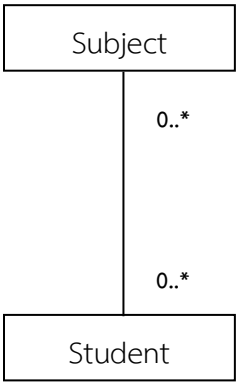

```
CREATE TABLE TEACHER
(       TCHID  CHAR(10) NOT NULL
        PRIMARY KEY(TCHID)
);
CREATE TABLE STUDENT
(       STDID           CHAR(10) NOT NULL,
        TCHID  CHAR(10) NOT NULL,
```

	PRIMARY KEY(STDID), FOREIGN KEY (TCHID) REFERENCES TEACHER);

2.4.2 คลาสที่ความสัมพันธ์แบบ Many-To-Many Association

Many-To-Many Association

1) แปลง Class ทั้งสองให้เป็น Table ตามหลักการการแปลง Class ให้เป็น Table พร้อมทั้งกำหนด Primary Key ของทั้งสองให้เหมาะสม
2) วาดภาพความสัมพันธ์แบบ Association ระหว่าง Table ทั้งสอง พร้อมทั้งระบุ Minimum และ Maximum Multiplicity ให้ครบถ้วน
3) ในกรณีที่ Many-to-Many Association นั้นยังไม่มี Association Class ให้สร้าง Associate Table ขึ้น พร้อมทั้งให้ Associate Table ที่สร้างใหม่มีความสัมพันธ์กับ Table ทั้งสองแบบ One-to-Many โดยให้ Associate Table นั้นมี Minimum และ Maximum Multiplicity เป็น 0 และ N ตามลำดับ และให้ Class ทั้งสองมี Minimum และ Maximum Multiplicity เป็น 1
4) ในกรณีที่พบว่า ใน Persistent Class Diagram มี Association Class สามารถแปลงเป็น Associate Table ได้ทันที
5) ไม่ว่า Associate Table จะมาด้วยวิธีการใด ให้ใส่ Primary Key ของ Table ทั้งสองข้างของ Association ให้เป็น Primary Key ของ Associate Table และให้ Primary Key ของ Associate Table ทำหน้าที่เป็น Foreign Key ที่อ้างอิงไปยัง Table ทั้งสองด้วย
6) ใช้หลักการของการสร้าง Table จาก One-to-Many เพื่อจัดการกับ Table และ Associate Table และสร้าง DLL

	CREATE TABLE STUDENT (STDID CHAR(10) NOT NULL PRIMARY KEY(STDID)); CREATE TABLE SUBJECT (SUBID CHAR(10) NOT NULL, PRIMARY KEY(SUBID)); CREATE TABLE STD_SUB (STDID CHAR(10),

	SUBID CHAR(10), PRIMARY KEY(STDID,SUBID), FOREIGN KEY(STDID) REFERENCES STUDENT, FOREIGN KEY(SUBID) REFERENCES SUBJECT);

การออกแบบสถาปัตยกรรมระบบ

การออกแบบสถาปัตยกรรมระบบ (System Architecture Design) หมายถึงการออกแบบในส่วนที่เป็นฮาร์ดแวร์และเครือข่ายที่มีความเกี่ยวข้องกันภายในระบบที่พัฒนา ซึ่งขั้นตอนการออกแบบสถาปัตยกรรมระบบจัดได้ว่าเป็นขั้นตอนสำคัญในระยะการออกแบบ โดยจะเกี่ยวข้องกับการวางแผนด้านฮาร์ดแวร์ ซอฟต์แวร์ และโครงสร้างของระบบเครือข่ายที่ใช้ในการพัฒนาระบบตามแนวทางเชิงวัตถุ ซึ่งในยูเอ็มแอลจะใช้แผนภาพคอมโพเน้นท์ (Component Diagram) ในการออกแบบสถาปัตยกรรมแอปพลิเคชัน และใช้แผนภาพดีพลอยด์เม้นท์ (Deployment Diagram) สำหรับการออกแบบสถาปัตยกรรมของฮาร์ดแวร์โดยสถาปัตยกรรมของฮาร์ดแวร์มีรูปแบบต่าง ๆ ได้แก่ สถาปัตยกรรมเครือข่ายแบบรวมศูนย์ สถาปัตยกรรมเครือข่ายแบบไฟล์เซิร์ฟเวอร์ สถาปัตยกรรมเครือข่ายแบบไคลเอนต์เซิร์ฟเวอร์ และ สถาปัตยกรรมเครือข่ายแบบไคลเอนต์เซิร์ฟเวอร์เทียร์ (สันติสุข แก้วไทย และคณะ, 2563)

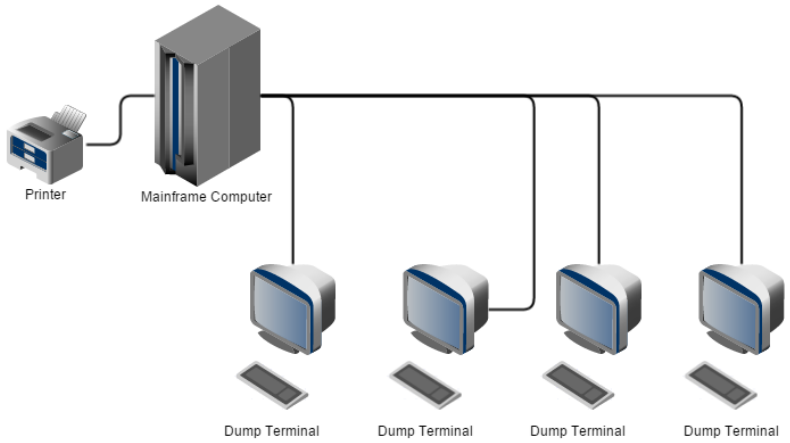
ประเภทของสถาปัตยกรรมระบบ

การออกแบบสถาปัตยกรรมระบบจัดได้ว่าเป็นขั้นตอนสำคัญในระยะการออกแบบ ซึ่งเกี่ยวข้องกับการวางแผนทั้งด้านฮาร์ดแวร์ ซอฟต์แวร์ และโครงสร้างของระบบเครือข่าย โดยสถาปัตยกรรมแบ่งออกเป็นประเภทต่าง ๆ ดังนี้

1. สถาปัตยกรรมระบบแบบรวมศูนย์
2. สถาปัตยกรรมระบบแบบไฟล์เซิร์ฟเวอร์
3. สถาปัตยกรรมระบบแบบไคลเอนต์เซิร์ฟเวอร์
4. สถาปัตยกรรมระบบแบบไคลเอนต์เซิร์ฟเวอร์เทียร์

1. สถาปัตยกรรมเครือข่ายแบบรวมศูนย์

สถาปัตยกรรมเครือข่ายแบบรวมศูนย์เป็นสถาปัตยกรรมที่ใช้เมนเฟรมคอมพิวเตอร์เป็นศูนย์กลาง ที่เรียกว่าโฮสต์ (Host) โดยจะมีเครื่องลูกข่ายเชื่อมต่อเข้ากับโฮสต์คอมพิวเตอร์ที่เรียกว่าเครื่องเทอร์มินัล (Terminal) ดังแสดงในภาพที่ 10.8



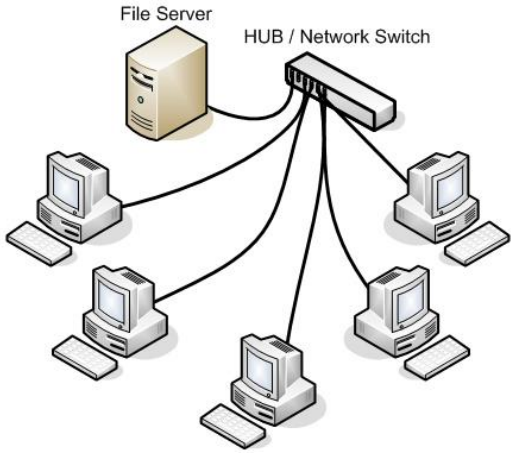
ภาพที่ 10.8 สถาปัตยกรรมเครือข่ายแบบรวมศูนย์

(ที่มา : http://chayo-website.blogspot.com/2015_02_01_archive.html)

จากรูปสถาปัตยกรรมแบบนี้จะเรียกเครื่องแม่ข่ายว่าโฮส (Host) ซึ่งมีความหมายเดียวกันกับ คำว่าเซิร์ฟเวอร์ โดยมีข้อดีคือเมื่อเป็นระบบควบคุมแบบรวมศูนย์ (Centralized Control) จะทำให้ง่ายต่อการจัดการโค้ดและดีบัก (Code and Debug) โปรแกรม แต่มีข้อเสีย คืออาจจะทำให้เกิดปัญหาคอขวดที่เซิร์ฟเวอร์ เนื่องจากงานทั้งหมดถูกทำที่เซิร์ฟเวอร์ซึ่งจะส่งผลให้มีประสิทธิภาพการทำงาน (Performance) จำกัด ในการอัพเกรดยุ่งยาก และมีราคาแพง

2. สถาปัตยกรรมเครือข่ายแบบไฟล์เซิร์ฟเวอร์

สถาปัตยกรรมเครือข่ายแบบไฟล์เซิร์ฟเวอร์ เป็นสถาปัตยกรรมที่นำพีซีคอมพิวเตอร์ที่นำมาทำเป็นเครื่องแม่ข่ายหรือเซิร์ฟเวอร์ ที่มีหน้าที่บริการแชร์ทรัพยากรต่าง ๆ ให้กับเครื่องลูกข่ายซึ่งประกอบด้วยการบริการแชร์ไฟล์ โปรแกรมเครื่องพิมพ์และเนื้อที่จัดเก็บข้อมูล เป็นต้นดังแสดงในภาพที่ 10.9

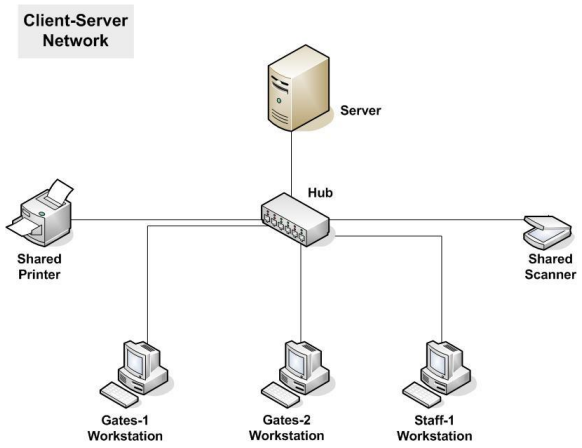


ภาพที่ 10.9 สถาปัตยกรรมเครือข่ายแบบไฟล์เซิร์ฟเวอร์

(ที่มา : http://not-comnetwork.blogspot.com/)

3. สถาปัตยกรรมเครือข่ายแบบไคลเอนต์เซิร์ฟเวอร์

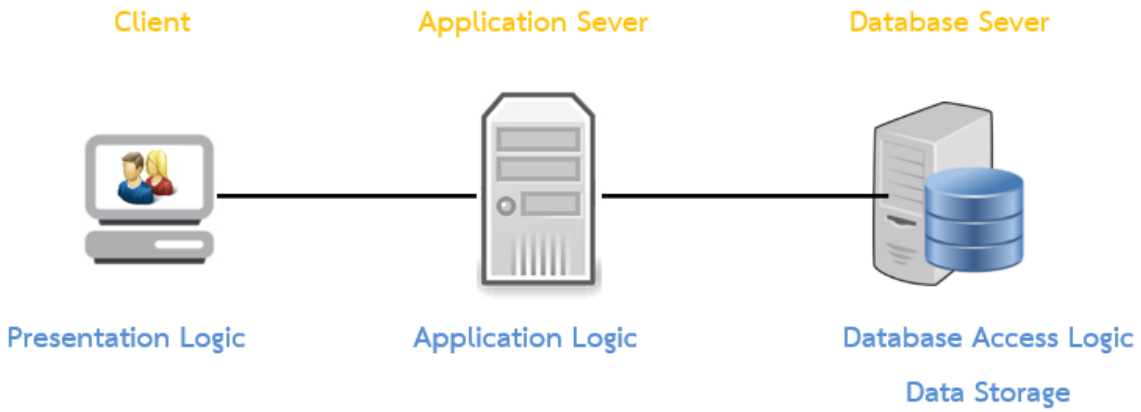
สถาปัตยกรรมเครือข่ายแบบไคลเอนต์เซิร์ฟเวอร์ เป็นสถาปัตยกรรมที่ประสิทธิภาพสูงเนื่องจากจะมีการแบ่งการประมวลผลระหว่างเครื่องเซิร์ฟเวอร์และเครื่องไคลเอนต์ในขณะเดียวกัน ตัวอย่างเช่นเมื่อไคลเอนต์ร้องขอข้อมูลจากเซิร์ฟเวอร์เครื่องเซิร์ฟเวอร์จะส่งข้อมูลส่วนที่ต้องการมาให้ยังเครื่องไคลเอนต์เท่านั้น ในขณะที่สถาปัตยกรรมเครือข่ายแบบไคลเอนต์เซิร์ฟเวอร์จะส่งข้อมูลไปให้เครื่องไคลเอนต์จัดการแทนทำให้ภาระงานตกอยู่กับเครื่องไคลเอนต์เป็นหลัก ไม่ว่าจะเป็นการล็อคเรคอร์ดการเลือกไฟล์เพื่อจัดการเก็บข้อมูลเป็นต้นแต่สำหรับสถาปัตยกรรมเครือข่ายแบบไคลเอนต์เซิร์ฟเวอร์จะมีแอปพลิเคชันที่พัฒนาขึ้นบนสภาพแวดล้อมแบบไคลเอนต์เซิร์ฟเวอร์ที่บรรจุอยู่ศูนย์กลางคือเซิร์ฟเวอร์ โดยมีระบบจัดการฐานข้อมูลคอยจัดการอยู่เบื้องหลัง ดังนั้นการจัดการกับข้อมูลต่าง ๆ จะช่วยลดความยุ่งยากให้กับโปรแกรมเมอร์มาก



ภาพที่ 10.10 สถาปัตยกรรมเครือข่ายแบบไคลเอนต์เซิร์ฟเวอร์
(ที่มา : http://not-comnetwork.blogspot.com/)

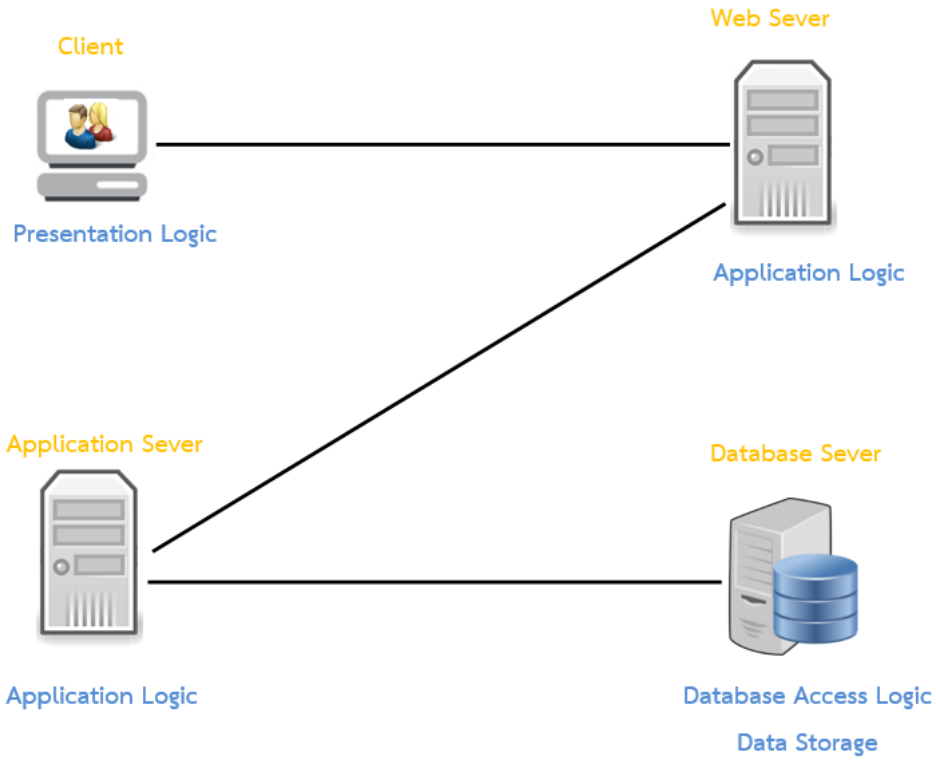
4. สถาปัตยกรรมเครือข่ายแบบไคลเอนต์เซิร์ฟเวอร์เทียร์

สถาปัตยกรรมเครือข่ายแบบไคลเอนต์เซิร์ฟเวอร์เทียร์คือสถาปัตยกรรมที่นำเซิร์ฟเวอร์มากกว่าหนึ่งตัวมาช่วยงานประมวลผลเฉพาะด้านทั้งนี้ช่วยให้ระบบสามารถทำงานได้อย่างมีประสิทธิภาพ เช่น สถาปัตยกรรมแบบทรีเทียร์(Tee-Tiered) ที่มีการนำคอมพิวเตอร์จำนวน 3 เครื่องดังภาพที่ 10.3 ที่ประกอบด้วยเครื่องไคลเอนต์แอปพลิเคชันเซิร์ฟเวอร์และดาต้าเบสเซิร์ฟเวอร์ ในขณะที่ภาพที่ 10.4 เป็นสถาปัตยกรรมแบบโฟร์เทียร์ (Four-Tiered) ที่ประกอบด้วยเครื่องไคลเอนต์เว็บเซิร์ฟเวอร์แอปพลิเคชันเซิร์ฟเวอร์ และดาต้าเบสเซิร์ฟเวอร์ซึ่งเป็นระบบที่รองรับการเชื่อมโยงเข้ากับเครือข่ายอินเตอร์เน็ตโดยมีเว็บเซิร์ฟเวอร์เป็นเครื่องแม่ข่ายและมีดาต้าเบสเซิร์ฟเวอร์เป็นที่จัดเก็บข้อมูลและแอปพลิเคชันเซิร์ฟเวอร์ที่บรรจุไปด้วยแอปพลิเคชันโปรแกรมต่าง ๆ



ภาพที่ 10.11 สถาปัตยกรรมแบบทรีเทียร์ (Tee-Tiered)

(ที่มา : https://sites.google.com/site/karxxkbaebrabbtxnthi1/kar-xxkbaeb-sthapatykrrm**)**



ภาพที่ 10.12 สถาปัตยกรรมแบบโฟร์เทียร์ (Four-Tiered)

(ที่มา : https://sites.google.com/site/karxxkbaebrabbtxnthi1/kar-xxkbaeb-sthapatykrrm**)**

จากภาพที่ 10.11 และ 10.12 เป็นสถาปัตยกรรมแบบแบบไคลเอนต์เซิร์ฟเวอร์ โดยมีข้อดีคือประสิทธิภาพในการทำงานดีกว่าแบบอื่น ๆ เนื่องจากกระจายงานกันทำ แต่มีข้อเสียคือการพัฒนาซอฟต์แวร์

ยากกว่าเพราะต้องดีบัก (Debug) ข้ามเครือข่าย อีกทั้งในการแบ่งงานกันทำบางครั้ง Application logic กับ Data access logic มาจากคนละยี่ห้อกัน จึงอาจไม่เข้ากันทำให้ต้องมี Middleware เกิดขึ้น

แผนภาพดีพลอยด์เม้นท์

ในการวิเคราะห์และออกแบบระบบโดยใช้หลักการเชิงวัตถุ (Object Oriented Analysis and Design) นั้น จะถือว่าทุก ๆ ส่วนประกอบของระบบนั้นเป็นวัตถุตัวหนึ่งเสมอ ซึ่งวัตถุนั้น ไม่ได้หมายถึงเฉพาะส่วนที่อยู่ในแอปพลิเคชันหรือซอฟต์แวร์เท่านั้น แต่ยังหมายรวมถึงเครื่องคอมพิวเตอร์หรือฮาร์ดแวร์ และเครือข่ายคอมพิวเตอร์อีกด้วย ซึ่งถือเป็นข้อได้เปรียบของการวิเคราะห์และออกแบบระบบเชิงวัตถุเพราะสามารถออกแบบซอฟต์แวร์และฮาร์ดแวร์โดยไม่ต้องเปลี่ยนหลักการที่ใช้เพื่อการออกแบบ เพราะทั้งซอฟต์แวร์และฮาร์ดแวร์ต่างก็ถือเป็นวัตถุเหมือนกัน การออกแบบในส่วนของฮาร์ดแวร์ของระบบนั้นเรียกว่าการออกแบบสถาปัตยกรรมระบบ (System Architecture Design) โดยเครื่องมือที่ใช้ในการออกแบบสถาปัตยกรรมระบบนั้นเรียกว่าแผนภาพดีพลอยด์เม้นท์ (Deployment Diagram) ซึ่งแผนภาพดีพลอยด์เม้นท์ เป็นแผนภาพที่มีลักษณะคล้ายกับแผนภาพคลาส โดยส่วนประกอบทางฮาร์ดแวร์ (Hardware Module) ตัวหนึ่ง ๆ ใน แผนภาพดีพลอยด์เม้นท์ซึ่งเปรียบเทียบได้กับคลาสหนึ่งในแผนภาพคลาสนั่นเอง โดยมีรายละเอียดดังต่อไปนี้

ความหมายของแผนภาพดีพลอยด์เม้นท์

แผนภาพดีพลอยด์เม้นท์ หมายถึง แผนภาพแสดงสถาปัตยกรรมของระบบในลักษณะสถาปัตยกรรมเชิงกายภาพ (Physical Architecture) เพื่อแสดงโครงสร้างการใช้ทรัพยากรฮาร์ดแวร์และซอฟต์แวร์ ซึ่งรวมไปถึงการแสดงความสัมพันธ์ของอุปกรณ์ในระบบ ซึ่งเป็นส่วนหนึ่งของการวิเคราะห์และออกแบบระบบเชิงวัตถุ โดยเปนแผนภาพแสดงโครงสรางทางดานฮารดแวรของระบบในขณะที่ทำงานจริงซึ่งไดแกหนวยประมวลผล(Processor) อุปกรณคอมพิวเตอรตาง ๆ (Devices) ตลอดจนองค์ประกอบทางซอฟต์แวร์ (Software Component) ที่ไดออกแบบไวดวย (ชาคริต กุลไกรศรี, 2556)

แผนภาพดีพลอยด์เม้นท์ในภาษายูเอ็มแอลจะใชสัญลักษณ “ลูกบาศก(Cube)” เพื่อแสดงโหนด (Node) โดยชื่อของโหนด จะทำหนาที่เปน Unique Identity และมี Path Name เพื่อใชแสดงประเภทของฮาร์ดแวร์(Hardware) ที่โหนดที่วางอยู่โดยที่ชื่อพาธ (Path Name) จะใชสัญลักษณในรปแบบที่วางอยู่โดยชื่อพาธจะใช้สัญลักษณ์ในรูปแบบสเตอริโอไทป์ (Stereotype)

สัญลักษณ์องค์ประกอบหลักแผนภาพดีพลอยด์เม้นท์

องค์ประกอบหลักในการออกแบบระบบด้วยแผนภาพดีพลอยด์เม้นท์ (Deployment Diagram) มีดังต่อไปนี้

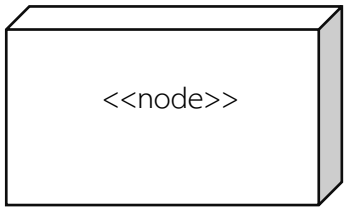
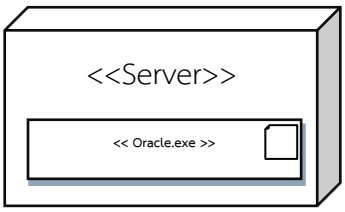
องค์ประกอบของซอฟต์แวร์ (Software)

1. Data Storage แหล่งเก็บข้อมูล
2. Data Access Logical
3. Application Logical

องค์ประกอบของฮาร์ดแวร์ (Hardware)

1. Server Computer
2. Client Computer
3. Connecting Network

ตารางที่ 10.3 สัญลักษณ์ของแผนภาพดีพลอยด์เม้นท์

สัญลักษณ์	ชื่อ	ความหมายและหน้าที่
	Node	เป็นสัญลักษณ์ที่ใช้แสดงแทนอุปกรณ์ (Hardware) ในแต่ละระบบ
	Application Name	เป็นสัญลักษณ์ ที่ใช้แสดงส่วนของ ซอฟต์แวร์ หรือ ฐานข้อมูลที่ใช้ภาย Node
	Application in Node	จากภาพตัวอย่าง แสดงถึงอุปกรณ์ Server โดยมี ซอฟแวร์ Oracle อยู่ภายใต้ Node
<< Communication Path >>	Communication Path	ใช้แสดง การเชื่อมต่อระบบระหว่าง อุปกรณ์ โดยระบุรูปแบบ การเชื่อมต่อ เช่น << Lan >> << TPC/IP >>

ที่มา: https://sites.google.com/site/umldeployment/project-definition (อรรถพล ดุลลาพันธ์ และคณะ, 2563)

สัญลักษณ์อื่น ๆ

บางครั้งในการสร้างแผนภาพดีพลอยด์เม้นท์อาจมีสัญลักษณ์อื่น ๆ เพิ่มเติมเข้ามา เช่น เมื่อระบบต้องใช้ระบบอินเทอร์เน็ตด้วย อาจต้องมีสัญลักษณ์เฉพาะเพื่อใช้แทนเครือข่ายอินเทอร์เน็ต (ปกติเป็นรูปเมฆ) เป็นต้น ซึ่งสัญลักษณ์ที่เพิ่มเข้ามาใหม่นี้ มักจะมีเพิ่มขึ้นตลอดเวลาตามการพัฒนาของเทคโนโลยี

ตัวอย่างการออกแบบสถาปัตยกรรมของระบบโดยใช้แผนภาพดีพลอยด์เม้นท์

ตัวอย่างที่ 1 ระบบการเรียกดูเอกสารผ่านทางอีเมล์

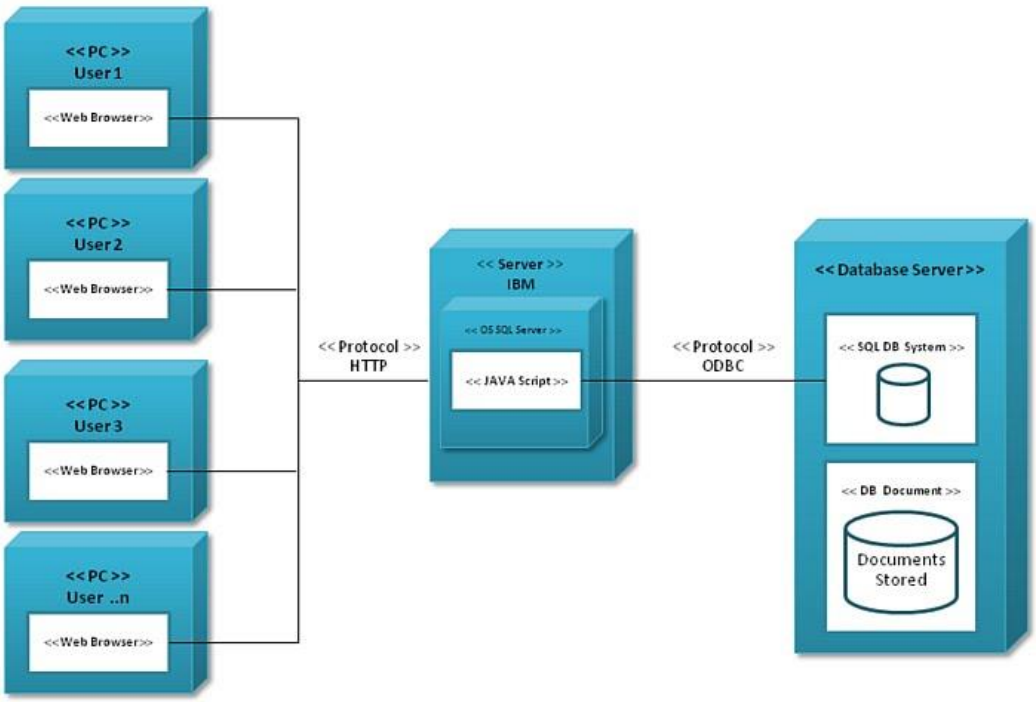
บริษัทแห่งหนึ่งต้องการวางระบบ การจัดการเรื่องการดูเอกสาร ผ่านระบบเครือข่าย (Network) ภายในบริษัทเพื่อแก้ปัญหาในการเรียกดูเอกสารจากส่วนกลางเพราะต้องคอยส่งอีเมล์ (e-mail) ร้องขอไปยังส่วนกลางจึงต้องการสร้างระบบการเรียกดูเอกสารแบบเครือข่ายขึ้นมา โดยได้กำหนดรายละเอียด ดังนี้

1. ระบบ ต้องสามารถเชื่อมต่อ เป็นเครือข่ายผ่านเว็บเบราเซอร์ (Web Browser)
2. โปรแกรมประยุกต์บนเว็บ (Web Application) ใช้ภาษาพีเฮชพี (PHP) และจาวาสคริปต์ (Java Script)
3. ระบบปฏิบัติการวินโดวส์เซิร์ฟเวอร์ (Windows Server)
4. Server Stored IBM
5. ระบบ Database ใช้ SQL Server

การวิเคราะห์ระบบ

ระบบที่ต้องการเป็นแบบเครือข่ายโดยมีศูนย์กลาง เครือข่ายที่ต้องเป็นผู้ให้บริการข้อมูล แก่ ผู้ใช้งานเฉพาะภายในองค์กรเท่านั้น จึงต้องใช้เครือข่ายแบบภายในองค์กร (LAN) และใช้ ระบบอินทราเน็ต (Intranet) ในการเชื่อมต่อ ผ่านระบบเว็บบราวเซอร์ (Web Browser) ผ่านการเชื่อมต่อแบบโพรโตคอล (Protocol) และต้องมีผู้ให้บริการ (Server IBM) เป็นผู้ให้บริการ โดยการเรียกดูข้อมูลในเอสคิวแอลเซิร์ฟเวอร์ (SQL Server) เพื่อดึงข้อมูล ให้กับผู้ใช้งานที่ร้องขอผ่านเว็บแอปพลิเคชันโดยใช้จาวาสคริปต์ (Java Script) เป็นซอฟต์แวร์ในการเชื่อมต่อกับผู้ใช้งาน (Booch, g. and author, 2007)

ในส่วนของการเชื่อมต่อกับฐานข้อมูล ใช้ลักษณะการเชื่อมต่อแบบ Protocol โดยใช้ ODBC(Open Database Connectivity) ซึ่งเป็น ส่วนที่สามารถ ติดต่อกับผู้ใช้ได้อย่างมีประสิทธิภาพ โดยผ่านมุมมองของโปรแกรมประยุกต์บนเว็บ (Web Application) ส่วนภายในฐานข้อมูลเซิร์ฟเวอร์ (Database Server) จะประกอบไปด้วยส่วนที่เป็นระบบทำหน้าที่จัดระบบการทำงาน และส่วนที่เป็นพื้นที่ในการเก็บข้อมูล



ภาพที่ 10.13 สถาปัตยกรรมของระบบการเรียกดูเอกสารแบบเครือข่าย

อธิบายแผนภาพ

- โหนด (Node) ซ้ายมือ หมายถึง กลุ่มของเครื่องคอมพิวเตอร์ส่วนบุคคลสำหรับผู้ใช้งาน (PC user) ซึ่งไม่ได้จำกัดปริมาณไว้ (User .. n) เชื่อมต่อเครือข่ายไปยังเซิร์ฟเวอร์โดยวิธี Link แบบ Protocol ผ่าน Web Browser ไปยัง Server IBM

- ภายใน Server IBM จะประกอบไปด้วย ระบบการจัดการ OS SQL Server โดยมี Component Web Application ซึ่งใช้จาวาสคริปซ์ (Java Script) เป็น Software เป็นตัวดำเนินการเพื่อเชื่อมต่อไปยังฐานข้อมูลเซิร์ฟเวอร์ (Database Server) โดยใช้ Link Protocol ด้วยวิธีเชื่อมต่อฐานข้อมูลผ่านทาง ODBC Service

- ฐานข้อมูลเซิร์ฟเวอร์ (Database Server) ภายในจะแบ่งออกเป็น 2 ส่วน คือ ส่วนที่เป็นระบบ (System) และ ส่วนที่เป็นเอกสารข้อมูล (Data Documents) โดยระบบฐานข้อมูล (DB System) จะกำหนดแอดเดรส (Address) และ ข้อมูลที่เครื่องลูกข่าย (Client) ร้องขอผ่านทางเว็บแอปพลิเคชัน (Web Application) ไปยังตัวข้อมูลเพื่อดึงข้อมูลส่งกลับไปยังผู้ใช้งาน (User)

ตัวอย่างที่ 2 ระบบบริการลูกค้าบริษัทประกันชีวิต

บริษัทประกันชีวิตแห่งหนึ่งต้องการใช้เทคโนโลยีการสื่อสารแบบไร้สายเพื่ออำนวยความสะดวกให้กับตัวแทนฝ่ายขาย จึงต้องสร้างระบบการขายผ่านอุปกรณ์ Tablet หรือคอมพิวเตอร์ส่วนบุคคล (Personal Computer) เพื่ออำนวยความสะดวกให้กับลูกค้าและตัวแทน โดยมีข้อกำหนดดังนี้

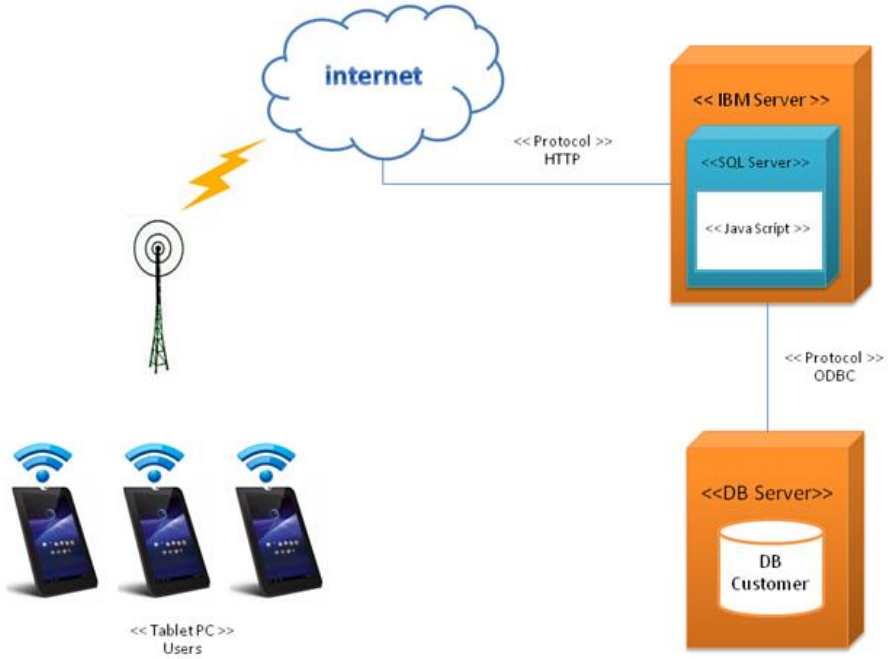
1) ใช้อุปกรณ์แท็ปเล็ต (Tablet) ในการบันทึกข้อมูลการขายของลูกค้าเพื่อส่งข้อมูลมายังสำนักงานใหญ่

2) ระบบต้องรองรับการใช้งานแบบไร้สายและการสื่อสารแบบเรียลไทม์ (Real time)

3) ระบบฐานข้อมูลจะใช้ไมโครซอฟต์เอสคิวแอลเซิร์ฟเวอร์ (MS-SQL Server)

วิเคราะห์ระบบ

เนื่องจากความต้องการในการเชื่อมต่อระบบเป็นอุปกรณ์ไร้สายและผู้ใช้งานอยู่นอกพื้นที่ จึงต้องวางระบบเครือข่ายเป็นแบบอินเตอร์เน็ต (Internet) โดยใช้อุปกรณ์ไร้สาย (Tablet) ในการติดต่อสื่อสารข้อมูล ผ่านเว็บแอปพลิเคชัน (Web Application) โดยการส่งสัญญาณผ่านจุดให้บริการเครือข่ายอินเตอร์เน็ตส่งผ่านข้อมูลไปยังเซิร์ฟเวอร์ผ่าน Protocol TCP/IP เพื่อเชื่อมต่อไปยังฐานข้อมูล



ภาพที่ 10.14 สถาปัตยกรรมของระบบการขายผ่านอุปกรณ์

(ที่มา : https://sites.google.com/site/umldeployment/home/study)

อธิบายแผนภาพ

จากภาพที่ 10.22 Node Tablet PC (ซึ่งเป็นอุปกรณ์สำหรับตัวแทนหรือนายหน้า) ใช้เข้าสู่ระบบ (Login) โดยผ่านเครือข่ายไร้สายเพื่อเชื่อมต่อ

Internet เพื่อเชื่อมต่อ (Link) ไปยัง IBM Server โดยภายใน Tablet จะมีส่วนของ Interface Web Application

ภายใน Server IBM จะประกอบไปด้วย ระบบการจัดการ OS SQL Server โดยมี Component Web Application ซึ่งใช้ภาษาจาวา (Java)

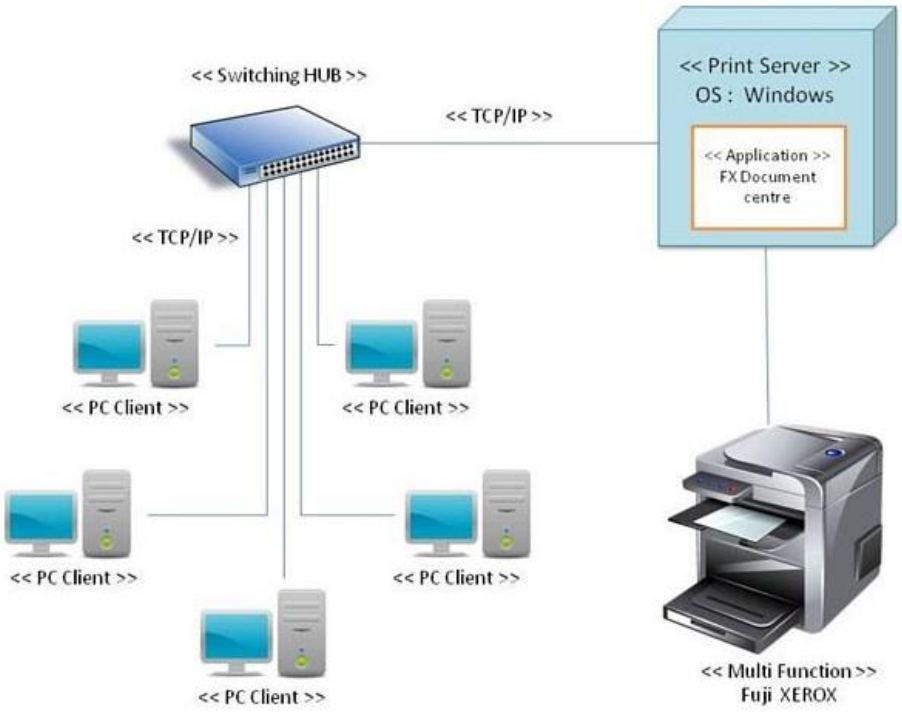
Script เป็นซอฟต์แวร์สำหรับดำเนินการเชื่อมต่อไปยังฐานข้อมูลฐานข้อมูลเซิร์ฟเวอร์ (Database Server) โดยใช้ Link Protocol เพื่อเชื่อมต่อฐานข้อมูลผ่านทาง ODBC Service เพื่อทำการบันทึกข้อมูลลูกค้าเข้าไปเก็บไว้ยังฐานข้อมูล

ตัวอย่างที่ 3 ระบบจัดการเครื่องพิมพ์เอกสารภายในองค์กร

บริษัทแห่งหนึ่งต้องการประหยัดค่าใช้จ่ายในการสั่งซื้อกระดาษ เพื่อใช้พิมพ์งานเอกสารต่าง ๆ เนื่องจากในแต่ละหน่วยงานจะมีเครื่องพิมพ์อยู่หลายเครื่อง ทำให้เสียค่าใช้จ่ายในการบำรุงรักษาอุปกรณ์ผ้าหมึกมากมาย จึงต้องการสร้างระบบ Printer Pool เพื่อลดค่าใช้จ่ายต่าง ๆ ลง โดยใช้เครื่องพิมพ์แบบเอนกประสงค์ (Multi-Function) คือ เป็นทั้งเครื่องถ่ายเอกสารและเครื่องพิมพ์ในตัวเดียวกัน

การวิเคราะห์ระบบ

จากความต้องการของระบบเป็นแบบภายในองค์กร จึงต้องมีการวางระบบเครือข่ายแบบท้องถิ่น (LAN) เพราะเสียค่าใช้จ่ายน้อยและจะต้องมีเครื่องพิมพ์เซิร์ฟเวอร์ (Printer Server) เพื่อจัดระบบการสั่งงานพิมพ์ผ่านเครือข่ายจาก User PC โดยใช้ Link LAN IP address เป็นตัวบ่งบอกเครื่องที่ทำการเชื่อมต่อ ซึ่งสามารถออกแบบแผนภาพได้ดังนี้



ภาพที่ 10.15 สถาปัตยกรรมของระบบ Printer Pool

(ที่มา: http://diagram-it54.blogspot.com/p/blog-page_25.html)

อธิบายแผนภาพ

จากภาพที่ 10.23 เป็นการเชื่อมต่อแบบ LAN Network โดยใช้การ Link แบบ TCP/IP เป็นตัวกำหนดแอดเดรส (Address) ของเครือข่ายลูก โดยเชื่อมโยงไปยัง Switching HUB เพื่อทำการจัดการคิวสัญญาณ เพื่อส่งต่อไปยังเครื่องพิมพ์เซิร์ฟเวอร์

- เครื่องพิมพ์เซิร์ฟเวอร์ภายในประกอบไปด้วย ระบบปฏิบัติการวินโดวส์ (Windows) โดยมี Application FX Document center เป็นซอฟแวร์ควบคุมการทำงานเพื่อสั่งงานพิมพ์ไปยังเครื่องพิมพ์ Fuji ซึ่งเป็นเครื่องพิมพ์เอกสารแบบเอนกประสงค์

ตัวอย่างที่ 4 ระบบจองตั๋วภาพยนตร์

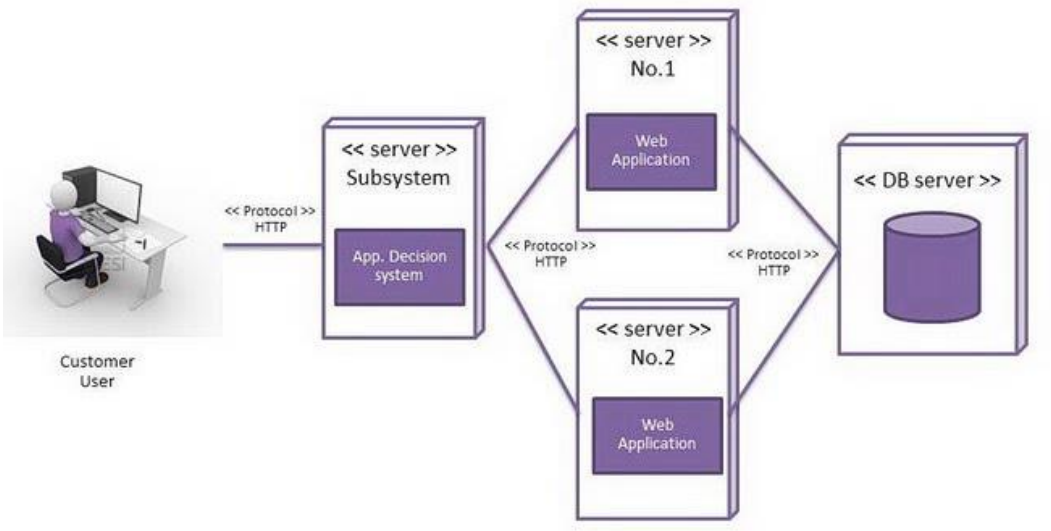
บริษัท Ticket Master ต้องการสร้างระบบการจองตั๋วชมการแสดงผ่านระบบอินเตอร์เน็ต โดยมีความต้องการเบื้องต้น ดังนี้

1) ระบบการจอง ต้องเป็นเครือข่ายอินเตอร์เน็ต

2) กรณี มีผู้ใช้จำนวนมาก ระบบ ต้องมีการป้องกัน ให้สามารถ ใช้งานได้ตลอด

วิเคราะห์ระบบ

ลักษณะการเชื่อมต่อเครือข่ายเป็นแบบอินเตอร์เน็ต และจะต้องมี Server 2 Node เพื่อเป็นการป้องกันระบบ ล่มในกรณีที่มีการเข้าใช้ในเวลาใกล้เคียงกัน เป็นจำนวนมาก ซึ่งจะใช้ระบบ Sub System server เพื่อเป็นตัวตัดสินใจที่จะใช้การติดต่อสื่อสารกับเซิร์ฟเวอร์เพื่อรองรับในกรณีผู้ใช้ มีจำนวนมาก



ภาพที่ 10.16 สถาปัตยกรรมของระบบการจองตั๋วชมการแสดง

(ที่มา: http://diagram-it54.blogspot.com/p/blog-page_25.html)

อธิบายแผนภาพ

จากภาพที่ 10.24 Customer User Node เชื่อมต่ออินเตอร์เน็ตผ่านระบบ Protocol Link มายัง Server subsystem

Node Subsystem ทำหน้าที่ช่วยตัดสินใจ เพื่อเลือก Server No.1 หรือ Server No.2 ว่า Server ใด มีการเข้าใช้งานน้อยกว่า หรือ ในกรณีเซิร์ฟเวอร์ใดเซิร์ฟเวอร์หนึ่งล่มก็จะทำการเลือกเซิร์ฟเวอร์อีกตัวหนึ่งใช้งานแทน

Server No 1 และ 2 ภายในบรรจุเว็บแอปพลิเคชัน (Web Application) เพื่อเป็นส่วนเชื่อมต่อ (Interface) กับผู้ใช้ในการจองตั๋วแล้วนำข้อมูลไปเก็บไว้ในฐานข้อมูลเซิร์ฟเวอร์ (Database Server)

ดีไซน์แพตเทิร์น

ดีไซน์แพตเทิร์น (Design Patterns) คือแบบแผนหรือแนวทางที่ใช้ในการแก้ไขปัญหาที่เกิดขึ้นเสมอ ๆ ในการออกแบบคอมพิวเตอร์ซอฟต์แวร์ แบบแผนและแนวทางเหล่านี้ไม่ใช่รูปแบบตายตัวที่จะถูกนำไปใช้โดยตรง แต่เป็นการอธิบายแนวทางหรือโครงที่จะถูกนำไปประยุกต์ใช้ในสถานการณ์ต่าง ๆ กล่าวเฉพาะในทางการเขียนโปรแกรมเชิงวัตถุ ดีไซน์แพตเทิร์นจะแสดงความสัมพันธ์ต่อกันระหว่างคลาสหรืออ็อบเจกต์ต่าง ๆ โดยไม่จำเพาะเจาะจงการนำไปใช้งานในขั้นสุดท้าย ขั้นตอนวิธีไม่จัดเป็นดีไซน์แพตเทิร์นเพราะเป็นการแก้ปัญหาในทางการประมวลผลมากกว่าในทางการออกแบบ (Oestereich, B., 2002)

โดยดีไซน์แพตเทิร์นมีจุดเริ่มต้นจากหนังสือชื่อ A Pattern Language: Towns, Buildings,Construction แต่งโดยสถาปนิกชื่อ Christopher Alexander เมื่อปี ค.ศ. 1977 จากนั้น Kent Beck และWard Cunningham ริเริ่มนำเอาแนวคิดนี้มาทดลองใช้กับการเขียนโปรแกรมในปี ค.ศ. 1987 และได้นำเสนอผลงานในงานประชุม OOPSLA ในปีเดียวกันนั้น ดีไซน์แพตเทิร์นเริ่มเป็นที่นิยมในวงการวิทยาการคอมพิวเตอร์ในปี ค.ศ. 1994 หลังจากมีหนังสือที่แต่งโดย Erich Gamma, Richard Helm, Ralph Johnson และ John Vlissides (Gang of four: GoF) ชื่อ Design Patterns: Elements of Reusable Object-Oriented Software (ISBN 0-201-63361-2)

ประโยชน์

ดีไซน์แพตเทิร์นช่วยทำให้กระบวนการพัฒนาโปรแกรมรวดเร็วขึ้นเพราะเป็นตัวอย่างที่ผ่านการพิสูจน์ทดสอบมาแล้ว การออกแบบซอฟต์แวร์ที่ดีต้องเตรียมการสำหรับปัญหาที่อาจจะไม่พบจนกว่าจะเริ่มนำไปใช้งาน การใช้ดีไซน์แพตเทิร์นช่วยป้องกันปัญหาเล็กน้อยที่อาจจะลุกลามใหญ่โต ทั้งยังทำให้การทำความเข้าใจโค้ดง่ายขึ้นในหมู่ผู้ร่วมงานในทีมที่คุ้นเคยกับดีไซน์แพตเทิร์น

การจัดหมวดหมู่และแพตเทิร์น

การจัดหมวดหมู่ดีไซน์แพตเทิร์นตามหนังสือ Design Patterns แบ่งตามวัตถุประสงค์การใช้งาน ดังนี้

แพตเทิร์นการสร้างอ็อบเจกต์ (Creational Patterns)

- Abstract Factory
- Builder
- Factory Method
- Prototype
- Singleton

แพตเทิร์นโครงสร้าง (structural patterns)

- Adapter
- Bridge
- Composite
- Decorator
- Façade
- Flyweight
- Proxy

แพตเทิร์นพฤติกรรม (behavioral patterns)

- Chain of responsibility
- Command
- Interpreter
- Iterator
- Mediator
- Memento
- Observer
- State
- Strategy
- Template method
- Visitor

หมวดหมู่แพตเทิร์นอื่นๆ

- concurrency patterns
- architectural patterns

สรุป

การออกแบบระบบเชิงวัตถุเป็นการพิจารณาว่าระบบจะดำเนินการไปได้อย่างไร โดยเป็นการตัดสินใจว่าจะพัฒนาระบบใหม่ด้วยแนวทางใด เช่น พัฒนาระบบขึ้นเอง หรือจัดซื้อโปรแกรมสำเร็จรูป หรือว่าจ้างบุคคลหรือบริษัทภายนอกมาพัฒนาระบบใหม่ เป็นต้น ซึ่งเกี่ยวข้องกับการออกแบบสถาปัตยกรรมระบบ โดยจะประกอบด้วยอุปกรณ์ฮาร์ดแวร์ ซอฟต์แวร์ เครือข่าย และฐานข้อมูลที่ใช้ในระบบ รวมถึงการเริ่มโปรแกรมและการออกแบบหน้าจอที่ใช้งาน โดยนำวัตถุดิบที่ได้มาจากขั้นตอนของการวิเคราะห์ระบบเชิงวัตถุมาปรับปรุงและเพิ่มเติมรายละเอียดลงไปในแผนภาพต่าง ๆ ซึ่งระยะของการออกแบบจะให้ความสำคัญด้านองค์ประกอบอีกรูปแบบหนึ่ง คือ การออกแบบสถาปัตยกรรมระบบ (System Architecture Design) ซึ่งจะเป็นการอธิบายถึงสภาพแวดล้อมของระบบ หรือเทคนิคการทำงานของระบบ โดยผู้ทำการออกแบบจะต้องตัดสินใจเกี่ยวกับการประมวลผลว่าต้องการออกแบบในลักษณะใด ซึ่งขั้นตอนการออกแบบสถาปัตยกรรมระบบถือว่าเป็นขั้นตอนสำคัญที่สุดในระยะการออกแบบ

ชื่อ-นามสกุล	รหัส	สาขาวิชา	รุ่น/หมู่	คะแนน	ลายเซ็นต์อาจารย์

แบบฝึกหัดท้ายบท

1. จงอธิบายความหมายของสถาปัตยกรรมของระบบ

.....

.....

.....

.....

2. จงบอกประเภทของสถาปัตยกรรมของระบบ

.....

.....

.....

.....

3. จงออกแบบส่วนประกอบของซอฟต์แวร์และเขียนแผนภาพคอมโพเน้นท์ (Component Diagram) ของระบบงานต่อไปนี้
    1) ระบบงานสินค้าคงคลัง (Inventory) ใน Supermarket
    2) ระบบงานการขายสินค้าใน Supermarket
    3) ระบบงานการเก็บข้อมูลคะแนนนักศึกษา
    4) ระบบงานทะเบียนประวัตินักศึกษา
    5) ระบบงานการลงทะเบียนเรียนของนักศึกษา
4. จงบอกความหมายของข้อมูลแบบภาวร (Persistent Data)

.....

.....

.....

.....

.....

5. จงออกแบบ Class Diagram และสร้าง Relational Database จาก Problem Domain นี้

   “สวนสัตว์แห่งหนึ่งประกอบด้วยเขตสวนสัตย์ 6 เขต คือเขตสัตว์แอฟริกา เขตสัตว์เอเชีย เขตสัตว์เมืองหนาว เขตสัตว์ลักษณะแปลก เขตการแสดง เขตโรงพยาบาลสัตว์ ในแต่ละส่วนจะประกอบด้วยสัตว์นานาชนิด อาจจะมีทั้งสัตว์บก สัตว์น้ำ สัตว์ครึ่งบกครึ่งน้ำ สัตว์ชนิดหนึ่งจะอยู่ในเขตใดเขตหนึ่งเท่านั้น ในบางกรณีจะทีการโยกย้ายสัตว์เพื่อความเหมาะสม เช่น อาจจะมีการโยกย้ายสัตว์จากเขตอื่นมาอยู่ในเขตการแสดง หรือโยกย้ายสัตว์ที่ป่วยมาไว้ในเขตโรงพยาบาล”

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

6. จงวิเคราะห์และออกแบบ Deployment Diagram สำหรับระบบ Book Store Service System ตามเงื่อนไขดังตอไปนี้ “เนื่องจากคลังหนังสือกับรานหนังสืออยูคนละอาคารซึ่งอยูหางกันประมาณ 2 กิโลเมตร โดยสวนของระบบที่ดูแลการซื้อขายและยืมหนังสือนั้นจะทำงานอยูที่ร่าน แตระบบในสวนของการสั่งซื้อหนังสือและอื่น ๆ จะทำงานอยู่ที่สำนักงานที่คลังหนังสือ”

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

เอกสารอ้างอิง

กิตติ ภักดีวัฒนะกุล และพนิดา พานิชกุล. (2548). *คัมภีร์การพัฒนาระบบเชิงวัตถุด้วย UML และ Java.* กรุงเทพฯ: เคทีพี คอมพ์ แอนด์ คอนซัลท์.

กิตติพงษ์ กลมกล่อม. (2552). *การวิเคราะห์และออกแบบระบบเชิงวัตถุด้วย UML.* กรุงเทพฯ: เคทีพี คอมพ์ แอนด์ คอนซัลท์.

กิติ ภักดีวัฒนะกุล และกิตติพงษ์ กลมกล่อม. (2544). *UML: วิเคราะห์และออกแบบระบบเชิงวัตถุ.* กรุงเทพฯ: เคทีพี คอมพ์ แอนด์ คอนซัลท์.

ชาคริต กุลไกรศรี. (2556). *UML คืออะไร.* สืบค้น 29 มีนาคม 2563, จาก https://msit5.wordpress.com/2013/09/04/uml-คืออะไร/

นัฐพงศ์ ส่งเนียม. (2563). *สื่อการสอนรายวิชา OOAD : Object-Oriented Analysis and Design.* สืบค้น 25 ตุลาคม 2563, จาก http://www.siam2dev.net

ไม่ปรากฏ. (2558). *การออกแบบระบบ.* สืบค้น 20 มกราคม 2563, จาก http://kanyaresgm301.blogspot.com/

ศรีไพร ศักดิ์รุ่งพงศากุล และเจษฎาพร ยุทธนวิบูลย์ชัย. (2549). *ระบบสารสนเทศและเทคโนโลยีการจัดการความรู้.* กรุงเทพฯ: ซีเอ็ดยูเคชั่น.

สันติสุข แก้วไทย และคณะ. (2563). *การวิเคราะห์และออกแบบระบบสารสนเทศ.* สืบค้น 20 มกราคม 2563, จาก http://suntisuk001.blogspot.com/p/blog-page.html

อรรถพล ดุลลาพันธ์ และคณะ. (2563). *Deployment diagram.* สืบค้น 20 มกราคม 2563, จาก https://sites.google.com/site/umldeployment/home/study

Booch, G., Maksimchuk, R., J Engle, M., Young, B., Conalle, J. and Houston, K. (2007). Object-Oriented Analysis and Design with Applications. 3nd ed., Addison-Wesley.

Booch, G., Rumbaugh, J. and Jacobson, I. (2005). *The Unified Modeling Language User Guide.* 2nd ed., Addison-Wesley.

Oestereich, B. (2002). *Developing software with UML Object-oriented analysis and design in practice.* 2nd ed., Addison-Wesley.