พิมพ์เขียวดิจิทัล: สถาปัตยกรรม

สถาปัตยกรรมระบบซอฟต์แวร์สมัยใหม่

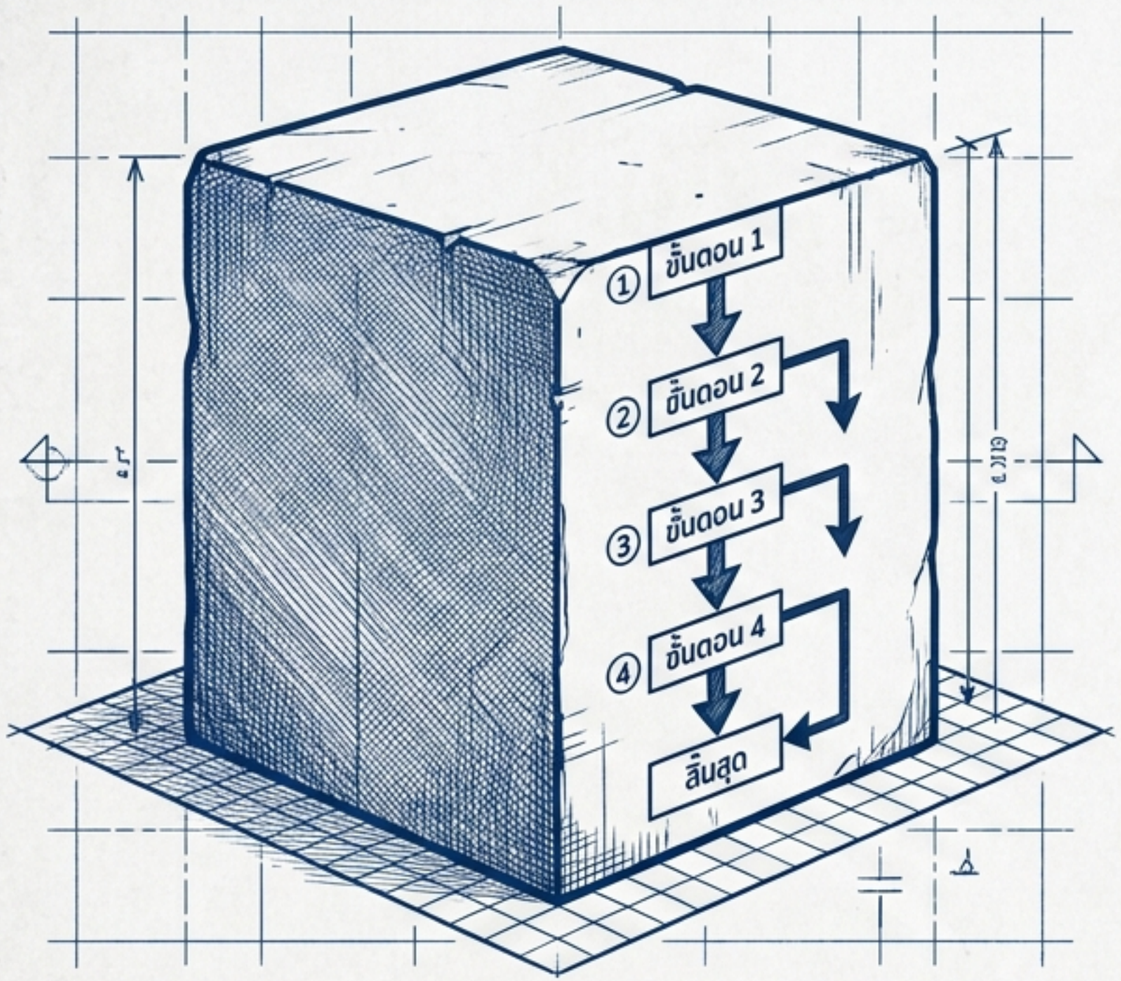
Specification

ระดับชั้น: ปวส. เทคโนโลยีสารสนเทศ

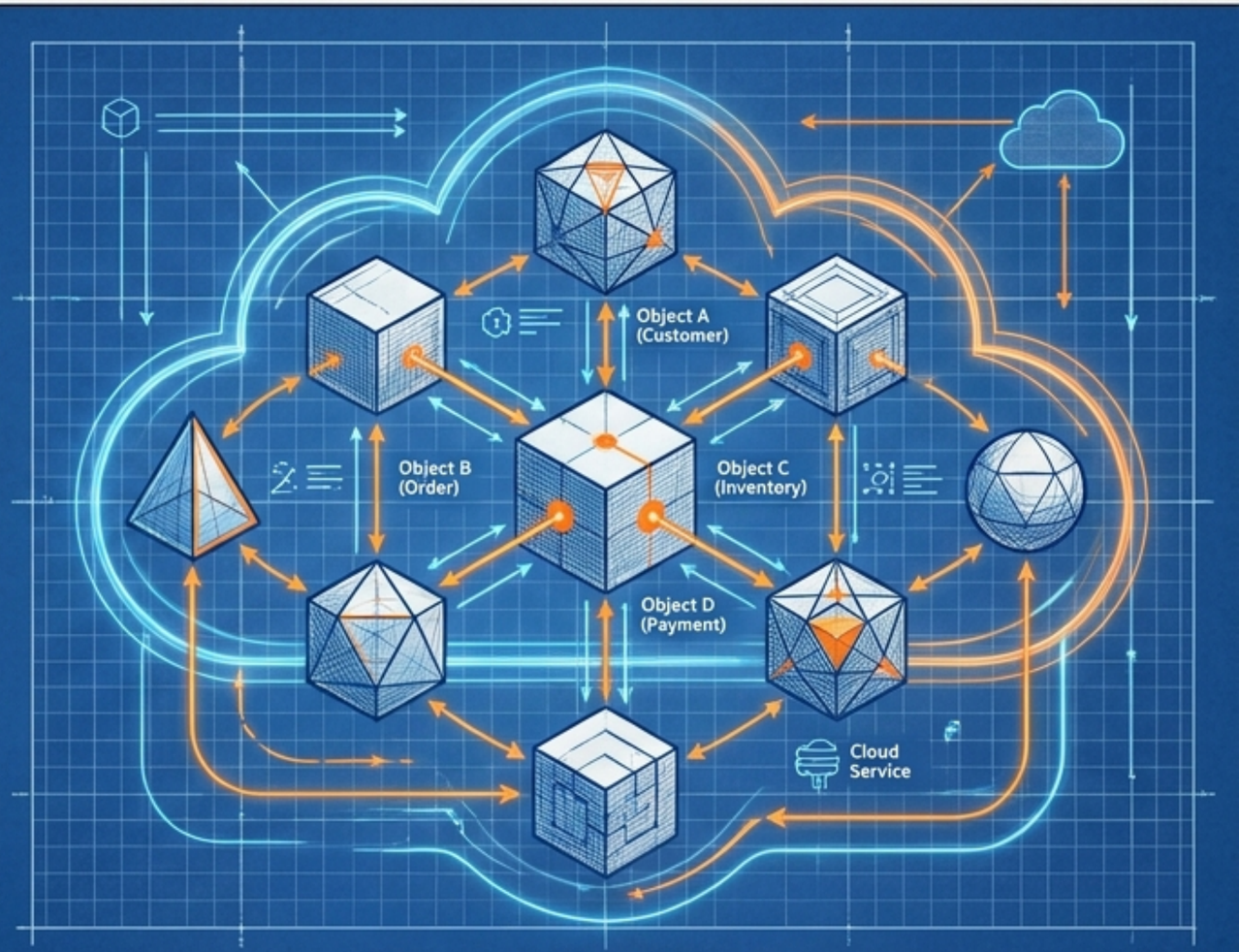
โครงสร้างเวลาเรียน:

ทฤษฎี 1 ชั่วโมง | ปฏิบัติ 4 ชั่วโมง | 3 หน่วยกิต

จากโปรแกรมเดี่ยว สู่ระบบบูรณาการบนคลาวด์



วิถีเดิม (Procedural):
การมองซอฟต์แวร์เป็นเพียงชุดคำสั่งแบบขั้นบันได (Step-by-step)
ซึ่งไม่เพียงพอต่อการจัดการระบบที่ซับซ้อนในเศรษฐกิจดิจิทัล



วิถีใหม่ (Object-Oriented):
การออกแบบระบบที่มองทุกสิ่งเป็น "วัตถุ" (Objects) ที่มีปฏิสัมพันธ์ต่อกัน
ช่วยสร้างสถาปัตยกรรมซอฟต์แวร์ที่ยืดหยุ่น ลดความซ้ำซ้อน
และรองรับการสเกลบนระบบ Cloud

5 เสาหลักของแนวคิดเชิงวัตถุ

1. Object & Class

คลาส (Class) คือแม่แบบพิมพ์เขียว
ส่วนวัตถุ (Object)
คือตัวตนที่ถูกสร้างขึ้นจริง
มีทั้งคุณลักษณะ (Attribute)
และพฤติกรรม (Method)

2. Inheritance (การสืบทอด)

รับคุณสมบัติจากคลาสแม่สู่คลาสลูก
ลดการเขียนโค้ดซ้ำซ้อน (Reusability)

3. Polymorphism (พหุสัณฐาน)

พฤติกรรมเดียวกัน แต่ตอบสนองต่าง
กันตามบริบทของวัตถุแต่ละประเภท

4. Encapsulation (การห่อหุ้ม)

ซ่อนรายละเอียดและปกป้อง
ข้อมูลภายใน ให้เข้าถึงได้เฉพาะ
ผ่าน Interface ที่กำหนด

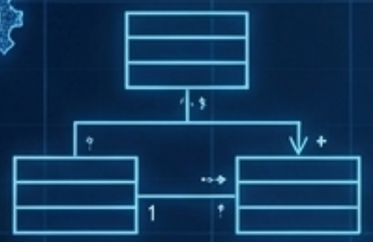
5. Abstraction (นามธรรม)

ตัดทอนรายละเอียดที่ไม่จำเป็น
คงไว้เฉพาะแก่นสารสำคัญใ
ในการแก้ปัญหา

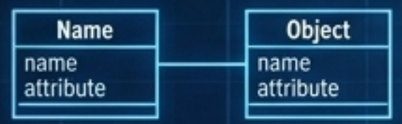
UML 2.0: ภาษาสากลของสถาปนิกซอฟต์แวร์

เครื่องมือสื่อสารความต้องการของระบบ แบ่งออกเป็น 2 มิติหลัก

มิติโครงสร้าง (Static View) - แสดงส่วนประกอบที่อยู่นิ่ง



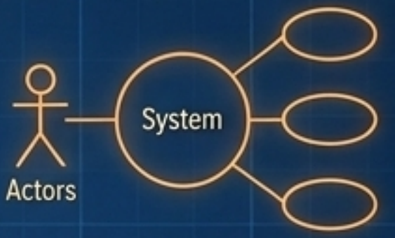
Class Diagram: หัวใจหลัก แสดงคลาส ความสัมพันธ์ และโครงสร้างข้อมูล



Object Diagram: จำลองสถานะของวัตถุ ณ ช่วงเวลาหนึ่ง

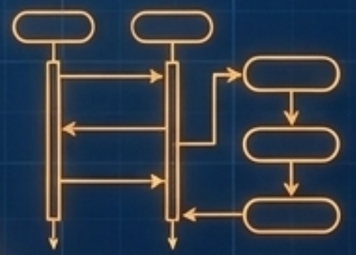
Deployment / Component Diagram: โครงสร้างทางกายภาพและการติดตั้งบนฮาร์ดแวร์

มิติพฤติกรรม (Dynamic View) - แสดงการทำงานเมื่อเวลาผ่านไป



Use Case Diagram: กำหนดขอบเขตระบบ ใครทำอะไรได้บ้าง

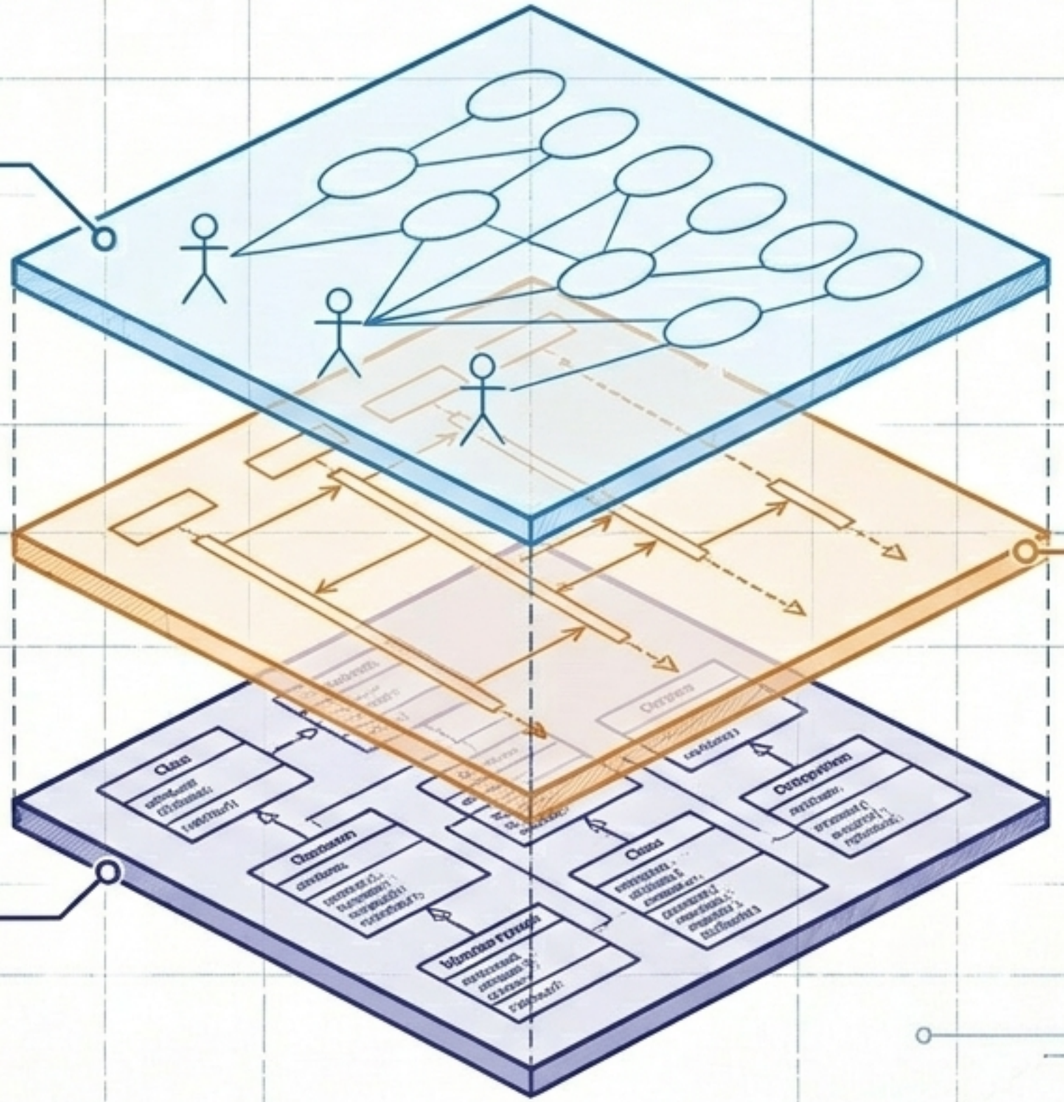
Activity Diagram: ลำดับขั้นตอนการทำงาน (Workflow) และเงื่อนไข



Sequence / State Machine: ลำดับเวลาในการส่งข้อความ และการเปลี่ยนสถานะของวัตถุ

สถาปัตยกรรมแบบจัดชั้น

ชั้นขอบเขตผู้ใช้งาน (User & Boundary):
ใช้ Use Case Diagram เพื่อตอบคำถามว่า "ระบบนี้สร้างมาเพื่อใคร และต้องทำอะไรได้บ้าง?"



ชั้นตรรกะและปฏิสัมพันธ์ (Business Logic & Interaction):
ใช้ Sequence Diagram และ Activity Diagram กำหนดลำดับการทำงานร่วมกันของระบบ

ชั้นโครงสร้างฐานราก (Structure & Data):
ใช้ Class Diagram แปลงเป็นโครงสร้างฐานข้อมูล (Database) และการเขียนโปรแกรมเชิงวัตถุ

เส้นทางสู่นักวิเคราะห์ระบบ: แผนการเรียนรู้ 15 สัปดาห์

การค้นหาและศึกษาความเป็นไปได้ (สัปดาห์ 1-3)

หน่วยที่ 1-3 วงจรการพัฒนาระบบ, การค้นหาปัญหา (Problem Recognition), วิทยฐาน, วิเคราะห์ความเป็นไปได้ 3 มิติ (เทคนิค, ปฏิบัติการ, เศรษฐศาสตร์)

การสร้างแบบจำลอง (สัปดาห์ 4-8)

หน่วยที่ 4-5 หลักการ UML และการเขียนโมเดลเชิงวัตถุ

ประเมินผล (สัปดาห์ 15)

หน่วยที่ 8 ตรวจสอบความสมบูรณ์และสอบปลายภาค

วิเคราะห์และออกแบบ (สัปดาห์ 9-13)

หน่วยที่ 6-7 วิเคราะห์ระบบงานเดิม/ใหม่ ออกแบบหน้าจอ (GUI), ฐานข้อมูล, และรายงาน

การวินิจฉัยโครงการ (Phase 1: Diagnostics)

1. การค้นหาปัญหา (Problem Recognition)

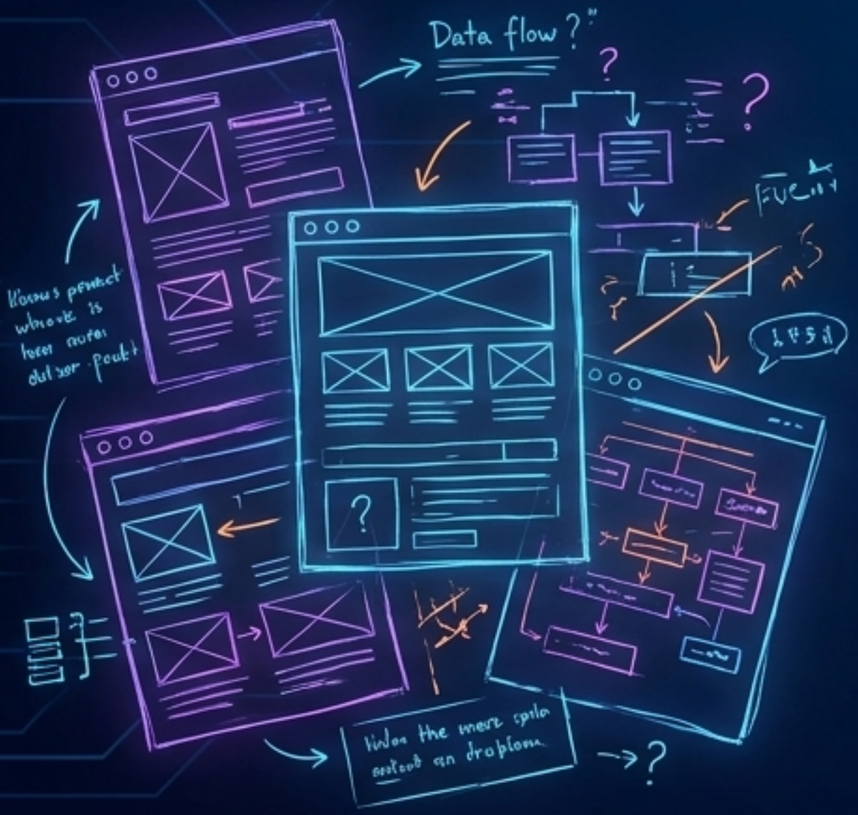
ใช้เครื่องมือวิเคราะห์รากเหง้าของปัญหา เช่น แผนภูมิก้างปลา เพื่อคัดเลือกโครงการที่ให้ประโยชน์สูงสุดแก่องค์กร

2. การศึกษาความเป็นไปได้ (Feasibility Study)

- Technical: เทคโนโลยีปัจจุบันรองรับหรือไม่?
- Operational: ผู้ใช้งานจะต่อต้านหรือยอมรับระบบใหม่?
- Economical: คุ้มค่าการลงทุนหรือไม่? (การวิเคราะห์กระแสเงินสด Cash Flow, ต้นทุน vs ผลตอบแทน)

จากทฤษฎีสู่พิมพ์เขียวระบบ (Phase 3: System Blueprinting)

ในขั้นตอนการออกแบบ นักวิเคราะห์ระบบต้องเปลี่ยนคำถามจาก "ระบบต้องทำอะไร (What)" เป็น "ระบบจะทำงานอย่างไร (How)"



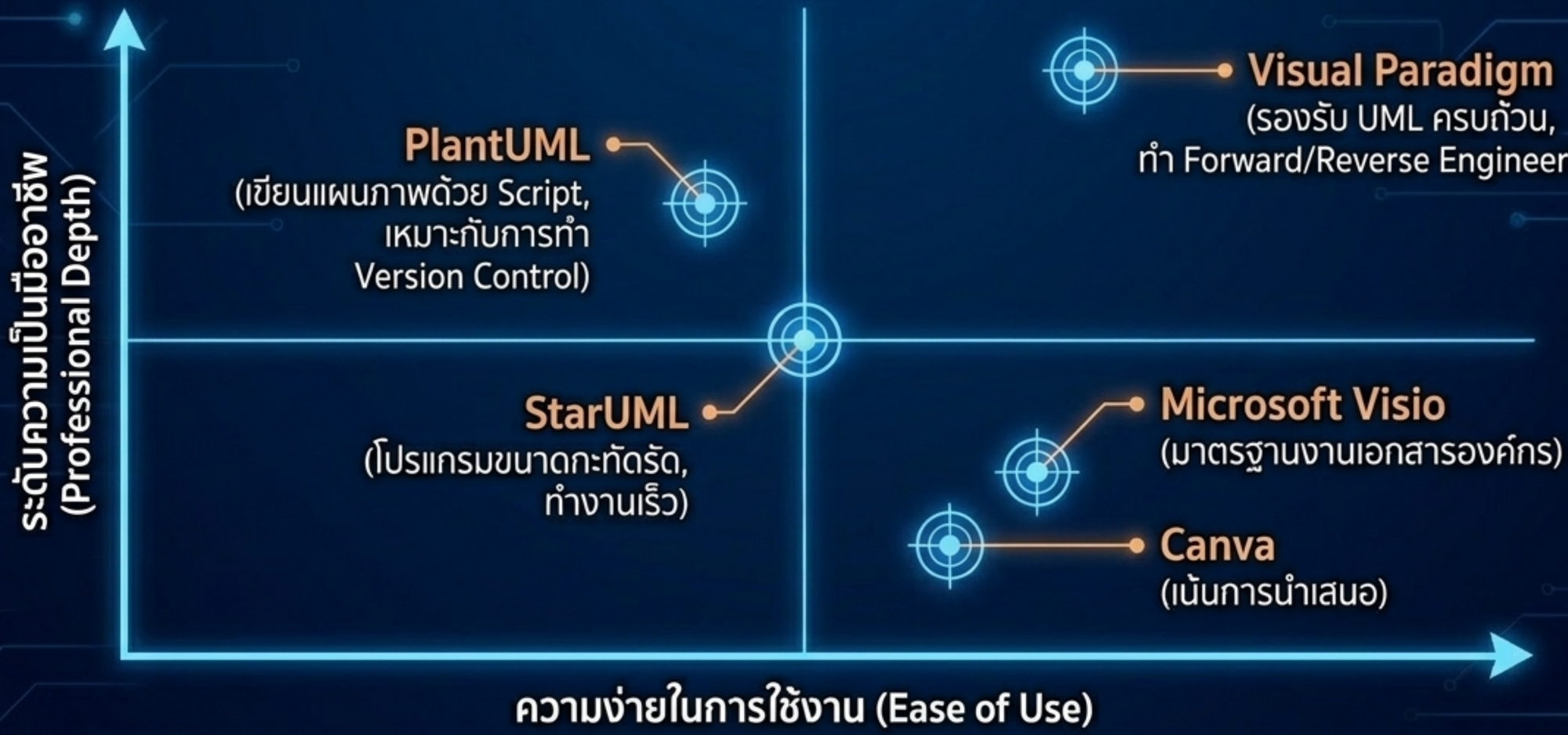
As-Is: Unstructured Theory & Concepts

ออกแบบฐานข้อมูล (Database Design):
แปลง Class Diagram เป็นตารางและพจนานุกรมข้อมูล (Data Dictionary)

ออกแบบส่วนติดต่อผู้ใช้ (GUI Design):
จัดทำ Screen Format ที่ใช้งานง่ายและป้องกันข้อผิดพลาด

ออกแบบรายงาน (Report Design):
ข้อมูลเฉพาะของการออกแบบ (System Design Specification) สำหรับโปรแกรมเมอร์

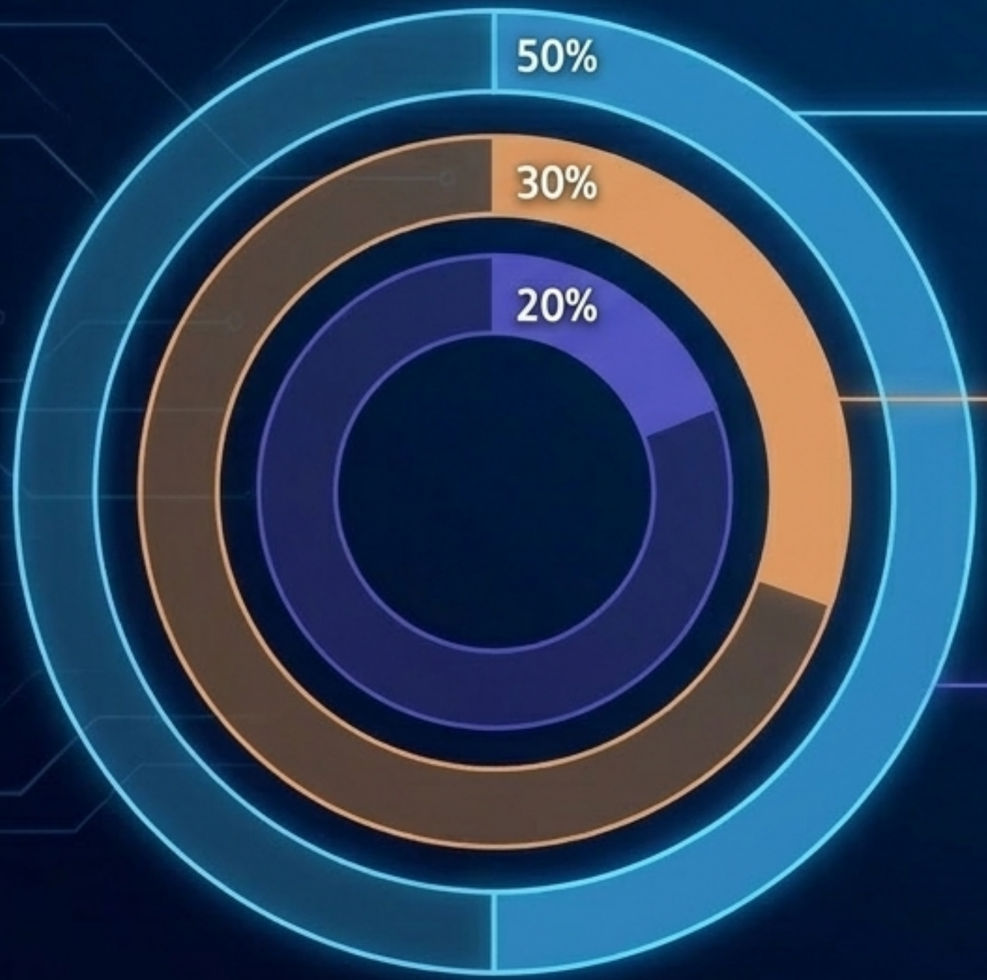
เครื่องมือของสถาปนิกซอฟต์แวร์



Computer-Aided Software Engineering (CASE) คือเครื่องมือสำคัญที่ช่วยลดระยะเวลา และรักษามาตรฐานของแผนภาพให้ตรงตามไวยากรณ์

เกณฑ์การวัดผลและประเมินสมรรถนะ

เน้นการลงมือปฏิบัติจริง (ทฤษฎี 1 : ปฏิบัติ 4) ประเมินตามสภาพจริง (Authentic Assessment)



ทักษะพิสัย (Psychomotor) 50%

50%

ทักษะการวาดแผนภาพ UML, การใช้ CASE Tools, การออกแบบฐานข้อมูลและ UI

พุทธิพิสัย (Cognitive) 30%

30%

ความเข้าใจหลักการ OOAD, SDLC, สอบปลายภาค (ปรนัย 60 ข้อ)

จิตพิสัย (Affective) 20%

20%

ความมีวินัย, ตรงต่อเวลา, ความรับผิดชอบ, และการทำงานเป็นทีม

ระดับคะแนนมาตรฐาน: 4.0 (80-100%) | 2.0 (60-64% เกณฑ์ผ่านขั้นต่ำ)

เป้าหมายสูงสุด: ความพร้อมสู่อุตสาหกรรม

รายวิชา 31901-2003 ไม่ได้สร้างเพียงผู้เขียนแผนภาพ
แต่สร้าง **"นักออกแบบสถาปัตยกรรมซอฟต์แวร์"** ที่ตอบโจทย์มาตรฐานอาชีพสากล

- **อ้างอิงมาตรฐานอาชีพ รหัส 10206:** อาชีพนักวิเคราะห์ออกแบบระบบ ระดับ 4
- **อ้างอิงมาตรฐานอาชีพ รหัส 11201-11203:** อาชีพนักออกแบบสถาปัตยกรรมซอฟต์แวร์ด้านเทคโนโลยีคลาวด์ ระดับ 4
- **ผลลัพธ์ :** ความสามารถในการประยุกต์ใช้ OOAD เพื่อพัฒนาโปรแกรมแบบ Integration System ได้อย่างมีประสิทธิภาพและการคิดเชิงนวัตกรรม