

Fundamental of **SOFTWARE ENGINEERING** in the Age of AI



ฉบับปรับปรุงครั้งที่ 2

เพิ่มเติม AI, Cyber Security
และ Digital Governance

**วิศวกรรมซอฟต์แวร์
เพื่อการเปลี่ยนผ่านสู่ดิจิทัล
ในยุคปัญญาประดิษฐ์**

จากการพัฒนาซอฟต์แวร์
สู่การออกแบบระบบ
ที่ขับเคลื่อนด้วยปัญญาประดิษฐ์

ดร.ปานใจ ธารทัศนวงศ์

CONTENT

CHAPTER

1

การปรับเปลี่ยนสู่ดิจิทัล

DIGITAL TRANSFORMATION 13

1.1 เศรษฐกิจดิจิทัล (Digital Economy)	14
1.2 ความหมายของเทคโนโลยีดิจิทัล (Digital Technology Definition)	16
1.3 องค์กรดิจิทัล (Digital Enterprises)	18
1.4 การปรับเปลี่ยนสู่ดิจิทัล (Digital Transformation)	19
1.5 บทบาทของวิศวกรรมซอฟต์แวร์กับการปรับเปลี่ยนสู่ดิจิทัล	27
1.6 การเปลี่ยนผ่านจากการพัฒนาระบบแบบดั้งเดิมสู่การพัฒนาระบบที่ขับเคลื่อนด้วยปัญญาประดิษฐ์	33
1.7 ค่านิยมและคำจำกัดความที่ใช้ในวิศวกรรมซอฟต์แวร์	38

CHAPTER

2

การวิเคราะห์เชิงโครงสร้าง

STRUCTURED ANALYSIS 45

2.1 การวิเคราะห์เชิงโครงสร้าง (Structure Analysis and Design)	46
2.2 แผนภาพการไหลของข้อมูล (Data flow Diagram)	47
2.3 ตัวอย่างการใช้แผนภาพการไหลของข้อมูลกับการปรับรีออกแบบการทางธุรกิจ (Using DFD in BPR)	52

CHAPTER

3

การวิเคราะห์และออกแบบเชิงวัตถุ

OBJECT-ORIENTED ANALYSIS AND DESIGN 57

3.1 กระบวนทัศน์เชิงวัตถุ (Object-Oriented Paradigm)	59
3.2 หลักการเชิงวัตถุ (Object-Oriented Concept)	60
3.3 นามธรรม (Abstraction)	62
3.4 การห่อหุ้ม (Encapsulation)	69
3.5 ความเป็นโมดูลาร์ (Modularity)	70
3.6 ลำดับชั้น (Hierarchy)	70
3.7 พอลิมอร์ฟิซึม (Polymorphism)	74
3.8 การวิเคราะห์และออกแบบเชิงวัตถุ (Object-Oriented Analysis and Design)	75

CHAPTER

4

UML ภาษาการสร้างแบบจำลอง
แบบครบวงจร

4.1 ภาษาการสร้างแบบจำลอง (Modeling Language)	81
4.2 ความเป็นมาของภาษาการสร้าง แบบจำลองแบบครบวงจร (Unified Modeling Language, UML)	82
4.3 แผนภาพยูเอ็มแอล (UML Diagram)	83
4.4 แผนภาพยูสเคส (Use Case Diagram)	91
4.5 แผนภาพคลาส (Class Diagram)	93
4.6 แผนภาพลำดับ (Sequence Diagram)	99
4.7 แผนภาพสถานะ (State Diagram)	111
4.8 แผนภาพกิจกรรม (Activity Diagram)	122
4.9 แผนภาพการนำไปใช้ (Deployment Diagram)	126
4.10 การสร้างรายงานโดยยูเอ็มแอล	131
4.11 การพัฒนาโปรแกรมโดยใช้แบบจำลอง	134
4.12 การพัฒนาด้วยวิศวกรรมย้อนกลับ (Reverse Engineering)	136
	139

CHAPTER

5

กระบวนการซอฟต์แวร์

SOFTWARE PROCESS	143
5.1 แบบจำลองกระบวนการซอฟต์แวร์ (Software Process Models)	146
5.2 กิจกรรมในกระบวนการซอฟต์แวร์ (Process Activities)	171

CHAPTER

6

การจัดการเทคโนโลยีดิจิทัล

DIGITAL TECHNOLOGY MANAGEMENT	183
6.1 ปัญหาในการบริหารจัดการองค์กร	185
6.2 สถาปัตยกรรมองค์กร (Enterprise Architecture)	188
6.3 สถาปัตยกรรมองค์กรกับวิศวกรรม ซอฟต์แวร์ (Enterprise Architecture and Software Engineering)	200
6.4 การกำกับดูแลการใช้เทคโนโลยีดิจิทัล	201
6.5 การจัดการซอฟต์แวร์ (Software Management)	206
6.6 การจัดการคุณภาพซอฟต์แวร์ ให้มีประสิทธิภาพ	226
6.7 ปัจจัยสู่ความสำเร็จ (Keys Success Factors)	230

CONTENT

CHAPTER

7

วิศวกรรมความต้องการ

REQUIREMENT ENGINEERING	235
7.1 ความหมายของความต้องการด้านซอฟต์แวร์ (Software Requirement Definition)	237
7.2 กระบวนการวิศวกรรมความต้องการ (Requirement Engineering Process)	240
7.3 กระบวนการจัดทำข้อกำหนดความต้องการ ด้านซอฟต์แวร์ (Software Requirements Specification Process, SRS)	245
7.4 การสกัดและวิเคราะห์ความต้องการ (Requirements Elicitation and Analysis)	246
7.5 การตรวจสอบความต้องการ (Requirements Validation)	286
7.6 การจัดการความต้องการ (Requirements Management)	286
7.7 วิศวกรรมความต้องการแบบอไจล์ (Agile Requirement Engineering)	288

CHAPTER

8

การออกแบบระบบ

SYSTEM DESIGN	295
8.1 กระบวนการออกแบบระบบ (System Design)	298
8.2 การออกแบบสถาปัตยกรรม (Architectural Design)	299
8.3 การออกแบบส่วนต่อประสาน (Interface Design)	317
8.4 การออกแบบส่วนประกอบ (Component Design)	323
8.5 การออกแบบฐานข้อมูล (Database Design)	326
8.6 รูปแบบการออกแบบ (Design Pattern)	333

CHAPTER

9

การติดตั้งระบบและการบำรุงรักษา

SYSTEM IMPLEMENTATION AND
MAINTENANCE

339

- 9.1 การพัฒนาและติดตั้งระบบ
(System Implementation) 340
- 9.2 การทดสอบซอฟต์แวร์สำหรับ
แบบจำลองน้ำตก (Water Fall Model
Software Testing) 354
- 9.3 การทดสอบซอฟต์แวร์สำหรับไจล์
(Agile Software Testing) 365
- 9.4 การติดตั้งระบบ
(System Installation) 367
- 9.5 การจัดทำเอกสาร (Documentation) 370
- 9.6 การฝึกอบรม (Training) 371
- 9.7 การบำรุงรักษาซอฟต์แวร์
(Software Maintenance) 371

CHAPTER

10

วิศวกรรมซอฟต์แวร์เชิงเซอร์วิส

SERVICE ORIENTED SOFTWARE
ENGINEERING

375

- 10.1 วิศวกรรมซอฟต์แวร์เชิงเซอร์วิส (Service
Oriented Software Engineering) 376
- 10.2 รูปแบบการพัฒนาซอฟต์แวร์
เชิงเซอร์วิส 381
- 10.3 เทคโนโลยีเว็บเซอร์วิส
(Web Services) 384
- 10.4 รูปแบบข้อมูลที่ใช้ในเว็บเซอร์วิส 397
- 10.5 สถาปัตยกรรมเชิงเซอร์วิส (Service
Oriented Architecture, SOA) 412
- 10.6 ไมโครเซอร์วิส (Microservices) 422
- 10.7 ซอฟต์แวร์คอนเทนเนอร์
(Software Container) 428

CONTENT

CHAPTER

11

การพัฒนาบุคลากรด้านดิจิทัล และปัญญาประดิษฐ์

HUMAN RESOURCE DEVELOPMENTS IN DIGITAL & ARTIFICIAL INTELLIGENCE

433

- 11.1 ทักษะด้านดิจิทัล และปัญญาประดิษฐ์ 434
- 11.2 สมรรถนะของบุคลากรด้านการพัฒนา
เทคโนโลยีดิจิทัล และปัญญาประดิษฐ์ 445
- 11.3 การพัฒนาบุคลากรด้านการพัฒนา
เทคโนโลยีดิจิทัล และปัญญาประดิษฐ์ 447
- 11.4 การพัฒนาทักษะความเข้าใจ
และการใช้ปัญญาประดิษฐ์ 455

CHAPTER

12

ปัญญาประดิษฐ์กับวิศวกรรมซอฟต์แวร์

ARTIFICIAL INTELLIGENCE AND SOFTWARE ENGINEERING

459

- 12.1 บริบทและแนวคิดพื้นฐาน 460
- 12.2 ปัญญาประดิษฐ์กับวิศวกรรมซอฟต์แวร์ 461
- 12.3 วิศวกรรมซอฟต์แวร์เพื่อพัฒนา
ปัญญาประดิษฐ์ 473

CHAPTER

13

การรักษาความมั่นคงปลอดภัยทางไซเบอร์ และปัญญาประดิษฐ์

CYBER SECURITY AND ARTIFICIAL
INTELLIGENCE SECURITY

483

13.1 หลักการรักษาความมั่นคงปลอดภัยไซเบอร์

(Cyber Security)

484

13.2 หลักการใช้ปัญญาประดิษฐ์อย่างมั่นคง

ปลอดภัย (AI Security)

485

13.3 กรอบมาตรฐานและแนวปฏิบัติด้าน

Cyber Security และ AI Security

493

13.4 การบูรณาการ Cyber Security และ

AI Security กับวิศวกรรมซอฟต์แวร์

499

CHAPTER

14

กฎหมายและจริยธรรมที่เกี่ยวข้อง

RELATED LAWS AND ETHICS 513

14.1 พรบ.คุ้มครองข้อมูลส่วนบุคคล 515

14.2 พรบ. การกระทำความผิด
เกี่ยวกับคอมพิวเตอร์ 531

14.3 พรบ. การรักษาความมั่นคง
ปลอดภัยไซเบอร์ 534

14.4 การดำเนินการตามกฎหมายสำคัญ 537

14.5 กฎหมายเกี่ยวกับทรัพย์สินทางปัญญา 543

14.6 กฎหมายเกี่ยวกับปัญญาประดิษฐ์ 545

14.7 กฎหมายเกี่ยวกับบริการคลาวด์ 546

14.8 จริยธรรมดิจิทัล 547

1

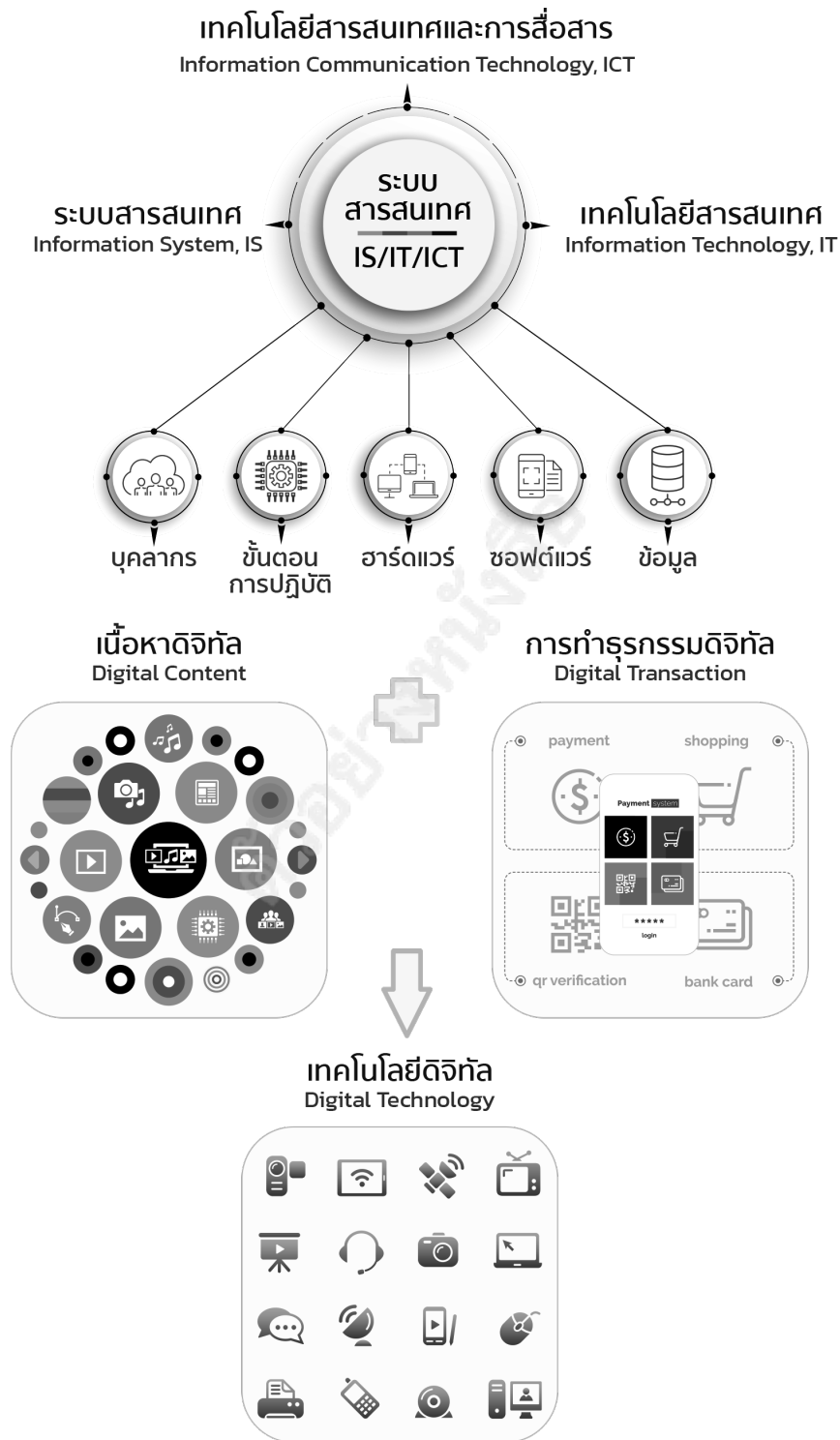
การปรับเปลี่ยนสู่ดิจิทัล DIGITAL TRANSFORMATION

วัตถุประสงค์ (Objectives)

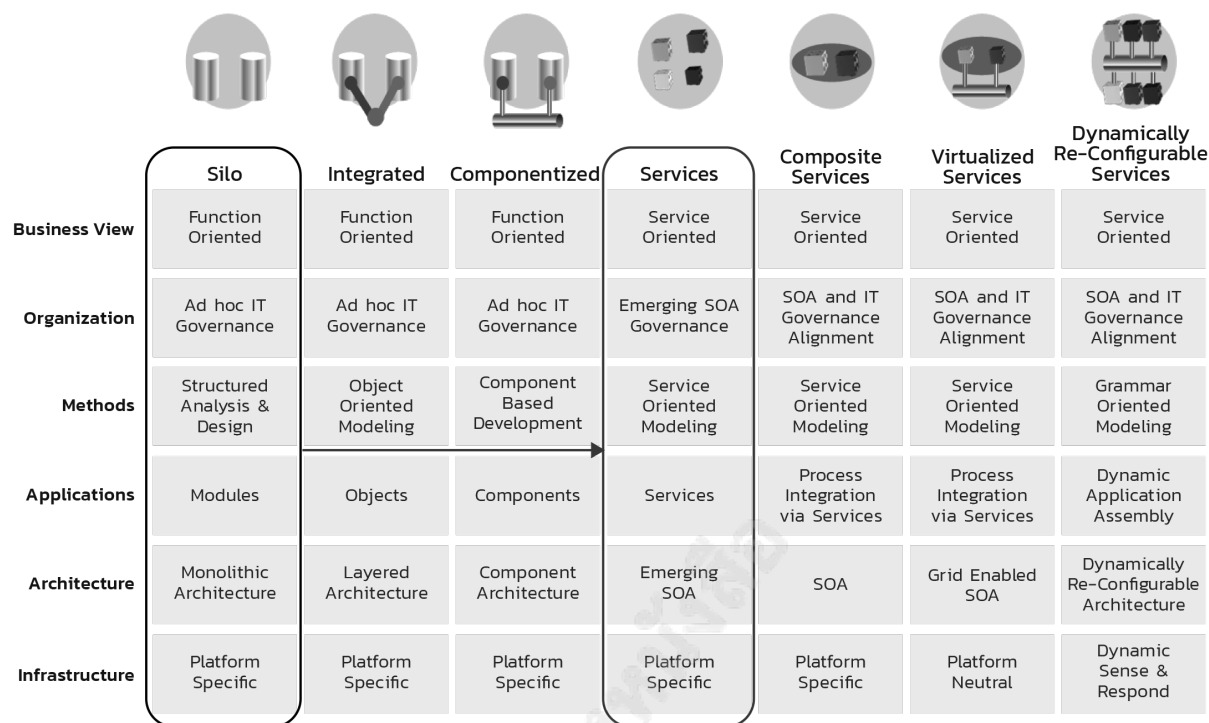
- เพื่ออธิบายรายละเอียดเกี่ยวกับเศรษฐกิจดิจิทัล (Digital Economy) องค์กรดิจิทัล (Digital Enterprise)
- เพื่ออธิบายแนวคิดในการปรับเปลี่ยนสู่ดิจิทัล (Digital Transformation)
- เพื่อแนะนำหลักการและความสำคัญของวิศวกรรมซอฟต์แวร์ (Software Engineering)
- เพื่ออธิบายการประยุกต์หลักการของวิศวกรรมซอฟต์แวร์กับการพัฒนาโปรแกรมประยุกต์เพื่อนำไปสู่การปรับเปลี่ยนสู่ดิจิทัล

เนื้อหาประจำบท (Contents)

- 1.1 เศรษฐกิจดิจิทัล (Digital Economy)
- 1.2 ความหมายของเทคโนโลยีดิจิทัล (Digital Technology Definition)
- 1.3 องค์กรดิจิทัล (Digital Enterprises)
- 1.4 การปรับเปลี่ยนสู่ดิจิทัล (Digital Transformation)
- 1.5 บทบาทของวิศวกรรมซอฟต์แวร์กับการปรับเปลี่ยนสู่ดิจิทัล
- 1.6 การเปลี่ยนผ่านจากการพัฒนาระบบแบบดั้งเดิมสู่การพัฒนาระบบที่ขับเคลื่อนด้วยปัญญาประดิษฐ์
- 1.7 คำนิยามและคำจำกัดความที่ใช้ในวิศวกรรมซอฟต์แวร์



รูปที่ 1.1 องค์ประกอบของเทคโนโลยีดิจิทัล



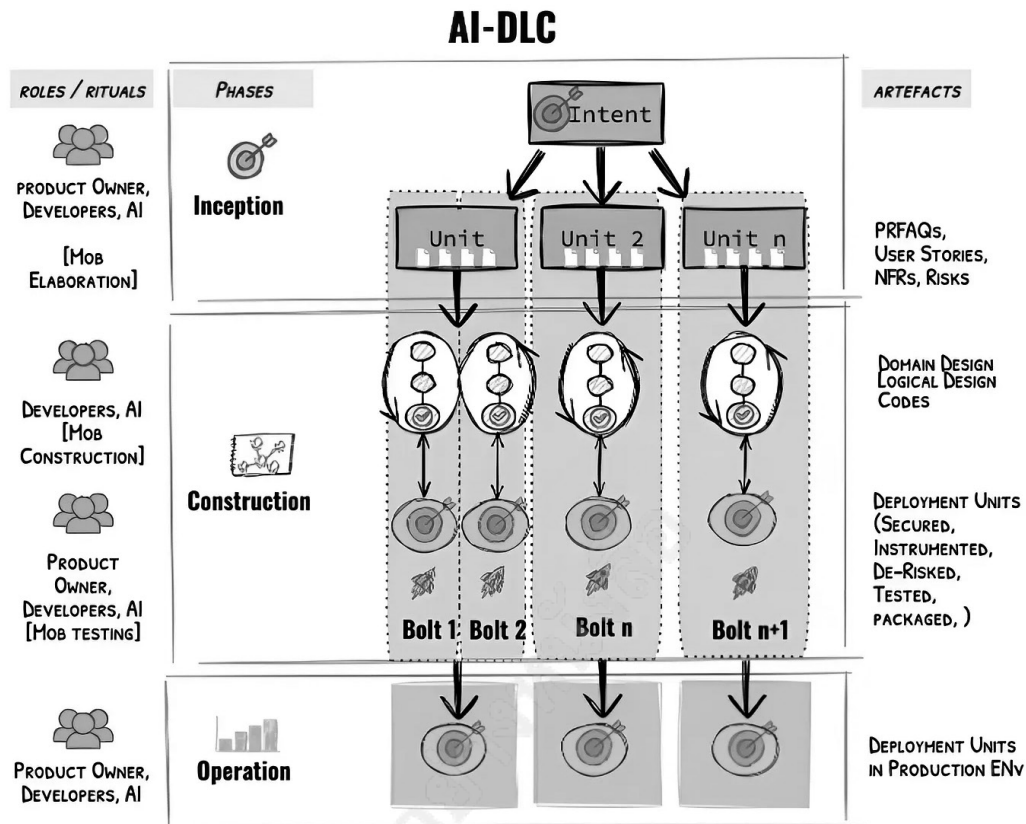
รูปที่ 1.6 การปรับเปลี่ยนสู่ดิจิทัลในบริบทเทคโนโลยีสารสนเทศจากแบบไซโลสู่แบบเซอร์วิส
(ที่มา : www.ibm.com/software/solution/soa)

4.4 ระบบนิเวศ (Ecosystem) องค์การที่ประสบความสำเร็จในการปรับเปลี่ยนสู่ดิจิทัลส่วนมากจะไม่ได้ทำงานตามลำพัง แต่อาจอาศัยพันธมิตรในระบบนิเวศดิจิทัล และมีการสร้างแพลตฟอร์มทางธุรกิจที่ให้หลาย ๆ ฝ่ายมีส่วนร่วมและได้รับประโยชน์ทุกฝ่าย

4.4.1 การจัดการทรัพย์สินดิจิทัล (Digital Asset Management) มีการใช้เครื่องมือในการบริหารทรัพย์สินทางปัญญา เช่น ข้อมูล ไฟล์รูปภาพ เนื้อหาดิจิทัล หรือไม

4.4.2 การจัดการสารสนเทศผลิตภัณฑ์ (Product Information Management) มีการจัดการข้อมูลผลิตภัณฑ์ไว้เป็นระบบ เพื่อนำเสนอลูกค้าได้อย่างเหมาะสมหรือไม่

4.4.3 การเพิ่มประสิทธิภาพการค้นหาข้อมูลในองค์กร มีการใช้เครื่องมือค้นหาที่ดีเพื่อให้สามารถเข้าถึงสารสนเทศต่าง ๆ ภายในหน่วยงานได้อย่างรวดเร็ว และระบบจัดการความรู้ (Knowledge Management) ของทั้งองค์กรหรือไม่



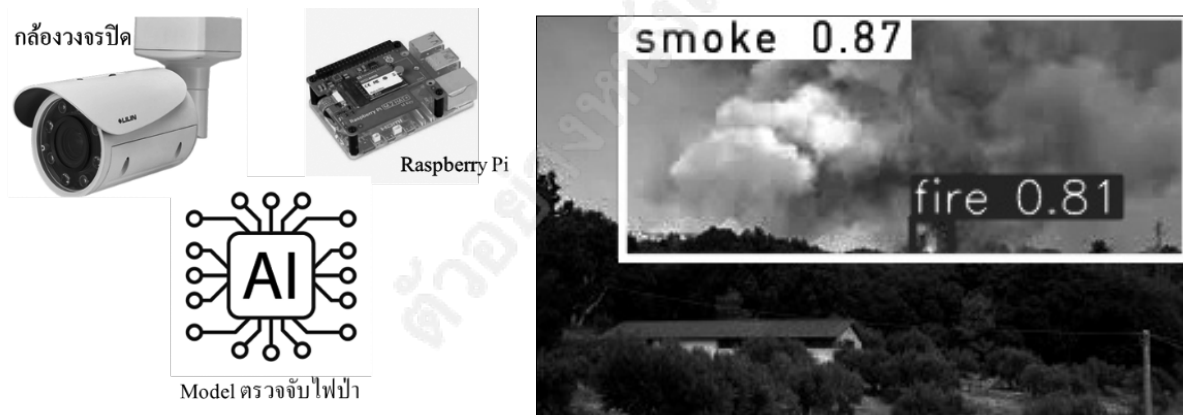
รูปที่ 1.7 การพัฒนาระบบที่ขับเคลื่อนด้วยปัญญาประดิษฐ์
(AI-Driven Development Lifecycle, AI-DLC) [9]

ในบริบทการปรับเปลี่ยนสู่ดิจิทัล (Digital Transformation) การใช้ AI-DLC ถือว่าเป็นการปรับระบบการพัฒนาเพื่อให้สอดคล้องกับสถานะการแข่งขันที่ต้องการการเปลี่ยนแปลงอย่างรวดเร็ว โดยเพิ่มความเร็วและประสิทธิภาพ ลดขั้นตอนและเอกสารที่ใช้ในการพัฒนาระบบ และลดเวลาในการส่งมอบ รวมถึงยกระดับคุณภาพของระบบ แต่ยังคงการควบคุมของมนุษย์ในประเด็นที่สำคัญ ปรับบทบาทบุคลากรสู่การตัดสินใจเชิงคุณค่า โดยให้ AI รับภาระงานทำซ้ำ และเพิ่มเวลาให้กับการแก้ปัญหาที่ซับซ้อน และการออกแบบเชิงกลยุทธ์ ซึ่งอธิบายไว้ในบทที่ 12

1.6.3 ปัญญาประดิษฐ์ในเอดจ์ (Edge Artificial Intelligence, Edge AI)

โดยทั่วไปปัญญาประดิษฐ์ (AI) ทำงานอยู่บนเครื่องคอมพิวเตอร์ประสิทธิภาพสูง แต่ปัจจุบันการพัฒนากำลังเปลี่ยนทิศทางไปสู่ปัญญาประดิษฐ์ในเอดจ์ (Edge AI) คือ การนำโมเดล AI ไปประมวลผลบนอุปกรณ์ปลายทางโดยตรง ไม่ว่าจะเป็นกล้องอัจฉริยะ เช่น เซอร์ IoT หุ่นยนต์ในโรงงาน หรือแม้แต่สมาร์ทโฟน

การย้ายการประมวลผลไปที่เอดจ์ของเครือข่ายมีข้อดีหลายประการ เช่น ลดเวลาการตอบสนอง (Latency) การลดการใช้แบนด์วิดท์เครือข่าย และการเพิ่มความเป็นส่วนตัวของข้อมูลเพราะไม่ต้องส่งข้อมูลขึ้น Cloud แต่สำหรับวิศวกรซอฟต์แวร์ การเปลี่ยนแปลงนี้นำมาซึ่งความท้าทายใหม่ ๆ ที่ซับซ้อนกว่าการเขียนโปรแกรมแบบเดิมหลายเท่าตัว และต้องผสมผสานองค์ความรู้ทั้งด้านวิทยาศาสตร์ข้อมูล (Data Science) ระบบฝังตัว (Embedded Systems) ระบบเครือข่าย (Networking) และความมั่นคงปลอดภัยไซเบอร์ (Cybersecurity) นี่คือการท้าทายสำคัญที่วิศวกรซอฟต์แวร์ต้องเผชิญกับการใช้เทคโนโลยีนี้



รูปที่ 1.8 ตัวอย่างการใช้กล้อง AI ในการตรวจจับไฟป่า [41]

จากรูปที่ 1.8 งานวิจัยการพัฒนาระบบตรวจจับและเฝ้าระวังไฟป่าแบบครบวงจร [4] เป็นการใช้ปัญญาประดิษฐ์ในเอดจ์ (Edge AI) โดยการใช้กล้องวงจรปิด (CCTV) ทั่วไปให้กลายเป็นระบบเฝ้าระวังอัจฉริยะที่สามารถตัดสินใจได้ด้วยตนเอง เริ่มต้นจากกล้องวงจรปิดเก็บข้อมูลภาพในพื้นที่ แล้วส่งต่อข้อมูลเหล่านั้นไปยัง Raspberry Pi ทำหน้าที่เป็นเอดจ์ (Edge) ด้วยการติดตั้งโมเดลการเรียนรู้ของเครื่อง (Machine Learning) ที่ใช้ในการประมวลผลข้อมูล ณ สถานที่ติดตั้ง เมื่อระบบตรวจพบความผิดปกติ เช่น กลุ่มควัน หรือไฟไหม้ AI จะทำการจำแนกวัตถุ (Object Detection) พร้อมตีล้อมกรอบระบุตำแหน่งอย่างชัดเจน การทำงานร่วมกันของอุปกรณ์เหล่านี้จึงถือเป็นแนวทางการป้องกันภัยเชิงรุกที่ช่วยให้เจ้าหน้าที่สามารถเข้าควบคุมไฟป่าได้ทัน่วงที ก่อนที่เหตุการณ์จะลุกลามเป็นความสูญเสียครั้งใหญ่

3

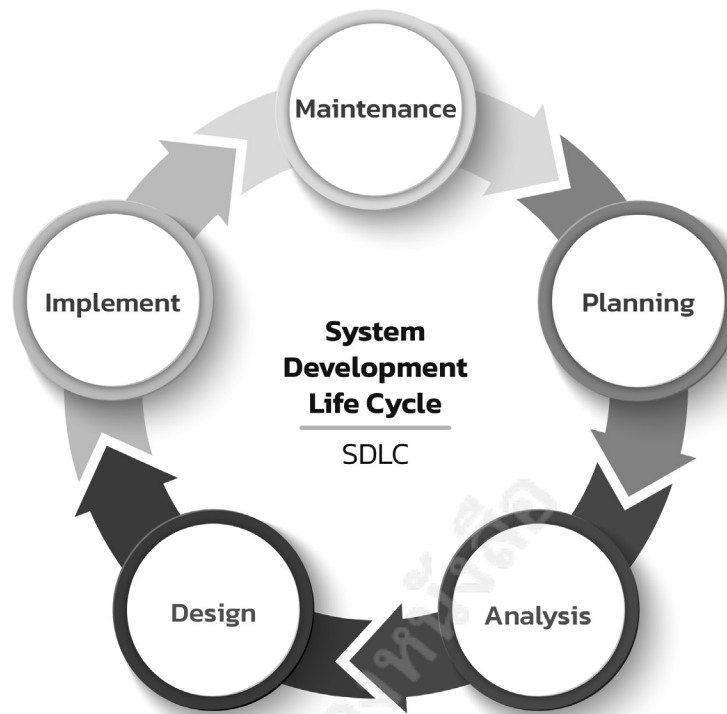
การวิเคราะห์และออกแบบเชิงวัตถุ OBJECT-ORIENTED ANALYSIS AND DESIGN

วัตถุประสงค์ (Objectives)

- เพื่ออธิบายกรอบความคิดเชิงวัตถุ (Object-Oriented Approach)
- เพื่ออธิบายการวิเคราะห์ระบบเชิงวัตถุ (Object-Oriented Analysis)
- เพื่ออธิบายการออกแบบเชิงวัตถุ (Object-Oriented Design)

เนื้อหาประจําบท (Contents)

- 3.1 กระบวนทัศน์เชิงวัตถุ (Object-Oriented Paradigm)
- 3.2 หลักการเชิงวัตถุ (Object-Oriented Concept)
- 3.3 นามธรรม (Abstraction)
- 3.4 การห่อหุ้ม (Encapsulation)
- 3.5 ความเป็นโมดูลาร์ (Modularity)
- 3.6 ลำดับชั้น (Hierarchy)
- 3.7 พอลิมอร์ฟิซึม (Polymorphism)
- 3.8 การวิเคราะห์และออกแบบเชิงวัตถุ (Object-Oriented Analysis and Design)



System Development Life Cycle (SDLC) Process

กลไกการปรับเปลี่ยนสู่ดิจิทัลมีขั้นตอนตามที่ได้กล่าวมาแล้วในบทที่ 1 และส่วนที่สำคัญที่สุดคือ การพัฒนาซอฟต์แวร์ให้สามารถใช้งานได้ตามความต้องการ ซึ่งส่วนใหญ่แล้วองค์กรมักจะมีปัญหาเรื่องความล้มเหลวของซอฟต์แวร์ที่จัดทำมาเนื่องจากไม่ตรงตามความต้องการหรือไม่มีประสิทธิภาพ การใช้หลักการของวิศวกรรมซอฟต์แวร์เป็นแนวทางที่สำคัญในการแก้ปัญหาดังกล่าว ในบทนี้จะอธิบายภาพรวมเกี่ยวกับลักษณะพื้นฐานของระบบเชิงวัตถุ และแนะนำการวิเคราะห์และออกแบบระบบเชิงวัตถุ

4

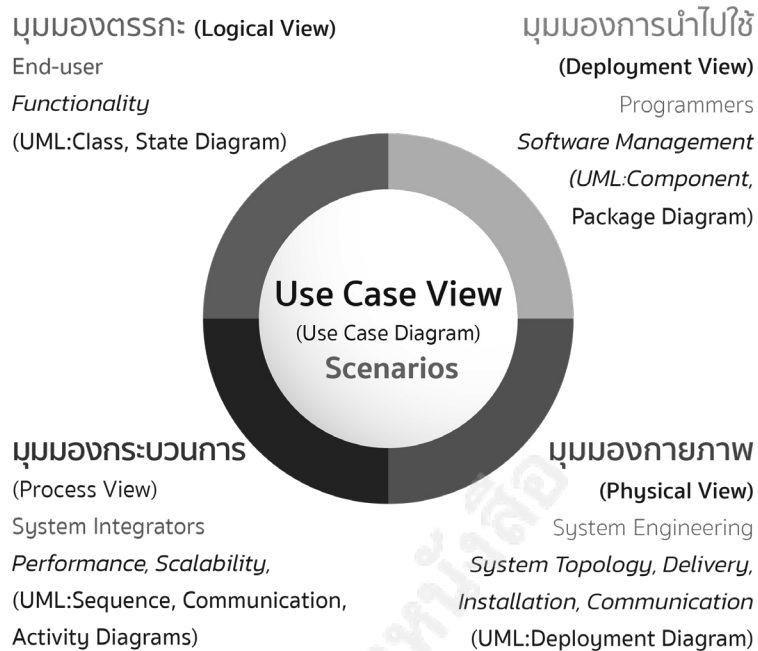
UML ภาษาการสร้างแบบจำลอง แบบครบวงจร

วัตถุประสงค์ (Objectives)

- เพื่ออธิบายภาษาการสร้างแบบจำลอง
- เพื่ออธิบายรายละเอียดของภาษาการสร้างแบบจำลองแบบครบวงจรแบบต่าง ๆ
- เพื่ออธิบายวิธีการทำแผนภาพแบบต่าง ๆ
- เพื่ออธิบายวิธีการพัฒนาระบบด้วยภาษาการสร้างแบบจำลอง

เนื้อหาประจำบท (Contents)

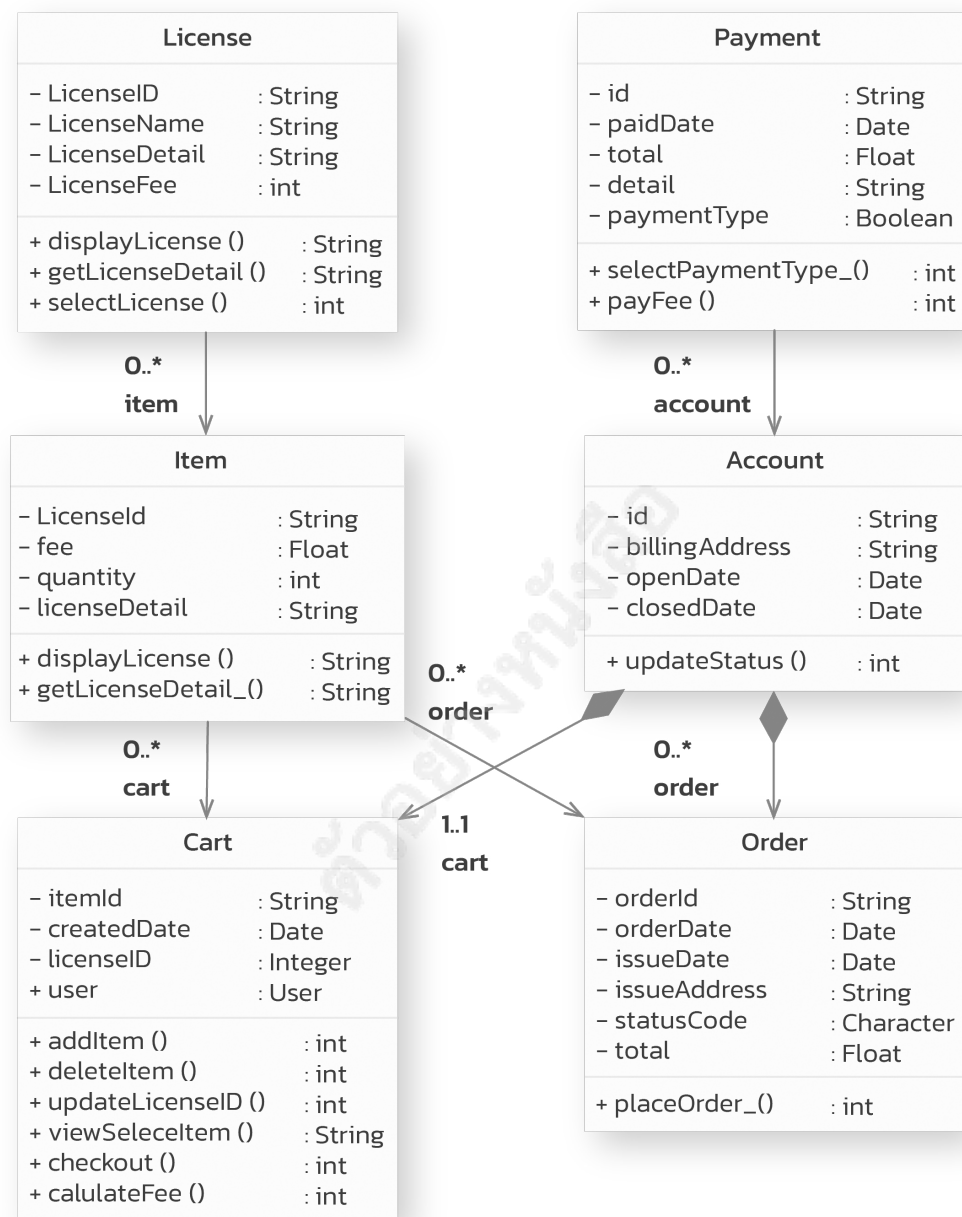
- 4.1 ภาษาการสร้างแบบจำลอง (Modeling Language)
- 4.2 ความเป็นมาของภาษาการสร้างแบบจำลองแบบครบวงจร (Unified Modeling Language, UML)
- 4.3 แผนภาพยูเอ็มแอล (UML Diagram)
- 4.4 แผนภาพยูสเคส (Use Case Diagram)
- 4.5 แผนภาพคลาส (Class Diagram)
- 4.6 แผนภาพลำดับ (Sequence Diagram)
- 4.7 แผนภาพสถานะ (State Diagram)
- 4.8 แผนภาพกิจกรรม (Activity Diagram)
- 4.9 แผนภาพการนำไปใช้ (Deployment Diagram)
- 4.10 การสร้างรายงานโดยยูเอ็มแอล
- 4.11 การพัฒนาโปรแกรมโดยใช้แบบจำลอง
- 4.12 การพัฒนาด้วยวิศวกรรมย้อนกลับ (Reverse Engineering)



รูปที่ 4.4 แผนภาพยูเอ็มแอลที่สัมพันธ์กับแบบจำลองมุมมองสถาปัตยกรรมแบบ 4 + 1 [19]

การพัฒนาระบบด้วยวิธีเชิงวัตถุนั้นจะใช้เครื่องมือที่เรียกว่า “ภาษาการสร้างแบบจำลองแบบครบวงจร” (Unified Modeling Language, UML) หรือยูเอ็มแอล เป็นภาษาเพื่อใช้อธิบายแบบจำลอง (Model) ต่าง ๆ โดยมีการใช้ภาษาหรือตัวแทนที่เป็นภาพสัญลักษณ์หรือกราฟิก (Graphics) และเป็นภาษามาตรฐานที่นิยมใช้ในการพัฒนาระบบ เปรียบเสมือนแบบพิมพ์เขียว (Blueprint) ให้กับระบบงาน ในการสร้างมุมมอง การกำหนดรายละเอียด สร้างระบบงาน และจัดทำเอกสารอ้างอิงให้กับงานที่ต้องการจะพัฒนา

ในการพัฒนาซอฟต์แวร์นั้น ปกติเราจะใช้ภาษาโปรแกรมคอมพิวเตอร์ในการพัฒนา เช่น C#, C++ หรือ JAVA ซึ่งก่อนที่โปรแกรมเมอร์จะเขียนโปรแกรมได้นั้น จำเป็นที่จะต้องวิเคราะห์ออกแบบระบบก่อน อันเป็นขั้นตอนที่มีความสำคัญมากที่สุดและใช้เวลามากที่สุด เมื่อได้แบบแปลนพิมพ์เขียวแล้ว จากนั้นจึงจะเริ่มเขียนโปรแกรมด้วยภาษาต่าง ๆ



รูปที่ 4.46 แผนภาพคลาสที่ได้จากการทำวิศวกรรมย้อนกลับ (Reverse Engineering)

5

กระบวนการซอฟต์แวร์ SOFTWARE PROCESS

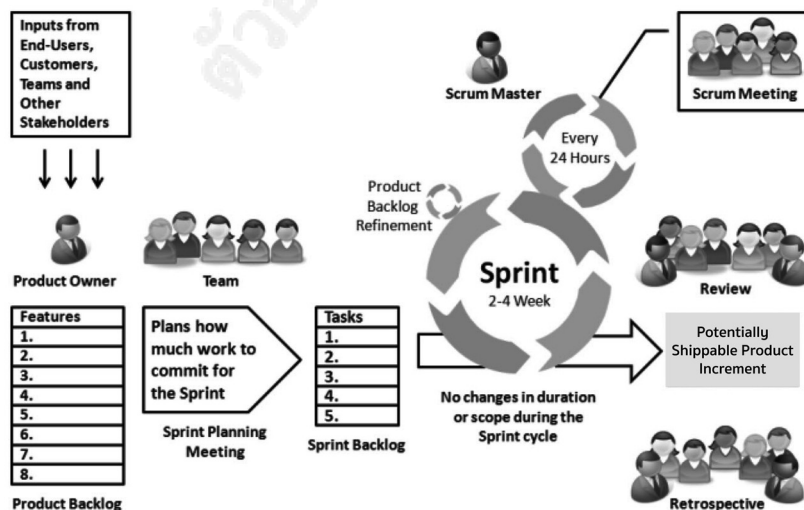
วัตถุประสงค์ (Objectives)

- เพื่อนำเสนอแนวคิดเกี่ยวกับกระบวนการซอฟต์แวร์
- เพื่อนำเสนอรายละเอียดเกี่ยวกับรูปแบบของซอฟต์แวร์แบบต่าง ๆ
- เพื่ออธิบายกิจกรรมพื้นฐานที่จำเป็นในการพัฒนาซอฟต์แวร์
- เพื่ออธิบายปัจจัยที่มีผลต่อคุณภาพของซอฟต์แวร์
- เพื่ออธิบายการปรับปรุงประสิทธิภาพของซอฟต์แวร์

เนื้อหาประจำบท (Contents)

- 5.1 แบบจำลองกระบวนการซอฟต์แวร์ (Software Process Models)
- 5.2 กิจกรรมในกระบวนการซอฟต์แวร์ (Process Activities)

- **คณะทำงาน (Team)** เป็นการพัฒนาลิทธิภัณฑ์ประกอบด้วยผู้เชี่ยวชาญหลายคนทำงานร่วมกัน เพื่อส่งมอบซอฟต์แวร์ที่เสร็จสมบูรณ์ โดยได้รับการจัดตั้งและมอบอำนาจจากองค์กรให้บริหารงานด้วยตนเอง ทั้งนี้ สมาชิกแต่ละคนมีความสามารถหลากหลายที่จำเป็นต่อการพัฒนาระบบ ขนาดของคณะทำงานนี้จะมีขนาดเล็กเพื่อให้เกิดความคล่องตัว และสามารถทำงานให้จบได้ภายในหนึ่งสปรินท์
- **สกรัมมาสเตอร์ (Scrum Master)** ซึ่งจะช่วยให้การประชุมเป็นไปอย่างราบรื่นและได้ผลลัพธ์ที่ดีออกมา รวมไปถึงคอยบอกให้ทุกคนรักษาเวลาในการประชุม และเข้าร่วมประชุมในฐานะสมาชิกของทีมพัฒนา โดยมีความรับผิดชอบดังนี้
 - การขจัดอุปสรรคหรือข้อขัดข้องต่าง ๆ ที่อาจจะเกิดขึ้น
 - การสร้างสภาพแวดล้อมที่มีประสิทธิภาพในการทำงานให้กับทีมพัฒนา
 - การจัดการกับพลวัตของทีมพัฒนา
 - สร้างความมั่นใจในความสัมพันธ์ที่ดีระหว่างทีมงาน และผู้เกี่ยวข้องทั้งภายในและภายนอกทีมพัฒนา
 - ปกป้องทีมพัฒนาจากการหยุดชะงักและการรบกวนภายนอก



รูปที่ 5.7 การประชุมระดมสมองในกระบวนการพัฒนาระบบแบบอไจล์

(ที่มา : <https://pkpmp.wordpress.com/2014/11/24/scrum-framework-at-a-glance/>)

6

การจัดการเทคโนโลยีดิจิทัล DIGITAL TECHNOLOGY MANAGEMENT

วัตถุประสงค์ (Objectives)

- เพื่ออธิบายปัญหาในการบริหารจัดการองค์กรและเทคโนโลยีดิจิทัล
- เพื่ออธิบายรายละเอียดสถาปัตยกรรมองค์กรในการปรับเปลี่ยนสู่ดิจิทัล
- เพื่ออธิบายความเชื่อมโยงของสถาปัตยกรรมองค์กรกับวิศวกรรมซอฟต์แวร์
- เพื่ออธิบายการบริหารจัดการเทคโนโลยีดิจิทัลให้มีประสิทธิภาพ
- เพื่ออธิบายการจัดการโครงการด้านซอฟต์แวร์
- เพื่ออธิบายความเสี่ยงและการจัดการความเสี่ยง
- เพื่ออธิบายการจัดการคุณภาพของซอฟต์แวร์

เนื้อหาประจำบท (Contents)

- 6.1 ปัญหาในการบริหารจัดการองค์กร
- 6.2 สถาปัตยกรรมองค์กร (Enterprise Architecture)
- 6.3 สถาปัตยกรรมองค์กรกับวิศวกรรมซอฟต์แวร์
(Enterprise Architecture and Software Engineering)
- 6.4 การกำกับดูแลการใช้เทคโนโลยีดิจิทัล (Digital Technology Governance)
- 6.5 การจัดการซอฟต์แวร์ (Software Management)
- 6.6 การจัดการคุณภาพซอฟต์แวร์ให้มีประสิทธิภาพ (Software Quality Management)
- 6.7 ปัจจัยสู่ความสำเร็จ (Keys Success Factors)

ปัญหาในการบริหารจัดการหน่วยงานหรือองค์กรต่าง ๆ มี 3 ลักษณะ กล่าวคือ

1. ปัญหาตามภาระหน้าที่หรือฟังก์ชันการทำงาน (Function Based) คือ ปัญหาที่เกี่ยวข้องกับภารกิจหลักของหน่วยงาน ซึ่งต้องการระบบสารสนเทศที่รองรับระดับปฏิบัติการ (Operation) และระดับสารสนเทศเพื่อการจัดการ (MIS)



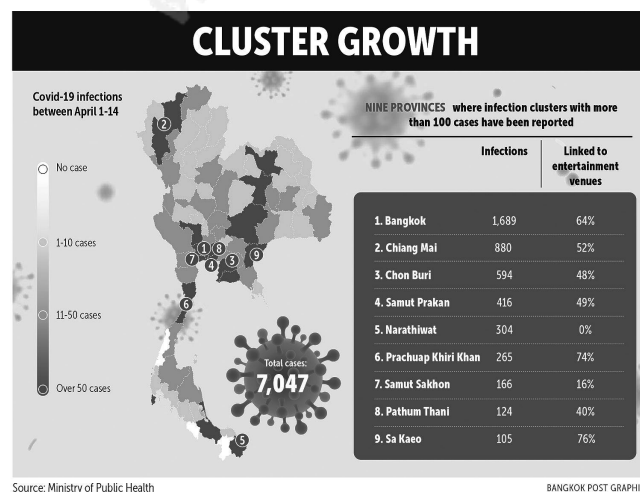
รูปที่ 6.2 ปัญหาเชิงฟังก์ชัน (Function Based Problems)
(ที่มา : สำนักงานประกันสังคม)

2. ปัญหาตามนโยบายหรือยุทธศาสตร์ (Agenda Based) เป็นปัญหาที่เกี่ยวข้องกับนโยบาย หรือยุทธศาสตร์ที่ปรับเปลี่ยนไปตามสถานการณ์ ระบบสารสนเทศที่รองรับปัญหานี้คือ ระบบสารสนเทศสำหรับผู้บริหาร (EIS) โดยใช้ระบบคลังข้อมูลหรือเหมืองข้อมูล (Data Warehouse/Data Mining)

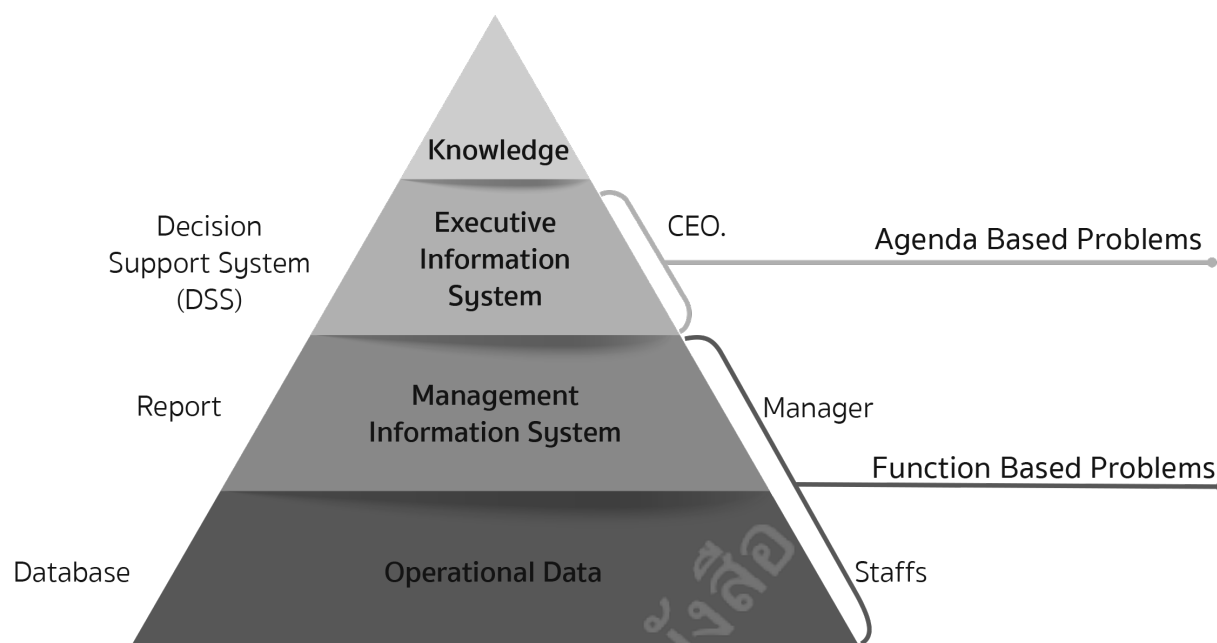


รูปที่ 6.3 ปัญหาเชิงนโยบาย (Agenda Based Problems) [1]

3. ปัญหาเชิงพื้นที่ (Area Based) เป็นปัญหาที่เกี่ยวข้องกับพื้นที่ มักจะเกิดกับหน่วยงานที่มีสาขาหลายแห่ง หรือมีการดำเนินงานที่เกี่ยวข้องกับพื้นที่ ระบบสารสนเทศที่ใช้กับระบบนี้คือ ระบบสารสนเทศภูมิศาสตร์ (GIS)



รูปที่ 6.4 ปัญหาเชิงพื้นที่ (Area Based Problems)
(ที่มา : www.bangkokpost.com)



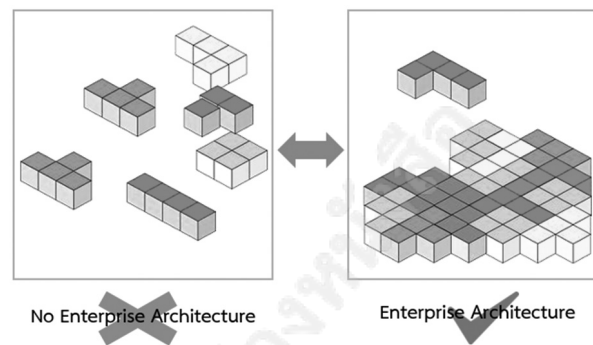
รูปที่ 6.5 ลำดับชั้นของระบบสารสนเทศกับปัญหาในการบริหารจัดการองค์กร

6.2 สถาปัตยกรรมองค์กร (Enterprise Architecture)

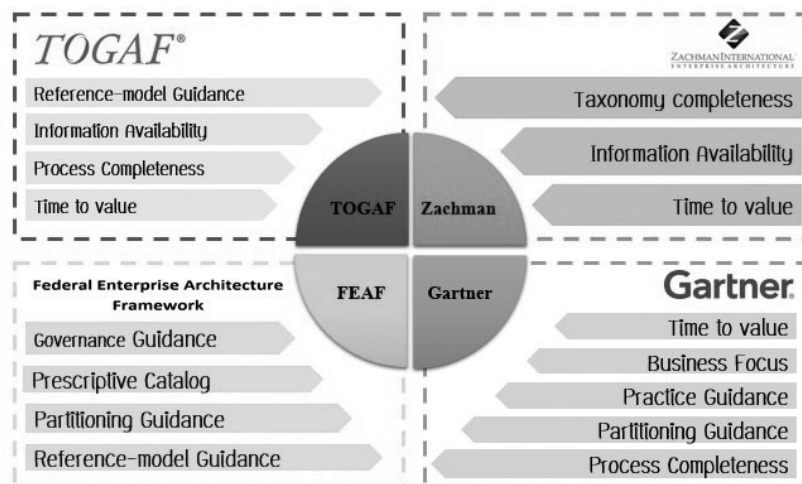
สถาปัตยกรรมองค์กร (Enterprise Architecture, EA) คือ กระบวนการนำเอาเทคโนโลยีสารสนเทศ (Information Technology) มาสนับสนุนการดำเนินงานธุรกิจ (Business) ให้เกิดประสิทธิภาพ (Efficiency) และประสิทธิผล (Effectiveness) สูงสุดต่อองค์กร สถาปัตยกรรมองค์กรเป็นเครื่องมือที่ช่วยให้เห็นภาพรวมของระบบต่าง ๆ ในองค์กร เพื่อกำหนดแนวทางการบูรณาการระบบเทคโนโลยีสารสนเทศ และการสื่อสาร เข้ากับการกิจขององค์กร เป็นระบบตั้งแต่การกำหนดโจทย์ภารกิจหรือธุรกิจ

เป้าหมายของสถาปัตยกรรมองค์กรคือ การสร้างสภาพแวดล้อมด้านเทคโนโลยีดิจิทัลที่เป็นหนึ่งเดียวกัน (ระบบฮาร์ดแวร์และซอฟต์แวร์ที่เป็นมาตรฐาน) ทั่วทั้งองค์กร ด้วยการเชื่อมโยงไปยังยุทธศาสตร์และกลยุทธ์ขององค์กร โดยเฉพาะอย่างยิ่งมีการกำหนดมาตรการในการนำทรัพยากรด้านดิจิทัลที่มีอยู่กลับมาใช้ใหม่ และการใช้ทรัพยากรได้อย่างคุ้มค่า ลดค่าใช้จ่าย และตอบสนองความต้องการได้ดีขึ้น

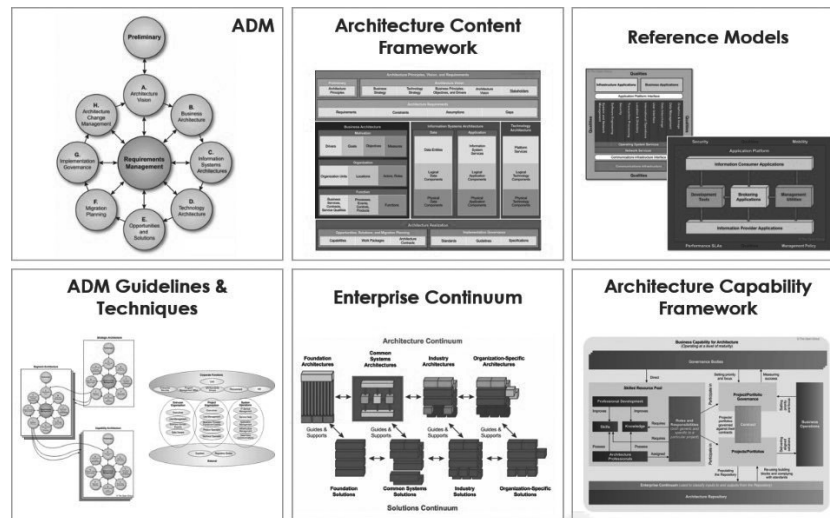
กรอบสถาปัตยกรรมองค์กร (Enterprise Architecture Framework) คือ แนวทางในการจัดทำและการนำเอาสถาปัตยกรรมองค์กรไปใช้งาน เป็นกรอบการวางแผนระดับยุทธศาสตร์ที่สัมพันธ์ และสอดคล้องกับแนวนโยบายด้านเทคโนโลยีดิจิทัล โดยมีวัตถุประสงค์เพื่อสร้างแผนที่ของทรัพยากรดิจิทัลที่ตอบสนองกระบวนการทางธุรกิจ และการกำกับดูแลที่ขับเคลื่อนอย่างต่อเนื่องเกี่ยวกับกลยุทธ์ทางธุรกิจและวิธีการดำเนินงานผ่านเทคโนโลยีดิจิทัล ซึ่งแสดงรายละเอียดได้เพียงพอ เพื่อก่อให้เกิดความสัมพันธ์กันระหว่างระบบเทคโนโลยีดิจิทัล ที่จะให้บริการกับผู้มีส่วนได้ส่วนเสียที่ได้ปรับให้สอดคล้องกับความต้องการพื้นฐานขององค์กรแล้ว นอกจากนี้ยังจะเป็นตัวชี้วัดถึงคุณภาพการพัฒนา ความคุ้มค่าในการใช้ทรัพยากรและการใช้จ่ายอย่างคุ้มค่า



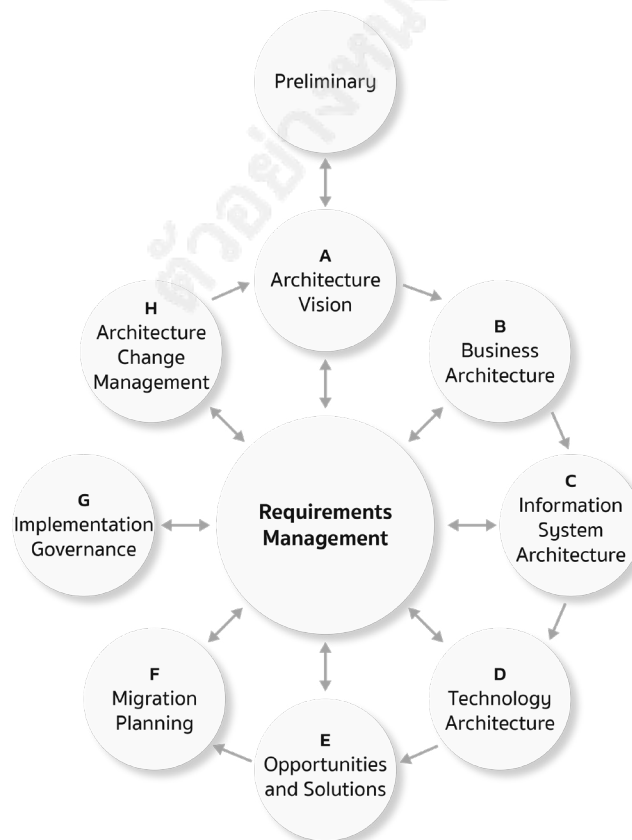
รูปที่ 6.6 การเปรียบเทียบองค์กรที่ใช้เทคโนโลยีดิจิทัลแบบต่างคนต่างทำและแบบที่มีสถาปัตยกรรมองค์กร
(ที่มา : <http://www.merlinssolutions.com/2016/04/11/enterprise-architecture-framework/>)



รูปที่ 6.7 กรอบการจัดทำสถาปัตยกรรมองค์กรที่นิยมใช้ [12]

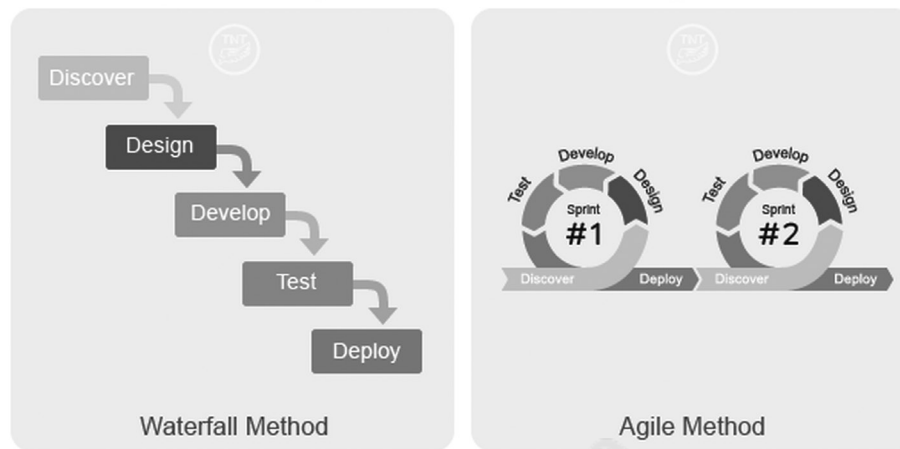


รูปที่ 6.9 แสดงโครงสร้างและส่วนประกอบของ TOGAF [5]



รูปที่ 6.10 วิธีการพัฒนาลำดับขั้น (Architecture Development Method) [5]

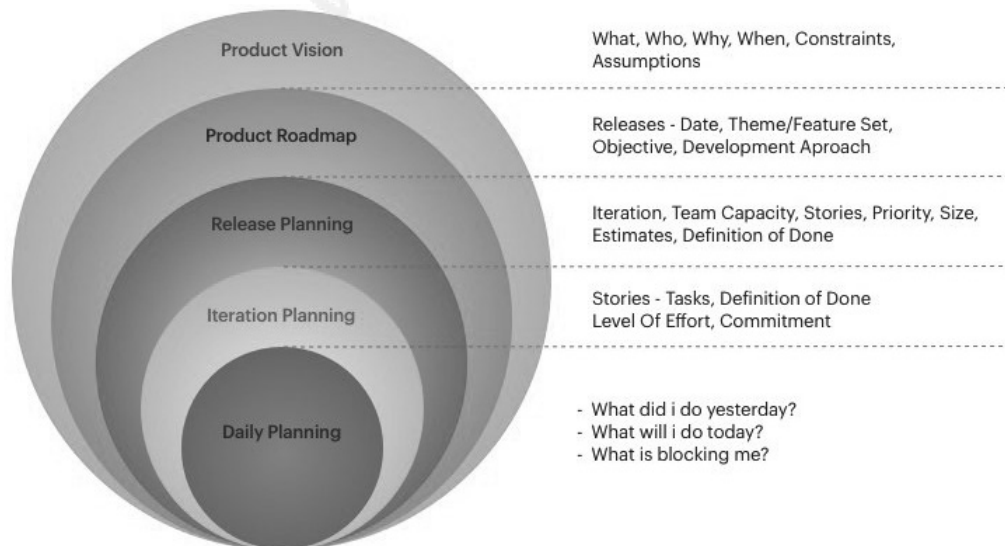
Development Methodologies



รูปที่ 6.29 เปรียบเทียบการจัดการโครงการแบบน้ำตกและแบบอไจล์ [20]

วิธีการพัฒนาซอฟต์แวร์แบบอไจล์ (Agile) เป็นวิธีการซ้ำ ๆ ที่ซอฟต์แวร์ได้รับการพัฒนาและส่งมอบให้กับลูกค้าทีละน้อย มีขั้นตอนในการวางแผน [11] ประกอบด้วย 5 ขั้นตอน ดังแสดงในรูปที่ 6.30

5 Levels of Agile Planning



รูปที่ 6.30 การวางแผนการพัฒนาซอฟต์แวร์แบบอไจล์ [21]

7

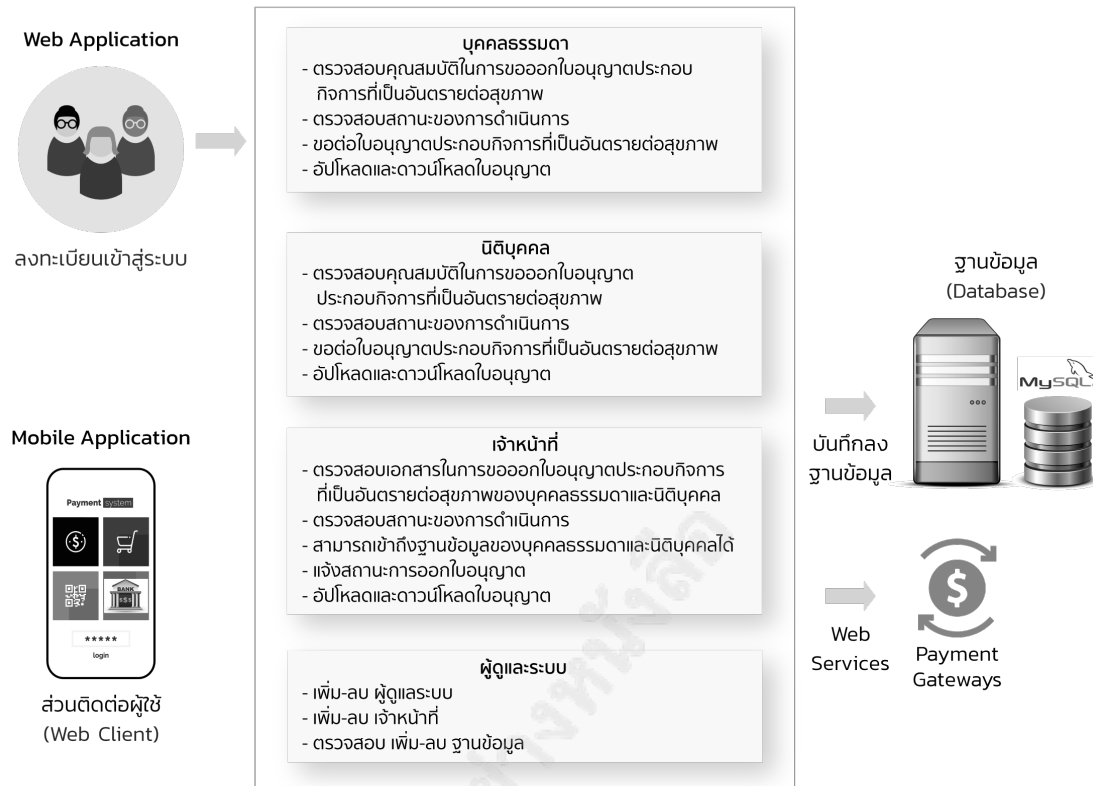
วิศวกรรมความต้องการ REQUIREMENT ENGINEERING

วัตถุประสงค์ (Objectives)

- เพื่อให้เข้าใจแนวคิดของความต้องการของระบบและการเขียนข้อกำหนดของซอฟต์แวร์
- เพื่อให้เข้าใจความแตกต่างระหว่างข้อกำหนดความต้องการซอฟต์แวร์ทั้งแบบฟังก์ชัน (Functional) และแบบไม่เป็นฟังก์ชัน (Nonfunctional)
- เพื่อให้เข้าใจกิจกรรมหลักของวิศวกรรมความต้องการ ซึ่งประกอบด้วยการตอบสนองความต้องการ (Elicitation) การวิเคราะห์และการตรวจสอบ รวมถึงความสัมพันธ์ระหว่างกิจกรรม
- เพื่อให้เข้าใจถึงวิธีการจัดทำเอกสารข้อกำหนดความต้องการด้านซอฟต์แวร์
- เพื่อให้เข้าใจถึงความจำเป็นในการจัดการความต้องการ
- เพื่อให้เข้าใจวิศวกรรมความต้องการทั้งแบบดั้งเดิมและแบบบอโลล์

เนื้อหาประจำบท (Contents)

- 7.1 ความหมายของความต้องการด้านซอฟต์แวร์ (Software Requirement Definition)
- 7.2 กระบวนการวิศวกรรมความต้องการ (Requirement Engineering Process)
- 7.3 กระบวนการจัดทำข้อกำหนดความต้องการด้านซอฟต์แวร์ (Software Requirements Specification Process, SRS)
- 7.4 การสกัดและวิเคราะห์ความต้องการ (Requirements Elicitation and Analysis)
- 7.5 การตรวจสอบความต้องการ (Requirements Validation)
- 7.6 การจัดการความต้องการ (Requirements Management)
- 7.7 วิศวกรรมความต้องการแบบบอโลล์ (Agile Requirement Engineering)



2.2 ฟังก์ชันของระบบ (Product Functions)

<สรุปฟังก์ชันหลักที่ซอฟต์แวร์ต้องทำ เช่น แผนภาพกระแสข้อมูลระดับบนสุดหรือแผนภาพคลาसออบเจกต์>

ระบบนี้เป็นการเก็บรวบรวมข้อมูลของผู้ขออนุญาตประกอบกิจการที่เป็นอันตรายต่อสุขภาพ โดยจะเก็บรวบรวมฐานข้อมูลจากการกรอกข้อมูลในเว็บแอปพลิเคชัน หลังจากนั้นข้อมูลที่ถูกกรอกไปจะถูกส่งไปฐานข้อมูลเพื่อเก็บรวบรวมและประมวลผลต่อไป ระบบนี้แบ่งผู้ใช้งานระบบออกเป็น 4 ส่วน คือ

1. ผู้ขอรับใบอนุญาต (User) แบ่งเป็นบุคคลธรรมดาและนิติบุคคล สามารถลงทะเบียนและกรอกข้อมูลที่ใช้ในการขออนุญาตประกอบกิจการที่เป็นอันตรายต่อสุขภาพ Upload เอกสารที่ต้องใช้ประกอบการขออนุญาต สามารถตรวจสอบสถานะการออกใบอนุญาตได้ ในกรณีที่ใช้อัปพลิเคชันบนโทรศัพท์เคลื่อนที่ที่สามารถชำระค่าธรรมเนียมผ่านทางโมบายแบงก์กิ้งได้

8

การออกแบบระบบ SYSTEM DESIGN

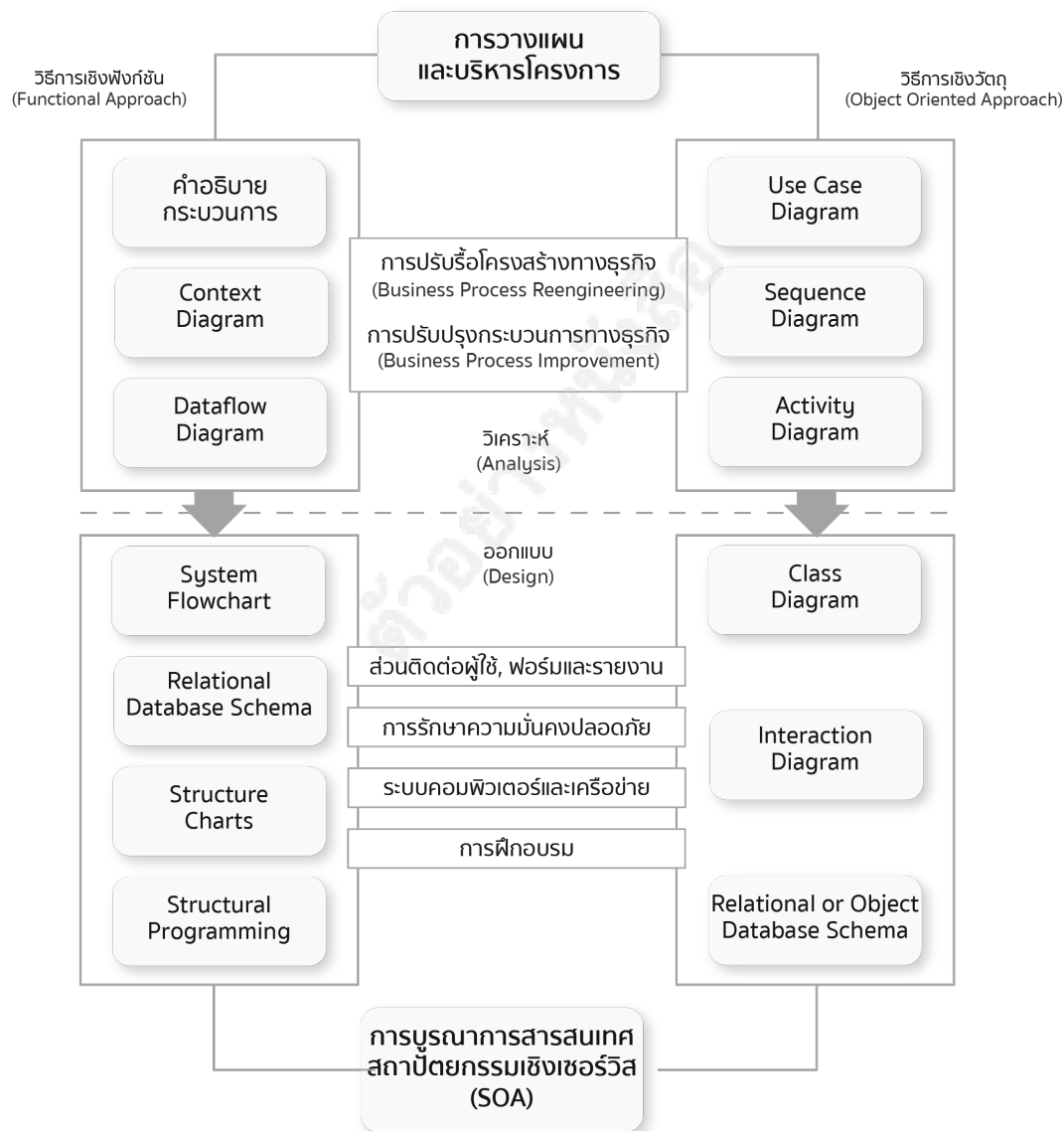
วัตถุประสงค์ (Objectives)

- เพื่ออธิบายกระบวนการออกแบบระบบ
- เพื่ออธิบายกระบวนการออกแบบสถาปัตยกรรม
- เพื่ออธิบายกระบวนการออกแบบส่วนเชื่อมต่อ
- เพื่ออธิบายกระบวนการออกแบบส่วนประกอบ
- เพื่ออธิบายกระบวนการออกแบบฐานข้อมูล
- เพื่ออธิบายรูปแบบการออกแบบ

เนื้อหาประจําบท (Contents)

- 8.1 กระบวนการออกแบบระบบ (System Design)
- 8.2 การออกแบบสถาปัตยกรรม (Architectural Design)
- 8.3 การออกแบบส่วนต่อประสาน (Interface Design)
- 8.4 การออกแบบส่วนประกอบ (Component Design)
- 8.5 การออกแบบฐานข้อมูล (Database Design)
- 8.6 รูปแบบการออกแบบ (Design Pattern)

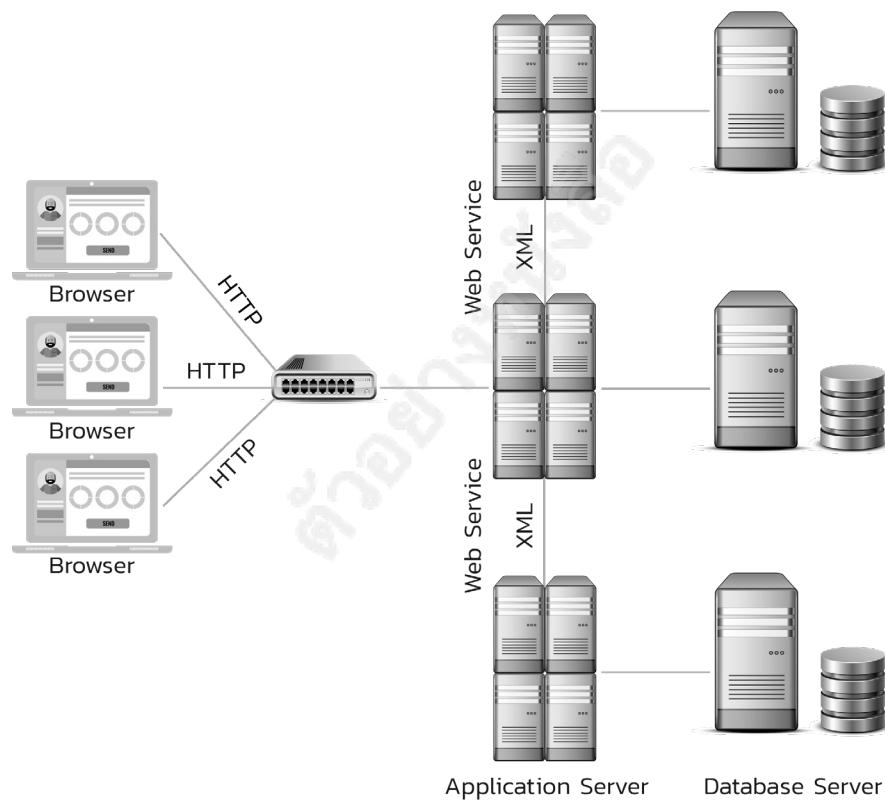
การพัฒนาระบบที่ดีไม่ควรจะสลับกระบวนการกันระหว่างเชิงฟังก์ชันและเชิงวัตถุ เช่น ออกแบบด้วยวิธีการเชิงฟังก์ชัน แต่พัฒนาโปรแกรมด้วยภาษาเชิงวัตถุ ซึ่งจะทำให้การแก้ไขและพัฒนาระบบมีปัญหาตามมาภายหลัง เนื่องจากแนวคิดและรายละเอียดของทั้งสองวิธีการนี้แตกต่างกันโดยสิ้นเชิง นอกจากนี้หากต้องการปรับหรือกระบวนการทางธุรกิจ (BPR) หรือปรับปรุงกระบวนการทางธุรกิจ (BPI) ก็ควรจะต้องจัดทำให้เสร็จสิ้นก่อน จากนั้นจึงใช้กระบวนการใหม่มาวิเคราะห์และออกแบบระบบต่อไป [4]



รูปที่ 8.1 ภาพรวมของการออกแบบระบบสารสนเทศ
ด้วยวิธีการเชิงฟังก์ชัน วิธีการเชิงวัตถุ และสถาปัตยกรรมเชิงเซอร์วิส [4]

3. สถาปัตยกรรมรับ-ให้บริการแบบ n-Tiers หรือระบบเว็บเซอร์วิส (Web Service)

จากสถาปัตยกรรมแบบ 3-Tiers หรือเว็บแอปพลิเคชัน (Web Application) เมื่อมีข้อมูลมากขึ้นและซับซ้อนขึ้น ข้อมูลมีการกระจายไปตามที่ต่าง ๆ จึงทำให้เกิดระบบการกระจายข้อมูล (Distributed System) ซึ่งเป็นการเชื่อมโยง Web Application หลาย ๆ ระบบเข้าด้วยกัน ตั้งแต่ 2 ระบบขึ้นไปจนถึง N (N คือ จำนวนระบบ Web Application ที่มาเชื่อมโยงข้อมูลกัน) การเชื่อมโยงข้อมูลระหว่างกันนั้นใช้เทคโนโลยีเว็บเซอร์วิส (Web Service) ดังแสดงในรูปที่ 8.10 [4]

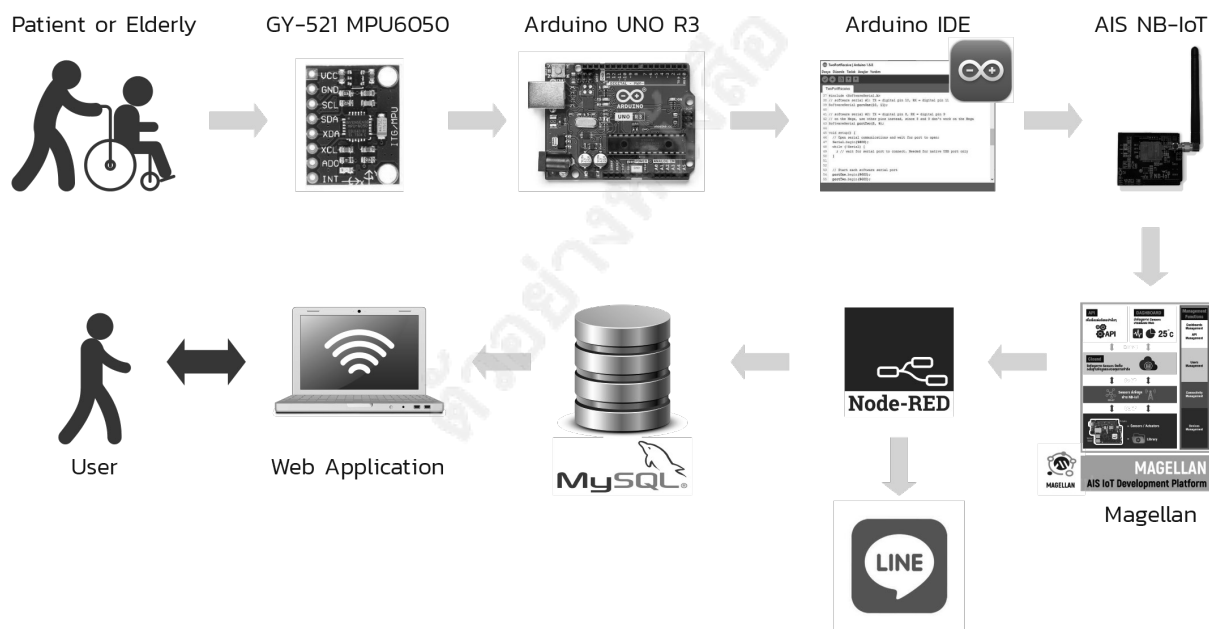


รูปที่ 8.10 สถาปัตยกรรมรับ-ให้บริการ (Client-Server) แบบ n-Tiers [4]

แนวคิดของเว็บบริการคือ เว็บที่สามารถทำงาน หรือให้บริการบางอย่างจากการร้องขอจากเซิร์ฟเวอร์ (Server) ต่าง ๆ ที่บูรณาการกันด้วยเทคโนโลยีเว็บบริการที่เอื้อต่อแนวคิดการประมวลผลแบบกระจาย และการที่เว็บบริการมี UDDI (Universal Description, Discovery and Integration) ทำให้สามารถค้นหาบริการต่าง ๆ ที่ต้องการได้

8.2.3.4 สถาปัตยกรรมแบบท่อและการกรอง (Pipe and Filter Architecture)

รูปแบบนี้ถูกนำมาใช้ตั้งแต่คอมพิวเตอร์ถูกใช้เป็นครั้งแรกเพื่อใช้การประมวลผลข้อมูล เมื่อการแปลงเป็นลำดับกับข้อมูลที่ประมวลผลในรูปแบบแบทช์ [15] สถาปัตยกรรมแบบท่อและการกรองนี้จะกลายเป็นวิธีการทำงานแบบเรียงลำดับ (Sequential) แบบกลุ่ม (Batch) การประมวลผลโดยมีการรวบรวมข้อมูลไว้ช่วงเวลาหนึ่งก่อนทำการประมวลผล การประมวลผลจะทำตามเวลาที่กำหนด ซึ่งอาจทำทุกวันหรือทุกสิ้นเดือน ผู้ใช้ไม่สามารถเห็นผลลัพธ์ทันทีและไม่สามารถโต้ตอบกับระบบได้ สถาปัตยกรรมแบบนี้จะใช้กับการทำงานของอินเทอร์เน็ตประสาสนเทศสรรพสิ่ง (Internet of Things, IoT) โดยมีการเก็บข้อมูลไว้ที่อุปกรณ์เซนเซอร์แล้วมีการส่งข้อมูลไปประมวลผลในเซิร์ฟเวอร์ [2] ดังแสดงในรูปที่ 8.11



รูปที่ 8.11 ตัวอย่างการใช้สถาปัตยกรรมแบบท่อและการกรองกับระบบตรวจจัดการหมอล้มของผู้ป่วย หรือผู้สูงอายุด้วยอินเทอร์เน็ตประสาสนเทศสรรพสิ่ง [2]

9

การติดตั้งระบบและการบำรุงรักษา SYSTEM IMPLEMENTATION AND MAINTENANCE

วัตถุประสงค์ (Objectives)

- เพื่ออธิบายรายละเอียดของการพัฒนาและติดตั้งระบบ
- เพื่ออธิบายวิธีการทดสอบซอฟต์แวร์แบบต่าง ๆ
- เพื่ออธิบายวิธีการติดตั้งระบบ
- เพื่ออธิบายการจัดทำเอกสาร
- เพื่ออธิบายการฝึกอบรม
- เพื่ออธิบายประเภทการบำรุงรักษาซอฟต์แวร์

เนื้อหาประจําบท (Contents)

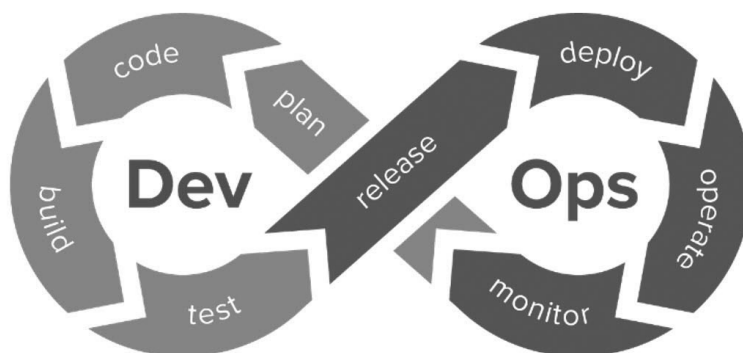
- 9.1 การพัฒนาและติดตั้งระบบ (System Implementation)
- 9.2 การทดสอบซอฟต์แวร์สำหรับแบบจำลองน้ำตก (Water Fall Model Software Testing)
- 9.3 การทดสอบซอฟต์แวร์สำหรับอไจล์ (Agile Software Testing)
- 9.4 การติดตั้งระบบ (System Installation)
- 9.5 การจัดทำเอกสาร (Documentation)
- 9.6 การฝึกอบรม (Training)
- 9.7 การบำรุงรักษาซอฟต์แวร์ (Software Maintenance)

9.1.3 วิธีการพัฒนาโปรแกรมแบบ Agile และ DevOps

การพัฒนาระบบแบบดั้งเดิมหรือที่เรียกว่า แบบจำลองน้ำตก (Water Fall Model) ตามที่ได้กล่าวมาแล้วนั้นเหมาะกับระบบงานที่มีการพัฒนาขึ้นมาใหม่ ซึ่งผ่านการวิเคราะห์ความต้องการมาอย่างดี ในขณะที่รูปแบบอไจล์ (Agile) คือ แนวคิดในการทำงานที่ให้ความสำคัญในการสื่อสารกับผู้เกี่ยวข้องทุกฝ่าย เพื่อการปรับปรุงและพัฒนาผลิตภัณฑ์อยู่ตลอดเวลา เพื่อตอบสนองความต้องการของผู้ใช้งาน ซึ่งได้กล่าวมาแล้วในบทที่ 5

ในการพัฒนาระบบแบบอไจล์นั้น ในบางครั้งหากมีการเปลี่ยนแปลงมาก ๆ หลังจากทีระบบได้ดำเนินการไปแล้วอาจทำให้เกิดปัญหาเรื่องของการดำเนินการ (Operation) เช่น ความเข้ากันได้ของระบบ ซึ่งอาจจะส่งผลไปยังโครงสร้างพื้นฐานที่เคยเตรียมไว้ และทำให้ทีมปฏิบัติการ (Operations) เองก็ไม่ทราบว่าจะตรวจสอบปัญหาได้จากที่ใด เพราะว่ามี การเปลี่ยนแปลงที่มากเกินไป จึงเกิดแนวทางการทำงานที่ผสมผสานทั้งสองทีมเข้าด้วยกัน ระหว่างทีมผู้พัฒนาซอฟต์แวร์ (Development, Dev) ที่ทำหน้าที่พัฒนาซอฟต์แวร์ที่สามารถตอบสนองความต้องการของลูกค้าได้อย่างรวดเร็ว กับทีมปฏิบัติการ (Operations, Ops) ที่ทำหน้าที่ให้บริการซอฟต์แวร์อย่างมีประสิทธิภาพ ทั้งสองทีมจะจับมือทำงานร่วมงานเป็นทีมเดียวกัน จึงเรียกว่า DevOps (Dev + Ops)

ประโยชน์ของโมเดล DevOps คือ ส่งมอบซอฟต์แวร์ได้รวดเร็วขึ้น ออกรุ่นใหม่ ๆ ได้ถี่ขึ้น แก้ไขจุดบกพร่องได้เร็วขึ้น ปรับตัวต่อการเปลี่ยนแปลงของตลาดได้ดียิ่งขึ้น ลดต้นทุน ลดค่าใช้จ่าย และลดการหยุดทำงาน (Downtime) ทำให้ได้รับความพึงพอใจและความน่าเชื่อถือจากผู้ใช้มากขึ้น นอกจากนี้ยังสามารถใช้การเข้ารหัสการทดสอบ การเตรียมโครงสร้างพื้นฐาน การปรับใช้และการตรวจสอบ และช่วยให้ย้อนกลับรหัสที่กำหนดเวอร์ชันได้ง่ายขึ้นในกรณีของการกู้คืนระบบ และทำให้สภาพแวดล้อมสามารถปรับขนาดได้ง่ายและปลอดภัย



รูปที่ 9.11 กระบวนการทำงานแบบ DevOps [11]

จากรูปที่ 9.11 แสดงให้เห็นกระบวนการทำงานที่เชื่อมโยง DevOps เป็นชุดของการปฏิบัติที่บูรณาการการพัฒนาซอฟต์แวร์ และการดำเนินงานด้านเทคโนโลยีสารสนเทศ โดยมีจุดมุ่งหมายเพื่อย่นระยะเวลาการพัฒนา ระบบ และให้การส่งมอบอย่างต่อเนื่องพร้อมกับคุณภาพซอฟต์แวร์ที่สูง ทั้งนี้ DevOps เป็นส่วนเสริมของการพัฒนาซอฟต์แวร์แบบอไจล์ [16]



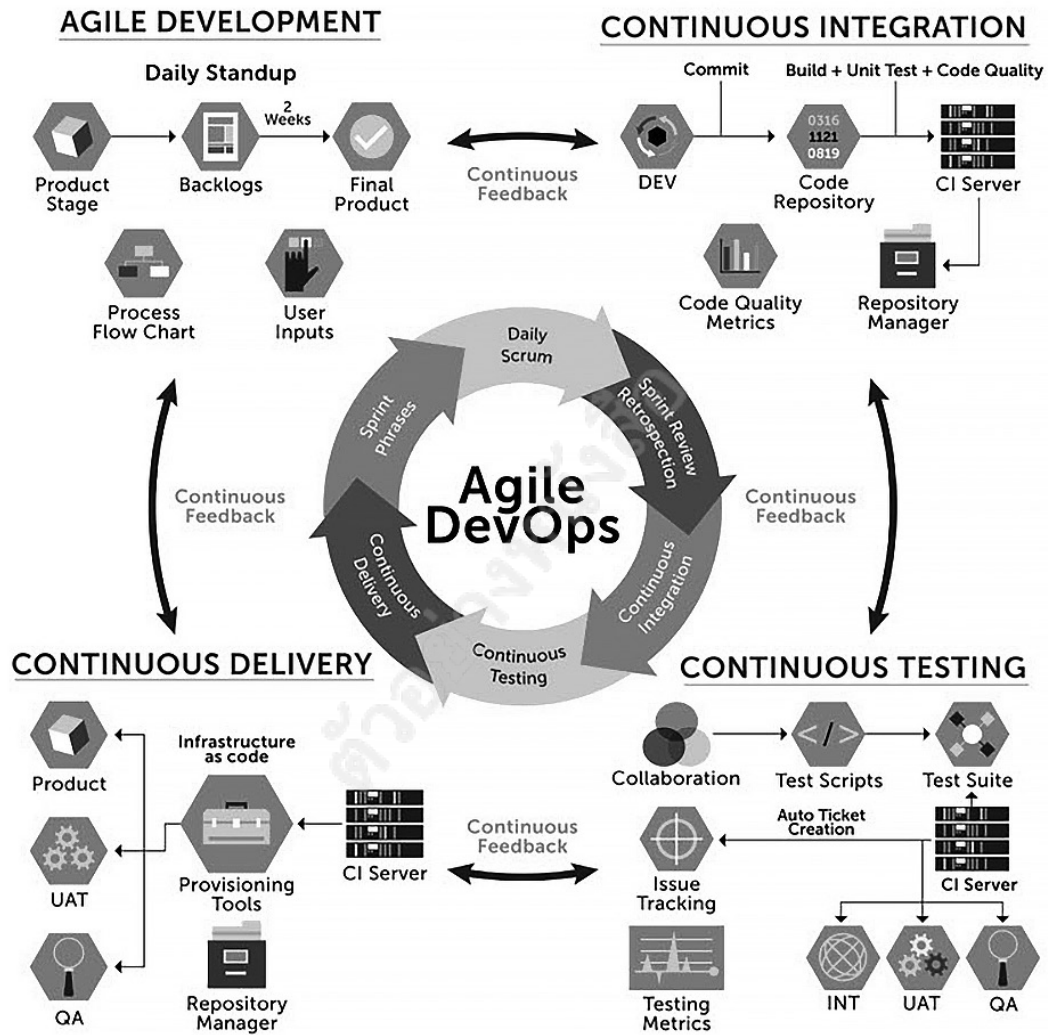
รูปที่ 9.12 แสดงการสื่อสารระหว่างผู้เกี่ยวข้องกับซอฟต์แวร์ (Stakeholders)
ผู้ใช้หรือลูกค้า (Customer) ผู้พัฒนา (Developer, Dev) และผู้ดูแลระบบ (IT Operation) [9]



รูปที่ 9.13 Agile ช่วยจัดการกับช่องว่างในการสื่อสารของ Customer กับ Developer [9]



รูปที่ 9.14 DevOps ช่วยจัดการกับช่องว่างในการสื่อสารของ Developer และ IT Operations [9]



รูปที่ 9.15 การพัฒนาโดย Agile DevOps

(ที่มา : <https://www.pinterest.com/pin/164592561369080180/>)

10

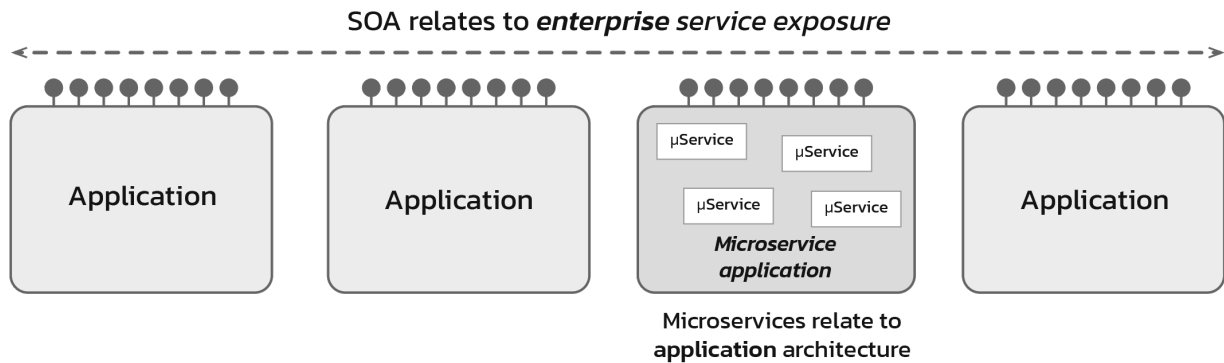
วิศวกรรมซอฟต์แวร์เชิงเซอร์วิส SERVICE ORIENTED SOFTWARE ENGINEERING

วัตถุประสงค์ (Objectives)

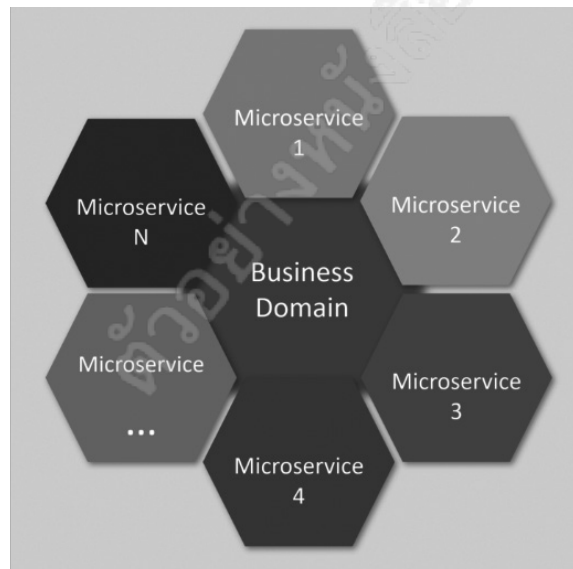
- เพื่ออธิบายวิศวกรรมซอฟต์แวร์เชิงเซอร์วิส
- เพื่ออธิบายวิธีการบูรณาการระบบด้วยสถาปัตยกรรมเชิงเซอร์วิส
- เพื่ออธิบายเทคโนโลยีเว็บเซอร์วิส
- เพื่ออธิบายวิธีการแลกเปลี่ยนข้อมูลระหว่างแอปพลิเคชัน
- เพื่ออธิบายการพัฒนากระบวนการด้วยสถาปัตยกรรมเชิงเซอร์วิส

เนื้อหาประจําบท (Contents)

- 10.1 วิศวกรรมซอฟต์แวร์เชิงเซอร์วิส (Service Oriented Software Engineering)
- 10.2 รูปแบบการพัฒนาซอฟต์แวร์เชิงเซอร์วิส
- 10.3 เทคโนโลยีเว็บเซอร์วิส (Web Services)
- 10.4 รูปแบบข้อมูลที่ใช้ในเว็บเซอร์วิส
- 10.5 สถาปัตยกรรมเชิงเซอร์วิส (Service Oriented Architecture, SOA)
- 10.6 ไมโครเซอร์วิส (Microservices)
- 10.7 ซอฟต์แวร์คอนเทนเนอร์ (Software Container)



รูปที่ 10.26 ความสัมพันธ์ของไมโครเซอร์วิสกับสถาปัตยกรรมเชิงเซอร์วิส
(ที่มา : <https://www.ibm.com/cloud/blog/soa-vs-microservices>)



รูปที่ 10.27 ภาพรวมของแนวคิดในการกระจายการทำงานแบบไมโครเซอร์วิส (Microservices)
(ที่มา : <https://www.edureka.co/blog/microservices-security>)

ไมโครเซอร์วิส เป็นวิธีการแยกส่วนของซอฟต์แวร์ ที่มีจุดมุ่งหมายเพื่อแบ่งระบบซอฟต์แวร์ขนาดใหญ่ออกเป็นส่วนประกอบขนาดเล็ก ด้วยสถาปัตยกรรมไมโครเซอร์วิส แอปพลิเคชันถูกสร้างขึ้นด้วยกลุ่มคอมโพเนนต์อิสระที่รันแต่ละกระบวนการแอปพลิเคชันเป็นเซอร์วิส สถาปัตยกรรมดังกล่าวช่วยให้แอปพลิเคชันปรับขนาดได้ง่ายขึ้น เร่งกระบวนการพัฒนามีพื้นที่สำหรับการทดลอง และลดเวลาออกสู่ตลาดสำหรับคุณสมบัติใหม่ ๆ

11

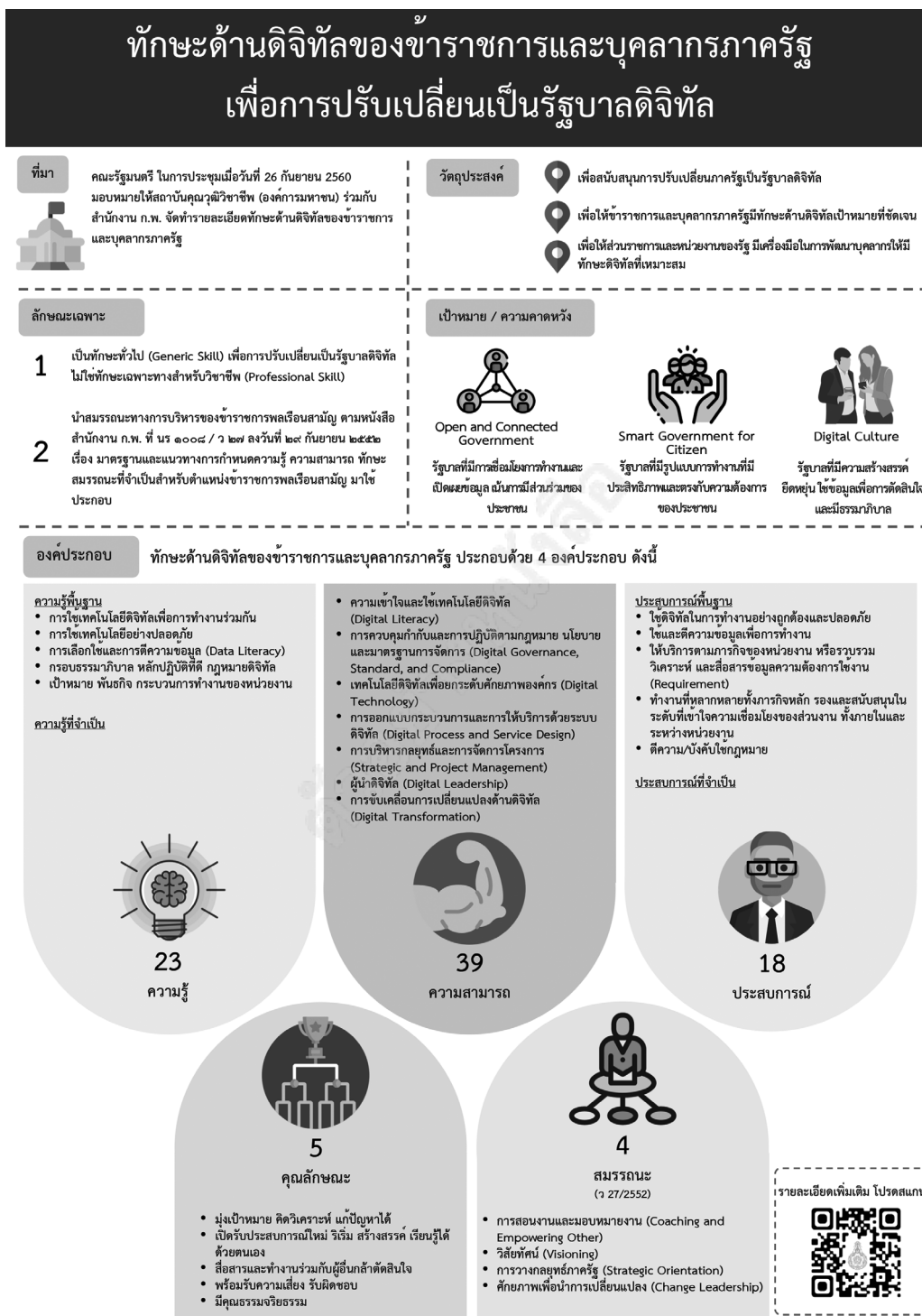
การพัฒนาบุคลากรด้านดิจิทัล และปัญญาประดิษฐ์ HUMAN RESOURCE DEVELOPMENTS IN DIGITAL & ARTIFICIAL INTELLIGENCE

วัตถุประสงค์ (Objectives)

- เพื่ออธิบายกระบวนการพัฒนาบุคลากรด้านดิจิทัล และปัญญาประดิษฐ์
- เพื่ออธิบายทักษะดิจิทัล และปัญญาประดิษฐ์ที่จำเป็นต่อการปรับเปลี่ยนสู่ดิจิทัล
- เพื่ออธิบายการพัฒนาบุคลากรด้านการพัฒนาเทคโนโลยีดิจิทัล และปัญญาประดิษฐ์

เนื้อหาประจําบท (Contents)

- 11.1 ทักษะด้านดิจิทัล และปัญญาประดิษฐ์
- 11.2 สมรรถนะของบุคลากรด้านการพัฒนาเทคโนโลยีดิจิทัล และปัญญาประดิษฐ์
- 11.3 การพัฒนาบุคลากรด้านการพัฒนาเทคโนโลยีดิจิทัล และปัญญาประดิษฐ์
- 11.4 การพัฒนาทักษะความเข้าใจ และการใช้ปัญญาประดิษฐ์



รูปที่ 11.2 องค์ประกอบทักษะด้านดิจิทัลของข้าราชการและบุคลากรภาครัฐ [4]

12

ปัญญาประดิษฐ์กับวิศวกรรมซอฟต์แวร์ ARTIFICIAL INTELLIGENCE AND SOFTWARE ENGINEERING

วัตถุประสงค์ (Objectives)

- เพื่ออธิบายการใช้ปัญญาประดิษฐ์ (AI) ในวิศวกรรมซอฟต์แวร์ได้อย่างเป็นระบบ
- เพื่ออธิบายแนวคิดและแนวปฏิบัติของธรรมาภิบาลปัญญาประดิษฐ์ (AI Governance) รวมถึงกรอบมาตรฐาน และแนวทางที่สำคัญในการใช้ปัญญาประดิษฐ์กับวิศวกรรมซอฟต์แวร์

เนื้อหาประจําบท (Contents)

- 12.1 บริบทและแนวคิดพื้นฐาน
- 12.2 ปัญญาประดิษฐ์กับวิศวกรรมซอฟต์แวร์
- 12.3 วิศวกรรมซอฟต์แวร์เพื่อพัฒนาปัญญาประดิษฐ์

สถาปัตยกรรมของระบบปัญญาประดิษฐ์เชิงตัวแทน มี 3 แบบ คือ

1. **Single Agent** เอเจนต์ที่การทำงานแบบเบ็ดเสร็จในตัวเอง โดยใช้โมเดลภาษาขนาดใหญ่ (LLM) เป็นสมองส่วนกลางระบบปัญญาประดิษฐ์ ที่ทำงานหรือตัดสินใจเพียงลำพังเพื่อบรรลุเป้าหมาย
2. **Multi-Agent** เอเจนต์หลายตัวที่มีความเชี่ยวชาญต่างกันมาทำงานร่วมกัน เพื่อแก้ปัญหาที่ซับซ้อนเกินกว่าเอเจนต์ตัวเดียวจะจัดการได้ โปรโตคอลมาตรฐานที่ช่วยให้เอเจนต์ต่างค่ายสื่อสารและใช้เครื่องมือร่วมกันได้ เช่น Model Context Protocol (MCP), Agent-to-Agent (A2A)
3. **Human-Machine Agent** ระบบความร่วมมือที่มนุษย์และระบบปัญญาประดิษฐ์อัตโนมัติทำงานร่วมกัน แทนที่ปัญญาประดิษฐ์จะเป็นผู้รับคำสั่งเพียงอย่างเดียว ใช้ในงานที่มีความเสี่ยงสูง เน้นความปลอดภัย (Safety) และการตรวจสอบ (Accountability)

Impact of AI Agents on SDLC



รูปที่ 12.8 การใช้ปัญญาประดิษฐ์เชิงตัวแทน (AI Agent) ในวงจรการพัฒนาซอฟต์แวร์
(ที่มา : <https://www.valuecoders.com/blog/ai-ml/ai-agents-and-their-impact-on-transforming-sdlc/>)

ระยะที่ 3 : สร้างวัฒนธรรม (Culture & Awareness)

เพื่อให้พนักงานตระหนักรู้และกล้าพูด

1. **การอบรม (Training)** ให้ความรู้เรื่องจริยธรรมปัญญาประดิษฐ์ (AI Ethics) ความมั่นคงปลอดภัยไซเบอร์ และการใช้ปัญญาประดิษฐ์อย่างมั่นคงปลอดภัย แก่พนักงานหน่วยงาน ไม่ใช่แค่หน่วยงานด้านเทคโนโลยีสารสนเทศ
2. **เปิดพื้นที่ปลอดภัย (Open Communication)** สร้างวัฒนธรรมที่พนักงานกล้ารายงานเมื่อสงสัยว่า AI ทำงานผิดพลาด หรือผลลัพธ์ที่ไม่ถูกต้องโดยไม่ถูกลงโทษ (Blame-Free Reporting)

ระยะที่ 4 : การตรวจสอบและติดตาม (Monitor & Review)

เพื่อกำหนดรอบการทบทวนนโยบาย เช่น ทุก 6 เดือน เพราะเทคโนโลยี AI เปลี่ยนเร็วมาก



รูปที่ 12.10 กรอบการดำเนินงาน Artificial Intelligence Risk Management Framework (AI RMF 1.0) [6]

13

การรักษาความมั่นคงปลอดภัยทางไซเบอร์ และปัญญาประดิษฐ์ CYBER SECURITY AND ARTIFICIAL INTELLIGENCE SECURITY

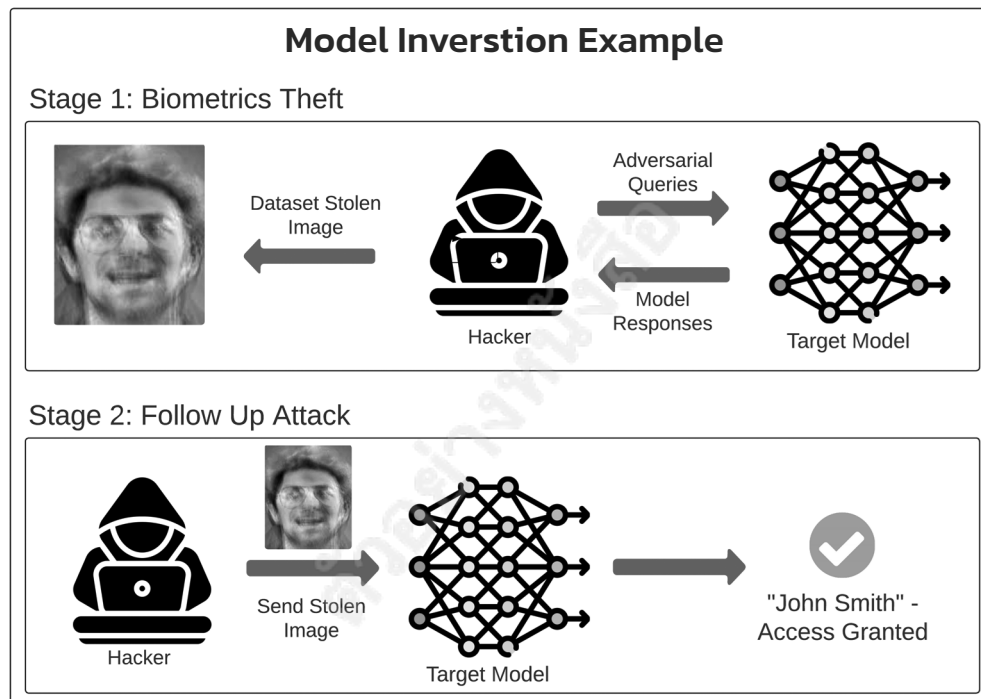
วัตถุประสงค์ (Objectives)

- เพื่ออธิบายหลักการของการรักษาความมั่นคงปลอดภัยไซเบอร์ (Cyber Security)
- เพื่ออธิบายหลักการใช้ปัญญาประดิษฐ์อย่างมั่นคงปลอดภัย (AI Security)
- เพื่อประยุกต์หลักการ Cyber Security และ AI Security กับกระบวนการวิศวกรรมซอฟต์แวร์

เนื้อหาประจำบท (Contents)

- 13.1 หลักการรักษาความมั่นคงปลอดภัยไซเบอร์
- 13.2 หลักการใช้ปัญญาประดิษฐ์อย่างมั่นคงปลอดภัย
- 13.3 กรอบมาตรฐานและแนวปฏิบัติ Cyber Security และ AI Security
- 13.4 การบูรณาการ Cyber Security และ AI Security กับวิศวกรรมซอฟต์แวร์

2. การละเมิดข้อมูลส่วนบุคคล (Data Privacy Breach) โมเดลปัญญาประดิษฐ์อาจจดจำข้อมูลส่วนบุคคลจากชุดข้อมูลฝึกสอนโดยไม่ได้เจตนา และอาจเปิดเผยข้อมูลดังกล่าวออกมาได้ในขั้นตอนการอนุมาน ผู้โจมตีใช้เทคนิคการโจมตีแบบอนุมาน (Inference Attack) เช่น Model Inversion Attack จากนั้นส่งคำถามเชิงโจมตี (Adversarial Queries) เพื่อพยายามสร้างข้อมูลฝึกสอนบางส่วนขึ้นมาใหม่ หรือใช้ Membership Inference เพื่อตรวจสอบว่าข้อมูลส่วนบุคคลเคยถูกใช้ในการฝึกโมเดลหรือไม่



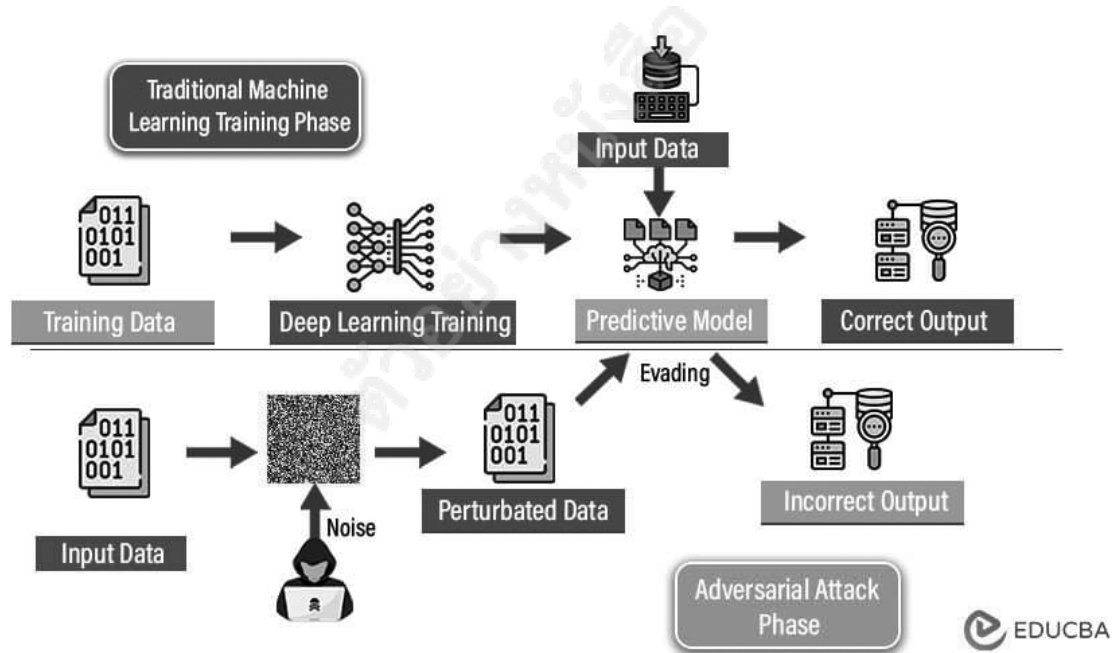
รูปที่ 13.3 การโจมตีแบบ Model Inversion Attacks

(ที่มา : <https://mindgard.ai/blog/ai-under-attack-six-key-adversarial-attacks-and-their-consequences>)

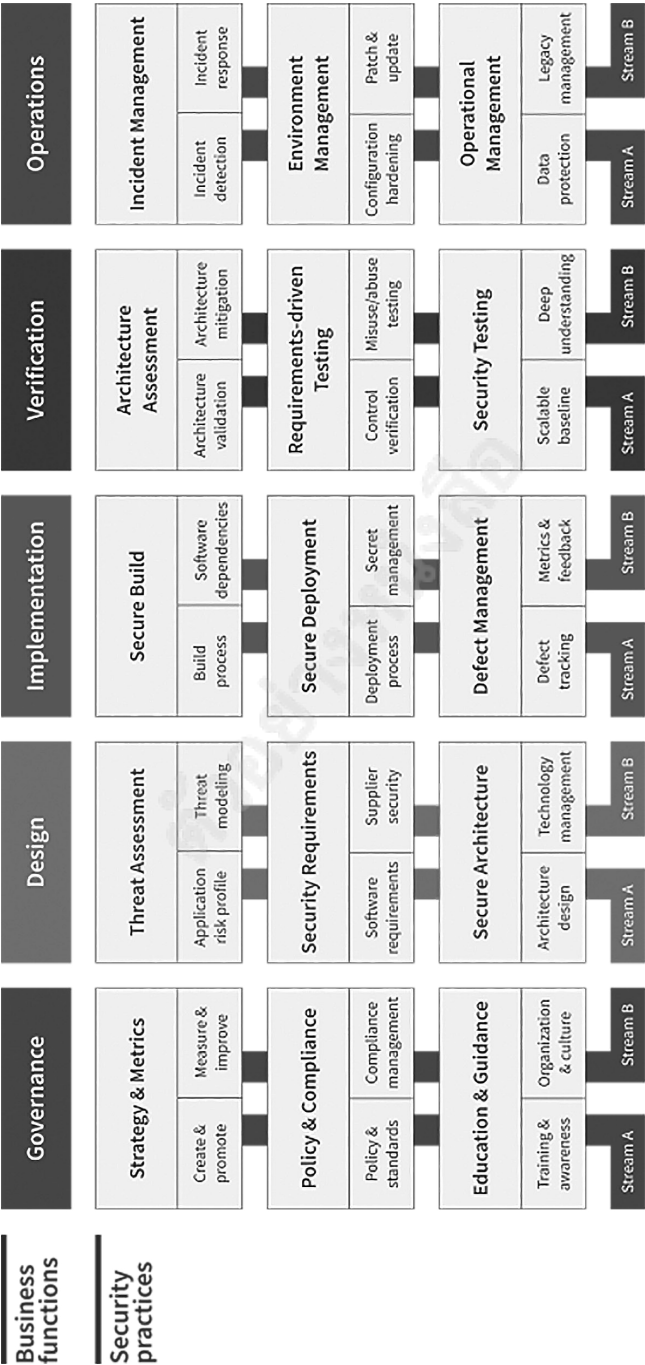
รูปที่ 13.3 ตัวอย่าง Model Inversion ขั้นที่ 1 คือ ผู้โจมตีขโมยข้อมูลชีวภาพ (Biometrics Theft) จากนั้นส่งคำถามเชิงโจมตี (Adversarial Queries) เข้าไปยังโมเดลเป้าหมาย และเก็บคำตอบของโมเดล (Model Responses) กลับมา เพื่อนำผลลัพธ์เหล่านี้ไปสร้างข้อมูลที่ระบุตัวตนที่เกี่ยวข้องกับเจ้าของข้อมูล ขั้นที่ 2 คือ การโจมตีต่อเนื่อง (Follow-up Attack) เมื่อผู้โจมตีมีข้อมูลหรือภาพที่ได้มาแล้ว ก็ส่งต่อให้ระบบยืนยันตัวตนอีกครั้งเพื่อใช้งานจริง ผลคือระบบอาจเชื่อว่าเป็นบุคคลนั้น และอนุญาตการเข้าถึง (Access Granted) และปล่อยให้ผ่านเข้าไปใช้งานได้

3. การหลบหลีก (Evasion) เป็นการสร้างข้อมูลนำเข้าเชิงประปรักษ์ในขั้นตอนการอนุมาน เพื่อให้โมเดลเกิดการจำแนกประเภทที่ผิดพลาด ผู้โจมตีสร้างตัวอย่างเชิงประปรักษ์ (Adversarial Example) โดยการเจือปนข้อมูลนำเข้าเล็กน้อย ซึ่งมนุษย์ไม่สามารถสังเกตเห็นได้ แต่เพียงพอที่จะหลอกให้โมเดลตัดสินใจผิดพลาด การโจมตีแบบนี้จำแนกได้เป็น 2 ประเภท ได้แก่

- 3.1 White-box attacks ผู้โจมตีมีความรู้โดยสมบูรณ์เกี่ยวกับสถาปัตยกรรม พารามิเตอร์ และข้อมูลการฝึกของโมเดล
- 3.2 Black-box attacks ผู้โจมตีไม่มีความรู้เกี่ยวกับกลไกภายในของโมเดล และสามารถโต้ตอบกับโมเดลได้ผ่านทาง API เท่านั้น



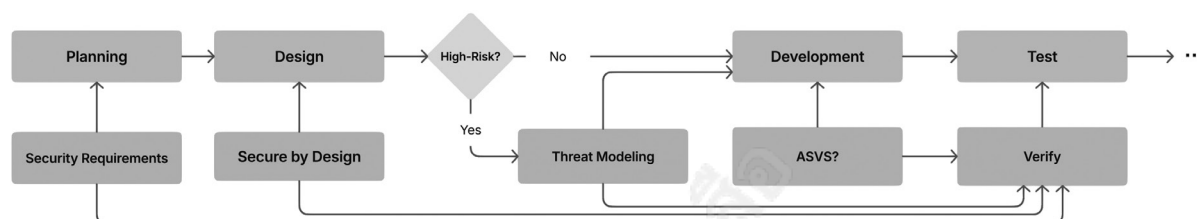
รูปที่ 13.4 ผู้โจมตีเติมการรบกวนเล็ก ๆ (Noise) ลงในข้อมูลเพื่อหลอกให้โมเดลตัดสินใจผิดพลาด
(ที่มา : <https://www.educba.com/adversarial-machine-learning/>)



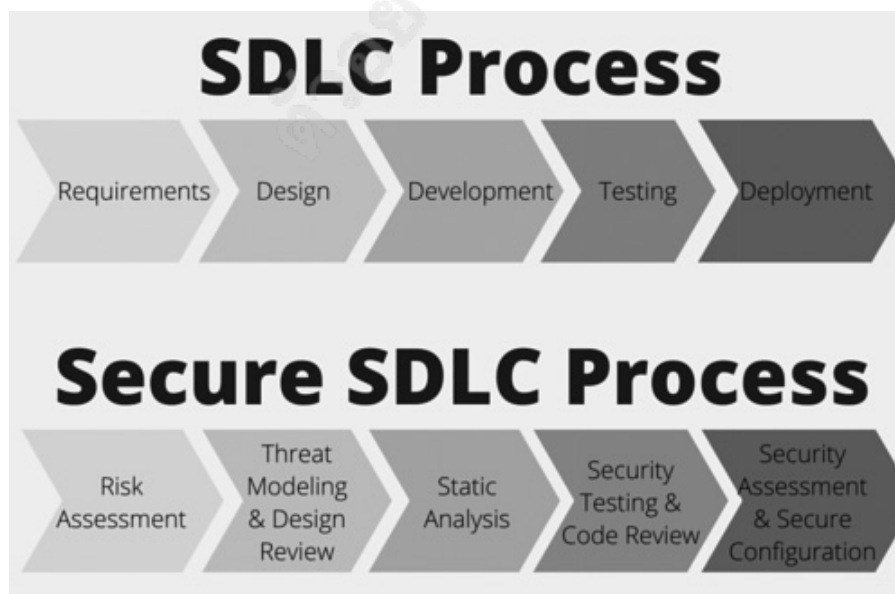
รูปที่ 13.8 Software Assurance Maturity Model (SAMM) [13]

ขั้นทดสอบ (Test) ซึ่งไม่ใช่แค่ทดสอบการทำงานของระบบ แต่รวมถึงการทดสอบด้านความมั่นคงปลอดภัย (Security Testing) เพื่อยืนยัน (Verify) ว่าระบบทำตามข้อกำหนดด้านความมั่นคงปลอดภัยที่กำหนดไว้

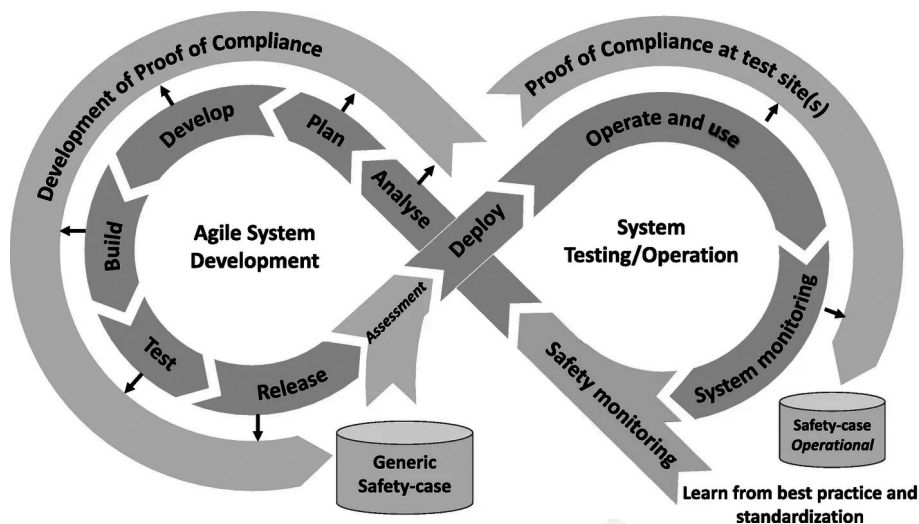
ทั้งนี้ หากพบปัญหาหรือช่องโหว่ในระหว่างออกแบบ พัฒนา หรือทดสอบ ก็จะมีการกลับไปแก้ไขตั้งแต่ระดับข้อกำหนดหรือการออกแบบใหม่ ดังรูป



รูปที่ 13.11 วงจรการพัฒนาซอฟต์แวร์ที่คำนึงถึงความมั่นคงปลอดภัย (Secure SDLC)
(ที่มา : <https://owasp.org/www-project-secure-by-design-framework/>)



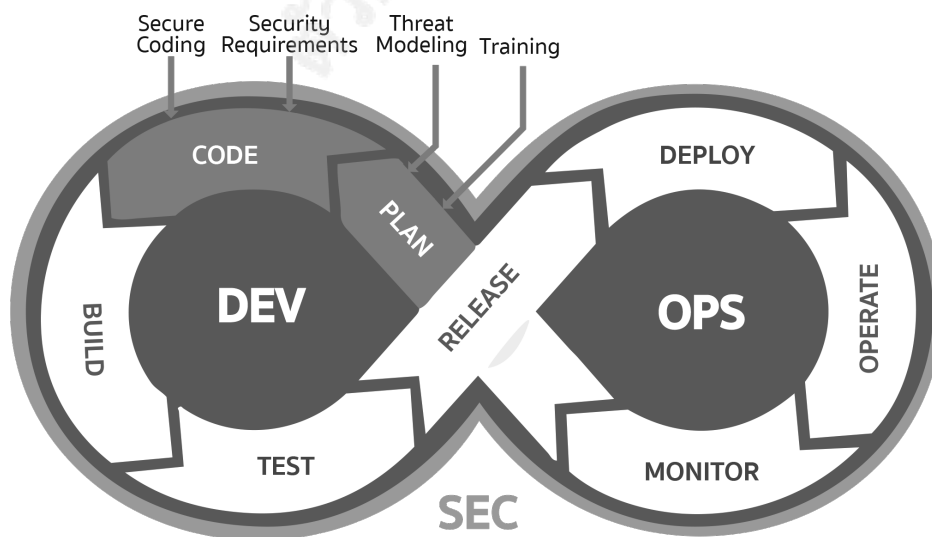
รูปที่ 13.12 เปรียบเทียบกระบวนการพัฒนาซอฟต์แวร์แบบ SDLC และ Secure SDLC
(ที่มา : <https://checkmarx.com/glossary/a-secure-sdlc-with-static-source-code-analysis-tools/>)



รูปที่ 13.13 วงจรชีวิตของซอฟต์แวร์ และ AI ในการพัฒนาระบบแบบ Agile [15]

DevSecOps ตามหลัก Security by Design คือ การผสมผสานหลักการด้านความมั่นคงปลอดภัยเข้าไปในทุกขั้นตอนของวงจรการพัฒนาซอฟต์แวร์ (SDLC) ตั้งแต่ระยะ Plan และ Code โดยคำนึงถึงความมั่นคงปลอดภัยก่อนส่งต่อไปยังขั้นตอนถัดไปของวงจรการพัฒนา ทั้งนี้ เพื่อให้ระบบมีความมั่นคงปลอดภัยตั้งแต่ต้นทาง

DevSecOps & Security by Design



รูปที่ 13.14 กระบวนการทำงานแบบ DevSecOps

(ที่มา : <https://www.securitycompass.com/blog/what-is-Security-by-Design/>)

14

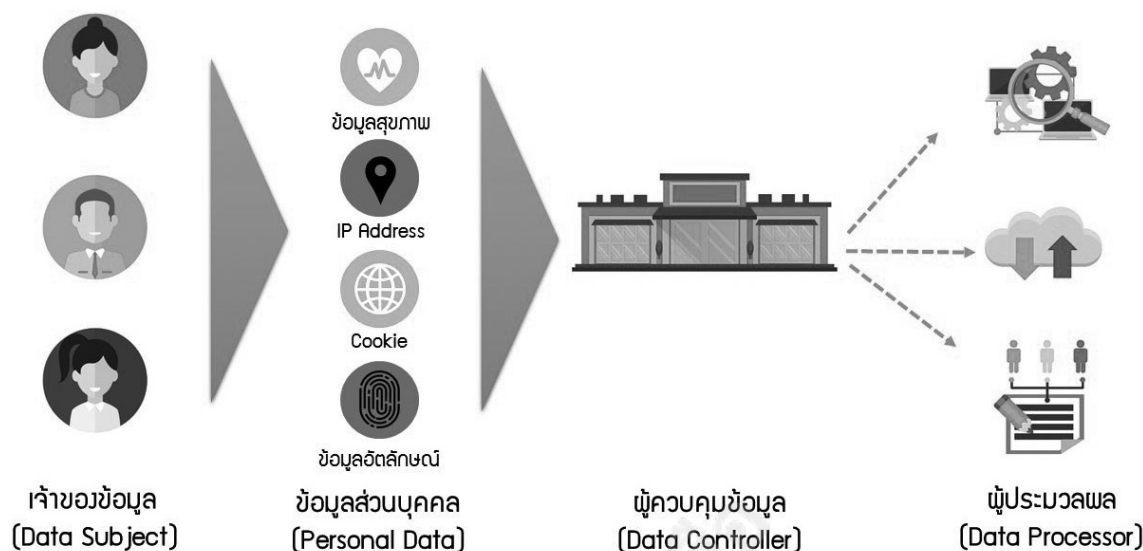
กฎหมายและจริยธรรมที่เกี่ยวข้อง RELATED LAWS AND ETHICS

วัตถุประสงค์ (Objectives)

- เพื่ออธิบายหลักการและสาระสำคัญของกฎหมายที่เกี่ยวข้องกับวิศวกรรมซอฟต์แวร์
- เพื่ออธิบายการวิเคราะห์และบูรณาการข้อกำหนดทางกฎหมายและจริยธรรมเข้าสู่วงจรชีวิตการพัฒนา ระบบ
- เพื่ออธิบายและประยุกต์ใช้หลักการคุ้มครองข้อมูลส่วนบุคคลในการออกแบบและพัฒนาซอฟต์แวร์
- เพื่ออธิบายแนวทางการบริหารความเสี่ยงและความรับผิดชอบด้านกฎหมาย ความมั่นคงปลอดภัยไซเบอร์ และทรัพย์สินทางปัญญา
- เพื่ออธิบายหลักการกำกับดูแลปัญญาประดิษฐ์ บริการคลาวด์ และจริยธรรมดิจิทัล

เนื้อหาประจำบท (Contents)

- 14.1 พรบ. คุ้มครองข้อมูลส่วนบุคคล
- 14.2 พรบ. การกระทำความผิดเกี่ยวกับคอมพิวเตอร์
- 14.3 พรบ. การรักษาความมั่นคงปลอดภัยไซเบอร์
- 14.4 การดำเนินการตามกฎหมายสำคัญ
- 14.5 กฎหมายเกี่ยวกับทรัพย์สินทางปัญญา
- 14.6 กฎหมายเกี่ยวกับปัญญาประดิษฐ์
- 14.7 กฎหมายเกี่ยวกับบริการคลาวด์
- 14.8 จริยธรรมดิจิทัล



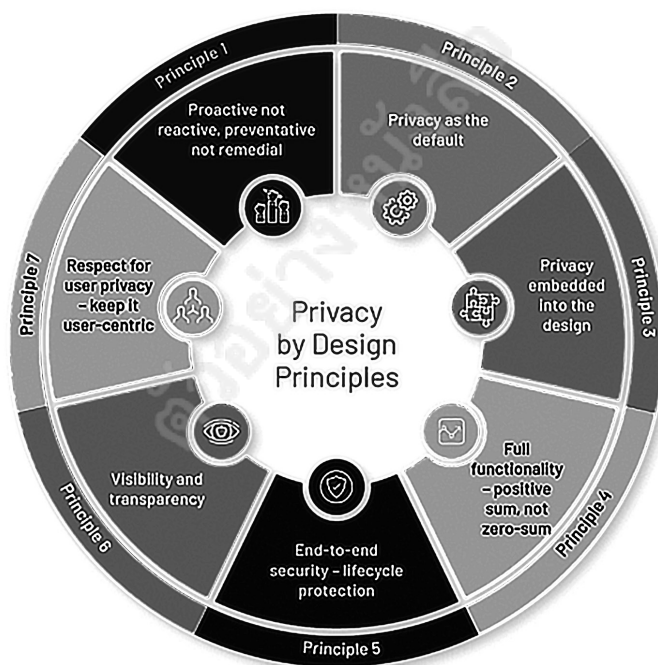
รูปที่ 14.1 องค์ประกอบตามกฎหมาย PDPA
(ที่มา : สำนักงานคณะกรรมการคุ้มครองข้อมูลส่วนบุคคล)

ผู้พัฒนาซอฟต์แวร์ต้องรู้ว่าข้อมูลไหนเป็นข้อมูลส่วนบุคคล ข้อมูลส่วนบุคคลอ่อนไหว และจำแนกบทบาทของผู้ควบคุมข้อมูล และผู้ประมวลผลข้อมูล เพื่อออกแบบสิทธิ์ สัญญา และการตรวจสอบย้อนกลับ (Audit trail) ได้อย่างถูกต้อง ทั้งนี้ เพราะกฎหมายบังคับใช้ทั้งนิติบุคคลและบุคคล ในบางกรณีอาจจะต้องวินิจฉัยบทบาทในขณะที่มีการประมวลผลข้อมูลส่วนบุคคล เช่น พนักงานของหน่วยงานทำการ download ข้อมูลส่วนบุคคลจาก server ของหน่วยงานมาไว้ในเครื่องคอมพิวเตอร์ส่วนบุคคล ถ้าหากเกิดเหตุละเมิดจาก server ของหน่วยงาน ผู้แทนนิติบุคคลของหน่วยงานในฐานะผู้ควบคุมข้อมูลต้องรับผิดชอบตามกฎหมาย แต่หากเหตุละเมิดเกิดในเครื่องคอมพิวเตอร์ส่วนบุคคลและพิสูจน์ได้ว่าเป็นการตัดสินใจในการประมวลผลข้อมูลส่วนบุคคลของบุคคลดังกล่าว บุคคลดังกล่าวจะกลายเป็นผู้ควบคุมข้อมูลส่วนบุคคลตามกฎหมาย ซึ่งต้องรับผิดชอบตามกฎหมาย

14.1.2 วงจรชีวิตข้อมูลส่วนบุคคล

วงจรชีวิตของข้อมูลส่วนบุคคล (Personal Data Lifecycle) เป็นลำดับขั้นตอนตั้งแต่ก่อนเก็บข้อมูลไปจนถึงการทำลายข้อมูล ซึ่งผู้ควบคุมข้อมูลต้องปฏิบัติตามกฎหมาย ประกอบด้วย 6 ขั้นตอน ดังนี้

6. **ความโปร่งใสและความสามารถในการตรวจสอบ (Visibility and Transparency)** ผู้ควบคุมข้อมูลต้องดำเนินการให้การประมวลผลข้อมูลส่วนบุคคลมีความโปร่งใส สามารถอธิบายและตรวจสอบได้ โดยเจ้าของข้อมูลและผู้มีหน้าที่ตรวจสอบสามารถเข้าใจได้ว่าการเก็บรวบรวมข้อมูลประเภทใด ใช้เพื่อวัตถุประสงค์ใด และมีมาตรการคุ้มครองข้อมูลอย่างไร
7. **การเคารพสิทธิของเจ้าของข้อมูลส่วนบุคคล (Respect for User Privacy)** ผู้ควบคุมข้อมูลส่วนบุคคลต้องเคารพและอำนวยความสะดวกในการใช้สิทธิของเจ้าของข้อมูลตามที่กฎหมายกำหนด เช่น สิทธิในการเข้าถึง แก้ไข ลบ ระงับการใช้ หรือถอนความยินยอมเกี่ยวกับข้อมูลส่วนบุคคลของตน



รูปที่ 14.4 หลักการออกแบบโดยคำนึงถึงความเป็นส่วนตัว (Privacy by Design)
(ที่มา : <https://sentinelafrikaconsulting.com/the-7-principles-of-privacy-by-design/>)