วิศวกรรมซอฟต์แวร์ และ SDLC Models

วัตถุประสงค์การเรียนรู้

- ความเข้าใจพื้นฐาน: อธิบายประวัติและพัฒนาการของวิศวกรรมซอฟต์แวร์
- ความรู้เกี่ยวกับ SDLC: เข้าใจแบบจำลองการพัฒนาซอฟต์แวร์ที่แตกต่างกัน (Waterfall, Agile, DevOps)
- การเลือกวิธีการ: สามารถวิเคราะห์และเลือก SDLC ที่เหมาะสมกับบริบทของโครงการ

ประวัติและพัฒนาการของวิศวกรรมซอฟต์แวร์

ปัญหา Software Crisis

- เมื่อประมาณปี ค.ศ. 1960s–1970s มีปัญหาใหญ่ที่เรียกว่า "Software Crisis" (วิกฤตซอฟต์แวร์)
  - ต้นทุนสูง: โครงการซอฟต์แวร์มักเกินงบประมาณ
  - ล่าช้า: การพัฒนาใช้เวลานานกว่ากำหนด
  - คุณภาพต่ำ: ผลิตภัณฑ์มีข้อบกพร่องมากมาย
  - การบำรุงรักษายาก: เปลี่ยนแปลงโค้ดอย่างไร้ระเบียบ
- ตัวอย่าง
  - NASA Apollo Program: ต้องจ่ายเงินนับล้าน แต่ยังมีปัญหาหลายครั้ง
  - IBM System/360: การพัฒนาล่าช้า และมีข้อบกพร่องมากมาย
- **สิ่งที่ค้นพบ**
  - ต้องมีวิธีการทีมีระบบ เพื่อให้การพัฒนาซอฟต์แวร์เป็นไปอย่างมีประสิทธิภาพ

เกิดของวิศวกรรมซอฟต์แวร์

- ปี ค.ศ. 1968 – NATO Software Engineering Conference ที่โกเพนเฮเกน
  - เป็นครั้งแรกที่ใช้คำว่า "Software Engineering"
  - วัตถุประสงค์: นำแนวคิดทางวิศวกรรมมาใช้ในการพัฒนาซอฟต์แวร์
  - เปลี่ยนจากการ "เขียนโค้ด" เป็น "การพัฒนาซิสเต็ม"

- **นิยาม:** วิศวกรรมซอฟต์แวร์ คือการนำหลักวิศวกรรมไปใช้ในการพัฒนาซอฟต์แวร์อย่างเป็นระบบ เพื่อสร้างผลิตภัณฑ์ที่มีคุณภาพ

ยุคสมัยของวิศวกรรมซอฟต์แวร์

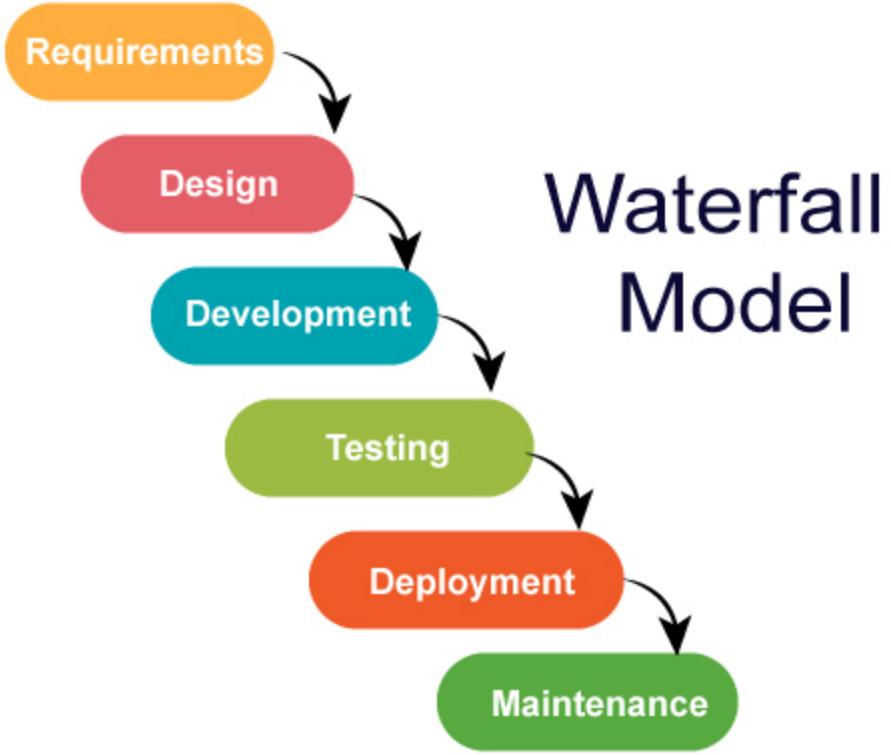
1. ยุคแรก (1960s–1970s): Waterfall Era
   - เน้นการวางแผนอย่างเข้มงวด
   - ทำตามขั้นตอนตามลำดับ
   - ผลสำเร็จขึ้นอยู่กับการวางแผนเบื้องต้น
2. ยุคที่สอง (1980s–1990s): Iterative & Component-based
   - เริ่มสร้างซ้ำแบบ (Iteration)
   - นำ Object-Oriented มาใช้
   - เน้นการใช้อุปกรณ์/ส่วนประกอบ (Components)
3. ยุคที่สาม (2000s): Agile Revolution
   - Agile Manifesto ประกาศในปี ค.ศ. 2001
   - เปลี่ยนจากการวางแผนเป็นการตอบสนอง
   - ทำงานแบบเพิ่มขึ้นทีละน้อย (Incremental)
4. ยุคปัจจุบัน (2010s–Present): DevOps & Cloud
   - DevOps รวม Development + Operations
   - การปรับใช้อย่างต่อเนื่อง (Continuous Deployment)
   - Microservices และ Cloud Computing

แบบจำลองการพัฒนาซอฟต์แวร์ (SDLC Models)

- https://www.tutorialscampus.com/sdlc/iterative-model.htm

Waterfall Model (แบบน้ำตก)

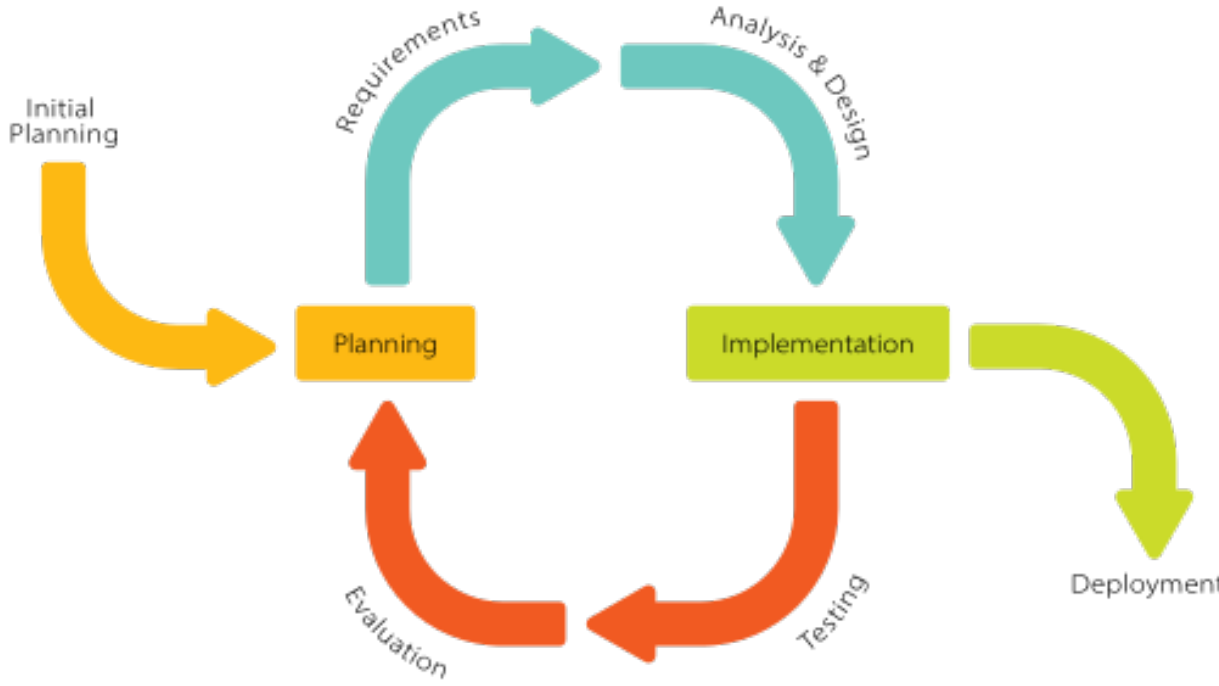
- ลักษณะเฉพาะ
  - ขั้นตอนตามลำดับ ไม่สามารถย้อนกลับ
  - ต้องจบแต่ละขั้นตอนก่อนไปขั้นถัดไป
  - ผลิตภัณฑ์จำหน่ายครั้งเดียว
- ข้อดี
  - ง่ายต่อการวางแผนและควบคุม
  - เหมาะกับโครงการที่มีความต้องการชัดเจน
  - เอกสารครบครัน
- ข้อเสีย
  - ไม่ยืดหยุ่น ยากต่อการเปลี่ยนแปลง
  - ปัญหาพบช่วงทดสอบ เสียเวลามาก
  - ไม่เหมาะกับโครงการขนาดใหญ่หรือไม่แน่นอน
- เหมาะสำหรับ
  - โครงการขนาดเล็ก
  - ความต้องการแน่นอน



https://www.instagantt.com/project-management/what-is-waterfall-project-management

Iterative Model (แบบวนซ้ำ)

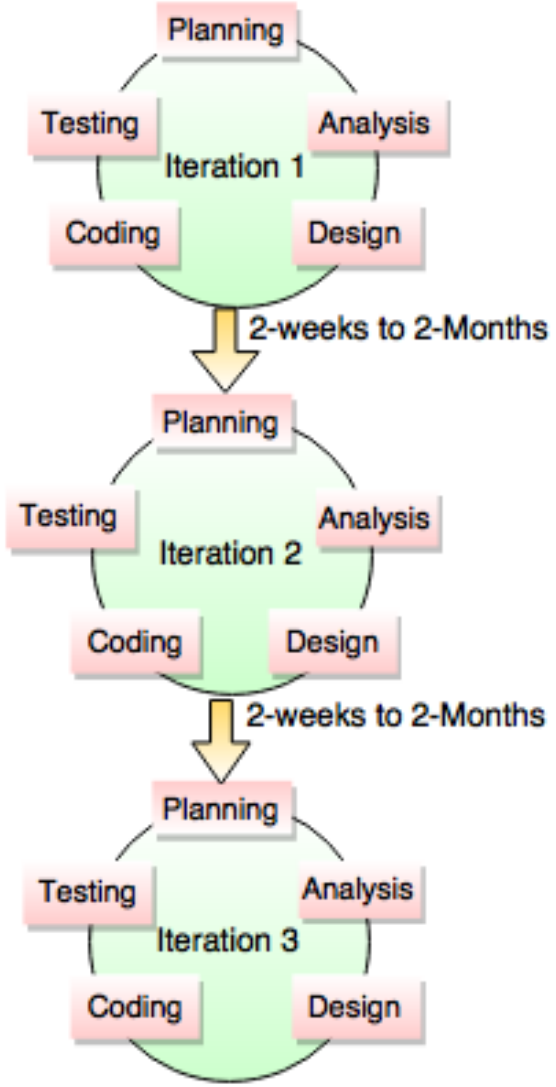
- ลักษณะเฉพาะ
  - ทำให้ลองได้ (Prototype) ให้ผู้ใช้ดู
  - ปรับปรุงตามการตอบรับ
  - ซ้ำกระบวนการจนกว่าจะพอใจ
- ข้อดี
  - ยืดหยุ่นพร้อมปรับเปลี่ยน (Flexible)
  - ได้รับตอบรับเร็ว
  - ลดความเสี่ยง
- ข้อเสีย
  - เพิ่มเติมฟีเจอร์ได้ช้า
  - ต้นทุนสูง
  - เอกสารไม่ชัดเจน
- เหมาะสำหรับ
  - โครงการที่ความต้องการไม่ชัดเจน
  - เช่น: แอปพลิเคชันใหม่



https://commons.wikimedia.org/wiki/File:Iterative_Process_Diagram.svg

Agile Model (แบบ Agile)

- ลักษณะเฉพาะ
  - แบ่งงานเป็น Sprint (จังหวะ) ยาว 1–2 สัปดาห์
  - ทุก Sprint ออกผลิตภัณฑ์ที่ใช้ได้
  - ได้รับตอบรับและปรับปรุงต่อเนื่อง
- ข้อดี
  - ตอบสนองต่อการเปลี่ยนแปลง
  - คุณภาพสูง (ทดสอบตลอด)
  - ลูกค้าพอใจ (ได้เห็นตลอด)
- ข้อเสีย
  - ต้องการ Commitment สูง
  - ยากต่อการควบคุมต้นทุน
  - ไม่เหมาะกับทีมกระจัดกระจาย
- เหมาะสำหรับ
  - โครงการที่เปลี่ยนแปลงเร็ว
  - ทีมเล็กถึงกลาง
  - เช่น: Startup, Mobile App



หลักการ 4 ประการของ Agile Manifesto (2001)
บุคคล และปฏิสัมพันธ์ **มากกว่า** กระบวนการและเครื่องมือ
ซอฟต์แวร์ที่ทำงาน **มากกว่า** เอกสารที่ครบครัน
ความร่วมมือกับลูกค้า **มากกว่า** การเจรจาต่อรองสัญญา
ตอบสนองต่อการเปลี่ยนแปลง **มากกว่า** ตามแผนอย่างตึง

DevOps Model (แบบ DevOps)

- ลักษณะเฉพาะ
  - รวม Developer และ Operations เข้าด้วยกัน
  - สอดส่วนไปยังการผลิต (Prod) อย่างต่อเนื่อง
  - เน้นการทำให้เป็นอัตโนมัติ
- ข้อดี
  - ปรับใช้เร็ว
  - ข้อมูลการทำงาน (Feedback) เร็ว
  - เข้าใจปัญหาจริง
- ข้อเสีย
  - ต้องเครื่องมือและทักษะมากมาย
  - ต้องการวัฒนธรรมองค์การที่เพิ่มเติม
- เหมาะสำหรับ
  - โครงการขนาดกลาง-ใหญ่
  - บริษัทที่มี Infrastructure
  - เช่น: Web Services, Cloud Apps

การเปรียบเทียบแบบจำลอง

ปัจจัย	Waterfall	Iterative	Agile	DevOps
ความยืดหยุ่น	ต่ำ	ปานกลาง	สูง	สูงมาก
ต้นทุน	ต่ำ (ทฤษฎี)	ปานกลาง-สูง	ปานกลาง	สูง
เวลา Release	นาน	ปานกลาง	สั้น	สั้นมากๆ
คุณภาพ	ขึ้นอยู่กับวางแผน	ปานกลาง	สูง	สูงมาก
ความเสี่ยง	สูง	ปานกลาง	ต่ำ	ต่ำ
เอกสาร	มากมาย	ปานกลาง	น้อย	ปานกลาง
ขนาดทีม	ใหญ่ได้	ปานกลาง	เล็ก	ปานกลาง

วิเคราะห์โครงการและเลือก SDLC

- ศึกษา Case Studies : ให้นักศึกษาอ่าน 3 โครงการจริงต่อไปนี้
- สำหรับแต่ละโครงการ ตอบคำถามต่อไปนี้

1. SDLC ที่เหมาะสมคืออะไร?
   - Waterfall / Iterative / Agile / DevOps

2. เหตุผลสำคัญ 3 ประการ:
   - Focus on constraints/requirements
   - Why NOT others?

3. Risk factors:
   - ถ้าเลือก SDLC นี้ risk อะไรมากที่สุด?

4. Success factors:
   - ต้อง fulfill อะไรบ้าง ให้โครงการสำเร็จ?

โครงการ A: Banking System (ระบบบัญชีธนาคาร)

```
Context:
- Deploy to 100+ branches nationwide
- Strict regulations (PCI DSS, Thai Banking Act)
- High security & reliability requirement
- Requirements: Well-defined, rarely change
- Timeline: 18 months
- Budget: High
- Team: Mixed seniority

Characteristics:
- Requirement stability: Very High (fixed)
- Risk level: Very High (financial data)
- Change frequency: Very Low
- Concurrent users: 1000+
- Integration: Multiple legacy systems
```

โครงการ B: Social Media Mobile App

```
Context:
- Startup with $1M funding
- Target: 100K users in Year 1
- Features must evolve based on user feedback
- Timeline: MVP in 3 months
- Team: 5 developers, young/energetic

Characteristics:
- Requirement stability: Low (changing)
- Risk level: Medium (startup risk)
- Change frequency: High (weekly features)
- Concurrent users: Growing (100 → 100K)
- Integration: Basic APIs
```

โครงการ C: E-Commerce Platform (Startup)

```
Context:
- Fast-growing online store
- Need rapid feature deployment
- Need continuous monitoring & improvement
- Dev team: 10 people
- Ops team: 3 people
- Infrastructure: Cloud-based

Characteristics:
- Requirement stability: Medium (evolving)
- Risk level: Medium (business risk)
- Change frequency: Medium-High (2-week sprints)
- Concurrent users: 5000+ real-time
- Integration: Payment, Email, Analytics services
```

Read more

- https://www.tutorialscampus.com/sdlc/iterative-model.htm
- https://gowide.com/devops-model-in-sdlc/