Team Communication & Risk Management

วัตถุประสงค์การเรียนรู้

- **ความเข้าใจ Team Dynamics:** อธิบายบทบาท หน้าที่ และการสื่อสารที่มีประสิทธิภาพในทีม
- **Communication Planning:** สร้างแผนการสื่อสารให้เหมาะสมกับโครงการ
- **Risk Management:** ระบุ วิเคราะห์ และจัดการความเสี่ยงของโครงการ
- **Stakeholder Management:** บริหารความคาดหวังของผู้เกี่ยวข้อง

แผนโครงงาน

- **Phase 1:** การวางแผนและข้อกำหนด (Week 1–5) → **D1 Deliverable (ส่ง Week 5)**
- **Week 3 Focus:** วิเคราะห์ข้อกำหนด & เอกสารพื้นฐาน
- **Alignment:** สัปดาห์นี้สร้าง Communication Plan + Risk Management + Project Charter
- **Deliverable:** ส่วนหนึ่งของ D1 (Project Charter + Risk Plan + Stakeholder Management)

Team Communication

ทำไม Team Communication ถึงสำคัญ

- **สถิติที่ สำคัญ:**
  - 90% ของการล้มเหลวโครงการ = ปัญหา Communication
  - Developer ใช้เวลา 30-50% ในการสื่อสาร (ไม่ทำโค้ด!)
  - เงินเดือนสูงสุดของ Developer = ทักษะ Communication ดี

```
การสื่อสารที่ไม่ดี:
- "แก้บักให้หน่อย" (คลุมเครือ)
- Meeting ยาวๆ ไม่มีวัตถุประสงค์
- Email ที่ส่งแล้วไม่ได้ตอบกลับ
- Slack message ที่มีบริบทไม่เพียงพอ

การสื่อสารที่ดี:
- "แก้บัก #123: Login Page ขัดข้องเมื่อ Email ไม่ถูกต้อง"
- Standup 15 นาที เป้าหมายชัดเจน
- ตอบกลับรวดเร็ว ให้ข้อเสนอแนะที่สามารถปฏิบัติได้
- บันทึกไว้ใน Wiki อ้างอิงใน PR
```

PR (Pull Request)

ความหมาย: การขออนุญาตเพื่อ "ดึง" (pull) การเปลี่ยนแปลงโค้ดจากสาขา (branch) หนึ่ง ไปยังสาขาหลัก

กระบวนการ:

- สร้าง branch ใหม่เพื่อทำงาน
- Commit การเปลี่ยนแปลง
- Push ขึ้น GitHub
- สร้าง PR เพื่อขออนุญาต
- Code review โดยเพื่อนร่วมทีม
- Merge เข้า main branch

ระดับการสื่อสาร

- **Synchronous**
  - ข้อดี: ได้ข้อเสนอแนะทันที ปฏิสัมพันธ์ได้ สร้างความไว้วางใจ
  - ข้อเสีย: รบกวน ต้องให้ความสนใจ
  - ตัวอย่าง: Standup, Pair Programming, Video Call
- **2. Asynchronous**
  - ข้อดี: มีความยืดหยุ่น มีเอกสารประกอบ มีเวลาคิด
  - ข้อเสีย: ตอบกลับช้า สูญเสียบริบท
  - ตัวอย่าง: Email, Slack, เอกสารประกอบ
- **3. Hybrid Approach**

แผนการสื่อสารสำหรับโครงการ

- **โครงงาน 4 คน ควรมี**

Channel	Purpose	Frequency	Participants
Daily Standup	Share progress	Every day 09:00	All 4
Slack Channel	Quick chat	As needed	All 4
Sprint Planning	Set goals	Every 2 weeks	All 4
Code Review	Quality check	Per PR	Lead Dev + Team
Weekly Sync	General discussion	1x/week	All 4
Email	Formal info	As needed	All 4

Communication Plan Document:

```
1. ข้อมูลติดต่อทีม
   - ชื่อ, โทรศัพท์, Email, Slack
   - ความคาดหวัง response time

2. ตารางการประชุม
   - Daily Standup: จันทร์-ศุกร์ 09:00
   - Sprint Planning: ทุก 2 สัปดาห์ วันศุกร์
   - Sprint Review: ทุก 2 สัปดาห์ วันศุกร์
   - Retrospective: หลังจาก Review

3. เส้นทาง Escalation
   - ปัญหา: บอก Scrum Master
   - Technical: ถาม Lead Developer
   - โครงการ: ถาม Product Owner

4. Repository เอกสาร
   - GitHub Wiki: Design docs
   - Google Drive: Presentations
   - Confluence: Knowledge base

5. เครื่องมือสื่อสาร
   - Code: GitHub/GitLab
   - Chat: Slack/Discord
   - Video: Zoom/Google Meet
   - Document: Google Docs/Markdown
```

การประชุมที่มีประสิทธิภาพ

- **กฎทองคำ**
  - **มีวัตถุประสงค์ชัดเจน**
    - ไม่ดี: "วันนี้มา Sync กัน"
    - ดี: "Standup: อัปเดต Progress + แก้สิ่งกีดขวาง"
  - **เชิญเฉพาะคนที่เกี่ยวข้อง**
    - เชิญทั้ง 20 คน → เสียเวลา
    - Standup: 4 คน + PM (ไม่บังคับ)
  - **ตั้งเวลาให้เหมาะสม**
    - Standup: 15 นาที (หยุดพอดี!)
    - Planning: สูงสุด 2 ชั่วโมง (ไม่ก็แบ่ง)
    - Review: สูงสุด 1 ชั่วโมง
  - **ทำ Agenda ก่อนประชุม**
    - Agenda:
      - ความเคลื่อนไหว Sprint (5 นาที)
      - สิ่งกีดขวาง (5 นาที)
      - วางแผนขั้นตอนต่อไป (5 นาที)
  - **บันทึก Action Items**
    - รายการที่ต้องทำ:
      - [ ] John: แก้ API ภายในวันพุธ
      - [ ] Sarah: ทบทวนออกแบบ ภายในวันพฤหัสบดี
      - [ ] Tom: ตั้งค่า CI/CD ภายในวันศุกร์

การสื่อสารทีมระยะไกล

สำหรับ Online/Hybrid Teams

- **วิธีปฏิบัติที่ดีสำหรับ Video Call:**
  - เปิดกล้อง (สร้างความไว้วางใจ)
  - ปิดเสียง Background Music (ลดการรบกวน)
  - ส่งลิงก์ผ่าน Chat
  - บันทึกการประชุม (สำหรับสมาชิก Async)
- **อัปเดตแบบ Asynchronous:**
  - รูปแบบ:
    - สิ่งที่ฉันทำ: [รายการพร้อมลิงก์]
    - สิ่งที่ฉันจะทำ: [แผน]
    - สิ่งกีดขวาง: [ถ้ามี]
  - ตัวอย่าง:
    - สิ่งที่ฉันทำ: สร้าง Login API Merged PR #45
    - สิ่งที่ฉันจะทำ: เขียน Tests ตั้งค่า CI/CD
    - สิ่งกีดขวาง: ต้องการ Database Schema จากทีม DB

Risk Management

- **ความเสี่ยง (Risk) คืออะไร**
  - **นิยาม:** ความเสี่ยง = ความเป็นไปได้ที่สิ่งไม่พึงประสงค์จะเกิด × ความรุนแรงของผลกระทบ

- ความเสี่ยง = ความเป็นไปได้ × ผลกระทบ

  ตัวอย่าง

  ความเสี่ยง: Server ขัดข้องในระบบ Production

  ความเป็นไปได้: 20% (ต่ำ)

  ผลกระทบ: $100,000/ชั่วโมง (สูง)

  ระดับความเสี่ยง: 20% × สูง = ปานกลาง

หมวดหมู่ความเสี่ยงในโครงการซอฟต์แวร์

- **ความเสี่ยงทางเทคนิค**
  - ออกแบบ Database ผิด
  - เลือก Technology ไม่เหมาะ
  - ปัญหา Performance
  - ช่องโหว่ด้าน Security
  - ปัญหา Integration

  ตัวอย่าง
    - ความเสี่ยง: "Database Schema ไม่สามารถ Scale เพื่อรองรับ 1M Users"
    - ความเป็นไปได้: ปานกลาง (30%)
    - ผลกระทบ: สูง (ต้อง Rewrite)
    - การบรรเทา: Load Test กับระเบียนหลายล้านรายการ

หมวดหมู่ความเสี่ยงในโครงการซอฟต์แวร์

- **ความเสี่ยงด้านเวลา**
  - เริ่มต้นช้า
  - Dependencies ใช้เวลานาน
  - สมาชิกทีมลาออก
  - ประมาณการเวลาต่ำเกินไป

  ตัวอย่าง
    - ความเสี่ยง: "Developer สำคัญลาออกระหว่างโครงการ"
    - ความเป็นไปได้: ต่ำ (10%)
    - ผลกระทบ: สูงมาก (โครงการหยุด)
    - การบรรเทา: โอนความรู้ + เอกสารประกอบ

หมวดหมู่ความเสี่ยงในโครงการซอฟต์แวร์

- **ความเสี่ยงด้านทรัพยากร**
  - ขนาดทีมลด
  - ลดงบประมาณ
  - เครื่องมือ/ลิขสิทธิ์ไม่พร้อม
  - Infrastructure ขัดข้อง

  ตัวอย่าง
    - ความเสี่ยง: "CI/CD Server ขัดข้อง"
    - ความเป็นไปได้: ปานกลาง (25%)
    - ผลกระทบ: ปานกลาง (ล่าช้า 1–2 วัน)
    - การบรรเทา: Infrastructure สำรอง + Monitoring

หมวดหมู่ความเสี่ยงในโครงการซอฟต์แวร์

- **ความเสี่ยงภายนอก**
  - Client เปลี่ยนความต้องการ
  - การเปลี่ยนแปลงกฎระเบียบ (GDPR เป็นต้น)
  - ตลาดเปลี่ยนแปลง
  - ปัญหาจาก Vendor

  ตัวอย่าง
    - ความเสี่ยง: "Client ต้องการ Feature ใหม่ระหว่าง Sprint"
    - ความเป็นไปได้: สูง (70%)
    - ผลกระทบ: ปานกลาง (Scope Creep)
    - การบรรเทา: Change Control Process + PO ควบคุม

หมวดหมู่ความเสี่ยงในโครงการซอฟต์แวร์

- **ความเสี่ยงองค์กร**
  - Manager เปลี่ยน
  - องค์กร Reorganize
  - Priority เปลี่ยน
  - ปัญหาวัฒนธรรม

  ตัวอย่าง
  - ความเสี่ยง: "องค์กรเปลี่ยนโปรแกรมธุรกิจ"
  - ความเป็นไปได้: ต่ำ (10%)
  - ผลกระทบ: สูงมาก (ยกเลิกโครงการ)
  - การบรรเทา: –

Risk Management Process

Step 1: Identify

- Brainstorm กับทีม
- Review ประสบการณ์ projects เดิม

→ Output: Risk List (10-20 items)

Step 2: Analyze

- ประเมิน Probability (Low/Med/High)
- ประเมิน Impact (Low/Med/High)
- คำนวณ Risk Score

→ Output: Risk Matrix

Step 3: Prioritize

- โฟกัสกับ High Risk เท่านั้น
- Ignore Low Risk (accept)

→ Output: Top 5-10 Risks

Step 4: Mitigation Planning

- สร้าง Action Plan ต่อ High Risk
- Assign Owner
- Set Deadline

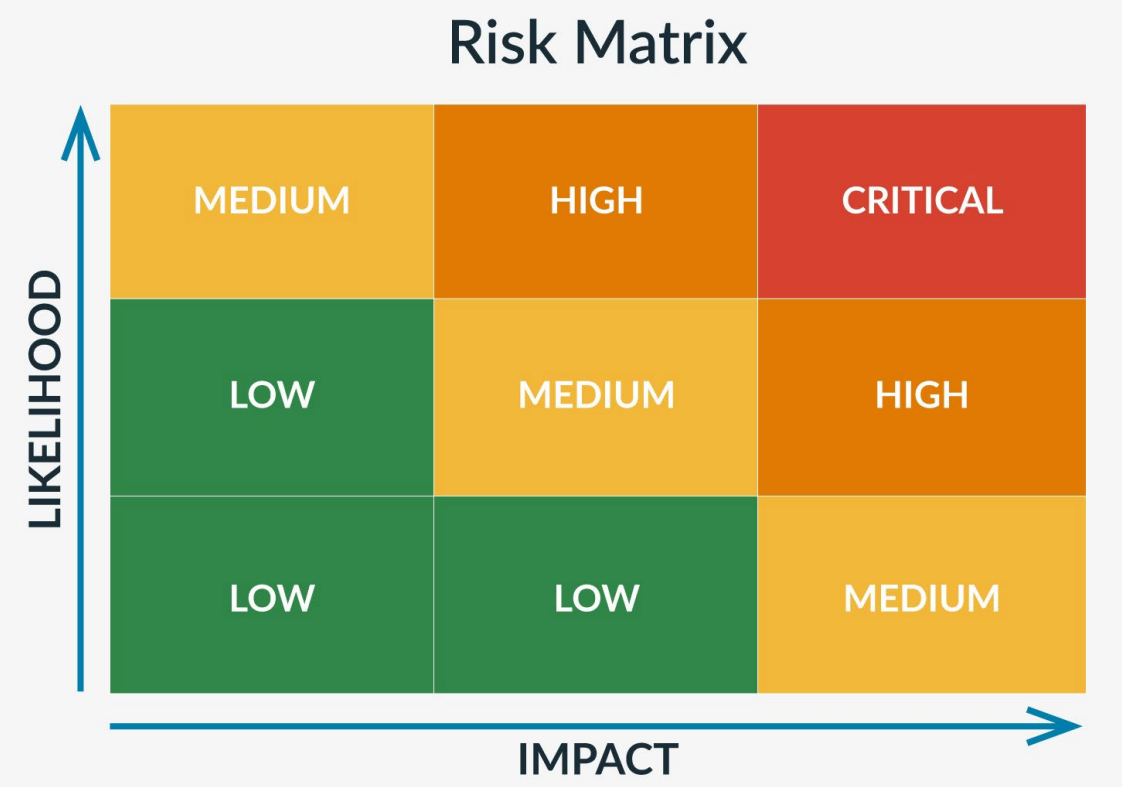
→ Output: Mitigation Plan

Step 5: Monitor & Control

- Review risk status every Sprint - Trigger mitigation if risk occurs - Update risk register

→ Output: Risk Status Report

Risk Matrix & การจัดลำดับความสำคัญ



- การดำเนินการ
  - CRITICAL (แดง): ต้องบรรเทาก่อนเริ่ม
  - HIGH (ส้ม): สร้าง mitigation plan
  - MEDIUM (เหลือง): ติดตามอย่างใกล้ชิด
  - LOW (เขียว): ยอมรับและบันทึก

Mitigation Strategies

4 เทคนิค

1. **Avoid (หลีกเลี่ยง)** เลี่ยงกิจกรรมที่มีความเสี่ยง
    - Risk: Vendor lock-in with AWS
    - Mitigation: Use Kubernetes (portable)
    - Result: Avoid lock-in risk entirely
2. **Reduce (ลดการเสี่ยง)** ลดความน่าจะเป็นหรือผลกระทบ
    - Risk: Performance issues
    - Mitigation: Load test early (reduce Impact)
    - Result: Catch performance early
3. **Transfer (โยกย้าย)** โยกความเสี่ยงให้คนอื่น
    - Risk: Server downtime
    - Mitigation: Buy insurance / Outsource to cloud provider
    - Result: Provider assume risk
4. **Accept (ยอมรับ)** ยอมรับและมี Plan B
    - Risk: Requirement change (high prob, medium impact)
    - Mitigation: Change control process + buffer time
    - Result: Handle as it comes

Risk Register Template

ID	Risk Description	Probability	Impact	Score	Owner	Mitigation	Status
R1	Tech stack incomp.	High (60%)	High	RED	Lead Dev	Evaluate stacks by Week 2	Open
R2	Resource shortage	Med (30%)	Very High	RED	SM	Hire contractor by Week 1	Open
R3	Requirement change	High (70%)	Med	ORANGE	PO	Change control process	Open
R4	API integration fail	Med (40%)	High	ORANGE	Dev	Mock APIs early	Open
R5	Testing behind	Med (35%)	Med	YELLOW	QA	Test framework by Week 3	Open

Stakeholder Management

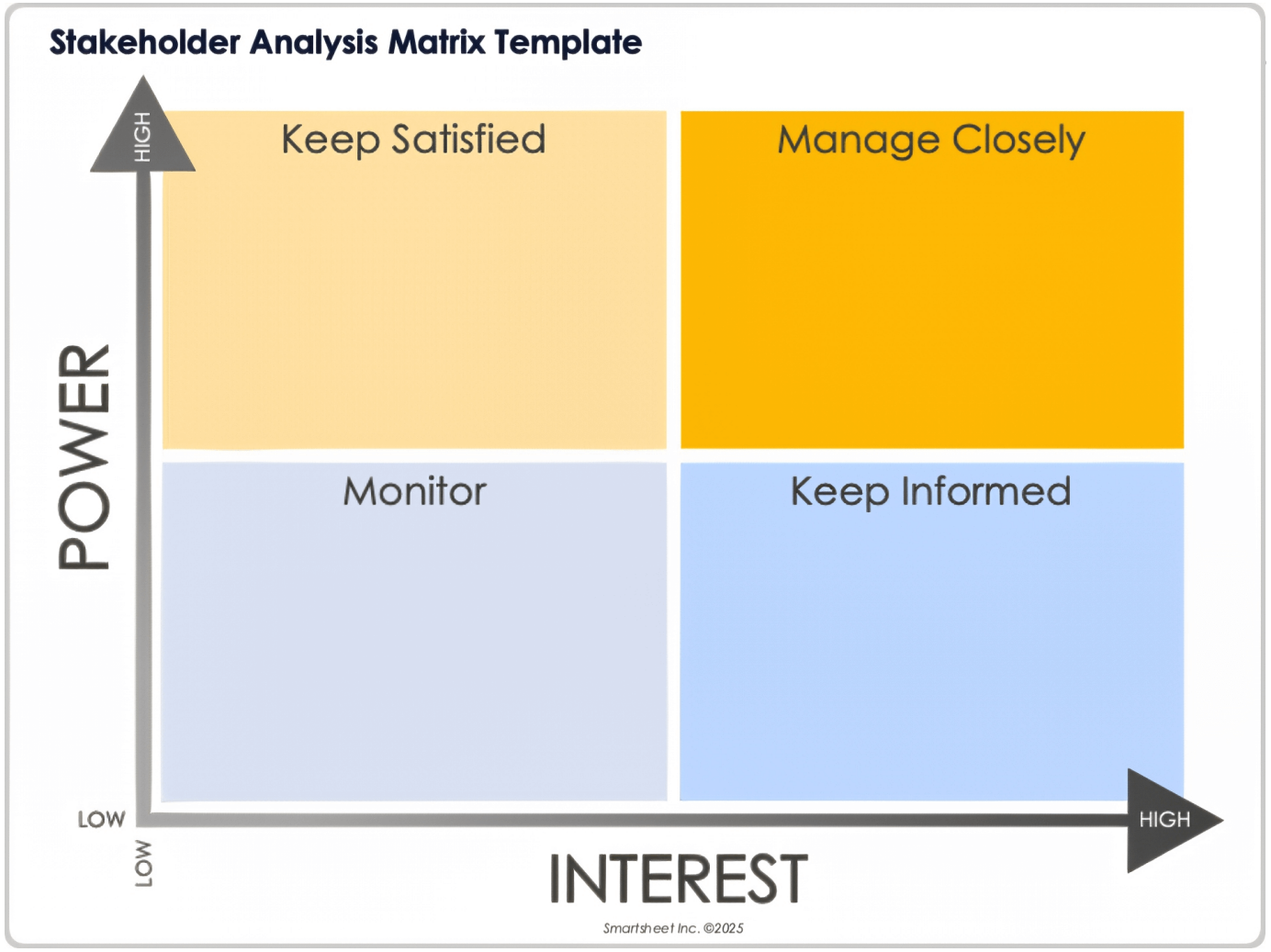
- **Stakeholder คืออะไร**

นิยาม: ทุกคนที่ได้รับผลกระทบจากโครงการ

- Stakeholder โดยตรง
  - Client/Product Owner
  - ผู้ใช้งาน
  - Development Team
  - Project Manager

- Stakeholder ทางอ้อม
  - CEO บริษัท
  - ทีม Support
  - ทีม Operations
  - ทีมคู่แข่ง

Stakeholder Analysis Matrix



การดำเนินการ

- รักษาความสุข: ตอบสนองความต้องการ สื่อสารดี
- จัดการใกล้ชิด: อัปเดตสม่ำเสมอ เกี่ยวข้องในการตัดสินใจ
- ติดตาม: สังเกต การเปลี่ยนแปลงของความสนใจ/อำนาจ
- แจ้งข่าว: อัปเดตเป็นครั้งคราว ฟังความกังวล

กลยุทธ์การสื่อสารต่อ Stakeholder

- **Client (อำนาจสูง ความสนใจสูง)**
  - การสื่อสาร: การประชุมรายสัปดาห์
  - ความถี่อัปเดต: ทุก Sprint
  - ระดับรายละเอียด: Executive Summary + Demo
  - รูปแบบ: พบเห็นตัวจริง หรือ Video Call
  - เวลาตอบกลับ: 24 ชั่วโมง
- **ผู้ใช้งาน (อำนาจต่ำ ความสนใจสูง)**
  - การสื่อสาร: แบบสำรวจรายเดือน / สัมภาษณ์รายไตรมาส
  - ความถี่อัปเดต: ทุก 2 Sprints
  - ระดับรายละเอียด: Demo Feature เท่านั้น
  - รูปแบบ: Prototype / Mockup
  - เวลาตอบกลับ: 1 สัปดาห์

กลยุทธ์การสื่อสารต่อ Stakeholder

- **ทีมภายใน (อำนาจสูง ความสนใจสูง)**
  - การสื่อสาร: Daily Standup + Sprint Events
  - ความถี่อัปเดต: ทุกวัน
  - ระดับรายละเอียด: Technical Deep Dive
  - รูปแบบ: พบตัวจริง + Slack
  - เวลาตอบกลับ: 1 ชั่วโมง
- **Vendor (ผันแปร)**
  - การสื่อสาร: ตามความจำเป็น
  - ความถี่อัปเดต: ตามสัญญา
  - ระดับรายละเอียด: ผลกระทบทางธุรกิจ
  - รูปแบบ: Email + การโทรเป็นระยะ
  - เวลาตอบกลับ: ตาม SLA

การจัดการความคาดหวัง

- **ปัญหา:**
  - ความจริง: 100 Features ใน 16 สัปดาห์
  - ความคาดหวัง: 150 Features ใน 16 สัปดาห์
  - ผลลัพธ์: Client ผิดหวัง

วิธีจัดการ:

1.ตั้งความคาดหวังที่สมจริงตั้งแต่ต้น

1. "เราทำได้ 100 Features"
2. ไม่ใช่ "ลองทำ 150 Features นะ"

2.สื่อสารเร็วและบ่อยๆ

1. Demo รายสัปดาห์
2. Backlog โปร่งใส
3. แจ้ง Risk

3.จัดการ Scope

1. Change Control
2. Priority ชัดเจน
3. บอก "ไม่ได้" กับคำขอนอก Scope

4.ส่งมอบคุณค่าแบบค่อยเป็นค่อยไป

1. MVP ก่อน
2. Release เร็ว
3. ได้ Feedback รวดเร็ว

กิจกรรม#1 – **Project Charter**

- **วัตถุประสงค์:** สร้างเอกสาร Project Charter สำหรับโครงการของตัวเอง

รายละเอียดกิจกรรม

ตัวอย่าง

กิจกรรม#2 – **ประเมินความเสี่ยง & วางแผนบรรเทา**

- **วัตถุประสงค์:** ระบุและวางแผนบรรเทาความเสี่ยงของโครงการ

รายละเอียดกิจกรรม

ตัวอย่าง