สัปดาห์ที่ 3: Test Planning, Test Strategy และ Exploratory Testing

---

วัตถุประสงค์การเรียนรู้ (Learning Objectives)

นิสิตจะสามารถ

- วางแผนการทดสอบอย่างเป็นระบบ
- เข้าใจความแตกต่างระหว่าง Test Strategy และ Test Plan
- ทำ Risk Assessment ประเมินเวลาในการทดสอบ และเริ่มต้นใช้ Exploratory Testing

---

ส่วนที่ 1: Test Planning Process (กระบวนการวางแผนการทดสอบ)

1.1 Test Planning คืออะไร

คำจำกัดความ:

> Test Planning (การวางแผนการทดสอบ) คือกระบวนการกำหนดวัตถุประสงค์ ขอบเขต แนวทาง ทรัพยากรและกำหนดการสำหรับกิจกรรมการทดสอบทั้งหมด

เปรียบเทียบกับชีวิตจริง

- การวางแผนการทดสอบ = การวางแผนการเดินทาง
    - ต้องรู้จุดหมาย (Test Objectives - วัตถุประสงค์)
    - ต้องรู้งบประมาณ (Resources - ทรัพยากร)
    - ต้องรู้เวลา (Schedule - กำหนดการ)
    - ต้องรู้อุปสรรค (Risks - ความเสี่ยง)

1.2 ทำไมต้องมี Test Plan

ประโยชน์

1. **ความชัดเจน** - ทุกคนเข้าใจเป้าหมายเดียวกัน
2. **การจัดการ** - ควบคุมเวลา ทรัพยากร งบประมาณ
3. **การสื่อสาร** - เอกสารกลางสำหรับทีม stakeholders (ผู้เกี่ยวข้อง)
4. **ความต่อเนื่อง** - คนใหม่สามารถเข้ามาทำงานต่อได้
5. **การตัดสินใจ** - มีข้อมูลสำหรับตัดสินใจ

สถิติจริง

- โปรเจกต์ที่มี Test Plan มีโอกาส 40% ที่จะส่งมอบตรงเวลา
- โปรเจกต์ที่ไม่มี Test Plan มักพบ bug ใน Production มากกว่า 2-3 เท่า

1.3 กระบวนการวางแผน (Test Planning Process)

1. วิเคราะห์ข้อกำหนด (Analyze Requirements)

↓

2. กำหนดขอบเขตการทดสอบ (Define Test Scope)

↓

3. ระบุกลยุทธ์การทดสอบ (Identify Test Strategy)

↓

4. กำหนดวัตถุประสงค์ (Define Test Objectives)

↓

5. กำหนดเกณฑ์เข้า/ออก (Set Test Criteria Entry/Exit)

↓

6. วางแผนทรัพยากร (Resource Planning)

↓

7. ประวัตินำการทดสอบ (Schedule Test Activities)

↓

8. ระบุความเสี่ยง (Identify Risks)

↓

9. จัดเอกสารแผน (Document Test Plan)

1.4 องค์ประกอบของ Test Plan (IEEE 829)

1. ตัวระบุแผน (Test Plan Identifier)

- ชื่อและเวอร์ชันของ Test Plan

2. บทนำ (Introduction)

- พื้นหลังโครงการ (Project background)
- จุดประสงค์ของการทดสอบ (Purpose of testing)
- ขอบเขต (Scope)

3. รายการทดสอบ (Test Items)

- รายการสิ่งที่จะทดสอบ
- Features/Modules ที่อยู่ในขอบเขต

4. ฟีเจอร์ที่จะทดสอบ (Features to be Tested)

- ลงทะเบียนและเข้าสู่ระบบ (User Registration & Login)
- การจัดการหนังสือ (Book Management CRUD)
- การค้นหาและกรองหนังสือ (Book Search & Filter)
- ระบบการยืม (Borrowing System)
- ระบบการคืน (Return System)
- การจัดการโปรไฟล์ (User Profile Management)

5. ฟีเจอร์ที่ไม่ทดสอบ (Features NOT to be Tested)

- แดชบอร์ดแอดมิน (Admin Dashboard Phase 2)
- การแจ้งเตือนทางอีเมล (Email Notifications External Service)
- ระบบชำระเงิน (Payment Gateway Not in scope)

6. แนวทาง/กลยุทธ์ (Test Approach/Strategy)

- แนวทางการทดสอบด้วยตนเอง (Manual testing approach)

- แนวทางอัตโนมัติ (Automation approach)
- ระดับการทดสอบ (Test levels Unit Integration System UAT)

7. เกณฑ์ผ่าน/ไม่ผ่าน (Pass/Fail Criteria)

- เกณฑ์เข้า (Entry Criteria)
- เกณฑ์ออก (Exit Criteria)
- เกณฑ์หยุด (Suspension Criteria)

8. ผลผลิตการทดสอบ (Test Deliverables)

- Test Plan document
- กรณีทดสอบ (Test Cases)
- สคริปต์การทดสอบ (Test Scripts)
- ข้อมูลทดสอบ (Test Data)
- รายงานข้อบกพร่อง (Bug Reports)
- รายงานสรุปการทดสอบ (Test Summary Report)

9. สภาพแวดล้อมการทดสอบ (Test Environment)

- ข้อกำหนดฮารดแวร์ (Hardware requirements)
- ข้อกำหนดซอฟต์แวร์ (Software requirements)
- การกำหนดค่าเครือข่าย (Network configuration)
- ข้อกำหนดข้อมูลทดสอบ (Test data requirements)

10. กำหนดการ (Schedule)

- Test phases และกำหนดเวลา
- เหตุการณ์สำคัญ (Milestones)

11. การจัดสรรบุคลากร (Staffing and Training)

- Test team members
- บทบาทและความรับผิดชอบ (Roles & responsibilities)
- ความต้องการด้านการฝึกอบรม (Training needs)

12. ความเสี่ยงและแผนสำรอง (Risks and Contingencies)

- ความเสี่ยงที่ระบุ (Identified risks)
- แผนบรรเทา (Mitigation plans)

13. การอนุมัติ (Approvals)

- การลงนาม (Sign-off from stakeholders)

ส่วนที่ 2: Test Strategy vs Test Plan

2.1 Test Strategy คืออะไร

คำจำกัดความ:

Test Strategy (กลยุทธ์การทดสอบ) คือ high-level approach ที่กำหนดว่าจะทดสอบอย่างไร โดยครอบคลุมทั้งองค์กรหรือหลาย ๆ โปรเจกต์

ตัวอย่าง Test Strategy องค์กร

- "เราจะใช้ Agile Testing approach"
- "เราจะ automate 80% ของ การทดสอบถดถอย (regression tests)"
- "เราจะทำ การทดสอบการรวมอย่างต่อเนื่อง (continuous integration testing)"

2.2 Test Plan คืออะไร

คำจำกัดความ

แผนการทดสอบ คือ แผนโดยละเอียดสำหรับโปรเจกต์เฉพาะ ที่ระบุรายละเอียดว่าจะทดสอบอะไร เมื่อไหร่ อย่างไร

2.3 ความแตกต่าง

Aspect	Test Strategy	Test Plan
Scope	องค์กร/หลายโครงการ	Single project (โครงการเดี่ยว)
Level	ระดับสูงตามแนวคิด	Detailed, tactical (รายละเอียด)
Timeline	ระยะยาว (ปี)	Short-term (ระยะเวลาโครงการ)
Owner	ผู้จัดการฝ่ายทดสอบ/ผู้อำนวยการฝ่าย QA	หัวหน้าทดสอบ/ผู้จัดการโครงการ
Content	แนวทางหลักการทั่วไป	กิจกรรมเฉพาะ กำหนดการ
Changes	ไม่ค่อยมีการเปลี่ยนแปลง	อาจมีการเปลี่ยนแปลงระหว่างโครงการ
Example	ใช้การทดสอบตามความเสี่ยง	ทดสอบการชำระเงินด้วยกรณีทดสอบ 50 กรณี

2.4 ตัวอย่างเปรียบเทียบ: Library Management System

กลยุทธ์การทดสอบ (ระดับองค์กร)

องค์กรของเราปฏิบัติตามหลักการทดสอบเหล่านี้

1. ทดสอบเร็ว (Shift-left testing approach)
2. อัตโนมัติ 70% การทดสอบด้วยตนเอง 30%
3. จัดลำดับตามความเสี่ยง (Risk-based prioritization)
4. การทดสอบอย่างต่อเนื่องในไปป์ไลน์ CI / CD
5. คุณสมบัติที่สำคัญทั้งหมดต้องมีความครอบคลุม 90%+

แผนการทดสอบ (โครงการ LMS)

แผนการทดสอบ Library Management System:

1. ทดสอบการลงทะเบียนผู้ใช้ด้วย 25 กรณีทดสอบ
2. ยืมหนังสือทดสอบ 30 กรณีทดสอบ
3. กำหนดการ: สัปดาห์ที่ 4-6
4. ทีม: ผู้ทดสอบ 3 คน
5. เครื่องมือ: ล้อเล่น, นักเขียนบทละคร, supertest
6. เป้าหมายความครอบคลุม: 85%

การทำงานร่วมกัน

กลยุทธ์การทดสอบ (วิธีที่เราทดสอบโดยทั่วไป)

↓

แผนการทดสอบ (สิ่งที่เราทดสอบในโครงการนี้)

↓

กรณีทดสอบ (ขั้นตอนโดยละเอียด)

ส่วนที่ 3: Risk-Based Testing (การทดสอบที่ขึ้นอยู่กับความเสี่ยง)

3.1 Risk คืออะไร

คำจำกัดความ

ความเสี่ยง (Risk) = โอกาสที่จะเกิด (Probability) × ผลกระทบ (Impact)

สูตร

Risk Priority = Probability × Impact

ซึ่ง:

- ต่ำถึงสูง (Probability: 1 Low to 5 High)
- Impact: 1 (Low) to 5 (High)
- Risk Priority: 1-25

3.2 ประเภทของ Risk

1. ความเสี่ยงของผลิตภัณฑ์ (Product Risks)

- ช่องโหว่ความปลอดภัย (Security vulnerabilities)
- ปัญหาประสิทธิภาพ (Performance issues)
- สูญหายข้อมูล (Data loss)
- การคำนวณผิด (Incorrect calculations)
- ประสบการณ์ผู้ใช้แย่ (Poor user experience)

2. ความเสี่ยงของโครงการ (Project Risks)

- ล่าช้ากำหนดการ (Schedule delays)
- ขาดแคลนทรัพยากร (Resource shortage)
- เปลี่ยนข้อกำหนด (Requirement changes)
- เครื่องมือใช้ไม่ได้ (Tool failures)
- สมาชิกทีมลาออก (Team turnover)

3.3 กระบวนการประเมินความเสี่ยง (Risk Assessment Process)

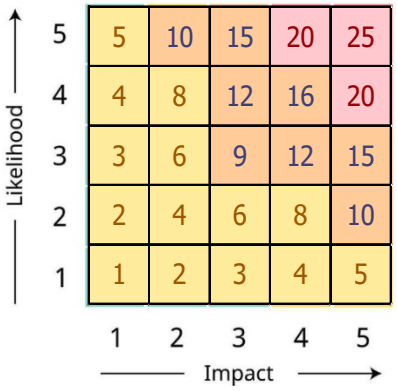
1. ระบุความเสี่ยง (Risk Identification)

- การระดมสมอง (Brainstorming sessions)
- ข้อมูลฐานในอดีต (Historical data analysis)
- สัมภาษณ์ผู้เกี่ยวข้อง (Stakeholder interviews)
- ตรวจสอบข้อกำหนด (Requirements review)

2. วิเคราะห์ความเสี่ยง (Risk Analysis)

สำหรับแต่ละความเสี่ยง

1. อัตราความน่าจะเป็น (1-5)
2. อัตราผลกระทบ (1-5)
3. คำนวณลำดับความสำคัญของความเสี่ยง = ความน่าจะเป็น×ผลกระทบ
4. จำแนกประเภท: ต่ำ (1-8), ปานกลาง (9-16), สูง (17-25)



3. บรรเทาความเสี่ยง (Risk Mitigation)

- High Risk ⟶ ตรวจอย่างละเอียด เร็ว และบ่อยครั้ง
- Medium Risk ⟶ วิธีการทดสอบมาตรฐาน
- Low Risk ⟶ การทดสอบหรือเลื่อนออกไปน้อยที่สุด

3.4 ตัวอย่าง: Risk Assessment สำหรับ Library Management System

Feature: User Login

ความเสี่ยง	ความน่าจะเป็น	ผลกระทบ	ลำดับความสำคัญ	ความรุนแรง	การบรรเทาผลกระทบ
SQL Injection ช่องโหว่	3	5	15	High	Security testing, parameterized queries
Password เก็บไว้ในข้อความธรรมดา	2	5	10	Medium	Code review, use bcrypt
Session timeout นานเกินไป	3	3	9	Medium	Configuration testing
Remember me ไม่ทำงาน	2	2	4	Low	Basic functional testing

Feature: Book Borrowing

Risk	Probability	Impact	Priority	Severity	Mitigation
ผู้ใช้สามารถยืมหนังสือได้ไม่จำกัด	4	4	16	High	การทดสอบตรรกะทางธุรกิจ
จำนวนหนังสืออัปเดตไม่ถูกต้อง	4	4	16	High	Integration testing
การคำนวณวันครบกำหนดไม่ถูกต้อง	3	4	12	Medium	Unit testing
การแจ้งเตือนทางอีเมลล้มเหลว	2	2	4	Low	Monitor logs

การจัดลำดับตามความเสี่ยง (Test Prioritization based on Risk):

Priority 1 (High Risk - Test First):

- SQL Injection testing
- Book borrowing limits
- Book inventory accuracy

Priority 2 (Medium Risk - Test Second):

- Password encryption
- Session management
- Due date calculations

Priority 3 (Low Risk - Test Last):

- Remember me functionality
- Email notifications

3.5 การออกแบบการทดสอบตามความเสี่ยง

High-Risk Features:

- กรณีทดสอบเพิ่มเติม
- การทดสอบที่ลึกขึ้น (edge cases, negative scenarios)
- การทดสอบเบื้องต้น
- การทดสอบซ้ำบ่อย (Frequent regression testing)
- การทดสอบหลายระดับ (unit, integration, system)

Low-Risk Features:

- Fewer test cases
- Happy path focus
- Late testing
- การทดสอบซ้ำเป็นครั้งคราว
- ความครอบคลุมของการทดสอบพื้นฐาน

ส่วนที่ 4: เทคนิคการประมาณการทดสอบ (Test Estimation Techniques)

4.1 ทำไมต้อง Estimate

เหตุผล:

1. การจัดสรรทรัพยากร (Resource allocation)
2. วางแผนกำหนดการ (Schedule planning)
3. วางแผนงบประมาณ (Budget planning)
4. การจัดการความเสี่ยง (Risk management)
5. คาดหวังของผู้เกี่ยวข้อง (Stakeholder expectations)

4.2 เทคนิคการประมาณการทดสอบ

1. Function Point Analysis (FPA)

แนวคิด: วิธีการวัดขนาดของระบบสารสนเทศตามฟังก์ชันการทำงานที่ผู้ใช้ต้องการ โดยไม่สนใจภาษาโปรแกรม ทำให้สามารถประเมินขนาด ความซับซ้อน และประมาณการค่าใช้จ่าย/ทรัพยากรสำหรับโครงการซอฟต์แวร์ได้อย่างเป็นรูปธรรม

- ประมาณขนาดของ software โดยนับ functions
- แปลง functions เป็น effort (ความพยายาม)

สูตร

Estimated Effort = Number of Function Points × Productivity Rate

ซึ่ง:

- Function Points = Inputs + Outputs + Queries + Files + Interfaces
- Productivity Rate = ชั่วโมงต่อ Function Point (industry average อยู่ที่ประมาณ 5-15 hours)

ตัวอย่าง: Library Management System

Function Points Calculation

Inputs (User Inputs):

- User Registration form: 1 FP
- Login form: 1 FP
- Book Search form: 1 FP
- Borrow Book action: 1 FP

Total Inputs: 4 FP

Outputs (Reports/Displays)

- User Dashboard: 2 FP
- Book List: 2 FP
- Borrowing History: 2 FP
- Search Results: 2 FP

Total Outputs: 8 FP

Queries (Database Queries)

- Find User: 1 FP
- Find Book: 1 FP
- Check Availability: 1 FP

Total Queries: 3 FP

Files (Data Tables):

- Users table: 1 FP
- Books table: 1 FP
- Borrowings table: 1 FP

Total Files: 3 FP

External Interfaces (ส่วนติดต่อภายนอก):

- Email service: 1 FP

Total Interfaces: 1 FP

Total Function Points = 4 + 8 + 3 + 3 + 1 = 19 FP

Testing Effort = 19 FP × 8 hours/FP = 152 hours

2. Use Case Point (UCP)

Concept: เทคนิคการประมาณขนาดและค่าใช้จ่ายของซอฟต์แวร์ โดยใช้วิเคราะห์จากแผนภาพ UML Use Case (ยูสเคสไดอะแกรม) ซึ่งจะประเมินจากจำนวนและความซับซ้อนของ "ยูสเคส" และ "แอคเตอร์" (ผู้ใช้งาน) แล้วนำมาปรับด้วยปัจจัยความซับซ้อนทางเทคนิค (TCF) และปัจจัยสภาพแวดล้อม (ECF) เพื่อให้ได้ค่าประมาณการขนาดซอฟต์แวร์ที่แม่นยำขึ้น เพื่อใช้ในการวางแผนโครงการพัฒนา

- ประมาณจาก use cases
- นับ actors และ transactions

Formula:

UCP = (UAW + UUCW) × TCF × EF

ซึ่ง:

- UAW = Unadjusted Actor Weight ขนาดซอฟต์แวร์ขึ้นกับจำนวนและความซับซ้อนของแอคเตอร์ยังไม่ได้ปรับ
- UUCW = Unadjusted Use Case Weight ขนาดของซอฟต์แวร์ขึ้นกับจำนวนและความซับซ้อนของยูสเคสยังไม่ได้ปรับ
- TCF = Technical Complexity Factor ปัจจัยที่ถูกใช้ในการปรับแต่งขนาดของซอฟต์แวร์โดยพิจารณาจากข้อมูลทางด้านเทคนิค
- EF = Environmental Factor) ปัจจัยที่ถูกใช้ในการปรับแต่งขนาดของซอฟต์แวร์โดยพิจารณาจากสภาพแวดล้อมในการพัฒนาระบบ

Testing Effort = UCP × Productivity (20-40 hours/UCP)

>>> Continue reading

3. Expert Judgment (การตัดสินใจของผู้เชี่ยวชาญ)

Concept:

- Break down ทำงานเป็น tasks เล็ก ๆ
- Estimate แต่ละ task จากประสบการณ์

Process:

1. Break down features
2. Identify test activities for each
3. Estimate each activity
4. Add buffer (20-30%)

ตัวอย่าง: User Login Feature

Activity	Estimated Hours
Test planning & design	4
Test case writing (20 cases)	10
Test environment setup	2
Test execution (1st round)	8
Bug reporting	3
Regression testing (2nd round)	6
Test report	2
Subtotal	**35**
Buffer (20%)	**7**
Total	**42 hours**

4. Three-Point Estimation (PERT) - ประมาณการ 3 จุด

Concept: เทคนิคการประมาณการสามจุด เป็นวิธีการประเมินเวลาหรือต้นทุนของกิจกรรมในโครงการ โดยใช้ค่าประมาณ 3 ค่า คือ เวลาที่ดีที่สุด (Optimistic - O), เวลาที่น่าจะเป็นไปได้มากที่สุด (Most Likely - M), และ เวลาที่แย่ที่สุด (Pessimistic - P) เพื่อคำนวณหา เวลาคาดการณ์ (Expected Time - E) ที่แม่นยำยิ่งขึ้น โดยให้ความสำคัญกับเวลาที่น่าจะเป็นไปได้มากที่สุด (M) มากกว่าค่าอื่นๆ เพื่อจัดการกับความไม่แน่นอนในโครงการและสร้างแผนที่สมจริงยิ่งขึ้น

Optimistic (O): สถานการณ์ที่ดีที่สุด ทุกอย่างเป็นไปอย่างราบรื่น งานเสร็จเร็วที่สุด

Most Likely (M): สถานการณ์ปกติที่น่าจะเป็นไปได้มากที่สุด ภายใต้เงื่อนไขการทำงานทั่วไป

Pessimistic (P): สถานการณ์ที่แย่ที่สุด ปัญหาอุปสรรคเกิดขึ้น ทำให้งานเสร็จช้าที่สุด

- Calculate weighted average

Formula:

$$\text{Expected Effort} = \frac{(\text{Optimistic} + 4\times\text{Most Likely} + \text{Pessimistic})}{6}$$

ตัวอย่าง: Book Search Testing

Optimistic (Best case): 20 hours

Most Likely (Normal case): 30 hours

Pessimistic (Worst case): 50 hours

Expected = (20 + 4×30 + 50) / 6

= (20 + 120 + 50) / 6

= 190 / 6

= 31.7 hours ≈ 32 hours

4.3 Comparison & Recommendations

Technique	Accuracy	Effort	When to Use
Function Point	Medium-High	High	Large projects, formal estimation
Use Case Point	Medium	Medium	When use cases available
Expert Judgment	Medium	Low	Quick estimates, experienced team
Three-Point	Medium-High	Low	Uncertainty exists

Best Practice:

- Use **multiple techniques** และ compare
- Start with **Expert Judgment** สำหรับ quick estimate
- Use **FPA/UCP** สำหรับ formal estimation
- Add **buffer** 20-30% สำหรับ สิ่งที่ไม่รู้ (unknowns)

ส่วนที่ 5: Entry และ Exit Criteria

5.1 Entry Criteria คืออะไร

คำจำกัดความ

> Entry Criteria = เงื่อนไขที่ต้องพร้อมก่อนจะเริ่มทดสอบได้ == Checklist ก่อนเริ่มทดสอบ

เปรียบเทียบ

- เหมือนการตรวจสอบก่อนขึ้นเครื่องบิน (passport, ticket, security check)

5.2 ตัวอย่าง Entry Criteria: Library Management System

General Entry Criteria:

[ ] เอกสารข้อกำหนดได้รับการอนุมัติ และ ไม่เปลี่ยนแปลงอีก

[ ] แผนการทดสอบได้รับการอนุมัติ โดย ผู้มีส่วนได้ส่วนเสีย - Manager, Product Owner, Business Analyst ลงนามแล้ว

[ ] Test environment พร้อมใช้งาน (Dev, QA environments) - โปรแกรมทดสอบ (servers, machines) ตั้งค่าเสร็จแล้ว Computers, Databases, Networks ทำงานได้

[ ] เตรียมข้อมูลทดสอบเสร็จแล้ว ข้อมูล (Users, Books, Borrowing records) ใส่ลงฐานข้อมูลแล้ว

[ ] Build deployed ไปยัง test environment สำเร็จ - โค้ดของแอป (Library app) ปล่อยไปบนเซิร์ฟเวอร์ทดสอบแล้ว, Deployment (นำมาใช้) เสร็จเรียบร้อย

[ ] Smoke test passed (ระบบขึ้นได้) - ทดสอบอย่างรวดเร็ว เพื่อให้แน่ใจว่าระบบทำงานพื้นฐาน ฟีเจอร์หลัก (login, search, borrow) ทำงานปกติ

[ ] Test team trained บน application - ทีม QA ได้รับการอบรมเรียนรู้เกี่ยวกับแอปแล้ว, ทีมรู้วิธีใช้ app, วิธีทดสอบ, workflow

[ ] กำหนดค่าเครื่องมือทดสอบ (Jest, Playwright, etc.) - เครื่องมือทดสอบ (testing tools) ติดตั้งและตั้งค่าแล้ว, Jest, Playwright, Postman, k6 พร้อมใช้

ตัวอย่าง

Checklist ก่อนเริ่มทดสอบ

❌ Requirements ยังปรับปรุง ⟶ STOP, รอให้นิ่ง

❌ Test Plan ยังไม่ได้อนุมัติ ⟶ STOP, ส่งให้ Manager

❌ Server ยังไม่ setup ⟶ STOP, รอ IT

❌ Database ว่างเปล่า ⟶ STOP, ใส่ test data

❌ Build deploy ผิด ⟶ STOP, ติดต่อ Dev

❌ Login crash ⟶ STOP, ให้ Dev แก้ก่อน (Smoke test fail)

❌ QA ไม่รู้ app ⟶ STOP, ให้ demo ก่อน

❌ Jest ติดตั้งไม่ได้ ⟶ STOP, หาคน setup

ทั้ง 8 ข้อ เสร็จแล้ว ⟶ เริ่มทดสอบได้เลย!

5.3 Exit Criteria คืออะไร

คำจำกัดความ:=

Exit Criteria (เกณฑ์ออก) = เงื่อนไขที่บอกว่าทดสอบเสร็จแล้ว พร้อม release == Checklist เสร็จสิ้นการทดสอบ

เปรียบเทียบ

- เหมือนเกณฑ์ผ่านการสอบ (ต้องได้คะแนน 60% ขึ้นไป)

5.4 ตัวอย่าง Exit Criteria: Library Management System

General Exit Criteria

[ ] All planned test cases executed (100%) - ทดสอบทั้งหมด 100% ของ test cases ที่วางแผนไว้, ไม่มีข้อใดที่ยังไม่ได้ทดสอบ

[ ] Test pass rate ≥ 95%

ตัวอย่าง:

- Total Test Cases: 150
- Passed: 145 (96.67%)
- Failed: 5 (3.33%)

Pass Rate = 145/150 = 96.67% ≥ 95% ✓

[ ] **No** High/Critical severity bugs open - **ไม่มี** Critical bugs ค้างอยู่ (ระบบขัดข้อง), ไม่มี High severity bugs ค้างอยู่ (ฟีเจอร์สำคัญใช้ไม่ได้)

[ ] Medium bugs < 5 open - ปัญหา Medium severity < 5 bugs ที่ยังค้างอยู่ อาจยอมรับได้ว่า "ดี พอที่จะ Release"

[ ] Code coverage ≥ 85% - โค้ดที่ถูกทดสอบ ≥ 85% Automated test ครอบคลุม 85% ของโค้ด

[ ] Performance benchmarks met (เป้าหมายประสิทธิภาพ)

Response time < 2 seconds

Concurrent users: 100+

[ ] Security scan passed (no Critical/High vulnerabilities) - ไม่มี Critical vulnerability (ช่องโหว่อันตราย), ไม่มี High vulnerability, ผ่าน Security test (OWASP ZAP, SonarQube)

[ ] All test deliverables completed - ส่งเอกสารทดสอบครบ

Test Summary Report

Defect Report

Traceability Matrix

[ ] Sign-off จาก Test Manager และ Product Owner - Test Manager ลงนาม (ทดสอบเสร็จ, พร้อม Release), Product Owner ลงนาม (ตรงตามข้อกำหนด)

5.5 Suspension Criteria (เกณฑ์หยุด)

คำจำกัดความ:

```
Suspension Criteria = เงื่อนไขที่ต้องหยุดการทดสอบชั่วคราว
```

ตัวอย่าง:

- [ ] Build มี show-stopper bug (ระบบ crash ทันที) - Build (เวอร์ชันโค้ด) มี bug ร้ายแรง, ระบบ crash หรือไม่สามารถใช้งาน, ไม่สามารถทดสอบต่ออื่น ๆ ได้
- [ ] Test environment down > 4 hours - Test environment (server, database, network) down, Down นาน > 4 ชั่วโมง, ไม่สามารถเข้าถึงระบบทดสอบ
- [ ] > 30% test cases fail ใน smoke test - Smoke test (ทดสอบอย่างรวดเร็ว) fail > 30%, ตรวจพบ 30% features ไม่ทำงาน
- [ ] Critical functionality ใช้งานไม่ได้ - Features หลัก (critical) ใช้ไม่ได้, สามารถทำงานต่อไม่ได้, ความล้มเหลว core feature
- [ ] Database corruption (ข้อมูลเสีย) - ข้อมูล database เสียหรือ corrupt, ไม่สามารถเชื่อถือข้อมูลทดสอบ, Test data unreliable
- [ ] Major requirement change - Requirement (ข้อกำหนด) เปลี่ยนแปลงครั้งใหญ่, Test plan ล้าสมัย (outdated), ต้องปรับแผน test

>>>>> Continue reading ENTRY-EXIT-SUSPENSION.md

ส่วนที่ 6: Test Environment และ Test Data Planning

6.1 Test Environment Planning

Components:

```
Test Environment = Hardware + Software + Network + Data + Tools
```

ตัวอย่าง: Library Management System

1. Hardware Requirements:

- Test Server: 4 CPU cores, 8GB RAM, 100GB disk
- Client Machines: Standard laptops/desktops
- Mobile Devices (optional): iOS, Android

2. Software Requirements:

Operating System:

- Server: Ubuntu 24.04 LTS
- Client: Windows 10+, macOS 12+, Ubuntu 22+

Application Stack:

- Node.js 18 LTS
- MySQL 8.0
- Apache/Nginx web server

Browsers:

- Chrome 120+
- Firefox 120+
- Safari 17+
- Edge 120+

Testing Tools:

- Jest 29+
- Playwright 1.40+
- supertest 6+
- k6 (performance)
- OWASP ZAP (security)

6.2 Test Data Planning

Types of Test Data (ประเภทข้อมูลทดสอบ)

1. **Master Data (Static) - ข้อมูลหลัก**
   - บัญชีผู้ใช้ (User accounts)
   - แค็ตตาล็กหนังสือ (Book catalog)
   - การกำหนดค่าระบบ (System configuration)
2. **Transactional Data (Dynamic) - ข้อมูลธุรกรรม**
   - บันทึกการยืม/คืน (Borrow/return records)
   - ประวัติการค้นหา (Search history)
   - บันทึกกิจกรรม (User activity logs)
3. **Edge Case Data (ข้อมูลกรณีขอบ)**
   - สตริงยาวมาก (Very long strings)
   - อักขระพิเศษ (Special characters)
   - ค่าว่าง (Empty values)
   - Null values
   - ค่าสูงสุด/ต่ำสุด (Maximum/minimum values)

ส่วนที่ 7: Exploratory Testing (การทดสอบเชิงสำรวจ)

7.1 Exploratory Testing คืออะไร

คำจำกัดความ:

การทดสอบแบบสำรวจ (Exploratory Testing) คือการทดสอบแบบที่ออกแบบ test, execute test, และเรียนรู้ไปพร้อมกันในเวลาเดียวกัน

Scripted Testing vs Exploratory Testing

Aspect	Scripted Testing	Exploratory Testing
Planning	Pre-planned test cases	Learn as you test
Flexibility	Rigid, follow scripts	Flexible, adaptive
Coverage	Known areas	Unknown areas, edge cases
Creativity	Low	High
When	Stable features	New features, bug hunting

Skill Level	Junior testers can do	Need experienced testers

เปรียบเทียบ

Scripted Testing = การเดินตามเส้นทาง GPS

- คุณรู้ว่าต้องไปที่ไหน
- เส้นทางที่กำหนดไว้ล่วงหน้า
- การค้นพบน้อยลง

Exploratory Testing = การผจญภัยบนท้องถนน

- สำรวจในขณะที่คุณไป
- ค้นพบสิ่งใหม่ๆ
- ข้อมูลเชิงลึกเพิ่มเติม

7.2 การจัดการการทดสอบแบบ Session - Session-Based Test Management (SBTM)

นิยาม วิธีการจัดการและวัดผลการทดสอบแบบสำรวจ (Exploratory Testing) โดยแบ่งการทดสอบออกเป็น "เซสชัน" ที่มีเป้าหมายชัดเจน และมีระยะเวลาจำกัด เพื่อให้การทดสอบมีความเป็นระบบ (Structured) แต่ยังคงความยืดหยุ่น (Flexible) ช่วยให้นักทดสอบสามารถค้นหาข้อบกพร่องได้อย่างมีประสิทธิภาพ พร้อมทั้งเก็บข้อมูลและเมตริกสำคัญ เพื่อให้ทีมเห็นภาพรวมความคืบหน้าและเรียนรู้จากกระบวนการได้ดีขึ้น โดยไม่ต้องยึดติดกับ Test Case แบบดั้งเดิมที่ตายตัว

SBTM Structure:

Session = Charter + Time Box + Notes + Debrief

หลักการสำคัญของ SBTM

1. กำหนดเป้าหมาย (Charter): ก่อนเริ่มเซสชัน นักทดสอบจะกำหนดวัตถุประสงค์เฉพาะเจาะจงว่าต้องการทดสอบส่วนไหนของแอปพลิเคชันหรือคุณสมบัติใด (เช่น "ทดสอบการลงทะเบียนผู้ใช้ใหม่ด้วยรูปแบบอีเมลที่หลากหลาย")
2. เซสชันที่มีกรอบเวลา (Time-boxed Sessions): การทดสอบจะเกิดขึ้นในช่วงเวลาที่กำหนดไว้ (เช่น 60-90 นาที) โดยไม่มีการขัดจังหวะเพื่อให้มีสมาธิ
3. การออกแบบและดำเนินการทดสอบทันที: นักทดสอบออกแบบและดำเนินการทดสอบไปพร้อมๆ กัน (On-the-fly) โดยใช้ความคิดสร้างสรรค์และอิสระในการสำรวจ
4. การบันทึก (Notes & Debriefing): บันทึกผลการทดสอบ ข้อค้นพบ ข้อผิดพลาด และปัญหาที่พบอย่างละเอียด รวมถึงการทำ Debriefing (สรุป) หลังจบเซสชันเพื่อแลกเปลี่ยนข้อมูล

1. Session Charter - พันธกิจสำหรับเซสชันการทดสอบ

Template

สำรวจ [พื้นที่]
ด้วย [แหล่งข้อมูล]
วิธีค้นหา [ข้อมูล]

ตัวอย่าง Charter: Library Management System

Charter 1:
สำรวจฟังก์ชันการค้นหาหนังสือ

ด้วยตัวกรองการค้นหาและตัวเลือกการจัดเรียงที่แตกต่างกัน
เพื่อค้นหากรณีขอบและปัญหาการใช้งานs

Charter 2:
สำรวจเวิร์กโฟลว์การยืมหนังสือ
ด้วยประเภทผู้ใช้และสถานะความพร้อมของหนังสือที่แตกต่างกัน
เพื่อค้นหาข้อบกพร่องของตรรกะทางธุรกิจและปัญหาการจัดการข้อผิดพลาด

Charter 3:
สำรวจการรับรองความถูกต้องของผู้ใช้
ด้วยการรวมอินพุตและสถานการณ์เซสชันที่หลากหลาย
เพื่อค้นหาช่องโหว่ด้านความปลอดภัยและปัญหาการเข้าสู่ระบบ

2. Time Boxing (การแบ่งเวลา)

- ระยะเวลาเซสชั่น: 45-90 นาที (โดยทั่วไป 60 นาที)
- เวลาไม่ขัดจังหวะ (Uninterrupted time)
- การทดสอบที่เน้นเจาะจง (Focused testing)

7.3 Test Heuristics (การแบ่งแยกการทดสอบ)

Heuristic คือ กฎทางลัด หรือ หลักการโดยประมาณ (Rules of thumb) ที่ผู้ทดสอบซอฟต์แวร์ (Tester) ใช้เพื่อช่วยในการตัดสินใจ ออกแบบกรณีทดสอบ (Test cases) หรือค้นหาจุดบกพร่อง (Bugs) ได้อย่างรวดเร็วและมีประสิทธิภาพ แทนที่จะต้องทำตามขั้นตอนที่ตายตัวหรือทดสอบทุกความเป็นไปได้ซึ่งทำได้ยากในทางปฏิบัติ Heuristics จะเน้นการใช้ประสบการณ์และรูปแบบปัญหาที่พบบ่อยมาเป็นแนวทาง

SFDIPOT Heuristic

S - โครงสร้าง (Structure)

- ทดสอบโครงสร้างของ code/data
- Example: Invalid HTML, malformed JSON, broken links

F - ฟังก์ชัน (Function)

- ทดสอบ core functionality
- Example: Does search actually search? Does sort work?

D - ข้อมูล (Data)

- ทดสอบข้อมูล input/output
- Example: Special characters, very long text, numbers, dates

I - อินเตอร์เฟส (Interface)

- ทดสอบ user interface และ APIs
- Example: Button clicks, form submissions, API calls

P - แพลตฟอร์ม (Platform)

- ทดสอบบน platforms ต่าง ๆ
- Example: Different browsers, OS, screen sizes

O - การดำเนินการ (Operations)

- ทดสอบ operational aspects
- Example: Install, upgrade, backup, performance

T - เวลา (Time)

- ทดสอบเรื่อง timing และ concurrency (ความเกิดพร้อมกัน)
- Example: Timeout, race conditions, date/time handling

>>>>> Continue reading SFDIPOT.md

FEW HICCUPS Heuristic

F - Familiarity (ความคุ้นเคย)

- ใช้ความรู้จากระบบคล้ายๆ (Use Knowledge from Similar Systems)
- "Library systems usually have..." (ระบบห้องสมุดมักจะมี...)

E - Explainability (ความสามารถในการอธิบาย)

- ถาม "ทำไม" ต่อเนื่อง (Ask "Why" Continuously)
- "Why does this button do that?" (ทำไมปุ่มนี้ทำแบบนี้)

W - World (โลก)

- ทดสอบกับ real-world scenarios (สถานการณ์โลกแห่งความเป็นจริง)
- "What if user's name has accent marks?" (จะเป็นอย่างไรถ้าชื่อผู้ใช้มีเครื่องหมายสำเนียง)

H - History (ประวัติศาสตร์)

- ดูจาก bug history, past issues (ดูจากประวัติข้อบกพร่อง ปัญหาในอดีต)
- "Similar feature had pagination bug before" (คุณลักษณะที่คล้ายกันมีข้อบกพร่องการแบ่งหน้าก่อนหน้านี้)

I - Image (ภาพ)

- ดูระบบจากมุมมอง stakeholders (มองจากมุมมองของผู้มีส่วนได้ส่วนเสีย)
- "How would end users use this?" (ผู้ใช้ปลายทางจะใช้นี้อย่างไร)

C - Comparable (เปรียบเทียบได้)

- เปรียบเทียบกับ competitors (เปรียบเทียบกับคู่แข่ง)
- "How does Amazon handle this?" (Amazon จัดการแบบนี้อย่างไร)

C - Claims (การอ้าง)

- ตรวจสอบ claims ในเอกสาร (Check Claims in Documentation)
- "Spec says it's fast - is it really?" (ข้อมูลจำเพาะบอกว่ามันเร็ว - จริงหรือ)

U - User (ผู้ใช้)

- คิดเป็น user (Think as a User)
- "Would I use this?" (ฉันจะใช้นี้ไหม)

P - Purpose (จุดประสงค์)

- เข้าใจจุดประสงค์ (Understand the Purpose)
- "This feature should help users..." (คุณลักษณะนี้ควรช่วยผู้ใช้...)

S - Statutes (กฎหมาย)

- กฎหมาย, regulations, standards (ข้อบัญญัติ มาตรฐาน)
- "GDPR compliance?" (ปฏิบัติตาม GDPR ไหม)

7.4 Test Tours

นิยาม: แนวทางการทดสอบซอฟต์แวร์ที่ใช้ **"การเปรียบเปรยกับการท่องเที่ยว"** เพื่อสร้างโครงสร้างให้กับการทดสอบแบบสำรวจ (Exploratory Testing) โดยแทนที่ผู้ทดสอบจะสำรวจไปเรื่อยๆ อย่างไม่มีจุดหมาย "Test Tour" จะกำหนด **"ธีม" (Theme)** หรือหัวข้อเฉพาะเจาะจงในแต่ละรอบ เพื่อช่วยให้การค้นหา Bug มีทิศทางและครอบคลุมประเด็นสำคัญได้มากขึ้น

Popular Test Tours

1. Guidebook Tour (ทัศนศึกษาแนวทาง) - Tourist Testing
    - ทดสอบตามที่เขียนไว้ใน manual/documentation (เขียน คู่มือ/เอกสาร)
    - ตามเส้นทางที่มีความสุข (Follow happy path)
    - ตัวอย่าง: ทำตาม user guide ทุกขั้นตอน

2. Money Tour (ทัศนศึกษาเงิน)
    - ทดสอบ features ที่สำคัญที่สุด (ทำเงิน) (Most Important Features)
    - Features ที่ users ใช้บ่อยที่สุด (Most Frequently Used)
    - ตัวอย่าง LMS: Search ⟶ View ⟶ Borrow

3. Landmark Tour (ทัศนศึกษาจุดสำคัญ)
    - ทดสอบ features ที่โดดเด่น, น่าสนใจ (Prominent, Interesting Features)
    - ตัวอย่าง: Advanced search filters, recommendation system

4. Intellectual Tour (ทัศนศึกษาทางปัญญา)
    - ทดสอบ complex features ที่ต้องคิดเยอะ (Complex Features)
    - ตัวอย่าง: Late fee calculation, hold/reserve logic

5. FedEx Tour (ทัศนศึกษา FedEx)
    - Follow data flow ผ่านระบบ (Follow Data Through System)
    - ตัวอย่าง: Book data จาก add ⟶ search ⟶ borrow ⟶ return

6. Bad Neighborhood Tour (ทัศนศึกษาย่านที่ไม่ดี)
    - ทดสอบส่วนที่เคย มีปัญหามาก่อน (Previously Problematic Areas)
    - ตัวอย่าง: ถ้า login มี bug บ่อย ก็ test login มากๆ

7. Back Alley Tour (ทัศนศึกษาซอยน้อย)
    - ทดสอบ features ที่ไม่ค่อยมีคนใช้ (Rarely Used Features)
    - คุณลักษณะที่ซ่อนไว้ คุณลักษณะผู้ดูแล (Hidden features, admin features)
    - ตัวอย่าง: Export to CSV, database maintenance

8. Obsessive-Compulsive Tour (ทัศนศึกษา OCD)
    - ทำสิ่งเดิมซ้ำๆ หลายครั้ง (Repeat Same Action Many Times)
    - ตัวอย่าง: Add/remove book to cart 100 times

7.5 เทคนิคการทดสอบเชิงสำรวจ (Exploratory Testing Techniques)

1. การทดสอบขอบเขต (Boundary Testing)

- หา boundaries แล้วทดสอบขอบ
- 0, 1, max, max+1

2. การจับคู่ (Pairing)

- ทดสอบการทำงานร่วมกันของ features
- Search + Sort, Filter + Pagination

3. การโจมตีข้อจำกัด (Input Constraint Attack)

- ฝ่าฝืน constraints ทุกข้อ
- Required field ⟶ leave empty
- Max 50 chars ⟶ enter 1000 chars

4. Goldilocks (โกลดิลอกส์)

- ทดสอบน้อยเกินไป มากเกินไป พอดี (Test too little, too much, just right)
- Very short input, very long input, normal input

5. Personas (บุคลิกลักษณ์)

- ทดสอบในฐานะ user แต่ละ type (Test as Different User Types)
- Novice user, expert user, malicious user

7.6 การบันทึกผลลัพธ์ (Documenting Results)

Session Notes Template (แม่แบบบันทึกเซสชัน)

Exploratory Testing Session (เซสชันการทดสอบแบบสำรวจ)

Session Information (ข้อมูลเซสชัน)

- Date (วันที่): 2024-12-10
- Tester (ผู้ทดสอบ): John Doe
- Duration (ระยะเวลา): 60 minutes (นาที)
- Build Version (เวอร์ชัน Build): v1.2.0

Charter

สำรวจฟังก์ชันการค้นหาหนังสือด้วยตัวกรองการค้นหาต่างๆ และตัวเลือกการเรียงลำดับเพื่อค้นหากรณีขอบและปัญหาการใช้งาน

Areas Tested (พื้นที่ที่ทดสอบ)

- การค้นหาพื้นฐาน (ชื่อเรื่อง ผู้แต่ง ISBN)
- ตัวกรองขั้นสูง (หมวดหมู่ ช่วงปี)
- ตัวเลือกการเรียงลำดับ (AZ, ใหม่ล่าสุด, เรตติ้ง)
- การแบ่งหน้าผลการค้นหา
-การจัดการผลการค้นหาที่ว่างเปล่า

Test Techniques Used (เทคนิคการทดสอบที่ใช้)

- SFDIPOT heuristic (Data, Interface, Time)
- Boundary testing
- Input constraint attack
- Pairing (search + filter + sort)

Bugs Found (ข้อบกพร่องที่พบ)

BUG-101: ค้นหาด้วยอักขระพิเศษ (@#$) ส่งกลับข้อผิดพลาด 500
ความรุนแรง: สูง | ลำดับความสำคัญ: สูง | ความสำคัญ: สูง)

BUG-102: การเรียงลำดับตาม "ใหม่ล่าสุด" จะไม่พิจารณาปีที่เผยแพร่
ความรุนแรง: ปานกลาง | ลำดับความสำคัญ: ปานกลาง | ความสำคัญ: ปานกลาง)

BUG-103: การแบ่งหน้าจะหยุดเมื่อตัวกรองเปลี่ยนไปกลางหน้า
ความรุนแรง: ต่ำ | ลำดับความสำคัญ: ปานกลาง | ความสำคัญ: ปานกลาง)

Questions/Issues Raised (คำถาม/ปัญหาที่เกิดขึ้น)

- การค้นหาควรมีคำอธิบายหนังสือหรือแค่ชื่อ/ผู้แต่ง
- พฤติกรรมที่คาดหวังเมื่อค้นหาด้วยอีโมจิคืออะไร
- การค้นหาช้าด้วยผลลัพธ์ >1000 รายการ (5+ วินาที)

Coverage (ความครอบคลุม)

Tested (ทดสอบ)

- การตรวจสอบความถูกต้องของอินพุตการค้นหา
- ตัวกรองหลายเงื่อนไข
- ฟังก์ชันการจัดเรียง
- การแบ่งหน้าด้วยตัวกรอง

Not Tested (ไม่ได้ทดสอบ)

- ประสิทธิภาพการค้นหาภายใต้ภาระงานสูง
- ค้นหาด้วยภาษาต่างๆ
- การค้นหาที่ปรับเปลี่ยนตามอุปกรณ์เคลื่อนที่

Ideas for Future Sessions (ไอเดียสำหรับเซสชันในอนาคต)

- ทดสอบการค้นหาด้วยอักขระสากล (จีน อาหรับ)
- สำรวจเวิร์กโฟลว์การค้นหา + ระงับ/จองหนังสือ
- ทดสอบการเข้าถึงการค้นหา (screen reader)

Time Breakdown (การแบ่งเวลา)

- Setup (การตั้งค่า): 5 min
- Testing (การทดสอบ): 45 min
- Reporting (การรายงาน): 10 min

Notes (บันทึก)

- โดยทั่วไปการค้นหาจะรวดเร็ว (<1 วินาที) สำหรับคำค้นหาที่สมเหตุสมผล
- UI ใช้งานง่าย แต่ข้อความแสดงข้อผิดพลาดอาจชัดเจนกว่า
- สังเกตเห็นบางครั้งข้อมูลหนังสือซ้ำกันในผลลัพธ์ - ตรวจสอบ

Summary & Key Takeaways (สรุป)

Test Planning & Strategy

- Test Plan คือ เอกสารรายละเอียด สำหรับโครงการ
- Test Strategy คือ แนวทางระดับสูง สำหรับองค์กร
- Entry/Exit Criteria เป็นตัวบ่งชี้ที่วัดได้

Risk-Based Testing

- ระบุและวิเคราะห์ความเสี่ยง
- จัดลำดับความสำคัญตามผลกระทบ
- ทดสอบความเสี่ยงสูง ก่อน

Test Estimation

- ใช้เทคนิคหลายอย่างและเปรียบเทียบ
- เพิ่มบัฟเฟอร์ 20-30%
- สื่อสารประมาณการอย่างชัดเจน

Exploratory Testing

- สำรวจ≠สุ่ม
- แม้การทดสอบแบบสำรวจจะเน้นความอิสระ แต่การขาดโครงสร้างจะทำให้ควบคุมเวลาและวัดผลได้ยาก **Session-Based Test Management (SBTM)** จึงเข้ามาช่วยในส่วนนี้
    - Time-boxing: กำหนดเวลาที่ชัดเจน (เช่น 60, 90 หรือ 120 นาที) ช่วยให้ Tester จดจ่อกับงานและไม่จมดิ่งกับจุดใดจุดหนึ่งนานเกินไป
    - Test Charter: มีภารกิจที่ชัดเจนว่า "จะทดสอบอะไร" และ "ต้องการเรียนรู้อะไร" ทำให้การสำรวจมีเป้าหมาย ไม่สะเปะสะปะ
    - Measurable Output: เมื่อจบเซสชัน จะมีการสรุปสิ่งที่ค้นพบ (Bugs, Issues, Risks) ทำให้ฝ่ายบริหารเห็นภาพรวมของคุณภาพซอฟต์แวร์ได้ชัดเจน
- Heuristics เปรียบเสมือน "เข็มทิศ" ที่ช่วยให้ Tester รู้ว่าควรจะทดสอบอย่างไรเมื่อเผชิญกับสถานการณ์ที่ซับซ้อน
    - Decision Making: ช่วยในการเลือกข้อมูลทดสอบหรือสถานการณ์ที่มีความเสี่ยงสูงได้อย่างรวดเร็ว โดยอาศัยรูปแบบปัญหาที่พบบ่อย (เช่น หลักการ Zero, One, Many)
    - Cognitive Support: ลดภาระทางความคิดของ Tester โดยมี "กฎทางลัด" (Rule of thumb) ให้ยึดถือ ทำให้สามารถโฟกัสไปที่การสังเกตพฤติกรรมของระบบที่ผิดปกติได้ลึกซึ้งขึ้น
    - Mnemonics: การใช้ตัวช่วยจำ (เช่น SFDPOT หรือ CRUD) ช่วยให้มั่นใจว่าครอบคลุมมิติการทดสอบที่สำคัญ เช่น โครงสร้าง (Structure), ฟังก์ชัน (Function), และ ข้อมูล (Data)

O[O