บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 แนวความคิดเกี่ยวกับการทดสอบ [5]

ในการทดสอบโปรแกรมสามารถแบ่งระดับ (Level) ของการทดสอบออกได้เป็นระดับต่างๆ ดังนี้

1) การทดสอบระดับหน่วย

เป็นการทดสอบการทำงานของแต่ละโมดูล (Module) ภายในโปรแกรม กรณีทดสอบจะสร้างขึ้นเพื่อทดสอบการทำงานภายในโมดูลนั้น โดยแต่ละโมดูลถูกทดสอบแยกจากกันทำให้ในบางโมดูลจำเป็นต้องใช้โปรแกรมที่ถูกเขียนขึ้นมาเฉพาะ นั่นคือตัวขับ (Driver) และตัวดำเนินการ (Stub) โดยตัวขับเป็นโปรแกรมที่ใช้จำลองการเรียกโมดูลที่ต้องการทดสอบพร้อมกับให้ข้อมูลเข้า (Input) และรับผลลัพธ์ (Output) จากการทำงานของโมดูลนั้น และตัวดำเนินการเป็นโปรแกรมจำลองที่โมดูลที่จะทดสอบนั้นต้องการเรียกใช้ ซึ่งกรณีทดสอบของการทดสอบระดับหน่วยต้องระบุตัวขับ และตัวดำเนินการ สำหรับแต่ละโมดูลไว้

2) การทดสอบการรวม

เป็นการตรวจสอบข้อผิดพลาดที่เกิดจากการรวมโมดูลเข้าด้วยกัน และเป็นการตรวจสอบความเข้ากันได้ระหว่างโมดูล โดยมีวิธีการในการรวม ดังนี้

2.1) การรวมจากบนลงล่าง (Top-down approach) เป็นการรวมโมดูลที่อยู่ระดับบนสุดในแผนภาพโครงสร้าง (Structure Chart) ก่อน ทำให้ต้องมีการสร้างตัวดำเนินการเพื่อให้โมดูลที่รวมเข้าด้วยกันสามารถเรียกใช้ได้ ซึ่งในกรณีทดสอบต้องระบุ ตัวดำเนินการที่ส่วนที่ทดสอบนั้นเรียกไว้ด้วย

2.2) การรวมจากล่างขึ้นบน (Bottom-up approach) เป็นการรวมโมดูลที่อยู่ระดับล่างสุดในแผนภาพโครงสร้างก่อน ทำให้ต้องมีการสร้างตัวขับ เพื่อเรียกและส่งข้อมูลไปยังโมดูลที่รวมเข้าด้วยกัน ในกรณีทดสอบต้องระบุตัวขับ ที่เรียกส่วนที่ทดสอบนั้นด้วย

2.3) การรวมแบบแซนด์วิช (Sandwich approach) เป็นการรวมโมดูลที่อยู่ด้านบนด้วยวิธีการรวมจากบนลงล่าง และรวมโมดูลที่อยู่ด้านล่างด้วยวิธีการแบบล่างขึ้นบน จากนั้นจึงรวมทั้ง 2 ส่วนเข้าด้วยกัน ต้องมีการสร้างทั้งตัวขับ และตัวดำเนินการ เพื่อให้โมดูลที่รวมกันนั้นเรียกใช้

2.4) การรวมแบบโมดูลวิกฤต (Critical Module approach) โดยเริ่มจากการรวมโมดูลที่มีความสำคัญมากกว่าโมดูลอื่น ๆ ก่อน เพื่อให้โมดูลเหล่านั้นได้ถูกทดสอบมากขึ้น

2.5) การรวมแบบบิ๊กแบง (Big-bang approach) เป็นการรวมโมดูลเข้าด้วยกันในครั้งเดียว ซึ่งเป็นวิธีที่นิยมใช้ในกรณีที่มีระยะเวลาในการทดสอบจำกัด

3) การทดสอบย้อนกลับ

เป็นการทดสอบที่สำคัญหลังจากการแก้ไขซอฟต์แวร์ โดยแบ่งออกเป็น 2 ประเภทคือ

3.1) การทดสอบหลังจากที่แก้ไขข้อผิดพลาดเรียบร้อยแล้ว เพื่อตรวจสอบการทำงานของซอฟต์แวร์ที่ได้ทำการ แก้ไข ว่าทำงานถูกต้องตามที่ได้ออกแบบไว้หรือไม่

3.2) การทดสอบหลังจากที่มีการเปลี่ยนแปลงหรือแก้ไขโปรแกรม เพื่อให้มั่นใจได้ว่าการทำงานของแต่ละโมดูลในโปรแกรมยังคงไม่มีข้อผิดพลาด และถูกต้องตามที่ได้ออกแบบไว้

4) การทดสอบระบบ

เป็นการตรวจสอบความถูกต้องในการทำงานของส่วน (Element) ต่าง ๆ ในระบบทั้งหมด โดยกรณีทดสอบที่สร้างขึ้นต้องครอบคลุมทุกๆหน้าที่ (Function) การทำงานของระบบทั้งซอฟต์แวร์และฮาร์ดแวร์ เพื่อให้ระบบที่พัฒนาไม่เกิดความผิดพลาดขึ้นในการทำงาน

4.1) การทดสอบตามหน้าที่ (Functional) โดยพยายามทดสอบความต้องการของผู้ใช้ทั้งหมดที่ได้กำหนดไว้ในข้อกำหนดความต้องการซอฟต์แวร์ (Software requirement specification)

4.2) การทดสอบอื่น ๆ (Non-Functional) ประกอบด้วย

4.2.1) การทดสอบความกดดัน (Stress testing) โดยทดสอบระบบภายใต้สภาพการใช้งานที่สูงมากๆ มีจุดประสงค์เพื่อต้องการทราบความสามารถสูงสุดของระบบภายใต้ทรัพยากรที่กำหนด เช่น การทดสอบจำนวนรายการที่เกิดขึ้น (Transaction) มากที่สุดที่ระบบสามารถรองรับได้ เป็นต้น

4.2.2) การทดสอบประสิทธิภาพ (Performance testing) เพื่อแสดงว่าระบบมีประสิทธิภาพตามที่ได้กำหนดไว้หรือไม่ และเพื่อปรับแต่งระบบให้มีประสิทธิภาพดีขึ้น โดยตรวจสอบจากเวลาตอบสนอง (Response time) ปริมาณงาน (Throughput) การใช้งานหน่วยประมวลผลกลาง (CPU utilization) เป็นต้น

4.2.3) การทดสอบส่วนหลัง (Background testing) เพื่อแสดงความสามารถของระบบในกรณีที่รองรับรายการมากกว่าหนึ่งรายการในเวลาเดียวกัน

4.2.4) การทดสอบโครงแบบ (Configuration testing) เพื่อต้องการทราบข้อกำหนดของฮาร์ดแวร์ (Hardware) ที่เหมาะสมกับระบบ

4.2.5) การทดสอบการกู้ระบบ (Recovery testing) เพื่อแสดงว่าระบบสามารถกู้ข้อมูลกลับคืนมาได้หากระบบเกิดความเสียหายขึ้น

4.2.6) การทดสอบความปลอดภัย (Security testing) เพื่อแสดงว่าระบบมีประสิทธิภาพเพียงพอในการรักษาความปลอดภัย และป้องกันระบบจากการลักลอบใช้งานของผู้ที่ไม่ได้รับอนุญาต

2.1.2 กรณีทดสอบ

กรณีทดสอบ คือข้อมูลที่ผู้ทดสอบใช้ทำการทดสอบรายการ (Item) ใด ๆ เพื่อให้ทราบว่าการทำงานของรายการ นั้นถูกต้องหรือไม่ ประกอบด้วยข้อมูลเข้า ผลที่คาดว่าจะได้ (Expected output) และข้อจำกัดต่าง ๆ ในการใช้กรณีทดสอบนั้น กรณีทดสอบที่ดีต้องมีคุณสมบัติคือ มีความน่าจะเป็นสูงที่จะเจอข้อผิดพลาด (Error) ซึ่งยังไม่ถูกค้นพบในโปรแกรม และกรณีทดสอบที่ประสบความสำเร็จ คือกรณีทดสอบที่สามารถค้นพบข้อผิดพลาดที่เกิดขึ้นในโปรแกรมได้ การเลือกกรณีทดสอบเพื่อนำไปทำการทดสอบ มีหลักการ ในการพิจารณาดังนี้ [6]

1) กรณีทดสอบที่ใช้ทดสอบความสามารถ ความมั่นคงของระบบมีความสำคัญมากกว่ากรณีทดสอบที่ใช้ทดสอบส่วนประกอบของระบบ เนื่องจากผู้ใช้จะให้ความสำคัญกับความมั่นคงของระบบในการทำงาน มากกว่า ข้อผิดพลาดเล็กน้อยที่เกิดขึ้นในระบบ ดังเช่นข้อผิดพลาดที่ทำให้เกิดการสูญเสีย หรือสูญหาย ของข้อมูล มีความสำคัญมากกว่าข้อผิดพลาดที่ทำให้ข้อมูลที่แสดงทางจอภาพขาดหายไป
2) ในกรณีทีมีการปรับปรุง หรือแก้ไขระบบใหม่ กรณีทดสอบที่ใช้ทดสอบความสามารถเดิมของระบบ มีความสำคัญมากว่ากรณีทดสอบสำหรับใช้ทดสอบความสามารถของระบบที่พัฒนาขึ้นใหม่
3) ในกรณีที่มีข้อจำกัดในเรื่องของจำนวนกรณีทดสอบ กรณีทดสอบที่มีข้อมูลทดสอบในลักษณะทั่วไปมีความสำคัญมากว่ากรณีทดสอบที่มีข้อมูลทดสอบอยู่ในช่วงที่เป็นค่าขอบเขต (Boundary value)

2.1.3 เทคนิคที่ใช้ในการออกแบบกรณีทดสอบ

การออกแบบกรณีทดสอบในแต่ละระดับแบ่งออกเป็นสองประเภท คือ

1) การออกแบบกรณีทดสอบแบบไวท์บอกซ์ (White-box testing) [7]

เป็นการสร้างกรณีทดสอบโดยพิจารณาจาก โครงสร้าง (Structure) ของโปรแกรม โดยสร้างกรณีทดสอบที่ครอบคลุมทุกเส้นทาง (Path) ตามโครงสร้างของโปรแกรม เพื่อให้แน่ใจว่าทุกคำสั่งในโปรแกรมได้รับการทดสอบอย่างน้อยหนึ่งครั้ง และทุกคำสั่งที่เป็นเงื่อนไขถูกทดสอบ ทั้งกรณีที่เงื่อนไขนั้นเป็นจริงและกรณีที่เงื่อนไขนั้นเป็นเท็จ นอกจากนี้ยังสามารถค้นพบข้อผิดพลาดที่เกิดขึ้นจากข้อมูลภายในโมดูลได้อีกด้วย

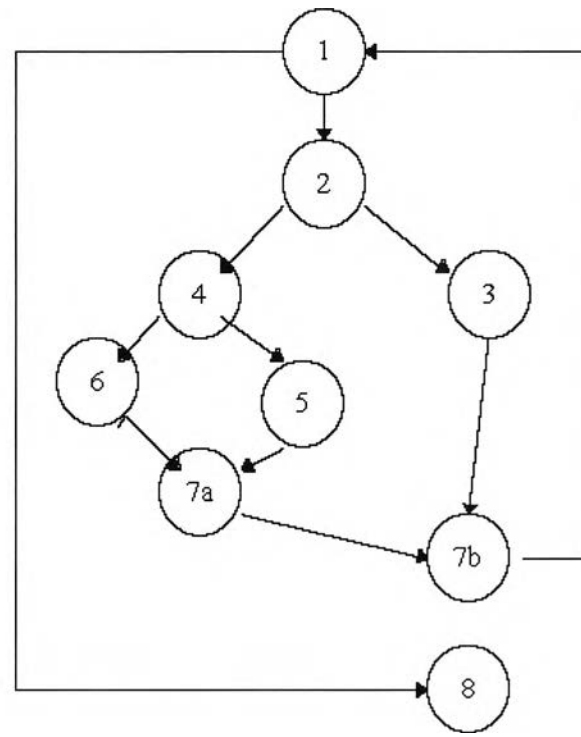
การสร้างกรณีทดสอบด้วยวิธีการของไวท์บอกซ์ ผู้ทดสอบต้องสร้างกราฟการควบคุมกระแส (Control flow graph) จากโปรแกรมที่พัฒนาขึ้น และนำกราฟที่ได้มาสร้างกรณีทดสอบเพื่อให้ครอบคลุมทุกคำสั่ง ทุกเส้นทาง และทุกเงื่อนไขของโปรแกรม ดังรูปที่ 2.1 แสดงรหัสเทียม (Psuedocode) และกราฟการควบคุมกระแสที่สร้างขึ้นจากรหัสเทียมนั้น ซึ่งสามารถเขียนเส้นทางที่ได้รับการทดสอบดังต่อไปนี้

1) 1, 8
2) 1, 2, 3, 7b, 1, 8

3) 1, 2, 4, 5, 7a, 7b, 1, 8

4) 1, 2, 4, 6, 7a, 7b, 1, 8

```
      PDL for PROCEDURE SORT
1:        do while records remain
              read record;
2:            if record field 1 = 0
3:                then process record;
                       store in buffer;
                       increment counter;
4:            elsif record field 2 = 0
5:                then reset record;
6:                else process record;
                  store in file;
7a:           endif
          endif
7b:       enddo
8:    end
```



รูปที่ 2.1 ตัวอย่างรหัสเทียม (Psuedocode) และกราฟการควบคุมกระแสที่สร้างขึ้นจากรหัสเทียม

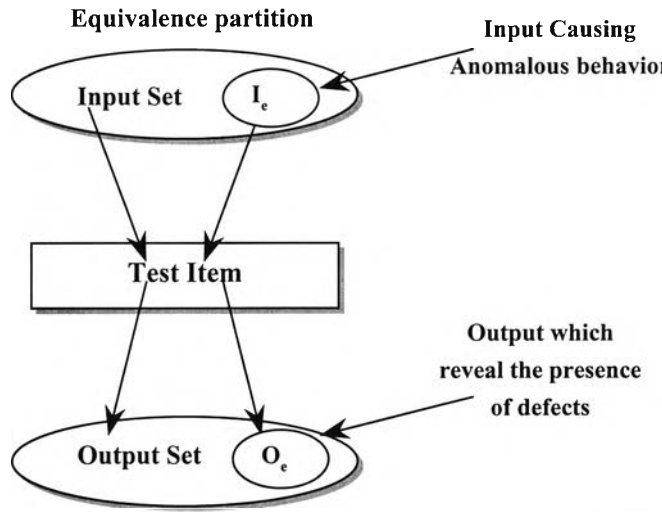
2) การออกแบบกรณีทดสอบแบบแบล็กบอกซ์ [5]

เป็นการสร้างกรณีทดสอบโดยพิจารณาจากข้อกำหนด (Specification) และเงื่อนไข (Condition) ของข้อมูลนั้น การทดสอบสนใจเฉพาะข้อมูลเข้าและผลลัพธ์ของโมดูลที่ต้องการทดสอบ เพื่อให้ตรงตามข้อกำหนดที่ได้ออกแบบไว้เท่านั้น ซึ่งจะไม่พิจารณาถึงโครงสร้างภายในโมดูลนั้น

การออกแบบกรณีทดสอบแบบแบล็กบอกซ์มีขั้นตอนที่สำคัญ 2 ขั้นตอน คือ

1.1) การแยกชั้นสมมูล

เป็นเทคนิคเพื่อใช้ตัดสินว่าชั้นของข้อมูลเข้า (Input data) ใดเป็นชั้นปรกติ และชั้นใดเป็นชั้นที่ทำให้เกิดข้อผิดพลาด ดังรูปที่ 2.2 แสดงส่วนสมมูล (Equivalence partition) ของข้อมูลเข้า ซึ่งประกอบด้วยชั้นของข้อมูลเข้าที่ปกติ (Input Set) เป็นชั้นของข้อมูลเข้าที่เมื่อเข้าสู่รายการทดสอบแล้วให้ผลลัพธ์อยู่ในเซตของผลลัพธ์ที่ถูกต้อง (Output Set) และชั้นของข้อมูลเข้าที่ไม่ถูกต้อง ($\mathbf{I}_e$) ที่เมื่อเข้าสู่รายการทดสอบแล้วให้ผลลัพธ์อยู่ในเซตของผลลัพธ์ที่ไม่ถูกต้อง ($\mathbf{O}_e$) ซึ่งจะแสดงให้เห็นถึงข้อผิดพลาดของรายการทดสอบนั้น



รูปที่ 2.2 แสดงส่วนสมมูลของข้อมูลเข้า ประกอบด้วยชั้นของข้อมูลเข้าที่ปกติ และชั้นของข้อมูลเข้าที่ทำให้เกิดข้อผิดพลาด

การแยกประเภทของส่วนสมมูลสามารถทำได้โดยพิจารณาจากข้อกำหนด และเงื่อนไขของข้อมูลเข้า จากเอกสารของผู้ใช้งาน (User Documentation) หรือจากประสบการณ์ของผู้ทดสอบเองว่าชั้นของข้อมูลเข้าใดเป็นชั้นที่จะทำให้เกิดข้อผิดพลาดขึ้นในโปรแกรม และชั้นของข้อมูลเข้าใดเป็นชั้นของข้อมูลที่ถูกต้อง

ผลที่ได้จากขั้นตอนของการแยกชั้นของข้อมูลเข้า (Input equivalence class) คือชั้นของข้อมูลเข้าที่ถูกต้อง (Valid class) และชั้นของข้อมูลเข้าที่ทำให้เกิดข้อผิดพลาด (Invalid class) บางครั้งอาจเรียกว่าชั้นของข้อมูลเข้าเหล่านั้นว่า ชั้นสมมูล (Equivalence class) สำหรับข้อกำหนด และเงื่อนไขของข้อมูลเข้านั้น

ข้อกำหนดหรือเงื่อนไขของข้อมูลเข้าที่พบโดยทั่วไป ได้แก่

- ข้อมูลที่ระบุค่าเฉพาะ
- ข้อมูลที่ระบุค่าอยู่ในช่วงของข้อมูล
- ข้อมูลที่ระบุเซตของข้อมูล
- ข้อมูลประเภทค่าตรรกศาสตร์

ซึ่งสามารถกำหนดชั้นสมมูลสำหรับแต่ละข้อกำหนดหรือเงื่อนไขของข้อมูล ได้ดังนี้

1) ถ้าข้อมูลเข้าระบุค่าเฉพาะ จะมีหนึ่งชั้นสมมูลที่เป็นชั้นของข้อมูลเข้าที่ถูกต้อง คือข้อมูลเข้าที่มีค่าเท่ากับค่าเฉพาะที่ระบุ และ สองชั้นของข้อมูลเข้าที่ทำให้เกิดข้อผิดพลาด คือข้อมูลที่มีค่ามากกว่า และ น้อยกว่าค่าเฉพาะที่ระบุ ตัวอย่างเช่น ถ้าข้อมูลเข้ากำหนดให้มีค่าเท่ากับ 1 จะมี 1 ชั้นสมมูลที่เป็นชั้นของข้อมูลเข้าที่ถูกต้อง คือชั้นสมมูลของข้อมูลเข้าที่มีค่าเท่ากับ 1 และ มี 2 ชั้นสมมูลที่เป็นชั้นสมมูลของข้อมูลเข้าที่ไม่ถูกต้อง คือชั้นสมมูลของข้อมูลเข้าที่มีค่าน้อยกว่า 1 และชั้นสมมูลของข้อมูลเข้าที่มีค่ามากกว่า 1
2) ถ้าข้อมูลเข้าระบุค่าอยู่ในช่วงของข้อมูล จะมีหนึ่งชั้นสมมูลที่เป็นชั้นของข้อมูลเข้าที่ถูกต้อง คือข้อมูลที่มีค่าอยู่ในช่วงนั้น และ สองชั้นสมมูลของข้อมูลเข้าที่ไม่

ถูกต้อง คือข้อมูลที่มีค่ามากกว่า และ น้อยกว่าช่วงของข้อมูลนั้น เช่น ถ้าข้อมูลเข้ามีค่าอยู่ระหว่าง 0 ถึง 10 จะมี 1 ชั้นสมมูลที่เป็นชั้นของข้อมูลเข้าที่ถูกต้อง คือ ชั้นสมมูลของข้อมูลเข้าที่มีค่าอยู่ระหว่าง 0 ถึง 10 และ มี 2 ชั้นสมมูลที่เป็นชั้นสมมูลของข้อมูลเข้าที่ทำให้เกิดข้อผิดพลาด คือชั้นสมมูลของข้อมูลเข้าที่มีค่าน้อยกว่า 0 และชั้นสมมูลของข้อมูลเข้าที่มีค่ามากกว่า 10

3) ถ้าข้อมูลเข้าที่ระบุเซตของข้อมูล จะมีหนึ่งชั้นสมมูลที่เป็นชั้นของข้อมูลเข้าที่ถูกต้อง คือข้อมูลที่เป็นสมาชิกของเซตที่ระบุ ที่ไม่ถูกต้อง คือข้อมูลที่ไม่เป็นสมาชิกของเซตที่ระบุ เช่น ถ้าข้อมูลเข้าเป็นเซตของจำนวนเฉพาะ จะมี 1 ชั้นสมมูลที่เป็นข้อมูลเข้าที่ถูกต้องคือข้อมูลที่เป็นเลขจำนวนเฉพาะ และมี 1 ชั้นสมมูลที่เป็นชั้นสมมูลของข้อมูลเข้าที่ทำให้เกิด ข้อผิดพลาด คือข้อมูลที่ไม่เป็นเลขจำนวนเฉพาะ

4) ถ้าข้อมูลเข้าประเภทค่าตรรกศาสตร์ จะมีหนึ่งชั้นสมมูลที่เป็นชั้นของข้อมูลเข้าที่ถูกต้อง และ หนึ่งชั้นของข้อมูลเข้าที่ไม่ถูกต้อง เช่นข้อมูลเข้าที่กำหนดค่าของข้อมูลเข้าคือ จริง ชั้นสมมูลที่เป็นข้อมูลเข้าที่ถูกต้องคือข้อมูลที่เป็นมีค่าความจริงเป็นจริง และมี 1 ชั้นสมมูลที่เป็นชั้นสมมูลของข้อมูลเข้าที่ทำให้เกิดข้อผิดพลาด คือข้อมูลที่มีค่าความจริงเป็นเท็จ

1.2) การวิเคราะห์ค่าขอบเขต (Boundary Value Analysis) [5]

เนื่องจากข้อผิดพลาดที่เกิดขึ้นในโปรแกรมส่วนใหญ่จะเกิดในบริเวณที่เป็นค่าขอบเขตของข้อมูล ทำให้เมื่อเราสร้างชั้นสมมูลแล้วต้องมีการทำขั้นตอนการวิเคราะห์ค่าขอบเขต เพื่อให้กรณีทดสอบที่สร้างขึ้นมีประสิทธิภาพมากที่สุดในการค้นหาข้อผิดพลาด

รูปแบบของการวิเคราะห์ค่าขอบเขตสำหรับลักษณะเฉพาะ หรือเงื่อนไขของข้อมูลเข้าโดยทั่วไปมีดังนี้

1) ถ้าข้อมูลเข้าระบุค่าอยู่ในช่วงของข้อมูล a และ b ข้อมูลสำหรับทดสอบค่าขอบเขตในกรณีทดสอบที่สร้างขึ้นควรมีค่าอยู่ เหนือและ ใต้ค่า a และ b เล็กน้อย

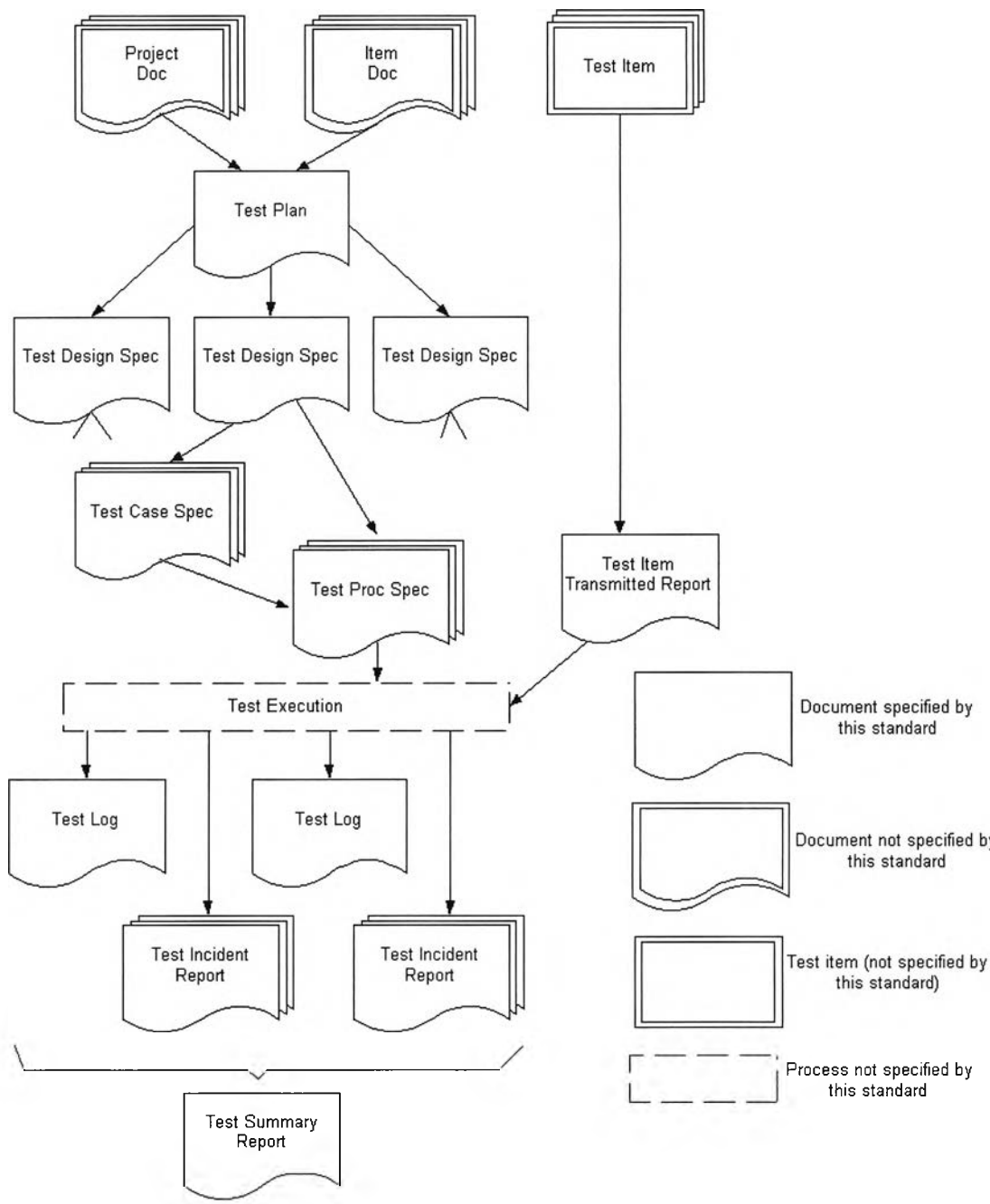
2) ถ้าข้อมูลเข้าระบุจำนวนมากที่สุด หรือน้อยที่สุดของข้อมูลเข้า กรณีทดสอบที่สร้างควรทดสอบค่ามากที่สุด น้อยที่สุด นั้น และทดสอบบริเวณที่เป็นขอบเขต คือ เหนือและใต้ จำนวนมากที่สุด น้อยที่สุดนั้นด้วย

3) ประยุกต์ข้อ 1 และ 2 สำหรับ ผลลัพธ์ ที่ต้องการ

4) ถ้าโครงสร้างภายในของโปรแกรมมีการกำหนดขอบเขตไว้ สร้างข้อมูลทดสอบเพื่อทดสอบโครงสร้างนั้น เช่นถ้าโปรแกรมกำหนดข้อมูลแบบอาเรย์ขนาด 100 ข้อมูล ต้องสร้างข้อมูลทดสอบในบริเวณที่เป็นขอบ (99,100,101) เซตสำหรับทดสอบข้อมูล อาเรย์ขนาดนั้น

2.1.4 มาตรฐาน IEEE Std 829-1998 [1]

เป็นมาตรฐานที่เกี่ยวกับเอกสาร และข้อมูลที่ใช้ในการทดสอบ โดยมาตรฐานนี้อธิบายเกี่ยวกับจุดมุ่งหมาย แนวทาง และองค์ประกอบเบื้องต้นของเอกสารและข้อมูลที่ใช้ในการทดสอบซอฟต์แวร์ ดังแสดงในรูปที่ 2.3 เพื่อให้เป็นมาตรฐานและเกิดความเข้าใจตรงกันระหว่างผู้พัฒนาและลูกค้า อีกทั้งยังทำให้การทดสอบควบคุมได้ เนื่องจากสามารถตรวจสอบการทำงานในแต่ละขั้นตอนจากเอกสารและ ข้อมูลที่บันทึกไว้ได้ ซึ่งมาตรฐานนี้ถูกออกแบบมาโดยไม่ขึ้นกับประเภทซอฟต์แวร์ ขนาดของคอมพิวเตอร์ วิธีการทดสอบ หรือเครื่องมือที่ช่วยในการทดสอบต่าง ๆ สามารถนำมาใช้ได้ในการทดสอบซอฟต์แวร์ทุกประเภท และในทุกระดับของการทดสอบ มาตรฐานนี้ประกอบด้วยส่วนต่าง ๆ ดังต่อ ไปนี้



รูปที่ 2.3 ความสัมพันธ์ของเอกสารที่ใช้ในการทดสอบตามมาตรฐาน IEEE Std.829-1998[1]

1) การวางแผนการทด สอบ (Test Planning)

ระบุถึงขอบเขต แนวทาง ทรัพยากร และกำหนดการ ของการทดสอบ ระบุส่วนของโปรแกรม ลักษณะ (Feature) ที่จะทดสอบ กิจกรรมที่ต้องทำ ความรับผิดชอบของผู้ทดสอบ และความเสี่ยงที่จะเกิดขึ้นในแผนการทดสอบ

2) รายละเอียดของการทดสอบ (Test Specification) ประกอบด้วยเอกสารและข้อมูลต่าง ๆ ได้แก่

2.1) รายละเอียดเกี่ยวกับการออกแบบการทดสอบ (Test-design Specification) ระบุการออกแบบการทดสอบเพื่ออธิบายลักษณะที่จะทดสอบที่ได้กำหนดไว้ในการวางแผนการทดสอบ นอกจากนั้นยังระบุถึงกรณีทดสอบที่เกี่ยวข้องและหลักในการพิจารณาการทดสอบในแต่ละกรณีทดสอบว่าผ่านหรือไม่ผ่านเกณฑ์ที่กำหนด

2.2) รายละเอียดเกี่ยวกับกรณีทดสอบ (Test-case Specification) ระบุค่าของข้อมูลเข้าเพื่อใช้ในการทดสอบและกำหนดผลลัพธ์ที่ต้องการ นอกจากนั้นยังระบุข้อจำกัดต่าง ๆ ที่มีในการใช้กรณีทดสอบนั้น

ข้อมูลในรายละเอียดเกี่ยวกับกรณีทดสอบ ได้แก่

- ตัวระบุกรณีทดสอบ (Test case specification identifier) เพื่อให้ผู้ทดสอบสามารถระบุกรณีทดสอบที่ใช้ทดสอบแต่ละรายการทดสอบได้
- รายการที่ทดสอบ (Test item) แสดงชื่อ และ/หรือ หน้าที่ของรายการที่จะทดสอบสำหรับกรณีทดสอบนั้น
- ค่าของข้อมูลเข้าเพื่อใช้ในการทดสอบ (Input specification)
- ผลลัพธ์ที่คาดว่าจะได้รับจากการทดสอบ (Output specification)
- ลักษณะสิ่งแวดล้อมที่ต้องการในการทดสอบ (Environment needs) เช่นลักษณะอุปกรณ์ ซอฟต์แวร์
- ความต้องการพิเศษอื่นๆ (Special procedural requirement) ระบุความต้องการพิเศษอื่นๆ เพื่อใช้สำหรับกรณีทดสอบนี้
- รายการกรณีทดสอบที่เกี่ยวข้อง (Intercase dependency) แสดงรายการกรณีทดสอบที่ต้องทำการทดสอบก่อนหน้ากรณีทดสอบนี้ และความสัมพันธ์กับกรณีทดสอบนี้

2.3) ขั้นตอนการทดสอบ (Test Procedure Specification) ระบุขั้นตอนทั้งหมดที่ต้องทำในการจัดการทดสอบโดยใช้กรณีทดสอบที่ได้ออกแบบไว้ วัตถุประสงค์ของขั้นตอนการทดสอบ และความต้องการพิเศษที่ใช้ในการทดสอบกรณีทดสอบตามขั้นตอนที่ระบุ

2.4) การรายงานผลการทดสอบ (Test Reporting) ประกอบด้วยเอกสารต่าง ๆ ดังนี้

2.4.1) รายงานส่วนของโปรแกรมที่ถูกส่งไปทดสอบ (Test item transmitted report) ระบุรายละเอียดของรายการทดสอบ สถานะของรายการทดสอบ ผู้รับผิดชอบการทดสอบ

2.4.2) บันทึกการทดสอบ (Test log) บันทึกผลลัพธ์ที่ได้จากการทดสอบโดยใช้กรณีทดสอบต่าง ๆ รวมทั้งกิจกรรมและเหตุการณ์ที่เกิดขึ้น (Activity and event entries) ประกอบด้วยข้อมูลดังนี้

- คำอธิบายการกระทำ(Execution description) อธิบายลักษณะ ขั้นตอนการทดสอบ ผู้ที่เกี่ยวข้องกับการทดสอบ วัน เวลาที่ทดสอบ และเหตุการณ์ต่างที่เกิดขึ้นในขณะทดสอบ
- ผลการทดสอบแต่ละขั้นตอน (Procedure result) ผลการทดสอบ และผลลัพธ์ (Output) ที่ได้จากการทดสอบแต่ละขั้นตอน
- ข้อมูลสภาพแวดล้อม (Environment information) ระบุสภาพแวดล้อม อุปกรณ์ซอฟต์แวร์ต่างๆ ในขณะทำการทดสอบ

เหตุการณ์ผิดปกติ (Anomalous event) ระบุลักษณะเหตุการณ์ผิดปกติที่เกิดขึ้น เหตุการณ์ก่อนและหลัง และสภาพแวดล้อมก่อน และหลังเกิดเหตุผิดปกติ

4.3.2) รายงานการทดสอบที่ต้องดำเนินการต่อ (Test incident report) ระบุถึงเหตุการณ์ที่เกิดขึ้นในระหว่างการทดสอบซึ่งต้องการการติดตามตรวจสอบต่อ ประกอบด้วย

- ผลสรุปเหตุการณ์ที่เกิดขึ้น (Summary) สรุปเหตุการณ์ที่เกิดขึ้น ลำดับขั้นตอนการเกิดเหตุการณ์ รายการที่ทดสอบ ลำดับขั้นตอนการทดสอบ
- คำอธิบายเหตุการณ์ (Incident description) ลักษณะเหตุการณ์ที่เกิดขึ้น วันที่ เวลาที่เกิด เหตุผิดปกติที่เกิด ข้อมูลเข้า ผลลัพธ์ ผลที่คาดว่าจะได้รับ และผู้ทดสอบ
- ผลกระทบของเหตุการณ์ที่เกิดขึ้น (Impact) ผลกระทบของเหตุการณ์ที่เกิดขึ้นกับระบบ หรือ กับกรณีทดสอบอื่นๆ

4.3.3) รายงานสรุปผลการทดสอบ (Test summary report) สรุปกิจกรรมที่เกิดขึ้นตามที่ได้ระบุไว้ในรายละเอียดเกี่ยวกับการออกแบบการทดสอบ สรุปผลการทดสอบ การประเมินผลการทดสอบ และผู้ที่รับผิดชอบการทดสอบ

2.2 งานวิจัยที่เกี่ยวข้อง

1) ระบบจัดการกรณีทดสอบ (Test case management System) [2]

พัฒนาโดย Hiren D. Desai เป็นระบบที่พัฒนาขึ้นเพื่อใช้ในการทดสอบโปรแกรมในหน่วยงานของบริษัทเบลเซาท์ (BellSouth Co. Ltd.) ที่ประเทศสหรัฐอเมริกา ระบบจะเก็บรวบรวมข้อมูลกรณีทดสอบ และผลการทดสอบสำหรับโครงการต่างๆที่กำลังพัฒนา ผู้ทดสอบจะทำการบันทึกกรณีทดสอบที่ได้ออกแบบไว้ หลังจากนั้นระบบจะสร้างบทคำสั่งการทดสอบ (Test script) เพื่อนำไปทดสอบโปรแกรม ในขั้นตอนการทดสอบโปรแกรมระบบสามารถรับผลการทดสอบจากผู้ทดสอบ หรือทดสอบและบันทึกผลการทดสอบโปรแกรมได้โดยอัตโนมัติ

เครื่องมือนี้สามารถนำข้อมูลผลการทดสอบมาประมวลผล และออกรายงานสรุปผลการทดสอบ และรายงานความก้าวหน้าในการทดสอบให้กับผู้ใช้ นอกจากนี้ผู้ใช้ยังสามารถกำหนดรูปแบบ และข้อมูลที่จะแสดงในรายงานได้ อีกด้วย

ระบบนี้สามารถทดสอบโปรแกรมที่เขียนด้วยภาษาซี (C Language) และทำงานบนระบบปฏิบัติการยูนิกซ์ (Unix) ซึ่งทำให้การพัฒนาโปรแกรมโดยใช้ภาษาอื่นไม่สามารถทำงานแบบอัตโนมัติได้

2) TestDirector 6 [3]

ถูกพัฒนาโดย บริษัท Mercury Interactive เป็นเครื่องมือที่ช่วยองค์กรในการทดสอบซอฟต์แวร์ทุกขั้นตอน โดยเริ่มจากการวางแผนการทดสอบ การดำเนินการทดสอบ และการจัดการข้อผิดพลาดที่เกิดขึ้น ในขั้นตอนการวางแผนการทดสอบ

เครื่องมือนี้จะช่วยสร้างแผนการทดสอบโดยผู้ใช้สามารถสร้างแผนการทดสอบในไมโครซอฟต์เวิร์ด (Microsoft Word) และนำเข้ามาใช้ในเครื่องมือนี้ได้ทันที จากนั้นผู้ทดสอบจะเป็นผู้สร้าง และบันทึกกรณีทดสอบเข้าสู่ระบบ ในส่วนของการดำเนินการทดสอบ เครื่องมือดังกล่าวสามารถทำการทดสอบโดยอัตโนมัติ หรือให้ผู้ทดสอบดำเนินการทดสอบเองแล้วรายงานผลการทดสอบกลับมายังเครื่องมือ หากเกิดข้อผิดพลาด เครื่องมือสามารถให้ข้อมูลที่เป็นประโยชน์ในการวิเคราะห์ข้อผิดพลาดเพื่อหาแนวทางแก้ไข นอกจากนี้เครื่องมือยังสามารถออกรายงานสรุปผลการทดสอบตามรูปแบบที่ใช้กำหนด โดยส่งข้อมูลไปยังไมโครซอฟต์เอ็คเซล (Microsoft Excel) และไมโครซอฟต์เวิร์ด (Microsoft Word) ได้อีกด้วย

3) พลัสวันเทส (+1 TEST) [4]

เป็นระบบจัดการกรณีทดสอบ ของบริษัทพลัสวันซอฟต์แวร์เอ็นจิเนียริ่ง (+1 Software Engineering Co. Ltd.) ซึ่งสามารถใช้ได้กับทุกขั้นตอนในการทดสอบโปรแกรมทั้งการทดสอบระดับหน่วย การทดสอบการรวม การทดสอบย้อนกลับ และการทดสอบระบบ โดยระบบจะสร้างบทคำสั่งการทดสอบ เพื่อให้ผู้ทดสอบนำไปทดสอบโปรแกรมที่พัฒนา และนำผลการทดสอบมาทำการบันทึกเข้าสู่ระบบเพื่อที่ระบบจะนำผลการทดสอบมาสรุป และออกรายงานผลการทดสอบต่อไป ผู้ใช้สามารถเพิ่ม-ลด หรือแก้ไขข้อมูลกรณีทดสอบที่ระบบบันทึกไว้ได้ ระบบพลัสวันเทสสามารถสร้างบททดสอบโปรแกรมที่พัฒนาขึ้นด้วยภาษาซี และทำงานบนเครื่องซันเวิร์กสเตชัน (SUN Workstation) บนระบบปฏิบัติการโซลาริส (Solaris 1.x and 2.x)

ในขณะนี้ยังไม่มีเครื่องมือที่สามารถช่วยผู้ทดสอบในการสร้างกรณีทดสอบด้วยวิธีการแบบแบล็คบ็อก ผู้วิจัยจึงได้พัฒนาเครื่องมือที่สามารถช่วยในการสร้างกรณีทดสอบแบบแบล็คบ็อกขึ้นเพื่อช่วยลดภาระของผู้ทดสอบ