บทที่ 2
งานวิจัยและทฤษฎีที่เกี่ยวข้อง

2.1 งานวิจัยที่เกี่ยวข้อง

1) TestDirector 6 [1]

ถูกพัฒนาโดย บริษัท Mercury Interactive เป็นเครื่องมือที่ช่วยองค์กรในการทดสอบซอฟต์แวร์ทุกขั้นตอน โดยเริ่มจากการวางแผนการทดสอบ การดำเนินการทดสอบ และการจัดการข้อผิดพลาดที่เกิดขึ้น ในขั้นตอนการวางแผนการทดสอบจะช่วยในการสร้างแผนการทดสอบที่เป็นลำดับขั้นเพื่อให้เข้าใจได้ง่ายและมองเห็นขอบเขตชัดเจน ผู้ใช้สามารถสร้างแผนการทดสอบในไมโครซอฟต์เวิร์ด (Microsoft Word) และสามารถนำเข้ามาใช้ในเครื่องมือนี้ได้ทันที ในส่วนของการดำเนินการทดสอบ เครื่องมือดังกล่าวสามารถให้ผู้ใช้ทดสอบได้ทั้งแบบอัตโนมัติโดยที่เครื่องมือจัดสรรไว้ หรือให้ผู้ทดสอบดำเนินการเองก็ได้ แล้วรายงานผลการทดสอบกลับมายังเครื่องมือ หลังจากนั้นหากเกิดข้อผิดพลาด เครื่องมือมีความสามารถในการให้ข้อมูลที่เป็นประโยชน์ในการวิเคราะห์ข้อผิดพลาดเพื่อหาแนวทางแก้ไข นอกจากนี้เครื่องมือยังสามารถออกรายงานสรุปผลการทดสอบและให้ผู้ใช้กำหนดรูปแบบของรายงานได้อีกด้วย

เครื่องมือดังกล่าวยังไม่มีความสามารถในการจัดกำหนดการทดสอบที่ให้ผู้ใช้สามารถควบคุมขั้นตอนต่าง ๆ ในการทดสอบให้เป็นไปตามแผนที่กำหนดไว้

2) ExecuTest [2]

ถูกพัฒนาโดย บริษัท Cottonwood Technology, Inc. เป็นเครื่องมือที่ช่วยองค์กรในการวางแผนการทดสอบซอฟต์แวร์โดยกำหนดแม่แบบและขั้นตอนของแผนการทดสอบไว้ 4 ระดับเพื่อให้ผู้ใช้ทำตามทีละขั้นตอน ผู้ใช้สามารถกำหนดลำดับของการทดสอบและการจัดสรรทรัพยากรสำหรับแต่ละแผนการทดสอบได้ ในส่วนของการดำเนินการทดสอบ เครื่องมือสามารถสร้างสภาพแวดล้อมในการทดสอบโดยอิงกับภาษาโปรแกรมที่กำหนดไว้ได้ ซึ่งทำให้สามารถรายงานผู้ใช้ได้ว่า ขณะนี้ได้ทดสอบที่โมดูลใดและผลการทดสอบนั้นสำเร็จหรือต้องนำไปแก้ไข นอกจากนี้ เครื่องมือดังกล่าวยังสามารถจัดการกับผลลัพธ์ที่ได้จากการทดสอบได้เป็นอย่างมีประสิทธิภาพ ทำให้ลดความยุ่งยากและลดข้อผิดพลาดในการจัดการกับข้อมูลที่มีจำนวนมากได้

เครื่องมือดังกล่าวยังไม่สามารถรองรับความต้องการของผู้ใช้ในการกำหนดแม่แบบของแผนการทดสอบขึ้นมาเองได้ และไม่สามารถสร้างเอกสารเกี่ยวกับการทดสอบโดยอิงกับมาตรฐานที่มีอยู่ในปัจจุบัน เช่น มาตรฐาน IEEE Std.829-1998 ได้

2.2 ทฤษฎีที่เกี่ยวข้อง

ในงานวิทยานิพนธ์นี้ มีทฤษฎีต่าง ๆ ที่เกี่ยวข้องดังนี้

1) ขั้นตอนในการวางแผนการทดสอบ [3]

ในการวางแผนการทดสอบที่ระดับต่าง ๆ องค์กรจำเป็นต้องทำตามขั้นตอนเหล่านี้ ได้แก่

1.1) กำหนดจุดมุ่งหมายและขอบเขตของการทดสอบที่ระดับนั้น

1.2) ระบุความต้องการของผู้ใช้ในการทดสอบระดับนั้น

1.3) พยายามเชื่อมโยงความต้องการแต่ละระดับของการทดสอบเข้ากับส่วนของโปรแกรม ซึ่งอาจเป็นโมดูล หรือกลุ่มของโมดูล โดยอาจนำเอกสารในการออกแบบระบบเข้ามาช่วย เช่น แผนภูมิโครงสร้าง (Structure chart) เป็นต้น เพื่อให้ทราบโครงสร้างทั้งหมดของโปรแกรม

1.4) กำหนดกิจกรรมเพื่อรองรับการทดสอบส่วนของโปรแกรมนั้น อาจมีการแบ่งเป็นหลาย ๆ กิจกรรม หากส่วนของโปรแกรมมีความซับซ้อน

1.5) ระบุสิ่งที่ต้องใช้ในการทดสอบในแต่ละกิจกรรม ได้แก่ ฮาร์ดแวร์ ซอฟต์แวร์ (ได้แก่ ตัวขับ (driver) และตัวดำเนินการ (stub)) และผู้ทดสอบ

1.6) กำหนดวันที่เริ่มต้นและระยะเวลาที่ใช้ในการทดสอบของแต่ละกิจกรรม ซึ่งภายหลังจากขั้นตอนนี้ องค์กรสามารถสร้างกำหนดการทดสอบและแผนการทดสอบได้

1.7) สร้างกรณีทดสอบสำหรับแต่ละกิจกรรมนั้น แล้วเรียงลำดับกรณีทดสอบตามลำดับของการทดสอบ รวมทั้งสร้างตัวขับและตัวดำเนินการตามที่ได้กำหนดไว้

1.8) ดำเนินการทดสอบด้วยกรณีทดสอบตามที่ได้ออกแบบไว้

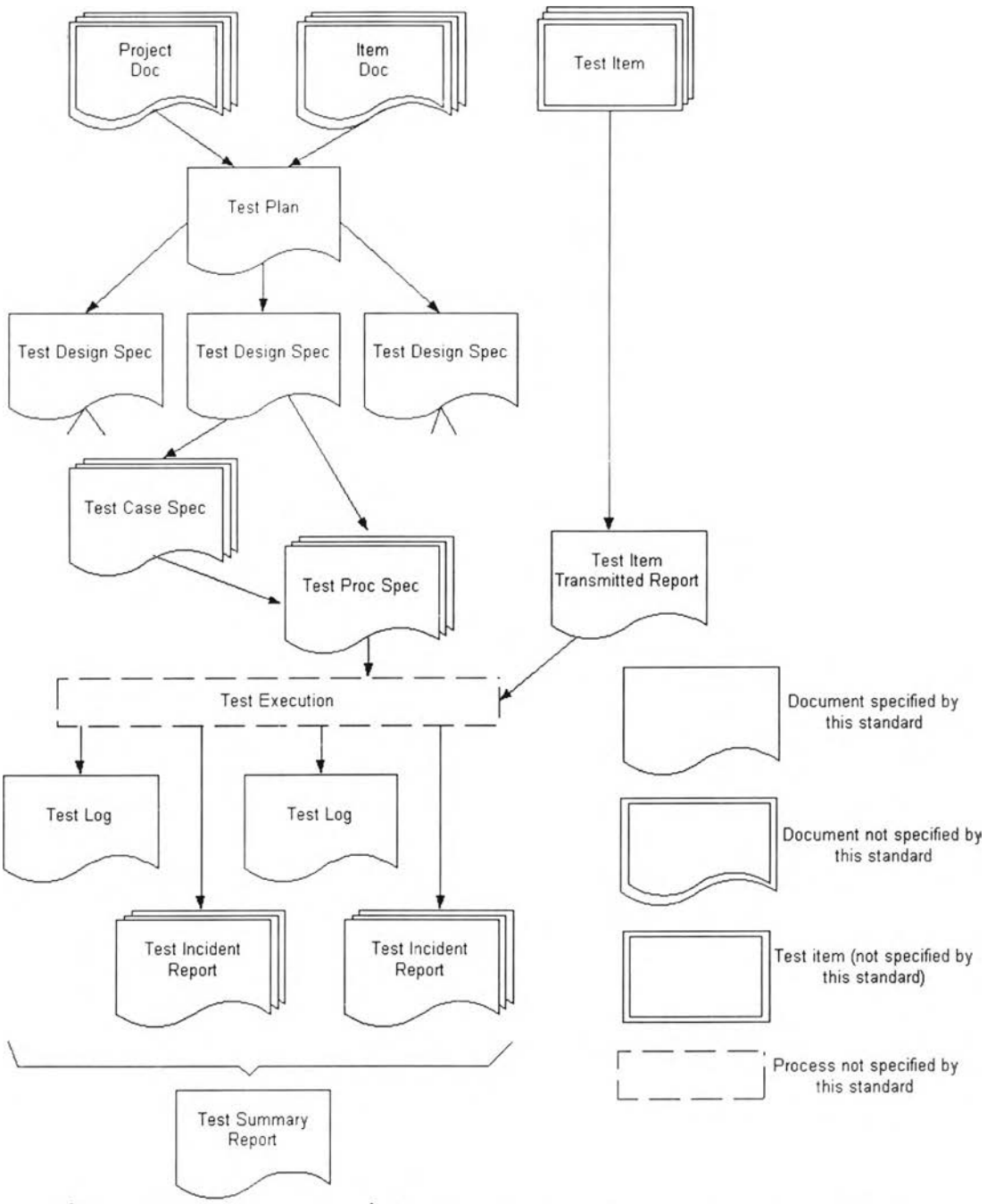
1.9) ตรวจสอบผลลัพธ์ที่ได้ว่าตรงกับที่ได้วางแผนไว้หรือไม่ หากเกิดข้อผิดพลาดต้องนำมาวิเคราะห์เพื่อหาสาเหตุและแนวทางแก้ไข เช่น การแก้ไขโปรแกรม (debug) เป็นต้น

1.10) สรุปผลการทดสอบ

2) มาตรฐาน IEEE Std 829-1998 [4]

เป็นมาตรฐานที่เกี่ยวกับเอกสารที่ใช้ในการทดสอบ โดยมาตรฐานนี้อธิบายเกี่ยวกับจุดมุ่งหมาย แนวทาง และองค์ประกอบของเอกสารเบื้องต้นที่ใช้ในการทดสอบซอฟต์แวร์ ดังแสดงในรูปที่ 2.1 เพื่อให้เกิดความเข้าใจตรงกันระหว่างผู้พัฒนาและลูกค้า อีกทั้งยังทำให้การทดสอบควบคุมได้ เนื่องจากสามารถตรวจสอบการทำงานในแต่ละขั้นตอนจากเอกสารได้ ซึ่งมาตรฐานนี้ถูกออกแบบมาโดยไม่ขึ้นกับประเภทซอฟต์แวร์ ขนาดของคอมพิวเตอร์ วิธีการทดสอบ หรือเครื่องมือที่ช่วยในการทดสอบต่าง ๆ และสามารถนำมาใช้ได้ในทุกระดับของการทดสอบ มาตรฐานนี้ประกอบด้วยส่วนต่าง ๆ ดังต่อไปนี้

2.1) การวางแผนการทดสอบ (Test Planning) ระบุถึงขอบเขต แนวทาง ทรัพยากร กำหนดการของการทดสอบ ระบุส่วนของโปรแกรม ลักษณะ (feature) ที่จะทดสอบ กิจกรรมที่ต้องทำ ความรับผิดชอบ และความเสี่ยงที่จะเกิดขึ้น



รูปที่ 2.1 ความสัมพันธ์ของเอกสารที่ใช้ในการทดสอบตามมาตรฐาน IEEE Std.829-1998[4]

2.2) รายละเอียดของการทดสอบ (Test Specification) ประกอบด้วยเอกสารต่าง ๆ ได้แก่

2.2.1) รายละเอียดเกี่ยวกับการออกแบบการทดสอบ (Test-design Specification) ระบุการออกแบบการทดสอบเพื่ออธิบายลักษณะที่จะทดสอบที่ได้กำหนดไว้ในการวางแผนการทดสอบ นอกจากนั้นยังระบุถึงกรณีทดสอบที่เกี่ยวข้องและหลักในการพิจารณาการทดสอบในแต่ละกรณีทดสอบว่าผ่านหรือไม่ผ่านเกณฑ์ที่กำหนด (pass/fail criteria)

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

2.2.3) ขั้นตอนการทดสอบ (Test Procedure Specification) ระบุขั้นตอนทั้งหมดที่ต้องทำในการจัดการทดสอบโดยใช้กรณีทดสอบที่ได้ออกแบบไว้

2.3) การรายงานผลการทดสอบ (Test Reporting) ประกอบด้วยเอกสารต่าง ๆ ดังนี้

2.3.1) รายงานส่วนของโปรแกรมที่ถูกส่งไปทดสอบ (Test item transmitted report)

2.3.2) บันทึกการทดสอบ (Test log) บันทึกผลลัพธ์ที่ได้จากการทดสอบโดยใช้กรณีทดสอบต่าง ๆ

2.3.3) รายงานการทดสอบที่ต้องดำเนินการต่อ (Test incident report) ระบุถึงเหตุการณ์ที่เกิดขึ้นในระหว่างการทดสอบ

2.3.4) รายงานสรุปผลการทดสอบ (Test summary report) สรุปกิจกรรมที่เกิดขึ้นตามที่ได้ระบุไว้ในรายละเอียดเกี่ยวกับการออกแบบการทดสอบ

3) แนวคิดเกี่ยวกับการทดสอบ [5,6]

3.1) การทดสอบระดับหน่วย

เป็นการตรวจสอบการทำงานของแต่ละโมดูล (module) แต่เนื่องจากทุก ๆ โมดูลถูกทดสอบแยกจากกัน ทำให้ในบางโมดูลจำเป็นต้องใช้โปรแกรมที่ถูกเขียนขึ้นมาเฉพาะ นั่นคือ ตัวขับ และตัวดำเนินการ โดยตัวขับเป็นโปรแกรมที่ใช้จำลองการเรียกโมดูลที่ต้องการทดสอบพร้อมกับให้ข้อมูลเข้า (input) และรับผลลัพธ์ (output) จากการทำงานของโมดูลนั้น และตัวดำเนินการเป็นโปรแกรมจำลองที่โมดูลที่จะทดสอบนั้นต้องการเรียกใช้ ซึ่งควรจะเป็นโปรแกรมที่ง่ายและมีขนาดเล็ก

3.2) การทดสอบการรวม

เป็นการตรวจสอบข้อผิดพลาดที่เกิดจากการรวมโมดูลเข้าด้วยกัน และเป็นการตรวจสอบความเข้ากันได้ระหว่างโมดูล โดยมีแนวความคิดในการรวมโมดูลดังนี้คือ

3.2.1) การรวมจากบนลงล่าง (Top-down approach) โดยเริ่มจากการทดสอบโมดูลที่อยู่ระดับบนสุดในแผนภาพโครงสร้างก่อน ซึ่งจำเป็นต้องมีการสร้างตัวดำเนินการเพื่อให้โมดูลที่อยู่ด้านบนสามารถเรียกใช้ได้ แนวทางในการรวมจากบนลงล่างมีอยู่ 2 วิธีได้แก่ ไปทางแนวกว้างก่อน (breadth-first) และไปทางแนวลึกก่อน (depth-first)

3.2.2) การรวมจากล่างขึ้นบน (Bottom-up approach) โดยเริ่มจากการทดสอบโมดูลที่อยู่ในระดับต่ำสุดในแผนภาพโครงสร้างก่อน ซึ่งจำเป็นต้องมีการสร้างตัวขับเพื่อใช้เรียกและส่งผ่านข้อมูลไปยังโมดูลที่ทำการทดสอบ

3.2.3) การรวมแบบแซนด์วิช (Sandwich approach) เป็นการนำข้อดีของการรวมจากบนลงล่างมาใช้รวมโมดูลในระดับบน และการรวมจากล่างขึ้นบนมาใช้รวมโมดูลในระดับล่าง

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

3.2.5) การรวมแบบบิ๊กแบง (Big-bang approach) เป็นการรวมทุกโมดูลเข้าด้วยกันในครั้งเดียว ซึ่งมีข้อดีคือไม่จำเป็นต้องสร้างตัวขับหรือตัวดำเนินการแต่ไม่สามารถหาสาเหตุของข้อ

ผิดพลาดได้ว่าเกิดจากโมดูลใด ทำให้วิธีนี้ไม่นิยมนำมาใช้ในการทดสอบการรวมในองค์กรต่าง ๆ

3.3) การทดสอบระบบ

เกิดขึ้นภายหลังจากทุก ๆ โมดูลได้ถูกรวมเข้าด้วยกันแล้ว โดยการทดสอบระดับนี้จะเน้นไปที่ข้อกำหนดและความต้องการของผู้ใช้เป็นหลัก สามารถแบ่งออกเป็นประเภทต่าง ๆ ได้ดังนี้

3.3.1) การทดสอบตามหน้าที่ (Functional) โดยพยายามทดสอบความต้องการของผู้ใช้ทั้งหมดที่ได้กำหนดไว้

3.3.2) การทดสอบอื่น ๆ (Non-Functional) ประกอบด้วย

3.3.2.1) การทดสอบความกดดัน (Stress testing) โดยทดสอบระบบภายใต้สภาพการใช้งานที่สูงมากๆ มีจุดประสงค์เพื่อต้องการทราบความสามารถสูงสุดของระบบภายใต้ทรัพยากรที่กำหนด เช่น จำนวนรายการที่เกิดขึ้น (Transaction) มากที่สุดที่ระบบสามารถรองรับได้ เป็นต้น

3.3.2.2) การทดสอบประสิทธิภาพ (Performance testing) เพื่อแสดงว่าระบบมีประสิทธิภาพตามที่ได้ระบุไว้ และเพื่อปรับแต่งระบบให้มีประสิทธิภาพดีขึ้น โดยตรวจสอบจากเวลาตอบสนอง (response time) ปริมาณงาน (throughput) การใช้งานหน่วยประมวลผลกลาง (CPU utilization) เป็นต้น

3.3.2.3) การทดสอบส่วนหลัง (Background testing) เพื่อแสดงความสามารถของระบบในกรณีที่รองรับรายการมากกว่า 1 รายการในเวลาเดียวกัน

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

3.3.2.5) การทดสอบการกู้ระบบ (Recovery testing) เพื่อแสดงว่าระบบสามารถกู้ข้อมูลกลับคืนมาได้หากระบบพัง

3.3.2.6) การทดสอบความปลอดภัย (Security testing) เพื่อแสดงว่ากลไกในการรักษาความปลอดภัยของระบบมีประสิทธิภาพเพียงพอที่จะป้องกันระบบจากการลักลอบใช้งาน

4) เทคนิคที่ใช้ในการออกแบบกรณีทดสอบ

ในการทดสอบระดับต่าง ๆ มีเทคนิคที่ใช้ออกแบบกรณีทดสอบ 2 วิธี ได้แก่

4.1) การทดสอบแบล็กบอกซ์ (Black-box Testing) สนใจเฉพาะข้อมูลเข้าและผลลัพธ์ของโมดูลที่ต้องการทดสอบ เพื่อให้ตรงตามข้อกำหนด (specification) ที่ได้ออกแบบไว้เท่านั้น ซึ่งจะไม่พิจารณาถึงโครงสร้างภายในโมดูลนั้น

4.2) การทดสอบไวท์บอกซ์ (White-box Testing) สนใจที่รายละเอียดการทำงานของโมดูลที่ต้องการทดสอบ ซึ่งจำเป็นต้องทราบโครงสร้างของโมดูลและพยายามทดสอบทุก ๆ เส้นทาง (path) ที่เป็น

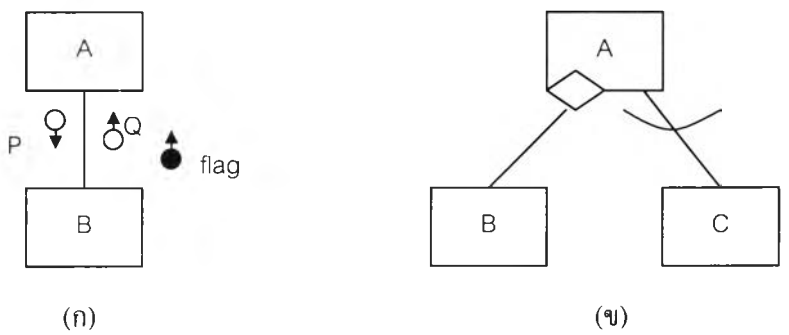
ไปได้ โมดูลที่ผ่านการทดสอบโดยใช้วิธีนี้จะมีประสิทธิภาพมากกว่า เนื่องจากข้อผิดพลาดต่าง ๆ ได้ถูกค้นพบทั้งหมด นอกจากนี้ยังสามารถค้นพบข้อผิดพลาดที่เกิดขึ้นจากข้อมูลภายในโมดูลได้อีกด้วย

5) แผนภาพโครงสร้าง (Structure Chart)

เป็นแผนภาพแสดงโครงสร้างการควบคุมของแต่ละโมดูลอย่างเป็นลำดับชั้น โดยจะแสดงให้เห็นการสื่อสารที่มีต่อกันระหว่างโมดูลในลักษณะที่โมดูลในระดับบนส่งข้อมูลไปให้โมดูลในระดับต่ำกว่าทำงาน และทำงานจากซ้ายไปขวา ซึ่งในการสร้างแผนภาพโครงสร้างนั้น มักนำแผนภาพการไหลของข้อมูล (Data Flow Diagram) แปลงให้เป็นแผนภาพโครงสร้าง [7]

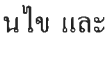
สัญลักษณ์ที่ใช้แสดงในแผนภาพโครงสร้าง ประกอบด้วย

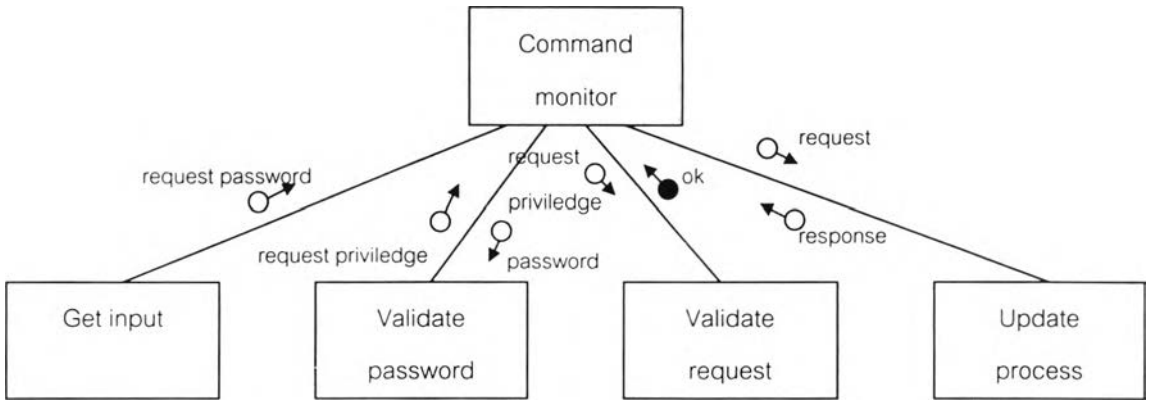
- โมดูล เป็นส่วนของโปรแกรมที่ทำงานเฉพาะอย่าง
- เส้นเชื่อมความสัมพันธ์ (Control Relations) ใช้แสดงความสัมพันธ์ระหว่างโมดูลหนึ่งกับโมดูลอื่น ๆ
- ข้อมูลที่รับ-ส่งระหว่างโมดูล (Module Coupling)
- เส้นเชื่อมโครงสร้างควบคุม (Control Structure) แสดงโครงสร้างการควบคุมของโมดูลหนึ่งกับโมดูลอื่น ๆ ซึ่งแบ่งออกเป็น 3 แบบ คือ
  - การเรียงลำดับ (Sequence) แสดงลำดับการทำงานของโมดูล
  - การเลือก (Selection) แสดงว่าโมดูลนั้นจะถูกทำงานเมื่อตรงตามเงื่อนไขที่กำหนด
  - การทำงานซ้ำ (Iteration) แสดงการที่โมดูลนั้นถูกทำงานหลายครั้ง



(ก) (ข)

รูปที่ 2.2 ตัวอย่างสัญลักษณ์ที่ใช้ในแผนภาพโครงสร้าง

จากตัวอย่างในรูปที่ 2.2 ก สังเกตได้ว่า โมดูล A มีการเรียกใช้โมดูล B โดยส่งข้อมูล P เมื่อต้องการเรียกใช้ เมื่อโมดูล B ทำงานเสร็จจะส่งผลลัพธ์กลับมายังโมดูล A ซึ่งประกอบด้วย ข้อมูล Q และ flag และจากตัวอย่างในรูปที่ 2.2 ข สังเกตได้ว่า โมดูล B จะทำงานเมื่อตรงตามเงื่อนไขที่กำหนด โดยที่สัญลักษณ์  แสดงถึงเงื่อนไข และโมดูล C จะถูกทำงานหลายครั้ง โดยที่สัญลักษณ์  แสดงถึงการทำงานซ้ำ รูปที่ 2.3 แสดงตัวอย่างแผนภาพโครงสร้างของการตรวจสอบผู้ใช้ระบบ โดยมี  และ  แสดงถึงข้อมูลที่รับส่งระหว่างโมดูล



รูปที่ 2.3 ตัวอย่างแผนภาพโครงสร้าง

6) การบริหารโครงการ [8,9]

การบริหาร (Management) เป็นการระบุเกี่ยวกับกิจกรรมต่าง ๆ ที่ต้องทำและมอบหมายไปยังบุคคลหรือกลุ่มบุคคลเพื่อทำให้กิจกรรมนั้นสำเร็จลุล่วง โดยมีจุดประสงค์หลักในการบรรลุเป้าหมายที่ได้กำหนดไว้ ซึ่งไม่สามารถทำให้เสร็จได้โดยคนเพียงคนเดียว

โครงการ (Project) เป็นงานที่เป็นเอกเทศ แน่นอน และเจาะจง ซึ่งจะต้องถูกดำเนินการให้แล้วเสร็จ มีเป้าหมายเพื่อให้บรรลุผลสำเร็จในตอนท้ายของโครงการซึ่งระบุไว้โดยแน่ชัด สามารถแบ่งโครงการออกเป็นงานย่อย ๆ ซึ่งแต่ละงานย่อยจะต้องทำตามลำดับก่อนหลัง เพื่อให้บรรลุเป้าหมายของโครงการ โครงการจะมีความสลับซับซ้อน จึงจำเป็นต้องมีการประสานงานและควบคุมงานย่อยต่าง ๆ เหล่านั้นในแง่ของการกำหนดเวลา การจัดลำดับก่อนหลัง การจัดสรรงบประมาณและบุคลากร เพื่อให้สามารถดำเนินการไปได้ด้วยดี

การบริหารโครงการ (Project Management) เป็นการกำหนดขั้นตอนวิธี (procedure) เทคนิคต่างๆ และวิธีการนำไปใช้ (know-how) ที่จำเป็นในการบริหารโครงการให้สำเร็จลุล่วงไปได้ การนำไปใช้ หมายถึงการนำทักษะต่าง ๆ (skills) พื้นความรู้ (background) และภูมิปัญญา (wisdom) ที่มีเข้ามาประยุกต์ใช้ได้

7) การจัดทำกำหนดการของโครงการ (Scheduling) [9]

กำหนดการของโครงการเป็นสิ่งที่สร้างขึ้นจากแผนการดำเนินงาน โดยกำหนดให้อยู่ในรูปของตารางเวลา เพื่อใช้ในการกำกับดูแลและควบคุมการทำงานของโครงการให้สำเร็จลุล่วงไปได้

เทคนิคการจัดทำกำหนดการของโครงการจะมีวิธีการพื้นฐานคล้ายคลึงกัน คือ การจัดสร้างผังข่ายงาน (Network of Activity) เพื่อแสดงให้เห็นถึงงานต่าง ๆ และความสัมพันธ์ระหว่างงานเหล่านั้น มีประโยชน์ดังนี้

- แสดงความสัมพันธ์ของงานทั้งหมดในโครงการ
- กำหนดวันที่คาดว่าโครงการจะแล้วเสร็จ
- แสดงให้เห็นงานวิกฤต (Critical Activity) ซึ่งถ้าถูกดำเนินงานล่าช้าไปจากที่กำหนดจะทำให้โครงการเสร็จล่าช้าออกไป

- แสดงเวลายืดหยุ่นของงาน (Slack) เป็นเวลาที่งานสามารถทำให้ล่าช้าลงไปภายในช่วงระยะเวลาหนึ่งได้โดยไม่ก่อให้เกิดผลเสีย
- แสดงวันที่อาจจะเริ่มงานได้ (Early Start Time) และวันที่ต้องเริ่มงานนั้น (Late Start Time)
- แสดงงานที่สามารถดำเนินงานแบบคู่ขนานกันได้

เทคนิคในการจัดทำกำหนดการของโครงการที่ได้รับความนิยม ได้แก่ เพิร์ท (PERT : Program Evaluation and Review Technique) ซีพีเอ็ม (CPM : Critical Path Method) และแผนภูมิแกนต์ (Gantt Chart) เป็นต้น ซึ่งในงานวิจัยนี้ จะนำเทคนิคของแผนภูมิแกนต์เข้ามาประยุกต์ใช้ ซึ่งจะกล่าวในรายละเอียดต่อไป

7.1) เพิร์ท และ ซีพีเอ็ม

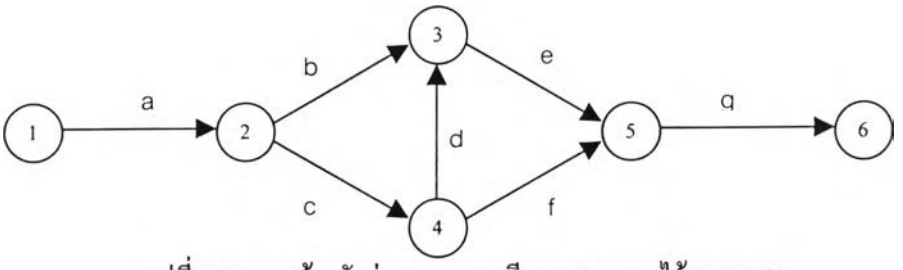
เพิร์ท ได้รับการพัฒนาขึ้นมาโดยกองทัพเรือสหรัฐอเมริกา และความร่วมมือของบริษัทเอกชนเพื่อใช้ในโครงการขีปนาวุธ ในช่วงปี พ.ศ.2501 ซีพีเอ็ม ได้รับการพัฒนาจากบริษัท ดูปองต์ (DuPont, Inc.) ในช่วงเวลาใกล้เคียงกัน ทั้ง 2 วิธีนี้มีลักษณะคล้ายคลึงกัน โดยเพิร์ทจะมุ่งเน้นในเรื่องของเวลาของโครงการและใช้ค่าคาดคะเนเวลาเป็นตัวกำหนดวันสิ้นสุดของโครงการ ส่วนซีพีเอ็มจะมีลักษณะที่ตรงข้าม คือจะใช้ค่าคาดคะเนเวลาของงานที่รู้ค่าแน่นอน และถูกออกแบบมาเพื่อให้ควบคุมทั้งเวลาและค่าใช้จ่าย

ในการสร้างผังข่ายงาน จะมีนิยามของคำต่าง ๆ เหล่านี้ ได้แก่

- งานหรือกิจกรรม (activity) เป็นงานหรือชุดของงานในโครงการ ซึ่งต้องใช้ทรัพยากรและเวลาที่จะทำให้สำเร็จ
- เหตุการณ์ (event) เป็นสถานะของการเริ่มต้นหรือเสร็จสิ้นงานหนึ่งหรือหลายงาน
- วิถี (path) เป็นชุดของงานที่ต่อเนื่องกันระหว่าง 2 เหตุการณ์ใด ๆ ในผังข่ายงาน
- วิถีวิกฤต (critical path) เป็นชุดของงานซึ่งหากถูกทำให้ล่าช้า จะมีผลทำให้โครงการเสร็จล่าช้าลงไปด้วย
- ผังข่ายงาน (network) เป็นแผนภาพที่แสดงงานทั้งหมดของโครงการ โดยแทนเหตุการณ์ต่างๆ ด้วยวงกลม มีลูกศรเชื่อมโยงระหว่างงาน ซึ่งแสดงถึงความสัมพันธ์ก่อนหลังของงานเหล่านั้น งานทุกงานที่อยู่หน้าเหตุการณ์จะต้องทำให้เสร็จลุล่วงไปก่อน จึงจะสามารถทำงานถัดไปได้

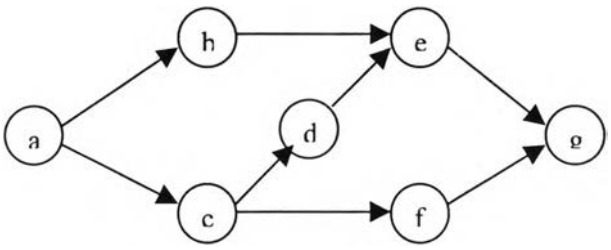
การสร้างผังข่ายงาน มีอยู่ 2 วิธีคือ

ก. เขียนแสดงงานไว้บนลูกศร (AOA : Activity-On-Arc) ดังตัวอย่างในรูปที่ 2.4



รูปที่ 2.4 การสร้างผังข่ายงานแบบเขียนแสดงงานไว้บนลูกศร

ข. เขียนแสดงงานไว้บนโหนด (AON : Activity-On-Node) ดังตัวอย่างในรูปที่ 2.5

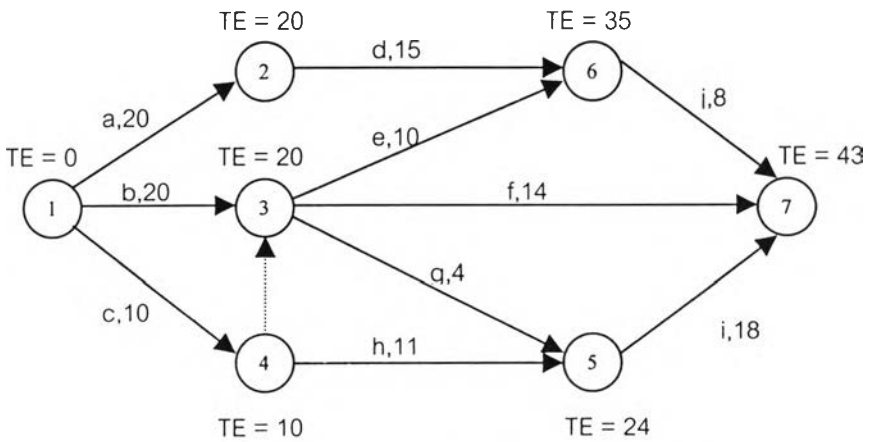


รูปที่ 2.5 การสร้างผังข่ายงานแบบเขียนแสดงงานไว้บนโหนด

การสร้างผังข่ายงาน จำเป็นต้องทราบรายละเอียดของงานอย่างน้อย 2 ประเภท คือ เวลาที่งานนั้นจะแล้วเสร็จ และงานที่อยู่ก่อนหน้างานนั้น ดังตัวอย่างในตารางที่ 2.1

ตารางที่ 2.1 ตัวอย่างการจัดสร้างผังข่ายงาน

งาน	เวลา	งานที่อยู่ก่อนหน้า
a	20	-
b	20	-
c	10	-
d	15	a
e	10	b,c
f	14	b,c
g	4	b,c
h	11	c
i	18	g,h
j	8	d,e



รูปที่ 2.6 ผังข่ายงานที่สอดคล้องกับตารางที่ 2.1

จากรูปที่ 2.6 สังเกตได้ว่า เมื่อเริ่มโครงการ สามารถทำงาน a,b และ c ได้พร้อม ๆ กัน โดยมี TE (Earliest occurrence time) เป็นเวลาที่เร็วที่สุดที่สามารถเกิดเหตุการณ์นั้นหลังจากเริ่มโครงการ และในเหตุการณ์ที่ 3 จะมีงานก่อนหน้า 2 งานคืองาน b และ c จึงจะเกิดงาน e,f และ g ได้ ดังนั้น จึงจำเป็นต้องสร้างงานจำลอง (dummy) ขึ้นมา ซึ่งแสดงด้วย ------➤ จากเหตุการณ์ที่ 4 ไปยัง 3 เพื่อให้ลำดับของงานยังคงอยู่ ซึ่งไม่ต้องใช้เวลาหรือทรัพยากรต่าง ๆ

สังเกตได้ว่า ในเหตุการณ์ที่ 6 มีงานก่อนหน้า 2 งานคือ งาน d และงาน e งาน d จะเริ่มได้ในวันที่ 20 ใช้เวลา 15 วันรวมเป็น 35 วัน ส่วนงาน e จะเริ่มได้ในวันที่ 20 ใ ช้เวลา 10 วันรวมเป็น 30 วัน เนื่องจากเหตุการณ์ที่ 6 ต้องทำงาน d และ e ให้เสร็จก่อน ดังนั้น TE จึงมีค่าเท่ากับ 35 วัน ซึ่งเป็นเวลาที่ทั้งงาน d และ e ทำเสร็จ

การพิจารณาวิถีวิกฤต จะพิจารณาจากเหตุการณ์ที่เป็นจุดเสร็จสิ้นของผังข่ายงาน นั่นคือเหตุการณ์ที่ 7 ซึ่งใช้เวลา 43 วัน มีวิถีของงาน 5 วิถีด้วยกัน ดังแสดงในตารางที่ 2.2

ตารางที่ 2.2 การคำนวณวิถีวิกฤต

วิถี	เวลาที่ใช้
a-d-j	**ใช้เวลา 20 + 15 + 8 = 43 วัน**
b-e-j	ใช้เวลา 20 + 10 + 8 = 38 วัน
b-f	ใช้เวลา 20 + 14 = 34 วัน
b-g-I	ใช้เวลา 20 + 4 + 18 = 42 วัน
c-h-I	ใช้เวลา 10 + 11 + 18 = 39 วัน

จากตารางที่ 2.2 พบว่า วิถีที่ใช้เวลานานที่สุดคือ a-d-j ซึ่งใช้เวลาทั้งสิ้น 43 วัน ซึ่งเป็นวันที่คาดว่าโครงการจะแล้วเสร็จ หากไม่เกิดการล่าช้าใด ๆ ขึ้นกับวิถีวิกฤตนี้

ในการพิจารณางานที่ไม่ได้อยู่ในวิถีวิกฤต ว่าสามารถเริ่มต้นช้ากว่ากำหนดได้เท่าไร โดยไม่ทำให้โครงการทั้งหมดล่าช้าออกไป จะใช้ 2 ค่ามาพิจารณาคือ เวลาเริ่มต้นเร็วที่สุด (EST : Earliest Start Time) และ เวลาเริ่มต้นช้าที่สุด (LST : Latest Start Time) โดยในการคำนวณค่า EST และ LST จะคำนวณแบบย้อนกลับ และจะถือว่างานที่อยู่ก่อนหน้างานนั้นไม่ได้เกิดการล่าช้ามาก่อน การคำนวณค่า EST และ LST สามารถพิจารณาได้ดังนี้

EST ของงาน A = ค่า TE ของเหตุการณ์ที่ทำให้เกิดงาน A

LST ของงาน A = เวลาวิกฤต – ระยะเวลาของงาน A - ผลรวมระยะเวลาของงานหลังจากงาน A จนสิ้นสุด

ตัวอย่างเช่น งาน i ต้องการเวลา 18 วันและโครงการนี้มีเวลาวิกฤต 43 วัน ดังนั้น งาน i จะต้องไม่เริ่มต้นช้าไปกว่าวันที่ 25 (43 – 18 = 25) ดังนั้นค่า LST สำหรับงาน i คือ 25 และงาน i ไม่สามารถเริ่มต้นได้จนกว่าเหตุการณ์ที่ 5 จะเกิดขึ้นซึ่งเป็นวันที่ 24 ซึ่งเป็นค่า EST ของงาน i

ผลต่างของ EST และ LST จะเรียกว่า เวลายืดหยุ่น (slack) สำหรับงาน i จะมีค่าเท่ากัน 1 วัน หมายความว่า งาน i สามารถถูกทำให้ล่าช้าลงได้ 1 วันโดยไม่ทำให้โครงการเสร็จล่าช้าออกไป และทุก ๆ งานในวิถีวิกฤตจะมีเวลายืดหยุ่นเป็น 0 แสดงว่าไม่สามารถทำงานให้ล่าช้ากว่าปกติได้

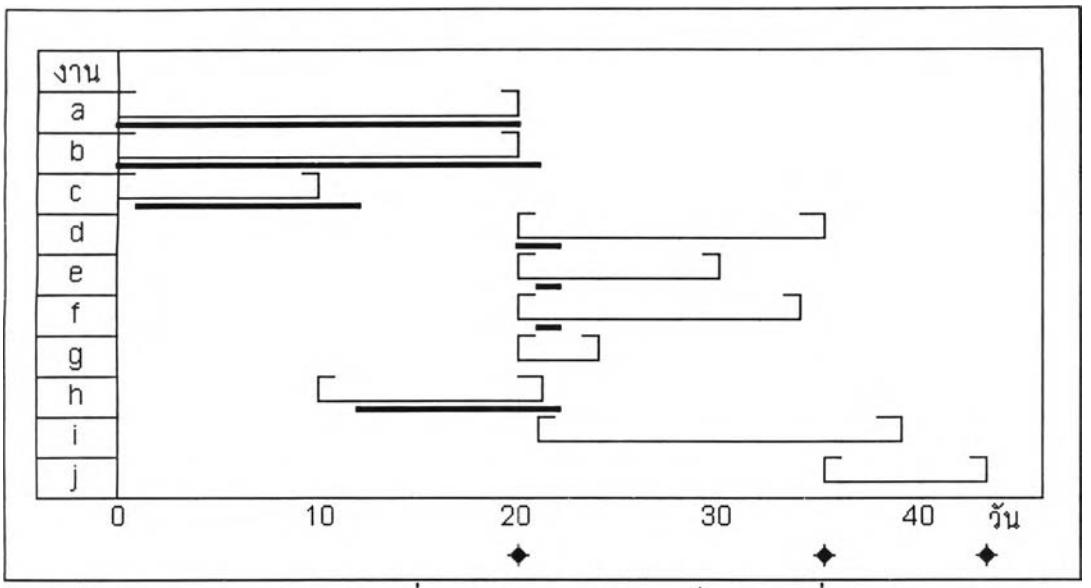
ตารางที่ 2.3 การคำนวณเวลายืดหยุ่นของแต่ละงานจากตัวอย่างในตารางที่ 2.1

งาน	เวลาเริ่มต้นช้าที่สุด	เวลาเริ่มต้นเร็วที่สุด	เวลายืดหยุ่น
a	0	0	0
b	1	0	1
c	4	0	4
d	20	20	0
e	25	20	5
f	29	20	9
g	21	20	11
h	14	10	4
i	25	24	1
j	35	35	0

7.2) แผนภูมิแกนต์ (Gantt Chart)

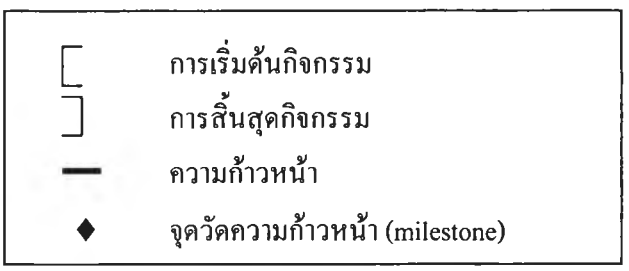
ถูกพัฒนาโดย เฮนรี แกนต์ ในระหว่างปี พ.ศ.2490 แผนภูมิแกนต์แสดงให้เห็นถึงความก้าวหน้าที่วางแผนไว้และเกิดขึ้นจริงของงานต่าง ๆ โดยนำเสนอเทียบกับแกนของเวลาในแนวนอน ซึ่งง่ายต่อการอ่าน นอกจากนี้ยังสามารถสร้างได้ง่าย ไม่จำเป็นต้องมีการร่างขึ้นมาก่อนเหมือน เพิร์ท/ซีพีเอ็ม

รูปที่ 2.7 แสดงตัวอย่างของแผนภูมิแกนต์ของงานในตารางที่ 2.1 สังเกตได้ว่า จุดวัดความก้าวหน้า 3 จุดจะเป็นเหตุการณ์ที่เกิดขึ้นบนวิถีวิกฤต ซึ่งวิธีการวัดความก้าวหน้าแบบนี้เป็นที่นิยมใช้โดยทั่วไป นอกจากนี้ แผนภูมิแกนต์ยังแสดงให้เห็นถึงความก้าวหน้าที่เกิดขึ้นจริงได้อีกด้วย โดยแสดงเป็นเส้นทึบใต้งานนั้น ๆ แต่แผนภูมิแกนต์จะไม่สามารถแสดงความสัมพันธ์ก่อนหลังของงานได้อย่างชัดเจน ตัวอย่างเช่น งาน d,e,f และ g ต่างเริ่มต้นวันที่ 20 แต่ไม่สามารถบอกได้ว่าแต่ละงานตามมาจากงานก่อนหน้างานใดบ้าง



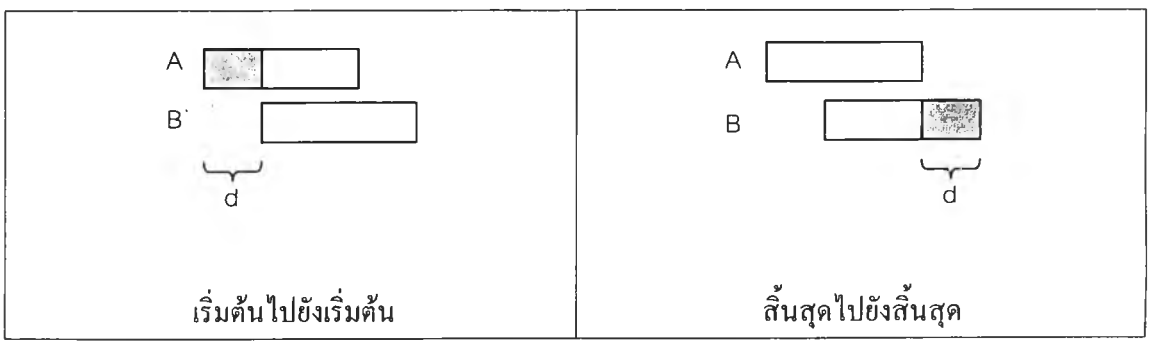
รูปที่ 2.7 ตัวอย่างแผนภูมิแกนต์จากตารางที่ 2.1

ความหมายของสัญลักษณ์

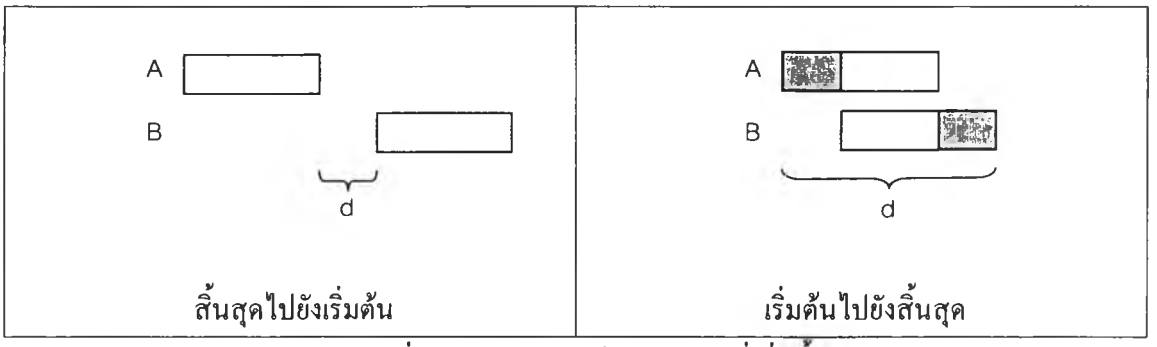


ความสัมพันธ์ระหว่างงานในปัจจุบันไม่ได้มีเพียงความสัมพันธ์แบบมีงานก่อนหน้าเท่านั้น ยังมีความสัมพันธ์ในรูปแบบอื่น ๆ ดังรูปที่ 2.8 ประกอบด้วย

- เริ่มต้นไปยังเริ่มต้น (SS : Start-to-Start) เช่น งาน B จะเริ่มต้นได้เมื่อได้ทำงาน A ไป d วันแล้ว
- สิ้นสุดไปยังสิ้นสุด (FF : Finish-to-Finish) เช่น งาน B จะเสร็จสิ้นได้เมื่อได้เสร็จสิ้นงาน A แล้ว d วัน
- สิ้นสุดไปยังเริ่มต้น (FS : Finish-to-Start) เช่น งาน B จะเริ่มต้นได้เมื่อเสร็จสิ้นงาน A แล้ว d วัน
- เริ่มต้นไปยังสิ้นสุด (SF : Start-to-Finish) เช่น งาน B จะเสร็จได้เมื่อเริ่มต้นงาน A แล้ว d วัน



รูปที่ 2.8 ความสัมพันธ์ระหว่างงานที่เพิ่มขึ้น



รูปที่ 2.8 ความสัมพันธ์ระหว่างงานที่เพิ่มขึ้น (ต่อ)

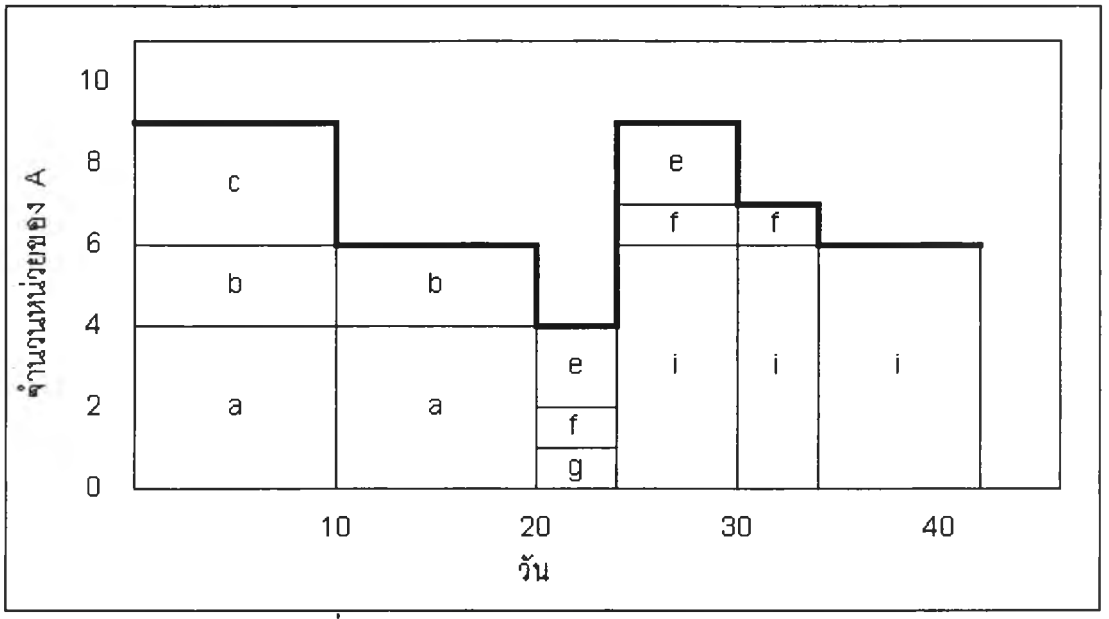
8) การจัดสรรทรัพยากร (Resource Allocation)

การจัดสรรทรัพยากรเป็นการวางแผนให้ความต้องการใช้ทรัพยากรตลอดทั้งโครงการให้อยู่ในระดับสม่ำเสมอ เพื่อหลีกเลี่ยงความล่าช้าของโครงการ และไม่ต้องเสียค่าใช้จ่ายกับทรัพยากรส่วนเกิน หากมีความต้องการใช้ทรัพยากรบางชนิดไม่สม่ำเสมอ จะมีผลทำให้ทรัพยากรต้องเสียเปล่าไปช่วงเวลาหนึ่งและขาดไปในอีกช่วงเวลาหนึ่ง ทำให้การใช้ทรัพยากรไม่คุ้มค่า

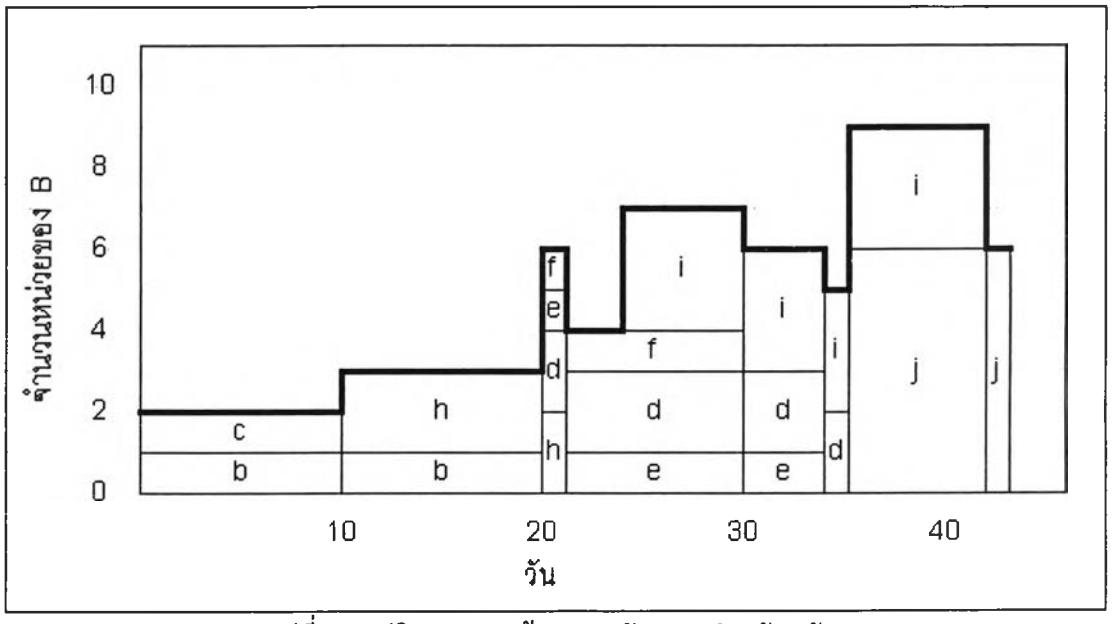
การหาปริมาณความต้องการใช้ทรัพยากร จะทำให้ทราบถึงปริมาณของทรัพยากรแต่ละชนิดที่ต้องการในช่วงระยะเวลาที่กำหนด เพื่อให้สามารถประมาณความต้องการทรัพยากรได้อย่างคร่าว ๆ ได้ในช่วงแรกของการจัดกำหนดการ ซึ่งปริมาณความต้องการใช้ทรัพยากรสามารถแสดงได้ในรูปกราฟแท่งแบบระดับ พิจารณาได้จากตัวอย่างในตารางที่ 2.4 ซึ่งแสดงถึงความต้องการทรัพยากรสำหรับผังข่ายงานในรูปที่ 2.6 ความต้องการทรัพยากรอาจมีหน่วยเป็น เครื่อง (ฮาร์ดแวร์) คน (บุคลากร) เป็นต้น สามารถแสดงเป็นกราฟแท่งแบบระดับได้ดังรูปที่ 2.9 และ 2.10

ตารางที่ 2.4 ตัวอย่างความต้องการทรัพยากรสำหรับผังข่ายงานในรูปที่ 2.6

งาน	ความต้องการทรัพยากร A	ความต้องการทรัพยากร B
a	4	0
b	2	1
c	3	1
d	0	2
e	2	1
f	1	1
g	1	0
h	0	2
i	6	3
j	0	6



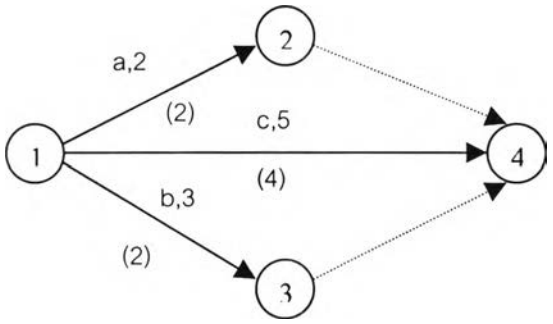
รูปที่ 2.9 ปริมาณความต้องการทรัพยากรสำหรับทรัพยากร A



รูปที่ 2.10 ปริมาณความต้องการทรัพยากรสำหรับทรัพยากร B

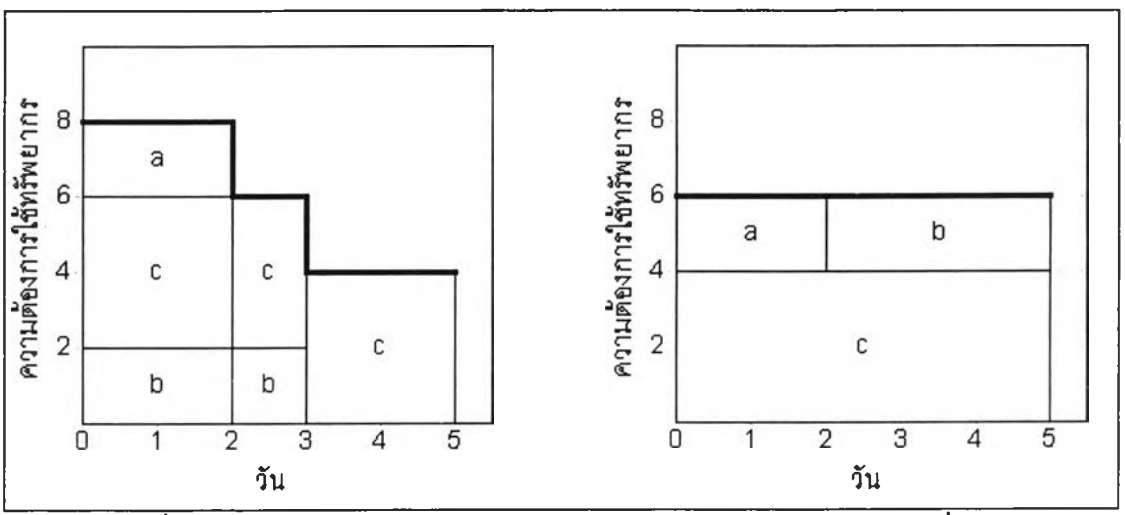
จากตัวอย่างในรูปที่ 2.9 และ 2.10 แสดงว่าความต้องการทรัพยากร A มีปริมาณสูงในช่วงแรก ต่อมาจะค่อย ๆ ลดต่ำลงและเพิ่มสูงขึ้นอีกครั้งในตอนท้าย และความต้องการทรัพยากร B มีปริมาณน้อยในช่วงแรก และ เพิ่มสูงขึ้นเรื่อง ๆ หากองค์กรต้องการให้มีทรัพยากรเพียงพอต่อการทำกิจกรรม องค์กรนั้นจะต้องจัดหาทรัพยากรให้มีค่าเท่ากับปริมาณสูงสุดของทรัพยากรที่ต้องการ ซึ่งจะทำให้การทำกิจกรรมบางช่วงไม่ได้ใช้ทรัพยากรนั้นอย่างคุ้มค่า จึงเกิดแนวคิดในการปรับระดับความต้องการทรัพยากร (Resource Leveling) เพื่อลดความแปรปรวนของความต้องการทรัพยากรในช่วงเวลาที่กำหนด ซึ่งมีวิธีคือเปลี่ยนแปลงกำหนดเวลาของกิจกรรมต่าง ๆ ให้อยู่ในเวลายืดหยุ่นของงานนั้น เพื่อทำให้การกระจายความ

ต้องการทรัพยากรสม่ำเสมอมากขึ้น มีข้อดีคือ ทรัพยากรต่าง ๆ จะถูกนำมาใช้เมื่อมีความต้องการในปริมาณที่คงที่ ทำให้ลดค่าใช้จ่ายลงได้มาก สามารถพิจารณาได้จากตัวอย่างผังข่ายงานในรูปที่ 2.11



รูปที่ 2.11 ตัวอย่างผังข่ายงานที่แสดงความต้องการใช้ทรัพยากร

จากรูปที่ 2.11 ตัวเลขที่อยู่ด้านข้างชื่อกิจกรรมจะแทนระยะเวลาของกิจกรรมนั้น และตัวเลขในวงเล็บด้านล่างของกิจกรรมจะแทนความต้องการใช้ทรัพยากรหนึ่งของกิจกรรมนั้น สามารถแสดงความต้องการใช้ทรัพยากรได้ดังรูปที่ 2.12



รูปที่ 2.12 ตัวอย่างการปรับระดับความต้องการทรัพยากรจากผังข่ายงานในรูปที่ 2.11

จากรูปที่ 2.12 สามารถเลื่อนงาน b ให้ช้าลงได้ 2 วัน ซึ่งเป็นเวลายืดหยุ่นของงาน b จะทำให้ปริมาณความต้องการมีความสม่ำเสมอมากขึ้น