บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 การประมาณค่าใช้จ่ายด้วยวิธีการโคโคโม2[3,4]

โคโคโม2 (COCOMO II) เป็นโมเดลที่พัฒนาต่อจากโคโคโม ซึ่งโคโคโมได้พัฒนาครั้งแรกในปี คศ. 1981 โดยการศึกษาโครงการซอฟต์แวร์ที่ได้รับการพัฒนาจนเสร็จสิ้นแล้วจำนวน 83 โครงการ แล้วนำมาวิเคราะห์ปัจจัยที่มีผลต่อค่าใช้จ่ายของการพัฒนาโครงการ ซึ่งสามารถสรุปได้ว่ามี 15 ปัจจัย หรือที่เรียกว่าตัวขับค่าใช้จ่าย(Cost Drivers) ที่มีผลต่อการพัฒนาโครงการ อย่างไรก็ตามทีมงานผู้พัฒนาโคโคโมได้เริ่มทำการพัฒนาและปรับปรุงอย่างต่อเนื่องเป็นโคโคโม2 ในปี คศ.1991 ซึ่งโคโคโม2โมเดลจะประกอบด้วย 3 โมเดลย่อยคือ แอปพลิเคชันคอมโพซิชัน(Application Composition Model) เออรีดีไซน์โมเดล(Early Design Model) และ โพสอาร์คิเท็กเจอร์โมเดล(Post-Architecture Model) ซึ่งในแต่ละโมเดลย่อยจะมีวิธีการประมาณค่าที่ต่างกันไปขึ้นอยู่กับว่าผู้ใช้สามารถให้ข้อมูลพื้นฐานได้ละเอียดมากน้อยเพียงใด

2.1.1.1 แบบจำลองแอปพลิเคชันคอมโพสซิชัน (Application Composition Model)

แบบจำลองแอปพลิเคชันคอมโพสซิชัน เป็นแบบจำลองที่ใช้ประมาณค่าใช้จ่ายในการพัฒนาซอฟต์แวร์อย่างหยาบๆ โดยพิจารณาจากส่วนที่ซอฟต์แวร์นั้นติดต่อกับผู้ใช้ (User Interface) ซึ่งในแบบจำลองนี้ผู้ใช้ทราบเพียง จำนวนรายงาน และ จำนวนหน้าจอ (Screens) ที่ใช้ในการรับข้อมูล(Input)และการแสดงผล (Output) เท่านั้น แล้วจึงนำข้อมูลเหล่านี้ไปคำนวณหาค่าออพเจ็คพอยท์ (Object Points) และ จำนวนคนที่ใช้พัฒนาซอฟต์แวร์ต่อเดือน ดังสมการที่ (1) และ (2)

$$NOP = \frac{(Object\ Points) * (100 - \%Reuse)}{100} \qquad \cdots(1)$$

$$PM = \frac{NOP}{PROD} \qquad \cdots(2)$$

โดย

NOP หมายถึง จำนวนออพเจ็คพอยท์ที่สร้างใหม่ (New Object Points)

%Reuse หมายถึง จำนวนเปอร์เซ็นต์ของหน้าจอ (Screens) รายงาน(Reports) และโมดุลของสามจีเอล(3GL modules) ที่นำกลับมาใช้

PM หมายถึง จำนวนคนที่พัฒนาต่อหนึ่งเดือน (Person Months)

PROD หมายถึง ค่าอัตราการผลิต ซึ่งจะขึ้นอยู่กับความสามารถและประสบการณ์ของผู้พัฒนา

ขั้นตอนการคำนวณหาค่าประมาณการมีดังนี้

1) การหาจำนวนออพเจ็คพอยท์

การหาจำนวนออพเจ็คพอยท์ โดยการนับจาก จำนวนหน้าจอ จำนวนรายงาน จำนวนองค์ประกอบของสามจีเอล (3GL Components) ที่มีในแอปพลิเคชันที่จะพัฒนา

2) การจัดกลุ่มให้กับออพเจ็คพอยท์

เมื่อได้จำนวนออพเจ็คพอยท์ทั้งหมดจากขั้นตอนแรกแล้ว จึงนำออพเจ็คพอยท์เหล่านั้นมาจัดแบ่ง ออกเป็นกลุ่มๆ คือ ง่าย (simple) ปานกลาง (medium) และยาก (difficult) โดยแบ่งตามเงื่อนไขดังตารางที่ 2.1

ตารางที่ 2.1 การแบ่งกลุ่มของออพเจ็คพอยท์

สำหรับหน้าจอ(Screens)				สำหรับรายงาน(Reports)			
จำนวนหน้าจอที่ปรากฏ	จำนวนแหล่งข้อมูลในตาราง (source of data tables)			จำนวนชนิดของรายงาน	จำนวนแหล่งข้อมูลในตาราง (source of data tables)		
	รวม <4 (<2 srvr <3 clnt)	รวม < 8 (2-3 srvr 3-5 clnt)	รวม > 8 (>3 srvr >5 clnt)		รวม <4 (<2 srvr <3 clnt)	รวม < 8 (2-3 srvr 3-5 clnt)	รวม > 8 (>3 srvr >5 clnt)
< 3	ง่าย	ง่าย	ปานกลาง	0 หรือ 1	ง่าย	ง่าย	ปานกลาง
3 – 7	ง่าย	ปานกลาง	ยาก	2 หรือ 3	ง่าย	ปานกลาง	ยาก
> 8	ปานกลาง	ยาก	ยาก	> 4	ปานกลาง	ยาก	ยาก

หมายเหตุ srvr หมายถึง ให้บริการ(Server) , clnt หมายถึง รับบริการ(Client)

3) การให้น้ำหนักกับออพเจ็คพอยท์ที่ได้จัดแบ่งกลุ่มจากข้อ 2)แล้ว โดยการให้น้ำหนักขึ้นอยู่กับกลุ่มที่แบ่งดังตารางที่ 2.2

ตารางที่ 2.2 การกำหนดน้ำหนักของออพเจ็คพอยท์

ชนิดของออพเจ็คพอยท์	การกำหนดน้ำหนัก		
	ง่าย	ปานกลาง	ยาก
หน้าจอ	1	2	3
รายงาน	2	5	8
จำนวนองค์ประกอบของสามจีเอล	-	-	10

4) นำจำนวนออพเจ็คพอยท์ทั้งหมดมาคูณกับน้ำหนักที่ได้ในแต่ละกลุ่ม แล้วนำมาบวกรวมกัน ก็จะได้ค่า ออพเจ็คพอยท์ ในสมการที่(1)

5) การหาค่าอัตราการผลิตสามารถหาได้จากตารางที่2-3

ตารางที่ 2.3 ค่าอัตราการผลิต

ประสบการณ์และความสามารถของผู้พัฒนา (Developers' experience and capability)	ต่ำมาก	ต่ำ	ปานกลาง	สูง	สูงมาก
ความสมบูรณ์และความสามารถของไอเคส (ICASE maturity and capability)	ต่ำมาก	ต่ำ	ปานกลาง	สูง	สูงมาก
ค่าอัตราการผลิต(Productivity Rate : PROD)	4	7	13	25	50

2.1.1.2 แบบจำลองเออรีดีไซด์ (Early Design Model)

แบบจำลองเออรีดีไซด์ เป็นแบบจำลองที่มีความละเอียดของการประมาณค่าใช้จ่ายในการพัฒนาซอฟต์แวร์มากกว่าแบบจำลองแรก เนื่องจากผู้ใช้ทราบข้อมูลเพียงคร่าวๆเท่านั้น เช่น ทราบจำนวนฟังก์ชันพอยท์ (function points) และ ภาษาที่ใช้พัฒนา แต่ในการประมาณค่าใช้จ่ายซอฟต์แวร์ ได้นำตัวแปรที่มีผลต่อการพัฒนาซอฟต์แวร์มาพิจารณาซึ่งเรียกว่า ตัวขับค่าใช้จ่าย (cost driver) 7 ลักษณะ เป็นข้อมูลพื้นฐานในการประมาณค่าใช้จ่ายซอฟต์แวร์ด้วย ซึ่งผู้ใช้ต้องเป็นผู้กำหนดระดับ(rating)ของตัวขับค่าใช้จ่ายทั้ง 7 ลักษณะ ดังนี้

1) ความเชื่อถือได้และความซับซ้อนของผลิตภัณฑ์ (Product Reliability and Complexity: RCPX) พิจารณาจาก ความเชื่อถือได้ของผลิตภัณฑ์ซอฟต์แวร์ที่ต้องการ ขนาดของฐานข้อมูลที่ใช้ในผลิตภัณฑ์ซอฟต์แวร์ ความซับซ้อนของผลิตภัณฑ์ซอฟต์แวร์ และ ความต้องการเอกสารที่ตรงกับวงจรชีวิต

2) ความต้องการนำกลับมาใช้(Required Reuse: RUSE) พิจารณาจากการออกแบบเพื่อมีการเตรียมนำกลับมาใช้ใหม่

3) แพลทฟอร์มที่แตกต่างกัน(Platform Difficulty: PDIF) พิจารณาจาก การจำกัดเวลาที่ใช้ในการประมวลผล, ขนาดของหน่วยความจำหลัก และ การเปลี่ยนแพลทฟอร์ม

4) ความสามารถของบุคลากร(Personnel Capability: PERS) พิจารณาจาก ความสามารถในการวิเคราะห์, ความสามารถของโปรแกรมเมอร์ และ การทำงานอย่างต่อเนื่องของบุคลากร

5) ประสบการณ์ของบุคลากร(Personnel Experience: PREX) พิจารณาจาก ประสบการณ์การใช้แอปพลิเคชั่น, ประสบการณ์การใช้แพลทฟอร์ม และ ประสบการณ์การใช้โปรแกรมภาษาและเครื่องมือ

6) อุปกรณ์อำนวยความสะดวก(Facilities: FCIL) พิจารณาจาก การใช้เครื่องมือซอฟต์แวร์ และ การพัฒนาในหลายสถานที่

7) กำหนดการการพัฒนา (Schedule: SCED) พิจารณาจาก กำหนดการการพัฒนาที่ต้องการ

การประมาณค่าใช้จ่ายในการพัฒนาซอฟต์แวร์ในแบบจำลองนี้ จะประมาณระยะเวลาที่ใช้ในการพัฒนาซอฟต์แวร์ จำนวนคนที่ใช้พัฒนาต่อเดือน เป็นต้น ซึ่งมีสมการสำหรับประมาณการดังสมการที่ (3) ถึง (10) และคำอธิบายในตารางที่ 2.4

สูตรคำนวณจำนวนคนที่ใช้พัฒนาต่อเดือนคือ

$$PM = A*[Size]^{B}\prod_{i=1}^{7} EM_i + PM_{AT} \qquad \cdots(3)$$

โดยที่

$$PM_{AT} = \frac{ASLOC\left(\frac{AT}{100}\right)}{ATPROD} \qquad \cdots(4)$$

และ

$$B = 1.01 + 0.01\sum_{j=1}^{5} SF_j \qquad \cdots(5)$$

สูตรคำนวณขนาดของซอฟต์แวร์ คือ

$$Size = \overline{Size}\left(1+\frac{BRAK}{100}\right) \qquad \cdots(6)$$

โดยที่

$$\overline{Size} = KNSLOC + KASLOC*\left(\frac{100-AT}{100}\right)*(AAM) \qquad \cdots(7)$$

$$AAM = \begin{cases} \dfrac{AA+AAF*(1+0.02*SU*UNFM)}{100}, AAF \le 0.05 \\ \dfrac{AA+AAF+SU*UNFM}{100}, AAF > 0.05 \end{cases} \qquad \cdots(8)$$

$$AAF = 0.4*DM + 0.3*CM + 0.3*IM \qquad \cdots(9)$$

สูตรคำนวณจำนวนระยะเวลาที่ใช้พัฒนาซอฟต์แวร์ คือ

$$TDEV = \left[3.0*(PM)^{(0.33+0.2*(B-1.01))}\right]\frac{SECD\%}{100} \qquad \cdots(10)$$

ตารางที่ 2.4 สัญลักษณ์และคำอธิบายของสมการจากเออรีลิไซด์โมเดล

สัญลักษณ์	คำอธิบาย
A	ค่าคงที่ กำหนดให้เป็น 2.5
AA	การประเมินสิ่งที่สามารถนำมาใช้ได้(Assessment and assimilation)
AT	เปอร์เซ็นต์ขององค์ประกอบ (Components)ที่สามารถแปลงได้โดยอัตโนมัติ (automatically translated)
ATPROD	ตัวคูณของการแปลงโดยอัตโนมัติ(Automatic translation productivity)
BRAK	เปอร์เซ็นต์ของการเปลี่ยนแปลงที่มีในความต้องการ(Requirements)
CM	เปอร์เซ็นต์ของการเปลี่ยนแปลงโปรแกรม(Percentage of code modified)
DM	เปอร์เซ็นต์ของการเปลี่ยนแปลงการออกแบบ(Percentage of design modified)
EM	ตัวคูณความพยายาม (Effort Multipliers : RCPX, RUSE, PDIF, PERS, PREX, FCIL, SCED)
IM	เปอร์เซ็นต์ของการเปลี่ยนแปลงการรวมและการทดสอบระบบ (Percentage of integration and test modified)
KASLOC	ขนาดของโปรแกรมที่ได้รับการปรับปรุง มีหน่วยเป็น หนึ่งพันบรรทัด
KNSLOC	ขนาดของโปรแกรมที่สร้างใหม่ มีหน่วยเป็น หนึ่งพันบรรทัด
PM	จำนวนคนที่พัฒนาต่อหนึ่งเดือน
SECD%	เปอร์เซ็นต์ของการลดและเพิ่มเวลาที่ใช้พัฒนาซอฟต์แวร์
SF	ค่าสเกลของปัจจัย(Scale Factors : PREC, FLEX, RESL, TEAM, PMAT)
SU	ระดับความเข้าใจของผู้พัฒนาที่มีต่อ โปรแกรม(Software Understanding)
TDEV	เวลาที่ใช้พัฒนาซอฟต์แวร์มีหน่วยเป็นเดือน (Time to develop)
UNFM	ความคุ้นเคยของโปรแกรมเมอร์ที่มีต่อซอฟต์แวร์(Programmer Unfamiliarity with Software)

2.1.1.3 แบบจำลองโพสอาร์คิเท็กเจอร์ (The Post-Architecture Model)

แบบจำลองโพสอาร์คิเท็กเจอร์ เป็นแบบจำลองที่ใช้ประมาณค่าใช้จ่ายในการพัฒนาซอฟต์แวร์ ซึ่งขยายขีดความสามารถของแบบที่สอง เนื่องผู้ใช้ต้องทราบข้อมูลเกี่ยวกับการพัฒนาซอฟต์แวร์ละเอียดขึ้น โดยผู้ใช้ต้องเป็นผู้ให้ข้อมูลเกี่ยวกับจำนวนบรรทัดของซอฟต์แวร์ที่จะพัฒนา และ ระดับของตัวขับค่าใช้จ่ายจากเดิม 7 ลักษณะเป็น 17 ลักษณะ เป็นข้อมูลพื้นฐานในการประมาณค่าใช้จ่ายซอฟต์แวร์ซึ่งตัวขับค่าใช้จ่ายทั้ง 17 ลักษณะแบ่งออกได้เป็น 4 กลุ่มใหญ่ๆดังนี้

1) ปัจจัยกลุ่มผลิตภัณฑ์ (Product Factors)

1.1) ความเชื่อถือได้ของผลิตภัณฑ์ซอฟต์แวร์ที่ต้องการ(Required Software Reliability: RELY) ซึ่งจะพิจารณาจากความสูญเสียที่เกิดขึ้นเมื่อระบบไม่สามารถทำงานได้

<u>คำจำกัดความ</u>

ความน่าเชื่อถือของซอฟต์แวร์หมายถึงซอฟต์แวร์ต้องสามารถทำงานได้ในช่วงเวลาที่กำหนดอย่างต่อเนื่องและถูกต้อง โดยไม่มีเหตุผิดปกติต่างๆ เช่น การคำนวณผิดพลาด การไม่ทำงานตามขั้นตอน การหยุดทำงานโดยที่ยังไม่จบการทำงานหรือไม่ถูกสั่งให้หยุดเป็นต้น โดยการพิจารณาความน่าเชื่อถือของผลิตภัณฑ์ซอฟต์แวร์ตามแนวทางของโคโคโม2 จะพิจารณาจากความเสียหายที่เกิดขึ้นหลังจากซอฟต์แวร์ไม่สามารถทำงานได้ เพื่อประเมินระดับของความพยายามที่จะใช้ในการพัฒนาซอฟต์แวร์ ดังแสดงในตารางที่ 2.5

ตารางที่ 2.5 ความแตกต่างของแต่ละระดับของ RELY

ระดับ	เงื่อนไข
ต่ำที่สุด	ถ้าซอฟต์แวร์ไม่สามารถทำงานได้ตามปกติจะทำให้เกิดความไม่สะดวกในการทำงานแต่สามารถดำเนินงานต่อไปได้
ต่ำ	ถ้าซอฟต์แวร์ไม่สามารถทำงานได้ตามปกติจะทำให้เสียเวลาหรือเสียค่าใช้จ่ายเพียงเล็กน้อยในการแก้ไขให้กลับมาทำงานได้เหมือนเดิม
ปานกลาง	ถ้าซอฟต์แวร์ไม่สามารถทำงานได้ตามปกติจะทำให้เสียเวลาหรือเสียค่าใช้จ่ายปานกลางในการแก้ไขให้กลับมาทำงานได้เหมือนเดิม
สูง	ถ้าซอฟต์แวร์ไม่สามารถทำงานได้ตามปกติจะทำให้เกิดความสูญเสียทางการเงินอย่างมาก
สูงมาก	ถ้าซอฟต์แวร์ไม่สามารถทำงานได้ตามปกติจะทำให้เกิดความเสี่ยงต่อชีวิตมนุษย์

1.2) ขนาดของฐานข้อมูลที่ใช้ในผลิตภัณฑ์ซอฟต์แวร์(Data Base Size: DATA) พิจารณาจากขนาดของฐานข้อมูลที่ใช้ในผลิตภัณฑ์ซอฟต์แวร์ ซึ่งคำนวณได้จากขนาดของฐานข้อมูลหารด้วยขนาดของโปรแกรม

<u>คำจำกัดความ</u>

ตัวขับค่าใช้จ่ายประเภทนี้จะใช้สมการในการคำนวณคือ D/P

โดย D คือ ขนาดของฐานข้อมูล(ไบต์)

P คือ ขนาดของซอฟต์แวร์(จำนวนบรรทัดของโปรแกรม)

ซึ่งจะพบว่าขนาดของฐานข้อมูลมีความสำคัญสำหรับการพิจารณา เนื่องจากถ้าขนาดของฐานข้อมูลมีขนาดใหญ่มากก็จะทำให้ได้ค่าจาก D/P มาก ซึ่งหมายถึงว่าข้อมูลที่ใช้ในการทดสอบซอฟต์แวร์ก็ต้องมีขนาดใหญ่มากขึ้นด้วย ดังแสดงในตารางที่2.6

ตารางที่ 2.6 ความแตกต่างของแต่ละระดับของ DATA

ระดับ	เงื่อนไข
ต่ำ	D/P น้อยกว่า 10
ปานกลาง	D/P มากกว่าเท่ากับ 10 แต่น้อยกว่า 100
สูง	D/P มากกว่าเท่ากับ 100 แต่น้อยกว่า 1000
สูงมาก	D/P มากกว่า 1000

1.3) ความซับซ้อนของผลิตภัณฑ์ซอฟต์แวร์(Complexity: CPLX) การพิจารณาความซับซ้อนของผลิตภัณฑ์ซอฟต์แวร์จะพิจารณา 5 อย่างด้วยกัน

1.3.1) การดำเนินการควบคุม(Control Operation)

การดำเนินการควบคุมหมายถึง กระบวนการที่ใช้ควบคุมการทำงานของโปรแกรม หรือลักษณะการทำงานของโปรแกรมในซอฟต์แวร์ที่จะพัฒนา ดังแสดงในตารางที่ 2.7

ตารางที่ 2.7 ความแตกต่างของแต่ละระดับของ Control Operation

ระดับ	เงื่อนไข
ต่ำมาก	การดำเนินการ(programming operation)ของซอฟต์แวร์ ส่วนใหญ่จะเป็นการดำเนินการแบบตรงไปข้างหน้า (straight-line code) แต่ก็มีลักษณะการดำเนินการแบบเป็นเงื่อนไขแต่เป็นเงื่อนไขที่ไม่ซับซ้อน(non-nested structured) เช่น DO, CASE, IF-THEN-ELSE นอกจากนี้อาจมีการใช้โมดูลง่ายๆ เช่น การทำโปรซีเดอร์คอร์(procedure call) เป็นต้น
ต่ำ	การดำเนินการของซอฟต์แวร์จะมีโครงสร้างที่ซับซ้อน(nesting of structured programming) และโดยส่วนใหญ่จะเป็นแบบมีเงื่อนไข เช่น มีลูป WHILE ซ้อน IF-THEN-ELSE หรือ IF-THEN-ELSE ซ้อน IF-THEN-ELSE เป็นต้น
ปานกลาง	การดำเนินการของซอฟต์แวร์โดยส่วนใหญ่จะเป็นแบบมีเงื่อนไขที่ซับซ้อน จึงมีการใช้ตารางตัดสินใจ(decision table) หรือ มีการเรียกใช้โปรแกรมหรือส่วนของโปรแกรมแบบ call back หรือแบบการส่งข้อความ(message passing) รวมทั้งมีการดำเนินการที่สนับสนุนการประมวลผลแบบกระจายบนเครือข่าย(distribute processing) เช่น ระบบclient/server
สูง	โครงสร้างของการดำเนินการมีความซับซ้อนมากซึ่งประกอบด้วยการดำเนินการที่เป็นแบบเงื่อนไขซ้อนเงื่อนไข หรือมีการใช้คิว(queue) และกองซ้อน(stack) เป็นตัวควบคุมลำดับการดำเนินการ หรือมีการประมวลผลแบบกระจายซึ่งมีอุปกรณ์ ฮาร์ดแวร์ และซอฟต์แวร์ที่มีความเหมือนกันหรือใกล้เคียงกัน (homogeneous) คือ มีซีพียูหรือเวอร์ชันเดียวกัน หรือมีตัวประมวลผลเพียงตัวเดียวและเป็นแบบทันทีทันใด(soft real time)
สูงมาก	การดำเนินการจะมีลักษณะเป็นแบบการเวียนบังเกิด(recursive) หรือมีการประมวลผลแบบกระจายซึ่งมีอุปกรณ์ ฮาร์ดแวร์ และซอฟต์แวร์ที่มีความความต่างกัน(heterogeneous) คือ มีซีพียูหรือเวอร์ชันต่างกัน หรือมีตัวประมวลผลเพียงตัวเดียวและเป็นแบบทันทีทันใด (hard real time)

ตารางที่ 2.7 ความแตกต่างของแต่ละระดับของ Control Operation(ต่อ)

ระดับ	เงื่อนไข
สูงที่สุด	การดำเนินการของซอฟต์แวร์จะต้องควบคุมการทำงานของฮาร์ดแวร์หรืออุปกรณ์โดยตรง (Microcode level control) หรือมีตารางการใช้ทรัพยากร(resource)ที่สามารถแก้ไขได้โดยอัตโนมัติตามความเหมาะสม หรือมีการประมวลผลแบบกระจาย(distribute processing) ที่ได้ผลลัพธ์แบบทันทีทันใด(hard real time)

1.3.2) การดำเนินการคำนวณ(Computation Operation)

การดำเนินการคำนวณ หมายถึง วิธีการหรือรูปแบบการคำนวณที่มีใช้ในซอฟต์แวร์ที่จะพัฒนา ดังแสดงในตารางที่ 2.8

ตารางที่ 2.8 ความแตกต่างของแต่ละระดับของ Computation Operation

ระดับ	เงื่อนไข
ต่ำมาก	การดำเนินการคำนวณสามารถหาค่าจากสมการพีชคณิตหรือสมการเส้นตรง เช่น ทำการบวก การลบ การคูณ การหาร ตัวอย่างเช่น A=B+C*(D-E)
ต่ำ	การดำเนินการคำนวณสามารถหาค่าจากสมการที่มีความยากระดับกลาง(moderate-level expressions) คือ นอกจากทำการบวก การลบ การคูณ การหาร แล้วยังสามารถทำการยกกำลัง การหารากการคำนวณหาค่าตรีโกณมิติ ตัวอย่างเช่น D= SQRT(B^2 – 4*A*C)
ปานกลาง	การดำเนินการคำนวณมีการใช้รูทีน(routine) ที่เป็นมาตราฐานทางคณิตศาสตร์ และสถิติ เช่น การหาส่วนเบี่ยงเบนมาตราฐาน การหาค่า Z เป็นต้น นอกจากนี้ยังคำนวณหาค่าของเมตริกซ์(matrix)และเวกเตอร์(vector)ได้
สูง	การดำเนินการคำนวณสามารถวิเคราะห์เชิงตัวเลข(numerical analysis)ขั้นพื้นฐานได้ เช่น การหาคำตอบของสมการ โดยวิธีการของนิวตัน(Newton's Method) หรือ โดยวิธีการของ Guassian การแก้สมการเชิงอนุพันธ์ เป็นต้น
สูงมาก	การดำเนินการคำนวณสามารถวิเคราะห์โครงสร้างเชิงตัวเลขชนิดที่มีโครงสร้างแน่นอน เช่น การหาคำตอบจากสมการเมตริกที่ไม่เอกฐาน (non-singular matrix equation) การหาค่าจากสมการเชิงอนุพันธ์ย่อย(partial differential equation) และการหารูปแบบของความสัมพันธ์จากข้อมูล 2 ชุด
สูงที่สุด	การดำเนินการคำนวณสามารถวิเคราะห์โครงสร้างเชิงตัวเลขชนิดที่มีโครงสร้างไม่แน่นอน เช่น การวิเคราะห์ระดับความสูงของเสียงรบกวน(highly accurate analysis of noisy) การวิเคราะห์ข้อมูลแบบ stochastic

1.3.3) การดำเนินการที่เกี่ยวกับอุปกรณ์(Device-dependent Operation)

การดำเนินการที่เกี่ยวกับอุปกรณ์ หมายถึง วิธีหรือกระบวนการที่ซอฟต์แวร์ที่จะพัฒนาใช้งานอุปกรณ์ เช่น การสั่งให้เครื่องพิมพ์เอกสารพิมพ์รายงาน การจัดเก็บข้อมูลลงในฮาร์ดดิส เป็นต้น ดังแสดงในตารางที่ 2.9

ตารางที่ 2.9 ความแตกต่างของแต่ละระดับของ Device-dependent Operation

ระดับ	เงื่อนไข
ต่ำมาก	คำสั่งที่ใช้ดำเนินการเกี่ยวกับอุปกรณ์เป็นคำสั่งที่ใช้งานง่าย หรือเป็นภาษาระดับสูง เช่น การใช้คำสั่ง อ่าน(read) เขียน(write) ในภาษาพาสคาล(Pascal) ซึ่งเป็นคำสั่งที่ใช้ สำหรับการอ่านและบันทึกข้อมูล
ต่ำ	คำสั่งที่ใช้ดำเนินการเกี่ยวกับอุปกรณ์ สามารถใช้คำสั่งที่เป็นพื้นฐานที่ภาษามีการเตรียมไว้ให้ เช่น คำสั่งเขียนข้อมูลลงแฟ้มข้อมูล(put) หรือ อ่านข้อมูลในแฟ้มข้อมูล(get) ในภาษาC โดยที่ผู้พัฒนาไม่จำเป็นต้องมีความรู้เกี่ยวกับอุปกรณ์ที่ใช้
ปานกลาง	ผู้พัฒนาจะต้องมีความรู้อย่างเจาะจงเกี่ยวกับอุปกรณ์ที่เป็นตัวรับข้อมูลและตัวแสดงผล (input/output) โดยการใช้งานอุปกรณ์ต้องทำ 3 ขั้นตอนดังนี้คือ เลือกอุปกรณ์ที่จะใช้งาน ตรวจสอบสถานะการทำงานของอุปกรณ์นั้นและตรวจสอบความผิดพลาดในการทำงานของอุปกรณ์
สูง	การดำเนินการที่เกี่ยวกับอุปกรณ์ของตัวรับข้อมูลและตัวแสดงผลจะกระทำในระดับกายภาพ (physical) เช่น การแปลงตำแหน่งของหน่วยความจำทางกายภาพ เพื่อช่วยในการค้นหาและการอ่านและการกำหนดรูปแบบการซ้อนทับ(overlap)ในหน่วยความจำของอุปกรณ์
สูงมาก	การดำเนินการที่เกี่ยวกับอุปกรณ์ของตัวรับข้อมูลและตัวแสดงผลจะมีรูทีน(routines) สำหรับจัดการเมื่อเกิดการผิดพลาดในขณะส่งและรับข้อมูล มีการจัดการเกี่ยวกับการส่งข้อมูลในสายการสื่อสาร หรือมีระบบการวัดประสิทธิภาพของอุปกรณ์แบบเข้มงวด(performance-intensive embedded system) เช่น มีการบันทึกการทำงานของอุปกรณ์ที่ทำงานผิดพลาดหรือไม่ทำงาน
สูงที่สุด	มีการเขียนโปรแกรมไปติดต่อกับการควบคุมการทำงานของฮาร์ดแวร์โดยตรง(micro-programmed operations) หรือมีระบบการวัดประสิทธิภาพของอุปกรณ์แบบวิกฤต (performance-critical embedded systems) เช่น มีการบันทึกการทำงานของอุปกรณ์ทุกชนิดตลอดเวลา

1.3.4) การดำเนินการเกี่ยวกับการจัดการกับข้อมูล(Data Management Operation)

การดำเนินการเกี่ยวกับการจัดการกับข้อมูล หมายถึง ซอฟต์แวร์ที่จะพัฒนามีวิธีการจัดการกับการเก็บข้อมูลและการใช้งานข้อมูล ดังแสดงในตารางที่ 2.10

ตารางที่ 2.10 ความแตกต่างของแต่ละระดับของ Data Management Operation

ระดับ	เงื่อนไข
ต่ำมาก	การดำเนินการเกี่ยวกับการจัดการกับข้อมูลมีการใช้แถวลำดับ(array)ในหน่วยความจำหลัก เพื่อเก็บข้อมูล และสามารถใช้สอบถาม(query)และแก้ไข(update)ได้
ต่ำ	การดำเนินการเกี่ยวกับการจัดการกับข้อมูลสามารถใช้งานแฟ้มข้อมูลเพียงแฟ้มเดียว และไม่สามารถเปลี่ยนแปลงโครงสร้างของแฟ้มข้อมูลได้ แฟ้มข้อมูลสามารถใช้สอบถามและแก้ไขได้
ปานกลาง	การดำเนินการเกี่ยวกับการจัดการกับข้อมูลสามารถใช้งานแฟ้มข้อมูลได้หลายแฟ้มข้อมูลโดยจะเป็นแฟ้มส่งข้อมูลออกเพียงแฟ้มเดียว และไม่สามารถเปลี่ยนแปลงโครงสร้างของแฟ้มข้อมูลได้ จะไม่มีการดำเนินการกับแฟ้มข้อมูลที่เกิดในระหว่างการทำงาน เช่น log file
สูง	การดำเนินการเกี่ยวกับการจัดการกับข้อมูลสามารถทำทริกเกอร์(trigger)โดยใช้ข้อมูลเพียงชุดเดียว สามารถทำการเปลี่ยนโครงสร้างของข้อมูลที่ซับซ้อน(complex data restructuring) เช่นการเปลี่ยนคีย์ การเปลี่ยนความสัมพันธ์ระหว่างแฟ้มข้อมูล เป็นต้น
สูงมาก	การดำเนินการเกี่ยวกับการจัดการฐานข้อมูลสามารถทำกับข้อมูลที่เก็บไว้ในหลายสถานที่ได้ สามารถทำทริกเกอร์(trigger)กับข้อมูลจำนวนมากๆได้ สามารถค้นหาข้อมูลในฐานข้อมูลได้
สูงที่สุด	การดำเนินการเกี่ยวกับการจัดการกับข้อมูลเป็นแบบพลวัต(dynamic relational) คือสามารถเปลี่ยนแปลงโครงสร้างบางอย่างโดยอัตโนมัติตามสภาพของข้อมูลที่เก็บได้ มีโครงสร้างเชิงวัตถุ(object structures) มีการจัดการข้อมูลด้วยภาษาธรรมชาติ(natural language data management)

1.3.5) การดำเนินการส่วนต่อประสานกับผู้ใช้(User Interface Management Operation) การปฏิบัติการส่วนต่อประสานกับผู้ใช้ หมายถึง วิธีหรือรูปแบบของซอฟต์แวร์ที่จะพัฒนาติดต่อกับผู้ใช้ ดังแสดงในตารางที่ 2.11

ตารางที่ 2.11 ความแตกต่างของแต่ละระดับของ User Interface Management Operation

ระดับ	เงื่อนไข
ต่ำมาก	มีโปรแกรมช่วยในการสร้าง(generators)รูปแบบการรับข้อมูลเข้า(input form) และการทำรายงาน (report) แบบง่าย เช่น มีwizardช่วย
ต่ำ	มีการใช้ตัวสร้างส่วนต่อประสานกับผู้ใช้(user interface)ที่ไม่ยุ่งยาก เช่น ใช้ตัวสร้างGUI (graphic user interface builders)
ปานกลาง	มีการใช้งานส่วนต่อประสานกับผู้ใช้เป็นแบบวินโดว์ที่ไม่มีความซับซ้อนมาก เช่น มีการติดต่อแบบ windows95 เป็นต้น
สูง	มีการใช้งานส่วนต่อประสานกับผู้ใช้แบบวินโดว์ซึ่งสามารถใช้เสียงเป็นตัวรับและแสดงผล และยังใช้ติดต่อแบบมัลติมีเดียอย่างง่ายได้ด้วย
สูงมาก	มีการใช้ภาพ2 มิติ หรือ 3มิติที่มีความซับซ้อนไม่มาก มีการใช้ภาพที่เป็นพลวัต(dynamic graphic) และมีการใช้มัลติมีเดีย(multimedia)ในการติดต่อกับผู้ใช้
สูงที่สุด	มีการใช้มัลติมีเดียที่ซับซ้อน เช่น วีดิโอคอนเฟอร์เร็น (video conference) และมีการใช้ภาพเสมือนจริง(virtual reality) ในการติดต่อกับผู้ใช้

1.4) ความต้องการเอกสารที่ตรงกับวงจรชีวิต(Documentation match to life-cycle needs: DOCU) พิจารณาถึงความละเอียดของเอกสารในทุกวงจรชีวิต

<u>คำจำกัดความ</u>

การกำหนดระดับของตัวขับค่าใช้จ่ายประเภทนี้จะอยู่ในรูปของการจัดทำเอกสารประกอบการพัฒนาซอฟต์แวร์ได้ครบทุกขั้นตอนทั้งโครงการ ซึ่งขั้นตอนหรือรูปแบบของการจัดทำเอกสารก็ขึ้นอยู่กับผู้พัฒนาว่าใช้วงจรชีวิต(life-cycle)แบบใดในการพัฒนาซอฟต์แวร์ เช่น วอลเตอร์ฟอร์ไมเดล(Waterfall Model) อินครีเมนต์ดิวิลอปต์ไมเดล(Increment Development Model) สไปรอลโมเดล(Spiral Model) โปรโตไทป์ปิงโมเดล(Prototyping Model) เป็นต้น ซึ่งการกำหนดระดับขึ้นกับเอกสารที่จัดทำครบทุกขั้นตอนและเพียงพอกับความต้องการใช้งานเพียงใด ดังแสดงในตารางที่ 2.12

ตารางที่ 2.12 ความแตกต่างของแต่ละระดับของ DOCU

ระดับ	เงื่อนไข
ต่ำ	ไม่มีการออกแบบสำหรับการนำโปรแกรมกลับมาใช้ใหม่ในอนาคต
ปานกลาง	มีการออกแบบเพื่อให้สามารถนำบางส่วนของโปรแกรมที่ไม่ได้อยู่ในโครงการ (Project)เดียวกับซอฟต์แวร์ที่จะพัฒนากลับมาใช้ใหม่ได้ในอนาคต เช่น นำส่วนของโปรแกรมของ Microsoft Word6 มาใช้กับ Microsoft Word7 เป็นต้น
สูง	มีการออกแบบเพื่อให้สามารถนำบางส่วนของโปรแกรมที่ไม่ได้อยู่ในโปรแกรม (Program)เดียวกับซอฟต์แวร์ที่จะพัฒนากลับมาใช้ใหม่ได้ในอนาคต เช่น นำบางส่วนของโปรแกรมของMicrosoft Excel มาใช้กับ Microsoft Word เป็นต้น
สูงมาก	มีการออกแบบเพื่อให้สามารถนำโปรแกรมหรือบางส่วนของโปรแกรมที่ไม่ได้อยู่ในระบบ(Product Line)เดียวกับซอฟต์แวร์ที่จะพัฒนากลับมาใช้ใหม่ได้ในอนาคต เช่น นำส่วนของโปรแกรมของระบบซื้อ-ขาย มาใช้กับระบบธนาคาร เป็นต้น
สูงที่สุด	มีการออกแบบเพื่อให้สามารถนำโปรแกรมหรือบางส่วนของโปรแกรมไปใช้ได้ในหลายๆระบบ เช่น นำส่วนของโปรแกรมของระบบบัญชี ระบบซื้อ-ขาย และระบบงานบุคคล มาใช้กับระบบธนาคาร เป็นต้น

1.5) ความต้องการที่จะนำผลิตภัณฑ์ซอฟต์แวร์กลับมาใช้ใหม่ (Required Reusability: RUSE) พิจารณาจากการออกแบบผลิตภัณฑ์ซอฟต์แวร์ที่มีการเตรียมสำหรับการนำกลับมาใช้ใหม่

<u>คำจำกัดความ</u>

ตัวขับค่าใช้จ่ายนี้จะพิจารณาความพยายามสร้างซอฟต์แวร์ให้มีลักษณะที่เป็นองค์ประกอบ (component) เพื่อสามารถนำกลับไปใช้(Reuse)กับซอฟต์แวร์อื่นๆในอนาคต ซึ่งการที่จะนำบาง

ส่วนของโปรแกรมกลับมาใช้ในลักษณะใดๆของซอฟต์แวร์ที่จะพัฒนานั้นก็ขึ้นอยู่กับ การออกแบบซอฟต์แวร์ให้มีลักษณะเป็นฟังก์ชันหรือเป็นโมดุลที่เล็กที่สุดที่มีฟังก์ชันเดียว ดังแสดงในตารางที่ 2.13

ตารางที่ 2.13 ความแตกต่างของแต่ละระดับของ RUSE

ระดับ	เงื่อนไข
ต่ำมาก	ไม่มีการจัดทำเอกสารการพัฒนาซอฟต์แวร์
ต่ำ	เอกสารจัดทำไม่ครอบคลุมในหลายๆขั้นของวงจรชีวิต
ปานกลาง	เอกสารจัดทำได้ครบตามข้อกำหนดขั้นต่ำของวงจรชีวิต
สูง	เอกสารจัดทำได้ครบตามข้อกำหนดของวงจรชีวิต และมีเอกสารเพิ่มเติมที่ใช้ภายในองค์กรแต่ไม่ได้เป็นข้อกำหนดไว้ในวงจรชีวิต เพื่อทำให้เกิดความสมบูรณ์มากขึ้น
สูงมาก	เอกสารจัดทำได้ครบตามข้อกำหนดของวงจรชีวิต และมีเอกสารเพิ่มเติมที่มีความละเอียดเพื่อให้เกิดความสมบูรณ์มากที่สุด

2) ปัจจัยกลุ่มแพลทฟอร์ม (Platform Factors)

2.1) การจำกัดเวลาที่ใช้ในการประมวลผล(Execution Time Constraint: TIME) พิจารณาจากเวลาที่ใช้ในการกระทำการของซอฟต์แวร์ระบบ

<u>คำจำกัดความ</u>

เนื่องจากเวลาที่เข้ากระทำการ(Execution Time)ขึ้นอยู่กับซอฟต์แวร์ระบบ(Software System) การกำหนดระดับของตัวขับค่าใช้จ่ายประเภทนี้จึงคิดเป็นเปอร์เซ็นต์ ของเวลาที่คาดว่าซอฟต์แวร์จะเข้ากระทำการ จากเวลาทั้งหมดที่ระบบมีให้กับซอฟต์แวร์ ดังแสดงในตารางที่ 2.14

ตารางที่ 2.14 ความแตกต่างของแต่ละระดับของ TIME

ระดับ	เงื่อนไข
ปานกลาง	เวลาที่ซอฟต์แวร์เข้ากระทำการน้อยกว่าหรือเท่ากับ 50% ของเวลาที่ซีพียูมีให้
สูง	เวลาที่ซอฟต์แวร์เข้ากระทำการมากกว่า 50% ของเวลาที่ซีพียูมีให้ แต่น้อยกว่าหรือเท่ากับ 70% ของเวลาที่ซีพียูมีให้
สูงมาก	เวลาที่ซอฟต์แวร์เข้ากระทำการมากกว่า 70% ของเวลาที่ซีพียูมีให้แต่น้อยกว่าหรือเท่ากับ 85% ของเวลาที่ซีพียูมีให้
สูงที่สุด	เวลาที่ซอฟต์แวร์เข้ากระทำการมากกว่า 85% ของเวลาที่ซีพียูมีให้ น้อยกว่าหรือเท่ากับ 95% ของเวลาที่ซีพียูมีให้

2.2) การจำกัดหน่วยความจำหลัก(Main Storage Constraint: STOR) พิจารณาขนาดของหน่วยความจำหลักซึ่งจะขึ้นกับความสามารถของซอฟต์แวร์ระบบ

<u>คำจำกัดความ</u>

การกำหนดระดับให้กับตัวขับค่าใช้จ่ายประเภทนี้จะพิจารณาจากการที่ซอฟต์แวร์ใช้พื้นที่ในหน่วยความจำหลักมากน้อยเพียงใด ซึ่งถ้าซอฟต์แวร์ใช้หน่วยความจำมากซอฟต์แวร์ก็จะต้องมีการจัดการหน่วยความจำที่ดี การกำหนดระดับจะพิจารณาโดยคิดเป็นเปอร์เซ็นต์ของขนาดหน่วยความจำหลักที่คาดว่าจะใช้จากขนาดของหน่วยความจำหลักที่ระบบมีให้ ดังแสดงในตารางที่ 2.15

ตารางที่ 2.15 ความแตกต่างของแต่ละระดับของ STOR

ระดับ	เงื่อนไข
ปานกลาง	ซอฟต์แวร์จะใช้พื้นที่ในหน่วยความจำหลักน้อยกว่าหรือเท่ากับ 50% ของหน่วยความจำหลักที่สามารถใช้ได้
สูง	ซอฟต์แวร์จะใช้พื้นที่ในหน่วยความจำหลักมากกว่า 50% ของหน่วยความจำหลักที่สามารถใช้ได้ แต่น้อยกว่าหรือเท่ากับ 70% ของหน่วยความจำหลักที่สามารถใช้ได้
สูงมาก	ซอฟต์แวร์จะใช้พื้นที่ในหน่วยความจำหลักมากกว่า 70% ของหน่วยความจำหลักที่สามารถใช้ได้ แต่น้อยกว่าหรือเท่ากับ 85% ของหน่วยความจำหลักที่สามารถใช้ได้
สูงที่สุด	ซอฟต์แวร์จะใช้พื้นที่ในหน่วยความจำหลักมากกว่า 85% ของหน่วยความจำหลักที่สามารถใช้ได้ แต่น้อยกว่าหรือเท่ากับ 95% ของหน่วยความจำหลักที่สามารถใช้ได้

2.3) การเปลี่ยนแพลทฟอร์ม(Platform Volatility: PVOL) แพลทฟอร์มในที่นี้รวมทั้งฮาร์ดแวร์และซอฟต์แวร์ ซึ่งจะพิจารณาจากระยะเวลาที่สามารถกระทำการได้ก่อนมีการเปลี่ยนแปลง

<u>คำจำกัดความ</u>

การเปลี่ยนแปลงได้ง่ายของแพลทฟอร์มหมายถึง ซอฟต์แวร์ที่จะพัฒนาต้องถูกออกแบบหรือพัฒนาให้มีความคล่องตัวสูงในการเปลี่ยนหรือปรับปรุงให้เข้ากับแพลทฟอร์มใหม่ ซึ่งถ้าซอฟต์แวร์ที่จะพัฒนาต้องทำการเปลี่ยนบ่อยๆ ก็ทำให้เกิดความความยากในการพัฒนา หรือ ใช้เวลามากในการพัฒนาเพื่อให้ซอฟต์แวร์สามารถทำงานบนหลายๆแพลทฟอร์มได้ เป็นต้น แพลทฟอร์มในที่นี้จะมีความหมายรวมทั้งฮาร์ดแวร์และซอฟต์แวร์(OS,DBMS etc) ตัวอย่างเช่น ถ้าซอฟต์แวร์ที่จะพัฒนาเป็นระบบปฏิบัติการ แพลทฟอร์มคืออุปกรณ์คอมพิวเตอร์ ถ้าซอฟต์แวร์ที่จะพัฒนาเป็นระบบจัดการฐานข้อมูล(database management system) แพลทฟอร์มคือฮาร์ดแวร์และระบบปฏิบัติการ ถ้าซอฟต์แวร์ที่จะพัฒนาคือระบบบัญชี แพลทฟอร์มคือระบบปฏิบัติการ ระบบจัดการฐานข้อมูล และฮาร์ดแวร์ ถ้าซอฟต์แวร์ที่จะพัฒนาคือตัวค้นหาข้อความบนเครือข่าย(network text browser) แพลทฟอร์มคือ ฮาร์ดแวร์ ระบบปฏิบัติการ และคลัง

เก็บข้อมูลแบบกระจาย(distribute information repositories) นอกจากนี้แพลทฟอร์มยังรวมไปถึงตัวแปลงภาษา(compiler)ด้วย ดังในตารางที่ 2.16

ตารางที่ 2.16 ความแตกต่างของแต่ละระดับของ PVOL

ระดับ	เงื่อนไข
ต่ำ	จะทำการเปลี่ยนแพลทฟอร์มทุกๆ 12 เดือน หรือทำการปรับปรุงบางส่วนทุกๆ 1 เดือน
ปานกลาง	จะทำการเปลี่ยนแพลทฟอร์มทุกๆ 6 เดือน หรือทำการปรับปรุงบางส่วนทุกๆ 2 สัปดาห์
สูง	จะทำการเปลี่ยนแพลทฟอร์มทุกๆ 2 เดือน หรือทำการปรับปรุงบางส่วนทุกๆ 1 สัปดาห์
สูงมาก	จะทำการเปลี่ยนแพลทฟอร์มทุกๆ 2 สัปดาห์ หรือทำการปรับปรุงบางส่วนทุกๆ 2 วัน

3) ปัจจัยกลุ่มบุคลากร (Personal Factors)

3.1) ความสามารถในการวิเคราะห์(Analyst Capability: ACAP) พิจารณาจากการวิเคราะห์และออกแบบระบบของทีมที่พัฒนา

<u>คำจำกัดความ</u>

ตัวขับค่าใช้จ่ายประเภทนี้จะพิจารณาจากความสามารถของบุคลากรในการวิเคราะห์ระบบงานตามที่ร้องขอ(request) และการออกแบบซอฟต์แวร์อย่างรายละเอียด(Detailed Design) เช่น การออกแบบเกี่ยวกับข้อมูล(Data Design) การออกแบบเกี่ยวกับโครงสร้าง(Architectural Design) การออกแบบเกี่ยวกับกระบวนการ(Procedural Design) การออกแบบเกี่ยวกับการเชื่อมต่อ(Interface Design) เป็นต้น ปัจจัยหลักที่ใช้ในการแบ่งระดับความสามารถคือ ความสามารถ(ability)ในการวิเคราะห์และออกแบบ ประสิทธิภาพ(efficiency)และความละเอียด(thoroughness)ในการวิเคราะห์และออกแบบ และความสามารถในการสื่อสารกับผู้อื่น ในการกำหนดระดับจะไม่นำประสบการณ์ในการวิเคราะห์มาพิจารณาด้วยเนื่องจากได้แยกออกเป็นอีกหนึ่งตัวขับค่าใช้จ่าย การกำหนดระดับของความสามารถในการวิเคราะห์จะพิจารณาความสามารถโดยเฉลี่ยของนักวิเคราะห์ระบบในทีมว่ามีความสามารถอยู่ในตำแหน่งเปอร์เซ็นต์ไทล์ที่เท่าใด ดังแสดงในตารางที่ 2.17

ตารางที่ 2.17 ความแตกต่างของแต่ละระดับของ ACAP

ระดับ	เงื่อนไข
ต่ำมาก	เปอร์เซ็นต์ไทล์ที่ 0 ถึง 25
ต่ำ	เปอร์เซ็นต์ไทล์ที่ 26 ถึง 45
ปานกลาง	เปอร์เซ็นต์ไทล์ที่ 46 ถึง 65
สูง	เปอร์เซ็นต์ไทล์ที่ 66 ถึง 85
สูงมาก	เปอร์เซ็นต์ไทล์ที่ 86 ถึง 100

3.2) ความสามารถของโปรแกรมเมอร์(Programmer Capability:PCAP) พิจารณาจากความสามารถของทีมที่พัฒนาในการสร้างโปรแกรม

ข้อจำกัดความ

แม้ว่ามีการให้ความสำคัญกับการวิเคราะห์ระบบอย่างมาก แต่เครื่องมือในปัจจุบันก็ได้พัฒนามีจนมีความซับซ้อนมากขึ้น ดังนั้นการใช้งานเครื่องมือเหล่านี้ก็ขึ้นกับความสามารถของโปรแกรมเมอร์ การจะกำหนดระดับให้นั้นควรจะพิจารณาบนพื้นฐานของความสามารถโดยรวมของโปรแกรมเมอร์ในทีมที่พัฒนามากกว่าที่จะพิจารณาโปรแกรมเมอร์เพียงคนเดียว ปัจจัยร่วมที่มีส่วนช่วยในการกำหนดระดับคือ ความสามารถในการพัฒนาซอฟต์แวร์ ประสิทธิภาพของซอฟต์แวร์ ความสมบูรณ์ของซอฟต์แวร์ที่พัฒนา และความสามารถในการติดต่อสื่อสารกับผู้อื่น ตัวขับค่าใช้จ่ายประเภทนี้จะไม่นำประสบการณ์ในการเขียนโปรแกรมมาพิจารณาด้วย เนื่องจากได้แยกออกเป็นอีกหนึ่งตัวขับค่าใช้จ่าย การกำหนดระดับจะพิจารณาจากความสามารถเฉลี่ยในการเขียนโปรแกรมของทีมโปรแกรมเมอร์ว่าอยู่ในตำแหน่งของเปอร์เซ็นต์ไทล์ที่เท่าใด ดังแสดงในตารางที่ 2.18

ตารางที่ 2.18 ความแตกต่างของแต่ละระดับของ PCAP

ระดับ	เงื่อนไข
ต่ำมาก	เปอร์เซ็นต์ไทล์ที่ 0 ถึง 25
ต่ำ	เปอร์เซ็นต์ไทล์ที่ 26 ถึง 45
ปานกลาง	เปอร์เซ็นต์ไทล์ที่ 46 ถึง 65
สูง	เปอร์เซ็นต์ไทล์ที่ 66 ถึง 85
สูงมาก	เปอร์เซ็นต์ไทล์ที่ 86 ถึง 100

3.3) ประสบการณ์การใช้แอปพลิเคชั่น(Applications Experience: AEXP) พิจารณาจากประสบการณ์ที่เคยใช้แอปพลิเคชั่นประเภทเดียวกับที่จะพัฒนาระบบใหม่ โดยจะพิจารณาจากประสบการณ์โดยรวมของทีมที่พัฒนา

คำจำกัดความ

การกำหนดระดับของตัวขับค่าใช้จ่ายประเภทนี้ขึ้นอยู่กับประสบการณ์ในการพัฒนาซอฟต์แวร์(แอปพลิเคชั่น)ชนิดเดียวกับซอฟต์แวร์ที่จะพัฒนา ซึ่งในตัวขับค่าใช้จ่ายนี้จะพิจารณาจากระยะเวลาเฉลี่ยที่ทีมพัฒนาเคยใช้พัฒนาซอฟต์แวร์ประเภทเดียวกับซอฟต์แวร์ที่จะพัฒนา ดังแสดงในตารางที่ 2.19

ตารางที่ 2.19 ความแตกต่างของแต่ละระดับของ AEXP

ระดับ	เงื่อนไข
ต่ำมาก	น้อยกว่าหรือเท่ากับ 2 เดือน
ต่ำ	มากกว่า 2 เดือนแต่น้อยกว่าหรือเท่ากับ 6 เดือน
ปานกลาง	มากกว่า 6 เดือนแต่น้อยกว่าหรือเท่ากับ 1 ปี
สูง	มากกว่า 1 ปีแต่น้อยกว่าหรือเท่ากับ3 ปี
สูงมาก	มากกว่า 3 ปี

3.4) ประสบการณ์แพลทฟอร์ม(Platform Experience:PEXP) พิจารณาจากการใช้แพลทฟอร์มได้เป็นอย่างดีของทีมที่พัฒนา เช่น มีการใช้รูปภาพ การใช้ส่วนต่อประสานกับผู้ใช้ การใช้ฐานข้อมูล หรือ การใช้เครือข่าย

คำจำกัดความ

ตัวขับค่าใช้จ่ายนี้จะพิจารณาจากการทีมที่พัฒนามีประสบการณ์ในการใช้แพลทฟอร์มได้เป็นอย่างดี ซึ่งแพลทฟอร์มในที่นี้รวมทั้งฮาร์ดแวร์และซอฟต์แวร์ เช่น มีการรวมรูปภาพกับตัวเชื่อมประสาน(user interface) มีการใช้ฐานข้อมูล มีการใช้ระบบเครือข่าย(networking) เป็นต้น ซึ่งในตัวขับค่าใช้จ่ายนี้จะพิจารณาจากระยะเวลาที่ทีมพัฒนาเคยใช้แพลทฟอร์มประเภทเดียวกับแพลฟอร์มที่จะใช้ในซอฟต์แวร์ที่จะพัฒนา ดังแสดงในตารางที่ 2.20

ตารางที่ 2.20 ความแตกต่างของแต่ละระดับของ PEXP

ระดับ	เงื่อนไข
ต่ำมาก	น้อยกว่าหรือเท่ากับ 2 เดือน
ต่ำ	มากกว่า 2 เดือนแต่น้อยกว่าหรือเท่ากับ 6 เดือน
ปานกลาง	มากกว่า 6 เดือนแต่น้อยกว่าหรือเท่ากับ 1 ปี
สูง	มากกว่า 1 ปีแต่น้อยกว่าหรือเท่ากับ3 ปี
สูงมาก	มากกว่า 3 ปี

3.5) ประสบการณ์โปรแกรมภาษาและเครื่องมือ(Language and Tool Experience: LTEX) พิจารณาจากประสบการณ์ของทีมที่พัฒนาว่าสามารถใช้เครื่องมือทำได้ตรงตามความต้องการ และ เวลาที่ใช้ในการพัฒนาระบบ

<u>คำจำกัดความ</u>

ตัวขับค่าใช้จ่ายนี้เป็นการวัดประสบการณ์ในการใช้ภาษาโปรแกรม(programming language) และใช้เครื่องมือในการพัฒนาซอฟต์แวร์ การพัฒนาซอฟต์แวร์ยังรวมถึงการใช้งานเครื่องมือพัฒนาซอฟต์แวร์ในการวิเคราะห์และออกแบบ การจัดการโครงแบบ(configuration management) การจัดทำเอกสารการพัฒนาซอฟต์แวร์ การจัดการไลบรารี่(library management) ของโปรแกรมภาษา การกำหนดรูปแบบการเขียนโปรแกรม(program style)ที่เป็นมาตราฐานเดียวกัน และการตรวจสอบความสอดคล้อง(consistency checking)ของโปรแกรม เป็นต้น นอกจากนี้การมีประสบการณ์ในการเขียนโปรแกรมในภาษาใดภาษาหนึ่งจะมีผลต่อระยะเวลาที่ใช้ในการพัฒนาด้วย ดังแสดงในตารางที่ 2.21

ตารางที่ 2.21 ความแตกต่างของแต่ละระดับของLTEX

ระดับ	เงื่อนไข
ต่ำมาก	น้อยกว่าหรือเท่ากับ 2 เดือน
ต่ำ	มากกว่า 2 เดือนแต่น้อยกว่าหรือเท่ากับ 6 เดือน
ปานกลาง	มากกว่า 6 เดือนแต่น้อยกว่าหรือเท่ากับ 1 ปี
สูง	มากกว่า 1 ปีแต่น้อยกว่าหรือเท่ากับ3 ปี
สูงมาก	มากกว่า 3 ปี

3.6) ความต่อเนื่องของบุคลากร(Personal Continuity: PCON) พิจารณาจากการหมุนเวียนของบุคลากรในทีมที่พัฒนา

<u>คำจำกัดความ</u>

ความต่อเนื่องของบุคลากรหมายถึงการหมุนเวียนของบุคลากรที่อยู่ในทีมพัฒนาซอฟต์แวร์ตั้งแต่เริ่มพัฒนาจนกระทั่งแล้วเสร็จสมบูรณ์ โดยถ้ามีการหมุนเวียนเพียงเล็กน้อย ซอฟต์แวร์ก็น่าจะมีความต่อเนื่องในการพัฒนา บุคลากรในที่นี้รวมถึง บุคคลในทุกตำแหน่งที่ร่วมอยู่ในทีมพัฒนา เช่น ผู้จัดการโครงการ ผู้ช่วยผู้จัดการโครงการ นักวิเคราะห์ระบบ โปรแกรมเมอร์ พนักงานทั่วไป เลขานุการ เป็นต้น ดังแสดงในตารางที่ 2.22

ตารางที่ 2.22 ความแตกต่างของแต่ละระดับของ PCON

ระดับ	เงื่อนไข
ต่ำมาก	บุคลากรมีการเปลี่ยนแปลงมากกว่าหรือเท่ากับ 48%ต่อปี
ต่ำ	บุคลากรมีการเปลี่ยนแปลงมากกว่าหรือเท่ากับ 24%ต่อปี แต่น้อยกว่า48%ต่อปี
ปานกลาง	บุคลากรมีการเปลี่ยนแปลงมากกว่าหรือเท่ากับ 12%ต่อปี แต่น้อยกว่า24%ต่อปี
สูง	บุคลากรมีการเปลี่ยนแปลงมากกว่าหรือเท่ากับ 6%ต่อปี แต่น้อยกว่า12%ต่อปี
สูงมาก	บุคลากรมีการเปลี่ยนแปลงน้อยกว่า 6%ต่อปี

4) ปัจจัยกลุ่มโครงการ (Project Factors)

4.1) การใช้เครื่องมือซอฟต์แวร์(Use of Software tools: TOOL) พิจารณาจากเครื่องมือที่ใช้ในการพัฒนาระบบว่ามีขีดความสามารถเพียงใด

<u>คำจำกัดความ</u>

ความสามารถหรือประสิทธิภาพของเครื่องมือซอฟต์แวร์ได้พัฒนาจนเป็นที่ยอมรับว่ามีผลต่อการพัฒนาโครงการ เนื่องจากเครื่องมือในสมัยแรกๆสามารถทำได้เพียงการบรรณาธิกร(edit)และเขียน(code)โปรแกรม แต่ต่อมาเครื่องมือได้มีการพัฒนาจนสามารถนำมาใช้ในทุกขั้นของวงจรชีวิตของการพัฒนา(Software Development Life Cycle: SDLC)ได้ เช่นขั้นตอนการวิเคราะห์(Analysis) ขั้นตอนออกแบบ(Design) และขั้นตอนการทำให้เกิดผล(Implementation)

ตารางที่ 2.23 ความแตกต่างของแต่ละระดับของ TOOL

ระดับ	เงื่อนไข
ต่ำมาก	เครื่องมือสามารถสามารถเขียน(Code) บรรณาธิกร(Edit) และตรวจสอบความผิดพลาดในการเขียนโปรแกรมได้(Debug)
ต่ำ	เครื่องมือที่ใช้เป็นแบบเครื่องมือเคสที่ทำฟ้อนเอนด์เคส (Frontend Computer Aided Software Engineering : Frontend CASE) คือ เครื่องมือเคสที่ช่วยพัฒนาซอฟต์แวร์ตามวงจรชีวิตในช่วงแรกๆ เช่น ช่วยในขั้นตอนของการเก็บรวบรวมความต้องการต่างๆเกี่ยวกับซอฟต์แวร์ที่จะพัฒนา(Requirements Phase) ช่วยในขั้นตอนของการวางแผนการพัฒนาซอฟต์แวร์ (Planning Phase) หรือ ช่วยในขั้นตอนการออกแบบ(Design Phase) หรือแบบเครื่องมือเคสที่ทำแบล็กเอนด์เคส(Backend CASE) คือ เครื่องมือเคสที่ช่วยพัฒนาซอฟต์แวร์ตามวงจรชีวิตในช่วงหลัง เช่น ช่วยในขั้นตอนของการทำให้เกิดผล(Implementation Phase) ช่วยในขั้นตอนของการรวม(Integration Phase) หรือช่วยในขั้นตอนของการบำรุงรักษา(Maintenance Phase)

ตารางที่ 2.23 ความแตกต่างของแต่ละระดับของ TOOL (ต่อ)

ระดับ	เงื่อนไข
ปานกลาง	เครื่องมือที่ใช้พัฒนาซอฟต์แวร์สามารถใช้ได้ในทุกขั้นตอนของวงจรชีวิต ในขั้นพื้นฐานได้ นอกจากนี้ยังสามารถทำการรวมบางขั้นตอนของวงจรชีวิตได้
สูง	เครื่องมือที่ใช้ในการพัฒนาซอฟต์แวร์สามารถใช้ได้ทุกขั้นของวงจรชีวิตและยังสามารถทำการรวมทุกขั้นของวงจรชีวิตได้
สูงมาก	เครื่องมือที่ใช้ในการพัฒนาซอฟต์แวร์สามารถใช้ได้ทุกขั้นตามรูปแบบของวงจรชีวิต นอกจากนี้ยังสามารถทำการรวมกันโดยระเบียบวิธีการ(Method) กระบวนการ(Process) หรือการนำกลับมาใช้(Reuse)

4.2) การพัฒนาในหลายสถานที่(Multisite Development: SITE) พิจารณาจากสถานที่และ รูปแบบการสื่อสารในระหว่างการพัฒนาระบบ

ค่าจำกัดความ

ตัวขับค่าใช้จ่ายประเภทนี้ได้ถูกเพิ่มเข้ามาในแบบจำลองโคโคโม2 เนื่องจากการวิจัยพบว่าสถานที่ที่ใช้พัฒนาและอุปกรณ์ที่ใช้ในการสื่อสารระหว่างบุคลากรในทีมพัฒนา มีผลต่อการพัฒนาซอฟต์แวร์ เช่น สถานที่ที่ห่างไกลกันหรืออุปกรณ์การสื่อสารที่ไม่ทันสมัยจะทำให้ไม่สามารถแก้ไขปัญหาได้ทันท่วงที เนื่องจากความล่าช้าในการติดต่อ ดังแสดงในตารางที่ 2.24 และ ตารางที่ 2.25

ตารางที่ 2.24 ความแตกต่างของแต่ละระดับของ SITE เกี่ยวกับสถานที่

ระดับ	เงื่อนไขของสถานที่
ต่ำมาก	สถานที่ที่พัฒนาซอฟต์แวร์มีทั้งในประเทศและต่างประเทศ
ต่ำ	สถานที่ที่พัฒนาซอฟต์แวร์ตั้งอยู่ในหลายจังหวัด และ มีบริษัทที่พัฒนาหลายบริษัท
ปานกลาง	สถานที่ที่พัฒนาซอฟต์แวร์ตั้งอยู่ในหลายจังหวัด หรือ มีบริษัทที่พัฒนาหลายบริษัท หรือ ตั้งอยู่ในจังหวัดที่ไม่สำคัญเพียงจังหวัดเดียว เช่น หนองบัวลำภู ปัตตานี เป็นต้น
สูง	สถานที่ที่พัฒนาซอฟต์แวร์ตั้งอยู่ในกรุงเทพฯ หรือจังหวัดสำคัญๆ(กรุงเทพฯ เชียงใหม่ นครราชสีมา ภูเก็ต สงขลา ..)
สูงมาก	สถานที่ที่พัฒนาซอฟต์แวร์มีหลายตึก หรือ เป็นหมู่ตึก
สูงที่สุด	สถานที่ที่พัฒนาซอฟต์แวร์จะอยู่ในบริเวณเดียวกัน เช่น อยู่ในตึกเดียวกัน อยู่ในชั้นเดียวกัน เป็นต้น

ตารางที่ 2.25 ความแตกต่างของแต่ละระดับของ SITE เกี่ยวกับอุปกรณ์การสื่อสาร

ระดับ	เงื่อนไขของการสนับสนุนด้านการสื่อสาร
ต่ำมาก	เทคโนโลยีสูงสุดที่ใช้ในการสื่อสารระหว่างผู้พัฒนา คือ โทรศัพท์รวม หรือ จดหมาย
ต่ำ	เทคโนโลยีสูงสุดที่ใช้ในการสื่อสารระหว่างผู้พัฒนา คือ โทรศัพท์ส่วนตัวหรือแฟกส่วนตัว
ปานกลาง	เทคโนโลยีสูงสุดที่ใช้ในการสื่อสารระหว่างผู้พัฒนา คือ การใช้ช่องสัญญาณแคบ เช่น ใช้สัญญาณคลื่นวิทยุ เครือข่ายเฉพาะ(private network) เครือข่ายท้องถิ่น(LAN) เป็นต้น
สูง	เทคโนโลยีสูงสุดที่ใช้ในการสื่อสารระหว่างผู้พัฒนา คือ การใช้ช่องสัญญาณกว้าง เช่น ใช้สัญญาณดาวเทียม เครือข่ายสาธารณะ(WAN) เป็นต้น
สูงมาก	เทคโนโลยีสูงสุดที่ใช้ในการสื่อสารระหว่างผู้พัฒนา คือ การประชุมทางไกลผ่านวิดีทัศน์
สูงที่สุด	เทคโนโลยีสูงสุดที่ใช้ในการสื่อสารระหว่างผู้พัฒนา คือ มัลติมีเดียที่สามารถโต้ตอบได้

4.3) กำหนดการการพัฒนาที่ต้องการ(Required Development Schedule: SCED) พิจารณาจากระยะเวลาที่ใช้ในการพัฒนาระบบ

<u>ข้อจำกัดความ</u>

การกำหนดระดับจะขึ้นอยู่กับข้อจำกัดของทีมที่พัฒนา ดังนั้นจึงจะพิจารณาเป็นทีมที่พัฒนา โดยจะเทียบเป็นเปอร์เซ็นต์ของการขยายเวลาการทำงานออกไปหรือการเร่งเวลาการทำงานให้เร็วขึ้นเมื่อเทียบกับเวลาการทำงานปกติ ซึ่งการเร่งตารางการทำงานมีแนวโน้มว่าจะใช้ความพยายามสูงในเฟสท้ายๆของการพัฒนา เนื่องจากอาจมีปัญหาหรือข้อสงสัยหลายอย่างที่ไม่ได้ถูกพิจารณาหรือแก้ไขในเฟสแรกๆ ดังนั้นจึงต้องมาทำในเฟสท้ายๆ และการขยายเวลาการทำงานออกไปมีแนวโน้มว่าจะใช้ความพยายามในเฟสแรกๆมาก เนื่องจากใช้ในการวางแผนการพัฒนาซอฟต์แวร์(Planning) การกำหนดคุณลักษณะ(Specifications)ของซอฟต์แวร์ที่จะพัฒนา และการวางแผนการตรวจสอบความถูกต้อง(Validation)ของซอฟต์แวร์ ดังแสดงในตารางที่ 2.26

ตารางที่ 2.26 ความแตกต่างของแต่ละระดับของ SCED

ระดับ	เงื่อนไข
ต่ำมาก	เวลาในการพัฒนาน้อยกว่าหรือเท่ากับ 75% ของเวลาปกติ
ต่ำ	เวลาในการพัฒนามากกว่า 75%ของเวลาปกติแต่น้อยกว่าหรือเท่ากับ 85%
ปานกลาง	เวลาในการพัฒนามากกว่า 85%ของเวลาปกติแต่น้อยกว่าหรือเท่ากับ100%
สูง	เวลาในการพัฒนามากกว่า 100%ของเวลาปกติแต่น้อยกว่าหรือเท่ากับ130%
สูงมาก	เวลาในการพัฒนามากกว่า 130%ของเวลาปกติแต่น้อยกว่าหรือเท่ากับ160%

ประมาณค่าใช้จ่ายในการพัฒนาซอฟต์แวร์ในแบบจำลองนี้ จะประมาณระยะเวลาที่ใช้ในการพัฒนาซอฟต์แวร์ จำนวนคนที่ใช้พัฒนาต่อเดือน เป็นต้น ซึ่งมีสมการสำหรับประมาณ

การคังสมการที่ (11) ถึง (18) และคำอธิบายในตารางที่ 2.27

สูตรคำนวณจำนวนคนที่ใช้พัฒนาต่อเดือนคือ

$$PM = A*[Size]^{B}\prod_{i=1}^{17} EM_i + PM_{AT} \qquad \cdots(11)$$

โดยที่

$$PM_{AT} = \frac{ASLOC\left(\frac{AT}{100}\right)}{ATPROD} \qquad \cdots(12)$$

และ

$$B = 1.01 + 0.01\sum_{j=1}^{5} SF_j \qquad \cdots(13)$$

สูตรคำนวณขนาดของซอฟต์แวร์ คือ

$$Size = \overline{Size}\left(1+\frac{BRAK}{100}\right) \qquad \cdots(14)$$

โดยที่

$$\overline{Size} = KNSLOC = KASLOC*\left(\frac{100-AT}{100}\right)*(AAM) \qquad \cdots(15)$$

$$AAM = \begin{cases} \dfrac{AA + AAF*(1+0.02*SU*UNFM)}{100}, AAF \le 0.05 \\ \dfrac{AA + AAF + SU*UNFM}{100}, AAF > 0.05 \end{cases} \qquad \cdots(16)$$

$$AAF = 0.4*DM + 0.3*CM + 0.3*IM \qquad \cdots(17)$$

สูตรคำนวณจำนวนระยะเวลาที่ใช้ในการพัฒนาซอฟต์แวร์ คือ

$$TDEV = \left[3.0*(PM)^{(0.33+0.2*(B-1.01))}\right]\frac{SECD\%}{100} \qquad \cdots(18)$$

ตารางที่ 2.27 สัญญาลักษณ์และคำอธิบายของสมการจากโพสอาร์คิเท็กเจอร์โมเดล

สัญลักษณ์	คำอธิบาย
A	ค่าคงที่ กำหนดให้เป็น 2.5
AA	การประเมินสิ่งที่สามารถนำมาใช้ได้(Assessment and assimilation)
AT	เปอร์เซ็นต์ขององค์ประกอบที่สามารถแปลงได้โดยอัตโนมัติ (Percentage of components that are automatically translated)
ATPROD	ตัวคูณของการแปลงโดยอัตโนมัติ(Automatic translation productivity)
BRAK	เปอร์เซ็นต์ของการเปลี่ยนแปลงที่มีในความต้องการ(Requirements)
CM	เปอร์เซ็นต์ของการเปลี่ยนแปลงโปรแกรม(Percentage of code modified)
DM	เปอร์เซ็นต์ของการเปลี่ยนแปลงการออกแบบ(Percentage of design modified)
EM	ตัวคูณความพยายาม (Effort Multipliers : RELY, DATA, CPLX, RUSE, DOCU, TIME, STOR, PVOL, ACAP, PCAP, AEXP, PEXP, LTEX, PCON, TOOL, SITE, SCED) ดังแสดงในตารางที่ 2.28
IM	เปอร์เซ็นต์ของการเปลี่ยนแปลงการรวมและการทดสอบระบบ (Percentage of integration and test modified)
KASLOC	ขนาดของโปรแกรมที่ได้รับการปรับปรุง มีหน่วยเป็น หนึ่งพันบรรทัด
KNSLOC	ขนาดของโปรแกรมที่สร้างใหม่ มีหน่วยเป็น หนึ่งพันบรรทัด
PM	จำนวนคนที่พัฒนาต่อหนึ่งเดือน
SECD%	เปอร์เซ็นต์ของการลดและเพิ่มเวลาที่ใช้พัฒนาซอฟต์แวร์
SF	ค่าสเกลของปัจจัย(Scale Factors : PREC, FLEX, RESL, TEAM, PMAT)
SU	ระดับความเข้าใจของผู้พัฒนาที่มีต่อโปรแกรม(Software Understanding)
TDEV	เวลาที่ใช้พัฒนาซอฟต์แวร์มีหน่วยเป็นเดือน (Time to develop)
UNFM	ความคุ้นเคยของโปรแกรมเมอร์ที่มีต่อซอฟต์แวร์(Programmer Unfamiliarity with Software)

ตารางที่ 2.28 ค่าตัวคูณความพยายามในแต่ละระดับ

ตัวขับค่าใช้จ่าย	ระดับ					
	ต่ำมาก	ต่ำ	ปานกลาง	สูง	สูงมาก	สูงที่สุด
RELY	0.75	0.88	1.00	1.15	1.39	
DATA		0.93	1.00	1.09	1.19	
CPLX	0.75	0.88	1.00	1.15	1.30	1.66
RUSE		0.91	1.00	1.14	1.29	1.49
DOCU	0.89	0.95	1.00	1.06	1.13	
TIME			1.00	1.11	1.31	1.67
STOR			1.00	1.06	1.21	1.57
PVOL		0.87	1.00	1.15	1.30	
ACAP	1.50	1.22	1.00	0.83	0.67	
PCAP	1.37	1.16	1.00	0.87	0.74	
AEXP	1.22	1.10	1.00	0.89	0.81	
PEXP	1.25	1.10	1.00	0.88	0.81	
PCON	1.24	1.10	1.00	0.92	0.84	
LTEX	1.22	1.10	1.00	0.91	0.84	
TOOL	1.24	1.12	1.00	0.86	0.72	
SITE	1.25	1.10	1.00	0.92	0.84	0.78
SCED	1.29	1.10	1.00	1.00	1.00	

2.1.2 เทคนิคเดลฟาย(Delphi Technique)

ในสารานุกรมเรื่องทฤษฎีและรูปแบบที่ใช้ในพฤติกรรมศาสตร์ประยุกต์[5] ระบุว่า เทคนิคเดลฟายเป็นเทคนิคที่มีประโยชน์ โดยเฉพาะในกรณีที่จะทำให้บุคคลที่เกี่ยวข้องเรื่องใดเรื่องหนึ่งที่อยู่ในที่ต่างๆ ไม่สามารถรวบรวมมาประชุมได้ และประโยชน์อีกอย่างหนึ่งของเทคนิคนี้คือ ให้โอกาสสมาชิกทุกคน ได้แสดงความคิดเห็นของตนอย่างอิสระ ซึ่งอาจมีความคิดเห็นที่แตกต่างกันออกไป ซึ่งทำให้การศึกษามีความละเอียดมากยิ่งขึ้น การศึกษาแบบเดลฟายเป็นเทคนิคที่ใช้ในการรวบรวมความคิดเห็นของกลุ่มผู้เชี่ยวชาญในประเด็นใดประเด็นหนึ่งที่ต้องการรู้ และเป็นเทคนิคที่ใช้หาความคิดเห็นที่สอดคล้องเป็นอันหนึ่งอันเดียวที่ไม่จำเป็นต้องให้ผู้เชี่ยวชาญเหล่านั้นมานั่งเผชิญหน้ากันในที่ประชุม โดยทั่วไปแล้วจะใช้วิธีการติดต่อกับผู้เชี่ยวชาญทางไปรษณีย์ เทคนิคนี้ได้รับการยอมรับในกลุ่มนักวิชาการทางการศึกษาอย่างมาก เนื่องจากเป็นวิธีที่มีระบบ ใช้ความคิดเห็นของกลุ่มผู้เชี่ยวชาญในเรื่องนั้นๆ อย่างแท้จริง และผู้เชี่ยวชาญแต่ละคนสามารถแสดงความคิดเห็นได้อย่างอิสระ โดยไม่ต้องคำนึงถึงความคิดเห็นของผู้อื่นและไม่

ต้องอยู่ภายใต้อิทธิพลของเสียงส่วนใหญ่หรือความคิดเห็นของผู้เชี่ยวชาญคนอื่นๆ ลักษณะของเทคนิคเดลฟาย มีดังต่อไปนี้

1)เทคนิคเดลฟายมุ่งแสวงหาความคิดเห็นของกลุ่มด้วยแบบสอบถาม ดังนั้นผู้เข้าร่วมโครงการจึงจำเป็นต้องตอบแบบสอบถามที่ผู้ทำการวิจัยได้กำหนดขึ้นในแต่ละขั้นตอน

2)เทคนิคเดลฟายไม่ต้องการให้ความคิดเห็นของผู้อื่นมีอิทธิพลต่อการพิจารณาตอบแบบสอบถาม ผู้ตอบแบบสอบถามแต่ละคนจะไม่ทราบว่า ผู้ใดบ้างที่ได้รับเลือกให้ทำแบบสอบถาม และไม่ทราบว่าแต่ละคนมีความคิดเห็นในแต่ละข้ออย่างไร ผู้ตอบแบบสอบถามจะมีความคิดเป็นอิสระ

3)เพื่อให้ผู้เชี่ยวชาญแต่ละคนตอบแบบสอบถามด้วยความคิดเห็นที่กลั่นกรองอย่างละเอียดรอบคอบ และเพื่อให้คำตอบที่ได้รับมีความเป็นอันหนึ่งอันเดียวกันมากยิ่งขึ้น ผู้ทำการวิจัยจะแสดงตำแหน่งของความคิดเห็นที่ผู้เชี่ยวชาญมีความเห็นสอดคล้องกัน ในคำตอบแต่ละข้อของแบบสอบถามที่ตอบไปในครั้งก่อนในรูปสถิติ และผู้ทำวิจัยจะส่งกลับไปให้แต่ละคนทราบ เพื่อจะได้ตัดสินใจว่าจะคงคำตอบเดิมหรือปรับปรุงเปลี่ยนแปลงคำตอบจากเดิมอย่างไรบ้าง ดังนั้นในการตอบแบบสอบถามของผู้เชี่ยวชาญแต่ละคนนั้น ผู้เชี่ยวชาญจะทราบว่าความคิดเห็นของตนเป็นอย่างไรต่างจากคนอื่นๆหรือไม่

4)การวิจัยแบบใช้เทคนิคเดลฟาย จะใช้สถิติวิเคราะห์การทำงานของกลุ่มทุกครั้ง สถิติที่นำมาใช้ในการพิจารณาคำตอบจากแบบสอบถามในแต่ละรอบคือ สถิติการวัดแนวโน้มเข้าสู่ศูนย์กลาง ได้แก่ ค่ามัธยฐาน(Median) ฐานนิยม(Mode) และค่าเฉลี่ย(Average) เพื่อแสดงตำแหน่งของความคิดเห็น โดยทั่วไปจะไม่ใช้ค่าเฉลี่ยกับข้อมูลที่มีจำนวนน้อย ดังนั้นจึงนิยมใช้ค่ามัธยฐานและฐานนิยม เพื่อแสดงความคิดเห็นเกี่ยวกับเวลา ปริมาณหรือสถานการณ์ต่างๆในอนาคตมากกว่า

ข้อได้เปรียบและข้อเสียของเทคนิคเดลฟาย

กระบวนการวิจัยทุกชนิดย่อมมีจุดเด่นและจุดด้อยอยู่เสมอ การที่นักวิจัยจะตัดสินใจเลือกกระบวนการวิจัยแบบใดควรจะต้องพิจารณาถึงจุดเด่นและจุดด้อยของกระบวนการวิจัยด้วย เพื่อจะได้เลือกใช้กระบวนการวิจัยที่เหมาะกับประเด็นปัญหาที่ต้องการศึกษา

ข้อได้เปรียบของเทคนิคเดลฟาย

1) สามารถหาความสอดคล้องของความคิดเห็นจากกลุ่มผู้เชี่ยวชาญได้ โดยไม่ต้องมีการจัดประชุมกลุ่มผู้เชี่ยวชาญ ซึ่งทำได้ยากในกรณีที่ผู้เชี่ยวชาญอยู่ในที่ที่ห่างไกลกัน

2) ข้อมูลที่ได้จะเป็นคำตอบที่น่าเชื่อถือ เพราะว่าเป็นความคิดเห็นจากกลุ่มผู้เชี่ยวชาญในสาขาวิชานั้นอย่างแท้จริง และได้มาจากการย้ำถามหลายๆรอบ จึงเป็นคำตอบที่ได้มาจากการกลั่นกรองหลายรอบ นอกจากนี้ผู้เชี่ยวชาญแต่ละคนแสดงความคิดเห็นของตนอย่างอิสระ ไม่ตกอยู่ภายใต้อิทธิพลหรือเสียงส่วนใหญ่ เพราะผู้เชี่ยวชาญแต่ละคนไม่ทราบว่าผู้เชี่ยวชาญท่านอื่นๆมีความคิดเห็นอย่างไร

3) เทคนิคเดลฟายสามารถรับข้อมูลจากคนจำนวนมาก โดยไม่มีข้อจำกัดทางภูมิศาสตร์

4) เป็นวิธีการระดมความคิดเห็นของผู้เชี่ยวชาญได้อย่างรวดเร็วและมีประสิทธิภาพสูง

5) ค่าใช้จ่ายในการดำเนินการค่อนข้างต่ำ

6) ผู้วิจัยสามารถทราบลำดับความสำคัญของข้อมูล และเหตุผลในการตอบรวมทั้งความสอดคล้องในเรื่องความคิดเห็นเป็นอย่างดี

ข้อเสียเปรียบของเทคนิคเดลฟาย

1) การคัดเลือกผู้เชี่ยวชาญเป็นส่วนสำคัญของเทคนิคเดลฟาย ความเชื่อถือได้ของงานวิจัยขึ้นอยู่กับผู้เชี่ยวชาญที่ร่วมตอบแบบสอบถามในการวิจัย ถ้าผู้วิจัยไม่กำหนดเกณฑ์ที่ชัดเจนเอาไว้ก็อาจจะไม่ได้ผู้เชี่ยวชาญในด้านที่ต้องการอย่างแท้จริง ซึ่งทำให้ข้อมูลที่ได้รับผิดพลาดไปด้วย

2) การถามซ้ำหลายๆ รอบตามกระบวนการวิจัยนั้น อาจทำให้ผู้เชี่ยวชาญเกิดความเบื่อหน่าย ไม่เต็มใจให้ความร่วมมือ ซึ่งจะมีผลต่อความเชื่อมั่นของข้อมูล

3) แบบสอบถามที่ส่งไปอาจสูญหายระหว่างทางหรือได้รับกลับมาไม่ครบในแต่ละรอบ

ลักษณะของผู้เชี่ยวชาญที่ได้รับคัดเลือกเพื่อให้ตอบแบบสอบถามในงานวิจัยนี้

1)กำหนดเกณฑ์การเลือกผู้เชี่ยวชาญ

- สำเร็จการศึกษาตั้งแต่ระดับปริญญาเอก
- มีประสบการณ์การเขียนโปรแกรมไม่ต่ำกว่า 5 ปี หรือมีประสบการณ์การออกแบบระบบ หรือมีประสบการณ์ด้านการบริหารโครงการ หรือเป็นผู้ที่มีประสบการณ์ทางวิชาการหรืองานวิจัยที่เกี่ยวกับการบริหารโครงการ

2)จำนวนผู้เชี่ยวชาญ

จำนวนผู้เชี่ยวชาญที่จะใช้ในการทำแบบสอบถามนั้น โดยทั่วไปไม่มีข้อกำหนดที่แน่นอนว่าควรจะใช้ผู้เชี่ยวชาญกี่คน แต่ในปี พ.ศ.2514 นายโทมัส ที แมคลิแลน(Thomas T. Machillan)[6] ได้เสนอผลการวิจัยเกี่ยวกับจำนวนผู้เชี่ยวชาญในการวิจัยแบบเดลฟายว่า ควรมีจำนวนเท่าใดจึงจะเหมาะสม ในการประชุมประจำปีของสมาคมโรงเรียนมัธยมศึกษาตอนต้นแห่งรัฐแคลิฟอร์เนีย (California Junior Colleges Association) พบว่าหากมีจำนวนตั้งแต่ 17 คนขึ้นไป อัตราการลดลงของความคลาดเคลื่อนจะมีน้อยลงมาก ดังแสดงในตารางที่ 2.29

ตารางที่ 2.29 การลดลงของความคลาดเคลื่อนของจำนวนผู้เข้าร่วมโครงการ

จำนวนผู้เชี่ยวชาญ	ช่วงของความคลาดเคลื่อน	ความคลาดเคลื่อนลดลง
1-5	1.02-0.70	0.50
5-9	0.70-0.58	0.12
9-13	0.58-0.54	0.04
13-17	0.54-0.50	0.04
17-21	0.50-0.48	0.02
21-25	0.48-0.46	0.02
25-29	0.46-0.44	0.02

ดังนั้นจึงอาจสรุปได้ว่าจำนวนผู้เชี่ยวชาญที่ใช้ในการวิจัยแบบเดลฟายไม่ควรต่ำกว่า 17 คนแต่ก็ไม่ควรมากเกินไป เพราะจะทำให้เกิดความล่าช้าในการติดตามผล

2.1.3 การสร้างคำถาม

การสร้างคำถามเพื่อใช้ในการทดสอบ เก็บข้อมูล หรือรวบรวมความคิดเห็น เช่น สร้างในลักษณะของแบบสอบถาม ควรมีองค์ประกอบดังนี้

1) ประเภทของแบบสอบถาม

ผู้สร้างควรทราบก่อนว่าคำถามที่สร้างควรจะสร้างในลักษณะใดจึงจะเหมาะสมกับงาน ซึ่งลักษณะแบบสอบถามมีหลายประเภทแต่ที่นิยมใช้กันมากคือ

1.1) แบบสอบถามปลายเปิด(Opened question) เป็นแบบสอบถามที่ตั้งคำถามกว้าง การใช้แบบสอบถามแบบนี้ต้องการความคิดเห็นต่างๆ โดยมากนิยมใช้เพื่อเป็นแนวทางในการสร้างแบบสอบถามแบบปลายปิด

1.2) แบบสอบถามปลายปิด(Closed question) เป็นแบบสอบถามที่ผู้สร้างมีจุดหมายแน่นอนอยู่ในใจ ข้อความต่างๆได้มาจากการตอบแบบสอบถามปลายเปิดจากผู้รู้ หนังสือ วารสาร สิ่งพิมพ์ ตลอดจนแหล่งข้อมูลต่างๆ แบ่งเป็น

- ให้เลือกตอบเพียงข้อเดียวใน 2 ข้อ
- ให้เลือกตอบเพียงข้อเดียวในหลายข้อ
- แบบสเกลประมาณค่า

- แบบเติมคำเช่น

อายุของท่าน.....ปี

2) ขั้นตอนการออกแบบสอบถาม

2.1) ต้องมีความรู้ และเข้าใจในเรื่องที่ตนวิจัยนั้นเป็นอย่างดี

2.2) ต้องทราบวัตถุประสงค์ เป้าหมาย และสมมติฐานของการวิจัยชัดแจ้ง

2.3) ยกร่างแบบสอบถาม โดยให้ตรงกับวัตถุประสงค์ของการวิจัย

2.4) ตรวจแก้แบบสอบถามที่ได้ยกร่างแล้ว

2.5) ทดสอบแบบสอบถาม

2.6) ปรับปรุง และแก้ไขแบบสอบถาม

2.7) พิมพ์แบบสอบถาม

2.8) นำแบบสอบถามไปเก็บรวบรวมข้อมูล

3) ขั้นตอนการให้ผู้เชี่ยวชาญตรวจสอบแบบสอบถาม

การตรวจสอบควรรวมถึงสิ่งต่างๆ ที่ได้กล่าวมาแล้วทั้งหมดในการร่างคำถามและอาจสรุปย่อได้ดังนี้

3.1) จำเป็นหรือไม่ที่ต้องมีคำถามนี้ มีความสัมพันธ์กับปัญหาหรือสมมติฐานอย่างไร

3.2) คำถามชัดเจนหรือไม่ ในการคำถามหนึ่งควรมีความหมายเดียว ถ้อยคำและประโยคที่ใช้ครอบคลุมข้อมูลที่ต้องการทราบทั้งหมดหรือไม่

3.3) คำถามปิดมีคำตอบไว้ให้เลือกพอหรือไม่

3.4) คำถามต้องง่ายพอที่ผู้ตอบเข้าใจความหมาย หลีกเลี่ยงคำยากหรือศัพท์ทางวิชาการ

3.5) ต้องคำนึงว่าคำถามนั้นควรใช้คำถามโดยตรงหรือคำถามทางอ้อม จึงจะได้คำตอบที่ตรงกับวัตถุประสงค์มากที่สุด

3.6) หลีกเลี่ยงคำถามที่ไวต่อความรู้สึก(Sensitive) หรือทำให้เกิดความละอายใจหรืออึดอัดใจ (Embarrassment) เช่นเรื่องหนี้สิน รายได้

3.7) คำถามที่มีข้อความกำกวมหลายนัย ควรแยกออกต่างหากหรือถามให้ชัดเจน

3.8) การสร้างคำถามควรจะให้มีความสะดวกในการเก็บรวบรวมข้อมูล

3.9) ตรวจดูการจัดหน้า การเรียงพิมพ์ ตัวสะกด การันต์ วรรคตอน เพราะเป็นหน้าเป็นตาของผู้ออกแบบสอบถาม ถ้ามีการผิดพลาดจะทำให้ผู้ตอบคำถามสังเกในการตอบและไม่อยากจะเชื่อในความสามารถทางวิชาการ ความละเอียดรอบคอบของผู้ออกแบบสอบถามมักทำให้เกิดทัศนคติที่ไม่ดีต่อผู้ออกแบบสอบถามในเบื้องแรก

2.1.4 วิธีวิเคราะห์ข้อมูลที่ได้จากแบบสอบถาม

ผู้วิจัยวิเคราะห์แบบสอบถามโดยการหาค่ามัธยฐาน ฐานนิยม และพิสัยระหว่างควอไทล์เป็นรายข้อ และเลือกระดับคะแนนที่มีค่าตั้งแต่ 3.5 ขึ้นไป มาสร้างแบบสอบถามในรอบที่2 และนำคำตอบที่ได้ในรอบที่ 2 มาวิเคราะห์หาค่ามัธยฐาน และพิสัยระหว่างควอไทล์ สำหรับการแปลผลแต่ละข้อนั้น ผู้วิจัยได้กำหนดเกณฑ์ดังนี้

1) มัธยฐาน(Median)

การคำนวณหาค่ามัธยฐานของแต่ละข้อใช้สูตร

$$Median = Lo + i\left[\frac{\frac{N}{2} - f1}{f2}\right]$$

เมื่อ Median = มัธยฐาน

Lo = ขีดจำกัดล่างที่แท้จริงของคะแนนในชั้นที่มีมัธยฐาน

N = จำนวนความถี่ทั้งหมด

i = อันตรภาคชั้น

f_1 = ความถี่สะสมจากคะแนนต่ำสุด ถึงคะแนนที่เป็นขีดจำกัดบนของคะแนนในชั้นที่มีมัธยฐาน

f_2 = ความถี่ของคะแนนในชั้นที่มีมัธยฐาน

2) พิสัยระหว่างควอไทล์

พิสัยระหว่างควอไทล์(Interquartile Range) คือค่าความแตกต่างระหว่างควอไทล์ที่ 3 กับควอไทล์ที่ 1 โดยใช้สูตร

$$Q1 = Lo + i\left[\frac{\frac{N}{4} - f1}{f2}\right]$$

$$Q3 = Lo + i\left[\frac{\frac{3N}{4} - f1}{f2}\right]$$

เมื่อ Q1 = ควอไทล์ที่ 1

Q3 = ควอไทล์ที่ 3

Lo = ขีดจำกัดล่างชั้นที่มีคะแนน ณ ตำแหน่งที่ควอไทล์

i = อันตรภาคชั้น

N = ความถี่สะสมทั้งหมด

f_1 = ความถี่สะสมจากคะแนนต่ำที่สุดถึงชั้นที่มาก่อนชั้นที่มีคะแนน ณ ตำแหน่งที่มีควอไทล์

f_2 = ความถี่ของคะแนนในชั้นที่มีคะแนนต่ำ ณ ตำแหน่งที่มีควอไทล์

3) ฐานนิยม(Mode)

$$Mo = Lmo + \left[\frac{d1}{d1 + d2}\right]w$$

เมื่อ Mo = ฐานนิยม

Lmo = ขีดจำกัดล่างของคะแนนที่มีค่าฐาน

d_1 = ผลต่างระหว่างความถี่ของชั้นคะแนนที่มีค่าฐานนิยมกับความถี่ของชั้นที่มีคะแนนน้อย ซึ่งอยู่ติดกับชั้นที่มีค่าฐานนิยม

d_2 = ผลต่างระหว่างความถี่ของชั้นคะแนนที่มีค่าฐานนิยมกับความถี่ของชั้นที่มีคะแนนมาก ซึ่งอยู่ติดกับชั้นที่มีค่าฐานนิยม

w = อันตรภาคชั้น

2.1.5 โปรแกรมการเรียนรู้เชิงอุปนัย

ในงานวิจัยนี้จะใช้โปรแกรมการเรียนรู้C4.5 ในการสร้างต้นไม้การตัดสินใจ เพื่อช่วยในการวิเคราะห์หาค่าสเกล(Rating Scale)ที่เหมาะสม โปรแกรมการเรียนรู้C4.5[7]เป็นโปรแกรมการเรียนรู้เชิงอุปนัยที่พัฒนาต่อมาจาก ID3 (Induction Decision Tree) ซึ่งเป็นโปรแกรมการเรียนรู้จากตัวอย่างที่อาศัยวิธีการจัดหมวดหมู่(Classification Model) จากตัวอย่างเฉพาะที่เรียกว่าข้อมูลสอน(Training Data) เพื่อนำมาสร้างเป็นต้นไม้ตัดสินใจ(Decision Tree) และ กฎการตัดสินใจ(Rule) ได้โดยอัตโนมัติ

1) ข้อมูลสอนจะมีลักษณะคล้ายกับข้อมูลในฐานข้อมูลแบบสัมพันธ์(Relational Database) ที่ประกอบด้วยแถว(Record) หรือตัวอย่าง(Case) และสดมภ์(Column) หรือลักษณะ(Attribute) ซึ่งมี 2 ชนิดด้วยกันคือ

1.1) ลักษณะแบ่งกลุ่ม(Category Attribute) หรือ กลุ่ม(Class) เป็นการกำหนดว่าตัวอย่างนั้นๆ ถูกจัดไว้ในกลุ่มใด โดยจะมีได้เพียง 1 กลุ่มเท่านั้นในแต่ละตัวอย่าง ดังแสดงในตารางที่ 2.30

ตารางที่ 2.30 ตัวอย่างลักษณะแบ่งพวก

ลักษณะ	ค่าที่เป็นไปได้(กลุ่ม)
ค่าสเกลของตัวขับค่าใช้จ่าย	ต่ำมาก , ต่ำ, ปานกลาง, สูง, สูงมาก, สูงที่สุด

1.2) ลักษณะไม่แบ่งกลุ่ม(Non-Category Attribute) เป็นลักษณะต่างๆ ของตัวอย่าง โดยอาจจะเก็บข้อมูลได้ทั้งชนิดค่าต่อเนื่องและ ค่าไม่ต่อเนื่อง ดังแสดงในตารางที่ 2.31

ตารางที่ 2.31 ตัวอย่างลักษณะไม่แบ่งพวก

ลักษณะ	ค่าที่เป็นไปได้
1)ซอฟต์แวร์ที่พัฒนาจะถูกนำไปใช้ในด้านที่อาจเกิดความเสี่ยงกับชีวิตมนุษย์ หรือมีผลกระทบต่อชีวิตมนุษย์ ใช่หรือไม่	ใช่ ไม่ใช่
2)ซอฟต์แวร์ที่จะพัฒนาเป็นซอฟต์แวร์ที่เกี่ยวข้องกับรายรับ-รายจ่ายขององค์กรเป็นอย่างมาก ซึ่งเมื่อซอฟต์แวร์ไม่สามารถทำงานได้จะกระทบต่อรายรับ-รายจ่ายขององค์กร ใช่หรือไม่	ใช่ ไม่ใช่
3) ถ้าซอฟต์แวร์ไม่สามารถทำงานได้ตามปกติ ท่านจำเป็นต้องดำเนินการให้มีการแก้ไขทันที ใช่หรือไม่	ใช่ ไม่ใช่
4)ท่านสามารถดำเนินงานในลักษณะนั้นต่อได้ ถึงแม้ว่าซอฟต์แวร์จะไม่สามารถทำงานได้ก็ตาม ใช่หรือไม่ เช่นการใช้บุคลากรทำงานแทน เป็นต้น	ใช่ ไม่ใช่

2) ต้นไม้การตัดสินใจ

การสร้างต้นไม้การตัดสินใจจะเริ่มที่ราก(root)ก่อนแล้วจึงไล่ลงมาที่ข้อ(node) และใบ(leaf) ซึ่งข้อ คือข้อมูลลักษณะไม่แบ่งพวก ส่วนใบคือข้อมูลลักษณะแบ่งพวก ดังแสดงในรูปที่ 2.1 จะเห็นว่าในการสร้างต้นไม้การตัดสินใจ จะเป็นการแบ่งตัวอย่างออกเป็นกลุ่มๆ ตามลักษณะไม่แบ่งกลุ่ม จนกระทั่งได้กลุ่มตัวอย่างที่เป็นกลุ่มเดียวกัน ซึ่งเป็นคำตอบหรือการตัดสินใจ จากชุดของคำถามและคำตอบข้างต้นจะได้ตัวอย่างดังตารางที่ 2.32 ซึ่งประกอบด้วยลักษณะไม่แบ่งกลุ่ม 7 ลักษณะ และลักษณะแบ่งกลุ่ม ที่แบ่งตัวอย่างออกเป็น 5 กลุ่ม (สูงมาก,สูง,ปานกลาง,ต่ำ,ต่ำมาก)

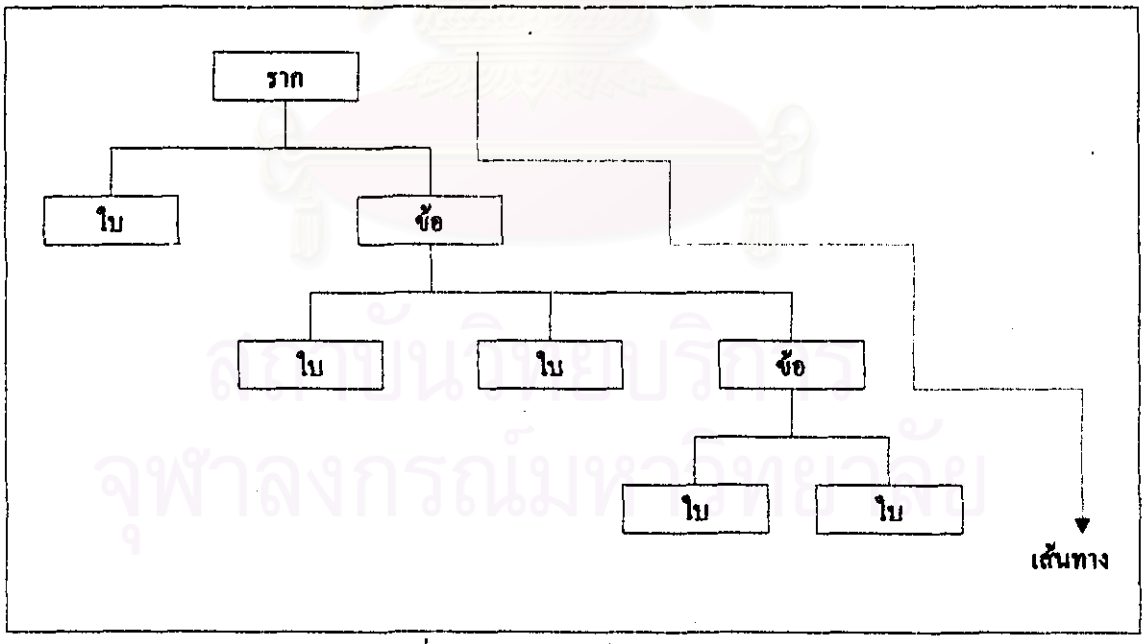
2.1) ราก เป็นจุดเริ่มต้นหรือ ข้อแรกของคำถาม โดยคำตอบจะเป็นค่าที่เป็นไปได้ของลักษณะไม่แบ่งพวกบนข้อมูลตัวอย่าง ถ้าคำตอบตรงกับค่าใดบนนี้ ก็จะวิ่งไปสู่กิ่งหรือใบต่อไป

2.2) ข้อ เป็น จุดข้อคำถามตามลักษณะไม่แบ่งพวก

2.3) กิ่ง เป็นค่าที่เป็นไปได้ตามลักษณะไม่แบ่งพวก ซึ่งนำไปสู่ของหรือใบ

2.4) ใบ เป็นคำตอบหรือการตัดสินใจ โดยจะเป็นค่าที่เป็นไปได้ของลักษณะแบ่งพวก

2.5) เส้นทาง (Tree Path) เป็นทางเดินตั้งแต่รากจนถึงใบแต่ละใบ ซึ่งจะนำไปสู่กฎต่อไป



รูปที่ 2.1 ลักษณะของต้นไม้

ตารางที่ 2.32 ตัวอย่างของตัวขับค่าใช้จ่ายประเภทความเชื่อถือได้ของผลิตภัณฑ์ซอฟต์แวร์

ตัวขับค่าใช้จ่าย	คำถาม				สรุประดับ
RELY	ข้อที่1	ข้อที่2	ข้อที่3	ข้อที่4	
ตัวอย่างที่ 1	ใช่	ไม่ใช่	ไม่ใช่	ไม่ใช่	สูงมาก
ตัวอย่างที่ 2	ใช่	ใช่	ไม่ใช่	ไม่ใช่	สูงมาก
ตัวอย่างที่ 3	ไม่ใช่	ใช่	ใช่	ไม่ใช่	สูง
ตัวอย่างที่ 4	ไม่ใช่	ใช่	ไม่ใช่	ใช่	สูง
ตัวอย่างที่ 5	ไม่ใช่	ไม่ใช่	ใช่	?	ปานกลาง
ตัวอย่างที่ 6	ไม่ใช่	ไม่ใช่	ใช่	ใช่	ปานกลาง
ตัวอย่างที่ 7	ไม่ใช่	ไม่ใช่	ไม่ใช่	ไม่ใช่	ต่ำ
ตัวอย่างที่ 8	ไม่ใช่	ไม่ใช่	ไม่ใช่	ใช่	ต่ำที่สุด

2.2 งานวิจัยที่เกี่ยวข้อง

งานวิจัยเรื่องการพัฒนาโปรแกรมสำหรับประมาณค่าใช้จ่ายซอฟต์แวร์ โดยใช้แนวความคิดตามแบบจำลองโคโคโม[8] ซึ่งเป็นการสร้างโปรแกรมที่เป็นเครื่องมือช่วยให้ผู้จัดการโครงการสามารถประมาณค่าใช้จ่ายซอฟต์แวร์ โดยใช้แนวความคิดตามแบบจำลองโคโคโม โดยรวบรวมข้อมูลและนำมาวิจัยเพื่อสร้างเป็นสมการค่าใช้จ่ายของความพยายาม (Effort) และ กำหนดการ (Schedule)ของหน่วยงาน โดยได้ปรับค่าคงที่ให้เหมาะสมที่สุดด้วยวิธีกำลังสองน้อยที่สุดในตัวขับค่าใช้จ่ายบางตัว แต่งานวิจัยนี้ผู้วิจัยไม่ได้ออกแบบการให้ความช่วยเหลือผู้ใช้ให้สามารถกำหนดระดับของตัวขับค่าใช้จ่ายมากนัก