การบริหารโครงงาน

โครงการจะต้องมีลักษณะสำคัญ ได้แก่ วัตถุประสงค์ชัดเจน มีระยะเวลาเริ่มต้นและสิ้นสุด ประกอบไปด้วย กลุ่มของงาน ดำเนินงานภายใต้เงื่อนไขของเวลา ต้นทุน และคุณภาพของงาน นอกจากนี้โครงการยังมีลักษณะเป็นแบบชั่วคราว คือเกิดขึ้นและสิ้นสุดลงในช่วงเวลาเวลาหนึ่ง ซึ่งอยู่กับการดำเนินงานและทรัพยากรให้มีประสิทธิภาพมากที่สุด นั่นคือ **“การบริหารโครงการ”**

การสร้างแผนงาน

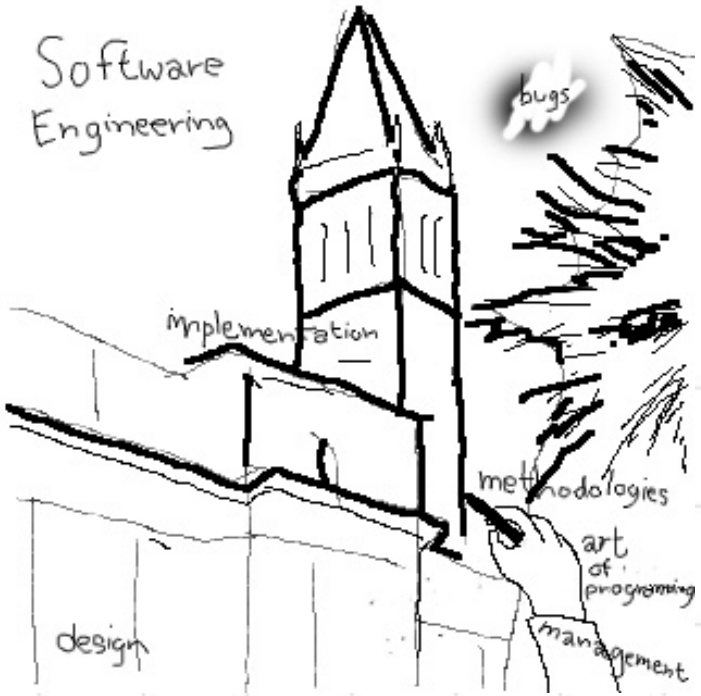
เมื่อผู้พัฒนาระบบและลูกค้าได้พบปะพูดคุยกันถึงรายละเอียดที่ลูกค้าต้องการ มักมีคำถามกลับเสมอว่า

- คุณเข้าใจปัญหาและความต้องการของฉันใช่ไหม ?
- คุณสามารถสร้างระบบที่สามารถแก้ปัญหานี้ได้ใช่ไหม ?
- คุณจะใช้ระยะเวลานานเท่าใด ?
- คุณจะคิดราคาเท่าไร ?

การบริหารโครงงาน

สำหรับคำถามเกี่ยวกับ ระยะเวลา และ ราคา สามารถหาคำตอบได้จากการสร้างแผนงานของโครงการ (Project Schedule)

“วัฏจักรในการพัฒนาซอฟต์แวร์ที่แบ่งออกเป็นระยะ (Stages) หรือเฟส (Phases) แต่ละระยะแบ่งออกเป็นงาน (task) หรือ กิจกรรม (Activities) วิเคราะห์ความสัมพันธ์ของกิจกรรมต่าง ๆ ในระบบ และทำการประมาณในแต่ละกิจกรรม นำมารวมเป็นระบบใหญ่ ก็สามารถประมาณราคาและเวลาได้”



การบริหารโครงงาน

ระยะของการบริหารโครงการ

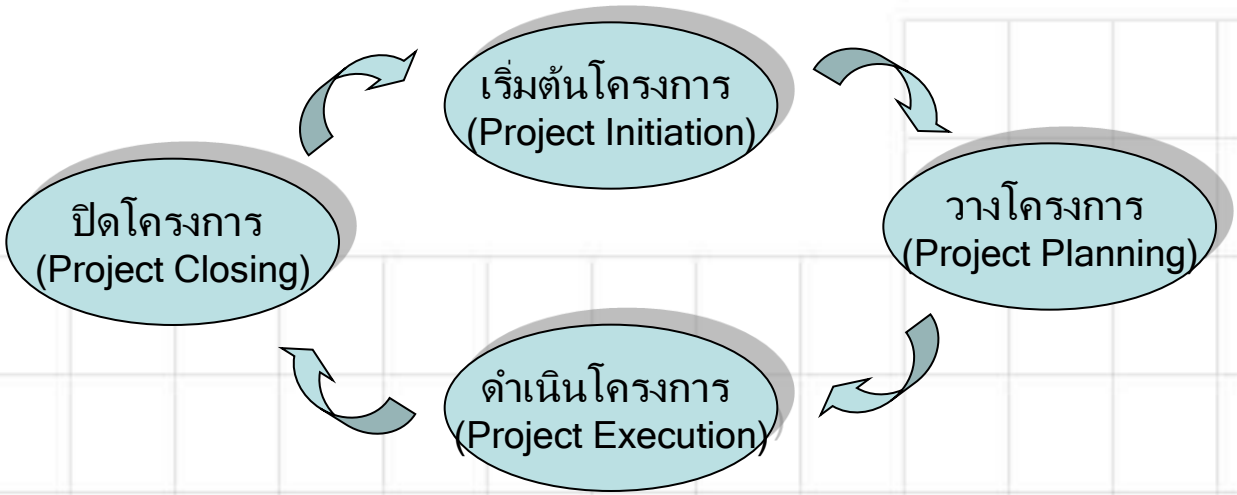
1. **ระยะเริ่มต้นโครงการ (Project Initiation)**
   ทีมงานที่รับผิดชอบต้องเริ่มต้นโครงการด้วยการกำหนดขอบเขตและขนาดของโครงการ รวมทั้งกำหนดกิจกรรมหรืองานที่ต้องทำในแต่ละขั้นตอนของการผลิตซอฟต์แวร์
2. **ระยะวางแผนโครงการ (Project Planning)**
   ผู้บริหารโครงการต้องมีการกำหนดกิจกรรมในแต่ละขั้นตอนของการผลิตอย่างชัดเจน รวมถึงประมาณการใช้ทรัพยากรต่าง ๆ ไม่ว่าจะเป็นเวลา ต้นทุน และแรงงาน นอกจากนี้ ยังต้องจัดตาราง ประเมินความเสี่ยง ตลอดจนกิจกรรมอื่น ๆ
3. **ระยะดำเนินโครงการ (Project Execution)**
   ทีมงานดำเนินกิจกรรมการผลิตซอฟต์แวร์ตามตารางที่จัดไว้ โดยผู้บริหารโครงการต้องติดตาม ดูแล สั่งการ และควบคุมการทำงานของลูกทีมให้เป็นไปตามแผนที่กำหนด ในระหว่างการดำเนินงาน สิ่งที่ขาดไม่ได้เมื่อดำเนินงานในแต่ละกิจกรรมเสร็จสิ้น คือ รายงานความคืบหน้าของโครงการให้ผู้มีอำนาจได้รับทราบ

ระยะของการบริหารโครงการ (ต่อ)

4. ระยะปิดโครงการ (Project Closing)

เป็นระยะสุดท้ายของการบริหารโครงงาน เป็นการดำเนินงานหลังจากติดตั้งระบบซอฟต์แวร์เพื่อใช้งานเสร็จสิ้นแล้ว กล่าวคือ ระยะปิดโครงการจะดำเนินงานในช่วงการบำรุงรักษาระบบ (Maintenance) ของกระบวนการผลิตซอฟต์แวร์ การปิดโครงการมี 2 ลักษณะ คือ

- การปิดโครงการด้วยความสำเร็จ
- การปิดโครงการเนื่องจากความล้มเหลว



การบริหารโครงงาน

ความยากในการบริหารโครงการผลิตซอฟต์แวร์

1. ซอฟต์แวร์เป็นผลิตภัณฑ์ที่จับต้องไม่ได้

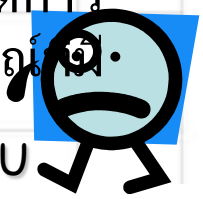
โครงการผลิตซอฟต์แวร์ ไม่เหมือนกับโครงการก่อสร้าง หรือผลิตรถยนต์ที่จับต้องได้ และเห็นอย่างชัดเจน ดังนั้นหากการดำเนินงานของโครงการไม่เป็นไปตามตารางเวลา ผู้บริหารโครงการจะไม่สามารถสังเกตเห็นผลกระทบที่จะเกิดขึ้นได้

2. กระบวนการผลิตซอฟต์แวร์ไม่มีมาตรฐานที่แน่นอน

กระบวนการผลิตซอฟต์แวร์มักถูกปรับให้เหมาะสมกับสภาพแวดล้อมขององค์กร จึงไม่สามารถใช้กระบวนการเดิมกับองค์กรอื่น ๆ ได้ ดังนั้นการบริหารโครงการจึงต้องมีกระบวนการไม่แน่นอนตามไปด้วย

3. โครงการผลิตซอฟต์แวร์ขนาดใหญ่ย่อมมีลักษณะพิเศษแตกต่างกัน

สืบเนื่องจากกระบวนการแตกต่าง การรับผิดชอบโครงการย่อยต้องพบกับปัญหาแตกต่างกันได้ ผู้บริหารโครงการจึงต้องใช้ประสบการณ์ในการบริหารจัดการ นอกจากนี้เทคโนโลยีต่าง ๆ ที่เปลี่ยนแปลงอย่างรวดเร็ว อาจทำให้ประสบการณ์ที่มาเกิดล้าสมัยได้



การบริหารโครงงาน

ความยากในการบริหารโครงการผลิตซอฟต์แวร์ (ต่อ)

4. ความต้องการในการผลิตซอฟต์แวร์เป็นวัตถุดิบที่ไม่สามารถจับต้องได้

นอกเหนือจากนั้น ความต้องการของลูกค้ามักเปลี่ยนแปลงอยู่เสมอ ดังนั้น ความยากที่เห็นชัดเจนอีกประการหนึ่ง คือ การจัดการกับความต้องการที่เปลี่ยนแปลงดังกล่าว

จาก 4 เหตุผลข้างต้น จึงมักพบว่าโครงการผลิตซอฟต์แวร์ส่วนใหญ่มักส่งงานล่าช้า ใช้งบประมาณเกินงบที่ประมาณการไว้ และทำงานช้ากว่ากำหนด

การบริหารโครงงาน

กิจกรรมในการบริหารโครงการ

1. การเขียนข้อเสนอโครงการ (Proposal Writing)
2. การวางแผนและจัดตารางงานโครงการ (Project Planning and Scheduling)
3. การประมาณการต้นทุนโครงการ (Cost Estimation)
4. การติดตามและแผนงานโครงการ (Project Monitoring and Review)
5. การคัดเลือกและประเมินบุคลากร
6. การเขียนและนำเสนอรายงาน

การบริหารโครงงาน

1. การเขียนข้อเสนอโครงการ (Proposal Writing)

สิ่งแรกที่ต้องทำคือ เขียนข้อเสนอโครงการ (Project Proposal) แล้วยื่นเสนอต่อเจ้าของโครงการ เพื่อขอรับการอนุมัติให้ดำเนินโครงการ การเขียนข้อเสนอโครงการจึงเป็นขั้นตอนที่สำคัญขั้นตอนหนึ่ง การประมูล หรือขออนุมัติจะสำเร็จลุล่วงหรือไม่ขึ้นอยู่กับข้อมูลในข้อเสนอ ซึ่งประกอบด้วยวัตถุประสงค์ (Objective) ของโครงการ การประมาณการ ต้นทุน และตารางงาน วิธีการดำเนินงาน และประโยชน์ที่จะได้รับจากโครงการ

2. การวางแผนและจัดตารางงานโครงการ (Project Planning and Scheduling)

การวางแผนโครงการ เป็นการกำหนดกิจกรรมหลัก กิจกรรมย่อย เป้าหมายของแต่ละกิจกรรม การส่งมอบงาน ส่วนการจัดตารางงาน เป็นกิจกรรมที่ผู้บริหารโครงการต้องเริ่มจากการนำกิจกรรมมาเป็นเป็นกิจกรรมย่อย และกำหนดระยะเวลาแล้วเสร็จให้กับแต่ละกิจกรรม

2. การวางแผนและจัดตารางงานโครงการ (ต่อ)

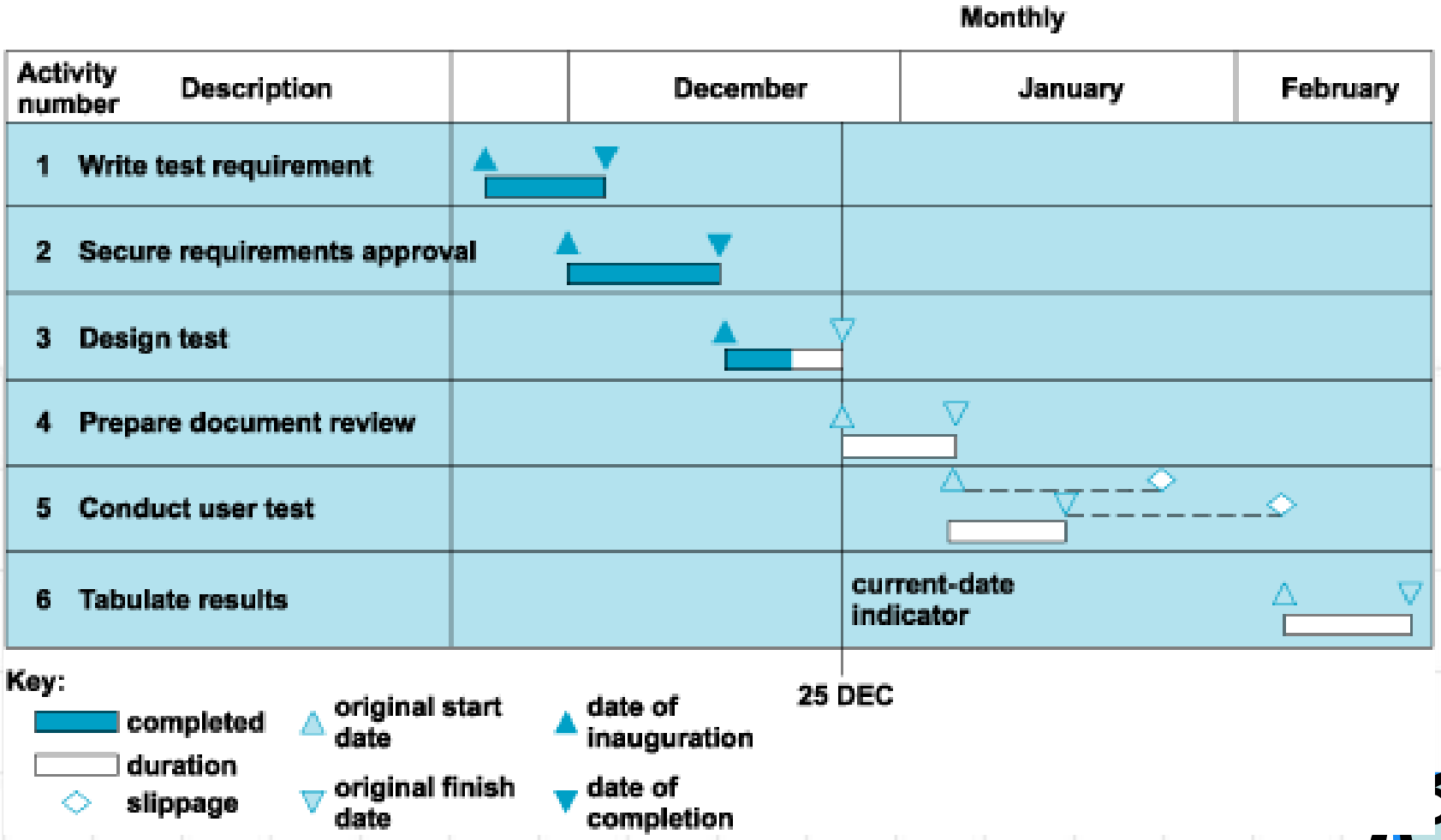
เทคนิคในการการจัดตารางงานโครงการ เพื่อช่วยให้สามารถจัดสรรทรัพยากรให้เหมาะสมกับกิจกรรมได้ง่ายขึ้น ได้แก่

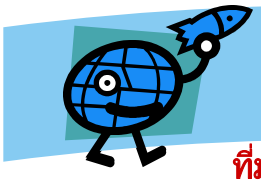
1) Gantt Chart

เกิดขึ้นโดย Henry L.Gantt ในปี 1917 ใช้จัดตารางการทำงานในโครงการด้วยกราฟแท่งในแนวนอน แสดงระยะเวลาของกิจกรรมแต่ละขั้นตอน รายชื่อกิจกรรมจะถูกแสดงไว้ในแนวตั้งด้านซ้าย ระยะเวลาการทำงานจะแสดงในแนวนอนของแผนภาพ อย่างไรก็ตามแผนภูมิลักษณะนี้ยังไม่สามารถบอกได้ว่างานที่ปฏิบัติการล่าช้าจะมีผลกระทบต่อโครงการอย่างไร ดังนั้นโครงการขนาดใหญ่จึงมักนำเทคนิคอื่นมาประยุกต์ใช้ด้วย เช่น PERT/CPM

ที่มา : www.answers.com/topic/gantt-chart

2. การวางแผนและจัดตารางงานโครงการ : Gantt Chart

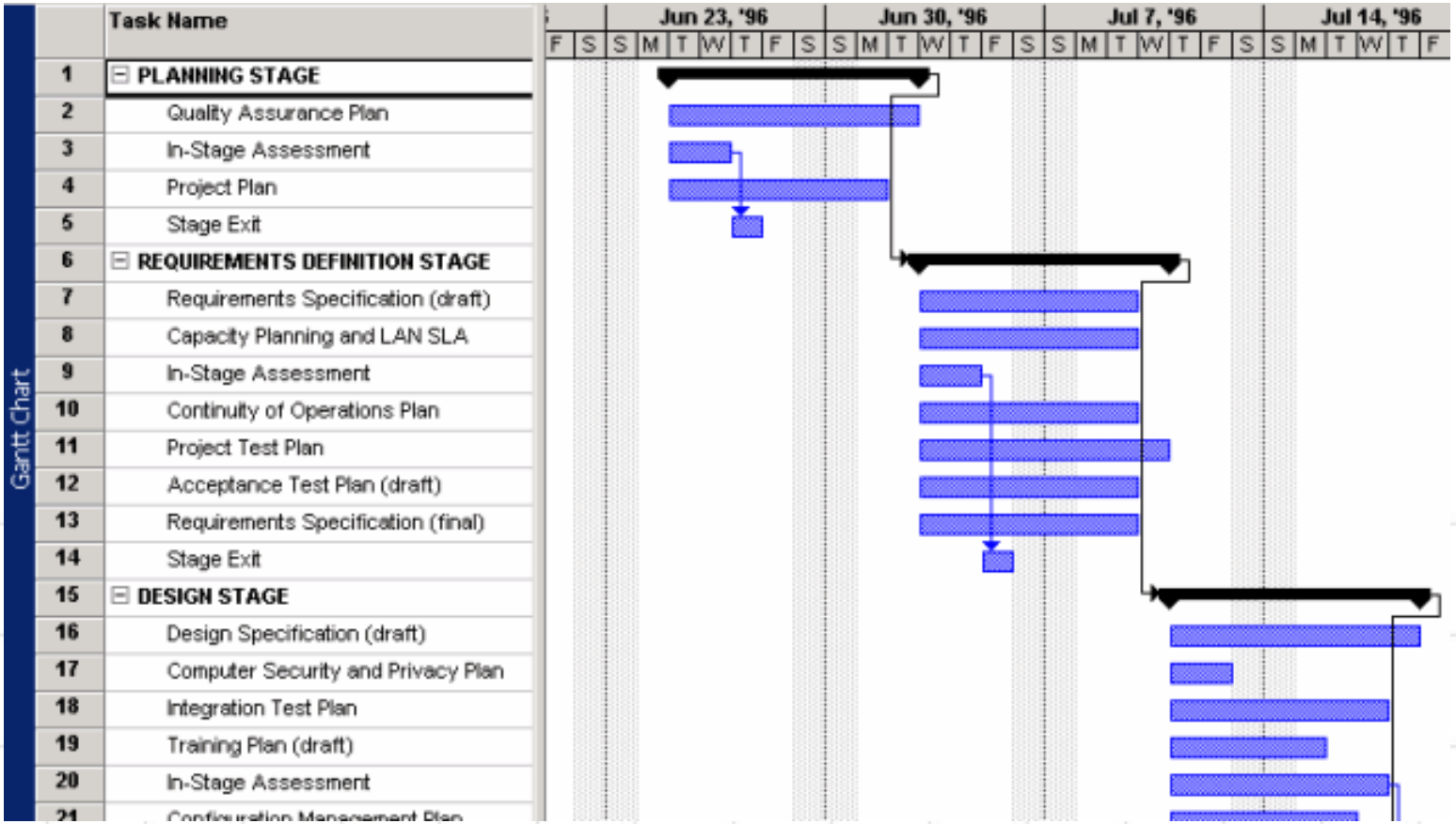




การบริหารโครงงาน

ที่มา : http://www.codinghorror.com/blog/2006/11/microsoft-project-and-the-gantt-waterfall.html

Gantt Chart (ตัวอย่าง)



https://www.smartsheet.com/c/simple-project-management?s=187&c=3&m=3103&a=27034057387&k=project%20software%20process&mtp=&adp=none&net=d&dev=c&devm=&plc=techforum4u.com&gclid=COjno7yGvroCFYLtpAodsm4AuA

การบริหารโครงงาน

2. การวางแผนและจัดตารางงานโครงการ (ต่อ)

2) PERT/CPM

PERT (Project Evaluation Review Technique) เป็นเทคนิคในการวิเคราะห์หรือประเมินเวลาที่ต้องใช้ในแต่ละกิจกรรมของโครงการ โดยแสดงเป็นแผนภาพกิจกรรมที่เชื่อมโยงกันในลักษณะของเครือข่าย
CPM (Critical Path Method) เป็นเทคนิคในการวิเคราะห์เส้นทางหรือกิจกรรมวิกฤติ โดยแสดงแผนภาพเครือข่ายของกิจกรรมเช่นกัน แต่ CPM จะแสดงด้วยสัญลักษณ์รูปวงกลม และเส้นลูกศร

อย่างไรก็ตาม PERT/CPM รวมเป็นเทคนิคเดียวกันเพราะมีวัตถุประสงค์เดียวกัน โดยมีจุดเด่น คือ การคำนวณหาเส้นทางวิกฤติในการดำเนินกิจกรรม ช่วยให้ผู้บริหารคำนวณได้หลายลักษณะ เช่น

- เวลาที่เร็วที่สุดของแต่ละกิจกรรม (Time Earliest : TE)
- เวลาที่ช้าที่สุดของแต่ละกิจกรรม (Time Latest : TL)
- ระยะเวลาเมื่อต้องการเร่งการทำงานของโครงการ
- ค่าใช้จ่ายและแรงงานเมื่อต้องการเร่งโครงการ

การบริหารโครงงาน

2. การวางแผนและจัดตารางงานโครงการ : PERT/CPM

ลำดับ	กิจกรรม	กิจกรรมก่อนหน้า
1	รวบรวมความต้องการ	-
2	ออกแบบรายงาน	1
3	ออกแบบหน้าจอ	1
4	ออกแบบฐานข้อมูล	2,3
5	จัดทำเอกสาร	4
6	เขียนโปรแกรม	4
7	ทดสอบโปรแกรม	6
8	ติดตั้งโปรแกรม	5,7

PERT : เน้นเวลาการดำเนินโครงการ
CPM : เน้นค่าใช้จ่ายในการดำเนินโครงการ

2. การวางแผนและจัดตารางงานโครงการ : PERT/CPM

สายงานวิกฤต (Critical Paths)
คือสายงานที่มีระยะเวลารวมยาวนานที่สุด ซึ่งโครงการหนึ่งอาจมีสายงานวิกฤตมากกว่าหนึ่งสายงานได้

การบริหารโครงงาน

ตัวอย่างตารางงาน

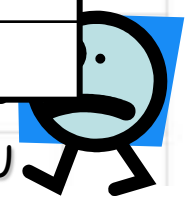
กิจกรรม	กิจกรรมที่ต้องเสร็จก่อน	เวลา (สัปดาห์)
A	-	2
B	-	1
C	-	1
D	A	3
E	B	3
F	C	2
G	D	3
H	F	2

การบริหารโครงงาน

การเร่งโครงการ

: สามารถทำได้ด้วยการเร่งกิจกรรมภายในสายงานวิกฤตเพื่อให้โครงการสำเร็จตามเวลาที่ได้กำหนดไว้ **[ถ้าต้องการให้โครงการเสร็จภายใน 28 วัน]**

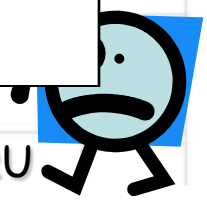
กิจกรรม	กิจกรรมที่ต้องเสร็จก่อน	ระยะเวลา (วัน)		ค่าใช้จ่ายในการเร่งกิจกรรมต่อ 1 วัน
		ปกติ	เร่ง	
A	-	7	6	150
B	-	8	6	75
C	A	9	7	200
D	A	11	9	125
E	B	8	5	115
F	B	10	7	100
G	C	13	11	200
H	D , E	13	12	100
I	F	14	10	125



การบริหารโครงงาน

การเร่งโครงการ

: สามารถทำได้ด้วยการเร่งกิจกรรมภายในสายงานวิกฤตเพื่อให้โครงการสำเร็จตามเวลาที่ได้กำหนดไว้ **[ถ้าต้องการให้โครงการเสร็จภายใน 28 วัน]**



การบริหารโครงงาน

3. การประมาณการต้นทุนโครงการ (Cost Estimation)

ข้อสังเกตในการประมาณการ

- การประมาณการเกี่ยวกับทรัพยากร ค่าใช้จ่าย และระยะเวลาดำเนินการ ต้องอาศัยประสบการณ์ ข้อมูลสารสนเทศ และความสามารถในการพยากรณ์ข้อมูลทั้งหมด
- การประมาณการเกี่ยวเนื่องกับความเสี่ยงอันเกิดจากการประมาณการ อาจนำไปสู่ความไม่แน่นอนทางด้านการวางแผนได้
- ปัจจัยที่มีผลกระทบต่อความไม่แน่นอน
  - ความซับซ้อนของโครงการ
  - ขนาดของโครงการ
  - ระดับความไม่แน่นอนทางด้านโครงสร้าง
- การประมาณการต้องกำหนดทั้งด้านดีและด้านไม่ดี

3. การประมาณการต้นทุนโครงการ (Cost Estimation)

ขอบเขตของซอฟต์แวร์

- ขอบเขตของซอฟต์แวร์ หมายถึงสิ่งต่างๆ ต่อไปนี้ ข้อมูล ขั้นตอนการควบคุมที่ต้องปฏิบัติ หน้าที่การทำงาน ประสิทธิภาพ ข้อจำกัด ส่วนที่ทำหน้า ที่เชื่อมต่อ และความเชื่อถือได้ของซอฟต์แวร์
- หน้าที่การทำงานที่ระบุในข้อตกลงนั้นต้องผ่านการประเมิน และกลั่นกรอง เพื่อให้ได้รายละเอียดก่อนที่จะเริ่มต้นการประมาณการ
- หน้าที่การทำงานจะอธิบายในรูปการกำหนดปัญหาโดยมีการกลั่นกรองเพื่อให้ได้รายละเอียดก่อนที่จะเริ่มต้นการประมาณการ
- ประสิทธิภาพของซอฟต์แวร์จะกล่าวในรูปของความต้องการของเวลาที่ระบบตอบสนอง
- ส่วนของข้อจำกัดนั้น ระบุปัจจัยที่มีผลต่อซอฟต์แวร์ทั้งจากฮาร์ดแวร์ภายนอก จากขนาดหน่วยความจำที่มีใช้งาน และจากระบบอื่นๆ

การบริหารโครงงาน

3. การประมาณการต้นทุนโครงการ (Cost Estimation)

ค่าใช้จ่ายที่เกิดขึ้นในการพัฒนา แบ่งเป็น 5 กลุ่ม คือ

(1) Computer-related : ค่าใช้จ่ายของฮาร์ดแวร์ และซอฟต์แวร์ที่จัดซื้อ/จัดหาในการพัฒนาโครงการ

(2) Facilities : ค่าใช้จ่ายทางด้านสาธารณูปโภค ความสะดวกสบายในการพัฒนาโครงงาน

(3) People : ค่าใช้จ่ายที่ต้องจ่ายให้กับบุคลากรในทีมงาน โดยประมาณการจากค่าความพยายามหรือกำลังคนที่ต้องการใช้ (effort)

หมายเหตุ : วิธีนี้มีความคลาดเคลื่อนสูง เพราะกิจกรรมมักขึ้นอยู่กับปัจจัยอื่น

(4) Project complexity : ค่าใช้จ่ายโดยคิดจากความซับซ้อนในการพัฒนา คำนวณจากปริมาณของเอกสารและจำนวนระบบย่อยที่เกี่ยวพันกับโครงการ

(5) Project methods and tools : ค่าใช้จ่ายในด้านมาตรฐาน, ขั้นตอนการดำเนินกิจกรรม เอกสาร การเปลี่ยนแปลง รวมทั้งวิธีการอบรม ฝึกสอน เป็นต้น (ค่าใช้จ่ายที่แฝงอยู่ในการพัฒนา)

การบริหารโครงงาน

3. Cost Estimation

บางโครงการอาจต้องใช้ค่าที่ได้จากการวัดประสิทธิภาพในการทำงาน (Productivity) ซึ่งหมายถึงผลที่ได้จากการผลิตงานของบุคลากรในโครงการไปคำนวณหาต้นทุนหรือจัดตารางงาน ซึ่ง Productivity วัดได้จากจำนวนหน่วยของงานที่ผลิตได้ (Size) หารด้วยจำนวนเวลาที่ต้องการใช้ในการผลิต (Effort) อาจมีหน่วยเป็น Person-Hours หรือ Man-Day หรือ Man-Month ก็ได้

Productivity = Size / Effort

หมายเหตุ : ฮาร์ดแวร์วัดจากจำนวนหน่วยของงานจะเป็นรูปธรรมคือจำนวนสินค้าที่ผลิตได้ แต่ซอฟต์แวร์วัดจากขนาดของซอฟต์แวร์

วิธีในการวัดขนาดของซอฟต์แวร์ ยกตัวอย่าง 2 ประเภท คือ

1. วัดจากจำนวนบรรทัดของซอร์สโค้ด (Line of Code : LOC)
2. วัดจากจำนวนฟังก์ชันที่ใช้งานได้ (Function Point : FP)

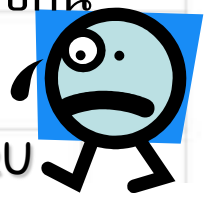
การบริหารโครงงาน

1.

3. Line of Code

วิธีแรกที่นำมาใช้ในยุคก่อน สำหรับการโปรแกรมที่มีลักษณะลำดับเป็นบรรทัดอย่างต่อเนื่องกันไป เช่น COBAL, ASSEMBLY, FORTRAN เป็นต้น การนับจำนวนบรรทัดจึงเป็นวิธีวัดขนาดซอฟต์แวร์ที่ง่ายและชัดเจน โดยเป็นแบ่งหลายวิธี

- **Simple Line Count**
  นับทุกบรรทัดที่อยู่ใน Source file โดยนับรวมบรรทัดว่างหรือ comment โปรแกรมด้วย
- **Physical Lines (LINES)**
  นับจำนวนบรรทัดของโค้ดที่ยังคงใช้กันอยู่ในการบริหารโครงการผลิตซอฟต์แวร์ โดยนับโค้ดทุกบรรทัด (ยกเว้นบรรทัดที่นิยามตัวแปร)
- **Physical Lines of Code**
  นับจำนวนบรรทัด แต่ไม่นับรวมบรรทัดว่างและบรรทัดที่เป็น comment บางครั้งจึงเรียกวิธีนี้ว่า Source Line Code : SLOC)
- **Logical Lines of Code (LLOC)**
  มีวิธีการนับคล้าย Physical แตกต่างกันคือ จะนับบรรทัดที่มีการเชื่อมต่อกันด้วยอักขระ _ รวมเป็นบรรทัดเดียว



การบริหารโครงงาน

3. (ต่อ) : Line of Code

- **Statements (STMT)**
  จะไม่นับจำนวนบรรทัดแต่จะนับจำนวนประโยคคำสั่ง โดยบางภาษาใช้ ; เป็นตัวระบุจบประโยคคำสั่ง

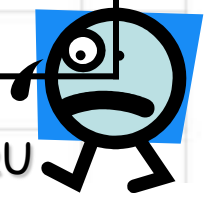
ข้อเสียของการประมาณการขนาดของซอฟต์แวร์ด้วยการนับ คือ
จำนวนบรรทัดที่นับได้ ขึ้นอยู่กับภาษาโปรแกรมที่ใช้และคุณภาพในการออกแบบโปรแกรม หากโปรแกรมแตกต่างกันจะไม่สามารถเปรียบเทียบกันได้ บางโปรแกรมมีฟังก์ชันสำเร็จในการใช้งาน ทำให้ไม่ต้องสร้างฟังก์ชันขึ้นมาใช้เอง ดังนั้นค่าที่ได้จากการนับจำนวนบรรทัด จึงไม่สามารถนำมาใช้ประมาณการขนาดของซอฟต์แวร์ได้อย่างสมบรณ์

การบริหารโครงงาน

3. (ต่อ) : Line of Code

C Programming Language	COBOL Programming Language

#include <stdio.h>
#include <conio.h>
void main()
{
printf(“Hello World”);
getch();
}
	100 IDENTIFICATION DIVISION.
	200 PROGRAM-ID. EXAMHELLO.
	300 ENVIRONMENT DIVISION.
	400 CONFIGURATION SECTION.
	500 SOURCECOMPUTER. KTP.
	600 OBJECT-COMPUTER. KTP.
	700 DATA DIVISION.
	800 FILE SECTION.
	900 PROCEDURE DIVISION.
	1000 MAIN-DISPLAY SECTION.
	2000 BEGIN
	3000 DISPLAY “Hello World” LINE 10
	POSITION 10
	4000 STOP RUN.



การบริหารโครงงาน

3. (ต่อ) : Line of Code

ภาษา	วิเคราะห์	ออกแบบ	พัฒนา	ทดสอบ	เอกสาร
Assembly	4 สัปดาห์	6 สัปดาห์	10 สัปดาห์	12 สัปดาห์	2 สัปดาห์
ภาษาระดับสูง	4 สัปดาห์	6 สัปดาห์	5 สัปดาห์	7 สัปดาห์	2 สัปดาห์

ภาษา	Size	Effort	Productivity
Assembly	6,000 บรรทัด	34 สัปดาห์	705 บรรทัด/เดือน
ภาษาระดับสูง	2,500 บรรทัด	24 สัปดาห์	416 บรรทัด/เดือน

หมายเหตุ : Effort ต้องหารด้วย 4 ก่อน เพื่อทำให้หน่วยเป็นเดือน

2.

3. Function Point : FP

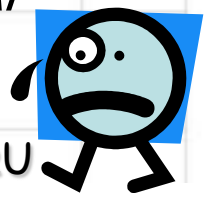
เป็นการวัดขนาดของซอฟต์แวร์ด้วยการนับจำนวนฟังก์ชันการทำงานของโปรแกรมจากข้อกำหนดความต้องการของซอฟต์แวร์ และลดปัญหาด้านความแตกต่างของโปรแกรมมิ่งที่ใช้ และความแตกต่างด้านเทคโนโลยีด้วย โดยวิธีนี้ได้รับการปรับปรุงโดยคณะกรรมการการกำหนดและรวบรวมกรรมวิธีในการวัดขนาดซอฟต์แวร์ด้วยฟังก์ชัพอยท์ (International Function Point User Group : IFPUG) โดยมีสูตรดังนี้

$$FP = UFP * VAF$$

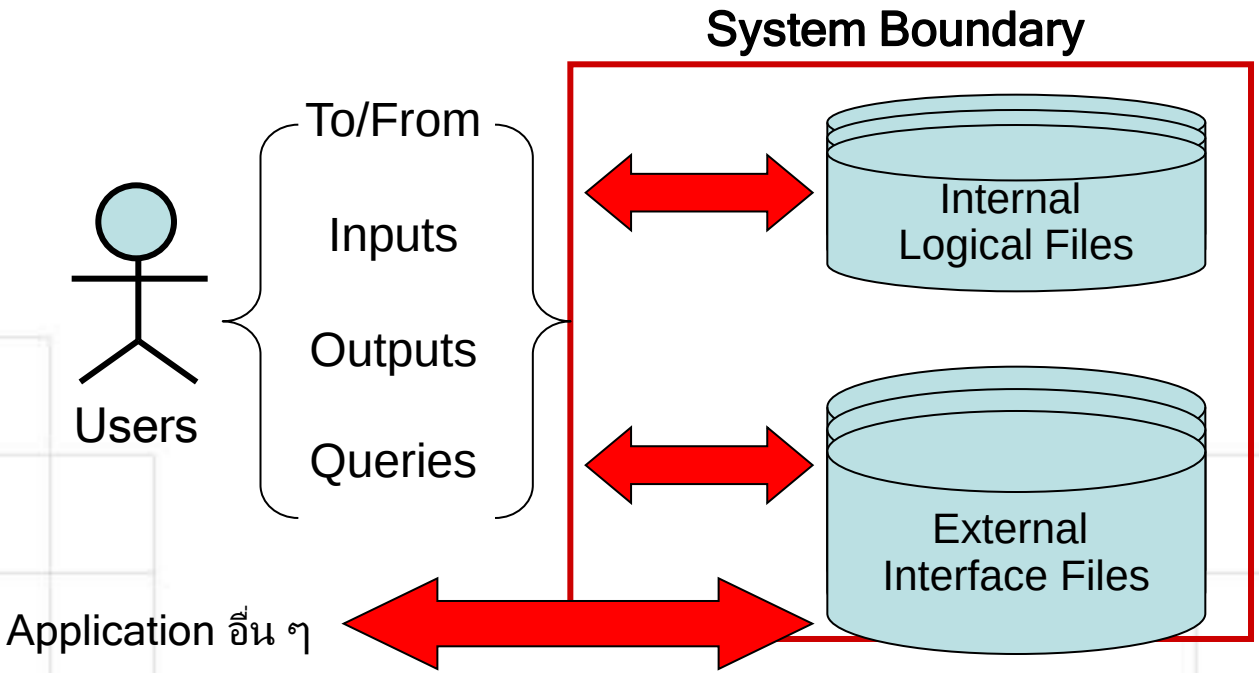
โดย

UFP คือ ฟังก์ชันพอยท์ที่ยังไม่ได้ปรับแต่ง (Unadjusted Function Point)
VAF คือ ค่าปัจจัยคุณลักษณะของระบบ (Value Adjustment Factor)

หมายเหตุ : ข้อมูลบางแหล่งอาจย่อ Unadjucted Function Point : UAF



3. (ต่อ) : Function Point



คุณลักษณะ 14 ประการของระบบ

(14's General System Characteristic)

Size (Function Point) = UFP x VAF

3. (ต่อ) : Function Point

1. คำนวณหา FP ที่ยังไม่ได้ปรับแต่ง (UFP)

โดยแบ่งประเภทฟังก์ชันการทำงานที่ต้องนับเป็น 5 ประเภท ได้แก่

- **External Inputs (EI)**

  ข้อมูลที่รับเข้ามาในระบบ (อาจเป็นข้อมูลทางธุรกิจหรือข้อมูลควบคุม) เพื่อนำไปอัพเดทข้อมูลใน ILF เช่น ข้อมูลในกระบวนการ เพิ่ม ลบ แก้ไขข้อมูล เป็นต้น

- **External Outputs (EO)**

  ข้อมูลที่เป็นผลลัพธ์จากการประมวลผลข้อมูลที่ได้รับจากภายในระบบ ให้นับการแสดงผลข้อมูลที่มีรูปแบบแตกต่างกัน

- **External Queries (EQ)**

  กระบวนการดึงข้อมูลและประมวลผลเพื่อแสดงผลต่อผู้ใช้ (คือ Query ข้อมูลนั่นเอง)

3. (ต่อ) : Function Point

1. คำนวณหา FP ที่ยังไม่ได้ปรับแต่ง (ต่อ)

- **Internal Logical Files (ILF)**

  ไฟล์ที่เกี่ยวข้องกับข้อมูลที่อยู่ในระบบตลอดช่วงอายุของระบบ และเป็นไฟล์มักจะถูกบำรุงรักษาหรือปรับปรุงด้วยข้อมูลที่ได้รับจากภายนอก (EI) ให้นับรวมเรคคอร์ดที่ทำหน้าที่เทียบเท่ากับไฟล์ด้วย

- **External Interface Files (EIF)**

  ไฟล์ที่เกี่ยวข้องกับข้อมูลที่ใช้เพื่อการอ้างอิงเท่านั้น และใช้ร่วมกับระบบอื่น ๆ EIF เป็นไฟล์ที่ถูกเรียกใช้โดยระบบจะพัฒนา แต่จะบำรุงรักษาหรือถูกสร้างโดยระบบอื่น

ฟังก์ชันแต่ละประเภทเกิดจากการทำรายการข้อมูลของผู้ใช้ จึงมีความซับซ้อนแตกต่างกันตามจำนวนข้อมูล เรคคอร์ด และไฟล์ที่เกี่ยวข้อง จากนั้นจึงนำมาเทียบกับตารางเกณฑ์ระดับความซับซ้อนของฟังก์ชั่น (ต่ำ ปานกลาง สูง) แล้วนำมาคูณกับตัวถ่วงน้ำหนัก แล้วหาผลรวมฟังก์ชันทั้งหมดที่นับได้

3. (ต่อ) : Function Point

1. คำนวณหา FP ที่ยังไม่ได้ปรับแต่ง (ตารางค่าน้ำหนักความซับซ้อน)

ประเภทของฟังก์ชันพอยต์	ค่าน้ำหนักความซับซ้อน		
	น้อย	ปานกลาง	มาก
ข้อมูลเข้าจากภายนอก (External Input)	3	4	6
ข้อมูลที่ส่งออกสู่ภายนอก (External Output)	4	5	7
ข้อมูลที่ดึงมาจากภายนอก (External Inquiries)	3	4	6
ข้อมูลที่ต้องการจากภายนอก (External Interface Files)	5	7	10
ข้อมูลเชิงตรรกะภายใน (Internal Logical Files)	7	10	15
	Total Unadjusted Function Points		

3. (ต่อ) : Function Point

คำนวณหา FP ที่ยังไม่ได้ปรับแต่ง (ตัวอย่าง)

หลังจากที่นับจำนวนฟังก์ชันของระบบสารสนเทศ จะต้องนำไปคูณกับค่าน้ำหนักของแต่ละหมวดหมู่ฟังก์ชันซึ่งจะไม่เท่ากัน โดยการคิดจะนำจำนวนของฟังก์ชันที่นับได้ในแต่ละหมวดหมู่ คูณกับค่าน้ำหนักของหมวดหมู่นั้นๆ โดยรายละเอียดแล้วน้ำหนักที่คูณอาจมีการคำนึงถึงความซับซ้อน อีกด้วย

หมวดหมู่	น้ำหนักของหมวดหมู่	จำนวนฟังก์ชัน	ผลคูณ
EIs	4	3	12
EOs	5	2	10
EQs	4	1	4
ILFs	10	1	10
EIFs	7	1	7
ค่าฟังก์ชันพอยต์ที่ยังไม่ได้ปรับค่า			**43**

3. (ต่อ) : Function Point

2. คำนวณค่าปัจจัยคุณลักษณะของระบบ (VAF)

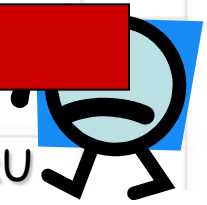
ก่อนคำนวณหาค่า FP สุดท้าย ให้คำนวณค่าปัจจัยที่ส่งผลต่อความแตกต่างกันของระบบ ปัจจัยดังกล่าว คือ คุณลักษณะเด่นของระบบทั้งหมด 14 ข้อ ตามข้อกำหนดความต้องการของลูกค้า โดยกำหนดค่าตั้งแต่ 0 (ไม่เกี่ยวข้อง) จนถึง 5 (เกี่ยวข้องมาก)

ลำดับ	คุณลักษณะ	ค่าระดับ
1	การติดต่อสื่อสาร (Data Communication)	
2	การประมวลผลข้อมูลแบบกระจาย (Distributed Data Processing)	
3	ประสิทธิภาพของระบบ (Performance)	
4	การแก้ไขค่าของระบบ (Configuration)	
5	ปริมาณรายการข้อมูล (Transaction)	

การบริหารโครงงาน

ลำดับ	คุณลักษณะ	ค่าระดับ
6	การป้อนข้อมูลเข้าสู่ระบบแบบออนไลน์ (Online Data Entry)	
7	ประสิทธิภาพการใช้งานของผู้ใช้ (End-user Efficiency)	
8	การปรับปรุงข้อมูลแบบออนไลน์ (Online Update)	
9	ความซับซ้อนของการประมวลผล (Complex Processing)	
10	การนำไปใช้ซ้ำได้ (Reusability)	
11	ความง่ายในการดำเนินงาน (Operational Ease)	
12	ความง่ายในการติดตั้ง (Installation Ease)	
13	การใช้งานได้หลายไซด์ (Multiple Sites)	
14	รองรับการเปลี่ยนแปลงความต้องการของผู้ใช้ (Change Requirement)	

VAF = 0.65 + [0.01 * ผลรวมค่าคุณลักษณะ 14 ด้าน]



3. (ต่อ) : Function Point

2. คำนวณค่าปัจจัยคุณลักษณะของระบบ (ตัวอย่าง)

ยกตัวอย่างเช่นหากประเมินแล้วแต่ละข้ออยู่ที่ 4 ดังนั้นจะมีผลรวมในการประเมิน (Sum of Score: SS) เท่ากับ **4x14 = 56** ดังนั้น

$$VAF = 0.65 + [\ 0.01 * 56\ ]$$
$$= 1.21$$

3. คำนวณค่า FP ที่ปรับแต่งแล้ว

เมื่อคำนวณหา UFP และ VAF แล้ว นำมาคูณกัน จะได้ผลลัพธ์เป็นค่า FP ที่ปรับแต่งแล้วตามคุณลักษณะเด่นของระบบ ตามสูตร FP = UFP * VAF

Size (Function Point) = UFP x VAF
= 43 x 1.21
= 52.03
= 53

3. (ต่อ) : Function Point

ทั้ง LoC และ FP คือ วิธีวัดขนาดของซอฟต์แวร์ โดยสามารถนำไปคำนวณหา Productivity และ Effort ต่อไปได้ อย่างไรก็ตาม สูตรคำนวณเพื่อประมาณการ Effort นั้น บางครั้งอาจต้องการใช้ขนาดของซอฟต์แวร์ที่เป็น LoC ดังนั้น ค่า FP ที่หาได้อาจต้องแปลงไปเป็น LoC ซึ่งมีตารางเปรียบเทียบตามมาตรฐานของ QSM (Quantitative Software Management: www.qsm.com)

Language	SLOC/FP
Access	38
C	128
C++	55
HTML 3.0	15
Java	53
Perl	27
Visual C++	34

(From Capers Jones, 1996)

จากตัวอย่าง ถ้าหากจัดทำซอฟต์แวร์จะได้ LOC โดยใช้

ภาษาจาวา = 53 x 53 = 2,809 SLOC

ภาษาซี = 53 x 128 = 6,784 SLOC

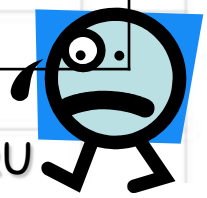
3. (ต่อ) : เทคนิคการประมาณต้นทุนและ Effort

การประมาณต้นทุน และ Effort ที่ดีจะต้องใกล้เคียงกับความเป็นจริง แต่การประมาณการให้ใกล้เคียงกับค่าที่เป็นจริงทำได้ยาก ในปี 1981 Boehm ได้กำหนดเทคนิคการประมาณการต้นทุน 7 แบบดังนี้

เทคนิค	รายละเอียด
Algorithmic Cost Modeling 1	รูปแบบนี้ประเมินราคาโดยการสร้างตัวแบบคณิตศาสตร์ โดยใช้ข้อมูลที่เกี่ยวข้องกับการพัฒนาในอดีตมานิยามค่าคงที่ต่าง ๆ ให้กับสมการตัวแบบที่นิยมใช้คือ COCOMO ซึ่งเป็นการประมาณการค่าใช้จ่ายการพัฒนา Software โดยพิจารณาจากจำนวนบรรทัด ของโปรแกรมหรือจำนวน Function point
Expert Judgement 2	วิธีนี้ให้ผู้บริหารโครงการที่มีประสบการณ์หลายคนมาเป็นผู้ประเมิน โดยต่างคนต่างประเมินค่าของตนเอง จากนั้นนำข้อมูลต่าง ๆ มาวิเคราะห์ร่วมกันเพื่อหาข้อสรุปของค่าที่เหมาะสม

3. (ต่อ) : เทคนิคการประมาณต้นทุนและ Effort

เทคนิค	รายละเอียด
Estimation by Analogy 3	วิธีนี้ใช้ประเมินกับโครงการซอฟต์แวร์ที่มีลักษณะคล้ายคลึงกัน (โครงงานในอดีตที่พัฒนาเสร็จสมบูรณ์แล้ว) ใช้ได้ดีกับโครงการขนาดใหญ่ที่มีวิธีการดำเนินงานที่คล้าย ๆ กัน แต่ไม่เหมาะกับโครงการขนาดเล็กที่มีลักษณะเฉพาะ
Parkinson's Law 4	การประเมินค่าใช้จ่ายวิธีนี้ยึดกฎของ Parkinson ที่อธิบายว่าปริมาณงานจะขยายตัวไปได้เรื่อยจนกระทั่งครบตามเวลาที่กำหนดไว้ เป็นวิธีการที่เน้นพิจารณารายละเอียดของทรัพยากรที่มีอยู่ (คน,เวลา) มากกว่าการประเมินจากจุดมุ่งหมายของโครงการ
Pricing to Win 5	ประเมินค่าใช้จ่ายจากความสามารถในการชำระเงินหรืองบประมาณของลูกค้า โดยไม่คำนึงคุณภาพของงานที่ต้องมีในซอฟต์แวร์ และความต้องการของลูกค้า



การบริหารโครงงาน

3. (ต่อ) : เทคนิคการประมาณต้นทุนและ Effort

เทคนิค	รายละเอียด
Top-down Estimation 6	การประเมินวิธีนี้ เริ่มวิเคราะห์จากค่าใช้จ่ายทั้งหมดของโครงการก่อนเป็นหลัก จากนั้นจึงเริ่มพิจารณาองค์ประกอบย่อยต่างๆของระบบ ข้อดีคือเน้นที่ความสำคัญของวัตถุประสงค์
Bottom-Up Estimation 7	ประเมินจากค่าใช้จ่ายของแต่ละองค์ประกอบในโครงการก่อน จากนั้นนำค่าใช้จ่ายต่าง ๆ มาสรุปร่วมกันเป็นค่าใช้จ่ายสุดท้าย

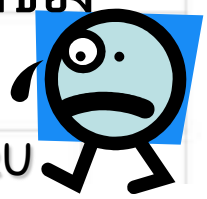
3. (ต่อ) : เทคนิคการประมาณการแบบ COCOMO

COCOMO (COnstructive COst MOdel) เป็นแบบจำลองประมาณการ Effort ต้นทุน และการจัดตารางการทำงานที่คิดค้นขึ้นโดย Dr.Barry Boehm ในปีค.ศ.1981 โดยการพิจารณาจากขนาดของซอฟต์แวร์และคุณลักษณะของซอฟต์แวร์ที่ผู้ใช้ต้องการ[1]

COCOMO เป็นการคำนวณจากขนาดของซอฟต์แวร์ ร่วมกับปัจจัยแวดล้อมอื่น ๆ ที่เกี่ยวข้อง เช่น

- ความแน่นอนของกระบวนการ และความต้องการไม่เปลี่ยนมากนัก
- ความสามารถในการผลิตซอฟต์แวร์ของทีมงาน มีความบริหารงานได้ดี
- มักใช้กับโครงการที่มีขนาด Delivered Source Instruction ไม่ต่ำกว่า 2,000 DSI
- ความยืดหยุ่น ความเสี่ยง และวิธีจัดการความเสี่ยง เป็นต้น

นอกจากนั้นยังใช้การคำนวณแบบ Exponential เพราะพบว่าขนาดของซอฟต์แวร์มีความสัมพันธ์กันแบบไม่เป็นเส้นตรง



การบริหารโครงงาน

3. (ต่อ) : เทคนิคการประมาณการแบบ COCOMO

Effort แปรผันตาม :

(1) ขนาดซอฟต์แวร์แบบยกกำลัง : หากเพิ่มขนาดของซอฟต์แวร์ จะทำให้
- จำนวนบุคลากรในทีมงานเพิ่มขึ้น
- ค่าใช้จ่ายอื่น ๆ ในระบบเพิ่มขึ้น อาทิ ด้านการสื่อสาร ด้านการบริหาร ด้านการรวมระบบ ซึ่งเรียกค่าใช้จ่ายเหล่านี้ ว่า Overhead

(2) ปัจจัยแวดล้อมอื่นๆ ที่ส่งผลกระทบได้แก่
- คุณลักษณะของซอฟต์แวร์
- คุณลักษณะของ Platform
- คุณลักษณะของทีมงาน และ
- คุณลักษณะของการบริหารโครงการ

ให้น้ำหนักกับคุณลักษณะดังกล่าวเพื่อปรับค่าจำนวน Effort ที่เหมาะสมที่สุด

3. : BASIC COCOMO

สูตรที่ใช้ในการคำนวณ Organic

Effort -> $PM = 2.4(KSDI)^{1.05}$

Schedule -> $TD = 2.5(PM)^{0.38}$

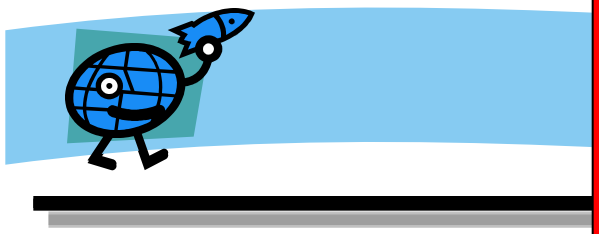
โดยที่

PM = Person-month

KDSI = Delivered source instruction, in thousands

TD = Number of months estimated for software development

สำหรับ PM ประกอบ 19 Day (152 hours) is working time (1 day : 8 hours)



PM = ระบุค่าตามสูตร โดย Line of code แปลงเป็น Kilo Line of code (128)
= 2.4(128)^1.05

Productivity = คำนวณจากการนำค่า Line of code / PM
= 128000/392

Schedule = ระบุค่าตามสูตร
= 2.5(392)^0.38

Staffing = คำนวณจากการนำค่า PM/Schedule
= 392/24

3. : BASIC COCO

A project is org
128,000 lines of code.

Effort : PM = 392 person-months
Productivity = 327 DSI/PM
Schedule TD = 24 months
Avg. Staffing = 16 FSP

Note : FSP = Full-time equivalent staff personnel

การบริหารโครงงาน

4. การติดตามและทบทวนโครงการ (Project Monitoring and Review)

ผู้บริหารโครงการจะติดตามความคืบหน้าของงาน พร้อมกับพิจารณาระยะเวลาและต้นทุนที่ใช้จริงเปรียบเทียบกับที่วางแผนไว้ ทบทวนการทำงานและปัญหาที่เกิดขึ้น และร่วมกันแก้ปัญหาดังกล่าว นอกจากนี้ ผู้บริหารโครงการยังต้องทบทวนผลที่ได้จากการทำงานและวัตถุประสงค์ขององค์กรด้วยว่ายังคงเหมือนเดิมหรือต้องมีการแก้ไข

5. การคัดเลือกและประเมินบุคลากร

โดยทั่วไป ผู้บริหารโครงการจะต้องคัดเลือกบุคลากรเพื่อเข้าไปทำหน้าที่ต่าง ๆ ในโครงการตามกิจกรรมที่กำหนดไว้ ซึ่งต้องเป็นบุคลากรที่มีความสามารถมากพอที่จะรับผิดชอบงานที่ได้รับมอบหมาย อย่างไรก็ตาม การจัดสรรบุคลากรย่อมขึ้นอยู่กับจำนวนงบประมาณที่ได้รับและข้อจำกัดด้านปริมาณบุคลากรที่มีอยู่ด้วย

6. การเขียนและนำเสนอรายงาน

การเขียนและนำเสนอรายงานเป็นอีกหน้าที่หนึ่งของผู้บริหารโครงการ เพื่อนำเสนอต่อลูกค้าและผู้รับเหมาช่วย (ถ้ามี) การเขียนรายงานจะต้องใช้ข้อความที่กระชับ เข้าใจง่าย และต้องสอดคล้องกัน นอกจากนี้ ผู้บริหารโครงการยังต้องนำเสนอรายงานต่อที่ประชุมเพื่อการติดตามความคืบหน้าของงานและการทบทวนเป้าหมายขององค์กร