บทที่ 4
การบริหารโครงการซอฟต์แวร์และการประมาณการต้นทุนซอฟต์แวร์
(Software Project Management and Cost Estimation)

วิชา วิศวกรรมซอฟต์แวร์ (04-06-322)

วัตถุประสงค์การเรียนรู้

- เพื่อให้ผู้เรียนมีความรู้ความเข้าใจเกี่ยวกับการบริหารโครงการ และกิจกรรมการผลิตซอฟต์แวร์ ตลอดจนการวางแผน และการจัดตารางงานได้อย่างเหมาะสม
- เพื่อให้ผู้เรียนสามารถรู้และเข้าใจหลักการประมาณการต้นทุนโครงการเบื้องต้นได้

หัวข้อ (Agenda)

- บทนำ (Overview)
- การบริหารโครงการผลิตซอฟต์แวร์
  - องค์ประกอบของการบริหารโครงการ
  - องค์ความรู้ของการบริหารโครงการ
  - วงจรการพัฒนาโครงการ
  - กิจกรรมในการบริหารโครงการ
- การประมาณการต้นทุนซอฟต์แวร์
  - ปัจจัยของการประมาณการราคา
  - การวัดขนาดของซอฟต์แวร์
    - Line of Code (LOC)
    - Function Point (FP)
- สรุป (Summary)

บทนำ (Overview)

วิศวกรรมซอฟต์แวร์

(Software Engineering)

การบริหารโครงการผลิตซอฟต์แวร์

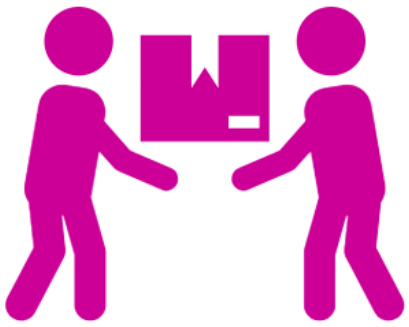
(Software Project Management)

นักบริหารโครงการ

(Project Manager)

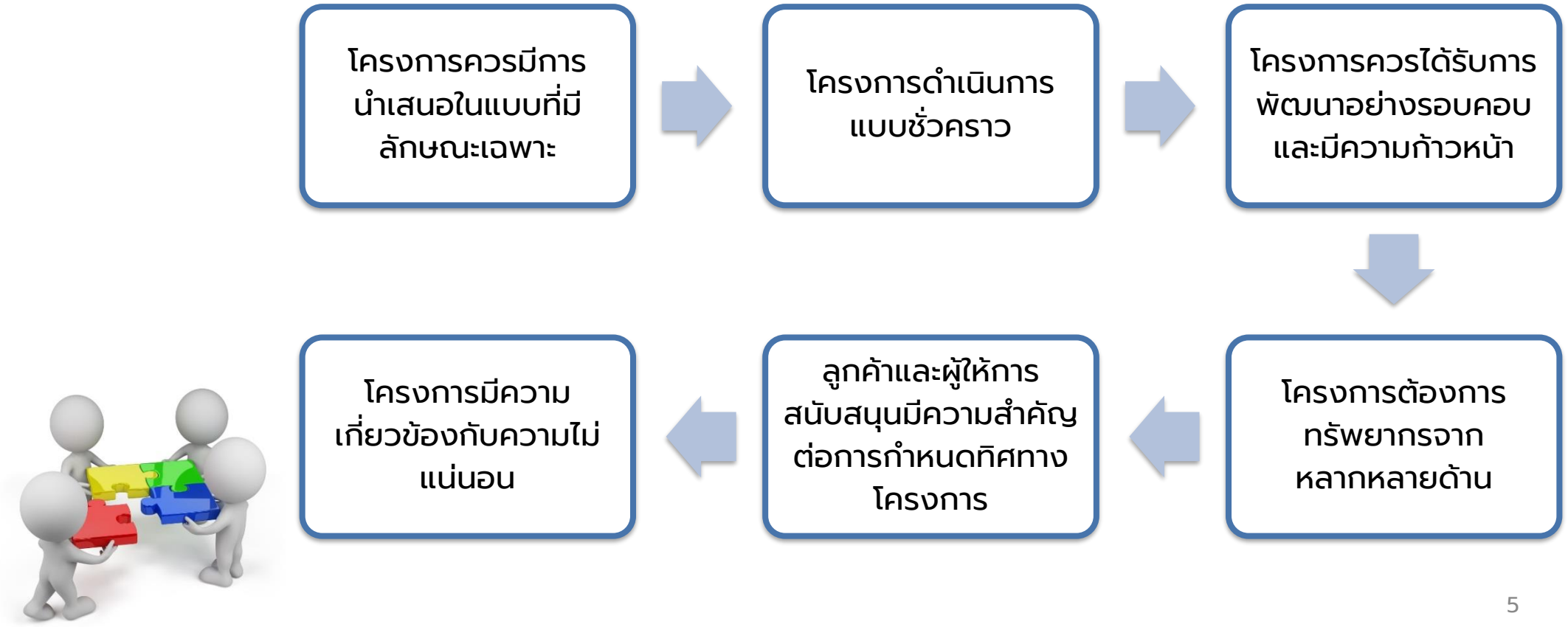
การดำเนินการดังกล่าวมีเป้าหมาย:

1. การส่งมอบซอฟต์แวร์ให้กับลูกค้าภายในระยะเวลาที่กำหนด
2. ควบคุมการดำเนินงานภายใต้งบประมาณที่กำหนด
3. ซอฟต์แวร์ที่ส่งมอบต้องตรงกับที่ผู้ใช้งานต้องการ/คาดหวัง
4. ทีมงานพัฒนามีความคิดเชิงบวกต่อการบำรุงรักษาและการพัฒนาฟังก์ชันต่าง ๆ



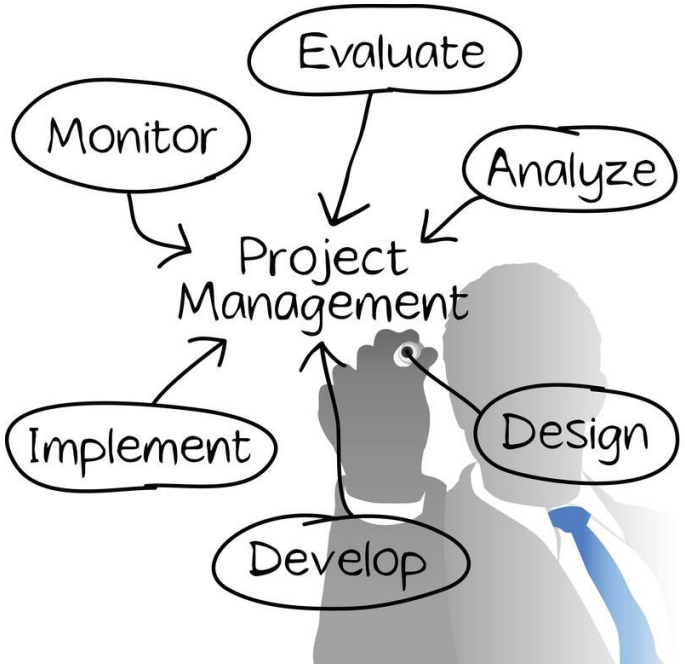
โครงการ (Project)

- การดำเนินกิจกรรมตามแผนที่จัดทำขึ้น เพื่อบรรลุเป้าหมายหรือวัตถุประสงค์ที่กำหนดไว้ภายใต้ Scope, Schedule, Cost และ Resource

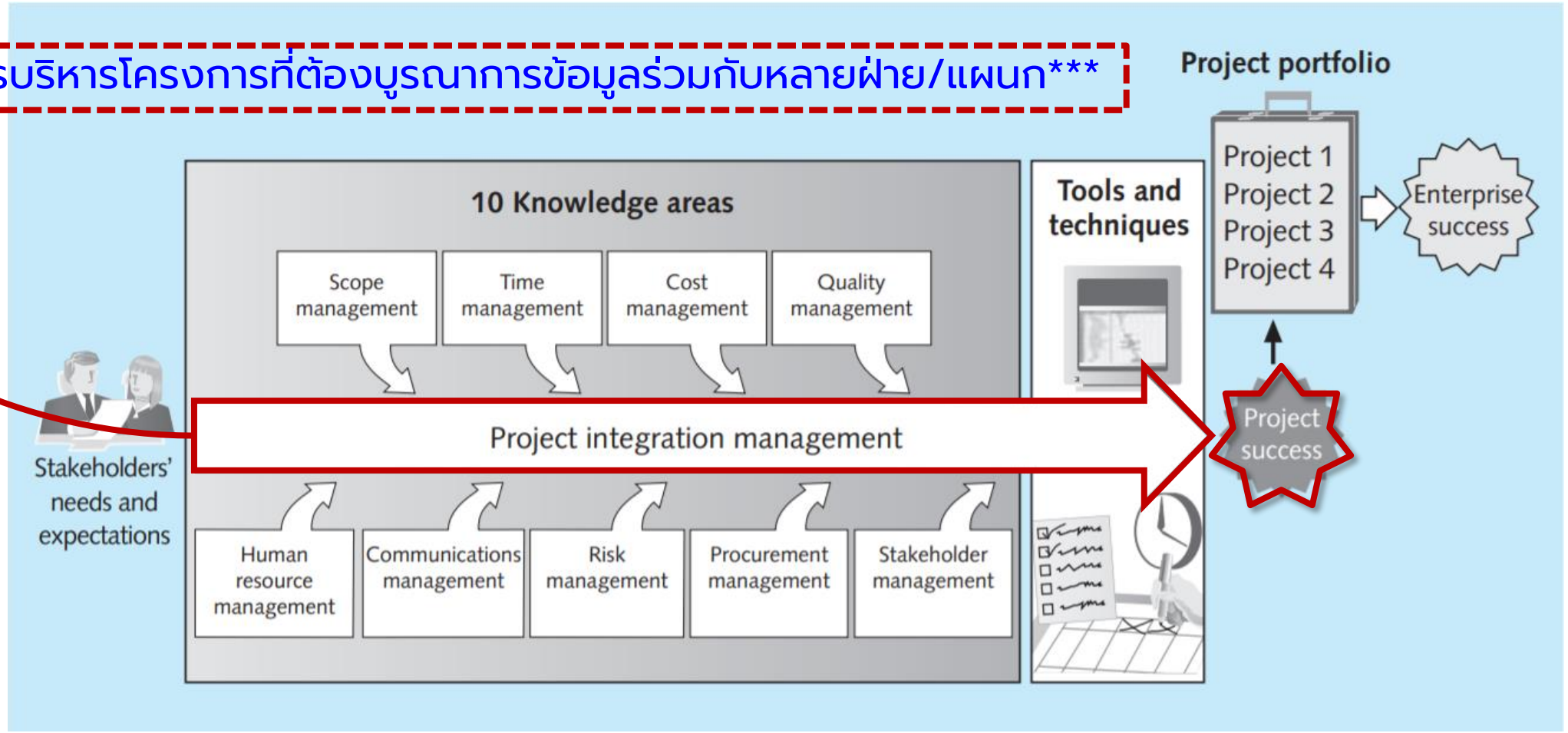


การบริหารโครงการ (Project Management)

- การประยุกต์ใช้องค์ความรู้ ทักษะ เครื่องมือ และเทคนิค
  - เพื่อดำเนินกิจกรรมตามความต้องการของโครงการ ให้บรรลุวัตถุประสงค์
  - ผู้บริหารโครงการ (Project Manager) มีบทบาทสำคัญ
  - มุ่งเน้นโครงการผลิตซอฟต์แวร์ (Software Project)

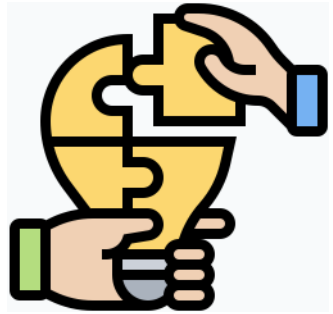


องค์ความรู้ของการบริหารโครงการ



ที่มา: Kathy Schwalbe, Information Technology Project Management, Eighth Edition, 2016.

วงจรการพัฒนาโครงการ



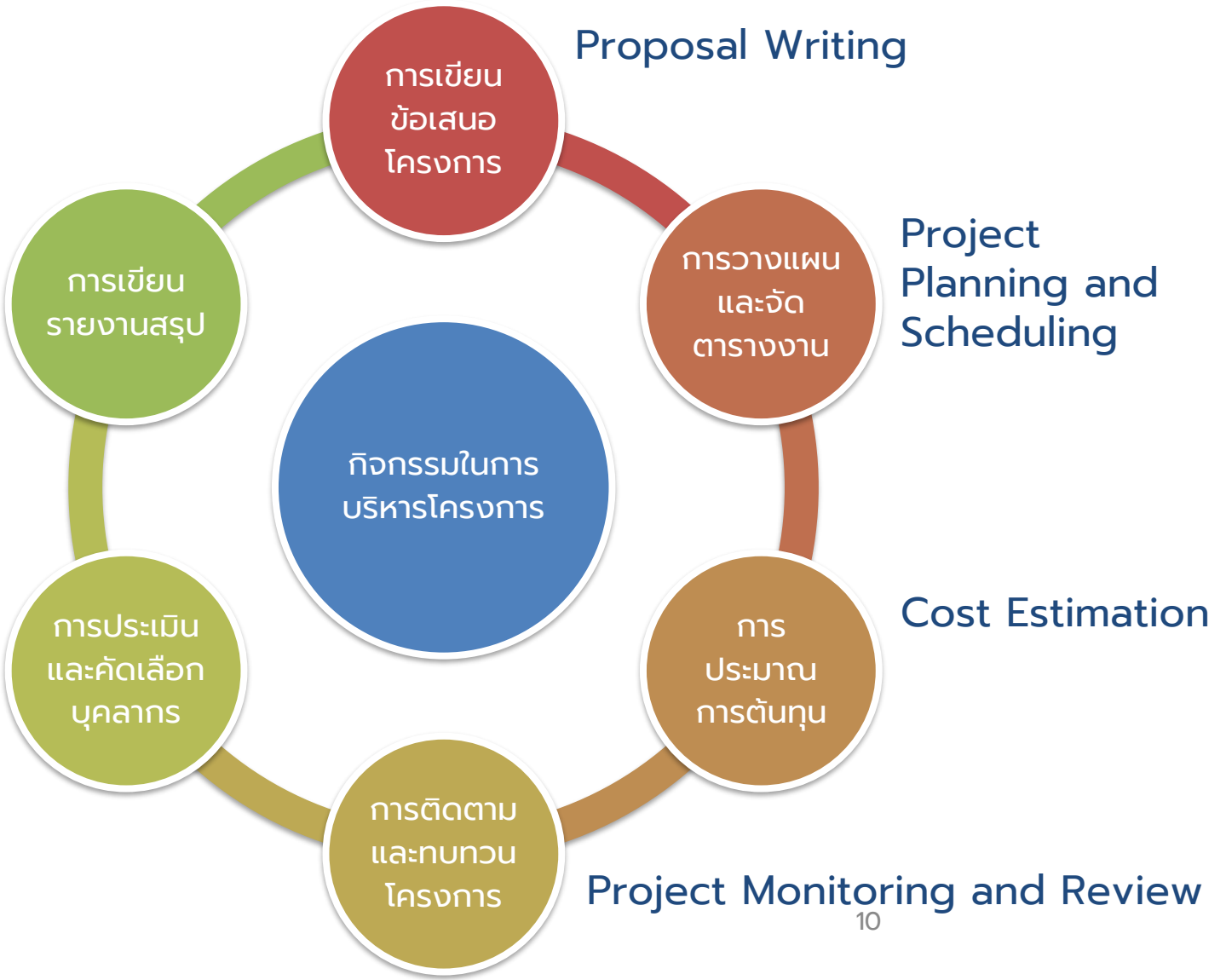
การเริ่มโครงการ
(Project Initiation)

การวางแผนโครงการ
(Project Planning)

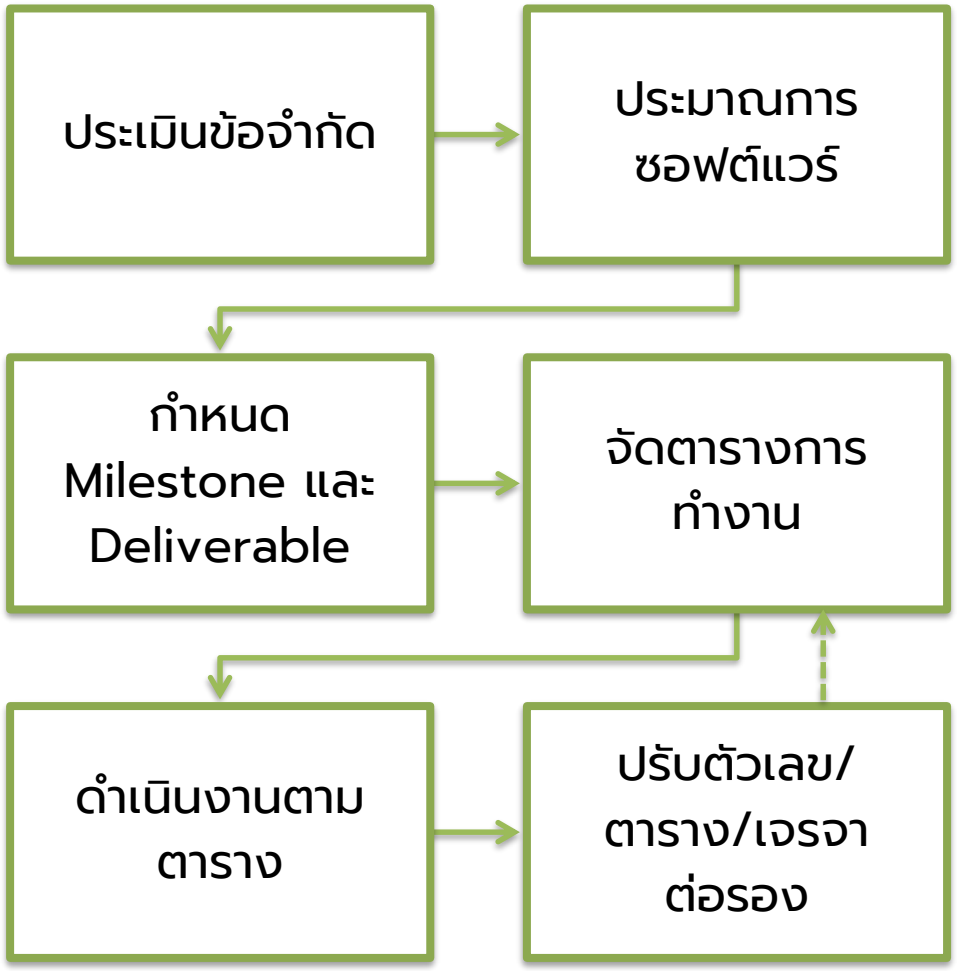
การดำเนินการโครงการ
(Project Execution)

การปิดโครงการ
(Project Closing)

กิจกรรมในการบริหารโครงการ



การวางแผนโครงการ



- แผนงานคุณภาพ (Quality Plan)
- แผนงานการตรวจสอบ (Validation Plan)
- แผนงานการจัดการการตั้งค่า (Configuration Management Plan)
- แผนงานการบำรุงรักษา (Maintenance Plan)
- การพัฒนาบุคลากร (Staff Development)

การจัดการความเสี่ยงโครงการ (Project Risk Management)

- การระบุความเสี่ยง (Risk Identification)
- การวิเคราะห์ความเสี่ยง (Risk Analysis)
- การวางแผนความเสี่ยง (Risk Planning)
- การติดตามความเสี่ยง (Risk Monitoring)

Risk Identification

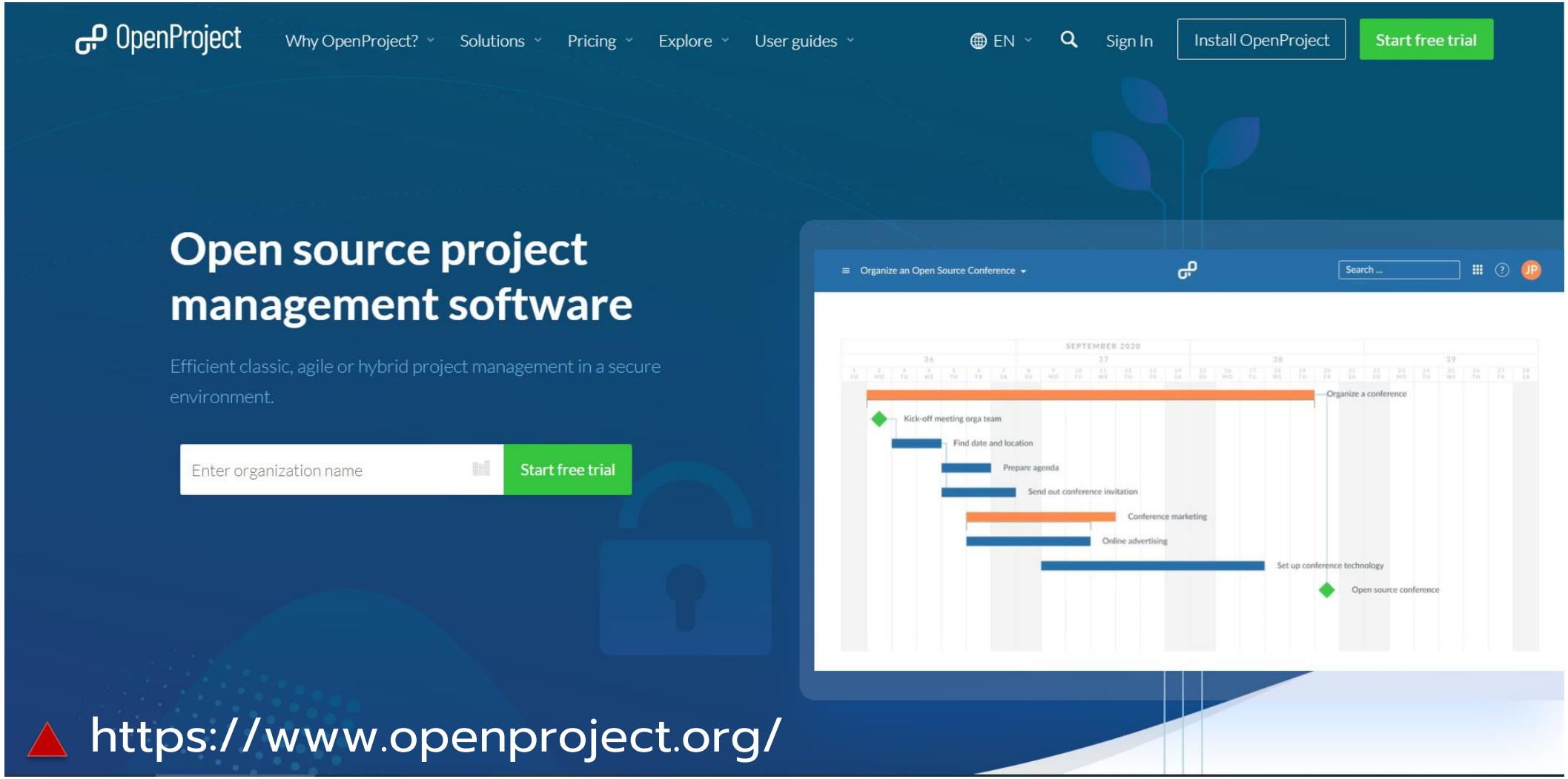
แนวทางการนิยามหรือกำหนดความเสี่ยง		
กำหนดความเสี่ยง	รายละเอียด	หมายเหตุ
ความต้องการ (Requirement)	- ไม่ครอบคุลม - สื่อสารกันคลาดเคลื่อน	- ปัญหาที่เกิดจากองค์กรเอง เช่น ปัญหาทางการเงิน ฯลฯ
บุคลากร (People)	- ขาดทักษะ - ขาดประสบการณ์ทำงาน	
เครื่องมือ (Tools)	- ขาดแคลน/ทักษะการใช้งาน	
การประมาณค่าใช้จ่าย (Cost Estimation)	- ประมาณการไว้ต่ำกว่าความจริง	
เทคโนโลยี (Technology)	- มีการเปลี่ยนแปลงเร็วกว่าที่กำหนด	

การจัดการความเสี่ยงโครงการ: การวิเคราะห์ความเสี่ยง

ลำดับ	ความเสี่ยง	ความน่าจะเป็น	โอกาสที่จะเกิด	ความเสียหาย	หมายเหตุ
1	ความต้องการ	X	X	X	ลำดับที่ 1 มีความสำคัญสูงสุด
2	การประมาณค่าใช้จ่าย	X	X	X	
3	เทคโนโลยี	X	X	X	
4	บุคลากร	X	X	X	
5	เครื่องมือ	X	X	X	

<u>หมายเหตุ</u> ค่าความน่าจะเป็นมีค่า 0-1 โดยค่าน้อยโอกาสเกิดความเสี่ยงก็น้อยมาก 1 มีโอกาสเกิดสูงมาก (ค่า x ได้จากผู้เชี่ยวชาญหรือข้อมูลในอดีต)

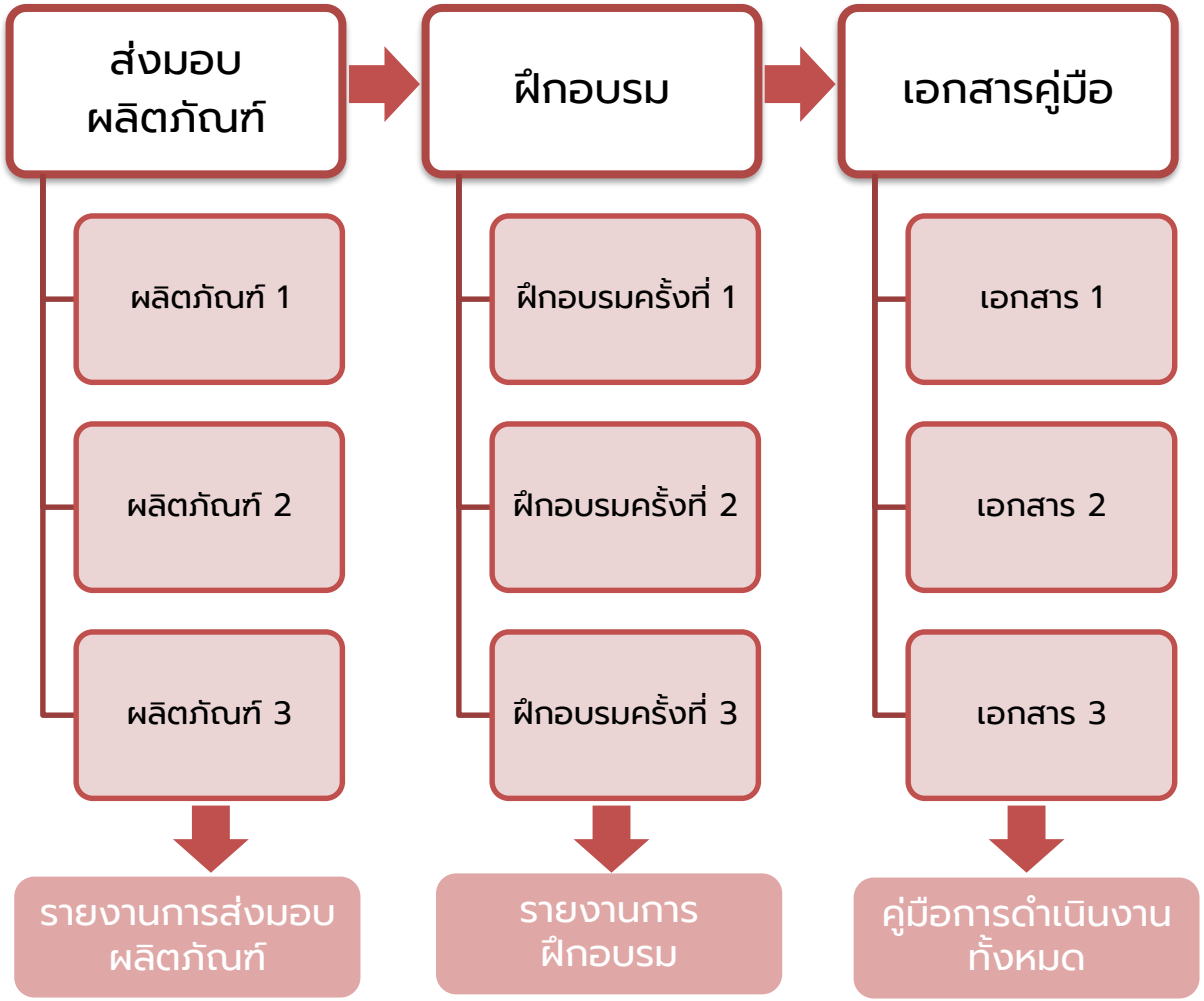
เครื่องมือการบริหารโครงการ (Project Management Tool)



- https://www.openproject.org/

กิจกรรมเป้าหมาย (Milestone) และผลลัพธ์ส่งมอบ (Deliverable)

- **กิจกรรมเป้าหมาย (Milestone)**
  - เป้าหมายของกิจกรรม มีประโยชน์ต่อการติดตามความก้าวหน้า
  - ทีมงานต้องส่งมอบให้แก่ผู้บังคับบัญชา ในรูปแบบที่เป็นทางการ
- **ผลลัพธ์การส่งมอบ (Deliverable)**
  - ผลลัพธ์ที่จะส่งมอบให้ลูกค้า ในแต่ละขั้นตอนของโครงการ

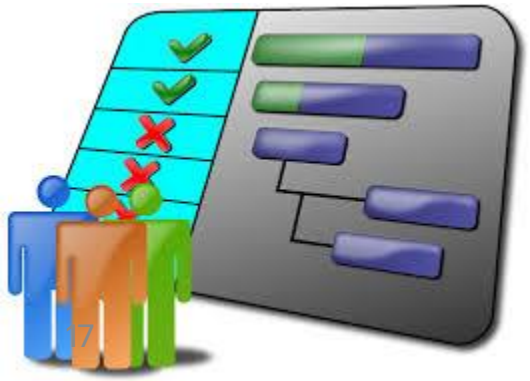


ผลลัพธ์จากขั้นตอนการวางแผนโครงการ

หัวข้อ	รายละเอียด
บทนำ (Introduction)	วัตถุประสงค์และข้อจำกัด
โครงสร้างโครงการ (Project Organization)	โครงสร้างบุคลากร จำแนกตามหน้าที่รับผิดชอบ
การวิเคราะห์ความเสี่ยง (Risk Analysis)	รายการความเสี่ยง/ความน่าจะเป็น/วิธีการลด
ฮาร์ดแวร์และซอฟต์แวร์ (Hardware and Software)	ฮาร์ดแวร์ และซอฟต์แวร์ที่ใช้ในการดำเนินโครงการ
การจัดกิจกรรม (Work Breakdown)	กิจกรรมหลัก/ย่อย เป้าหมาย และวันส่งมอบงาน
การจัดตารางงาน (Project Schedule)	ความสัมพันธ์ระหว่างกิจกรรม และระยะเวลา เพื่อให้บรรลุเป้าหมาย
การติดตามและรายงานผล (Monitoring and Reporting)	รายงานสำหรับการบริหารและติดตาม

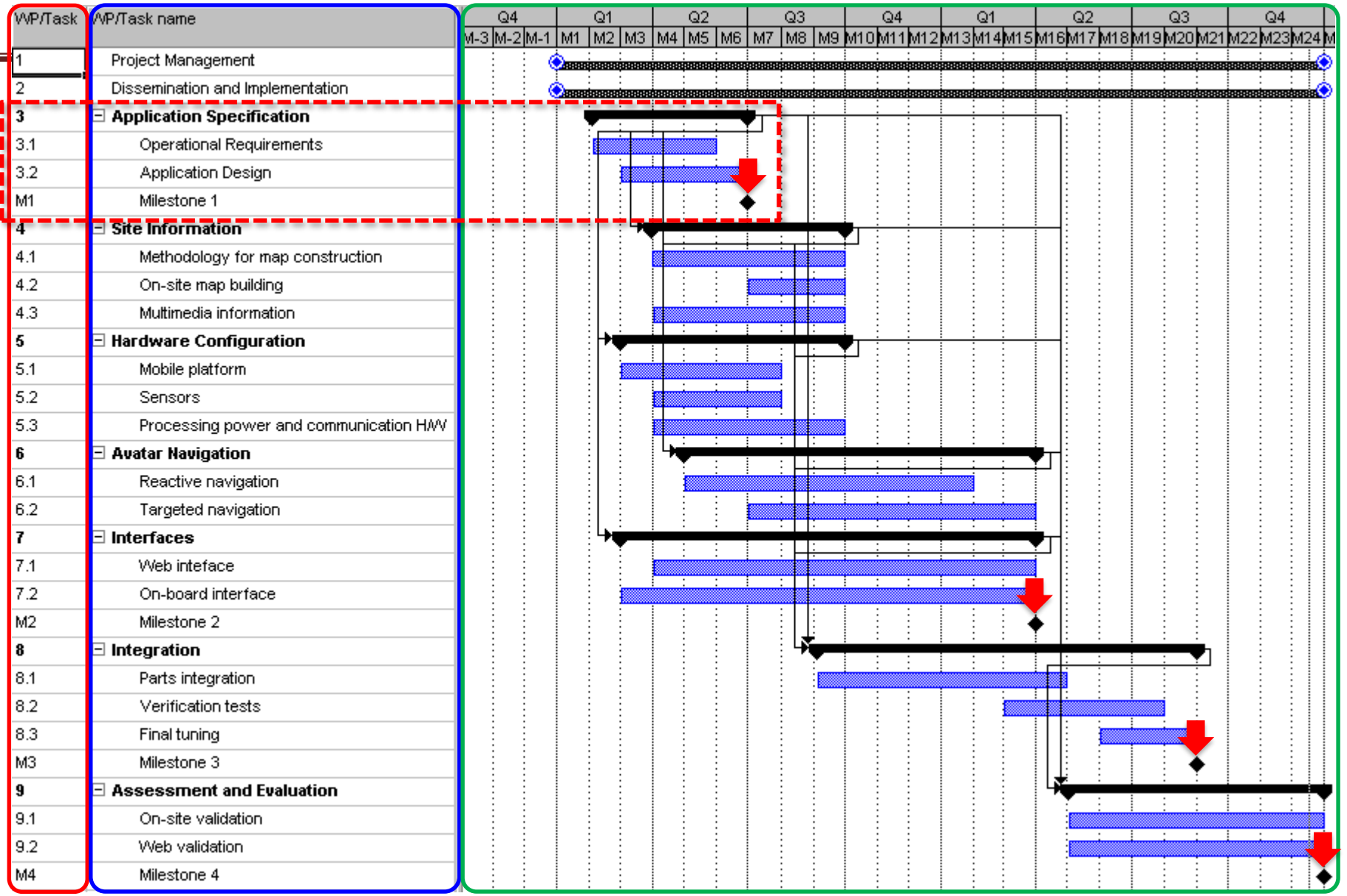
การจัดตารางงานโครงการ (Project Scheduling)

- ผู้บริหารโครงการ
  - แบ่งกิจกรรมหลักเป็นกิจกรรมย่อย เพื่อกำหนดระยะแล้วเสร็จ
  - สร้างความสัมพันธ์ให้กับทุกกิจกรรม
  - จัดสรรบุคลากรให้เหมาะสม
  - เพื่อป้องกัน Critical Task
  - เริ่มด้วย ระยะเวลาที่กิจกรรมให้แล้วเสร็จ + ระยะเวลาที่ต้องแก้ปัญหา
  - เครื่องมือและเทคนิคต่าง ๆ
    - PERT/PCM, Gantt Chart
    - Microsoft Project



Gantt Chart

- พัฒนาโดย Henry L. Gantt ในปี 1917
- Project Scheduling
- กราฟแท่งแนวนอนแสดงระยะเวลาของกิจกรรมแต่ละขั้นตอน ดังนี้
  - รายชื่อกิจกรรมแสดงในแนวตั้งด้านซ้าย
  - ระยะเวลาการทำงานแสดงในแนวนอน

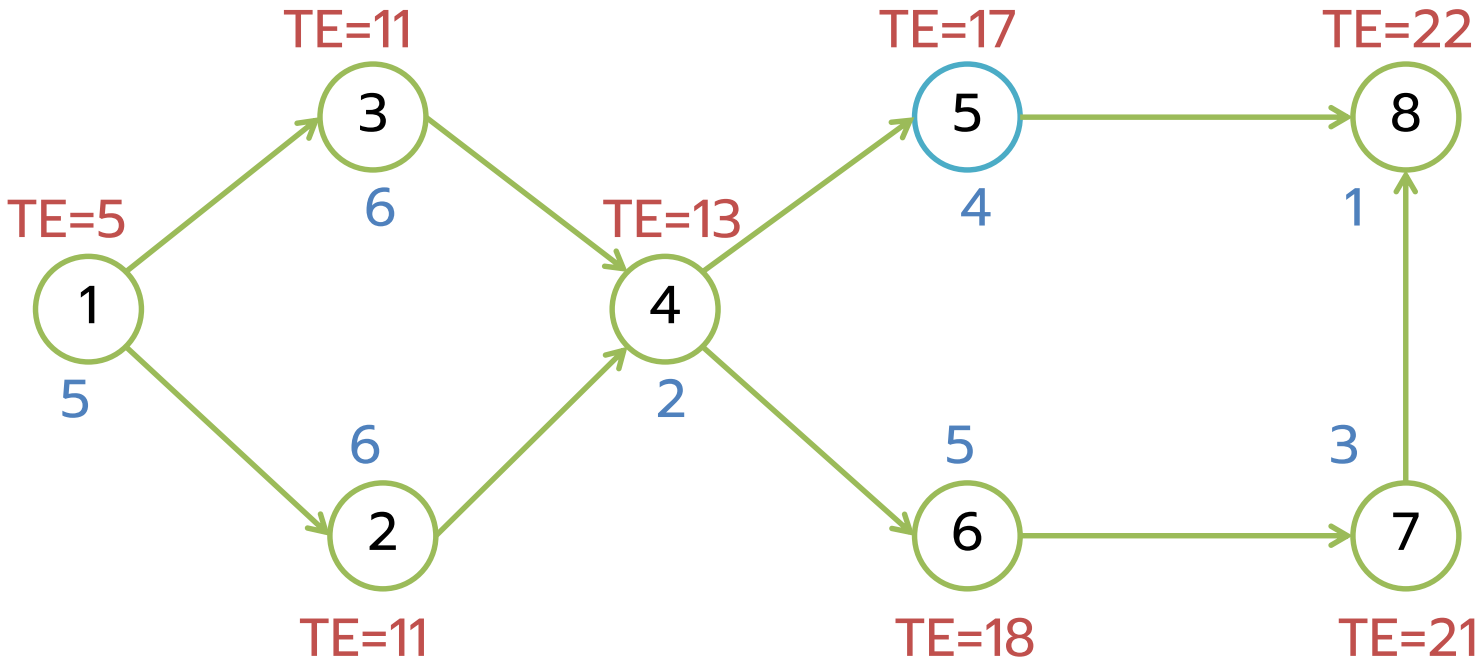


เทคนิคการทบทวนการประเมินผลโครงการ (PERT) / วิธีการหาเส้นทางวิกฤต (CPM)

- เทคนิคการทบทวนการประเมินผลโครงการ (Project Evaluation Review Technique: PERT)
  - เทคนิคการวิเคราะห์ประเมินเวลาที่ใช้ในกิจกรรม
  - แสดงเป็นแผนภาพกิจกรรมของโครงการที่เชื่อมโยงกันในลักษณะของเครือข่าย
  - โครงการใหม่ เป็นการกำหนดรูปแบบของความน่าจะเป็น
- วิธีการหาเส้นทางวิกฤต (Critical Path Method: CPM)
  - เทคนิคในการวิเคราะห์เส้นทาง/กิจกรรมวิกฤติ
  - โครงการที่เกิดขึ้นแล้ว มีข้อมูลเดิมสำหรับกำหนดระยะเวลาของกิจกรรม

กิจกรรม	กิจกรรมก่อนหน้า (Predecessor)	เวลา (หน่วย: Week)
1. รวบรวมความต้องการ	-	5
2. ออกแบบรายงาน	1	6
3. ออกแบบหน้าจอ	1	6
4. ออกแบบฐานข้อมูล	2, 3	2
5. จัดทำเอกสาร	4	4
6. เขียนโปรแกรม	4	5
7. ทดสอบโปรแกรม	6	3
8. ติดตั้งโปรแกรม	5, 7	1

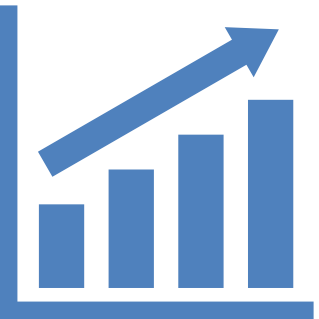
PERT/CPM (Activity On Node)

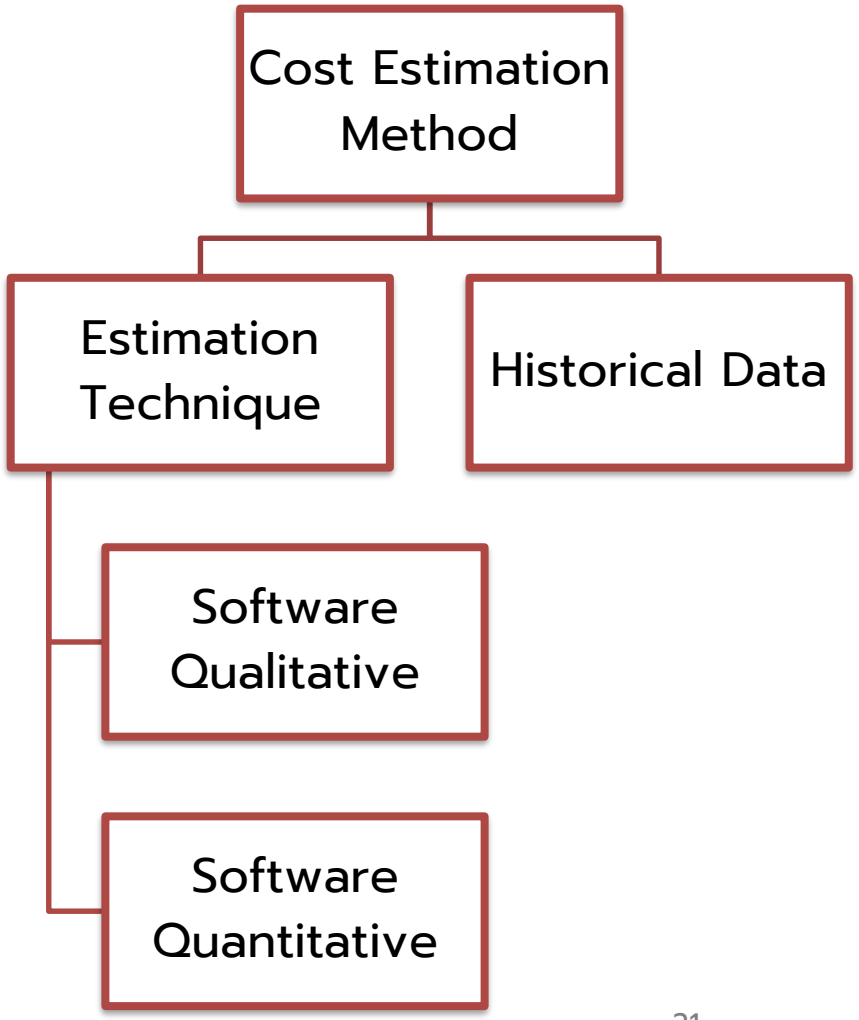


คำนวณหาเส้นทางวิกฤติในการดำเนินกิจกรรม
คำนวณเวลาเร็ว/ช้า/เร่ง
คำนวณค่าใช้จ่ายและแรงงาน

การประมาณการต้นทุนซอฟต์แวร์

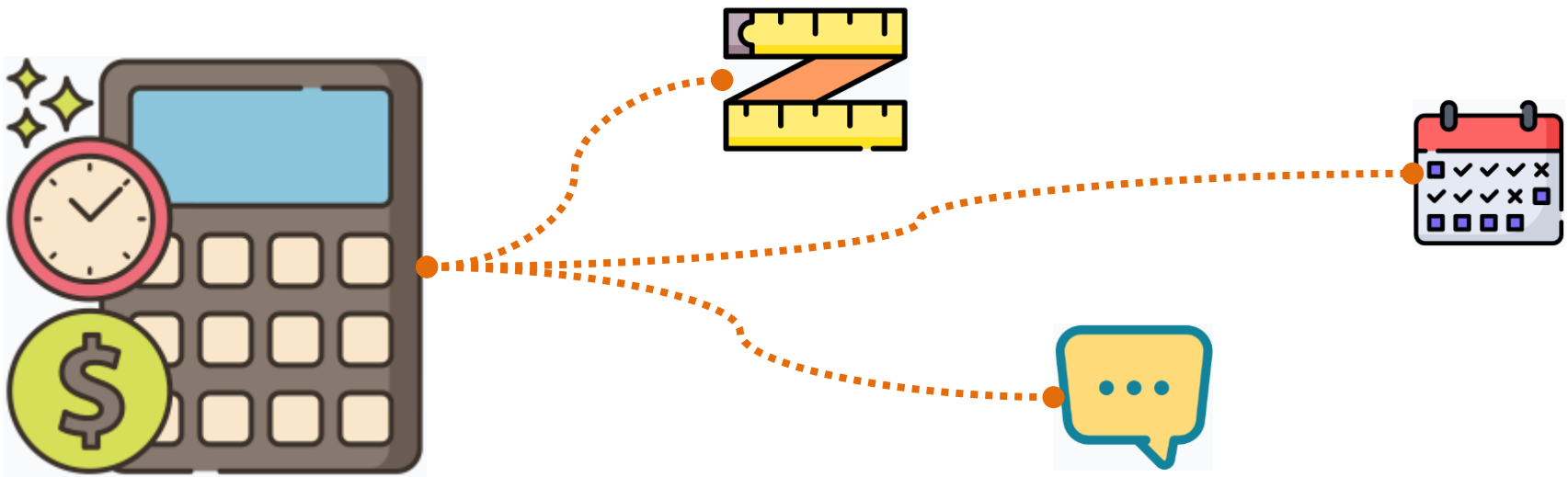
- วิธีการวัดผล (Measurement method)
  - เทคนิคการประมาณต้นทุน (Estimation Techniques) แบ่งเป็น 2 ลักษณะ ดังนี้
    - ซอฟต์แวร์เชิงคุณภาพ (Software Qualitative)
    - ซอฟต์แวร์เชิงปริมาณ (Software Quantitative)
  - ข้อมูลเดิม (Historical Data)





ปัจจัยของการประมาณการราคา

ปัจจัย	รายละเอียด	หมายเหตุ
ขนาด (Size)	- Line of Code (LOC) - Function Point (FP)	นับจำนวนฟังก์ชัน (Function)
ตารางงาน (Schedule)	ระยะเวลาแล้วเสร็จโครงการ	พิจารณาด้านผลกระทบน้อยหรือมาก
อื่น ๆ (Other)	ขึ้นอยู่กับสภาพแวดล้อม	ตัวอย่าง ประเด็นการพิจารณา - เร่งด่วนแข่งขันประกวดฯ - ขาดผู้เชี่ยวชาญในการประเมินราคา



Language	QSM SLOC/FP Data			
	Avg	Median	Low	High
ABAP (SAP) *	28	18	16	60
ASP*	51	54	15	69
Assembler *	119	98	25	320
Brio +	14	14	13	16
C *	97	99	39	333
C++ *	50	53	25	80
C# *	54	59	29	70
COBOL *	61	55	23	297
Cognos Impromptu Scripts +	47	42	30	100
Cross System Products (CSP) +	20	18	10	38
Cool:Gen/IEF *	32	24	10	82
Datastage	71	65	31	157
Excel *	209	191	131	315
Focus *	43	45	45	45
FoxPro	36	35	34	38
HTML *	34	40	14	48
J2EE *	46	49	15	67
Java *	53	53	14	134
JavaScript *	47	53	31	63

Note: * Languages with updated gearing factors.
+ New languages for which gearing factor data was not previously reported.

ที่มา: http://www.qsm.com/resources/function-point-languages-table

การวัดขนาดของซอฟต์แวร์

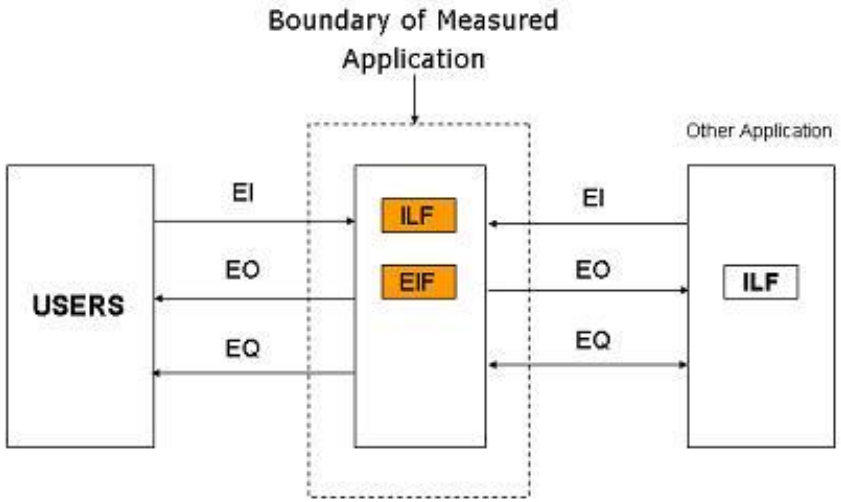
- Line of Code (LOC)
  - นับจำนวนบรรทัดโปรแกรม (Loc)
  - 1 คำสั่ง = Loc
- Function Point (FP)
  - นับจำนวนฟังก์ชัน
  - วัดภายใต้ความต้องการของซอฟต์แวร์
  - ผูกพันกับเวลา บุคลากรและอื่น ๆ

Block-based coding:

JavaScript:

```
for (var count = 0; count < 4; count++) {
  moveForward(100);
  turnRight(144);
}
```

(We count both of the above as 3 lines of code)



การวัดขนาดของซอฟต์แวร์ด้วยวิธีประมาณการ

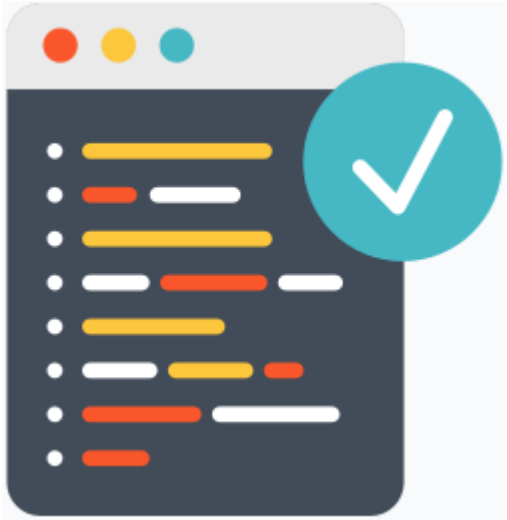
- การประมาณการด้วยวิธีเชิงอ้างอิง (Benchmark)
  - ใช้การเปรียบเทียบกับซอฟต์แวร์อื่นประเภทเดียวกัน
  - ใช้ซอฟต์แวร์อื่นเป็นข้อมูลอ้างอิง
- การประมาณการโดยผู้เชี่ยวชาญ
  - ให้ผู้เชี่ยวชาญประมาณขนาดซอฟต์แวร์
  - สามารถประมาณการแบบหลายคนร่วมตัดสินใจ (Delphi Approach)

Line of Code (LOC)

- **Line of Code** แบ่งเป็นหลายวิธี ดังนี้
  - ☑ Simple Line of Code นับทุกบรรทัด
  - ☑ Physical Lines (LINES) ไม่นับบรรทัดที่นิยามตัวแปร
  - ☑ Physicals Line of Code
    - ไม่นับบรรทัดว่างและ comment
    - **Source Line Code (sLOC)**
  - ☑ Logical Lines of Code (LLOC) เหมือน Physicals
    - นับบรรทัดที่เชื่อมต่อด้วย "_" เป็นหนึ่งบรรทัด
  - ☑ Statements (STMT)
    - เป็นการนับจำนวนประโยคคำสั่ง

ข้อจำกัด

- ขึ้นอยู่กับภาษาโปแกรม
- การออกแบบโปรแกรม



```java
/*
    Calculate Circle Area using Java Example
    This Calculate Circle Area using Java Example shows how to calculate
    area of circle using it's radius.
*/

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class CalculateCircleAreaExample {

    public static void main(String[] args) {

        int radius = 0;
        System.out.println("Please enter radius of a circle");

        try
        {
            //get the radius from console
            BufferedReader br = new BufferedReader(newInputStreamReader(System.in));
            radius = Integer.parseInt(br.readLine());
        }
        catch(NumberFormatException ne) //if invalid value was entered
        {
            System.out.println("Invalid radius value" + ne);
            System.exit(0);
        }
        catch(IOException ioe)
        {
            System.out.println("IO Error :" + ioe);
            System.exit(0);
        }

        /*
         * Area of a circle is
         * pi * r * r
         * where r is a radius of a circle.
         */

        //NOTE : use Math.PI constant to get value of pi
        double area = Math.PI * radius * radius;

        System.out.println("Area of a circle is " + area);
    }
}

/*
Output of Calculate Circle Area using Java Example would be
Please enter radius of a circle
19
Area of a circle is 1134.1149479459152
*/
```

Java: ....

JavaScript: ....

```javascript
function circle(radius)
{
  this.radius = radius;
 // area method
  this.area = function ()
  {
    return Math.PI * this.radius * this.radius;
  };
 // perimeter method
  this.perimeter = function ()
  {
    return 2*Math.PI*this.radius;
  };
}
var c = new circle(3);
console.log('Area =', c.area().toFixed(2));
console.log('perimeter =', c.perimeter().toFixed(2));
```

การวัดประสิทธิผลในการผลิต

- **ประสิทธิผลในการผลิต (Productivity)** ประมาณการต้องใช้บุคลากรจำนวนเท่าใด

$$Productivity = \frac{Size\ [FP\ or\ LOC]}{Effort\ [Man\text{-}Month]}$$

- โดย
  - *FP* คือ Function Point
  - *Loc* คือ Line of Code

ตัวอย่าง โปรแกรมเมอร์ 1 คน สามารถผลิตซอฟต์แวร์ขนาด 60 FP ได้ในระยะเวลา 2 เดือน จงคำนวณหา Productivity

การวัดประสิทธิผลในการผลิต (ต่อ)

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

ภาษา (Language)	ขนาด (Size)	เวลาดำเนินการ (Effort)	ประสิทธิผลการผลิต (Productivity)
Assembly	6,000 บรรทัด		
C Language	2,500 บรรทัด		

หมายเหตุ: การดำเนินการกับค่า Effort ต้องหารด้วย 4 ก่อน เพื่อแปลงเป็นหน่วยเป็นเดือน

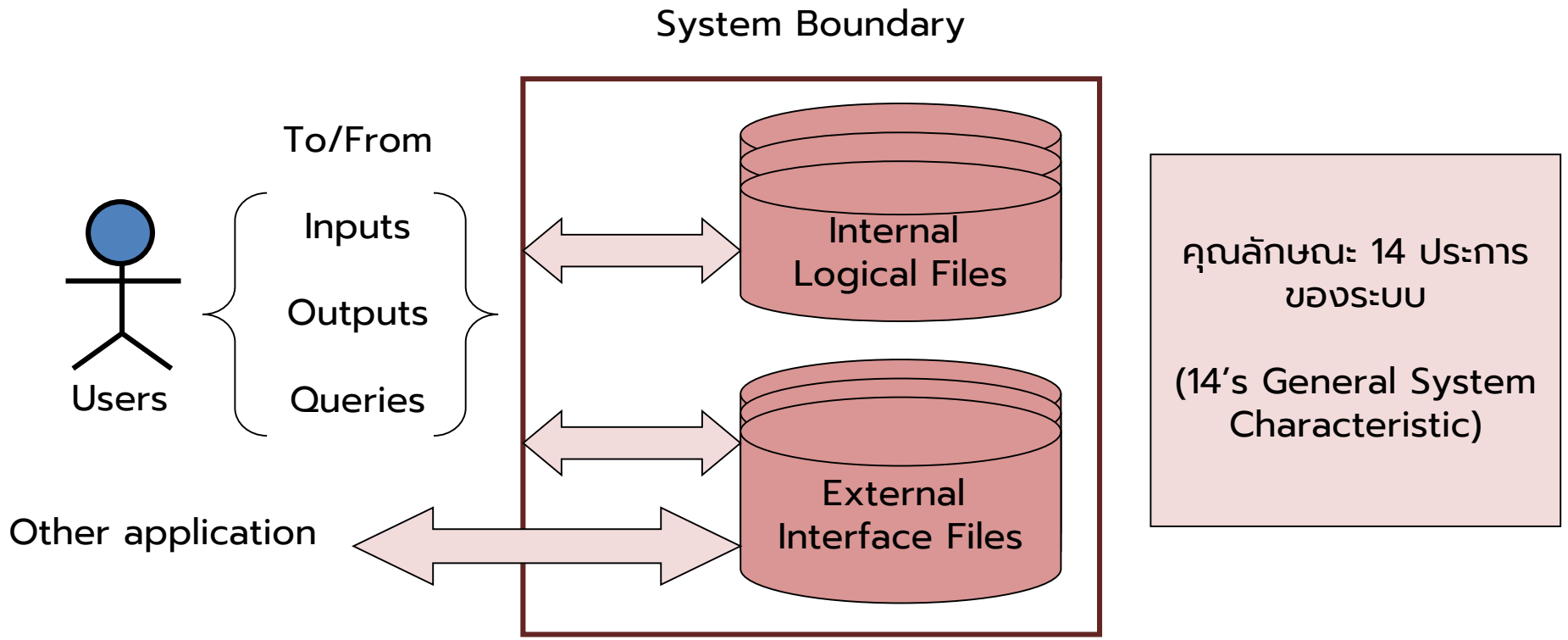
Function Point (FP)

- เพื่อลดปัญหาด้านความแตกต่างของ Programming และ Technology
- Function Point มีแบบถ่วงน้ำหนัก (Weight) มีรูปแบบ ดังนี้

$$FP = UFP \times VAF$$

- โดยกำหนดดังนี้
  - UFP คือ FP ที่ยังไม่ได้ปรับแต่ง (Unadjusted Function Point)
  - VAF คือ ค่าปัจจัยคุณลักษณะของระบบ (Value Adjustment Factor)

Function Point (ต่อ)

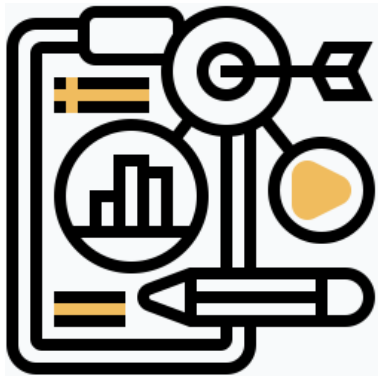


Size (Function Point) = UFP x VAF

Function Point (ต่อ)

- การคำนวณ แบ่งออกเป็น 3 ขั้นตอน ดังนี้

1. คำนวณหา FP ที่ยังไม่ได้ปรับแต่ง (UFP)
2. คำนวณค่าปัจจัยคุณลักษณะของระบบ (VAF)
3. คำนวณค่า FP ที่ปรับแต่งแล้ว



1.) คำนวณหา FP ที่ยังไม่ได้ปรับแต่ง (UFP)

ฟังก์ชัน	รายละเอียด
External Inputs (EI)	ข้อมูลที่รับเข้ามาในระบบ (อาจเป็นข้อมูลทางธุรกิจหรือข้อมูลควบคุม) เพื่อนำไปอัพเดทข้อมูลใน ILF เช่น ข้อมูลในกระบวนการ เพิ่ม ลบ แก้ไขข้อมูล เป็นต้น
External Outputs (EO)	ข้อมูลที่เป็นผลลัพธ์จากการประมวลผลข้อมูลที่ได้รับจากภายในระบบ ให้นับการแสดงผลข้อมูลที่มีรูปแบบแตกต่างกัน
External Queries (EQ)	กระบวนการดึงข้อมูลและประมวลผลเพื่อแสดงผลต่อผู้ใช้ (การ Query ข้อมูล)
Internal Logical Files (ILF)	ไฟล์ที่เกี่ยวข้องกับข้อมูลที่อยู่ในระบบตลอดช่วงอายุของระบบ และเป็นไฟล์มักจะถูกบำรุงรักษาหรือปรับปรุงด้วยข้อมูลที่ได้รับจากภายนอก (EI) ให้นับรวมเรคอร์ดที่ทำหน้าที่เทียบเท่ากับไฟล์ด้วย
External Interface Files (EIF)	ไฟล์ที่เกี่ยวข้องกับข้อมูลที่ใช้เพื่อการอ้างอิงเท่านั้น และใช้ร่วมกับระบบอื่นๆ EIF เป็นไฟล์ที่ถูกเรียกใช้โดยระบบจะพัฒนา แต่จะบำรุงรักษาหรือถูกสร้างโดยระบบอื่น

Note: ฟังก์ชันแต่ละประเภทเกิดจากการทำรายการข้อมูลของผู้ใช้ จึงมีความซับซ้อนแตกต่างกันตามจำนวนข้อมูล (DET) เรคคอร์ด (RET) และไฟล์ที่เกี่ยวข้อง (FTR) จากนั้นจึงนำมาเทียบกับตารางเกณฑ์ระดับความซับซ้อนของฟังก์ชั่น (ต่ำ ปานกลาง สูง) แล้วนำมาคูณกับตัวถ่วงน้ำหนัก แล้วหาผลรวมฟังก์ชันทั้งหมดที่นับได้

ตารางเกณฑ์ความซับซ้อน

ประเภทของ Function Point	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
	ค่าผลรวมฟังก์ชันพอยต์ที่ยังไม่ปรับค่า (Total Unadjusted Function Points)		

1.) คำนวณหา FP ที่ยังไม่ได้ปรับแต่ง (UFP) (ต่อ)

- จากการนับจำนวนฟังก์ชันของระบบสารสนเทศต้องนำไปคูณกับค่าน้ำหนักของแต่ละหมวดหมู่ฟังก์ชัน โดยนำจำนวนของฟังก์ชันที่นับได้ในแต่ละหมวดหมู่ x ค่าน้ำหนักของหมวดหมู่นั้นๆ โดยน้ำหนักที่คูณอาจมีการคำนึงถึงความซับซ้อน

หมวดหมู่	น้ำหนักของหมวดหมู่	จำนวนฟังก์ชัน	ผลคูณ
EIs	4		
EOs	5		
EQs	4		
EIFs	7		
ILFs	10		
ค่าฟังก์ชันพอยต์ที่ยังไม่ได้ปรับค่า			

2.) คำนวณค่าปัจจัยคุณลักษณะของระบบ (VAF)

- คำนวณค่าปัจจัยที่ส่งผลต่อความแตกต่างกัน หมายถึง <u>คุณลักษณะเด่นของระบบทั้งหมด 14 ข้อ</u> ตามข้อกำหนดความต้องการของลูกค้า โดยกำหนดค่าตั้งแต่ 0 (ไม่เกี่ยวข้อง) - 5 (เกี่ยวข้องมาก)

VAF = 0.65 + [0.01 x ผลรวมค่าคุณลักษณะ 14 ด้าน]

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

2.) คำนวณค่าปัจจัยคุณลักษณะของระบบ (VAF) (ต่อ)

- **ตัวอย่าง** หากประเมินแล้วแต่ละปัจจัยอยู่ที่ 4 จะมีผลรวมในการประเมิน (Sum of Score: SS) เท่ากับ A x B = X ดังนั้น

VAF = 0.65 + [0.01 x X]
= X

3.) คำนวณค่า FP ที่ปรับแต่งแล้ว

- เมื่อคำนวณหา UFP และ VAF แล้วนำมาคูณกันจะได้ผลลัพธ์เป็นค่า FP ที่ปรับแต่งแล้วตามคุณลักษณะเด่นของระบบ
- จากสูตรการคำนวณ FP = UFP * VAF

Function Point (ต่อ)

- LoC และ FP คือ วิธีวัดขนาดของซอฟต์แวร์
  - สามารถนำไปคำนวณหา Productivity และ Effort
  - สูตรคำนวณเพื่อประมาณการ Effort นั้น บางครั้งอาจต้องการใช้ขนาดของซอฟต์แวร์ที่เป็น LoC
  - ค่า FP ที่ได้ต้องแปลงเป็น LoC ตามตารางเปรียบเทียบตามมาตรฐานของ QSM *(Quantitative Software Management: http://www.qsm.com)*

จากตัวอย่างถ้าหากจัดทำซอฟต์แวร์จะได้ LoC โดยใช้ภาษา Java หรือ C ดังนี้

Java = 52 x ..... = ... SLOC

C = 52 x ..... = ... SLOC

เทคนิคการประมาณการต้นทุนและ Effort

เทคนิคการประมาณการต้นทุนและ Effort	
Algorithmic Cost Modeling	ประเมินราคาโดยการสร้างตัวแบบคณิตศาสตร์ ใช้ข้อมูลที่เกี่ยวข้องกับการพัฒนาในอดีตมานิยามค่าคงที่ต่างๆ สมการตัวแบบที่นิยมใช้คือ COCOMO ซึ่งเป็นการประมาณการค่าใช้จ่ายการพัฒนา Software โดยพิจารณาจาก จำนวนบรรทัด ของโปรแกรมหรือจำนวน Function point
Expert Judgement	ผู้บริหารโครงการที่มีประสบการณ์หลายคนมาเป็นผู้ประเมิน โดยต่างคนต่างประเมินค่าของตนเอง จากนั้นนำข้อมูลต่างๆ มาวิเคราะห์ร่วมกันเพื่อหาข้อสรุปของค่าที่เหมาะสม
Estimation by Analogy	ใช้ประเมินกับโครงการซอฟต์แวร์ที่มีลักษณะคล้ายคลึงกัน (โครงงานในอดีตที่พัฒนาเสร็จสมบูรณ์แล้ว) ใช้ได้ดีกับโครงการขนาดใหญ่ที่มีวิธีการดำเนินงานที่คล้ายๆ กัน แต่ไม่เหมาะกับโครงการขนาดเล็กที่มีลักษณะเฉพาะ
Parkinson's Law	ยึดกฎของ Parkinson "ปริมาณงานจะขยายตัวไปได้เรื่อยจนกระทั่งครบตามเวลาที่กำหนดไว้ เน้นพิจารณารายละเอียดของทรัพยากรที่มีอยู่ (คน, เวลา) มากกว่าการประเมินจากจุดมุ่งหมายของโครงการ"
Pricing to Win	ประเมินค่าใช้จ่ายจากความสามารถในการชำระเงินหรืองบประมาณของลูกค้า ไม่คำนึงคุณภาพของงานที่ต้องมีในซอฟต์แวร์ และความต้องการของลูกค้า
Top-down Estimation	เริ่มวิเคราะห์จากค่าใช้จ่ายทั้งหมดของโครงการก่อนเป็นหลัก จากนั้นจึงเริ่มพิจารณาองค์ประกอบย่อยต่างๆ ของระบบ ข้อดีเน้นที่ความสำคัญของวัตถุประสงค์
Bottom-Up Estimation	ประเมินจากค่าใช้จ่ายของแต่ละองค์ประกอบในโครงการก่อน จากนั้นนำค่าใช้จ่ายต่างๆ มาสรุปร่วมกันเป็นค่าใช้จ่ายสุดท้าย

การประมาณด้วยโมเดล COCOMO II

- Constructive Cost Model
  - ใช้ข้อมูล Historical Data ประกอบการคำนวณ ประกอบกับใช้หลักสถิติและคณิตศาสตร์กับการปรับแต่งค่าร่วมกันคำนวณ
- COCOMO II เป็นแบบประมาณการต้นทุนและ Effort
  - แบ่งออกเป็น 3 ชนิด เพื่อประมาณการต้นทุน Effort ในระยะต่างๆ ดังนี้
    - Application-Composition Model ระยะสรุป concept ในการดำเนินโครงการใช้ *Object Point* แทนขนาดของ SW
      - **Object Point* จำนวน object หมายถึงคอมโพเน้นท์ 3 ส่วน คือ หน้าจอ (screen) รายงาน (report) และโมดูล (module)
    - Early Design Model ใช้ประมาณการในระยะก่อนออกแบบซอฟต์แวร์ แต่ต้องหลังจากกำหนดความต้องการเรียบร้อยแล้ว
    - Post-Architecture Model ใช้ประมาณการในระยะหลังออกแบบซอฟต์แวร์ (เป็นการประมาณอีกรอบเพื่อความถูกต้องแม่นยำของค่าประมาณ

ระยะการออกแบบองค์ประกอบของแอปพลิเคชัน (Application-Composition Model)

- เหมาะกับการผลิตซอฟต์แวร์แบบ Component-Based Development
- การดำเนินงานอยู่ในระยะสรุป concept
- ใช้ Object Point (OP) เป็นตัววัดขนาดซอฟต์แวร์จำนวน OP แต่ต่างกันขึ้นกับความซับซ้อน ซึ่งจำเป็นต้องมีการปรับแต่ค่า OP ดังนี้

	ง่าย (Simple)	ซับซ้อน (Complex)	ซับซ้อนมาก (Very Complex)
Screen	1	2	3
Reports	2	5	8
3GL Module	4	10	-

ระยะการออกแบบองค์ประกอบของแอปพลิเคชัน (ต่อ)

$$Revised\ Object\ Point\ (ROP) = Object\ Point\ x \frac{(100 - \%reuse)}{100}$$

- ดังนี้น นำค่า ROP ไปคำนวณหาค่า Effort ดังนี้

$$ManMonthEffort\ (MME) = \frac{ROP}{Productivity\ Constant}$$

โดย

ค่าคงที่ที่บอกถึงประสิทธิผลของการพัฒนา (Productivity Constant) หน่วยเป็น NOP (Number of OP per month) ซึ่งขึ้นกับระดับประสบการณ์และความสามารถของทีมพัฒนา

MME ความพยายามที่ลงทุนไปนับเป็นจำนวนคนที่ต้องใช้ในเวลา 1 เดือนที่สามารถพัฒนาซอฟต์แวร์จนแล้วเสร็จ

ระดับประสบการณ์และความสามารถ	Very Low	Low	Nominal	High	Very High
Productivity Constant (NOP Per Month)	4	7	13	25	50

ตัวอย่าง การคำนวณหาค่าความพยายามจำนวนคน/เดือน (MME)

- ในระยะการก่อนดำเนินงาน โครงการไทยแลนด์ 4.0 นับจำนวน OP ได้ 40 OP มีอัตราการนำโค้ดไปใช้ใหม่ 10% และเมื่อประเมินประสบการณ์และความสามารถของทีมงานแล้ว พบว่าอยู่ในระดับ Nominal สามารถคำนวณหาค่า Effort ของโครงการได้ ดังนี้

ระยะก่อนการออกแบบ (Early Design Model)

- ใช้ประมาณการ Effort ในช่วงการออกแบบ หลังได้ความต้องการเรียบร้อยแล้ว โดยมีรูปแบบ ดังนี้

$$MME = A \times (Size)^{B}$$

- โดยกำหนดให้ ดังนี้
  - MME: Effort หน่วยเป็น Man-Month
  - A: ค่าคงที่ของประสิทธิผลในการผลิต ตามระดับความซับซ้อนคิดที่ระดับ Nominal
  - B: ค่าปัจจัยผลกระทบ (5 Factors)
  - Size: ขนาดของซอฟต์แวร์ *KLoC*

ระยะก่อนการออกแบบ (ต่อ)

- การคำนวณหาค่า B : ค่าปัจจัยผลกระทบ (5 Factors) เรียกปัจจัยเหล่านี้ว่า Scaling Factor, Economics Scale หรือ Cost Driver (ปัจจัยขับ)
- ซึ่ง B นี้แปรผันกับ Effort ในลักษณะ Exponential *(พิจารณาจากสูตรที่ MME = A*Size^B)*
  - ถ้า B=1 หมายถึง Scaling Factor ไม่มีผลต่อ Effort
  - ถ้า B>1 หรือ B<1 มีผลต่อ Effort ให้เพิ่มขึ้นหรือลดลง

$$B = 0.91 + 0.01 \sum_1^5 Ratings$$

ตาราง คะแนนของปัจจัยแต่ละระดับ (Value of Rating for Scaling Factor)

Factor Code	ต่ำมาก (Very Low)	ต่ำ (Low)	ปานกลาง (Nominal)	สูง (High)	Factor Name
PREC	6.20	4.96	3.72	2.48	Precedentness
FLEX	5.07	4.05	3.04	2.03	Flexibility
RESL	7.07	5.65	4.24	2.83	Risk Resolution
TEAM	5.48	4.38	3.29	2.19	Team Cohesion
PMAT	7.80	6.24	4.68	3.12	Process Maturity

ตาราง ปัจจัยขับสำหรับ COCOMO II ในระยะ Early Design

ปัจจัย	รายละเอียด
PREC	ความเหมือนของซอฟต์แวร์ใหม่กับซอฟต์แวร์เดิมที่เคยพัฒนามาแล้ว (เหมือนมาก คะแนนน้อยอยู่ในระดับสูง แปลว่าผลกระทบน้อย แต่ถ้าเหมือนน้อย อยู่ในระดับต่ำ คะแนนสูง เพราะส่งผลกระทบมาก)
FLEX	การวัดระดับความยืดหยุ่นในการบริหารจัดการและดำเนินโครงการ (เช่น การใช้เครื่องมือ)
RESL	การวัดระดับความสามารถในการจัดการหรือควบคุมความเสี่ยงขององค์กรหรือทีมงานของโครงการ
TEAM	การวัดระดับของการทำงานเป็นทีมขององค์กรหรือทีมงานโครงการ
PMAT	การวัดระดับวุฒิภาวะความสามารถขององค์กร หรือทีมงานโครงการ ตั้งแต่ระดับต่ำสุดคือ 1 จนถึงระดับสูงสุดคือ 5

ตัวอย่าง การคำนวณหาค่า Effort ที่มีหน่วยเป็น Man-Months

- ปัจจัยขับทั้ง 5 ข้อ ถูกจัดอันดับอยู่ในระดับต่ำมาก (Very Low) ทั้งหมด และกำหนดให้ขนาดของซอฟต์แวร์ที่นับแบบ FP และแปลงเป็น LoC เท่ากับ 10 Kloc สามารถคำนวณหาแรงงานโดยประมาณ บนพื้นฐานของค่าคงที่ประสิทธิผลในการผลิตที่ระดับ Nominal ดังนี้

ระยะหลังการออกแบบ (Post Architecture Model)

- ใช้ประเมินในระยะหลังการออกแบบ (ให้ค่าประเมินมีความถูกต้องมากขึ้น) เนื่องจากยังมีปัจจัยอื่นอีกที่มีผลกระทบร่วมด้วย ได้แก่
  - Product Factor
  - Platform Factor
  - Personal Factor
  - Project Factor
- เรียกปัจจัยเหล่านี้ว่า *Effort Multiplier (EM)*
- โดยมีสูตรคำนวณ MME (เพื่อปรับค่าใหม่) ดังนี้

$$MME\ (Modified) = MME \times (EM)$$

- โดย
  - EM คือผลคูณของปัจจัยที่ส่งผลให้ค่า Effort เปลี่ยนแปลง (ซึ่งมีทั้งหมด 16 ค่า แบ่งตาม 4 กลุ่มปัจจัย) เท่ากับ EM1 x EM2 x ... x EM16

ตาราง ปัจจัยสำหรับ COCOMO II ในระยะหลังการออกแบบ

กลุ่มปัจจัย	ปัจจัย	รายละเอียด
ผลิตภัณฑ์ซอฟต์แวร์ (Product)	RELY: Software Reliability	ระดับความน่าเชื่อถือและไว้วางใจได้ของซอฟต์แวร์ที่ต้องการ
	DATA: Database Size	ขนาดของฐานข้อมูล
	CPLX: Software Complexity	ระดับความซับซ้อนของซอฟต์แวร์
	RUSE: Required Reusability	ความต้องการในการนำโค้ดไปใช้ซ้ำ
	DOCU: Documentation	ระดับมาตรฐานของเอกสาร
แพลตฟอร์ม (Platform)	TIME: Time Constraint on Execution	ข้อจำกัดด้านเวลาในการรันซอฟต์แวร์
	STOR: Main Storage Constraint	ข้อจำกัดด้านเนื้อที่การเก็บข้อมูล
	PVOL: Platform Volatility	ความถี่ในการเปลี่ยนแพลทฟอร์มหรือระบบปฏิบัติการ
บุคลากร (Personnel)	ACAP: Analyst Capability	ความสามารถของนักวิเคราะห์ระบบ
	PCAP: Programmer Capability	ความสามารถของโปรแกรมเมอร์
	PCON: Personnel Continuity	ความถี่ในการเปลี่ยนแปลงพนักงานหรือทีมงาน
	AEXP: Analyst Experience	ประสบการณ์ของนักวิเคราะห์ระบบ
	PEXP: Programmer Experience	ประสบการณ์ของโปรแกรมเมอร์
	LTEX: Language and Tools Experience	ประสบการณ์ในการใช้ภาษาโปรแกรมมิ่งและเครื่องมือ
โครงการ (Project)	TOOL: Use of Software Tools	การใช้เครื่องมือในการบริหารโครงการ
	SITE: Site Environment	จำนวนของไซต์งาน

ตาราง ระดับการส่งผลกระทบต่อ Effort ของปัจจัยทั้ง 16 ประการ ในระยะหลังการออกแบบ

Factor	Levels and Ratings			
	ต่ำมาก (Very Low)	ต่ำ (Low)	ปานกลาง (Nominal)	สูง (High)
Product Factor				
1. RELY	0.82	0.92	1.00	1.10
2. DATA	0.80	0.90	1.00	1.14
3. CPLX	0.73	0.87	1.00	1.17
4. RUSE	0.85	0.95	1.00	1.07
5. DOCU	0.81	0.91	1.00	1.11
Platform Factors				
1. TIME	-	-	1.00	1.11
2. STOR	-	-	1.00	1.05
3. PVOL	-	-	1.00	1.15
Personnel Factors				
1. ACAP	1.42	1.19	1.00	0.85
2. PCAP	1.34	1.15	1.00	0.88
3. AEXP	1.22	1.10	1.00	0.88
4. PEXP	1.19	1.09	1.00	0.91
5. LTXP	1.20	1.09	1.00	0.91
6. PCON	1.29	1.12	1.00	0.90
Project Factors				
1. TOOL	1.17	1.09	1.00	0.90
2. SITE	1.22	1.09	1.00	093

ตัวอย่าง การคำนวณหาค่า MME ในระยะหลังการออกแบบ

- จากหน้า 48 เป็นการหาค่า MME (Effort) โดยประมาณในระยะก่อนการออกแบบ เมื่อถึงระยะหลังการออกแบบ ต้องนำค่า Effort ที่ได้มาปรับค่าใหม่ตามปัจจัยขับเพิ่มเติม 16 ปัจจัย โดยยังคงให้ขนาดของซอฟต์แวร์มีค่าเท่ากับ 10 KLoC เช่นเดิม แต่ในตัวอย่างนี้ จะคำนวณหา MME ในกรณีที่ 1 ปัจจัยขับทั้ง 16 ประการอยู่ในระดับ “Very Low” ดังนี้

สรุป

- การบริหารโครงการ (Project Management)
  - เป็นการประยุกต์ใช้องค์ความรู้ ทักษะ เครื่องมือและกลไก เพื่อดำเนินกิจกรรมตามความต้องการของโครงการให้บรรลุวัตถุประสงค์ตามที่กำหนดไว้ ซึ่งมีกิจกรรมที่สำคัญ ดังนี้
  - Project Planning ประเมินข้อจำกัดต่างๆ ที่ส่งผลต่อโครงการ
  - Project Scheduling กำหนดระยะเวลาของโครงการและการดำเนินงานทุกกิจกรรม
- การประมาณการราคาซอฟต์แวร์
  - เพื่อให้ทราบถึงงบประมาณหรือเงินทุนที่ต้องใช้ในการบริหารจัดการโครงการ

กิจกรรมท้ายบท

- กำหนดให้เขียนแผนงานโครงการเพื่อพัฒนาระบบการตรวจสอบยานพาหนะที่ผ่านเข้าออก มหาวิทยาลัยฯ โดยประกอบด้วยหัวข้อ ดังนี้
  - วัตถุประสงค์และข้อจำกัด
  - คุณสมบัติและขอบเขตงาน
  - ตารางกิจกรรมการทำงานในรูปแบบ Gantt Chart
  - ประโยชน์ที่คาดว่าจะได้รับ
- จากโค้ดในเอกสารประกอบกิจกรรม ให้คำนวณจำนวน LoC ด้วยวิธีต่าง ๆ ดังนี้
  - Simple Line of Code
  - Physical Lines (LINES)
  - Physicals Line of Code

กิจกรรมท้ายบท (ต่อ)

- กำหนดให้แสดงการหาค่า MME (Effort) โดยประมาณซึ่งกำหนดให้ขนาดของซอฟต์แวร์ที่นับแบบ FP แปลงเป็น LoC มีค่าเท่ากับ 10 KLoC ใน 2 กรณี ดังนี้
  - ในระยะก่อนการออกแบบ ปัจจัยขับทั้ง 5 ข้ออยู่ในระดับ “High” ยกเว้นปัจจัยขับ TEAM อยู่ในระดับ “Nominal” บนพื้นฐานของค่าคงที่ประสิทธิผลในการผลิตที่ระดับ “Nominal”
  - ในระยะหลังการออกแบบ โดยปัจจัยขับทั้ง 16 ประการอยู่ในระดับ “High” ยกเว้นกลุ่มปัจจัยโครงการอยู่ในระดับ “Nominal”

เอกสารอ้างอิง

- กิตติ ภักดีวัฒนะกุล, วิศวกรรมซอฟต์แวร์ (Software Engineering), กรุงเทพฯ: เคทีพี คอมพ์ แอนด์ คอนซัลท์, 2552.
- วิทยา สุคตบวร, วิศวกรรมซอฟต์แวร์เบื้องต้น, กรุงเทพฯ: ซีเอ็ดยูเคชั่น, 2551.
- Kathy Schwalbe, Information Technology Project Management, Eighth Edition, Cengage Learning, 2016.
- Lan Sommerville, Software Engineering Ninth Edition, Pearson Education, Inc., publishing as Addison-Wesley, 2011.