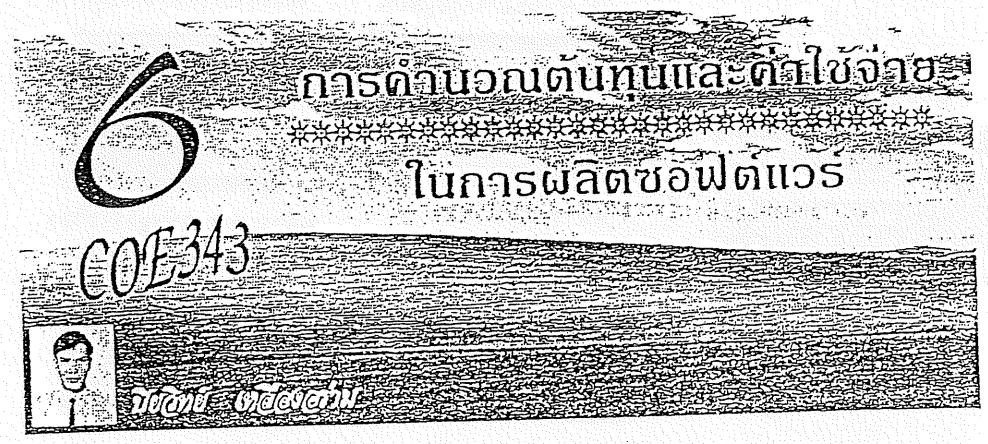


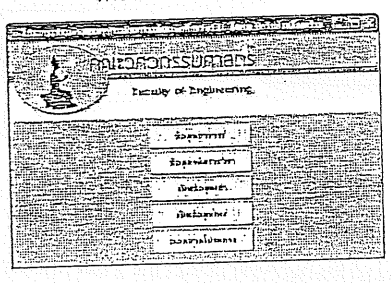
ขั้นตอนแรกเมื่อท่านได้รับการติดต่อมาจากลูกค้า ท่านจะต้องไปพูดคุยเกี่ยวกับลูกค้าว่าระบบหรือซอฟต์แวร์ที่ลูกค้าต้องการมีลักษณะเป็นอย่างไร

ในขั้นตอนนี้ค่อนข้างสำคัญท่านจะต้องซักถามอย่างละเอียดว่าระบบหรือซอฟต์แวร์ที่ลูกค้าต้องการนี้จะนำไปใช้ที่ไหน ใครเป็นผู้ใช้ และระบบงานเดิมเป็นอย่างไรซึ่งในส่วนนี้จะเป็นการเก็บข้อมูลคร่าว ๆ ก่อนในขั้นตอนแรก

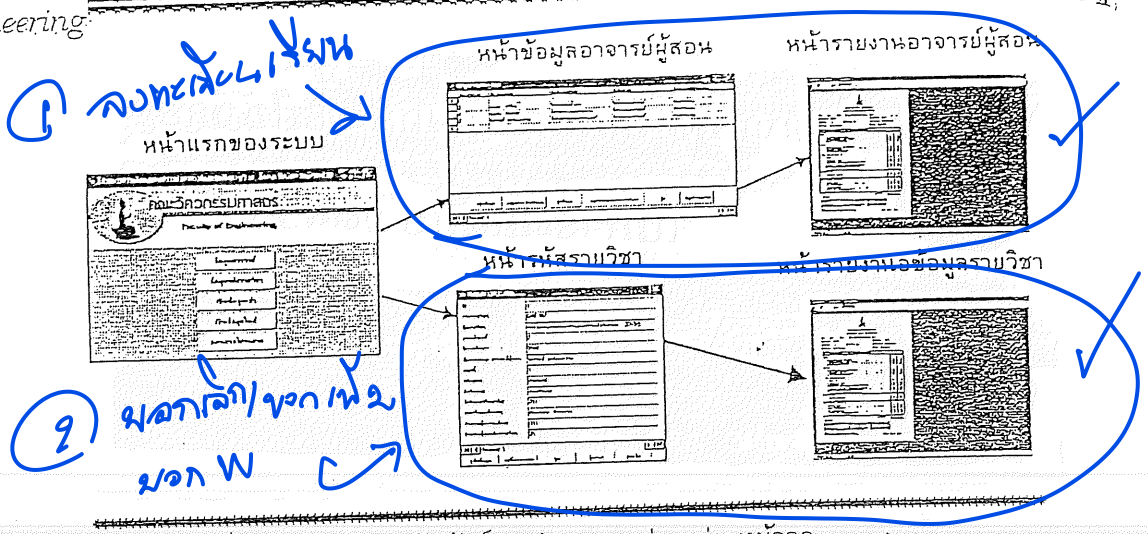
เมื่อท่านเข้าใจแล้วว่าระบบงานหรือซอฟต์แวร์ที่จะพัฒนาขึ้นจะนำไปใช้ในงานอะไร และใครเป็นผู้ใช้ ท่านจะต้องซักถามเกี่ยวกับรายละเอียดของระบบหรือซอฟต์แวร์ที่ลูกค้าต้องการให้ชัดเจน

เมื่อท่านได้ข้อมูลเรียบร้อยแล้วเพื่อให้รายละเอียดที่ได้ชัดเจนขึ้นท่านจะต้องสร้างหน้าจอ หรือ user interface ให้ลูกค้าได้ดูเพื่อให้ลูกค้ามองภาพได้ชัดเจน เป็นการสร้างหน้าจอขึ้นมาอย่างคร่าว ๆ ดังรูปที่ 6-1

หน้าแรกของระบบ



รูปที่ 6-1: แสดงตัวอย่างหน้าจอหรือ user interface ที่ได้ออกแบบขึ้นมา



รูปที่ 6-2: แสดงความสัมพันธ์การทำงานระหว่างแต่ละหน้าจอ

จากรูปที่ 6-2 เมื่อท่านทำการร่างหน้าจอที่ใช้ในการทำงานในแต่ละหน้าจอออกมาท่านจะต้องแสดงความสัมพันธ์ระหว่างแต่ละหน้าจอ เพื่อเชื่อมโยงความเข้าใจของลูกค้า บางส่วนลูกค้าอาจจะมองข้ามไปแต่เมื่อเห็นตัวอย่างหน้าจอที่ท่านออกแบบไป ลูกค้าอาจจะนึกถึงส่วนที่ขาดหายไปได้

เมื่อท่านได้ข้อมูลส่วนใหญ่มาแล้วในขั้นตอนต่อไปท่านจะต้องประเมินค่าใช้จ่ายที่ท่านต้องใช้ในการพัฒนาระบบหรือซอฟต์แวร์ให้กับลูกค้า เพื่อนำเสนอให้กับลูกค้าในการตัดสินใจว่าจะจ้างบริษัทท่านหรือไม่

การประมาณการต้นทุนในการสร้างระบบหรือซอฟต์แวร์

การประมาณต้นทุนในการสร้างระบบหรือซอฟต์แวร์ตามที่ลูกค้าต้องการให้ท่านดำเนินการตามขั้นตอนต่อไปนี้

ขั้นตอนที่ 1 (Step 1) การคำนวณจำนวนโค้ดที่เขียนขึ้น

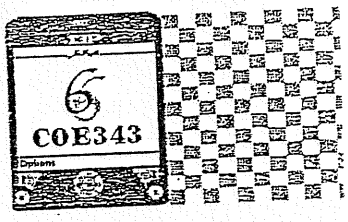
ก่อนอื่นท่านจะต้องทราบก่อนว่าปริมาณโค้ดที่ท่านต้องเขียนขึ้นมีจำนวนกี่บรรทัดหรือศัพท์ทางคอมพิวเตอร์เรียกว่า Line of Code (LOC)

ในปัจจุบันนี้เราจะมองเป็นฟังก์ชันในการทำงาน เช่น ร้านค้าพาณิชย์อิเล็กทรอนิก ท่านอาจมอง

ว่าประกอบด้วยฟังก์ชันดังนี้

- ฟังก์ชันการผลิตสินค้า
- ฟังก์ชันการขายสินค้าหน้าร้าน
- ฟังก์ชันการรับการสั่งซื้อ
- ฟังก์ชันการหักบัตรเครดิต
- ฟังก์ชันการคำนวณค่าจัดส่ง ภาษี
- ฟังก์ชันบริหารจัดการหลังร้าน
- ฟังก์ชันการติดตามลูกค้า
- ฟังก์ชันส่งเสริมการขาย
- ฟังก์ชันการส่งสินค้า

ท่านจะพบว่าระบบงานของเราสามารถที่จะแบ่งออกเป็นฟังก์ชันย่อย ๆ ได้มากมายขึ้นกับการแบ่งของผู้สร้างระบบ เราเรียกการแบ่งออกเป็นฟังก์ชันย่อย ๆ แบบนี้ว่า Function Point



ในการคำนวณจำนวนบรรทัดในการเขียนโค้ด (line of code) หรือ LOC ท่านจะต้องดำเนินการตามขั้นตอนดังนี้

ขั้นตอนที่ 1 (การแจกแจงฟังก์ชันย่อย)

ท่านจะต้องจำแนกแจกแจงออกมาให้ได้ว่าระบบงานของท่านประกอบไปด้วยฟังก์ชันการทำงานย่อย ๆ อะไรบ้าง ดังตัวอย่างต่อไปนี้

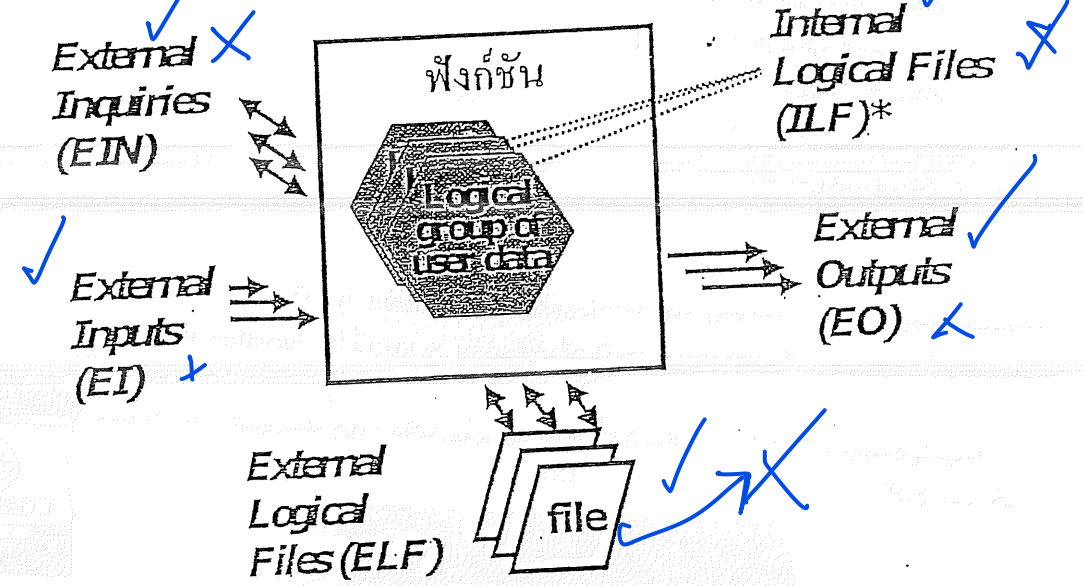
ถ้าท่านกำลังดำเนินการสร้างวิดีโอเกมส์ ท่านแบ่งฟังก์ชันการทำงานออกเป็น 2 ฟังก์ชันคือ

1. ฟังก์ชันการกำหนดตัวละครที่ผู้เล่นเลือกที่จะใช้เล่นในเกมส์
2. ฟังก์ชันการปะทะหรือการเจอกันระหว่างตัวละครภายในเกมส์

ท่านจพต้องทำการแบ่งระบบใหญ่ของท่านออกเป็นระบบย่อย ๆ หรือก็คือฟังก์ชันย่อย ๆ นั่นเอง คล้าย ๆ กับการบริหารงานในบริษัท อาจแบ่งงานตามหน้าที่รับผิดชอบซึ่งก็คือแบ่งตามฟังก์ชันการทำงานนั่นเอง เช่น ประธานกรรมการ รองประธานกรรมการ หัวหน้าแผนก เป็นต้น

ขั้นตอนที่ 2
(การคำนวณในแต่ละฟังก์ชัน)

ในขั้นตอนนี้แต่ละฟังก์ชันจะแบ่งการออกเป็นองค์ประกอบย่อย ๆ ดังรูปที่ 6-2



❖ รูปที่ 6-3: แสดงการแบ่งฟังก์ชันแต่ละฟังก์ชันออกเป็นองค์ประกอบย่อยต่าง ๆ

จากรูปที่ 6-3 เราจะแบ่งฟังก์ชันแต่ละฟังก์ชันออกเป็นองค์ประกอบย่อยต่าง ๆ ดังนี้

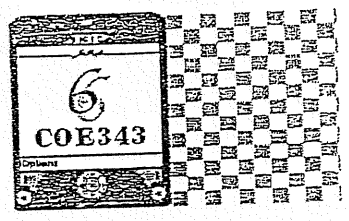
1. หน้าจอที่รับอินพุตจากผู้ใช้ (external input) จะเป็นหน้าจอที่ผู้ใช้ป้อนค่าให้ระบบ
2. หน้าจอแสดงเอาร์พุตให้ผู้ใช้ (external output) จะเป็นหน้าจอที่แสดงผลแก่ผู้ใช้
3. การตอบสนองความต้องการของผู้ใช้ (external inquire) ระบบของท่านสามารถที่จะรองรับการร้องขออะไรบ้างจากผู้ใช้ เช่น ต้องการดูแผนที่ในเกมส์ ต้องการซื้ออาวุธเพิ่ม ต้องการเคลื่อนที่ไปซ้าย-ขวา-หน้า-หลัง เป็นต้น
4. ไฟล์ในระบบที่เกี่ยวข้องกับผู้ใช้ (internal logicalfiles) อาจเป็นไฟล์ที่ถูกสร้างขึ้นหรือเป็นไฟล์ที่มีอยู่แล้วในระบบที่มาเกี่ยวข้องกับผู้ใช้
5. ไฟล์นอกระบบที่เกี่ยวข้องกับผู้ใช้ (external logical files) เป็นไฟล์ที่อยู่นอกระบบที่มาเกี่ยวข้องกับผู้ใช้

ท่านจะต้องแบ่งฟังชันการทำงานแต่ละฟังก์ชันออกองค์ประกอบย่อย ๆ 5 องค์ประกอบแล้ว นำมาแทนค่าในตาราง 6-1 ต่อไปนี้

1 table = 1 Module

Function Point Computations (IFPUG)

PARAMETER	simple		medium		complex	
Ext. inputs EI	1	3	2	4	0	6 = 11
Ext. outputs EO	1	4	1	5	1	7 = 16
Ext. inquiries EIN	0	3	3	4	0	6 = 12
Int. logical files ILF	0	7	1	10	0	15 = 10
Ext. logical files ELF	0	5	0	7	0	10 = 0
					Count Total		49



ตารางที่ 6-1: แสดงตารางที่ใช้คำนวณการทำงานของฟังก์ชันแต่ละฟังก์ชัน

จากตารางที่ 6-1 จะมีการให้น้ำหนักสำหรับซอฟต์แวร์ออกเป็น 3 ระดับ คือ ระดับไม่ซับซ้อนมาก (simple) ระดับซับซ้อนปานกลาง (medium) ระดับซับซ้อนมาก (complex) โดยแต่ละระดับจะมีค่าน้ำหนักไม่เท่ากันดังแสดงในตารางที่ 6-1

	Simple		Medium		Complex		Sub-	Total
	count	factor	count	factor	count	factor	totals	
Ext. inputs	1	3	1	4	1	6	13	
comments:	Name		Ready/move		Qualities			
Ext. outputs	0	4	0	5	0	7	0	
Ext. inquiries	0	3	0	4	0	6	0	25
Int. logical files	1	7	0	10	0	15	7	
comments:	Data about the user's character							
Ext. interface files	1	5	0	7	0	10	5	
comments:	Data about the user's character							

รูปที่ 6-4: แสดงผลการแทนค่าของฟังก์ชันการกำหนดตัวละครที่ผู้เล่นเลือกที่จะใช้เล่นในเกมส์

		Simple		Medium		Complex		Sub-	Total
		count	factor	count	factor	count	factor	totals	
Ext. inputs		0	3	0	4	0	6	0	
Ext. outputs		1	4	0	5	0	7	4	
comments:		Report on results							16
Ext. inquiries		0	3	0	4	0	6	0	
Int. logical files		1	7	0	10	0	15	7	
comments:		Data about the user's character							
Ext. interface files		1	5	0	7	0	10	5	
comments on a		Data about the user's character							

รูปที่ 6-5: แสดงผลการแทนค่าของฟังก์ชันการปะทะหรือการเจอกันระหว่างตัวละครภายในเกมส์

จากรูปที่ 6-4 ถ้าท่านมองซอฟต์แวร์ของท่านว่าถ้าท่านเขียนโค้ดในระดับที่ไม่ซับซ้อนมากนัก ท่านอาจแบ่งองค์ประกอบย่อยออกได้ดังนี้

1. external inputs = 1
2. external outputs = 0
3. external inqueries = 0
4. internal logicals files = 1
5. external logical files = 1

แต่ถ้าท่านจะใช้อัลกอริทึมที่ซับซ้อนขึ้นในระดับกลาง ๆ ท่านจะได้ค่าดังนี้

1. external inputs = 1
2. external outputs = 0
3. external inqueries = 0
4. internal logical files = 0
5. external logical files = 0

ท่านจะเห็นว่าเมื่อท่านใช้อัลกอริทึมที่ซับซ้อนขึ้นท่านอาจไม่จำเป็นที่จะต้องมี internal logical files และ external logical files

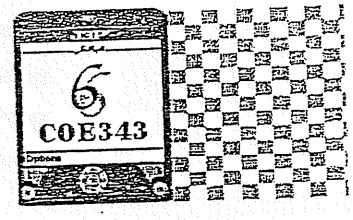
แต่ถ้าท่านจะใช้อัลกอริทึมที่ซับซ้อนมากขึ้นในระดับสูง ท่านจะได้ค่าดังนี้

1. external inputs = 1
2. external outputs = 0
3. external inqueries = 0
4. internal logical files = 0
5. external logical files = 0

ให้ท่านหาผลรวมของการคูณทั้งหมด (count total) ไว้ ซึ่งก็คือ 25 เป็นค่าสำหรับฟังก์ชันการกำหนดตัวละครที่ผู้เล่นเลือกที่จะใช้เล่นในเกมส์เท่านั้น

ส่วนตารางที่ 6-5 สำหรับฟังก์ชันการปะทะหรือการเจอกันระหว่างตัวละครภายในเกมส์ก็เช่นเดียวกัน

จากรูปที่ 6-5 ถ้าท่านมองซอฟต์แวร์ของท่านว่าถ้าท่านเขียนโค้ดในระดับที่ไม่ซับซ้อนมากนัก ท่านอาจแบ่งองค์ประกอบย่อยออกได้ดังนี้



1. external inputs = 0
2. external outputs = 1
3. external inqueries = 0
4. internal logicals files = 1
5. external logical files = 1

แต่ถ้าท่านจะใช้อัลกอริทึมที่ซับซ้อนขึ้นในระดับกลาง ๆ ท่านจะได้ค่าดังนี้

1. external inputs = 0
2. external outputs = 0
3. external inqueries = 0
4. internal logical files = 0
5. external logical files = 0

ท่านจะเห็นว่าเมื่อท่านใช้อัลกอริทึมที่ซับซ้อนขึ้นท่านอาจไม่จำเป็นที่จะต้องมี internal logical files และ external logical files

แต่ถ้าท่านจะใช้อัลกอริทึมที่ซับซ้อนมากขึ้นในระดับสูง ท่านจะได้ค่าดังนี้

1. external inputs = 0
2. external outputs = 0
3. external inqueries = 0
4. internal logical files = 0
5. external logical files = 0

ให้ท่านหาผลรวมของการคูณทั้งหมด (count total) ไว้ ซึ่งก็คือ 16 เป็นค่าสำหรับฟังก์ชันการปะทะหรือการเจอกันระหว่างตัวละครภายในเกมส์

ผลรวมของทั้งสองฟังก์ชัน = 25+16 = 41 คือค่า Unadjusted function point หรือค่าฟังก์ชันที่ไม่มีการปรับเปลี่ยน

ถ้าในซอฟต์แวร์ของท่านมีหลายฟังก์ชันให้ทำการคำนวณอย่างนี้ทุกฟังก์ชันแล้วนำมารวมกัน อย่างนี้ ถ้าท่านมี 10 ฟังก์ชัน ก็ทำตามตารางอย่างนี้ 10 ตารางแล้วนำผลรวมที่ได้จากทุกตารางมารวมกันก็จะเป็นค่า Unadjusted function point หรือค่าฟังก์ชันที่ไม่มีการปรับเปลี่ยน ของซอฟต์แวร์ของท่าน

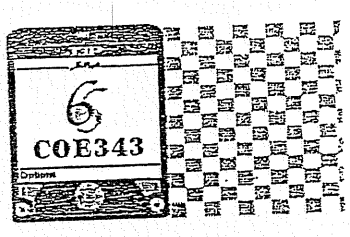
ขั้นตอนที่ 3
(การคำนวณค่าคุณลักษณะทั่วไป)

ในขั้นตอนต่อไปท่านจะต้องคำนวณค่าคุณลักษณะทั่วไปของซอฟต์แวร์ของท่าน ซึ่งได้มาจากปัจจัยที่มีการปรับเปลี่ยนได้ของซอฟต์แวร์ของท่าน

โดยปัจจัยที่มีการปรับเปลี่ยนได้นี้ IEEE ได้กำหนดเอาไว้ 14 ข้อด้วยกันคือ

รายการ	คะแนนที่ให้
1. ซอฟต์แวร์ของท่านต้องการสำรองข้อมูลและกู้ข้อมูล Requires backup/recovery?	.....
2. ซอฟต์แวร์ของท่านต้องการมีการสื่อสารข้อมูลระหว่างกัน Data communications required?	.....

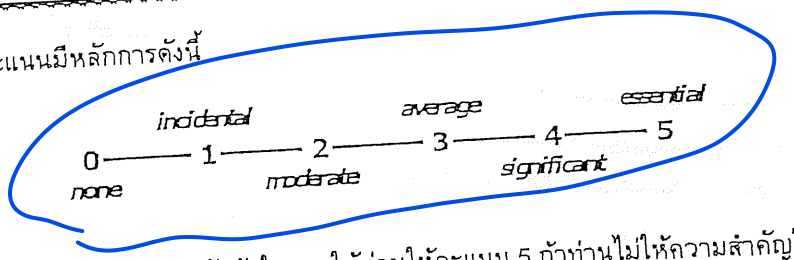
รายการ	คะแนนที่ให้
3. ความต้องการมีฟังก์ชันการประมวลผลแบบกระจาย Distributed processing functions?	.....
4. ความต้องการมีการวิเคราะห์ศักยภาพในการทำงาน Performance critical?	.....
5. ต้องการให้รองรับทำงานในสภาวะที่มีการใช้งานสูง Run on existing heavily utilized environment?	.....
6. ต้องการให้มีการรับส่งข้อมูลแบบออนไลน์ Requires on-line data entry?	.....
7. สามารถเปิดหน้าจออินพุตได้ทีละหลาย ๆ หน้า Multiple screens for input?	.....
8. มีการปรับปรุงข้อมูลในแบบออนไลน์ Master fields updated on-line?	.....
9. อินพุต เอาร์พุต การร้องขอเกี่ยวไฟล์ไฟล์มีความซับซ้อน Inputs, outputs, inquiries of files complex?	.....
10. กระบวนการภายในมรความซับซ้อน Internal processing complex?	.....
11. การเขียนโค้ดถูกออกแบบมาให้สามารถนำกลับมาใช้งานได้อีก Code designed for re-use?	.....
12. ซอฟต์แวร์สามารถที่จะถูกติดตั้งและแก้ไขคุณลักษณะต่าง ๆ ได้ Conversion and installation included?	.....
13. สามารถนำไปใช้งานในหลาย ๆ หน่วยงาน Multiple installation in different orgs.?	.....
14. รองรับการเปลี่ยนแปลงในปัจจัยเกื้อหนุนและง่ายในการใช้งาน Must facilitate change & ease-of-use by user?	.....
ผลรวมค่าคุณลักษณะทั่วไปของซอฟต์แวร์ total general characteristics	.....



จากข้อพิจารณา 14 ข้อนี้ เป็นปัจจัยที่ถูกกำหนดโดย IEEE เพื่อใช้ในการพิจารณาในการคำนวณต้นทุนในการผลิตซอฟต์แวร์

ท่านสามารถที่หาข้อมูลเพิ่มเติมได้จาก IEEE 1983 : General Characteristics for Function Point : Adjust Factor 1-14

การให้คะแนนมีหลักการดังนี้



ถ้าท่านให้ความสำคัญในปัจจัยข้อใดมากให้ท่านให้คะแนน 5 ถ้าท่านไม่ให้ความสำคัญในปัจจัยนั้นเลยให้ท่านให้คะแนน 0 แต่ถ้าท่านบอกว่าปัจจัยนั้นมีความสำคัญที่ไม่อาจกำหนดเป็นตัวเลขตัวเดียวได้ให้ท่านตอบเป็นช่วง เช่น 3-5 เป็นต้น

จากตัวอย่างที่ได้ยกมาตั้งแต่ต้นแต่ซอฟต์แวร์วิดีโอเกมส์ที่จะสร้างขึ้นท่านให้คะแนนดังนี้

รายการ	คะแนนที่ให้
1. ซอฟต์แวร์ของท่านต้องการสำรองข้อมูลและกู้ข้อมูล Requires backup/recovery?	0-2
2. ซอฟต์แวร์ของท่านต้องการมีการสื่อสารข้อมูลระหว่างกัน Data communications required?	0-1
3. ความต้องการมีฟังก์ชันการประมวลผลแบบกระจาย Distributed processing functions?	0
4. ความต้องการมีการวิเคราะห์ศักยภาพในการทำงาน Performance critical?	3-4
5. ต้องการให้รองรับทำงานในสภาวะที่มีการใช้งานสูง Run on existing heavily utilized environment?	0-1
6. ต้องการให้มีการรับส่งข้อมูลแบบออนไลน์ Requires on-line data entry?	5
7. สามารถเปิดหน้าจออินพุตได้ทีละหลาย ๆ หน้า Multiple screens for input?	4-5
8. มีการปรับปรุงข้อมูลในแบบออนไลน์ Master fields updated on-line?	3-4
9. อินพุต เอาร์พุต การร้องขอเกี่ยวไฟล์ไฟล์มีความซับซ้อน Inputs, outputs, inquiries of files complex?	1-2
10. กระบวนการภายในมรความซับซ้อน Internal processing complex?	1-3
11. การเขียนโค้ดถูกออกแบบมาให้สามารถนำกลับมาใช้งานได้อีก Code designed for re-use?	2-4

12. ซอฟต์แวร์สามารถที่จะถูกติดตั้งและแก้ไขคุณลักษณะต่าง ๆ ได้ 0-2
Conversion and installation included?

13. สามารถนำไปใช้งานในหลาย ๆ หน่วยงาน 1-3
Multiple installation in different orgs.?

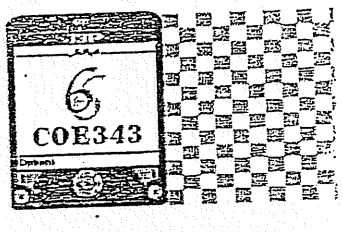
14. รองรับการเปลี่ยนแปลงในปัจจัยเกื้อหนุนและง่ายในการใช้งาน 4-5
Must facilitate change & ease-of-use by user?

ผลรวมค่าคุณลักษณะทั่วไปของซอฟต์แวร์ 24-41
total general characteristics

การที่ท่านมีการให้คะแนนเป็นช่วงเกิดจากการที่ท่านแบ่งโครงสร้างของซอฟต์แวร์ของท่านออกเป็นหลายฟังก์ชัน เช่นในข้อ 13. ที่ให้คะแนน 1-3 ท่านอาจมองว่า

สำหรับฟังก์ชันการกำหนดตัวละครที่ผู้เล่นเลือกที่จะใช้เล่นในเกมส์

13. สามารถนำไปใช้งานในหลาย ๆ หน่วยงาน 1
Multiple installation in different orgs.?



สำหรับฟังก์ชันการปะทะหรือการเจอกันระหว่างตัวละครภายในเกมส์ ท่านให้คะแนน

13. สามารถนำไปใช้งานในหลาย ๆ หน่วยงาน 3
Multiple installation in different orgs.?

เพราะฉะนั้นค่าที่ได้อกมาจึงเป็น

13. สามารถนำไปใช้งานในหลาย ๆ หน่วยงาน 1-3
Multiple installation in different orgs.?

แต่ถ้าซอฟแวร์ของท่านมีฟังก์ชันเดียวผลที่ได้จะออกมามีค่าเดียวดังตัวอย่างต่อไปนี้

1. ซอฟต์แวร์ของท่านต้องการสำรองข้อมูลและกู้ข้อมูล 4
Requires backup/recovery?

2. ซอฟต์แวร์ของท่านต้องการมีการสื่อสารข้อมูลระหว่างกัน 0
Data communications required?

3. ความต้องการมีฟังก์ชันการประมวลผลแบบกระจาย Distributed processing functions?	0
4. ความต้องการมีการวิเคราะห์ศักยภาพในการทำงาน Performance critical?	3
5. ต้องการให้รองรับทำงานในสภาวะที่มีการใช้งานสูง Run on existing heavily utilized environment?	1
6. ต้องการให้มีการรับส่งข้อมูลแบบออนไลน์ Requires on-line data entry?	5
7. สามารถเปิดหน้าจออินพุตได้ทีละหลาย ๆ หน้า Multiple screens for input?	3
8. มีการปรับปรุงข้อมูลในแบบออนไลน์ Master fields updated on-line?	5
9. อินพุต เอาร์พุต การร้องขอเกี่ยวไฟล์ไฟล์มีความซับซ้อน Inputs, outputs, inquiries of files complex?	2
10. กระบวนการภายในมรความซับซ้อน Internal processing complex?	1
11. การเขียนโค้ดถูกออกแบบมาให้สามารถนำกลับมาใช้งานได้อีก Code designed for re-use?	3
12. ซอฟต์แวร์สามารถที่จะถูกติดตั้งและแก้ไขคุณลักษณะต่าง ๆ ได้ Conversion and installation included?	3
13. สามารถนำไปใช้งานในหลาย ๆ หน่วยงาน Multiple installation in different orgs.?	3
14. รองรับการเปลี่ยนแปลงในปัจจัยเกื้อหนุนและง่ายในการใช้งาน Must facilitate change & ease-of-use by user?	2
ผลรวมค่าคุณลักษณะทั่วไปของซอฟต์แวร์ total general characteristics	35

หรับตัวอย่างนี้แสดงให้เห็นในกรณีที่ท่านมองว่าซอฟต์แวร์ของท่านมีหน้าที่การทำงานเดียว หรือมีแค่ฟังก์ชันเดียวเท่านั้น

เมื่อท่านได้ค่า ผลรวมค่าคุณลักษณะทั่วไปของซอฟต์แวร์ หรือ ค่า total general characteristics ออกมา ท่านจะต้องนำค่านี้และค่า unadjusted function points มาคำนวณค่า FP(function point) ซึ่งมีสูตรการคำรวณดังนี้

FP(function point) = [unadjusted function points] x [0.65 + 0.01 x (total general characteristics)]

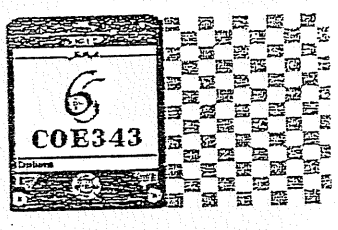
จากการคำนวณก่อนหน้านี้ที่ได้มา

unadjusted function points = 41

total general characteristics = 24 - 41

FP(function point) = [41] x [0.65 + (0.01 x (24 ถึง 41))] = 36 ถึง 43 หน่วยเป็น FP

ภาษา	LOC / FP (ค่าเฉลี่ย)
assembly	320
C	128
Cobol	105
Fortran	105
Pascal	90
Ada	70
OOP	30
4GLs	20
Visual C++	34
Visual Basic	29
Delphi	29
Java	53
Foxpro 2.5	34
C++	30



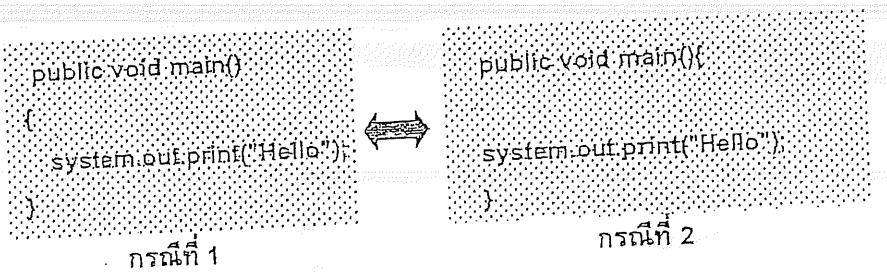
ตารางที่ 6-2: แสดงตารางเปรียบเทียบของ LOC / FP

จากผลการคำนวณที่ได้ 36 ถึง 43 FP ถ้าท่านใช้ภาษา Java ในการเขียนท่านจะต้องเขียนโค้ด 53 LOC / FP หรือ 53 บรรทัด / FP

ดังนั้นท่านจะต้องเขียนโค้ด 36x53 ถึง 43x53 = 1,908 ถึง 2,279 บรรทัด(LOC)

จากที่ท่านได้ศึกษามาเป็นวิธีการนับ LOC(line of code) ที่ใช้การนับฟังก์ชันการทำงานแทน แทนที่จะไปนับทีละบรรทัดเหมือนก่อน เนื่องจากการนับทีละบรรทัดเหมือนก่อนมีปัญหามาก เช่น

1. ในบรรทัดที่เป็นคำอธิบายโปรแกรมท่านจะนับด้วยหรือไม่
2. บรรทัดที่ว่าง หรือเว้นไว้ท่านจะนับหรือไม่
3. ชื่อตัวแปรที่บางภาษาจะต้องประกาศเป็นหน้า ๆ เช่น Cobol ท่านจะนับด้วยหรือไม่
4. ใน 1 บรรทัดมีมากกว่า 1 คำสั่ง ท่านจะนับเป็นหลายบรรทัดหรือบรรทัดเดียว
5. สไตล์ในการเขียนโค้ด เช่น ภาษา C/C++ Java



กรณีที่ 1 กรณีที่ 2

จากตัวอย่างกรณีที่ 1 กับ 2 ต่างกันที่การเขียน " { " ที่กรณีแรกนำไปไว้บรรทัดบน แต่อีกกรณีตบลงมาอีกบรรทัดหนึ่ง ถ้าเป็นแบบนี้จะนับจำนวนบรรทัดกันอย่างไร

6. ถ้าใช้ภาษาพวก Visual จะนับโค้ดกันอย่างไรในเมื่อซอฟต์แวร์ที่เป็นทูลได้ช่วยสร้างหรือเขียนโค้ดให้ท่าน
7. ถ้าใช้โค้ดที่เป็น reuse โค้ดหรือโค้ดที่เขียนครั้งเดียว แล้วก๊อปปี้ไปใช้ในหลาย ๆ ที่ในโปรแกรมท่านจะนับเป็นหลาย ๆ ชุดหรือนับแค่ชุดเดียว

จากปัญหาที่ได้ยกมาจะพบว่ามีปัญหามากมายในการนับบรรทัดในการเขียนโค้ด(LOC) ดังนั้นจึงใช้การนับโดยมองจากฟังก์ชันในซอฟต์แวร์ที่ท่านสร้างขึ้นแทน

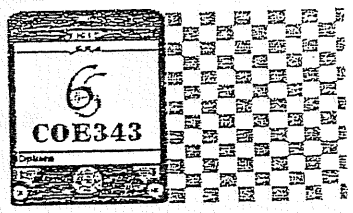
โดยจะมองว่าในระบบของท่านมีฟังก์ชันอะไรบ้างกี่ฟังก์ชัน ไม่นับจำนวนบรรทัดว่ามีกี่บรรทัด แล้วมาคิดเป็นปริมาณงาน แล้วมาเทียบกับตารางเปรียบเทียบของ LOC / FP ท่านก็สามารถที่จะคำนวณหาจำนวนบรรทัดในการเขียนโค้ด (LOC) ออกมาได้

เมื่อท่านคำนวณจำนวนบรรทัดในการเขียนโค้ดออกมาได้ในขั้นตอนต่อไปท่านจะนำจำนวนบรรทัดในการเขียนโค้ดที่ได้ไปคำนวณปริมาณแรงงานที่ท่านต้องการต่อไป

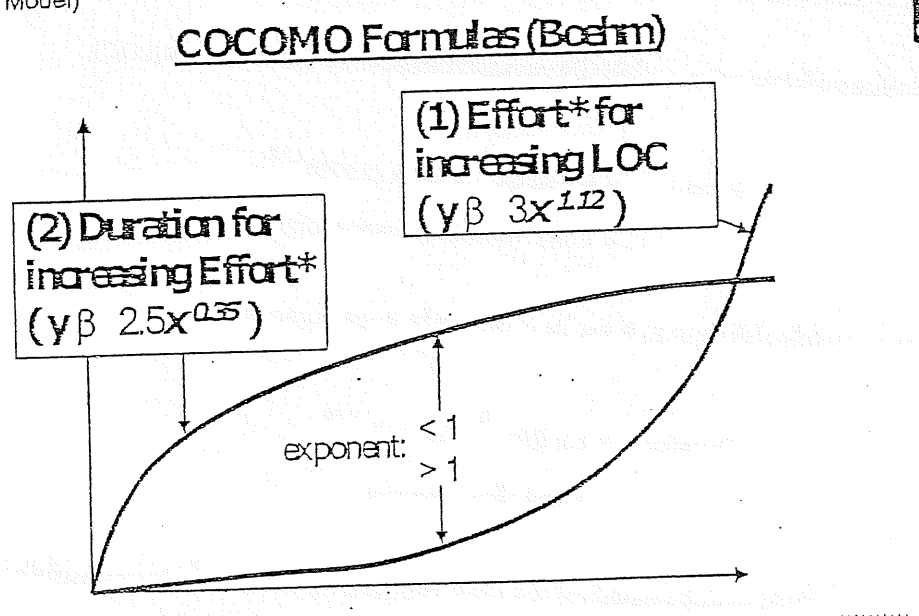
ขั้นตอนที่ 2 (Step II) การคำนวณแรงงานและระยะเวลาที่ใช้ในการผลิตซอฟต์แวร์

ก่อนที่ท่านจะดำเนินการต่อไปให้ท่าน ในการแบ่งซอฟแวร์ในขั้นตอนนี้จะแบ่งออกเป็น 3 กลุ่ม หรือ 3 ประเภท คือ

1. Organic เป็นซอฟต์แวร์หรือแอพพลิเคชันที่มีการทำงานแบบ stand alone คือทำงานบนเครื่อง ๆ เดียวไม่ยุ่งเกี่ยวกับใคร
2. Embedded เป็นซอฟต์แวร์หรือแอพพลิเคชันที่ถูกเขียนลงบนชิปเพื่อให้มันทำงานบนอุปกรณ์ต่าง ๆ เช่น ซอฟต์แวร์ที่ใช้ควบคุมการจ่ายน้ำมันของหัวฉีดในรถยนต์ หรือที่ท่านพบกันมากก็คือซอฟแวร์ที่เขียนฝังตัวอยู่ในอุปกรณ์ต่าง ๆ ที่ใช้ในโรงงานอุตสาหกรรม หุ่นยนต์
3. Semidetached เป็นซอฟต์แวร์หรือแอพพลิเคชันที่ไม่อยู่ใน Organic และ Embedded เช่น แอพพลิเคชันบนเว็บ



โดยสมการที่ใช้คำนวณแรงงานที่ต้องใช้ในการผลิตซอฟต์แวร์ (effort) และ ระยะเวลาในการผลิตซอฟแวร์ (duration) ได้ถูกคิดขึ้นโดย Boehm เรียกสมการนี้ว่า COCOMO (Constructive Cost Model)



รูปที่ 6-6: แสดงกราฟของสมการ cocomo ที่ใช้ในการคำนวณ

Basic COCOMO Formulae (Boehm)

$$\text{Effort in Person-months} = a \times KLOC^{b}$$

$$\text{Duration} = c \times Effort^{d}$$

Software Project	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semidetached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Due to Boehm [Bo]

รูปที่ 6-7: แสดงสมการ cocomo และค่าคงที่ที่ใช้ในการคำนวณ

สมมติกรณีที่เราพัฒนาวิดีโอเกมส์เรามองว่ามันเป็นโปแกรมในลักษณะ semidetached ที่รันบนเว็บ เราจะใช้ค่าคงที่ a = 3.0 , b = 1.12 , c = 2.5 , d = 0.35

จากขั้นตอนที่ผ่านมาคำนวณได้ว่ามีจำนวนบรรทัดของคำสั่ง 2,279 บรรทัด(LOC)

Effort $= a \times KLOC^{b} = 3.0 \times (2.279)^{1.12}$

$= 7.547$ คน-เดือน(person-month)

จำนวนคนที่ต้องใช้ประมาณ 8 คน ใน 1 เดือน หรือ 8 person-month

Duration $= c \times Effort^{d} = 2.5 \times (8)^{0.38}$

$= 5.509$ เดือน (months)

ระยะเวลาที่ใช้ในการผลิตซอฟต์แวร์ 5.5 เดือน โดยประมาณ เพราะฉะนั้นจำนวนคนที่ต้องใช้ในการพัฒนาซอฟแวร์ = Effort/Duration = 8/5.5 = 1.45 คน หรือ 2 คน

ถ้าเงินเดือนนักพัฒนาเดือนละ = 20,000 บาท ท่านจะต้องเสียค่าใช้จ่ายในส่วนเงินเดือน สำหรับนักพัฒนา 20,000 x 5.5 = 110,000 บาท โดยประมาณ

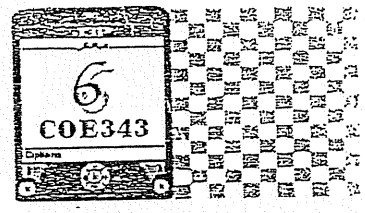
220,000 บ. ในระยะเวลา 5.5 เดือน

ค่าใช้จ่ายที่คำนวณได้นี้ยังไม่ได้คูณด้วยตัวปรับค่า ใช้เป็นการประเมินคร่าว ๆ

Computing COCOMO Case Study Models

		a	K	b		approx.
Effort						aK^b
	LO	2.4	4.2	1.05		10
	HI	2.4	300	1.05		1000
		c	P	d		approx.
Duration						cP^d
	LO	2.5	10	0.38		6
	HI	2.5	1000	0.38		35

รูปที่ 6-8: แสดงสมการตัวอย่างของ COCOMO กรณีโค้ดมีขนาด 4-300 KLOC



จากรูปที่ 6-8 แสดงการคำนวณโดยใช้สมการ COCOMO คำนวณจำนวนคนที่ต้องใช้ และระยะเวลาในการพัฒนาซอฟต์แวร์ สำหรับตัวอย่างในรูปที่ 6-8 เป็นตัวอย่างในกรณีที่โค้ดมีขนาดอยู่ระหว่าง 4,000-30,0000 บรรทัด หรือ 4-300 KLOC