บทที่ 1

บทนำ

ความเป็นมาของปัญหา

ในปัจจุบันมีการพัฒนาระบบงานคอมพิวเตอร์เพื่อช่วยงานด้านต่างๆอย่างมากทั้งทางภาครัฐบาลและเอกชน ซึ่งช่วยทำให้การปฏิบัติงานดำเนินไปอย่างมีประสิทธิภาพ แต่เมื่อพิจารณาถึงค่าใช้จ่ายต่างๆที่ลงทุนไป เป็นที่น่าสังเกตุว่าค่าใช้จ่ายทางด้านฮาร์ดแวร์จะมีแนวโน้มถูกลง แต่ค่าใช้จ่ายทางด้านซอฟต์แวร์ยังคงสูงอยู่เช่นเดิม

ในปี พศ. 2523 ประเทศสหรัฐอเมริกาได้ทำการรวบรวมค่าใช้จ่ายที่ได้ใช้ไปในการพัฒนาระบบคอมพิวเตอร์รวมทั้งสิ้นประมาณ 57 ล้านดอลล่าร์ ปรากฏว่าค่าใช้จ่ายถึง 56 เปอร์เซนต์ของยอดรวมเป็นค่าใช้จ่ายทางด้านซอฟต์แวร์ จากสถิตินี้เองจึงได้มีการตื่นตัวหาหนทางแก้ไขเพื่อลดค่าใช้จ่ายทางด้านซอฟต์แวร์ เพื่อลดต้นทุนการผลิตผลิตภัณฑ์ซอฟต์แวร์ ได้มีการเก็บรวบรวมข้อมูล และนำเอาเทคนิคทั้งทางคณิตศาสตร์ สถิติศาสตร์ เพื่อวิเคราะห์ข้อมูลโดยการกำหนดแบบจำลองการประมาณการค่าใช้จ่ายทางด้านซอฟต์แวร์ (Software Cost Estimate Models) เพื่อช่วยจัดการทรัพยากรการผลิต(resource)ให้เหมาะสม รวมทั้งวัดความก้าวหน้าของการพัฒนาในแต่ละระยะ(phase)ของวัฏจักรชีวิต(Life-cycle) ของผลิตภัณฑ์ซอฟต์แวร์ เพื่อควบคุมมิให้เกิดความล่าช้าในการผลิตผลิตภัณฑ์ซอฟต์แวร์ ทำให้ต้นทุนการผลิตลดต่ำลง

แนวความคิดที่ใช้ในการประมาณค่าใช้จ่ายซอฟต์แวร์มีด้วยกันหลายวิธี เช่น

1. แบบจำลองอัลกอริทมิค(Algorithmic Models) เป็นวิธีที่นำเอาอัลกอริทึมมาประมาณค่าใช้จ่ายทางด้านซอฟต์แวร์ ซึ่งขึ้นกับค่าของกลุ่มตัวแปรที่วิจัยแล้วพบว่ามีผลกระทบต่อการพัฒนาซอฟต์แวร์ ตัวแปรเหล่านี้เรียกว่า ตัวขับค่าใช้จ่าย(cost driver)

2. เอ็กเปริดจัสเม้นท์(Expert Judgement) เป็นวิธีที่นำกลุ่มของผู้เชี่ยวชาญในการพัฒนาซอฟต์แวร์มาประชุมปรึกษาหารือ เพื่อตกลงและตัดสินใจประมาณการค่าใช้จ่ายทางด้านซอฟต์แวร์

3. อานาโลจี(Analogy) เป็นเทคนิคที่ประมาณการค่าใช้จ่ายโดยใช้วิธีเปรียบเทียบกับโครงงานเก่าที่คล้ายๆกัน

4. พากินสัน(Parkinson) เป็นการประมาณการค่าใช้จ่ายทางด้านซอฟต์แวร์โดยดูจากทรัพยากรการผลิตในการพัฒนาระบบ

5. ไพลส์ทูวิน(Price-to-win) เป็นการประมาณการค่าใช้จ่ายทางด้านซอฟต์แวร์โดยคิดค่าใช้จ่ายให้สูง เพื่อให้แน่ใจว่าโครงงานนั้นๆสำเร็จได้

6. ท็อปดาว(Top-down)เป็นเทคนิคในการประมาณการค่าใช้จ่ายทางด้านซอฟต์แวร์ซึ่งประมาณค่าใช้จ่ายทั้งระบบก่อนแล้วจึงแบ่งการประมาณการค่าใช้จ่ายทางด้านซอฟต์แวร์ออกเป็นองค์ประกอบ(component)ย่อยๆ

7. บ็อตทอมอัฟ(Bottom-up) เป็นเทคนิคที่ตรงข้ามกับท็อปดาว โดยจะประมาณการค่าใช้จ่ายทางด้านซอฟต์แวร์ในแต่ละองค์ประกอบก่อนแล้วจึงนำมารวมกันเพื่อประมาณการค่าใช้จ่ายทางด้านซอฟต์แวร์ทั้งระบบ

การประมาณค่าใช้จ่ายทางด้านซอฟต์แวร์นิยมใช้แนวความคิดตามแบบจำลองอัลกอริทมิคเพื่อประมาณค่าใช้จ่ายทางด้านซอฟต์แวร์ เนื่องจากเป็นแนวความคิดที่มีหลักการที่ดีมีการนำเอารายละเอียดของโครงงานเก่ามาวิเคราะห์ เป็นสูตรในการประมาณค่าใช้จ่ายทางด้านซอฟต์แวร์รวมทั้งพิจารณาถึงสภาพแวดล้อมในการพัฒนาซอฟต์แวร์ ซึ่งมีผลกระทบต่อต่อการประมาณค่าใช้จ่ายทางด้านซอฟต์แวร์ด้วย แบบจำลองซึ่งใช้แนวความคิดตามแบบจำลองอัลกอริทมิคที่ได้รับความสนใจมาก เช่น

1. แบบจำลองพุทนัมสลิม (THE PUTNUM SLIM MODEL)

เป็นแบบจำลองที่เกิดขึ้นในปี พศ.2520 โดย แอล เฮท พุทนัม (L.H. Putnum) เป็นการประมาณการค่าใช้จ่ายทางด้านซอฟต์แวร์บนพื้นฐานการวิเคราะห์ของฟังก์ชั่นนอเดิน-รีไลท์ (Norden/Rayleigh) ซึ่งเป็นเทคนิคที่สมบูรณ์ที่สุดสำหรับประมาณการเริ่มต้นจากระยะแรก ซึ่งเหมาะสำหรับโครงงานที่มีขนาดใหญ่มากๆ

รูปแบบของแบบจำลองในการประมาณการความพยายาม (effort) ที่ใช้ใน SLIM (Software Life Cycle Model) ตามสมการที่ 1.1

$$S_s = C_k K^{1/3} t^{4/3} \quad \text{.....1.1}$$

โดยที่ S_s คือ จำนวนบรรทัดคำสั่งที่สามารถแปลเป็นภาษาเครื่องได้

K คือ ความพยายาม คน-ปี (man-years)

t คือ เวลาในการพัฒนา (ปี)

C_k คือ ค่าคงที่

SLIM เป็นเทคนิคซึ่งประมาณการ การแจกแจงกำลังคน การวัดแผนการจัดกำหนดการ(milestone schedules) ระดับความเชื่อถือได้(reliability level) เวลาคอมพิวเตอร์(computer time) ค่าใช้จ่ายด้านจัดทำเอกสารและวัสดุต่างๆ

2. แบบจำลองดอตี้ (THE DOTY MODEL)

เป็นแบบจำลองการประมาณการค่าใช้จ่ายซอฟต์แวร์เกิดในปี พศ. 2520 โดยพัฒนาสำหรับศูนย์พัฒนาโรมแอร์ (Rome Air Developement Centre) ในมลรัฐนิวยอร์ก โปรแกรมประยุกต์ที่ใช้ประมาณการครอบคลุม 4 ชนิดคือ คำสั่งและการควบคุม วิทยาศาสตร์ ธุรกิจ อัตถประโยชน์ ใช้ขนาด(size)เป็นข้อมูลในการคำนวณค่าใช้จ่ายซึ่งกลั่นกรองโดยตัวคูณ (Multipliers) 14 ตัวซึ่งเป็นที่ยอมรับในสภาพแวดล้อมของการพัฒนาซอฟต์แวร์

แบบจำลองของการประยุกต์ โดยทั่วไป ตามสมการ 1.2 และ 1.3

$$MM = 5.288(KDSI)^{1.047} \quad \text{สำหรับ KDSI >= 10} \quad \text{..... 1.2}$$

$$MM = 2.060(KDSI)^{1.047} \left(\Sigma f_j\right) \quad \text{สำหรับ KDSI < 10} \quad \text{..... 1.3}$$

โดยที่ KDSI คือ จำนวน 1000 บรรทัดคำสั่งที่สามารถแปลเป็นภาษาเครื่องได้

f_j คือ ตัวคูณความพยายาม 14 ลักษณะ

3. แบบจำลองอาร์ซีเอ ไพรซ์ เอส (THE RCA PRICE S MODEL)

เป็นแบบจำลองการประมาณการซึ่งถูกพัฒนาขึ้นโดย ไพรส์ ซิสเต็ม ดิวิชั่น ออฟ อาร์ซีเอ(PRICE System Devision of RCA) มลรัฐนิวเจอซี่ ใช้ขนาด ชนิดของการประยุกต์ และความยากง่ายของโครงงาน เป็นข้อมูลในประมาณการค่าใช้จ่ายทางด้านซอฟต์แวร์

4. แบบจำลองคอนสตั๊กทีพ คอส (THE CONSTRUCTIVE COST MODEL)

เป็นแบบจำลองที่เกิดขึ้นในปี พศ.2523 โดย บี ดับบลิว โบม(B.W. Boehm) แบ่งการประมาณค่าใช้จ่ายทางด้านซอฟต์แวร์ออกเป็น 3 แบบคือ เบสิก (Basic) อินเตอร์มีเดท (Intermediate) และ ดีเทล (Detail) โดยใช้ขนาดและค่าของสภาพแวดล้อม 15 ลักษณะ ที่วิจัยแล้วพบว่ามีผลกระทบกับการพัฒนาซอฟต์แวร์ของระบบ นำมาประมาณการค่าใช้จ่ายทางด้านซอฟต์แวร์ จัดกำหนดการ(Schedule)

แบบจำลองต่างๆที่เกิดขึ้นมาเพื่อใช้ในการประมาณการค่าใช้จ่ายซอฟต์แวร์นั้นมีมากมาย ซึ่งขึ้นอยู่กับการวิเคราะห์วิจัย ซึ่งในการวิจัยของ ซิบา เอ็น โมฮันตี้ (Siba N.Mohanty)ได้ ทำการเปรียบเทียบแบบจำลองต่างๆที่ใช้ในการประมาณการในปัจจุบัน 15 แบบจำลองบนพื้นฐาน ของสภาพแวดล้อม 49 ลักษณะแสดง ในตารางที่ 1-1

สามารถสรุปได้ว่าการประมาณการซอฟต์แวร์นั้นขึ้นอยู่กับ

1. สภาพแวดล้อมในการพัฒนาซอฟต์แวร์ 36%
2. จำนวนบรรทัดคำสั่ง 20%
3. ความซับซ้อนของผลิตภัณฑ์ซอฟต์แวร์ 19%

ในประเทศไทย ปัจจุบันเริ่มเกิดปัญหาในการพัฒนาทางด้านซอฟต์แวร์เช่นกันเนื่องจาก การประมาณการค่าใช้จ่ายทางด้านซอฟต์แวร์ขึ้นอยู่กับประสบการณ์ของผู้จัดการโครงงาน ซึ่งอาจ ประมาณการค่าใช้จ่ายต่ำหรือสูงเกินไปไม่มีหลักเกณฑ์แน่นอน ทั้งยังไม่มีวิธีการควบคุมการผลิต ผลิตภัณฑ์ซอฟต์แวร์ได้ อัตราเสี่ยงของความล้มเหลวของโครงงานสูง หรือไม่ก็ใช้เวลาในการ พัฒนานานเกินไป ทำให้การพัฒนาระบบงานคอมพิวเตอร์ในหน่วยงานล้าหลัง ไม่ทันคู่แข่ง ทั้ง ยังก่อให้เกิดค่าใช้จ่ายทางด้านซอฟต์แวร์บานปลาย จากสาเหตุนี้เองจึงได้ทำการวิจัยเพื่อหาวิธี การเพื่อช่วยลดต้นทุนการผลิต โดยช่วยทำแผนงานในการปฏิบัติงาน ช่วยผู้จัดการโครงงาน

สามารถควบคุมการผลิตและวัดผลผลิตของผลิตภัณฑ์ซอฟต์แวร์ในแต่ละระยะตามวัฏจักรในการพัฒนาซอฟต์แวร์ได้ ทั้งยังเพิ่มประสิทธิภาพ ในการผลิตผลิตภัณฑ์ซอฟต์แวร์ให้ดีขึ้นอีกด้วย

ตารางที่ 1-1 แสดงการเปรียบเทียบวิธีการประมาณการโดยแบ่งตามสภาพแวดล้อมต่าง ๆ [1]

Models	Number and Type of Attributes							
	Size (8)	Data Base (1)	Complexity (10)	Type of Program (3)	Documentation (3)	Environment (15)	Other Items (9)	Total Attributes (49)
Farr-Zagorsky	5	1	1	–	2	3	1	13
SDC	2	–	1	1	1	6	1	12
Aron	–	–	3	–	–	1	–	4
NADC	5	1	1	–	2	3	1	13
Putnam	2	–	3	–	–	2	–	7
Wolverton	1	–	3	1	–	4	6	15
GRC	1	–	1	–	–	2	–	4
Tecolote	1	–	–	–	–	1	1	3
ESD	2	1	1	–	–	2	–	6
COCOMO	2	1	7	3	–	12	4	29
Daly	2	–	6	1	1	5	1	16
Aerospace	1	–	1	1	–	1	–	4
Walston-Felix	1	–	–	–	–	–	1	2
Doty	2	–	–	2	–	5	–	9
Kustanowitz	1	1	–	–	–	3	–	5
Price-S	2	–	1	2	–	4	1	10
Total	30	5	29	11	6	54	17	152
%	20	3	19	7	4	36	11	100

[1] Bernard Londeix, Cost Estimatation for Software Development (London, Addison-Wesley Publishers Company, Inc., 1987), p. 41

วัตถุประสงค์ของการวิจัย

1. ผลิตเครื่องมือซอฟต์แวร์เพื่อประมาณการค่าใช้จ่ายซอฟต์แวร์

2. เพื่อวิเคราะห์และปรับค่า ตัวขับค่าใช้จ่าย และ ค่าคงที่ เพื่อความเหมาะสมกับการพัฒนางานในประเทศมากขึ้น

3. เป็นเครื่องมือชี้แนะนักวิเคราะห์ระบบให้สามารถประเมินค่าใช้จ่ายโดยใช้แนวความคิดตามแบบจำลองโคโคโม

ขอบเขตการวิจัย

1. ต้นแบบไทยโคโคโม จะพัฒนาขึ้นมาโดยให้ใช้งานบนเครื่องไมโครคอมพิวเตอร์

2. โครงงานพัฒนาซอฟต์แวร์ที่นำมาวิจัยจะเป็นโครงงานในประเทศที่พัฒนาสิ้นสุดโครงงานแล้ว ในกลุ่มโครงงาน ภาวะออแกนนิค ภาวะเซไมดิเท็ด และภาวะเอ็มเบ็ดเด็ด

3. ต้นแบบสามารถปรับปรุงโครงสร้างแบบจำลองให้เหมาะสมและถูกต้องมากยิ่งขึ้นโดยสามารถพิจารณาจากข้อมูลโครงงานพัฒนาซอฟต์แวร์ที่มีเพิ่มขึ้นต่อมาได้

ขั้นตอนและวิธีดำเนินงานวิจัย

1. ศึกษาแบบจำลองโคโคโม

2. รวบรวมโครงงานพัฒนาซอฟต์แวร์ในประเทศที่พัฒนาสิ้นสุดโครงงานแล้วมาประเมินค่าใช้จ่ายกำลังคน และบริหารทรัพยากรใหม่ดูว่า ตัวขับค่าใช้จ่ายชุดใดที่ให้ผลตรงกับโครงงานลักษณะนั้น เมื่อประมาณค่าใช้จ่ายโครงแบบจำลองโคโคโม

3. พัฒนาซอฟต์แวร์เพื่อใช้ในการประมาณการค่าใช้จ่าย

4. ทดสอบซอฟต์แวร์ เพื่อปรับพิจารณาปรับค่าในแบบจำลองให้เหมาะสม

5. ประเมินผล

ผลประโยชน์ที่คาดว่าจะได้รับ

1. สามารถนำซอฟต์แวร์ที่พัฒนาขึ้นนำไปใช้ประโยชน์ได้
2. ช่วยให้ผู้จัดการโครงการสามารถประมาณค่าใช้จ่ายในการผลิตซอฟต์แวร์
3. สามารถควบคุมการผลิต และวัดผลผลิต
4. ช่วยทำแผนงานในการปฏิบัติงาน
5. สามารถวัดประสิทธิภาพบุคลากร
6. ลดต้นทุนการผลิต
7. สามารถตรวจปรับความต้องการ ในด้านบุคคลากร อุปกรณ์การผลิต หรือ การนำเอาเทคโนโลยีใหม่ๆมาใช้ เพื่อเพิ่มประสิทธิภาพในการผลิต
8. สามารถเก็บรวบรวมสถิติ การประมาณค่าใช้จ่ายของโครงการพัฒนาซอฟต์แวร์ที่ผ่านมา เพื่อเป็นประโยชน์ในการชี้แนะโครงการถัดไป