บทที่ 1

บทนำ

ความเป็นมาและความสำคัญของปัญหา

ในการดำเนินงานโครงการต่าง ๆ การจัดการโครงการนับเป็นสิ่งสำคัญที่จะช่วยควบคุมโครงการให้ดำเนินงานได้เสร็จสิ้นตามวัตถุประสงค์ และเวลาที่กำหนด การจัดการโครงการทางวิศวกรรมนั้น โดยมากจะมีวิธีการวัด ค่าที่วัดได้จะมีความชัดเจนที่จะช่วยให้ทราบถึงคุณภาพ และความสามารถในการทำงานของค่าความพยายามที่ลงไป เพื่อนำไปพัฒนาคุณภาพของผลิตภัณฑ์ และจัดการบริหารโครงการให้มีประสิทธิภาพยิ่งขึ้นการวัดจึงนับได้ว่า มีความสำคัญอย่างยิ่งต่อการจัดการโครงการทางวิศวกรรมต่างๆ

ในโครงการพัฒนาซอฟต์แวร์นั้น วิธีการวัดยังเป็นที่ถกเถียงถึงวิธีการวัดที่เหมาะสมกับการดำเนินงานโครงการและการวัดผลิตภัณฑ์ วิธีวัดแบบใด ที่จะมีความชัดเจนที่จะใช้ในการเปรียบเทียบความสามารถของบุคคลากร วิธีดำเนินงาน และตัวผลิตภัณฑ์

สำหรับโครงการพัฒนาซอฟต์แวร์ในประเทศไทยนั้น การจัดการโครงการในหน่วยงานส่วนใหญ่ยังไม่ได้มีการนำวิธีการวัดซอฟต์แวร์มาใช้กันอย่างจริงจัง อันเนื่องมาจากการขาดวิธีการวัดที่เหมาะสมและทันต่อเทคโนโลยีที่เปลี่ยนไป การจัดการโครงการ จึงมักจะขึ้นอยู่กับประสบการณ์ของผู้จัดการโครงการ ทำให้ขาดกฎเกณฑ์ที่แน่นอน ในการวัดเพื่อควบคุม และวางแผนการผลิตซอฟต์แวร์ อัตราเสี่ยงต่อความล้มเหลวของโครงการสูง หรือทำให้เกิดความล่าช้าของโครงการ ทำให้ผลิตภัณฑ์ซอฟต์แวร์ที่ได้ไม่ทันต่อความต้องการใช้งาน ธุรกิจมีความล้าหลังและไม่ทันต่อการเปลี่ยนแปลงของสังคม ซึ่งก่อให้เกิดความเสียหายต่อธุรกิจโดยรวมของบริษัท

นอกจากนี้ เทคโนโลยีทางด้านคอมพิวเตอร์ มีความเจริญก้าวหน้าและเปลี่ยนแปลงไปอย่างมากทั้งด้านฮาร์ดแวร์และซอฟต์แวร์ โดยเฉพาะซอฟต์แวร์นั้น แต่เดิมมีการทำงานเป็นไปตามลำดับของโปรแกรม และพัฒนาโดยการเขียนโปรแกรมเป็นบรรทัดของคำสั่ง (Line of Code) แต่ในปัจจุบัน ซอฟต์แวร์สมัยใหม่ มักมีการทำงานในลักษณะของการขึ้นอยู่กับเหตุการณ์ (Event driven software) และการพัฒนาซอฟต์แวร์ มักใช้เครื่องมือช่วยพัฒนาในลักษณะของการประกอบชิ้นส่วนซอฟต์แวร์ในรูปของวัตถุ (Object) ขึ้นเป็นซอฟต์แวร์ระบบงาน ทำให้วิธีการวัดซอฟต์แวร์โดยวัด

จำนวนบรรทัดของคำสั่งของซอฟต์แวร์ที่เป็นที่ยอมรับในอดีต ไม่สามารถใช้ได้กับซอฟต์แวร์สมัยใหม่ได้อย่างสมบูรณ์

การวิจัยนี้ เพื่อพัฒนามาตรวัดซอฟต์แวร์ที่ใช้กับสภาพแวดล้อมในประเทศไทย โดยที่ไม่คำนึงถึงภาษาคอมพิวเตอร์ และเครื่องมือที่ใช้ในการพัฒนาซอฟต์แวร์ ทั้งนี้ เพื่อให้ผู้จัดการโครงการสามารถใช้เป็นกฎเกณฑ์ในการวัดซอฟต์แวร์ ทำให้การจัดการและบริหารโครงการเป็นไปอย่างมีหลักเกณฑ์ และมีประสิทธิภาพยิ่งขึ้น

แนวความคิดและทฤษฎีที่เกี่ยวข้อง

จากแนวความคิดที่จะออกแบบและพัฒนาวิธีการวัดซอฟต์แวร์ ที่ไม่คำนึงถึงภาษาที่ใช้พัฒนา เพื่อใช้ในโครงการพัฒนาซอฟต์แวร์ จึงต้องอาศัยทฤษฎีและแนวความคิดต่างๆ ที่กล่าวถึงเทคนิควิธีการวัดซอฟต์แวร์ต่างๆ กระบวนการพัฒนาซอฟต์แวร์ และทฤษฎีอื่นที่เกี่ยวข้องกับปัจจัยที่มีผลต่อการวัดซอฟต์แวร์ดังนี้ คือ

1. มาตรวัดซอฟต์แวร์ (Software Metrics)

ในการทำงานด้านวิศวกรรมส่วนใหญ่ จะมีวิธีการวัดค่าต่างๆ เพื่อช่วยให้เราเข้าใจกระบวนการทางเทคนิคที่ใช้ในการพัฒนาผลิตภัณฑ์ รวมทั้งเข้าใจในตัวผลิตภัณฑ์เองด้วย ทำให้เราสามารถจัดการคุณภาพและประสิทธิภาพของผลิตภัณฑ์ที่ผลิตได้อย่างชัดเจน

การวัดได้มีการใช้ในงานวิศวกรรมด้านต่างๆ อยู่ทั่วไป เช่น มีการวัดความสิ้นเปลืองของพลังงาน การวัดน้ำหนัก การวัดขนาดความกว้างและความยาว การวัดอุณหภูมิ การวัดแรงดันไฟฟ้า เป็นต้น แต่ในส่วนของงานทางด้านวิศวกรรมซอฟต์แวร์ วิธีการวัดในงานวิศวกรรมซอฟต์แวร์กลับเป็นเรื่องที่ซับซ้อน ยากที่จะหาวิธีการวัดที่เหมาะสม และเป็นที่ยอมรับโดยรวมได้ วัตถุประสงค์ที่ต้องมีวิธีการวัดซอฟต์แวร์ ก็คือ

1. เพื่อชี้ถึงคุณภาพของซอฟต์แวร์
2. เพื่อประเมินความสามารถพัฒนาของพนักงานที่ใช้ผลิตซอฟต์แวร์
3. เพื่อประเมินความสามารถผลิตและคุณภาพที่ได้จากกระบวนการทางวิศวกรรมซอฟต์แวร์และเครื่องมือที่ใช้
4. เพื่อเตรียมรูปแบบที่ใช้เป็นพื้นฐานในการประมาณราคาซอฟต์แวร์

5. เพื่อให้เห็นถึงความต้องการในการปรับ หรือเปลี่ยนเครื่องมือในการพัฒนา และเพิ่มการอบรมแก่บุคคลากรที่ใช้ในโครงการพัฒนาซอฟต์แวร์

วิธีการวัดซอฟต์แวร์โดยทั่วไป สามารถแบ่งออกเป็นสองลักษณะ คือ การวัดโดยตรงและการวัดโดยอ้อม การวัดโดยตรงนั้น ในด้านของกระบวนการทางวิศวกรรมซอฟต์แวร์ จะประกอบด้วย การวัดต้นทุนและความพยายามที่ใช้ในกระบวนการ ส่วนในด้านของตัวผลิตภัณฑ์ซอฟต์แวร์ การวัดจะประกอบด้วย การวัดจำนวนบรรทัดของคำสั่งที่ผลิตขึ้น ความเร็วที่ใช้ในการประมวลผล ขนาดของหน่วยความจำ และปริมาณรายงานที่เกี่ยวกับความบกพร่องของซอฟต์แวร์

ส่วนการวัดโดยอ้อมของซอฟต์แวร์ จะประกอบด้วย การวัดปริมาณฟังก์ชันการทำงาน คุณภาพ ความซับซ้อน ประสิทธิภาพ ความน่าเชื่อถือและ ความสามารถในการบำรุงรักษาของตัวซอฟต์แวร์ เป็นต้น

ต้นทุนและความพยายามที่ใช้ดำเนินงานการสร้างซอฟต์แวร์ จำนวนบรรทัดของโปรแกรมที่เขียน และการวัดโดยตรงอื่นๆ สามารถจะรวบรวมและกำหนดขึ้นมาใช้ได้โดยง่าย แต่ในการประเมินคุณภาพและความสามารถของซอฟต์แวร์ หรือประสิทธิภาพและความสามารถในการบำรุงรักษาของซอฟต์แวร์นั้น การประเมินทำได้ยากและทำได้แต่เพียงการวัดโดยทางอ้อมเท่านั้น

มาตรวัดที่ใช้ในการวัดซอฟต์แวร์ที่เป็นที่นิยมกันในปัจจุบัน มีดังนี้

1.1 มาตรวัดเชิงขนาด (Size-Oriented Metrics)

มาตรวัดเชิงขนาด จัดเป็นวิธีการวัดโดยตรงของผลิตภัณฑ์ซอฟต์แวร์และกระบวนการที่ผลิตซอฟต์แวร์ โดยวัดขนาดของซอฟต์แวร์ในรูปของการนับจำนวนบรรทัดคำสั่งของซอฟต์แวร์ ถ้าส่วนประกอบของซอฟต์แวร์มีการแก้ไข จะมีการสร้างตารางข้อมูลเชิงขนาด ดังรูปที่ 1 ในตารางจะแสดงข้อมูลเชิงขนาดของแต่ละโครงการที่ดำเนินการเสร็จสิ้นแล้ว เช่น โครงการ ก: มีขนาด 12.1 พันบรรทัดคำสั่ง(Kilo Line Of Code) มีความพยายามในการพัฒนา 24 เดือน-คน person-month) โดยใช้ต้นทุน 168,000 บาท ค่าความพยายามและต้นทุน ที่บันทึกลงในตารางนี้ พิจารณาจากการดำเนินงานทั้งหมดทุกกระบวนการทางวิศวกรรมซอฟต์แวร์ ซึ่งรวมถึงขั้นตอนการวิเคราะห์ การออกแบบ การพัฒนาคำสั่ง และการทดสอบ นอกจากนี้ ยังมีข้อมูลในโครงการ ก. ที่บ่งชี้ถึงการจัดทำเอกสารอีก 365 หน้า และมีข้อผิดพลาดที่พบในปีแรกที่ได้ส่งมอบซอฟต์แวร์ให้ลูกค้านำไปใช้ปฏิบัติงานนับได้ ทั้งสิ้น 29 ข้อ และ มีการใช้บุคคลากรในการพัฒนาซอฟต์แวร์ของโครงการ ก. ทั้งสิ้น 3 คน

โครงการ	ความพยายาม (เดือน-คน)	ต้นทุน (พันบาท)	บรรทัดคำสั่ง (พันบรรทัด)	เอกสาร (หน้า)	จำนวน ข้อผิดพลาด	บุคคลากร (คน)
ก	24	168	12.1	365	29	3
ข	62	440	27.2	1,224	86	5
ค	43	258	20.2	1,050	64	6
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.

รูปที่ 1 การวัดซอฟต์แวร์เชิงขนาด

จากข้อมูลเบื้องต้นที่เก็บในตารางดังกล่าว สามารถที่จะพัฒนาวิธีการวัดความสามารถในการพัฒนาและคุณภาพของซอฟต์แวร์อย่างง่ายในแต่ละโครงการ โดยคำนวณค่าเฉลี่ยจากตาราง คือ

ความสามารถพัฒนาซอฟต์แวร์ = พันบรรทัดคำสั่ง / เดือน-คน

คุณภาพของซอฟต์แวร์ = จำนวนข้อผิดพลาด / พันบรรทัดคำสั่ง

นอกจากนี้ การวัดอย่างอื่นที่น่าสนใจก็สามารถคำนวณได้ เช่น

ต้นทุน = ต้นทุนโครงการ / พันบรรทัดคำสั่ง

การผลิตเอกสาร = จำนวนหน้าของเอกสาร / พันบรรทัดคำสั่ง

มาตรวัดแบบเชิงขนาดยังไม่เป็นที่ยอมรับโดยทั่วไป เนื่องจากมีการใช้จำนวนบรรทัดคำสั่งเป็นหลักในการวัด ซึ่งจะขึ้นอยู่กับภาษาคอมพิวเตอร์ที่ใช้ในการพัฒนา ดังนั้น ถ้ามีการออกแบบที่ดี ทำให้พัฒนาโปรแกรมได้สั้นจะมีความเสียเปรียบกว่า และวิธีการวัดเชิงขนาดนี้ยังยากที่จะปรับไปใช้กับภาษาที่ไม่เป็นขั้นตอน (Nonprocedural Language) นอกจากนี้ ในการประมาณค่าซอฟต์แวร์ด้วยวิธีนี้ จะต้องทราบว่าผู้ผลิตจะใช้ภาษาคอมพิวเตอร์ใดก่อน ซึ่งเป็นการยากที่จะตัดสินใจในช่วงเริ่มโครงการ เนื่องจากผู้ออกแบบอาจจะต้องใช้เวลาประมาณปริมาณบรรทัดของคำสั่งที่จะผลิตนาน ทำให้การวิเคราะห์และออกแบบใช้เวลานานกว่าจะเสร็จสิ้น

1.2 มาตรวัดเชิงฟังก์ชัน (Function-Oriented Metrics)

มาตรวัดเชิงฟังก์ชัน เป็นการวัดโดยอ้อมของซอฟต์แวร์และกระบวนการผลิตซอฟต์แวร์ ตรงกันข้ามกับการนับจำนวนบรรทัดของคำสั่ง (LOC) มาตรวัดเชิงฟังก์ชันจะมีการพิจารณาที่ฟังก์ชันและอรรถประโยชน์ของโปรแกรม มาตรวัดเชิงฟังก์ชันได้ถูกเสนอขึ้นมาครั้งแรกโดยนายอัลเลน เจ. อัลเบรชท์ (Allan J. Albrecht) ซึ่งได้แนะนำวิธีการวัดความสามารถในการผลิตซอฟต์แวร์ ที่เรียกว่า ฟังก์ชันพอยต์ (Function Point) วิธีการของฟังก์ชันพอยต์ เกิดจากการสังเกตวิธีการวัดที่นับโดเมนสารสนเทศต่างๆ ของซอฟต์แวร์ และการประเมินความซับซ้อนของซอฟต์แวร์

ฟังก์ชันพอยต์มีการคำนวณดังตารางรูปที่ 2 จะมีการพิจารณาคุณลักษณะของโดเมนสารสนเทศของซอฟต์แวร์ เพื่อใช้เป็นพารามิเตอร์ในการวัด ทำการนับจำนวนพารามิเตอร์เหล่านี้ และบันทึกลงในตาราง โดยโดเมนสารสนเทศที่ใช้เป็นพารามิเตอร์ในการวัด ประกอบด้วย

- จำนวนอินพุทของผู้ใช้
- จำนวนเอาท์พุทของผู้ใช้
- จำนวนการสอบถามของผู้ใช้
- จำนวนแฟ้มข้อมูล
- จำนวนการอินเตอร์เฟสกับภายนอก

		ปัจจัยถ่วง (Weighting Factor)			
พารามิเตอร์ในการวัด	นับได้		ปกติ	เฉลี่ย	ซับซ้อน	FP
อินพุทของผู้ใช้		x	3	4	6	
เอาท์พุทของผู้ใช้		x	4	5	7	
แฟ้มข้อมูล		x	7	10	15	
การอินเตอร์เฟสกับภายนอก		x	5	7	10	
การสอบถามของผู้ใช้		x	3	4	6	
จำนวน FP ทั้งสิ้น						

รูปที่ 2 การคำนวณการวัดแบบฟังก์ชันพอยต์ [1]

[1] Roger S. Pressman, Software Engineering. 3rd ed., (Singapore: McGraw-Hill, Inc., 1992), p. 49

หอสมุดกลาง สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

เมื่อมีการพิจารณาและนับพารามิเตอร์ต่าง ๆ ของซอฟต์แวร์ดังกล่าวข้างต้นแล้ว ค่าที่นับได้จะมีความสัมพันธ์กับความซับซ้อน ซึ่งในโครงสร้างของฟังก์ชันพอยต์ได้พัฒนาเกณฑ์ในการตัดสินใจเลือกระดับความซับซ้อนออกเป็นระดับง่าย ระดับเฉลี่ย และระดับซับซ้อน การคำนวณค่าของฟังก์ชันพอยต์ จะคำนวณได้ตามสูตรความสัมพันธ์ คือ

$$\text{ฟังก์ชันพอยต์ (FP)} = \text{จำนวน FP รวม} * [\ 0.65 + 0.01 * \text{SUM}(F_i)\ ]$$

โดยที่ จำนวน FP รวม เป็นผลรวมของจำนวน FP ที่นับได้ในแต่ละพารามิเตอรที่ใช้ในการวัดจากตาราง ในรูปที่ 2

F_i (i=1 ถึง 14) เป็นค่าที่ใช้ปรับตามความซับซ้อน(Complexity Adjustment Value) ซึ่งขึ้นอยู่กับการพิจารณาเลือกระดับอัตราถ่วงของแต่ละคำถามในรูปที่ 3

ค่าคงที่และค่าปัจจัยถ่วงในสมการข้างต้น เป็นค่าที่ได้มาจากการวิจัย[2]

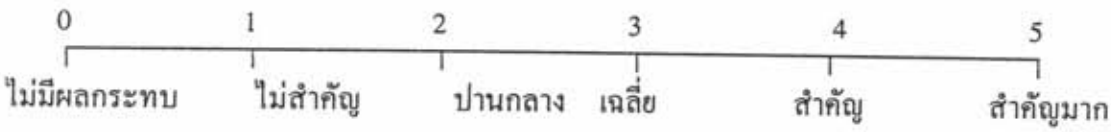
เมื่อคำนวณได้ค่าฟังก์ชันพอยต์แล้ว การนำไปใช้ ก็เช่นเดียวกันกับวิธี LOC ในการวัดความสามารถพัฒนาซอฟต์แวร์ คุณภาพของซอฟต์แวร์ และคุณสมบัติอื่นๆของอฟต์แวร์ อาทิ

ความสามารถพัฒนาซอฟต์แวร์	=	จำนวนฟังก์ชันพอยต์ / เดือน-คน
คุณภาพซอฟต์แวร์	=	จำนวนข้อผิดพลาด / จำนวนฟังก์ชันพอยต์
ต้นทุน	=	ต้นทุนโครงการ / ฟังก์ชันพอยต์
เอกสาร	=	จำนวนหน้าของเอกสาร / ฟังก์ชันพอยต์

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

[2] Roger S. Pressman, Software Engineering. 3rd ed., (Singapore: McGraw-Hill, Inc., 1992), p. 49

ระดับอัตราถ่วงของปัจจัย 0 ถึง 5



F_i

1. ความต้องการในการสำรองข้อมูลและการกู้คืนสภาพ (Backup and Recovery)?
2. ความต้องการในการสื่อสารข้อมูล?
3. ฟังก์ชันการประมวลผลแบบกระจาย?
4. การวิเคราะห์ถึงประสิทธิภาพของระบบ?
5. ระบบจะมีการวิ่งภายใต้สิ่งแวดล้อมที่มีการปฏิบัติงานเป็นจำนวนมาก?
6. ความต้องการในการป้อนข้อมูลแบบออนไลน์?
7. ในการป้อนข้อมูลแบบออนไลน์ข้อมูลที่ได้เกิดจากหลายการปฏิบัติงานหรือหลายจอภาพ?
8. ความต้องการปรับปรุงข้อมูลหลักแบบออนไลน์?
9. ความซับซ้อนของอินพุท เอาท์พุท แฟ้มข้อมูล หรือ การสอบถามข้อมูล?
10. ความซับซ้อนของการประมวลผลภายใน?
11. โค้ดโปรแกรมได้ถูกออกแบบสำหรับการนำกลับมาใช้อีก?
12. ในการออกแบบได้ออกแบบรวมถึงการแปลงข้อมูลและการติดตั้งระบบ?
13. ระบบมีการออกแบบเพื่อติดตั้งได้หลายครั้งในหลายองค์กร?
14. การออกแบบแอพลิเคชันเพื่อสะดวกในการแก้ไขและง่ายต่อการใช้งาน?

รูปที่ 3 ปัจจัยถ่วงในการคำนวณฟังก์ชันพอยต์ [3]

[3] Roger S. Pressman, Software Engineering. 3rd ed., (Singapore: McGraw-Hill, Inc., 1992), p. 50

การออกแบบวิธีการวัดแบบเทคนิคฟังก์ชันพอยต์ คิดค้นขึ้นเพื่อประยุกต์ใช้กับระบบสารสนเทศทางธุรกิจ อย่างไรก็ตาม มีการนำเสนอวิธีการเพิ่มเติมโดยนายเคเปอร์ โจนส์ เป็นวิธีที่เรียกว่า ฟีเจอร์พอยต์ (Feature Point) วิธีการวัดนี้ สามารถประยุกต์ใช้ได้กับระบบงานแอพลิเคชันและวิศวกรรมซอฟต์แวร์ การวัดแบบฟีเจอร์พอยต์ จะเหมาะกับแอพลิเคชันที่เป็นลักษณะอัลกอริทึมที่ซับซ้อนมาก เช่น แอพลิเคชันที่มีการทำงานแบบเรียลไทม์ แอพลิเคชันที่ควบคุมกระบวนการต่างๆ หรือ/และ แอพลิเคชันซอฟต์แวร์ที่ฝังการทำงานภายใต้ซอฟต์แวร์อื่น(Embedded software application) ซึ่งมีอัลกอริทึมที่มีความซับซ้อนสูง

ในการคำนวณแบบฟีเจอร์พอยต์ โดเมนสารสนเทศ (Information domain) จะถูกนับและถ่วงน้ำหนักค่าเช่นเดียวกับวิธีการของฟังก์ชันพอยต์ตามที่ได้กล่าวไปก่อนหน้าแล้ว โดยมีเพิ่มการพิจารณานับคุณลักษณะของซอฟต์แวร์อีกคุณลักษณะหนึ่ง คือ การนับอัลกอริทึม อัลกอริทึมจะรวมถึงกรรมวิธีในการจัดการงานต่างๆ ในโปรแกรมคอมพิวเตอร์ เช่น การทำอินเวอร์สเมตริกซ์ การถอดรหัส หรือการตรวจสอบการอินเตอร์รัพท์ เป็นต้น

การคำนวณฟีเจอร์พอยต์ ใช้ตารางในรูปที่ 4 ซึ่งมีการถ่วงน้ำหนักค่าพารามิเตอร์แต่ละตัว และค่าฟีเจอร์พอยต์ จะคำนวณโดยใช้สมการ ดังนี้

$$\text{ฟีเจอร์พอยต์} = \text{จำนวน FP รวม} * [0.65 + 0.01 * \text{SUM}(F_i)]$$

โดยที่ จำนวน FP รวม เป็นผลรวมของจำนวน FP ที่นับได้ในแต่ละพารามิเตอร์ที่ใช้ ในการวัด จากตาราง ในรูปที่ 4

F_i (i=1 ถึง 14) เป็นค่าที่ใช้ปรับตามความซับซ้อน ซึ่งขึ้นอยู่กับการพิจารณาเลือกระดับความสำคัญของแต่ละคำถาม ในรูปที่ 3 ข้างต้น

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

พารามิเตอร์ในการวัด	นับได้		อัตราถ่วง		FP
อินพุทของผู้ใช้		X	4	=	
เอาท์พุทของผู้ใช้		X	5	=	
การสอบถามข้อมูลของผู้ใช้		X	4	=	
แฟ้มข้อมูล		X	7	=	
การอินเตอร์เฟสกับภายนอก		X	7	=	
อัลกอริทึม		X	3	=	
จำนวน FP ทั้งสิ้น					

รูปที่ 4 การคำนวณฟีเจอร์พอยต์ [4]

จะเห็นว่าฟีเจอร์พอยต์ และฟังก์ชันพอยต์ แสดงถึงคุณลักษณะของซอฟต์แวร์แบบเดียวกัน คือ ฟังก์ชันและอรรถประโยชน์ของซอฟต์แวร์ ผลลัพธ์ที่ได้จากการวัดทั้งสองแบบจะเป็นค่าฟังก์ชันพอยต์ สำหรับระบบงานเรียลไทม์ที่มีความซับซ้อนมาก การใช้ฟีเจอร์พอยต์มัก จะมีค่าสูงกว่าการใช้ฟังก์ชันพอยต์ ประมาณ 20-35%[1]

การวัดแบบฟังก์ชันพอยต์หรือฟีเจอร์พอยต์ ต่างกับการวัดแบบบรรทัดคำสั่งตรงที่ ฟังก์ชันพอยต์นั้นไม่คำนึงภาษาคอมพิวเตอร์ที่ใช้ โดยแนวความคิดของฟังก์ชันพอยต์เหมาะสำหรับแอพลิเคชันที่ใช้ภาษาแบบคอนเวนชันนัล (Conventional Language) หรือ ภาษาแบบไม่เป็นขั้นตอน(nonprocedure language) โดยการใช้โดเมนสารสนเทศของซอฟต์แวร์เป็นฐานในการวัดแทน ซึ่งทำให้สามารถที่จะทราบขอบเขตโครงการเพื่อจะประเมินโครงการได้เร็ว ทำให้ฟังก์ชันพอยต์เป็นที่นิยมใน การประมาณการโครงการมากขึ้นเรื่อยๆ

---

[4] Roger S. Pressman, Software Engineering. 3rd ed., (Singapore: McGraw-Hill, Inc., 1992), p. 51

2. กระบวนการทางวิศวกรรมซอฟต์แวร์

กระบวนการทางวิศวกรรมซอฟต์แวร์ คือ ขั้นตอนต่างๆ ในการดำเนินโครงการพัฒนาซอฟต์แวร์ มีดังนี้ คือ

2.1 การวิเคราะห์

การรวบรวมความต้องการใช้ซอฟต์แวร์ เพื่อเข้าใจลักษณะของซอฟต์แวร์ที่จะสร้าง โดยทำความเข้าใจเกี่ยวกับขอบเขตของสารสนเทศ ฟังก์ชันที่ต้องการ ประสิทธิภาพ และการอินเตอร์เฟส ความต้องการทั้งในส่วนของระบบและซอฟต์แวร์นี้ จะจัดทำเป็นเอกสาร และทำการสอบทวน (Review) กับลูกค้า

2.2 การออกแบบ

การออกแบบกระบวนการของซอฟต์แวร์โดยมุ่งที่คุณลักษณะทั้งสี่ของโปรแกรม คือ โครงสร้างข้อมูล ลักษณะของซอฟต์แวร์ รายละเอียดของโปรซีเดอร์ และลักษณะการอินเตอร์เฟส ในกระบวนการออกแบบ จะแปลงความต้องการของผู้ใช้ไปเป็นการนำเสนอโดยซอฟต์แวร์ เพื่อที่จะประเมินผลก่อนการลงมือเขียนโปรแกรมจริง

2.3 การเขียนโปรแกรม

ดำเนินการแปลงผลจากการออกแบบ เป็นรูปแบบที่เครื่องสามารถอ่านได้ โดยการเขียนโปรแกรมด้วยภาษาคอมพิวเตอร์ต่างๆ

2.4 การทดสอบ

การนำโปรแกรมที่เขียนขึ้นมา ทำการทดสอบการทำงานของโปรแกรม เพื่อให้โปแกรมที่เขียนขึ้นให้ผลลัพธ์ถูกต้องตรงความต้องการ

2.5 การบำรุงรักษา

โดยปกติซอฟต์แวร์จะมีความสมบูรณ์เมื่อส่งมอบให้ลูกค้า การเปลี่ยนแปลงซอฟต์แวร์จะเกิดขึ้นเมื่อเกิดความผิดพลาดขึ้นในภายหลัง เนื่องมาจากการเปลี่ยนแปลงสภาพแวดล้อมภายนอก อาทิเช่น การเปลี่ยนแปลงฟังก์ชันของผู้ใช้ การเปลี่ยนแปลง ระบบปฏิบัติการ เป็นต้น ซึ่งมักจักต้องทำการแก้ไขซอฟต์แวร์มากกว่าที่จะพัฒนาขึ้นใหม่

ซอฟต์แวร์เป็นผลรวมของโปรแกรมคอมพิวเตอร์ต่างๆ โพรซีเดอร์ กฎข้อบังคับ และเอกสาร และข้อมูลที่เกี่ยวกับการปฏิบัติงานของระบบคอมพิวเตอร์ ซึ่งโปรแกรมและเอกสารที่ถูกพัฒนาขึ้น จะส่งให้ผู้ใช้ในรูปของผลิตภัณฑ์ซอฟต์แวร์

การพัฒนาซอฟต์แวร์เป็นการที่ยากในการกำหนดรูปแบบที่แน่นอน แต่ยังมีผู้ที่มีประสบการณ์ได้มีการพยายามรวบรวมเทคนิคในการทำงาน และปรับเป็นระบบเพื่อใช้ในการพัฒนา ปฏิบัติงาน บำรุงรักษา และปลดซอฟต์แวร์ เรียกว่า วิศวกรรมซอฟต์แวร์

3. ลักษณะของซอฟต์แวร์

ในส่วนของซอฟต์แวร์ระบบงานต่างๆ จะมีแบบฉบับของซอฟต์แวร์ ที่ประกอบด้วยประโยคคำสั่งประเภทต่างๆ ดังนี้

3.1 โมเดลการคำนวณ

ทำการคำนวณและปฏิบัติตามฟังก์ชันการทำงานของซอฟต์แวร์

3.2 อินพุทของผู้ใช้

โต้ตอบกับผู้ใช้เพื่อที่จะเก็บข้อมูลอินพุทของผู้ใช้ ซึ่งอาจเกี่ยวข้องกับความยากง่ายหรือซับซ้อนในการโต้ตอบ เช่น การตรวจสอบความผิดพลาดของข้อมูลที่นำเข้า และการใส่ข้อมูลลงในโครงสร้างข้อมูลที่เตรียมไว้

3.3 เอาท์พุทของผู้ใช้

รูปแบบการพิมพ์หรือการแสดงผลลัพธ์ที่ได้จากการคำนวณ

3.4 การควบคุม

การควบคุมในรูปแบบของการเปรียบเทียบ การทำงานวนรอบ การแยกไปทำงานต่างๆ ตามตรรกของโปรแกรม

3.5 การประมวลผลข้อความแสดงความช่วยเหลือ

แสดงข้อความแนะนำที่เหมาะสมตอบสนองต่อผู้ใช้ เมื่อต้องการความช่วยเหลือ

3.6 การประมวลผลความผิดพลาด

ในกรณีที่เกิดความผิดพลาดในระหว่างการอินพุท เอาท์พุท การคำนวณ การติดต่อสื่อสาร และอื่นๆ อาจตอบสนองโดยแสดงข้อความแสดงความผิดพลาดและกู้คืนสภาพสิ่งที่ผิดพลาดนั้น

3.7 การเคลื่อนย้ายข้อมูล

การเคลื่อนย้ายข้อมูลจากโครงสร้างหนึ่งไปอีกโครงสร้างหนึ่ง หรือจากฐานข้อมูลไปสู่โครงสร้างข้อมูลภายในของโปรแกรม การเรียงลำดับ การค้นหา(Search) และการเปลี่ยนรูปแบบข้อมูลเพื่อเตรียมสำหรับการประมวลผลต่อไป.0

3.9 การประกาศข้อมูล

การประกาศโครงสร้างข้อมูลทั้งหมดที่ใช้โดยแอพลิเคชัน

3.10 หมายเหตุ

แสดงความอธิบาย สาระสำคัญ และหมายเหตุต่างๆ

4. ลักษณะการดำเนินงานของผู้พัฒนา

ในการพัฒนาซอฟต์แวร์นั้น ผู้พัฒนามักจะมีวิธีการดำเนินงานต่างๆ ดังนี้

4.1 การอ่าน

ศึกษาระบบที่จักต้องพัฒนาและเครื่องมือหรือเทคนิคที่สามารถนำมาใช้ได้

4.2 การออกแบบ

ขั้นตอนนี้ เป็นกระบวนการของการกำหนดโครงสร้างของแอพลิเคชันทั้งหมด ส่วนประกอบของแอพลิเคชัน โมดูล อินเตอร์เฟส และโครงสร้างข้อมูล และเอกสารการออกแบบ การออกแบบนี้ ไม่เหมือนการเขียนโปรแกรมหรือการออกแบบโปรแกรม การออกแบบจะเกี่ยวข้องกับการเลือกโครงสร้างข้อมูล อัลกอริทึม ข้อกำหนดการไหลของสารสนเทศ

4.3 การวางแผน

แจกแจงสิ่งที่ต้องดำเนินงานและตารางเวลา เช่น รูปแบบของ WBS (Work Breakdown Structure) ที่ประกอบด้วย แผนงานที่บอกถึงสิ่งที่ต้องทำ ผู้ที่จะทำ และกำหนดเวลาที่งานต้องเสร็จ

4.4 การเขียนโปรแกรม

การติดตั้งอัลกอริทึมและโครงสร้างข้อมูล

4.5 การทำเอกสาร

การรวบรวมและจัดทำเอกสาร

4.6 การทดสอบ

กระบวนการประเมินส่วนประกอบของระบบ เพื่อดำเนินการตรวจสอบความแตกต่างระหว่างความต้องการของผู้ใช้ และผลลัพธ์ที่ได้จากระบบ

4.7 การสอบทวน

ขั้นตอนการสอบทวนขั้นตอนต่างๆ ของโครงการ เพื่อตรวจสอบความถูกต้อง ความสมบูรณ์ ความขัดแย้ง และความชัดเจนของการออกแบบ

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

4.8 การประชุม

เพื่อแสดงข้อคิดเห็นถึงการออกแบบ การวางแผน การเขียนโปรแกรม และการจัดทำเอกสารของระบบที่ได้ทำขึ้น

4.9 การปรับปรุง

การค้นหาข้อผิดพลาดระหว่างการรีวิว การทดสอบ การวิ่ง แอพลิเคชัน

5. ระดับของภาษา (Language Level)

ระดับของภาษาได้ถูกกำหนดขึ้นเพื่อแบ่งภาษาออกเป็นระดับต่าง ๆ กัน ทั้งนี้เนื่องจากในการพัฒนาซอฟต์แวร์ที่มีฟังก์ชันการทำงานเท่ากัน แต่ใช้ภาษาหรือเครื่องมือที่ต่างกัน ก็ทำให้จำนวนประโยคคำสั่งของซอฟต์แวร์แตกต่างกันออกไป ตัวอย่างเช่น ซอฟต์แวร์หนึ่งที่พัฒนาด้วยภาษาโคบอล ต้องใช้ถึง 1,000 ประโยคคำสั่ง ในขณะที่ถ้าใช้ภาษาออบเจ็คซี จะใช้เพียง 250 ประโยคคำสั่งเท่านั้น ดังนั้น ค่าเฉลี่ยของจำนวนประโยคคำสั่งในแต่ละภาษาที่ใช้สามารถใช้เป็นสัดส่วนเพื่อระบุเป็นตัวเลขระดับของภาษาต่างๆ ได้ เมื่อค่าระดับของภาษาสูงขึ้น จำนวนประโยคคำสั่งที่ใช้ต่อหนึ่งฟังก์ชันพอยต์ก็น้อยลงด้วย

ในปี ค.ศ. 1995 นายเคเปอร์ โจนส์ ได้นำเสนอตารางระดับของภาษาที่ใช้พัฒนาซอฟต์แวร์มากกว่า 500 ภาษา ซึ่งส่วนหนึ่งของรายละเอียดแสดงความสัมพันธ์ของจำนวนประโยคคำสั่งกับฟังก์ชันพอยต์ของระดับภาษาคอมพิวเตอร์ต่างๆ ได้แสดงในรูปที่ 5

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

ภาษา	ระดับภาษา	จำนวนประโยคคำสั่งที่ใช้ต่อหนึ่งฟังก์ชันพอยต์
ANSI BASIC	5.00	64
BASIC A	2.50	128
Basic ASSEMBLY	1.00	320
Berkeley PASCAL	3.50	91
C	2.50	128
C ++	6.50	49
CICS	7.00	46
COBOL (default)	3.00	107
COBOL (ANSI 85)	3.50	91
dBase III	8.00	40
dBase IV	9.00	35
DL/1	8.00	40
EASYTRIEVE +	25.00	13
EXCEL	55.00	6
FORTRAN	3.00	107
GW BASIC	3.25	98
IBM VS COBOL	3.50	91
LOTUS 123 DOS	50.00	6
PowerBuilder	20.00	16
PROLOG	5.00	64
QBE	25.00	13
SQL	27.00	12
Visual Basic	10.00	32
Visual COBOL	16.00	20

รูปที่ 5 ตัวอย่างบางส่วนของตารางระดับของภาษา[5]

[5] Capers Jones, Progarmming Language Table, (Software Productivity Research, Inc. 1995).

จากการศึกษาทฤษฎีการวัดซอฟต์แวร์ ซึ่งได้แบ่งการวัดออกเป็น 2 ประเภท คือ การวัดเชิงขนาดและการวัดเชิงฟังก์ชัน การวัดเชิงขนาดนั้นมีการวัดจากจำนวนบรรทัดคำสั่งของซอฟต์แวร์ ซึ่งจะให้ ผลที่แตกต่างกันออกไปในแต่ละภาษาที่ใช้พัฒนา แต่วิธีการวัดเชิงฟังก์ชัน จะวัดจากฟังก์ชันการทำงานของซอฟต์แวร์ จึงเป็นวิธีการวัดที่ไม่ขึ้นกับภาษาที่ใช้พัฒนา ในการวิจัย ได้ใช้วิธีการวัดเชิงฟังก์ชันโดยเทคนิคฟังก์ชันพอยต์ ในการพัฒนาวิธีการวัด เพื่อใช้สำหรับโครงการพัฒนาซอฟต์แวร์ในประเทศไทย แต่การวัดโดยเทคนิคฟังก์ชันพอยต์ ยังมีปัจจัยถ่วงต่างๆ ที่เกี่ยวข้องกับค่าฟังก์ชันพอยต์ที่นับได้ ซึ่งในส่วนของลักษณะของซอฟต์แวร์ และลักษณะการดำเนินงานของผู้พัฒนาที่ได้กล่าวข้างต้น จะใช้เป็นแนวทางในการปรับปัจจัยถ่วงต่างๆ ในการวัดซอฟต์แวร์

ส่วนตารางระดับภาษา จะมีความสัมพันธ์ระหว่างจำนวนบรรทัดคำสั่งต่อฟังก์ชันพอยต์ ซึ่งใช้เป็นประโยชน์ในการเปลี่ยนค่าฟังก์ชันพอยต์ของซอฟต์แวร์ที่วัดได้ เป็นจำนวนบรรทัดคำสั่งในภาษาต่างๆ เพื่อประโยชน์ในการเปรียบเทียบขนาดซอฟต์แวร์เมื่อพัฒนาด้วยภาษาต่างๆ

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

วัตถุประสงค์

1. เพื่อให้เกิดแนวความคิดในการพัฒนามาตรวัดซอฟต์แวร์ในประเทศไทย
2. เพื่อพัฒนาโปรแกรมต้นแบบ ใช้ในการวัดขนาดของซอฟต์แวร์ที่ไม่ขึ้นกับภาษา และประมาณการเวลาและต้นทุนในการพัฒนาซอฟต์แวร์

ขอบเขตการวิจัย

1. พัฒนามาตรวัดซอฟต์แวร์ที่พัฒนาขึ้นในประเทศไทย โดยใช้เทคนิคฟังก์ชันพอยต์ (Function Point) เป็นต้นแบบหลัก หรือผสมผสานกับเทคนิคอื่นเพื่อความเหมาะสม
2. มาตรวัดที่ได้ สามารถใช้วัดขนาดของซอฟต์แวร์โดยไม่ขึ้นกับภาษาที่ใช้พัฒนาหรือขึ้นกับภาษาที่ใช้พัฒนาน้อยที่สุด
3. สามารถวัดความสามารถในการผลิตซอฟต์แวร์ของผู้พัฒนาซอฟต์แวร์
4. ใช้ข้อมูลการพัฒนาซอฟต์แวร์ขนาดเล็กในประเทศไทยที่พัฒนาเสร็จสิ้นแล้ว และพัฒนาขึ้นด้วย เครื่องมือพัฒนาซอฟต์แวร์ คือ Power Builder หรือ Visual Basic หรือ MS Access ตัวใดตัวหนึ่ง เป็นกรณีศึกษาในการวิจัย
5. พัฒนาโปรแกรมต้นแบบ เพื่อใช้วัดซอฟต์แวร์ตามวิธีการวัดที่ได้ออกแบบไว้ ภายใต้ระบบวินโดว์โดยใช้ MS Access หรือเครื่องมืออื่นตามความเหมาะสม

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

ขั้นตอนและวิธีดำเนินการวิจัย

1. ศึกษามาตรวัดซอฟต์แวร์โดยเทคนิคของฟังก์ชันพอยต์ และเทคนิคอื่นๆ เพิ่มเติมตามเหมาะสม
2. ออกแบบและพัฒนาวิธีการวัดซอฟต์แวร์ ซึ่งประกอบด้วยข้อมูลการดำเนินโครงการพัฒนาซอฟต์แวร์ วิธีการวัด และวิธีการคำนวณ
3. ออกแบบและพัฒนาโปรแกรมต้นแบบภายใต้ระบบวินโดว์โดยใช้ MS Access หรือเครื่องมืออื่นตามความเหมาะสม เพื่อใช้ในการทดสอบวิธีการวัดซอฟต์แวร์ที่ได้ออกแบบไว้

4. ออกแบบวิธีการเก็บรวบรวมข้อมูลโครงการพัฒนาซอฟต์แวร์ในประเทศไทย

5. รวบรวมข้อมูลโครงการพัฒนาซอฟต์แวร์ที่พัฒนาเสร็จสิ้นแล้วในประเทศไทย อาทิ ต้นทุน จำนวนบุคคลากรที่ใช้ เวลาที่ใช้ เป็นต้น เฉพาะซอฟต์แวร์ที่พัฒนาโดยใช้เครื่องมือช่วยในการพัฒนา คือ Power Builder หรือ Visual Basic หรือ MS Access

6. ทดสอบวิธีการวัดต้นแบบ โดยใช้ข้อมูลโครงการพัฒนาซอฟต์แวร์ที่รวบรวมมา

7. ปรับปรุงวิธีการวัด โดยการใช้ข้อมูลเดิมทดสอบความผิดพลาดซ้ำ และปรับปรุงวิธีการวัด

8. สรุปผลการวิจัย และข้อเสนอแนะ

ประโยชน์ที่คาดว่าจะได้รับ

1. มีวิธีการวัดซอฟต์แวร์ที่ไม่ขึ้นกับภาษาที่ใช้พัฒนา
2. มีหน่วยของการวัดค่าความพยายามที่เหมาะสม
3. มีวิธีการวัดค่าความสามารถในการผลิตซอฟต์แวร์ของผู้พัฒนาซอฟต์แวร์
4. การจัดการโครงการพัฒนาซอฟต์แวร์ในประเทศไทย มีหลักเกณฑ์แน่นอนและชัดเจน
5. สามารถวางแผนกำลังคน และการประมาณการต้นทุนและเวลาของโครงการได้ถูกต้องยิ่งขึ้น
6. ได้แนวทางและวิธีการในการพัฒนาวิธีการวัดซอฟต์แวร์ประเภทอื่นต่อไป

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย