การวัดซอฟต์แวร์เชิงแง่มุมโดยการวิเคราะห์ฟังก์ชันพอยต์

นางสาวเสาวลักษณ์ รัตนธรรมสกุล



วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2548

ISBN 974-53-2849-9

# MEASURING AOP-BASED SOFTWARE BY FUNCTION POINT ANALYSIS

Miss Saowaluk Ratanatamskul



A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science Program in Computer Science
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2005
ISBN 974-53-2849-9

| | |
|---|---|
| หัวข้อวิทยานิพนธ์ | การวัดซอฟต์แวร์เชิงแง่มุมโดยการวิเคราะห์ฟังก์ชันพอยต์ |
| โดย | นางสาว เสาวลักษณ์ รัตนธรรมสกุล |
| สาขาวิชา | วิทยาศาสตร์คอมพิวเตอร์ |
| อาจารย์ที่ปรึกษา | อาจารย์ ดร. โปรดปราน บุณยพุกกณะ |

---

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์

(ศาสตราจารย์ ดร. ดิเรก ลาวัณย์ศิริ)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ

(ผู้ช่วยศาสตราจารย์ ดร. พรศิริ หมื่นไชยศรี)

..... อาจารย์ที่ปรึกษา

(อาจารย์ ดร. โปรดปราน บุณยพุกกณะ)

..... กรรมการ

(อาจารย์ ดร. ดาริชา สุธีวงศ์)

..... กรรมการ

(อาจารย์ เชษฐ พัฒโนทัย)

นางสาว เสาวลักษณ์ รัตนธรรมสกุล : การวัดซอฟต์แวร์เชิงแง่มุมโดยการวิเคราะห์ฟังก์ชันพอยต์ (MEASURING AOP-BASED SOFTWARE BY FUNCTION POINT ANALYSIS). อาจารย์ที่ปรึกษา : อาจารย์ ดร. โปรดปราน บุณยพุกกณะ, 109 หน้า. ISBN 974-53-2849-9.

การวัดขนาดซอฟต์แวร์เป็นกิจกรรมที่นำไปสู่การประเมินขนาด และค่าใช้จ่ายในการพัฒนาซอฟต์แวร์ หากสามารถประเมินได้อย่างแม่นยำตั้งแต่ระยะเริ่มแรกของโครงงาน จะทำให้การจัดสรรทรัพยากรและการดำเนินงานเป็นไปอย่างมีประสิทธิภาพ การวัดขนาดซอฟต์แวร์โดยใช้เทคนิคการวิเคราะห์ฟังก์ชันพอยต์ (Function Point Analysis) เป็นวิธีการวัดขนาดซอฟต์แวร์วิธีหนึ่งที่มีผู้นิยมใช้มากและมีงานวิจัยทำการทดสอบและพบว่า ไม่ว่าจะทำการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยีแบบใดก็ตามค่าฟังก์ชันพอยต์ที่ได้จะไม่มีการเปลี่ยนแปลง เนื่องจากฟังก์ชันพอยต์เป็นการวัดจำนวนฟังก์ชันที่มีอยู่ในซอฟต์แวร์ โดยใช้ข้อกำหนดความต้องการของซอฟต์แวร์ (Software Requirements Specification) เป็นข้อมูลพื้นฐานในการวิเคราะห์คำนวณขนาดของซอฟต์แวร์

เมื่อการเขียนโปรแกรมเชิงแง่มุม (Aspect-Oriented Programming) ได้ถูกนำเสนอขึ้นมา จุดเด่นประการหนึ่งคือทำให้ขนาดซอฟต์แวร์ลดลง ถึงแม้การเขียนโปรแกรมเชิงแง่มุมจะมีแนวความคิดที่เพิ่มเติมขึ้นมาจากการเขียนโปรแกรมเชิงวัตถุ (Object-Oriented Programming) แต่ข้อกำหนดความต้องการของซอฟต์แวร์ได้แก่แผนภาพ Use Case ในเชิงแง่มุมอาจจะมีความแตกต่างจากแผนภาพ Use Case ในเชิงวัตถุก็ได้ ดังนั้นในวิทยานิพนธ์นี้เป็นการทดสอบค่าฟังก์ชันพอยต์ที่ได้จากการวัดขนาดซอฟต์แวร์ที่พัฒนาด้วยการเขียนโปรแกรมเชิงแง่มุม ว่าสามารถใช้ได้กับการพัฒนาซอฟต์แวร์ภายใต้เทคโนโลยีเชิงแง่มุมได้หรือไม่

จากการวิเคราะห์และทดสอบโดยนำแผนภาพ Use Case ในเชิงวัตถุของระบบงานทางด้านการเงินที่เป็นกรณีศึกษาจำนวน 12 ระบบ มาแปลงเป็นแผนภาพ Use Case ในเชิงแง่มุมด้วยวิธีการของ Jacobson พบว่าค่าฟังก์ชันพอยต์ที่ได้จากการนับจากแผนภาพ Use Case ในเชิงแง่มุมมีค่าสูงกว่าค่าฟังก์ชันพอยต์ที่ได้จากการนับจากแผนภาพ Use Case ในเชิงวัตถุที่ระดับนัยสำคัญทางสถิติ 0.05 ($\alpha$=0.05) นอกจากนี้ในวิทยานิพนธ์ยังทำการทดสอบการวัดขนาดซอฟต์แวร์ด้วยวิธี Use Case Points ก็พบว่าได้ผลเช่นเดียวกัน

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต เสาวลักษณ์ รัตนธรรมสกุล

สาขาวิชา วิทยาศาสตร์คอมพิวเตอร์ ลายมือชื่ออาจารย์ที่ปรึกษา

ปีการศึกษา 2548

## 4671448921 : MAJOR COMPUTER SCIENCE

KEY WORD: ASPECT-ORIENED PROGRAMMING / USE CASE DIAGRAM / FUNCTION POINT ANALYSIS

SAOWALUK RATANATAMSKUL : MEASURING AOP-BASED SOFTWARE BY FUNCTION POINT ANALYSIS. THESIS ADVISOR : PROADPRAN PUNYABUKKANA, Ph.D., 109 pp. ISBN 974-53-2849-9.

Measuring the size of software leads to an estimate of the effort and the expense of software development activities. We hope to obtain as accurate as possible the expense the estimate and as early as possible in order to effectively management resource allocation and track its progress. Function Point Analysis is a popular means to measure the size of the software that many research found to be independent of technology employed since the inputs for the measurement are derived from software requirements specification.

However, when Aspect-Oriented Programming (AOP) is introduced, one of its major promises includes fewer lines of code or smaller program. AOP may be thought as an extension of Object-Oriented Programming (OOP). The requirements specification such as use case in the AOP paradigm may differ from that of OOP. Therefore, this thesis aims to investigate if function point is still usable to measure software development under AOP environment.

Twelve case studies from a large financial institute were carried out using Jacobson method to convert use case under the OOP environment to AOP. The result shows that the numbers of function point under AOP are significantly higher than those under OOP ($\alpha$=0.05). This thesis further analyzes the counting of use case points and found the results to confirm the findings.



Department Computer Engineering ..... Student's signature .....

Field of study Computer Science ..... Advisor's signature .....

Academic year 2005 .....

# กิตติกรรมประกาศ

ด้วยความกรุณาเป็นอย่างยิ่งของ อาจารย์ ดร.โปรดปราน บุณยพุกกณะ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่แนะนำให้ความรู้ คำปรึกษา ความช่วยเหลือต่าง ๆ ตลอดจนคอยดูแลการทำวิจัยของข้าพเจ้าเป็นอย่างดียิ่งจนสำเร็จลุล่วงได้ด้วยดี ใคร่ขอกราบขอบพระคุณท่านไว้เป็นอย่างสูง ณ โอกาสนี้

ขอกราบขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร. พรศิริ หมื่นไชยศรี เป็นประธานกรรมการ อาจารย์ ดร. ดาริชา สุธีวงศ์ และ อาจารย์ เชษฐ พัฒโนทัย เป็นกรรมการสอบวิทยานิพนธ์ ซึ่งได้สละเวลาในการสอบวิทยานิพนธ์ และกรุณาให้คำแนะนำแนวทางการปรับปรุงเนื้อหาวิทยานิพนธ์ฉบับนี้ให้สมบูรณ์ยิ่งขึ้น

ขอขอบคุณบริษัทกรุงไทยคอมพิวเตอร์เซอร์วิสเซส จำกัด (KCS) ที่ได้ให้โอกาสในการศึกษาต่อ และขอขอบคุณเพื่อนร่วมงานทุกท่านที่ได้เอื้อเฟื้อข้อมูลและคำแนะนำต่าง ๆ ที่เป็นประโยชน์ต่อการทำวิจัย

ขอขอบคุณ เพื่อน ๆ ที่เคยศึกษามาด้วยกัน โดยเฉพาะอย่างยิ่ง คุณชนะ ปรีชามานิตกุล, คุณไกรสีห์ อัญชนานนท์, คุณพรทิพย์ มาลีลัย และคุณปิยะพร พลหาญ ที่ได้ให้กำลังใจ คำแนะนำ และความช่วยเหลือมาโดยตลอด

ท้ายที่สุด ข้าพเจ้าใคร่ขอกราบพระคุณบิดา และ ผู้ช่วยศาสตราจารย์ ดร. ชวลิต รัตนธรรมสกุล พี่ชาย ที่ได้ให้การสนับสนุนและกำลังใจแก่ข้าพเจ้าเสมอมา

# สารบัญ

หน้า

บทที่ หน้า

# สารบัญตาราง

ตารางที่ หน้า

ตารางที่ หน้า

# สารบัญภาพ

รูปที่ หน้า

# บทที่ 1

# บทนำ

## 1.1 ความเป็นมาและความสำคัญของปัญหา

การบริหารโครงงานเป็นกระบวนการสำคัญในการพัฒนาระบบงาน ซึ่งจะดูแลทรัพยากรที่มีอยู่อย่างจำกัดให้ได้ประสิทธิผลอย่างเต็มที่ สำหรับการบริหารโครงงานการพัฒนาซอฟต์แวร์ในปัจจุบันนั้น การประเมินขนาดซอฟต์แวร์ตั้งแต่ขั้นตอนการวิเคราะห์และออกแบบ เป็นขั้นตอนหนึ่งที่ช่วยให้องค์กรสามารถจัดสรรทรัพยากรได้อย่างถูกต้อง เหมาะสม และติดตามความก้าวหน้าของโครงการได้ จึงนับได้ว่าการวัดขนาดซอฟต์แวร์เพื่อประเมินขนาดและค่าใช้จ่ายในการพัฒนาซอฟต์แวร์ เป็นส่วนสำคัญในความสำเร็จของโครงงาน

ฟังก์ชันพอยต์ (Function Point: FP) เป็นวิธีการวัดขนาดซอฟต์แวร์วิธีหนึ่งที่ได้รับการทดสอบมาแล้วว่า ไม่ว่าจะทำการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยีแบบใดก็ตามค่าฟังก์ชันพอยต์ที่ได้จะไม่มีการเปลี่ยนแปลง[2] เนื่องจากฟังก์ชันพอยต์เป็นการวัดจำนวนฟังก์ชันที่มีอยู่ในซอฟต์แวร์โดยใช้ข้อกำหนดความต้องการของซอฟต์แวร์ (Software Requirements Specification) เป็นข้อมูลพื้นฐานในการวิเคราะห์คำนวณขนาดของซอฟต์แวร์ นอกจากนี้ ฟังก์ชันพอยต์ยังสามารถนำมาใช้ประเมินขนาดซอฟต์แวร์ได้ ตั้งแต่ขั้นการกำหนดความต้องการและออกแบบ ตลอดจนถึงขั้นการพัฒนาซอฟต์แวร์ โดยมี International Function Point Users Group (IFPUG) เป็นผู้ดูแลและปรับปรุงมาตรฐานการนับฟังก์ชันพอยต์ในปัจจุบัน

ถึงแม้ว่าเทคโนโลยีที่ใช้ในการพัฒนาซอฟต์แวร์ จะก้าวหน้าและเปลี่ยนแปลงไปเพียงใด แต่ฟังก์ชันการทำงานของซอฟต์แวร์นั้นยังคงเหมือนเดิม ดังนั้นค่าฟังก์ชันพอยต์ที่ได้จึงไม่แตกต่างกัน ในปัจจุบันนักวิจัยทางด้านวิศวกรรมซอฟต์แวร์ (Software Engineering) ได้ทำการ ค้นคว้าหาวิธีการที่จะทำให้การพัฒนาซอฟต์แวร์มีประสิทธิภาพดียิ่งขึ้น การเขียนโปรแกรมเชิงแง่มุม (Aspect-Oriented Programming: AOP) จึงได้ถูกนำเสนอขึ้นมาเป็นแนวคิดใหม่ ซึ่งมีจุดมุ่งหมายเพื่อแยกส่วนประกอบสำคัญที่เหมือนกันในหลาย ๆ หน่วยย่อยในซอฟต์แวร์ออกมาเพื่อเป็นการลดความซับซ้อนในการพัฒนาซอฟต์แวร์ จากงานวิจัยของ Laddad [5] ได้นำเสนอข้อดีในการพัฒนาซอฟต์แวร์ด้วยกรรมวิธี AOP ทำให้ซอฟต์แวร์นั้นมีคุณภาพสูงขึ้น และเป็นการลดค่าใช้จ่ายในการพัฒนาอีกด้วย และเนื่องจาก AOP เกิดขึ้นมาจากการแก้ไขปัญหาบางประการที่เกิดขึ้นในการเขียนโปรแกรมเชิงวัตถุ (Object-Oriented Programming: OOP) [1] คือในวัตถุหลาย ๆ วัตถุมักมีโปรแกรมบางส่วนเหมือนกัน และยากที่จะนำออกมาเป็นซูเปอร์คลาสร่วมกัน จึง

ไม่สามารถทำการสืบทอดคุณสมบัติ (Inheritance) ได้ ตัวอย่างเช่น การทำ Logging ที่ไม่ว่าคลาสไหนก็ต้องทำการเขียนโปรแกรมที่เหมือนกัน ตามแนวความคิดของ AOP นั้น โปรแกรมเมอร์ไม่ต้องเขียนโปรแกรมเหมือนกันในทุก ๆ คลาส โดยแยกเอาส่วนที่เหมือนกันออก มาไปนิยามไว้ที่เดียวกัน และสามารถกำหนดได้ว่าจะเรียกใช้งานส่วนที่นิยามไว้เมื่อใด ซึ่งทำให้เวลาและค่าใช้จ่ายที่ใช้ในพัฒนาซอฟต์แวร์ลดลง [1] เมื่อเปรียบเทียบกับการเขียนโปรแกรมแบบ OOP

จากแนวความคิดของ AOP ที่แยกเอาส่วนที่เหมือนกันออกมาไปนิยามไว้ที่เดียวกัน ก็น่าจะทำให้ขนาดซอฟต์แวร์มีขนาดเล็กลงตามไปด้วย อย่างไรก็ดีในงานวิจัยของ Fetcke และคณะ [2] พบว่าเทคนิคฟังก์ชันพอยต์เป็นวิธีการวัดขนาดซอฟต์แวร์ที่ได้รับการยอมรับว่า สามารถนำมาใช้ได้กับการพัฒนาซอฟต์แวร์ในทุกยุคทุกสมัย ดังนั้นผู้วิจัยจึงมีความสนใจนำเทคนิคฟังก์ชันพอยต์มาประยุกต์ใช้กับการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี AOP โดยการแปลงแผนภาพ Use Case ในเชิง OOP มาเป็นแผนภาพ Use Case ในเชิง AOP ตามแนวความคิดของ Jacobson [13] แล้วใช้เทคนิคฟังก์ชันพอยต์จากแผนภาพ Use Case ในเชิง AOP ซึ่งน่าจะได้ค่าฟังก์ชันพอยต์ที่แตกต่างกันเนื่องจากมีจำนวน use case ในแผนภาพเพิ่มขึ้น ดังนั้นในวิทยานิพนธ์นี้ จึงได้มีแนวคิดที่จะเปรียบเทียบขนาดซอฟต์แวร์ที่พัฒนาด้วยเทคโนโลยี OOP กับซอฟต์แวร์ที่พัฒนาด้วยเทคโนโลยี AOP โดยใช้เทคนิคฟังก์ชันพอยต์มาประยุกต์ใช้กับแผนภาพ Use Case ค่าฟังก์ชันพอยต์ที่ได้จากการนับแผนภาพ Use Case ทั้งสองเทคโนโลยียังคงมีค่าเท่ากันหรือไม่

## 1.2 วัตถุประสงค์

วิทยานิพนธ์นี้มีจุดประสงค์ เพื่อทำการตรวจสอบค่าฟังก์ชันพอยต์ที่ได้จากการวัดขนาดซอฟต์แวร์ที่พัฒนาด้วยเทคโนโลยี OOP กับ ซอฟต์แวร์ที่พัฒนาด้วยเทคโนโลยี AOP จากแผนภาพ Use Case โดยการเปรียบเทียบค่าฟังก์ชันพอยต์ที่ได้ว่ามีความเหมือนหรือแตกต่างกันอย่างไร

## 1.3 ขอบเขตของการวิจัย

วิทยานิพนธ์นี้มีขอบเขตของการวิจัยดังนี้

1.3.1 ระบบงานที่ใช้เป็นกรณีศึกษาในวิทยานิพนธ์นี้ เป็นระบบงานที่ได้พัฒนาขึ้นมาสำหรับธนาคารกรุงไทยและบริษัทในเครือที่มีการสร้างแผนภาพ Use Case ไว้แล้ว ซึ่งจะเป็นระบบงานที่มีขนาด Use Case ในระดับสุดท้าย 8 Use Case ขึ้นไป

1.3.2 จำนวนระบบงานที่ใช้เป็นกรณีศึกษาประมาณ 10 - 15 ระบบ เนื่องจากในการพัฒนาระบบแต่ละระบบ อาจจะเป็นกลุ่มนักพัฒนาระบบกลุ่มเดียวกัน หรือคนละ

กลุ่มก็ได้ เพื่อเป็นการกระจายความแปรปรวนที่จะเกิดขึ้น และ ในระบบงานธนาคารกรุงไทยที่พัฒนาระบบด้วยเทคโนโลยี OOP ยังมีจำนวนไม่มากนักเมื่อเทียบกับระบบงานที่ใช้เทคโนโลยีอื่นในการพัฒนา

## 1.4 ขั้นตอนและวิธีการดำเนินงาน

วิทยานิพนธ์นี้มีขั้นตอนและวิธีการดำเนินงานดังนี้

1.4.1 ศึกษาทฤษฎีเกี่ยวกับการเขียนโปรแกรมแบบ AOP, การสร้างแผนภาพ Use Case และการนับค่าฟังก์ชันพอยต์

1.4.2 ศึกษาและทำความเข้าใจเกี่ยวกับระบบงานที่ใช้เป็นกรณีศึกษา และนำ Use Case ของระบบงานนี้มาใช้ในขั้นตอนการสร้างแผนภาพ Use Case เชิง AOP

1.4.3 สร้างแผนภาพ Use Case เชิง AOP จากความต้องการของระบบงานที่ใช้เป็นกรณีศึกษา

1.4.4 นับค่าฟังก์ชันพอยต์จากแผนภาพ Use Case ทั้งสอง

1.4.5 ทำการเปรียบเทียบค่าฟังก์ชันพอยต์ที่ได้ โดยวิธีทางสถิติมาช่วยในการตรวจสอบสมมติฐานการวิจัยดังนี้

1) ค่าฟังก์ชันพอยต์ที่ได้จากการนับแผนภาพ Use Case ด้วยวิธีการของ Jacobson ในเชิง AOP และค่าฟังก์ชันพอยต์ที่ได้จากการนับแผนภาพ Use Case ในเชิง OOP ไม่แตกต่างกัน

2) ค่าฟังก์ชันพอยต์ที่ได้จากการนับแผนภาพ Use Case ด้วยวิธีการของ Jacobson ในเชิง AOP จะมีค่ามากกว่าค่าฟังก์ชันพอยต์ที่ได้จากการนับแผนภาพ Use Case ในเชิง OOP

3) ค่าฟังก์ชันพอยต์ที่ได้จากการนับแผนภาพ Use Case ด้วยวิธีการของ Jacobson ในเชิง AOP โดยผู้วิจัยสร้างขึ้น และค่าฟังก์ชันพอยต์ที่ได้จากการนับแผนภาพ Use Case ด้วยวิธีการของ Jacobson ในเชิง AOP โดยกลุ่มตัวอย่างในสายงานคอมพิวเตอร์สร้างขึ้น ไม่แตกต่างกัน

4) ค่าฟังก์ชันพอยต์ที่ได้จากการนับแผนภาพ Use Case ด้วยวิธีการของ Jacobson ในเชิง AOP โดยกลุ่มตัวอย่างในสายงานคอมพิวเตอร์สร้างขึ้น และค่าฟังก์ชันพอยต์ที่ได้จากการนับแผนภาพ Use Case ในเชิง OOP ไม่แตกต่างกัน

5) ในการวัดขนาดซอฟต์แวร์ด้วยวิธี Use Case Points ค่าUse Case Points ที่ได้จากแผนภาพ Use Case ด้วยวิธีการของ Jacobson ในเชิง AOP และค่า Use Case Points ที่ได้จากแผนภาพ Use Case ในเชิง OOP ไม่แตกต่างกัน

1.4.6 สรุปผลการวิจัย และจัดทำรายงานวิทยานิพนธ์

## 1.5 ประโยชน์ที่คาดว่าจะได้รับ

ประโยชน์ที่คาดว่าจะได้รับจากวิทยานิพนธ์มีดังนี้

1.5.1 สามารถนำเทคนิคที่ใช้ในการพิจารณาความต้องการของซอฟต์แวร์ทั้งFunctional Requirements และ Non-Functional Requirements เพื่อสร้างแผนภาพ Use Case ที่ถูกต้องและสมบูรณ์ยิ่งขึ้น

1.5.2 สามารถนำเทคนิคที่ใช้ในการสร้างแผนภาพ Use case ในเชิง AOP เป็นแนวทางในการสร้างหรือปรับเปลี่ยนแผนภาพ Use case ที่ใช้ในระบบงานปัจจุบัน

1.5.3 ได้ศึกษาการนับฟังก์ชันพอยต์ในเชิงวัตถุ และสามารถนำมาประยุกต์ใช้กับ AOP ได้เช่นกัน และเพื่อนำไปสู่แนวทางค้นคว้าวิจัยหาเทคนิคการนับฟังก์ชันพอยต์ที่เหมาะสมสำหรับ AOP ต่อไป

# บทที่ 2

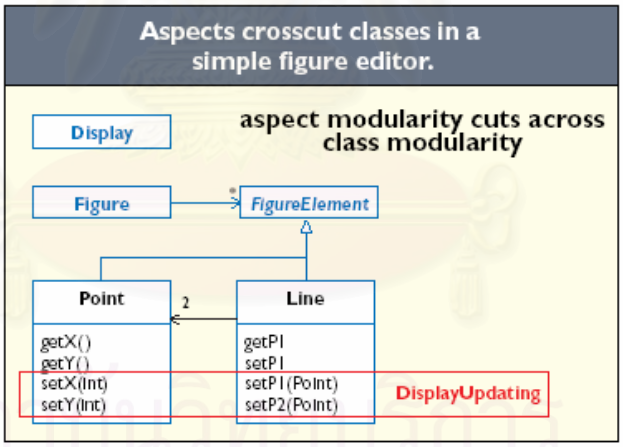
# ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

## 2.1 ทฤษฎีที่เกี่ยวข้อง

ทฤษฎีที่จะนำมาใช้ในวิทยานิพนธ์นี้ ได้แก่ การเขียนโปรแกรมเชิงแง่มุม (Aspect-Oriented Programming) ยูเอ็มแอล (UML) แผนภาพการใช้งาน (Use Case Diagram) การวิเคราะห์ฟังก์ชันพอยต์ (Function Point Analysis) และ ยูสเคสพอยต์ (Use Case Points)

### 2.1.1 การเขียนโปรแกรมเชิงแง่มุม (Aspect-Oriented Programming : AOP)

เทคโนโลยี AOP เป็นเทคโนโลยีที่ถูกพัฒนาออกมาเพื่อทำการแก้ไขปัญหาบางประการของแนวความคิดแบบ OOP โดยเฉพาะปัญหาในเรื่องของการแบ่งส่วนประกอบสำคัญที่เหมือนกัน ในหลาย ๆ หน่วยย่อยในซอฟต์แวร์ ที่เรียกว่า Crosscutting concern ให้แยกจากออกกันอย่างชัดเจน

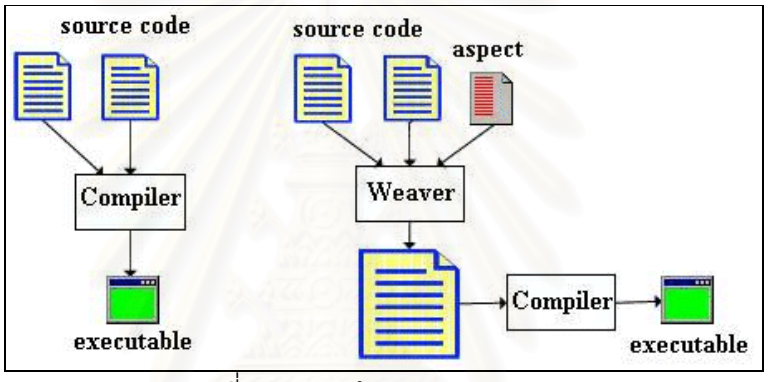



รูปที่ 2.1 แผนภาพ UML ของ Simple Figure Editor

ตัวอย่างของ Crosscutting concern ดังรูปที่ 2.1 เป็นแผนภาพ UML ที่ประกอบด้วยคลาสสองคลาสใน Figure Element ได้แก่ คลาส Point และ คลาส Line [6] ทั้งสองคลาสมีคุณลักษณะที่ต่างกันอย่างชัดเจน และมีความสัมพันธ์กันอย่างใกล้ชิด (Cohesion) เมื่อผู้ใช้งานทำการเคลื่อนย้ายเปลี่ยนตำแหน่งของ Figure Element ใหม่ ต้องแสดงข้อความเตือนขึ้นมาบนหน้าจอให้ผู้ใช้งานทราบ ในกรอบสี่เหลี่ยม DisplayUpdate จากรูปที่ 2.1 แสดงให้เห็นถึงเมทธอดที่มีผลกระทบต้องทำการเขียนโปรแกรมเพิ่มทุกเมทธอด ถึงแม้ว่าจะทำงานเหมือนกันก็ตาม เนื่อง

จากข้อความที่แสดงบนหน้าจอนั้นจะมีความแตกต่างกันไปในแต่ละคลาส ซึ่งในกรอบสี่เหลี่ยมนี้จะถูกเรียกว่า Crosscutting concern

AOP จึงถูกออกแบบมาเพื่อทำการแบ่ง Crosscutting concern ให้เป็น single unit เรียกว่า Aspect ซึ่งจะทำการซ่อนพฤติกรรมของคลาสต่าง ๆ ที่มีผลกระทบถึงกันไปไว้รวมกันที่หน่วยโปรแกรม reusable โดยเริ่มต้นการพัฒนาซอฟต์แวร์ด้วย OOP และทำการแบ่ง Crosscutting concern ของโปรแกรมด้วย Aspect จนในขั้นตอนสุดท้ายก็ทำการรวมโปรแกรมและ Aspect เข้าด้วยกัน ด้วยวิธีที่เรียกว่า Aspect Weaver [3] ดังรูปที่ 2.2 เป็นการอธิบายขั้นตอนการทำงานของ Weaving




รูปที่ 2.2 การทำงานของ Weaving

AOP ไม่ได้มาแทนที่ OOP แต่จะเป็นแนวทางที่ช่วยในการเขียนโปรแกรมให้มีประสิทธิภาพมากขึ้น โดยเฉพาะการดึงเอาส่วนที่ทุก ๆ หน่วยย่อยในโปรแกรมสามารถนำมาใช้ได้เหมือน ๆ กัน ออกมา และเมื่อมีหน่วยย่อยเพิ่มเข้ามาก็สามารถดึงส่วนนั้นมาใช้งานได้ทันที เป็นการลดเวลาในการพัฒนาซอฟต์แวร์ลง ซึ่งก็ส่งผลให้ค่าใช้จ่ายในการพัฒนาซอฟต์แวร์ลดลงด้วย

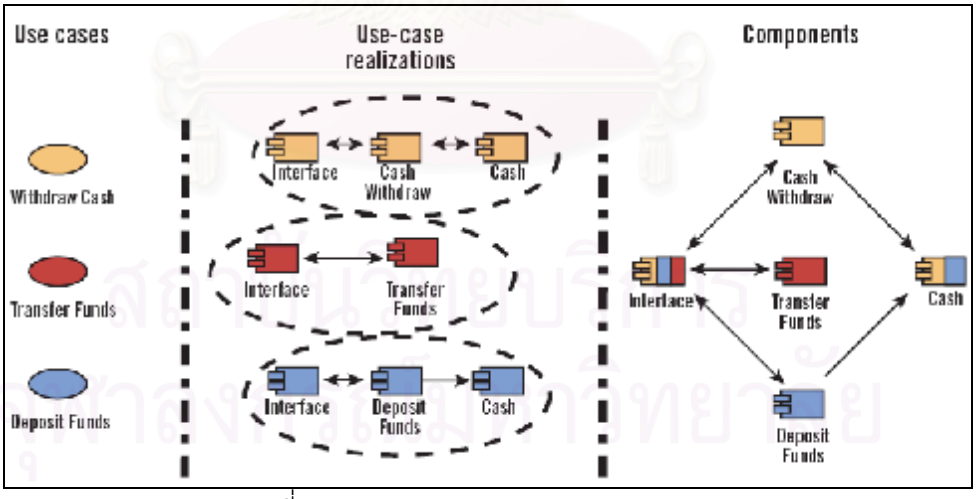
ในปัจจุบันได้มีนักวิจัยและหน่วยงานต่าง ๆ ได้ทำการค้นคว้าวิจัยทางด้านเทคโนโลยี AOP เพิ่มมากขึ้น โดยเฉพาะในระดับขั้นตอนการกำหนดความต้องการของซอฟต์แวร์ (Requirements Level) ซึ่งได้แก่ทฤษฎี Aspect-Oriented Software Development with Use Cases (AOSD/UC) ของ Jacobson และ Ng [13]

Aspect-Oriented Software Development (AOSD) เป็นเทคนิคกระบวนการ พัฒนาซอฟต์แวร์ด้วย Aspect โดยเริ่มจากขั้นตอนการกำหนดความต้องการของซอฟต์แวร์ การวิเคราะห์และออกแบบ การพัฒนาระบบ ตลอดจนถึงการทดสอบ AOSD เป็นกระบวนการที่ถูกออกแบบให้เหมาะสมกับระบบที่ให้ความสนใจกับสิ่งต่าง ๆ ภายในระบบเป็นจำนวนมาก ๆ สามารถ

แบ่งแยกสิ่งที่สนใจเหล่านั้นออกจากกัน ทำให้เข้าใจโครงสร้างระบบได้ดียิ่งขึ้น และง่ายต่อการกำหนดและเพิ่มเติมความต้องการของซอฟต์แวร์ใหม่จากผู้ใช้งาน

โดยปกติแล้ว Aspect Orientation จะเป็นกรรมวิธีที่ช่วยในการลดปัญหาของ Crosscutting concern ที่อาจเกิดจากการพัฒนา แต่มีสิ่งที่ช่วยลดปัญหา Crosscutting concern ได้ในขั้นตอนของการกำหนดความต้องการ นั่นคือการใช้ Use Case เนื่องจากในความเป็นจริงแล้วจุดมุ่งหมายในการสร้าง Use Case ก็คือการแบ่งแยกสิ่งที่ผู้ใช้งานให้ความสนใจต้องการใช้งานระบบ แต่ในขณะเดียวกันการทำงานภายในแต่ละ Use Case เองนั้นมีส่วนเกี่ยวข้องกับการทำงานในคลาสหลาย ๆ คลาส จึงถือได้ว่า Use Case ก็คือ Crosscutting concern นั่นเอง ตามแนวคิดของ AOSD/UC ของ Jacobson และ Ng สามารถแบ่งชนิดของ Crosscutting concern ออกเป็น 2 ชนิดดังนี้

1. Peers Use Case เป็น Use Case ที่แตกต่างกัน และเป็นอิสระไม่ขึ้นกับ Use Case อื่น ซึ่งเรียกว่า Base Use Case ตัวอย่างเช่น ระบบ ATM ประกอบไปด้วย Cash withdrawal, Fund transfer และ Cash deposit เหล่านี้ถือว่าเป็น Peers Use Case เนื่องจากทั้งสาม Use Case นี้มีส่วนเข้าไปเกี่ยวข้องในหลาย ๆ คลาส ทำให้เกิดปัญหาเรื่องการกระจัดกระจาย (Scattering) และ การยุ่งเหยิง (Tangling) ของโปรแกรมในขณะการพัฒนาซอฟต์แวร์ ดังแสดงให้เห็นดังรูปที่ 2.3



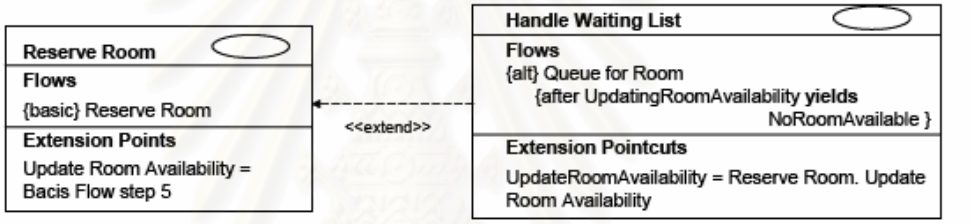

รูปที่ 2.3 Scattering และ Tangling ของระบบ ATM

2. Extensions Use Case เป็น Use Case ที่เพิ่มเติมขึ้นไม่ได้เป็นความต้องการพื้นฐาน แต่จะถูกเรียกใช้โดย Base Use Case ที่เป็นความต้องการพื้นฐาน ดังนั้นในคลาสที่เรียกใช้งาน Extensions Use Case ก็ต้องทำการเขียนโปรแกรมเพิ่มเติมเข้าไปด้วย

ซึ่งอาจจะไม่ได้ถูกเรียกใช้งานเลย แต่ในคลาสก็ยังคงต้องมีโค้ดในส่วนที่เติมเพิ่มอยู่เสมอ

AOSD/UC เป็นเทคนิคส่วนขยายที่เพิ่มเติมขึ้นจากเทคนิค Use Case ปกติ ซึ่งประกอบไปด้วย 2 ส่วนประกอบหลัก คือ Pointcuts สำหรับ Use Case และ กลุ่มของส่วนที่ใช้ในการพัฒนาที่เรียกว่า Use Case Slice และ Use Case Modules

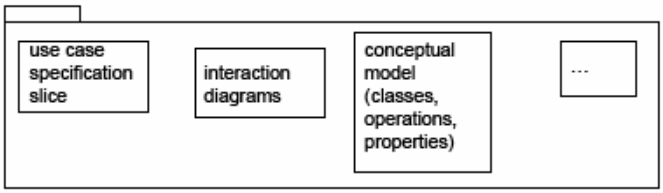
- **Pointcuts** เป็นการนิยามจุดที่ระบุตำแหน่งเพื่อจะไปเรียก Aspect ขึ้นมาทำงาน หรือที่เรียกว่า join points ใน AOP ซึ่งหมายถึง Extension Points ใน Use Case Specification ที่ใช้เพียงแค่อ้างอิงไปยัง Extensions Use Case เท่านั้น แต่ใน AOSD/UC จะกลายมาเป็นส่วนสำคัญที่ช่วยในการลดการขึ้นต่อกันระหว่าง Extensions Use Case กับ Base Use Case ให้น้อยลง




รูปที่ 2.4 Extension Points และ Pointcuts สำหรับ AOSD/UC

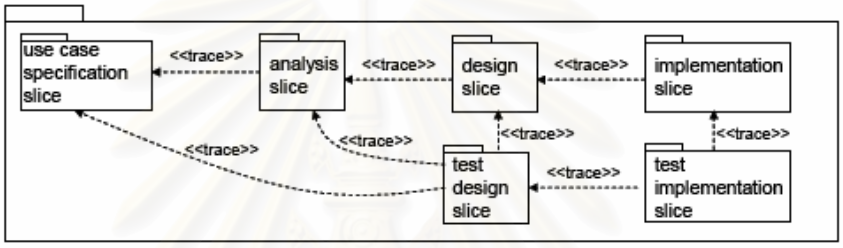
รูปที่ 2.4 แสดงให้เห็น Extension Points และ Pointcuts สำหรับ AOSD/UC ซึ่งยกตัวอย่างระบบการสำรองห้องพักโรงแรมออนไลน์ ใน Use Case Reserve Room ซึ่งเป็น Base Use Case จะมี Extension Points ระบุตำแหน่งว่าจะทำการเรียกใช้ Handle Waiting List ซึ่งเป็น Extensions Use Case ในลำดับขั้นตอนไหน ซึ่งก็คือ Extension Pointcuts ใน Extensions Use Case ที่จะทำงานตามขั้นตอนที่กำหนดไว้ ซึ่งในขั้นตอนการพัฒนาต้องคำนึงถึงเงื่อนไขของการเรียกใช้ Extensions Use Case ด้วยว่าจะเรียกใช้เมื่อใด โดยกำหนดเงื่อนไขได้ดังนี้ before, after และ around

- **Use Case Slice** ประกอบด้วยรายละเอียดต่าง ๆ ของ Use Case ที่ได้มาจากการอธิบายข้อมูลใน Use Case Specification ซึ่งเป็นข้อมูลสำหรับในขั้นตอนการพัฒนาซอฟต์แวร์ ส่วน **Use Case Modules** จะประกอบด้วยรายละเอียดที่มีความสัมพันธ์กับ Use Case ทั้งหมดตลอดจนถึงในขั้นตอนการพัฒนาซอฟต์แวร์




รูปที่ 2.5 ตัวอย่างองค์ประกอบใน Use Case Slice

รูปที่ 2.5 แสดงให้เห็นองค์ประกอบบางส่วนที่สามารถกำหนดใน Use Case Slice ซึ่งเป็นตัวอย่างในระดับการกำหนดความต้องการของซอฟต์แวร์ Use Case Slice ต่าง ๆ ในระบบ ก็จะถูกนำมารวมกันใน Use Case Modules ดังแสดงในรูปที่ 2.6




รูปที่ 2.6 Use Case Modules

เนื่องจากในการพัฒนาซอฟต์แวร์มีการแบ่งประเภทความต้องการของซอฟต์แวร์ ออกเป็นสองชนิด คือ Functional Requirements และ Non-Functional Requirements ดังนั้นการพิจารณาในส่วนของ Crosscutting concern เพื่อทำการกำหนด Aspect ของระบบ ในขั้นตอนของการกำหนดความต้องการ ตามแนวคิดของ AOSD/UC สามารถทำได้ดังนี้

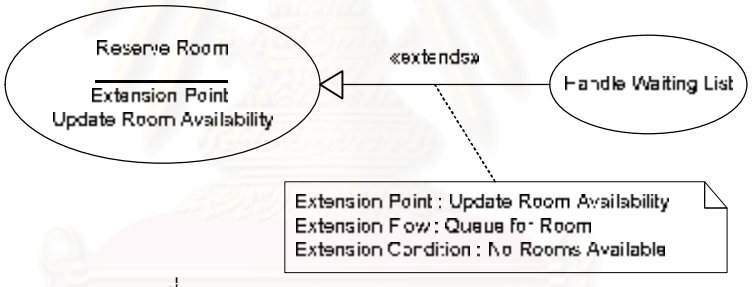
ในส่วนของ Functional Requirements ซึ่งเป็นความต้องการที่เป็นหน้าที่หลักของระบบ ซึ่งจะแสดงอยู่ในรูปแบบของ Use Case หลักการพิจารณาหา Crosscutting concern จาก Use case มีดังนี้

1) การระบุปัจจัยผันแปร (Variability) ในแต่ละ Use Case มีลำดับขั้นตอนของเหตุการณ์มากมาย ซึ่งเป็นการอธิบายการจัดการความผันแปรที่เกิดขึ้นตลอดทั้ง Use Case นั้น จึงมีความสำคัญในการจำแนกปัจจัยที่ทำให้เกิดความผันแปรเหล่านั้น บางปัจจัยเป็นจุดสำคัญต่อผู้ที่มีส่วนได้ส่วนเสียในระบบ ตัวอย่างเช่น ระบบการสำรองห้องพักโรงแรมออนไลน์ ประเภทของลูกค้าก็เป็นปัจจัยผันแปรปัยจัยหนึ่งของระบบ ถ้าลูกค้าคนนี้เป็นสมาชิกของโรงแรมเงื่อนไขในการบริการก็จะแตกต่างจากลูกค้าทั่ว ๆ ไป เช่น คิดค่าบริการในอัตราพิเศษ หรือมีการคิดสะสมแต้มเพื่อเป็นมอบสิทธิพิเศษให้ เป็นต้น จะเห็นได้ว่าค่าของปัจจัยผันแปรที่แตกต่างกันก็ส่งผลทำ

ให้การทำงานของระบบก็แตกต่างด้วยเช่นกัน ดังนั้นการระบุปัจจัยผันแปรและค่าของปัจจัยผันแปรเหล่านั้น จึงเป็นส่วนสำคัญที่ช่วยให้เข้าใจความซับซ้อนของระบบมากยิ่งขึ้น

2) การจัดการดูแลปัจจัยผันแปร เนื่องจากปัจจัยเหล่านี้อาจมีส่วนเกี่ยวข้องกับหลาย ๆ Use Case จึงสามารถพิจารณาได้ว่าปัจจัยผันแปรก็คือ Aspect ของ Use Case ดังนั้นวิธีที่จะจัดการดูแลปัจจัยผันแปรทำได้โดยระบุและอธิบายลงไปใน Use Case Specification ในส่วนของ Alternate Flows แต่ในบางครั้งค่าของปัจจัยผันแปรก็ขึ้นอยู่กับเงื่อนไขอื่น เช่น ลูกค้าที่เป็นสมาชิกของโรงแรมเดียวกันแต่คนละสาขา สาขาหนึ่งอาจจะให้สิทธิพิเศษที่มากกว่าสาขาอื่นก็จะเป็นการทำงานของระบบที่เพิ่มเติมขึ้น ซึ่งจะระบุและอธิบายปัจจัยผันแปรนั้น ในส่วนของ Extension Flows

3) Extension Use Case ซึ่งถือได้ว่าเป็น Crosscutting Concern รูปแบบหนึ่ง จึงต้องมีการระบุ Extension Points ใน Base Use Case เพื่อให้ทราบถึง Pointcuts ที่ระบุตำแหน่งเพื่อจะไปเรียก Aspect ขึ้นมาทำงาน ดังรูปที่ 2.7 แสดงให้เห็นการระบุ Extension Point ใน Reserve Room ซึ่งเป็น Base Use Case




รูปที่ 2.7 Handle Waiting List extends reserve room

สำหรับ Non-Function Requirements เป็นความต้องการอื่นๆ ที่ไม่ใช่หน้าที่หลักที่ระบบต้องทำ แต่เป็นคุณสมบัติอื่นๆ ที่ผู้ใช้ต้องการได้จากระบบ เช่น ความปลอดภัยของระบบ ความเชื่อถือได้ของระบบ มีความสามารถทางด้าน I/O ความสามารถในการเชื่อมต่อกับระบบอื่นๆ ตามแนวคิดของ AOSD/UC สนับสนุนให้สามารถทำการกำหนด Use Case สำหรับ Non-Functional Requirements ซึ่งเรียกว่า Infrastructure Use Case ใช้สำหรับอ้างอิงถึงกิจกรรมของระบบที่จำเป็นสำหรับการทำงานตามความต้องการของผู้ใช้งาน ซึ่งอาจจะขัดแย้งกับเทคนิค Use Case ที่แสดงให้เห็นว่าสิ่งใดที่ผู้ใช้งานสามารถใช้งานอะไรได้บ้าง และสิ่งที่ระบบตอบสนองตามการใช้งานคืออะไร ดังนั้นการพิจารณาว่า Non-Functional Requirements ในส่วนใดบ้างที่สามารถนำมาสร้างเป็น Use Case ได้ มีขั้นตอนดังนี้

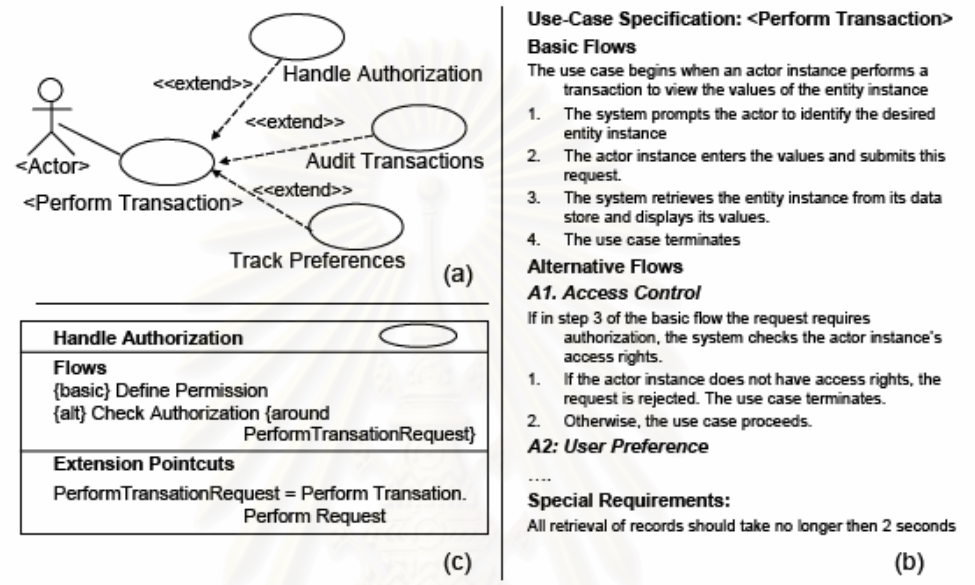
1) สิ่งที่ผู้ใช้งานระบบต้องการคืออะไร

2) สามารถทำการทดสอบได้อย่างไร

3) ออกแบบการทดสอบ

4) เปลี่ยนแปลงให้เป็น Use Case

5) เพิ่มเติมในส่วน Functional Requirements ด้วย Use Case ที่สร้างขึ้นมาใหม่




รูปที่ 2.8 Infrastructure Use Case

จากรูปที่ 2.8 แสดงให้เห็นรูปแบบของ Infrastructure Use Case โดยใช้ <Perform Transaction> Use Case แสดงถึง Infrastructure Requirements ของระบบ และมีความสัมพันธ์กับ use case ที่เพิ่มเติมเข้ามาเป็นแบบ <<extend>> ซึ่งสามารถช่วยมองเห็นภาพของการทำงานของระบบในส่วนของเครื่องได้ดียิ่งขึ้น

การพิจารณาในส่วนของ Crosscutting concern ของแผนภาพ Use Case ในขั้นตอนของการกำหนดความต้องการ ทั้งในส่วนของ Functional และ Non-Functional Requirements จะถูกระบุอยู่ใน use Case Specification เพื่อนำไปสร้าง Use Case Slice และ Use Case Modules ในขั้นตอนของการวิเคราะห์และออกแบบต่อไป

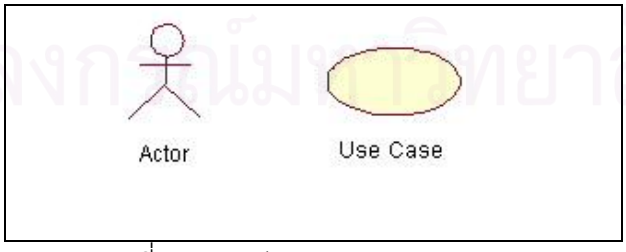
ในวิทยานิพนธ์นี้ จะนำเทคนิค AOSD/UC มาใช้ในขั้นตอนของการวิเคราะห์ความต้องการของซอฟต์แวร์ เพื่อที่จะทำการสร้างแผนภาพ Use case ในเชิง AOP ออกมา แล้วนำแผนภาพที่ได้ไปใช้ในการนับค่าฟังก์ชันพอยต์

2.1.2 UML และ Use case Diagram

UML เป็นภาษามาตรฐานหนึ่งซึ่งใช้แสดงแบบจำลอง (Modeling Language) ที่ได้จากกระบวนการกำหนดแนวคิดให้กับวัตถุ การเชื่อมโยงแบบจำลองที่สร้างขึ้นด้วยภาษา UML กับภาษาต่าง ๆ ทำได้ง่ายเนื่องจากมีแนวความคิดเชิงวัตถุเหมือนกัน โดยความสามารถในการแสดงออก (Expressiveness) ของ UML เกิดจากการมองระบบจากหลายมุมมอง ผ่านการแสดงออกจากหลายแผนภาพ ได้แก่ Use case diagram, Class diagram, Sequence diagram, State diagram

แผนภาพ UML ที่ใช้ในวิทยานิพนธ์นี้คือแผนภาพ Use Case เนื่องจากเป้าหมายของการสร้าง Use Case นั้น ก็เพื่ออธิบายเรื่องราวของขอบเขตของปัญหา ทั้งหมดว่ามีส่วนประกอบอะไรบ้างและเกี่ยวพันกันเป็นระบบได้อย่างไร ซึ่งจะช่วยให้ผู้พัฒนาระบบสามารถแยกแยะได้ว่าจะมีกิจกรรมอะไรที่น่าจะเกิดขึ้นในระบบบ้าง แผนภาพ Use Case นั้นถือว่าเป็นแผนภาพพื้นฐานซึ่งมีขีดความสามารถในการอธิบายสิ่งต่าง ๆ ด้วยรูปภาพที่ไม่ซับซ้อน และถือเป็นรากฐานในการเริ่มต้นวิเคราะห์ระบบ ดังนั้น Use Case ที่สมบูรณ์ถูกต้องย่อมช่วยในการวิเคราะห์ระบบมีความสมบูรณ์และถูกต้องเช่นเดียวกัน นอกจากนี้ Use case จะนำไปใช้ในโครงการพัฒนาระบบทั้งนี้เพื่อเป็นการกำหนด Requirements ของระบบ และนำข้อมูลนี้ไปใช้ในการวางแผนโครงการ อีกด้วย

Use Case เป็นแผนภาพรูปแบบหนึ่งของการทำ Functional Requirements โดยจะแสดงภาพการทำงานของระบบโดยอธิบายลักษณะของสถานะการณ์การใช้งานระบบ โดยจะอธิบายวิธีการทำงานของระบบ และการโต้ตอบกัน (Interaction) ระหว่างผู้ใช้ (user) กับตัวระบบ โดยจะแสดงเป็นรูปภาพ โดยมีองค์ประกอบสำคัญคือ Actor และ use case ดังรูปที่ 2.9




รูปที่ 2.9 องค์ประกอบของ Use case

1. Actor คือคน หรือสิ่งที่อยู่ภายนอกระบบ และมีการโต้ตอบกับระบบซึ่งอาจจะเป็นการส่ง หรือรับข้อมูล หรือทั้งส่งทั้งรับ

2. Use case เป็นการทำงานโดยระบบเมื่อได้รับ (หรืออาจไม่ได้รับ) การร้องของ และข้อมูลจาก Actor เพื่อส่งข้อมูลให้กับ Actor ส่วนใหญ่จะใช้คำกริยาแทนตัว Use case

ในปัจจุบันตามมาตรฐานของ Object Management Group (OMG) อยู่ที่ UML 2.0 ซึ่งรองรับแนวความคิดเชิง AOP สอดคล้องกับแนวทางของ AOSD/UC

### 2.1.3 การวิเคราะห์ฟังก์ชันพอยต์ (Function Point Analysis : FPA)

ฟังก์ชันพอยต์เป็นวิธีวัดขนาดของซอฟต์แวร์วิธีหนึ่ง ที่สามารถใช้ได้กับทุก ๆ ภาษา ถูกนำเสนอโดย Allan Albrecht ในขณะทำงานวิจัยอยู่ที่บริษัท IBM ในปี 1979 ฟังก์ชันพอยต์เป็นเทคนิคที่หาจำนวนฟังก์ชันที่ประกอบอยู่ในซอฟต์แวร์ในมุมมองของการใช้งานของผู้ใช้ Allan Albrecht สังเกตว่า ซอฟต์แวร์ต่าง ๆ มีองค์ประกอบพื้นฐานที่คล้ายคลึงกัน ซึ่งแบ่งออกได้เป็น 5 ส่วน [2] ดังรูปที่ 2.10 โดยมีรายละเอียดดังนี้

Data Functions เป็นส่วนที่อ้างถึงความต้องการข้อมูลของผู้ใช้

1. Internal Logical Files (ILF) เป็นข้อมูลที่อยู่ภายในซอฟต์แวร์ ที่ถูกปรับปรุงและทดสอบโดยผู้ใช้

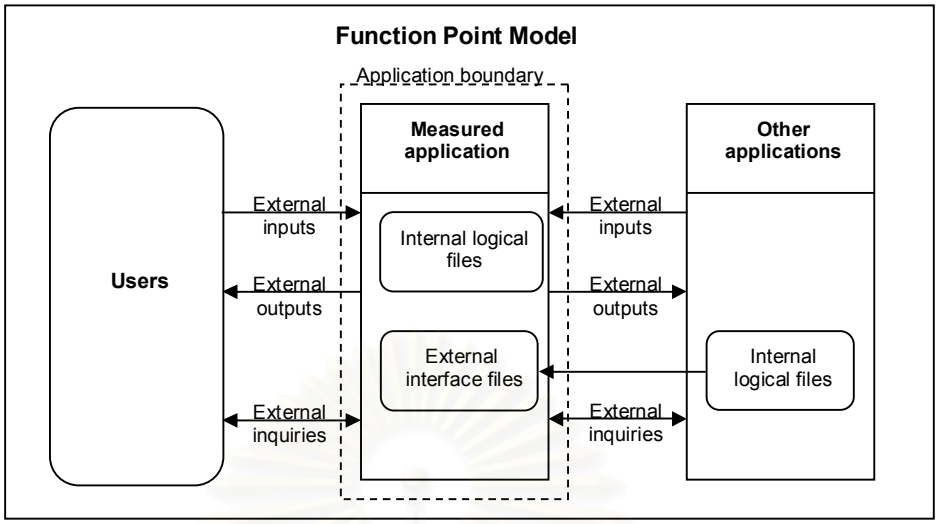
2. External Interface Files (EIF) เป็นข้อมูลที่มาจากซอฟต์แวร์อื่น ที่มีจุดประสงค์เพียงแค่ใช้อ้างอิงภายในซอฟต์แวร์นี้เท่านั้น ผู้ใช้ไม่สามารถทำการแก้ไขปรับปรุงข้อมูลเหล่านี้ได้ ดังนั้น EIF นี้จะเป็น ILF ของซอฟต์แวร์อื่นที่ดูแล file นี้นั่นเอง

Transactional Functions เป็นส่วนที่อ้างถึงการประมวลผลข้อมูลของผู้ใช้

3. External Input (EI) เป็นข้อมูลจากภายนอกที่ซอฟต์แวร์ต้องการ ผู้ใช้สามารถนำมาใช้ในการแก้ไขปรับปรุงข้อมูล ILF ได้ ไม่ว่าจะเป็นการเพิ่ม แก้ไข หรือว่า ลบข้อมูล

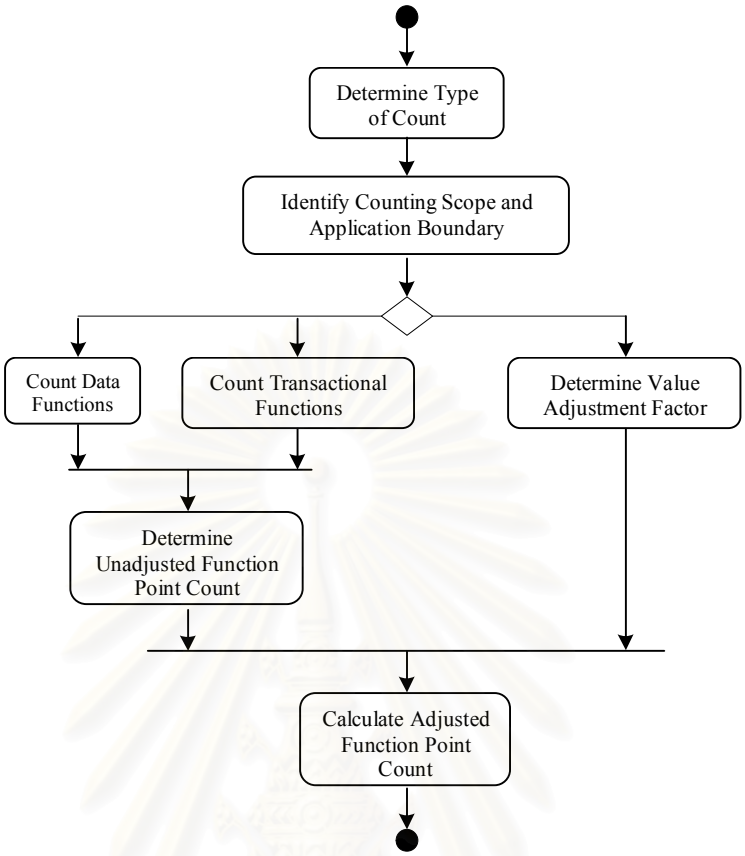
4. External Output (EO) เป็นผลลัพธ์ของการทำงานของซอฟต์แวร์ที่ออกสู่ภายนอก

5. External Inquiries (EQ) เป็นการสอบถามข้อมูลโดยผู้ใช้ ซึ่งไม่สามารถทำการปรับปรุงข้อมูลได้ เป็นการเรียกดูข้อมูลได้เพียงอย่างเดียว




รูปที่ 2.10 โมเดลของฟังก์ชันพอยต์

ถึงแม้ว่าในปัจจุบันเครื่องมือในการพัฒนาซอฟต์แวร์ และภาษาคอมพิวเตอร์จะก้าวหน้าและแตกต่างกันเพียงใด แต่คุณสมบัติและความสามารถของซอฟต์แวร์ที่พัฒนานั้นยังคงเหมือนเดิม ดังนั้นค่าฟังก์ชันพอยต์ที่ได้จึงไม่แตกต่างกัน [2] ซึ่งเป็นข้อดีที่ชัดเจนของฟังก์ชันพอยต์ที่ไม่ขึ้นกับเทคโนโลยีในการเขียนโปรแกรม ขั้นตอนการวัดแบบฟังก์ชันพอยต์ (Function Point Counting Procedure) ตามมาตรฐานของ IFPUG [4] แสดงดังรูปที่ 2.11 ซึ่งมีรายละเอียดดังต่อไปนี้




รูปที่ 2.11 แผนภาพกิจกรรมการนับฟังก์ชันพอยต์

**การกำหนดประเภทของการนับ (Determine Type of Count)** ซึ่งเป็นขั้นตอนแรกในกระบวนการนับของฟังก์ชันพอยต์ แบ่งได้เป็น 3 ประเภท

- โครงการพัฒนาซอฟต์แวร์ (Development project) เป็นการนับจำนวนฟังก์ชันการทำงานของซอฟต์แวร์ที่ได้รับการติดตั้งใช้งานเป็นครั้งแรกภายหลังจากการพัฒนาเสร็จสิ้น กล่าวคือเป็นซอฟต์แวร์เวอร์ชันแรกสุดที่ให้ผู้ใช้ได้ใช้งาน
- โครงการประเภทปรับปรุงคุณภาพซอฟต์แวร์ (Enhancement project) เป็นการนับจำนวนฟังก์ชันการทำงานของซอฟต์แวร์ที่พัฒนาต่อจากเวอร์ชันก่อนหน้า โดยการเปลี่ยนแปลงของซอฟต์แวร์อาจเป็นการเพิ่มฟังก์ชันการทำงานการแก้ไขปรับปรุงฟังก์ชันที่มีอยู่เดิม หรือ ลบฟังก์ชันที่ไม่จำเป็นหรือ ผู้ใช้ ไม่ต้องการออกไป เป็นต้น
- โครงการที่เสร็จสิ้นแล้ว (Application) เป็นจำนวนฟังก์ชันพอยต์สุทธิของซอฟต์แวร์ จำนวนฟังก์ชันพอยต์ของโครงการพัฒนาซอฟต์แวร์ เป็นฟังก์ชันพอยต์เริ่มต้นของโครงการที่เสร็จสิ้นแล้ว และจะถูกปรับปรุงทุกครั้งเมื่อจำนวนฟังก์ชันพอยต์ของโครงการประเภทปรับปรุงคุณภาพซอฟต์แวร์ได้เกิดขึ้น

ขั้นตอนกระบวนการนับฟังก์ชันพอยต์จะเหมือนกันทั้งหมด ไม่ว่าจะเป็นประเภทการนับแบบใดก็ตาม แต่สิ่งที่ต่างกันมีเพียงสูตรที่ใช้ในการคำนวณค่าฟังก์ชันพอยต์สุดท้ายในขั้นตอน Calculate Adjusted Function Point Count

**การกำหนดขอบเขตการนับ (Identify Counting Scope and Application Boundary)** เป็นขั้นตอนในการกำหนดขอบเขตว่ามีคุณสมบัติ และความสามารถของซอฟต์แวร์ใดบ้างที่เข้าข่ายในเงื่อนไขการนับ นอกจากนี้ยังต้องกำหนดขอบเขตของ Application ซึ่งแสดงให้เห็นถึงเส้นแบ่งระหว่างซอฟต์แวร์ที่ถูกวัด และ ผู้ใช้

**การนับจำนวนในส่วนที่อ้างถึงความต้องการข้อมูลของผู้ใช้ (Count Data Functions)** เป็นขั้นตอนการนับจำนวน Internal Logical Files (ILF) และ External Interface Files (EIF)

**การนับจำนวนในส่วนที่อ้างถึงการประมวลข้อมูลของผู้ใช้ (Count Transactional Functions)** เป็นขั้นตอนการนับจำนวนฟังก์ชันการทำงานที่ทำหน้าที่ดังต่อไปนี้

- External Input (EI)
- External Output (EO)
- External Inquiries (EQ)

**การนับจำนวนฟังก์ชันพอยต์ที่ยังไม่ได้ปรับค่า (Unadjusted Function Point Count :UAF)** เป็นขั้นตอนที่สะท้อนให้เห็นถึงจำนวนฟังก์ชันการทำงานทั้งหมดที่ซอฟต์แวร์เตรียมไว้ให้กับ ผู้ใช้ ซึ่งอยู่ในรูปแบบของฟังก์ชันการทำงานอะไรที่มีให้กับ ผู้ใช้ แต่ไม่ใช่ฟังก์ชันนั้นทำงานอย่างไร โดยฟังก์ชันการทำงานมีอยู่ด้วยกัน 2 ประเภทคือ Data Functions และ Transactional Functions

**การกำหนดตัวแปรปรับค่า (Value Adjustment Factor :VAF)** เป็นขั้นตอนการกำหนดค่า VAF ให้กับซอฟต์แวร์ ซึ่งค่าดังกล่าวถูกแบ่งตามลักษณะและประเภทของซอฟต์แวร์ (General System Characteristics : GSCs) ซึ่งมีอยู่ด้วยกัน 14 ประเภท ดังนี้

1. Data communication
2. Distributed functions
3. Performance objectives
4. Heavily used configuration
5. Transaction rate

6. Online data entry
7. End-user efficiency
8. Online update
9. Complex processing
10. Reusability
11. Installation ease
12. Operational ease
13. Multiple sites
14. Facilitate change

แต่ละประเภทจะมีค่า Degree of Influence ตั้งแต่ 0 ถึง 5 และค่านี้จะใช้ในการคำนวณฟังก์ชันพอยต์สุดท้ายต่อไป

**การคำนวณหาค่าฟังก์ชันพอยต์ (Calculate Adjusted Function Point Count)** เป็นขั้นตอนสุดท้ายในการคำนวณหาจำนวนของฟังก์ชันพอยต์ โดยใช้สูตรคำนวณเฉพาะ ซึ่งข้อมูลต่าง ๆ ที่ใช้ประกอบการคำนวณได้จากขั้นตอนการทำงานข้างต้น และสูตรที่ใช้เจาะจงสำหรับประเภทของโครงงาน ซึ่งได้แก่ โครงการพัฒนาซอฟต์แวร์ โครงการประเภทปรับปรุงคุณภาพซอฟต์แวร์ และโครงการที่เสร็จสิ้นแล้ว

ขั้นตอนการนับฟังก์ชันพอยต์ในวิทยานิพนธ์นี้ จะใช้ขั้นตอนในการนับแบบ UAF เนื่องจาก เป็นขั้นตอนที่ทำให้เห็นจำนวนฟังก์ชันทั้งหมดที่มีอยู่ในซอฟต์แวร์

2.1.4 ยูสเคสพอยต์ (Use Case Points : UCP)

UCP เป็นวิธีการประมาณขนาดซอฟต์แวร์จากแผนภาพ Use Case ถูกนำเสนอโดย Gustav Karner ในปี 1993 UCP เป็นเทคนิคการวัดขนาดโดยการนับจำนวนของ actors และรายการเปลี่ยนแปลง (transactions) ที่ประกอบอยู่ในลำดับเหตุการณ์การทำงานของแต่ละ use case โดยมีขั้นตอนการคำนวณค่า UCP ดังนี้ [14]

1) ระบุและกำหนดน้ำหนักให้กับ actor ซึ่งเป็นการกำหนดน้ำหนักตามลักษณะของ actor โดยที่ Sample actor จะหมายถึงระบบงานอื่น ๆ ที่ติดต่อเข้ามาผ่านส่วนต่อประสานโปรแกรมประยุกต์ (Application Programming Interface: API) ส่วน average actor จะหมายถึงระบบงานอื่นที่ติดต่อเข้ามาผ่านโพรโทคอล ตัวอย่าง เช่น ทีซีพี/ไอพี (TCP/IP) หรือหมายถึงผู้ใช้งานระบบที่ติดต่อผ่านเข้าด้วยเชิงข้อความ

(text-based) และ complex actor จะหมายถึงผู้ใช้งานระบบที่ติดต่อผ่านเข้ามาด้วยส่วนต่อประสานกราฟิก (GUI) ในการคำนวณน้ำหนักของ actor จะทำการนับจำนวน actor ในแต่ละชนิดคูณด้วยค่าปัจจัยน้ำหนัก (Weight Factor) ดังตารางที่ 2.1 และนำมาสรุปผลรวมเป็นน้ำหนักของ actor ทั้งหมด (Weighted Actor)

ตารางที่ 2.1 การกำหนดน้ำหนักของ actor

| Actor Type | Description | Weight Factor |
|---|---|---|
| Sample | Defined API | 1 |
| Average | Interactive or Protocol driven interface | 2 |
| Complex | Graphical User Interface | 3 |

2) ระบุและกำหนดน้ำหนักให้กับ use case ซึ่งเป็นการกำหนดน้ำหนักตามจำนวน transactions ที่ประกอบอยู่ในลำดับเหตุการณ์ของการทำงานในแต่ละ use case ในการคำนวณน้ำหนักของ transactions จะทำการนับจำนวน transactions ในแต่ละชนิดคูณด้วยค่าปัจจัยน้ำหนัก (Weight Factor) ดังตารางที่ 2.2 และนำมาสรุปผลรวมเป็นน้ำหนักของ use case ทั้งหมด (Weighted Use Case)

ตารางที่ 2.2 การกำหนดน้ำหนักของ use case

| Use Case Type | Description | Weight Factor |
|---|---|---|
| Sample | 3 or fewer transactions | 5 |
| Average | 4 to 7 transactions | 10 |
| Complex | Greater than 7 transactions | 15 |

3) ระบุและกำหนดน้ำหนักให้กับปัจจัยทางเทคนิค (Technical Factor) ซึ่งเป็นการกำหนดน้ำหนักปัจจัยทางเทคนิค ที่มีผลต่อการพัฒนาซอฟต์แวร์ในแต่ละโครงการ โดยการกำหนดน้ำหนักในแต่ละปัจจัยมีค่าตั้งแต่ 0 – 5 ถ้ากำหนดน้ำหนักเป็น 0 จะหมายถึงปัจจัยนี้ไม่มีผลกระทบเลย แต่ถ้ากำหนดน้ำหนักเป็น 5 หมายถึงปัจจัยนี้มีผลกระทบมาก ในการคำนวณน้ำหนักจะทำการกำหนดน้ำหนักในแต่ละปัจจัยแล้วคูณด้วยค่าปัจจัยน้ำหนัก (Weight Factor) นั้น ๆ ดังตารางที่ 2.3 และนำมาสรุปผลรวมเป็นน้ำหนักของปัจจัยทางเทคนิคทั้งหมด (TFactor) หลังจากนั้นนำไปคำนวณค่า Technical Complexity Factor (TCF) ตามสูตรดังนี้

$$TFC = 0.6 + (0.01 * TFactor)$$

ตารางที่ 2.3 การกำหนดน้ำหนักของปัจจัยทางเทคนิค

| Technical Factor | Description | Weight Factor |
|---|---|---|
| T1 | Must have a distributed solution | 2 |
| T2 | Must respond to specific performance objectives | 1 |
| T3 | Must meet end-user efficiency desires | 1 |
| T4 | Complex internal processing | 1 |
| T5 | Code must be reusable | 1 |
| T6 | Must be easy to install | 0.5 |
| T7 | Must be easy to use | 0.5 |
| T8 | Must be portable | 2 |
| T9 | Must be easy to change | 1 |
| T10 | Must allow concurrent users | 1 |
| T11 | Includes special security features | 1 |
| T12 | Must provide direct access for 3rd parties | 1 |
| T13 | Requires special user training facilities | 1 |

4) ระบุและกำหนดน้ำหนักให้กับปัจจัยทางสภาพแวดล้อม (Environmental Factor) เป็นการกำหนดน้ำหนักปัจจัยทางสภาพแวดล้อม ที่มีผลต่อการพัฒนาซอฟต์แวร์ในแต่ละโครงการ โดยการกำหนดน้ำหนักในแต่ละปัจจัยมีค่าตั้งแต่ 0 – 5 ถ้ากำหนดน้ำหนักเป็น 0 จะหมายถึงปัจจัยนี้ไม่มีผลกระทบเลย แต่ถ้ากำหนดน้ำหนักเป็น 5 หมายถึงปัจจัยนี้มีผลกระทบมาก ในการคำนวณน้ำหนักจะทำการกำหนดน้ำหนักในแต่ละปัจจัยแล้วคูณด้วยค่าปัจจัยน้ำหนัก (Weight Factor) นั้น ๆ ดังตารางที่ 2.4 และนำมาสรุปผลรวมเป็นน้ำหนักของปัจจัยทางเทคนิคทั้งหมด (EFactor) หลังจากนั้นนำไปคำนวณค่า Environmental Factor (EF) ตามสูตรดังนี้

$$EF = 1.4 + (-0.03 * EFactor)$$

ตารางที่ 2.4 การกำหนดน้ำหนักของปัจจัยทางสภาพแวดล้อม

| Environmental Factor | Description | Weight Factor |
|---|---|---|
| E1 | Familiar with FPT software process | 1 |

| Environmental Factor | Description | Weight Factor |
|---|---|---|
| E2 | Application experience | 0.5 |
| E3 | Paradigm experience (OO) | 1 |
| E4 | Lead analyst capability | 0.5 |
| E5 | Motivation | 1 |
| E6 | Stable Requirements | 2 |
| E7 | Part-time workers | -1 |
| E8 | Difficulty of programming language | -1 |

5) คำนวณค่า UCP จากค่าที่คำนวณได้จากการกำหนดน้ำหนักปัจจัยต่าง ๆ ตามสูตรดังนี้

UCP = (Weighted Actor + Weighted Use Case) * TCF * EF

จากทฤษฎีทั้ง 4 หัวข้อนี้ จะนำเทคนิคการนับฟังก์ชันพอยต์มาประยุกต์ใช้กับแผนภาพ Use Case ที่ถูกสร้างขึ้นมาตามแนวความคิดของ AOP

## 2.2 งานวิจัยที่เกี่ยวข้อง

จากงานวิจัยของ Fetcke และคณะ [2] ซึ่งเป็นงานวิจัยที่เกี่ยวข้องกับ การนำแนวความคิดเชิงวัตถุของ Jacobson มาทำการวัดขนาดซอฟต์แวร์ด้วยฟังก์ชันพอยต์ ที่ทำการพัฒนาขึ้นตามมาตรฐานของ IFPUG โดยการนำแผนภาพ Use Case ที่ได้มาจากขั้นตอนของวิธี Object-Oriented Software Engineering (OOSE) มานับฟังก์ชันพอยต์ โดยหลังจากการวิจัยแล้ว ก็ได้ผลสรุปว่า การวัดขนาดซอฟต์แวร์ด้วยฟังก์ชันพอยต์ จะไม่ขึ้นอยู่กับเทคโนโลยีที่ใช้ในการพัฒนาซอฟต์แวร์ จากการทดสอบกับเทคโนโลยี OOP ก็ได้ผลเช่นเดียวกับเทคโนโลยีอื่น ๆ นอกจากนี้ยังได้มีงานวิจัยของ Anda และคณะ [8] ซึ่งได้ทำการประเมินค่าใช้จ่ายในการพัฒนาซอฟต์แวร์จากการวัดขนาดซอฟต์แวร์ด้วยฟังก์ชันพอยต์ โดยนับค่าฟังก์ชันพอยต์จากแผนภาพ Use Case ซึ่งผลจากการทดลองการประเมินค่าใช้จ่ายจากแผนภาพ Use Case มีค่าใกล้เคียงความเป็นจริง และสามารถนำมาใช้กับการประเมินค่าใช้จ่ายในโครงงานการพัฒนาซอฟต์แวร์ได้เป็นอย่างดี

เนื่องจากในปัจจุบันเทคโนโลยี AOP เป็นเทคโนโลยีใหม่ที่ได้เกิดขึ้น มีงานวิจัยที่เกี่ยวข้องกับเทคโนโลยี AOP เพิ่มมากขึ้น แต่ยังไม่มีงานวิจัยใดที่นำเอาเทคนิคฟังก์ชันพอยต์มาใช้กับ AOP ในวิทยานิพนธ์นี้จึงได้มีแนวคิดที่จะนำเอาเทคนิคฟังก์ชันพอยต์มาวัดขนาดซอฟต์แวร์ที่ได้จากการพัฒนาด้วยเทคโนโลยี AOP โดยใช้แผนภาพ Use Case ในการนับค่าฟังก์ชันพอยต์ งานวิจัยทางด้าน AOP ในระดับการกำหนดความต้องการที่นำไปสู่การสร้างแผนภาพ Use Case ในเชิง AOP ที่เกี่ยวข้อง ได้แก่ งานวิจัยของ Jacobson และ Ng [13] ได้นำเสนอเทคนิคที่เรียกว่า Aspect-Oriented Software Development with Use Cases (AOSD/UC) มาใช้ในขั้นตอนการแบ่งแยกเอาส่วน Crosscutting concern มาออกในระดับการกำหนดความต้องการ ทั้งในส่วนของ Functional และ Non-Functional requirements แล้วนำมาสร้างแผนภาพ Use Case ในเชิง AOP ขึ้นมา

เพื่อเป็นการตรวจสอบว่าค่าฟังก์ชันพอยต์ที่ได้นั้น จะไม่ขึ้นอยู่กับเทคโนโลยีที่ใช้ในการพัฒนาซอฟต์แวร์ดังผลสรุปจากงานวิจัยของ Fetcke ในวิทยานิพนธ์นี้จึงได้ทำการเปรียบเทียบค่าฟังก์ชันพอยต์ โดยนำเทคนิคทางสถิติเข้ามาทำการตรวจสอบ ซึ่งได้นำแนวความคิดในงานวิจัยของ Abrahao และคณะ [7] ที่ได้ทำการเปรียบเทียบค่าฟังก์ชันพอยต์ที่ได้จากวิธีนับตามมาตรฐานของ IFPUG กับ วิธีที่เรียกว่า OO Method Function Points (OOmFP) ซึ่งเป็นเทคนิคที่ถูกพัฒนาขึ้นมาใช้กับเทคโนโลยี OOP โดยเฉพาะ เพื่อเป็นการทดสอบว่าสามารถนำเทคนิค OOmFP มาใช้แทนวิธีนับตามมาตรฐานของ IFPUG ได้หรือ ไม่ โดยนำเทคนิคทางสถิติเข้ามาทำการตรวจสอบค่าฟังก์ชันพอยต์ที่ได้ มาเป็นตัวอย่างในการเปรียบเทียบค่าฟังก์ชันพอยต์

นอกจากนี้ยังมีงานวิจัยอื่นๆ ที่ได้มีการประมาณค่าใช้จ่ายในการพัฒนาซอฟต์แวร์โดยการวัดขนาดซอฟต์แวร์จากแผนภาพ Use Case ได้แก่ งานวิจัยของ Edward R Carroll [14] ซึ่งได้ทำการวัดขนาดซอฟต์แวร์ด้วยวิธี Use Case Points มาประยุกต์ใช้กับองค์กรขนาดใหญ่ทางด้าน Software Engineering พบว่าสามารถนำมาประมาณค่าใช้จ่ายได้ใกล้เคียงและรวดเร็ว ในส่วนงานวิจัยของ Mohagheghi และคณะ [15] ได้ศึกษาเปรียบเทียบค่าใช้จ่ายจริงกับค่าใช้จ่ายประมาณในระบบอุตสาหกรรมที่มีการพัฒนาซอฟต์แวร์เพิ่มเติมขึ้นจากของโครงการเดิม โดยวิธี Use Case Points ซึ่งได้นำปัจจัยต่าง ๆ ที่มีผลกระทบเข้ามาประมาณค่าใช้จ่ายด้วย

# บทที่ 3

# การสร้างแผนภาพ Use Case ในเชิง AOP

ในบทนี้จะกล่าวถึงขั้นตอนการสร้างแผนภาพ Use Case ในเชิง AOP ตามแนวความคิด Aspect-Oriented Software Development with Use Cases ของ Jacobson และ Ng [13] ซึ่งได้กำหนดแนวทางการพิจารณาหา Crosscutting concern จากความต้องการของระบบทั้งในส่วนของ Functional Requirements และ Non-Functional Requirements เพื่อนำมาสร้างแผนภาพ Use Case ในขั้นตอนของการกำหนดความต้องการของระบบ

## 3.1 ศึกษาทำความเข้าใจความต้องการของระบบ

การพัฒนาซอฟต์แวร์ด้วยเทคนิค AOSD จะเริ่มต้นด้วยการพิจารณาแบ่งสิ่งที่ระบบให้ความสนใจตั้งแต่ในขั้นตอนของการกำหนดความต้องการของซอฟต์แวร์ ซึ่งจะใช้หลักเกณฑ์การพิจารณากำหนดสิ่งที่เกี่ยวข้องกับผู้ที่ได้รับผลกระทบจากระบบ (Stakeholder concerns) โดยจะนำแผนภาพ Use Case มาช่วยในการพิจารณาแบ่งสิ่งที่ระบบให้ความสนใจ เนื่องจากแผนภาพ Use Case จะแสดงความสัมพันธ์ระหว่างผู้ใช้งานระบบกับการทำงานของระบบ ทำให้สามารถมองเห็นภาพรวมทั้งหมดของระบบว่าระบบสามารถทำงานอะไรได้บ้าง ส่วนไหนในระบบมีสัมพันธ์หรือมีผลกระทบต่อกันอย่างไร และมีความสัมพันธ์กับผู้ใช้งานระบบในรูปแบบไหน

## 3.2 พิจารณา Crosscutting concern ในส่วนของ Functional Requirements

ในส่วนของ Functional Requirements ซึ่งเป็นความต้องการที่เป็นหน้าที่หลักของระบบซึ่งจะแสดงอยู่ในรูปแบบของ Use Case หลักการพิจารณาหา Crosscutting concern จาก Use case มีดังนี้

1) การระบุปัจจัยผันแปร (Variability) ในแต่ละ Use Case มีลำดับขั้นตอนของเหตุการณ์มากมาย ซึ่งเป็นการอธิบายการจัดการความผันแปรที่เกิดขึ้นตลอดทั้ง Use Case นั้น จึงมีความสำคัญในการจำแนกปัจจัยที่ทำให้เกิดความผันแปรเหล่านั้น บางปัจจัยเป็นจุดสำคัญต่อผู้ที่มีส่วนได้ส่วนเสียในระบบ ซึ่งค่าของปัจจัยผันแปรที่แตกต่างกันก็ส่งผลทำให้การทำงานของระบบก็แตกต่างด้วยเช่นกัน ดังนั้นการระบุปัจจัยผันแปรและค่าของปัจจัยผันแปรเหล่านั้น จึงเป็นส่วนสำคัญที่ช่วยให้เข้าใจความซับซ้อนของระบบมากยิ่งขึ้น

2) การจัดการดูแลปัจจัยผันแปร เนื่องจากปัจจัยเหล่านี้อาจมีส่วนเกี่ยวข้องกับหลาย ๆ Use Case จึงสามารถพิจารณาได้ว่าปัจจัยผันแปรก็คือ Aspect ของ Use Case ดังนั้นวิธีที่จะ

จัดการดูแลปัจจัยผันแปรทำได้โดยระบุและอธิบายลงไปใน Use Case Specification ในส่วนของ Alternate Flows หรือ Extension Flows ตัวอย่างเช่นระบบงานการสำรองห้องพักโรงแรมออนไลน์ มีปัจจัยผันแปร 3 ปัจจัย คือ ชนิดของลูกค้า (Customer Types), ช่องทางการให้บริการสำรองห้องพัก (Reservation Channels) และระยะเวลาที่สำรองห้องพัก (Reservation Periods) สามารถทำการระบุและอธิบายใน Use Case Specification ได้ดังนี้

**Basic Flow**

The use case begins when an individual customer wishes to make a reservation for a single period over the internet.

1…..

2…..

3…..

**Alternate Flows**

***Customer Types***

Corporate Customer

***Reservation Channels***

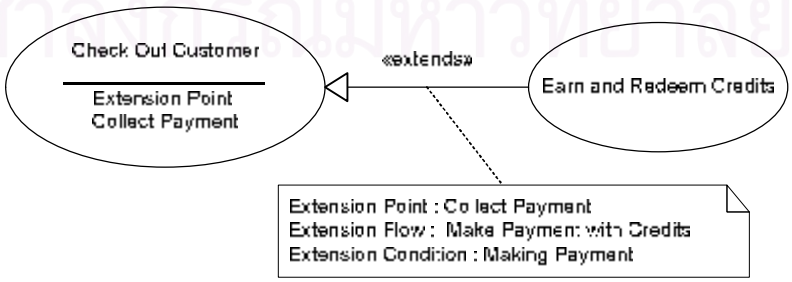
Agent Reservation

Phone Reservation

***Reservation Periods***

Multiple Periods

3) Extension Use Case ซึ่งถือได้ว่าเป็น Crosscutting Concern รูปแบบหนึ่ง จึงต้องมีการระบุ Extension Points ใน Base Use Case เพื่อให้ทราบถึง Pointcuts ที่ระบุตำแหน่งเพื่อจะไปเรียก Aspect ขึ้นมาทำงาน ตัวอย่างระบบการสำรองห้องพักโรงแรมออนไลน์ ลูกค้าสามารถชำระเงินด้วยบัตรเครดิตได้ จึงมี Earn and Redeem Credits เป็น Extension Use Case ไว้เพื่อบริการลูกค้าอีกทางเลือกนึง สามารถระบุ Extension Points ได้ดังรูปที่ 3.1

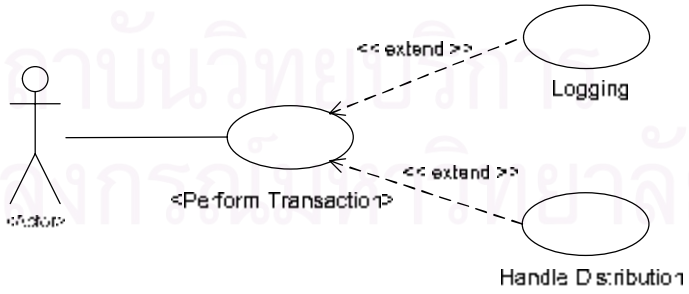



รูปที่ 3.1 Earn and Redeem Credits extends Check Out Customer

### 3.3 พิจารณา Crosscutting concern ในส่วนของ Non-Functional Requirements

ในส่วนของ Non-Function Requirements เป็นความต้องการอื่นๆ ที่ไม่ใช่หน้าที่หลักที่ระบบต้องทำ แต่เป็นคุณสมบัติอื่นๆ ที่ผู้ใช้ต้องการได้จากระบบ เช่น ความปลอดภัยของระบบ ความเชื่อถือได้ของระบบ มีความสามารถทางด้าน I/O ความสามารถในการเชื่อมต่อกับระบบอื่นๆ ตามแนวคิดของ AOSD/UC สนับสนุนให้สามารถทำการกำหนด Use Case สำหรับ Non-Functional Requirements ซึ่งเรียกว่า Infrastructure Use Case ใช้สำหรับอ้างอิงถึงกิจกรรมของระบบที่จำเป็นสำหรับการทำงานตามความต้องการของผู้ใช้งาน ซึ่งอาจจะขัดแย้งกับเทคนิค Use Case ที่แสดงให้เห็นว่าสิ่งใดที่ผู้ใช้งานสามารถใช้งานอะไรได้บ้าง และสิ่งที่ระบบตอบสนองตามการใช้งานคืออะไร ดังนั้นการพิจารณาว่า Non-Functional Requirements ในส่วนใดบ้างที่สามารถนำมาสร้างเป็น Use Case ได้ มีขั้นตอนดังนี้

1) สิ่งที่ผู้ใช้งานระบบต้องการคืออะไร และมีผลกระทบกับแผนภาพ Use Case ที่สร้างขึ้นมาจาก Functional Requirements ใดบ้าง เลือกเฉพาะที่มีผลกระทบบาง use case เท่านั้น
2) พิจารณาว่าผู้ใช้งานระบบสามารถทำการทดสอบได้ด้วยตนเอง
3) ออกแบบการทดสอบ
4) เปลี่ยนแปลงให้เป็น use case เพิ่มเติมในส่วน Functional Requirements ด้วย use case ที่สร้างขึ้นมาใหม่โดยมี <Actor> และ <Perform Transaction> use case แสดงถึง Infrastructure Requirements ของระบบ และมีความสัมพันธ์กับ use case ที่เพิ่มเติมเข้ามาเป็นแบบ <<extend>> ซึ่งสามารถช่วยมองเห็นภาพของการทำงานของระบบในส่วนของเครื่องได้ดียิ่งขึ้น ดังตัวอย่างในรูปที่ 3.2




รูปที่ 3.2 Structuring infrastructure use case

# บทที่ 4

## การนับฟังก์ชันพอยต์จากแผนภาพ Use Case

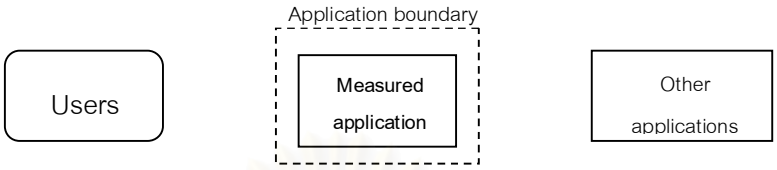
ในขั้นตอนการพัฒนาซอฟต์แวร์นั้นโดยทั่วไปจะทำการพัฒนาจนเสร็จสิ้น จึงสามารถคิดคำนวณค่าใช้จ่ายในการพัฒนาได้ บางครั้งอาจจะมีค่าใช้จ่ายที่สูงกว่าความเป็นจริงเนื่องมาจากหลายสาเหตุ การไม่มีการวางแผนและประมาณค่าใช้จ่ายเบื้องต้นก่อนการพัฒนาจริงก็เป็นสาเหตุหนึ่ง การนำเทคนิคการวิเคราะห์ฟังก์ชันพอยต์มาใช้ในการวัดขนาดซอฟต์แวร์ ก็เป็นแนวทางหนึ่งในการประมาณค่าใช้จ่ายเบื้องต้น และวางแผนได้ว่าสำหรับโครงงานพัฒนาซอฟต์แวร์นั้น ๆ ต้องใช้นักพัฒนาระบบจำนวนเท่าไหร่ และเวลาที่ใช้ในการพัฒนาเป็นอย่างไร โดยการนำเทคนิคการวิเคราะห์ฟังก์ชันพอยต์มาประยุกต์ใช้ในขั้นตอนการวิเคราะห์ และออกแบบตามความต้องการของผู้ใช้ ซึ่งจะทราบการทำงานของซอฟต์แวร์ทั้งหมดว่า จะต้องพัฒนาซอฟต์แวร์ให้มีความสามารถปฏิบัติงานใดได้บ้าง ในปัจจุบันการพัฒนาซอฟต์แวร์ตามแนวความคิดเชิงวัตถุเป็นที่นิยมแพร่หลายในหลายองค์กร และมีแบบจำลองที่เรียกว่า UML มาช่วยในการพัฒนาซอฟต์แวร์ ซึ่งจะประกอบไปด้วยหลาย ๆ แผนภาพ แผนภาพ Use Case ก็เป็นแผนภาพหนึ่งที่อธิบายขอบเขตของการทำงานของซอฟต์แวร์ และเกิดขึ้นในขั้นตอนเริ่มต้นของการวิเคราะห์ระบบ

เทคนิคฟังก์ชันพอยต์เป็นการวัดขนาดซอฟต์แวร์ ที่ได้มาจากมุมมองของการใช้งานของผู้ใช้ ส่วน Use Case นั้นก็เป็นแผนภาพที่อธิบายว่าระบบมีส่วนประกอบอะไรบ้าง มีความสัมพันธ์กันอย่างไร ช่วยในการพัฒนาตามความต้องการของผู้ใช้ ซึ่งจะเห็นได้ว่าทั้งเทคนิคการวิเคราะห์ฟังก์ชันพอยต์ และแผนภาพ Use Case นั้น ต่างก็ไม่ขึ้นอยู่กับเทคโนโลยีที่ใช้ในการพัฒนาซอฟต์แวร์ ข้อดีในการนำเทคนิคการวิเคราะห์ฟังก์ชันพอยต์มาประยุกต์ใช้กับแผนภาพ Use Case นั่นคือ ทำให้การประมาณค่าใช้จ่ายของโครงงานจะมีความแม่นยำมากขึ้น [9] เนื่องจากแผนภาพ Use Case แสดงให้เห็นถึงความต้องการของผู้ใช้ว่า ผู้ใช้สามารถใช้งานระบบทำอะไรได้บ้าง ซึ่งเป็น Requirements Documentation ที่เป็นพื้นฐานในการจัดเตรียมขอบเขตในการทดสอบ และคู่มือการใช้งานของผู้ใช้ต่อไป ดังนั้นการนำเทคนิคการวิเคราะห์ฟังก์ชันพอยต์มาประยุกต์ใช้กับแผนภาพ Use Case เพื่อใช้ประมาณค่าใช้จ่ายของโครงงานน่าจะสามารถประเมินค่าใช้จ่ายในการพัฒนาซอฟต์แวร์ได้ค่าใกล้เคียงกับค่าใช้จ่ายจริงที่จะเกิดขึ้นอย่างแน่นอน

ขั้นตอนการนับฟังก์ชันพอยต์จากแผนภาพ Use Case ตามแนวคิดจากงานวิจัยของ Fetcke และคณะ [2] มีขั้นตอนดังนี้

## 4.1 การระบุขอบเขตการนับ

การระบุขอบเขตการนับ (Identification of the counting boundary) เป็นการแบ่งขอบเขต Application ที่จะทำการวัดขนาดออกจาก Application อื่น ๆ ดังรูปที่ 4.1




รูปที่ 4.1 การระบุขอบเขตการนับ

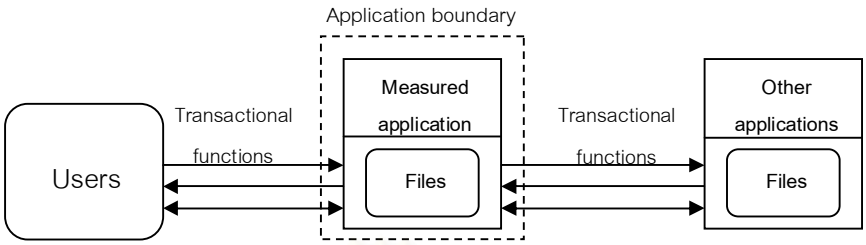
ตามแนวคิดของฟังก์ชันพอยต์นั้น Application boundary จะอธิบายถึงขอบเขตที่กั้นระหว่างซอฟต์แวร์ที่จะทำการวัดขนาดกับผู้ใช้งานระบบ ซึ่งมีแนวความคิดเช่นเดียวกับแผนภาพ Use Case ที่แยก actors ไว้ภายนอก Application แต่ actors ในแผนภาพ Use Case ก็ไม่ได้หมายถึงผู้ใช้งานระบบในแนวความคิดของฟังก์ชันพอยต์เสมอไป เนื่องจาก actors มีความหมายที่เปิดกว้างกว่าผู้ใช้งานระบบ ซึ่ง actors นั้นสามารถเป็นไปได้ทั้งผู้ใช้งานระบบ และ other applications หรือไม่เป็นทั้งสองอย่างก็ได้ เนื่องจาก actors ที่ถูกนำเสนอในแผนภาพ Use Case จะถูกนำเสนอในมุมมองของการทำงานของระบบ ทำให้ไม่สามารถสรุปได้ว่า Actors นั้นจะเป็นผู้ใช้งานระบบหรือเป็น other applications ได้เสมอไป หลักการในการพิจารณาในขั้นตอนของการระบุขอบเขตในการนับฟังก์ชันพอยต์ ดังนี้

- actors ที่เป็นคนจะหมายถึงผู้ใช้งานระบบ
- actors ที่เป็นระบบงานอื่นที่ไม่ได้มีการทำงานภายในระบบนั้นมาเกี่ยวข้องกับระบบที่จะทำการวัดขนาด จะหมายถึง external applications
- actors ที่ไม่ใช่คนแต่เป็นส่วนประกอบหนึ่งในระบบ เช่น printer ก็จะไม่ถูกนำมาพิจารณา

นอกจากนี้ actors subclass ที่เกิดจากการสืบทอดคุณสมบัติมาจาก actors superclass และมีการโต้ตอบกับระบบ จะนำมาพิจารณาตามหลักการข้างต้นด้วยเช่นกัน [10]

## 4.2 การระบุรายการในขอบเขต

การระบุรายการในขอบเขต (Identification of items within boundary) ในเทคนิคการวิเคราะห์ฟังก์ชันพอยต์จะแบ่งรายการที่จะทำการนับออกเป็น 2 ประเภท ได้แก่ Transaction functions และ Files ดังแสดงในรูปที่ 4.2




รูปที่ 4.2 การระบุรายการภายในขอบเขต

4.2.1 Transaction functions เป็นส่วนที่อ้างถึงการประมวลผลข้อมูลของผู้ใช้ระบบงาน ซึ่งประกอบด้วย External Input (EI), External Output (EO) และ External Inquiries (EQ) use cases ในแผนภาพ Use Case ซึ่งแสดงให้เห็นถึงการทำงานของระบบและมีความสัมพันธ์อย่างไรกับ actors ก็จะมีความสอดคล้องกับ Transactions ตามเทคนิคของฟังก์ชันพอยต์ ดังนั้น ทุก ๆ use cases ในระบบ จะถูกนำมาพิจารณาตามหลักการที่ถูกนำเสนอจากงานวิจัยของ Iorio [11] และงานวิจัยของ Pace และคณะ [10] ดังนี้

- use cases ที่ถูกนำมาพิจารณาจะต้องมีความสัมพันธ์โดยตรงกับ actors ที่ถือว่าเป็นผู้ใช้งานระบบ หรือ external application ตามหลักการในขั้นตอนของการระบุขอบเขต
- use cases ที่มีความสัมพันธ์กับ use cases ที่ถูกคัดเลือกแล้ว และมีชนิดความสัมพันธ์เป็น “extends” และ “include” จะถูกนำไปพิจารณาด้วย เนื่องจาก use cases เหล่านั้นอาจจะส่งผลกระทบกับการทำงานของระบบและ actors หรือไม่ก็ได้ [10]
- use cases เกิดจากการสืบทอดคุณสมบัติ (use case generalization) มาจาก use cases ที่ถูกคัดเลือกแล้ว จะถูกนำมาพิจารณาด้วย แต่จะทำการนับ transaction เฉพาะที่ยังไม่ได้ถูกนับใน use cases superclass เท่านั้น [10]
- use cases อื่น ๆ จะไม่ถูกนำมาพิจารณา

4.2.2 Files เป็นส่วนที่อ้างถึงความต้องการข้อมูลของผู้ใช้งานระบบ ซึ่งประกอบด้วย Internal Logical Files (ILF) และ External Interface Files (EIF) ในส่วนนี้จะนำ Requirements Document มาพิจารณาประกอบร่วมกับแผนภาพ Use Case ด้วย นั้นก็คือ

Scenario ซึ่งเป็นลำดับเหตุการณ์ที่จะเกิดขึ้นในระบบ ทำให้มองเห็นข้อมูลที่มีส่วนเกี่ยวข้องกับระบบได้ชัดเจนยิ่งขึ้น

## 4.3 การกำหนดประเภทของรายการ

การกำหนดประเภทของรายการ (Determination of types of the items) ในเทคนิคการวิเคราะห์ฟังก์ชันพอยต์จะแบ่งรายการที่จะทำการนับออกเป็น 5 ประเภทหลัก โดยในแต่ละรายการก็จะมีองค์ประกอบต่าง ๆ ที่แตกต่างกัน ตามรายละเอียดที่คุณเมสินีได้กล่าวไว้ [12] ได้แก่

- องค์ประกอบข้อมูล (Data Element Type : DET) คือ ฟิลด์ของไฟล์ข้อมูลที่สนใจในแต่ละฟิลด์
- เรคคอร์ดข้อมูล (Record Element Type : RET) คือ กลุ่มข้อมูล หรือกลุ่มย่อยของ DET หรือการนับประเภทของเรคคอร์ดข้อมูลที่เกี่ยวข้องสัมพันธ์กับฟังก์ชันที่สนใจ
- ประเภทไฟล์ข้อมูลอ้างอิง (File Type Referenced : FTR) คือ ประเภทของไฟล์ที่เกี่ยวข้องในแต่ละ Transaction functions

ซึ่งสามารถสรุปความสัมพันธ์ได้ดังตารางที่ 4.1 ดังนี้

ตารางที่ 4.1 ตารางความสัมพันธ์ของประเภทรายการและองค์ประกอบ

| Component | DET | RET | FTR |
|---|---|---|---|
| Internal Logical Files (ILF) | ✓ | ✓ | |
| External Interface File (EIF) | ✓ | ✓ | |
| External Inputs (EI) | ✓ | | ✓ |
| External Outputs (EO) | ✓ | | ✓ |
| External Inquiries (EQ) | ✓ | | ✓ |

การกำหนดประเภทรายการภายใต้ขอบเขตที่ได้ระบุไว้จากขั้นตอนที่ 2 จะทำการกำหนดตามกฎของ IFPUG [12] ดังนี้

4.3.1 Internal Logical Files (ILF)

- กลุ่มของข้อมูลที่มีความสามารถทางตรรกะ (Logical) กำหนดไว้ในส่วนของผู้ใช้งานระบบ
- กลุ่มของข้อมูลที่ได้รับการเก็บรักษา (Maintain) ไว้ตลอดกระบวนการภายในระบบงาน

4.3.2 External Interface Files (EIF)

- กลุ่มของข้อมูลที่มีความสามารถทางตรรกะ (Logical) กำหนดไว้ในส่วนของผู้ใช้งานระบบ
- กลุ่มของข้อมูลที่ได้รับอ้างอิงจากระบบงาน
- กลุ่มของข้อมูลจะต้องได้รับการเก็บรักษา (Maintain) ไว้ โดยระบบงานภายนอก

จากตารางที่ 4.1 แสดงให้เห็นว่า ILF และ EIF ยังมีองค์ประกอบย่อยอีก 2 อย่าง คือ DET และ RET ที่ต้องนำมาพิจารณาตามกฎดังนี้

DET

- นับแต่ละข้อมูลแต่ละ DET สำหรับแต่ละฟิลด์ข้อมูล
- หากระบบงานตั้งแต่ 2 ระบบขึ้นไปมีการเก็บข้อมูล หรือ อ้างอิงไปยัง ILF หรือ ELF เดียวกัน แต่แยก DET ให้นับ DET เฉพาะที่มีการเรียกใช้แต่ละระบบงาน
- นับ DET หนึ่ง DET สำหรับแต่ละกลุ่มข้อมูลที่ถูกกำหนด โดยผู้ใช้งานระบบเพื่อทำการสร้างความสัมพันธ์กับ ILF หรือ ELF อื่น (อ้างอิงถึงกรณีการนับ Foreign Key)

RET

- นับหนึ่ง RET สำหรับแต่ละกลุ่มย่อยของ LIF หรือ ELF
- ถ้าหากไม่มีกลุ่มย่อย ให้นับเป็น 1 RET

4.3.3 External Input (EI)

- ข้อมูลจะต้องได้รับมาจากระบบงานภายนอก
- ข้อมูลอย่างน้อย 1 ILF จะต้องมีการทำการเก็บรักษา (Maintain) ไว้ตลอดกระบวนการพื้นฐาน หรือ กระบวนการย่อยสุดภายในระบบงาน
- กระบวนการจะต้องเป็นหน่วยที่เล็กที่สุดของกิจกรรมในระบบงานที่มีความสำคัญต่อผู้ใช้ระบบงาน
- กระบวนการจะต้องมีลักษณะที่ทำงานได้ภายในตัวเอง และแยกจากลักษณะทางธุรกิจ

จากตารางที่ 4.1 แสดงให้เห็นว่า EI ยังมีองค์ประกอบย่อยอีก 2 อย่าง คือ DET และ FTR ที่ต้องนำมา พิจารณาตามกฎดังนี้

DET

- นับหนึ่ง DET สำหรับแต่ละฟิลด์ของข้อมูลที่ได้รับการกำหนดไว้ ในลักษณะที่มีการกำหนดมาจากความต้องการของซอฟต์แวร์ และไม่ซ้ำ รวมทั้งในส่วนของ Foreign Key ที่เกิดขึ้นระหว่างระบบงาน เพื่อให้กระบวนการย่อยสุดของระบบงานได้ทำงานอย่างสมบูรณ์
- ไม่นับรวม DET สำหรับฟิลด์ข้อมูลที่ผู้ใช้งานระบบไม่ได้ทำการกรอกข้อมูล เช่น ฟิลด์วันที่ที่ระบบขึ้นให้อัตโนมัติ
- นับหนึ่ง DET สำหรับแต่ละความสามารถในการส่งข้อความตอบรับ ไปยังภายนอกของระบบงานเพื่อแจ้งเตือนความผิดพลาด หรือ ยืนยันว่ากระบวนการได้ทำงานเสร็จสมบูรณ์
- นับหนึ่ง DET สำหรับแต่ละความสามารถในการกำหนดการทำงานที่จัดการโดย EI

RET

- นับหนึ่ง FTR สำหรับแต่ละประเภทของ ILF ที่ได้รับการเก็บรักษาไว้ภายใต้กระบวนการของ EI
- นับหนึ่ง FTR สำหรับแต่ละประเภทของ ILF หรือ EIF ที่ได้รับการอ้างอิงจากกระบวนของ EI
- นับเพียงหนึ่ง FTR สำหรับแต่ละ ILF ทั้งที่ได้รับการอ้างอิง หรือเรียกอ่านหรือเก็บรักษาไว้ภายใต้ EI

4.3.4 External Output (EO)

- กลุ่มของข้อมูลนั้นจะต้องอยู่ภายนอกขอบเขตของระบบงาน
- ลอจิกของกระบวนการย่อยสุดของ EO จะต้องกระทำการ
  - บรรจุสูตรทางคณิตศาสตร์ หรือการคำนวณอย่างน้อยหนึ่งสูตร
  - สร้างข้อมูลแบบสืบทอดต่อกันมา (Derived Data)
  - เก็บรักษาข้อมูลอย่างน้อย 1 ILF
  - เปลี่ยนแปลงพฤติกรรมของระบบงาน
- กระบวนการจะต้องเป็นหน่วยที่เล็กที่สุดของกิจกรรมในระบบงานที่มีความสำคัญต่อผู้ใช้ระบบงาน
- กระบวนการจะต้องมีลักษณะที่ทำงานได้ภายในตัวเองและแยกจากลักษณะทางธุรกิจ

จากตารางที่ 4.1 แสดงให้เห็นว่า EO ยังมีองค์ประกอบย่อยอีก 2 อย่าง คือ DET และ FTR ที่ต้องนำมาพิจารณาตามกฎดังนี้

DET

- นับหนึ่ง DET สำหรับแต่ละฟิลด์ของข้อมูลที่ได้รับการกำหนดไว้ ในลักษณะที่มีการกำหนดมาจากความต้องการของซอฟต์แวร์ และไม่ซ้ำ รวมทั้งในส่วนของ Foreign Key ที่เกิดขึ้นระหว่างระบบงาน เพื่อให้กระบวนการย่อยสุดของระบบงานได้ทำงานอย่างสมบูรณ์
- นับหนึ่ง DET สำหรับแต่ละฟิลด์ที่อยู่ภายนอกระบบ หาก DET นั้น ๆ เกิดขึ้นทั้งภายนอก และเข้ามาสู่ระบบงานให้นับเพียงครั้งเดียว
- นับหนึ่ง DET สำหรับแต่ละความสามารถในการส่งข้อความตอบรับ ไปยังภายนอกของระบบงานเพื่อแจ้งเตือนความผิดพลาด หรือ ยืนยันว่ากระบวนการได้ทำงานเสร็จสมบูรณ์
- นับหนึ่ง DET สำหรับแต่ละความสามารถในการกำหนดการทำงานที่จัดการโดย EO
- นับหนึ่ง DET สำหรับฟิลด์ข้อมูลที่มีการจัดเก็บหลายที่ แต่เป็นฟิลด์เดียวกัน

RET

- นับหนึ่ง FTR สำหรับแต่ละประเภทของ ILF ที่ได้รับการเก็บรักษาไว้ภายใต้กระบวนการของ EO
- นับหนึ่ง FTR สำหรับแต่ละประเภทของ ILF หรือ EIF ที่ได้รับการอ้างอิงจากกระบวนของ EO
- นับเพียงหนึ่ง FTR สำหรับแต่ละ ILF ทั้งที่ได้รับการอ้างอิง หรือเรียกอ่านหรือเก็บรักษาไว้ภายใต้ EO

4.3.5 External Inquiries (EQ)

- กลุ่มของข้อมูลนั้นจะต้องอยู่ภายนอกขอบเขตของระบบงาน
- กลุ่มของข้อมูลนั้นจะมีการเรียกดึงข้อมูลอย่างน้อยหนึ่ง ILF หรือ EIF
- ลอจิกของกระบวนการย่อยสุดของ EQ จะต้องไม่
  - บรรจุสูตรทางคณิตศาสตร์ หรือการคำนวณอย่างน้อยหนึ่งสูตร
  - สร้างข้อมูลแบบสืบทอดต่อกันมา (Derived Data)
  - เก็บรักษาข้อมูลอย่างน้อย 1 ILF

- เปลี่ยนแปลงพฤติกรรมของระบบงาน

- กระบวนการจะต้องเป็นหน่วยที่เล็กที่สุดของกิจกรรมในระบบงานที่มีความสำคัญต่อผู้ใช้ระบบงาน
- กระบวนการจะต้องมีลักษณะที่ทำงานได้ภายในตัวเอง และแยกจากลักษณะทางธุรกิจ

จากตารางที่ 4.1 แสดงให้เห็นว่า EQ ยังมีองค์ประกอบย่อยอีก 2 อย่าง คือ DET และ FTR ที่ต้องนำมาพิจารณาตามกฎดังนี้

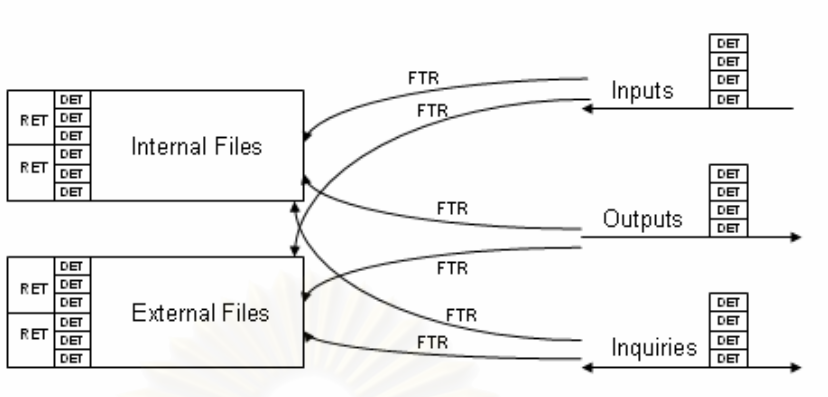
<u>DET</u>

- นับหนึ่ง DET สำหรับแต่ละฟิลด์ของข้อมูลที่ได้รับการกำหนดไว้ ในลักษณะที่มีการกำหนดมาจากความต้องการของซอฟต์แวร์ และไม่ซ้ำ รวมทั้งในส่วนของ Foreign Key ที่เกิดขึ้นระหว่างระบบงาน เพื่อให้กระบวนการย่อยสุดของระบบงานได้ทำงานอย่างสมบูรณ์
- นับหนึ่ง DET สำหรับแต่ละฟิลด์ที่อยู่ภายนอกระบบ หาก DET นั้น ๆ เกิดขึ้นทั้งภายนอก และเข้ามาสู่ระบบงานให้นับเพียงครั้งเดียว
- นับหนึ่ง DET สำหรับแต่ละความสามารถในการส่งข้อความตอบรับ ไปยังภายนอกของระบบงานเพื่อแจ้งเตือนความผิดพลาด หรือ ยืนยันว่ากระบวนการได้ทำงานเสร็จสมบูรณ์
- นับหนึ่ง DET สำหรับแต่ละความสามารถในการกำหนดการทำงานที่จัดการโดย EQ
- นับหนึ่ง DET สำหรับฟิลด์ข้อมูลที่มีการจัดเก็บหลายที่ แต่เป็นฟิลด์เดียวกัน

<u>RET</u>

- นับหนึ่ง FTR สำหรับแต่ละประเภทของ ILF ที่ได้รับการเก็บรักษาไว้ภายใต้กระบวนการของ EQ
- นับหนึ่ง FTR สำหรับแต่ละประเภทของ ILF หรือ EIF ที่ได้รับการอ้างอิงจากกระบวนของ EQ
- นับเพียงหนึ่ง FTR สำหรับแต่ละ ILF ทั้งที่ได้รับการอ้างอิง หรือเรียกอ่านหรือเก็บรักษาไว้ภายใต้ EQ

## 4.4 การกำหนดน้ำหนักของปัจจัยสำคัญ




รูปที่ 4.3 การกำหนดน้ำหนักของปัจจัยสำคัญ

จากรูปที่ 4.3 เป็นการกำหนดน้ำหนักของปัจจัยสำคัญ (Weighting factors) โดยค่า DET, RET และ FTR เป็นค่าที่เกิดจากการวัด ซึ่งได้จากการนับจำนวนของ transaction functions และ files ในข้อที่ 4.3 โดยทำการกำหนดลำดับ (Ranking) และอัตรา (Rating) ที่ใช้ในการนับ ดังนี้

ใช้ตารางที่ 4.2 ในการพิจารณาลำดับของ External Inputs

ตารางที่ 4.2 ลำดับของ External Inputs

| FTR | DET | | |
|---|---|---|---|
| | 1 - 4 | 5 - 15 | >15 |
| 0 -1 | Low | Low | Average |
| 2 | Low | Average | High |
| >=3 | Average | High | High |

ใช้ตารางที่ 4.3 ในการพิจารณาลำดับของ External Output/External Inquiries

ตารางที่ 4.3 ลำดับของ External Output/External Inquiries

| FTR | DET | | |
|---|---|---|---|
| | 1 - 5 | 6 - 19 | >19 |
| 0 -1 | Low | Low | Average |
| 2 | Low | Average | High |
| >3 | Average | High | High |

ใช้ตารางที่ 4.4 ในการพิจารณาอัตราสำหรับ Transaction Function

ตารางที่ 4.4 อัตราสำหรับ Transaction Function

| Ranking | DET | | |
|---|---|---|---|
| | External Input | External Output | External Inquiry |
| Low | 3 | 4 | 3 |
| Average | 4 | 5 | 4 |
| High | 6 | 7 | 6 |

ใช้ตารางที่ 4.5 ในการพิจารณาลำดับของ ILF และ EIF

ตารางที่ 4.5 ลำดับของ ILF และ EIF

| RET | DET | | |
|---|---|---|---|
| | 1 – 19 | 20 – 50 | >50 |
| 1 | Low | Low | Average |
| 2 - 5 | Low | Average | High |
| >5 | Average | High | High |

ใช้ตารางที่ 4.6 ในการพิจารณาอัตราสำหรับ ILF และ EIF

ตารางที่ 4.6 อัตราสำหรับ ILF และ EIF

| Ranking | DET | |
|---|---|---|
| | Internal Logical File | External Interface File |
| Low | 7 | 5 |
| Average | 10 | 7 |
| High | 15 | 10 |

จากการกำหนดลำดับและอัตราข้างต้น นำมาคำนวณค่า UAF ดังตารางที่ 4.7

ตารางที่ 4.7 การคำนวณค่า Unadjusted Function Points

| Component Type | Component Complexity | | | |
|---|---|---|---|---|
| | Low | Average | High | Total |
| External Inputs | x 3 = | x 4 = | x 6 = | |
| External Outputs | x 4 = | x 5 = | x 7 = | |
| External Inquiries | x 3 = | x 4 = | x 6 = | |
| Internal Logical Files | x 7 = | x 10 = | x 15 = | |
| External Interface Files | x 5 = | x 7 = | x 10 = | |
| Total Number Of Unadjusted Function Points | | | | |

# บทที่ 5

# การดำเนินการวิจัย

## 5.1 ระบบงานที่ใช้เป็นกรณีศึกษา

การวิจัยนี้ได้ทำการทดลองโดยใช้ระบบงานของธนาคารกรุงไทย และบริษัทในเครือที่มีการสร้างแผนภาพ Use Case ไว้แล้วมาเป็นกรณีศึกษา จำนวนทั้งหมด 12 ระบบงานดังนี้

5.1.1 ระบบงานที่ 1 ระบบงาน Corporate Cash Management System (CCMS) เป็นระบบงานที่ให้บริการด้านการทำธุรกรรมทางการเงิน (Financial) และการรับชำระเงิน (payment) สำหรับลูกค้าที่เป็นบริษัท โดยรูปแบบที่นำเสนอจะผ่านช่องทางอินเทอร์เน็ตในรูปแบบเว็บสำหรับการให้บริการติดต่อกับผู้ใช้งานระบบ

5.1.2 ระบบงานที่ 2 ระบบงานสินเชื่อเพื่อการซื้อสินค้า (Asset Financing) ระบบการให้สินเชื่อเพื่อการซื้อสินค้าของบริษัท KTC เมื่อลูกค้าเลือกการผ่อนชำระค่าสินค้าใด ๆ บริษัท KTC จะเป็นผู้ชำระค่าสินค้าเต็มจำนวนให้กับร้านค้า แล้วจึงเรียกเก็บเงินต้นและดอกเบี้ยจากลูกค้าเป็นรายงวด

5.1.3 ระบบงานที่ 3 ระบบงาน GHB Mortgage Card เป็นระบบให้บริการบัตรเครดิตสำหรับลูกค้าธนาคารอาคารสงเคราะห์ เพื่อใช้ในการบริหารจัดการบัตร รวมถึงการอนุมัติการใช้บัตรในการซื้อสินค้า และการออกรายงานต่างๆ

5.1.4 ระบบงานที่ 4 ระบบงาน Human Resource Management System (HRMS) เป็นระบบงานการบริหารทรัพยากรบุคคล เพื่อใช้ในการจัดเก็บข้อมูลพนักงานและนำข้อมูลมาใช้ในการบริหารบุคคลากรของบริษัทให้มีประสิทธิภาพสูงสุด

5.1.5 ระบบงานที่ 5 ระบบงาน KTB Cash Inventory Management System (KCIM) เป็นระบบที่ใช้จัดการบริหารเงินสด (Cash Inventory Management : CIM) รองรับการจัดการขนส่งระหว่าง CIM ศูนย์ รวมทั้งสาขาหรือลูกค้าอื่นๆ

5.1.6 ระบบงานที่ 6 ระบบงานจองซื้อหุ้นหรือหน่วยลงทุน (Selling Agent System) สำหรับธนาคารกรุงไทย ใช้เป็นเครื่องมือในการให้บริการจองซื้อและชำระเงินที่สาขาของธนาคาร

5.1.7 ระบบงานที่ 7 ระบบงาน KTC VISA Shape Card เป็นระบบการให้บริการด้านบัตรเครดิตที่เน้นความสะดวกสบายไว้บริการลูกค้า ซึ่งลูกค้ามีความต้องการที่แตกต่างกันตาม Life Style จุดเด่นของบัตรชุดนี้อยู่ที่การออกแบบบัตรให้โดดเด่นจากบัตรเครดิตทั่วไป โดยมีรูปทรงที่ไม่ใช่สี่เหลี่ยม และภาพบนหน้าบัตรที่ออกแบบให้เป็นภาพชุด โดยในแต่ละแบบจะมีสิทธิพิเศษที่แตกต่างกัน

5.1.8 ระบบงานที่ 8 ระบบงาน Teller Payment (Lotto) เป็นระบบการให้บริการชำระค่าสลากเสรีผ่านเคาน์เตอร์สาขาธนาคาร โดยสาขารับชำระเงินตามฐานข้อมูลของสำนักงานสลากฯ พร้อมออกหลักฐานการชำระเงินให้ผู้ได้สิทธิจองซื้อนำไปขอรับสลากที่สำนักงานสลากฯ

5.1.9 ระบบงานที่ 9 ระบบงาน Time Sheet System เป็นระบบการบันทึกข้อมูลการปฏิบัติงานของพนักงาน ที่เป็นกิจกรรมสำหรับการพัฒนาระบบงานใหม่ (New Development), การปรับปรุงระบบงาน (Enhancement) และการบำรุงรักษาระบบงานเดิม (Maintenance) รวมทั้งการปฏิบัติงานที่เป็นการสนับสนุนหน่วยงานภายในบริษัทฯ

5.1.10 ระบบงานที่ 10 ระบบงาน Real-Time Transaction Monitoring เป็นระบบที่ใช้ในการตรวจสอบรายการเคลื่อนไหวที่เข้ามาในระบบงาน Core banking System โดยระบบจะทำการตรวจสอบรายการที่เข้ามาจากช่องทางต่าง ๆ ว่ามีเงื่อนไขตามที่กำหนดหรือไม่ ถ้าเข้าเงื่อนไขจะทำการส่งข้อความไปยังผู้รับผิดชอบทราบทางอีเมล หรือรายงาน เป็นต้น

5.1.11 ระบบงานที่ 11 ระบบงาน Financial Management System (FMS) ระบบข้อมูลสำคัญในการจัดทำงบทดลอง งบการเงิน งบกำไรขาดทุน และรายงานด้านการเงินและบัญชีของธนาคาร

5.1.12 ระบบงานที่ 12 ระบบงาน KTC Happy Mobile Airtime Refill เป็นระบบงานที่ให้บริการเติมเงินจากบัญชีเงินฝากธนาคารกรุงไท และบัตรเครดิต KTC ซึ่งลูกค้าที่ได้ผ่านการลงทะเบียนจะสามารถใช้บริการระบบเติมเงินสดได้อย่างสะดวก

## 5.2 ดำเนินการวิจัย

5.2.1 ผู้วิจัยได้ทำการศึกษาทฤษฎีการสร้างแผนภาพ Use Case ในเชิง AOP โดยนำกรณีศึกษาทั้ง 12 ระบบมาทดสอบ เริ่มต้นด้วยการทำความเข้าใจความต้องการของแต่ละระบบจากเอกสาร Requirements Specification เพื่อให้ทราบถึงส่วนที่เกี่ยวข้องกับผู้ที่ได้รับผลกระทบจากระบบ (Stakeholder concerns) แล้วทำการพิจารณาหา Crosscutting concern ทั้งในส่วนของ Functional Requirements และ Non-Functional Requirement จากนั้นจึงได้ทำการสร้างแผนภาพ Use Case ในเชิง AOP ขึ้นมาใหม่ทั้ง 12 ระบบ

5.2.2 ผู้วิจัยได้ทำการศึกษาทฤษฎีการวิเคราะห์ฟังก์ชันพอยต์ โดยเฉพาะในส่วนที่เกี่ยวข้องกับการนับฟังก์ชันพอยต์จากแผนภาพ Use Case และได้กำหนดหลักการพิจารณา actor กับ use cases จากงานวิจัยที่เกี่ยวข้อง เพื่อใช้เป็นเกณฑ์สำหรับการวิจัยครั้งนี้ หลังจากนั้นก็นับจำนวนฟังก์ชันพอยต์ตามหลักการของ IFPUG ที่นำมาประยุกต์ใช้กับแผนภาพ Use Case โดยแบ่งกลุ่มกรณีศึกษาออกเป็น 2 กลุ่ม คือ กลุ่มของแผนภาพ Use Case เชิง OOP กับ กลุ่มของแผนภาพ Use Case เชิง AOP แล้วจึงเริ่มทำการนับจำนวนฟังก์ชันพอยต์ทีละกลุ่มจนครบ

5.2.3 ผู้วิจัยได้เตรียมข้อมูลเพื่อสำหรับการฝึกอบรมกลุ่มตัวอย่างในการทดลองซึ่งจะทำการสร้างแผนภาพ Use Case ในเชิง AOP โดยใช้ทฤษฎีเดียวกันคือทฤษฎี AOSD/UC จากแนวความคิดของ Jacobson จำนวน 12 คน ซึ่งเป็นผู้ที่เกี่ยวข้องอยู่ในสายงานคอมพิวเตอร์ นอกจากนี้ยังได้ทำการศึกษาเทคนิคการวัดขนาดซอฟต์แวร์ด้วยวิธี Use Case Points เพื่อมาใช้ในการทดลองอีกด้วย

5.2.4 นำค่าฟังก์ชันพอยต์ที่ได้มาวิเคราะห์ข้อมูลเชิงสถิติ เพื่อทำการทดสอบสมมติฐานที่ตั้งไว้แล้วจึงทำการสรุปผลการวิจัย

## 5.3 ตัวอย่างการสร้างแผนภาพ Use Case ในเชิง AOP

ตัวอย่างของระบบงานที่ใช้เป็นกรณีศึกษา เพื่อทำการสร้างแผนภาพ Use Case ในเชิง AOP ได้แก่ ระบบงาน Corporate Cash Management System (CCMS) ซึ่งเป็นระบบงานที่ให้บริการด้านการทำธุรกรรมทางการเงิน (Financial) และ การรับชำระเงิน (payment) สำหรับลูกค้าที่เป็นบริษัทของธนาคารกรุงไทย ในการศึกษาทำความเข้าใจของระบบจะพิจารณาจากกลุ่มผู้ใช้งานระบบว่า ผู้ใช้งานระบบกลุ่มใดมีความต้องการแบบไหน ในระบบงาน CCMS แบ่งกลุ่มผู้ใช้งานระบบในการปฏิษัติงาน เป็น 3 กลุ่ม ดังนี้

- กลุ่มลูกค้าของธนาคาร ซึ่งได้แก่ บริษัท ห้างร้าน ส่วนข้าราชการ และรัฐวิสาหกิจ ซึ่งเป็นผู้ใช้งานระบบหลัก มีความต้องการให้ระบบสามารถทำรายการธุรกรรมต่าง ๆ ผ่านระบบได้
- หน่วยงานธุรกิจของธนาคาร ใช้ระบบในการเข้าดูและแก้ไขข้อมูลให้ทันสมัยเพื่อใช้ข้อมูลเหล่านั้นในการรองรับการทำรายการกับระบบต่าง ๆ
- หน่วยงานที่ควบคุมการทำงานของระบบ (Data Center) ใช้ข้อมูลในระบบเพื่อทำการวิเคราะห์ ทดสอบคุณสมบัติและลักษณะเด่นของระบบ

### 5.3.1 พิจารณาในส่วนของ Functional Requirements

Functional Requirements ของระบบงาน CCMS แบ่งออกเป็น 10 ฟังก์ชัน ดังต่อไปนี้

1) การเข้าสู่ระบบ (Login)
2) สอบถามข้อมูลด้านบัญชี (Account Information Inquiry )
    - สอบถามยอดคงเหลือในบัญชี (Account Balance Inquiry)
    - สอบถามรายการเคลื่อนไหวของบัญชี (Account Statement Inquiry)
3) บริการด้านการโอนเงิน (Transfer Funds)
4) บริการด้านเช็ค (Cheque Service)
    - สอบถามข้อมูลเกี่ยวกับสถานะเช็ค (Cheque Status Inquiry)
    - อายัดเช็ค (Stop Cheque)
        - อายัดเช็ครายฉบับ (Stop Single Cheque)
        - อายัดเช็คเป็นช่วง (Stop Multiple Cheque)
    - ยกเลิกอายัดเช็ค (Cancel Stop Cheque)
    - สั่งซื้อสมุดเช็ค (Order Cheque Book)
    - ขอออกเช็คอัตโนมัติ (Automatic Outsourcing Cheque)

5) บริการสอบถามข้อมูลรายละเอียดการรับชำระ (Detail Payment Information Service)

6) Bulk Payment Service ทำรายการเกี่ยวกับจ่ายเงินค่าสินค้าและบริการ รวมทั้งการจ่ายเงินเดือนพนักงาน บริการเรียกเก็บเงินโดยที่บริษัทสามารถส่งข้อมูลและรับข้อมูลผ่านระบบ

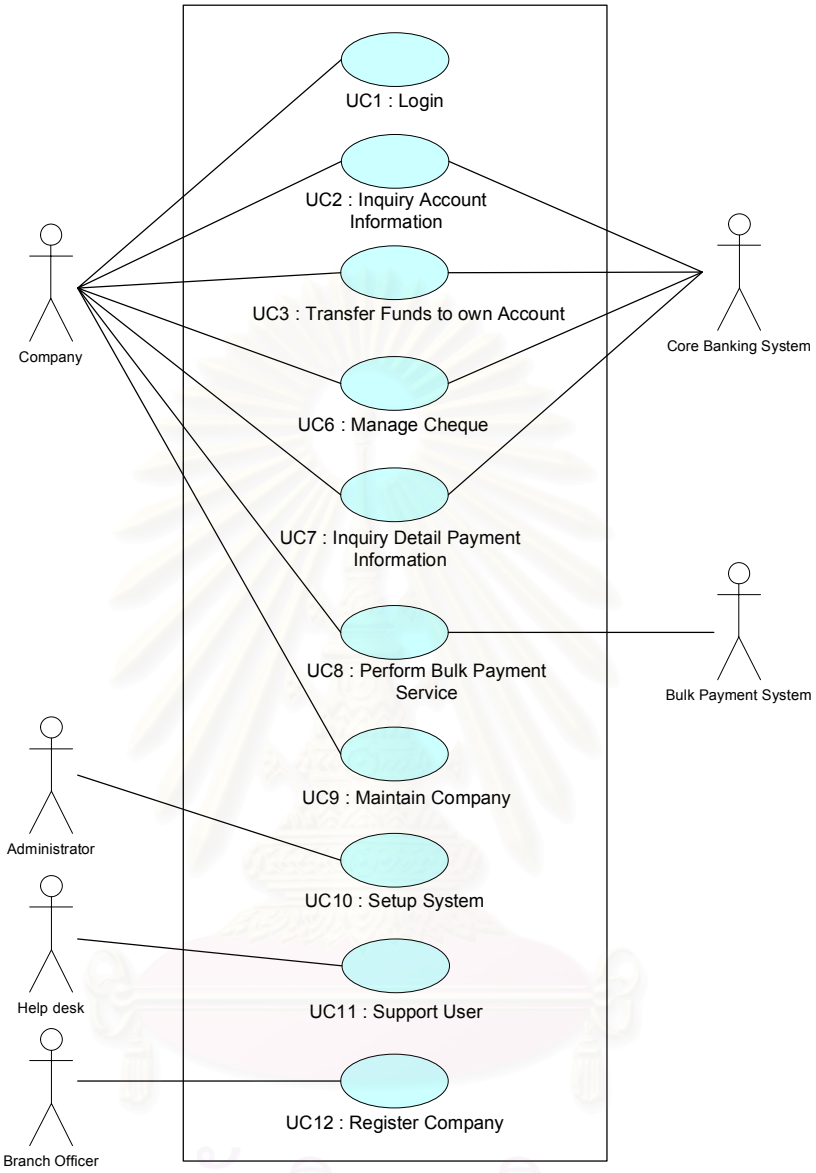
- Bill Payment
- Direct Credit
- Payroll
- Direct Debit
- Submit Payment File
- Inquiry Payment File Status
- Download Payment File

7) Maintain Own Company

8) Prepare Data and Setup

9) User Problem

10) Register Company




รูปที่ 5.1 แผนภาพ Use Case summary ของระบบงาน CCMS

จากรูปที่ 5.1 แสดงให้เห็นแผนภาพ Use Case summary ของระบบงาน CCMS ซึ่งสามารถทำการระบุปัจจัยผันแปรได้ 2 ปัจจัย คือ ประเภทของบริษัท (Company Type) และประเภทของบริการ (Service Type) เนื่องจากการให้บริการลูกค้าของธนาคารรองรับลูกค้าหลายประเภทได้แก่ บริษัท ส่วนข้าราชการ และรัฐวิสาหกิจ การให้บริการก็จะแตกต่างกันในลักษณะของการทำธุรกรรมของแต่ละประเภทลูกค้า เช่น ส่วนข้าราชการสามารถใช้บริการระบบได้เพียงการสอบถามข้อมูลด้านบัญชีและการจ่ายเงินเดือนพนักงานเท่านั้น ในขณะเดียว กันลูกค้าในรูปแบบบริษัทสามารถใช้บริการที่เปิดให้ใช้ได้ทั้งหมด นอกจากนี้ในกลุ่มของลูกค้าประเภทเดียวกัน

ก็มีความต้องการที่แตกต่างกัน ดังนั้นการจัดการดูแลปัจจัยผันแปรจะทำการระบุและอธิบายลงใน Use Case Specification ของ Use Case UC12 : Register Company เพื่อให้ Branch Officer กำหนดสิทธิ์ในการใช้บริการของแต่ละประเภทลูกค้า และ Use Case UC9 : Maintain Company เพื่อให้ลูกค้าแต่ละประเภทสามารถเลือกใช้บริการได้ตามบริการที่เปิดให้ใช้ ดังนี้

**Use Case Name: UC12: Register Company**

**Brief Description**

*Use case* – ลงทะเบียนการใช้บริการในครั้งแรกให้กับบริษัท

**Flow of Events**

***Basic Flow***

1. Branch Officer ป้อน User id ที่มีความรับผิดชอบเป็น Admin ของบริษัทเข้าสู่ระบบ
2. Branch Officer กำหนดสิทธิการใช้บริการให้กับ User id ที่มี Role เป็น Admin เพื่อให้บริษัท ที่ถือ user id นี้สามารถไปกำหนดสิทธิการใช้ ของ user id อื่น ของบริษัทตัวเองได้
3. ระบบบันทึก user id และสิทธิที่กำหนดไว้

**Alternative Flows**

2a. ไม่สามารถใช้บริการนี้ได้ : ระบบแสดงข้อความ " ไม่สามารถใช้บริการนี้ได้ กรุณาติดต่อเจ้าหน้าที่ที่ดูแลระบบ (Administrator) "

**Customer Type**

1. บริษัท สามารถใช้บริการได้ดังนี้
    - Account Information Inquiry
    - Transfer Funds
    - Cheque Service
    - Detail Payment Information Service
    - Bulk Payment Service
2. ส่วนราชการ สามารถใช้บริการได้ดังนี้
    - Account Information Inquiry
    - Bulk Payment Service – Input Payroll

3. รัฐวิสาหกิจ สามารถใช้บริการได้ดังนี้
    - Account Information Inquiry
    - Transfer Funds
    - Detail Payment Information Service
    - Bulk Payment Service

**Preconditions**

เจ้าหน้าที่ต้องทำการ login เข้าระบบ และ มีสิทธิที่จะใช้บริการนี้

**Post conditions**

-

**Extension Points**

-

**Use Case Name: UC9: Maintain Company**

**Brief Description**

*Use case* – บริษัทสามารถดูแลและกำหนดสิทธิการใช้ระบบของบริษัทตนเอง

**Flow of Events**

***Basic Flow***

1. บริษัทสามารถเลือกทำขั้นตอน 2 – 3 โดยไม่ต้องเรียงลำดับ
2. บริษัททำรายการ UC9.1: Create User ID
3. บริษัททำรายการ UC9.2: Setup Authorization

**Alternative Flows**

1a. เลือกบริการไม่ถูกต้อง: กลับไปเลือกบริการใหม่

**Customer Type**

1. บริษัท สามารถใช้บริการได้ดังนี้
    - Account Information Inquiry
    - Transfer Funds
    - Cheque Service

- Detail Payment Information Service
- Bulk Payment Service

2. ส่วนราชการ สามารถใช้บริการได้ดังนี้
   - Account Information Inquiry
   - Bulk Payment Service – Input Payroll
3. รัฐวิสาหกิจ สามารถใช้บริการได้ดังนี้
   - Account Information Inquiry
   - Transfer Funds
   - Detail Payment Information Service
   - Bulk Payment Service

**Preconditions**

บริษัทต้องทำการ login เข้าระบบ และ มีสิทธิที่จะใช้บริการนี้

**Post conditions**

บริษัทสามารถเลือกทำรายการการสร้าง User ID และการกำหนดสิทธิการใช้ในแต่ละ User ID ได้

**Extension Points**

None

3.4.2 พิจารณาในส่วนของ Non-Functional Requirements

Non-functional Requirements ของระบบงาน CCMS มีดังนี้

1) Usability
   - Standard Graphic User Interface
   - Web Based Application
   - Design of consistent layout and template
   - Online help
   - User friendly
   - Function grouping
   - System demonstration

- สามารถรองรับภาษาไทย

2) Performance

- การตอบรับของระบบอยู่ที่ระดับ 80% ของการทำรายการซึ่งใช้เวลาน้อยกว่า 5 วินาที บน LAN terminal.
- ความสามารถของระบบที่จะรองรับการทำงานของหลายๆ User ได้ในเวลาพร้อมๆกัน
- ความสามารถของระบบที่จะรองรับได้ในปัจจุบันและอนาคต
- ระยะเวลาของการประมวลผลที่รวดเร็ว

3) Operational Environment

- Physical Environment
  - มีระบบการจัดเก็บข้อมูลที่สม่ำเสมอ (Back up data)
  - ระบบรักษาความปลอดภัย
  - มีระบบสำรอง DRC (Disaster Recovery Center)
- Technological Environment
  - Run on Unix Platform
  - Database ที่ใช้ คือ Oracle
  - Tools ที่ใช้ในการพัฒนาระบบ คือ BEA Weblogic
  - มีการเก็บบันทึกการทำงาน Logging
  - Performance tuning
  - Load balancing technique

4) Security ระบบความปลอดภัยมีสิ่งที่จะต้องคำนึงถึงในด้านต่าง ๆ ดังต่อไปนี้

- มี username ,password และ security type (smart card, token ,…) ในการ log เข้าระบบ
- Application Control มีการกำหนดสิทธิการใช้ของแต่ละ user (user profile)
- Operation control
  - Keep all servers at secure data center and monitor
  - Create procedure for change control

5) User Document

คู่มือในการอธิบายหรือวิธีการใช้งานระบบ จากตารางที่ 5.1 อธิบายถึงรายชื่อของเอกสาร กลุ่มผู้อ่านจุดประสงค์และเนื้อหาโดยย่อของเอกสารที่ควรจะต้องมี

ตารางที่ 5.1 ตารางคู่มือวิธีการใช้งานระบบ

| Document Name | Intended User | Purpose | Content Brief |
|---|---|---|---|
| 1.Corporate Cash Management System (CCMS) User Guide | General users | Reference guide of how to use system | Synopsis of all function and operation |
| 2. Operational guide | IT operators | Operating the system | Monitoring, control, backup and trouble shooting |
| 3. Tutorial | General users | Self learning | General use of the system |

6) Maintainability

กรณีที่มีการเพิ่มความต้องการใหม่เข้าไปในกระบวนการเพื่อเพิ่มประสิทธิภาพของระบบให้กับผู้ใช้งานนั้น จะต้องมีการทำกระบวนการ change control

การพิจารณา Non-Functional Requirements เพื่อนำมาสร้าง Infrastructure Use Case สามารถทำได้โดยการพิจารณาจาก 2 ปัจจัย คือ ผลกระทบที่มีต่อ Use Case โดยเลือกเอาเฉพาะที่ส่งผลกระทบบางส่วน ทำให้ต้องให้ความสนใจเป็นพิเศษ และอีกปัจจัยคือ ผู้ใช้งานระบบสามารถทำการทดสอบได้ด้วยตนเอง ดังตารางที่ 5.2 ดังนี้

ตารางที่ 5.2 ตารางพิจารณา Non-Functional Requirements

| Non-Functional Requirements | Affect Use Case | Testing By User | Candidate |
|---|---|---|---|
| 1. Usability | | | |
| • Standard Graphic User Interface | Every step | No | No |
| • Web Based Application | Every step | No | No |
| • Design of consistent layout and template | Every step | No | No |
| • Online help | Every step | Yes | No |
| • User friendly | Every step | Yes | No |
| • Function grouping | Every step | Yes | No |

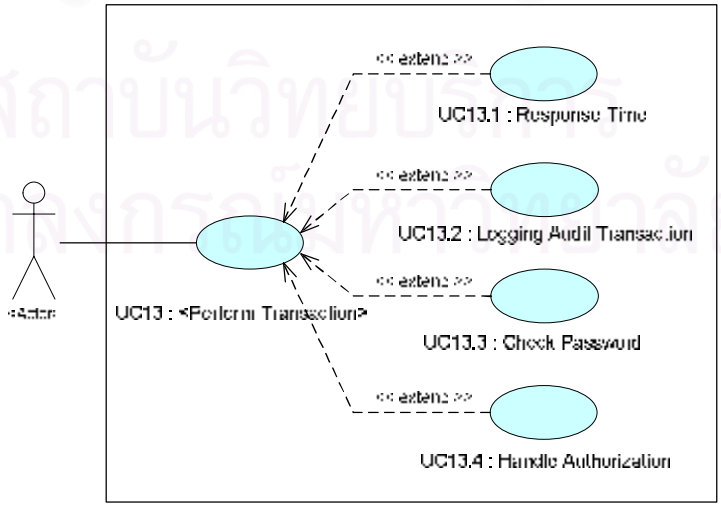
| Non-Functional Requirements | Affect Use Case | Testing By User | Candidate |
|---|---|---|---|
| • System demonstration | Every step | No | No |
| • สามารถรองรับภาษาไทย | Every step | Yes | No |
| 2. Performance | | | |
| • การตอบรับของระบบอยู่ที่ระดับ 80% ของการทำรายการซึ่งใช้เวลาน้อยกว่า 5 วินาที บน LAN terminal. | Every step | Yes | No |
| • ความสามารถของระบบที่จะรองรับการทำงานของหลายๆ User ได้ในเวลาพร้อม ๆ กัน | Every step | Yes | No |
| • ความสามารถของระบบที่จะรองรับได้ในปัจจุบันและอนาคต | Every step | Yes | No |
| • ระยะเวลาของการประมวลผลที่รวดเร็ว | Each step | Yes | Yes |
| 3. Operational Environment | | | |
| • Physical Environment | | | |
| - มีระบบการจัดเก็บข้อมูลที่สม่ำเสมอ (Back up data) | - | No | No |
| - ระบบรักษาความปลอดภัย | - | No | No |
| - มีระบบสำรอง DRC (Disaster Recovery Center) | - | No | No |
| • Technological Environment | | | |
| - Run on Unix Platform | - | No | No |
| - Database ที่ใช้ คือ Oracle | - | No | No |
| - Tools ที่ใช้ในการพัฒนาระบบ คือ BEA Weblogic | - | No | No |
| - มีการเก็บบันทึกการทำงาน Logging | Each step | Yes | Yes |
| - Performance tuning | Every step | No | No |
| - Load Balancing technique | Every step | No | No |
| 4. Security | | | |
| • มี username, password และ security | Each step | Yes | Yes |

| Non-Functional Requirements | Affect Use Case | Testing By User | Candidate |
|---|---|---|---|
| type (smart card , token ) ในการ log เข้าระบบ<br>• Application Control มีการกำหนดสิทธิการใช้ ของแต่ละuser (user profile)<br>• Operation control | Each step<br><br>- | Yes<br><br>No | Yes<br><br>No |
| 5. User Document | - | - | No |
| 6. Maintainability | - | - | No |

จะเห็นได้ว่า Non-Functional Requirements ที่ถูกคัดเลือกมาสร้าง Infrastructure Use Case มี 4 ประเด็น คือ

1. ระยะเวลาของการประมวลผลที่รวดเร็ว
2. การเก็บบันทึกการทำงาน Logging
3. มี username, password และ security type (smart card, token) ในการlog เข้าระบบ
4. การกำหนดสิทธิการใช้ ของแต่ละuser (user profile)

ดังนั้นจึงนำมาสร้างเป็น Infrastructure Use Case ดังรูปที่ 5.2 และได้แผนภาพ Use Case summary ในเชิง AOP ดังรูปที่ 5.3




รูปที่ 5.2 แผนภาพ Infrastructure Use Case ของระบบงาน CCMS

**Use Case Name: UC13: <Perform Transaction>**

**Brief Description**

*Use case* – Infrastructure

**Flow of Events**

***Basic Flow***

Infrastructure Use Case จะเริ่มต้นทำงานเมื่อ actor เข้ามาทำรายการในระบบ

1. ระบบเตรียมพร้อมรับข้อมูลที่ actor ต้องการนำเข้าสู่ระบบ
2. actor ทำรายการที่ต้องเข้าสู่ระบบ พร้อมทั้งทำการ Submit รายการที่ทำ
3. ระบบประมวลผลรายการ พร้อมกับแสดงผลที่หน้าจอ
4. จบการทำงาน

**Alternative Flows**

**A1. Access Control**

ในขั้นตอนที่ 3 ใน Basic Flow เป็นขั้นตอนที่ต้องทำการตรวจสอบว่า actor เป็นผู้ใช้งานระบบและมีสิทธิ์ในการใช้งานระบบอะไรได้บ้าง

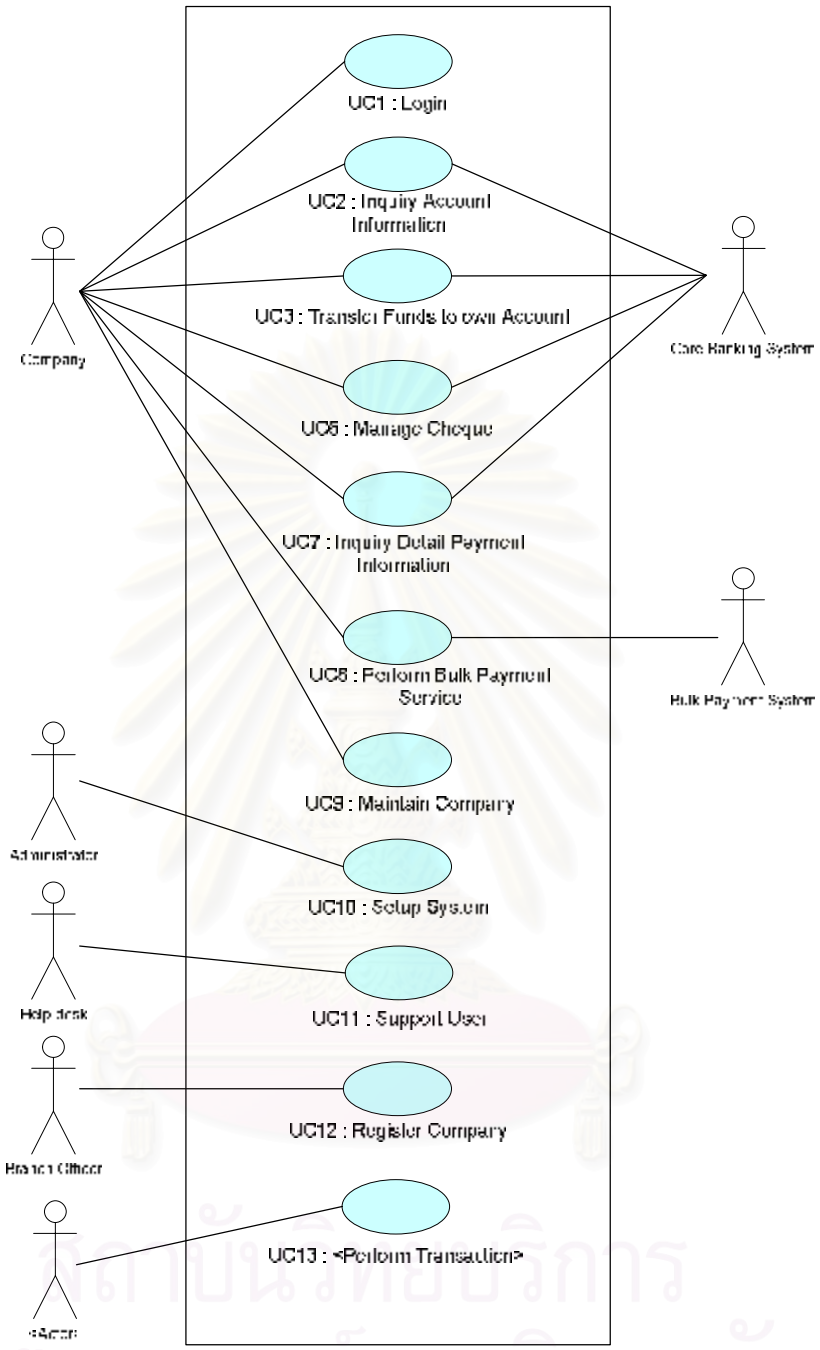
1. ถ้าไม่ใช่เป็นผู้ใช้งานของระบบก็ไม่มีสิทธิ์ในการใช้งานระบบ รายการจะถูกปฏิเสธ
2. ถ้าเป็นผู้ใช้งานระบบจริงก็มีสิทธิ์ใช้งานระบบตามที่ได้กำหนดสิทธิ์ไว้

**A2. Logging**

ในขั้นตอนที่ 2 ใน Basic Flow เป็นขั้นตอนที่มีการ Submit และต้องทำการ Logging ลงฐานข้อมูลไว้เพื่อการตรวจสอบ

**Special Requirements**

ในการประมวลผลของระบบต้องทำได้อย่างรวดเร็ว




รูปที่ 5.3 แผนภาพ Use Case summary ในเชิง AOP ของระบบงาน CCMS

## 5.4 ตัวอย่างการนับฟังก์ชันพอยต์จากแผนภาพ Use Case

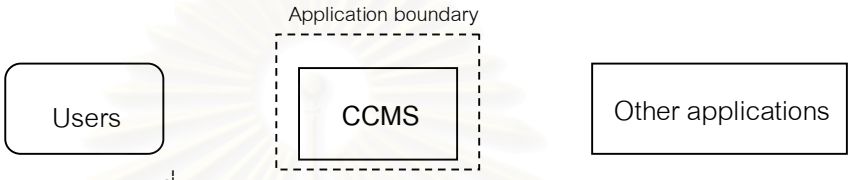
หลังจากที่ได้ทำการสร้างแผนภาพ Use Case ในเชิง AOP ของระบบงาน CCMS ขั้นตอนต่อไปคือการนับฟังก์ชันพอยต์ จากแผนภาพ Use Case ดังรูปที่ 5.3 แบ่งออก เป็น 11 ฟังก์ชัน ดังต่อไปนี้

1. การเข้าสู่ระบบ (Login)
2. สอบถามข้อมูลด้านบัญชี (Account Information Inquiry )
    - สอบถามยอดคงเหลือในบัญชี (Account Balance Inquiry)
    - สอบถามรายการเคลื่อนไหวของบัญชี (Account Statement Inquiry)
3. บริการด้านการโอนเงิน (Transfer Funds)
4. บริการด้านเช็ค (Cheque Service)
    - สอบถามข้อมูลเกี่ยวกับสถานะเช็ค (Cheque Status Inquiry)
    - อายัดเช็ค (Stop Cheque)
        - อายัดเช็ครายฉบับ (Stop Single Cheque)
        - อายัดเช็คเป็นช่วง (Stop Multiple Cheque)
    - ยกเลิกอายัดเช็ค (Cancel Stop Cheque)
    - สั่งซื้อสมุดเช็ค (Order Cheque Book)
    - ขอออกเช็คอัตโนมัติ (Automatic Outsourcing Cheque)
5. บริการสอบถามข้อมูลรายละเอียดการรับชำระ (Detail Payment Information Service)
6. Bulk Payment Service ทำรายการเกี่ยวกับจ่ายเงินค่าสินค้าและบริการ รวมทั้งการจ่ายเงินเดือนพนักงาน บริการเรียกเก็บเงินโดยที่บริษัทสามารถส่งข้อมูลและรับข้อมูลผ่านระบบ
    - Bill Payment
    - Direct Credit
    - Payroll
    - Direct Debit
    - Submit Payment File
    - Inquiry Payment File Status
    - Download Payment File
7. Maintain Own Company
8. Prepare Data and Setup

9. User Problem
10. Register Company
11. Perform Transaction

ขั้นตอนในการนับฟังก์ชันพอยต์

5.4.1 การระบุขอบเขตการนับ (Identification of the counting boundary)




รูปที่ 5.4 การระบุขอบเขตการนับของระบบงาน CCMS

จากรูปที่ 5.4 เป็นการระบุขอบเขตการนับของระบบงาน CCMS ทำให้สามารถพิจารณา actor ของระบบงาน CCMS ได้ดังนี้

Users ของระบบงาน CCMS คือ

1. Company
2. Administrator
3. Help Desk
4. Branch Officer
5. <Actor>

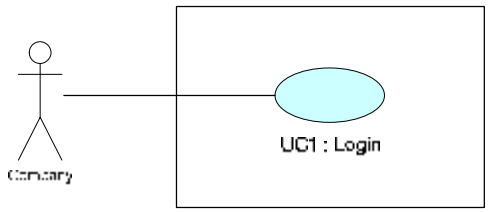
Other Applications ของระบบงาน CCMS คือ

1. Core Banking System
2. Bulk Payment System

5.4.2 การระบุรายการในขอบเขต (Identification of items within boundary)

จากแผนภาพ Use Case ของระบบงาน CCMS use case ที่ถูกเลือกตามหลักการพิจารณาในส่วนของ Transaction functions และ Scenario มีดังนี้

1) UC1 : Login ตรวจสอบสิทธิในการใช้ระบบ ดังแสดงในรูปที่ 5.5 และมี Scenario ของแผนภาพ Use Case ดังตารางที่ 5.3




รูปที่ 5.5 แผนภาพ Use Case Login

ตารางที่ 5.3 Scenario ของแผนภาพ Use Case Login

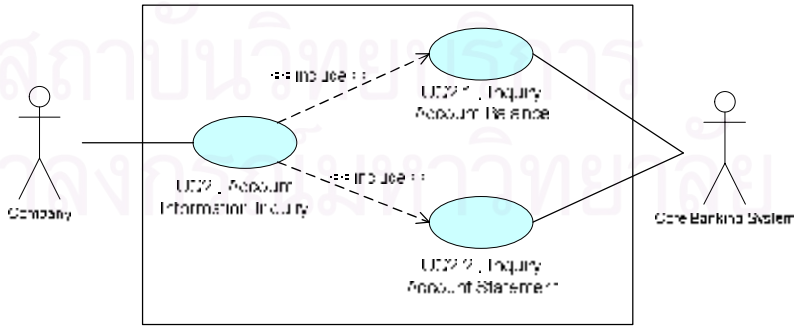
| Use Case # 1 : Login | |
|---|---|
| Main Success Scenario | |
| Step | Action |
| 1 | บริษัทป้อนข้อมูล รหัสประจำตัวและ รหัสผ่าน |
| 2 | ระบบทำการตรวจสอบความถูกต้องของรหัสประจำตัวและรหัสผ่าน. |
| 3 | ระบบตอบรับและแสดงหน้าจอการใช้บริการ |

2) UC2 : Account Information Inquiry - สอบถามข้อมูลด้านบัญชีของบริษัท

2.1) UC2.1 : Inquiry Account Balance - เรียกดูยอดคงเหลือของบัญชีบริษัท

2.2) UC2.2 : Inquiry Account Statement - เรียกดูรายการเคลื่อนไหวของบัญชีบริษัท

ดังแสดงในรูปที่ 5.6 และมี Scenario ของแผนภาพ Use Case ดังตารางที่ 5.4



รูปที่ 5.6 แผนภาพ Use Case Account Information Inquiry

ตารางที่ 5.4 Scenario ของแผนภาพ Use Case Account Information Inquiry

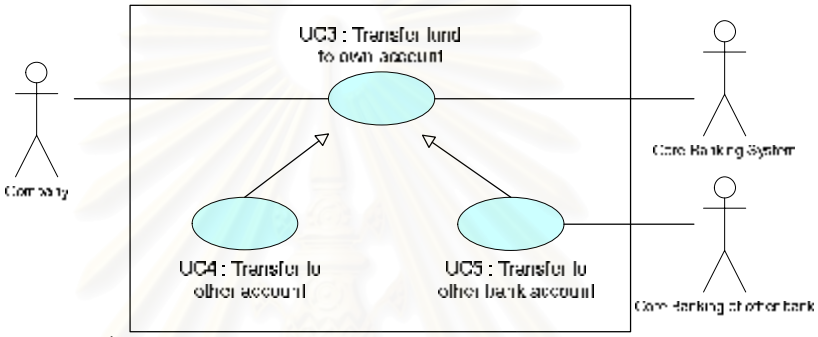
| Use Case # 2 : Inquiry Account Information | |
|---|---|
| Main Success Scenario | |
| Step | Action |
| 1 | บริษัทสามารถเลือกทำขั้นตอน 2 – 3 โดยไม่ต้องเรียงลำดับ |
| 2 | บริษัททำรายการ UC2.1: Inquiry Account Balance |
| 3 | บริษัททำรายการ UC2.2: Inquiry Account Statement |
| **Use Case # 2.1 : Inquiry Account Balance** | |
| Main Success Scenario | |
| Step | Action |
| 1 | ระบบแสดงรายการเลขที่บัญชีของบริษัทที่มีอยู่ในระบบ |
| 2 | บริษัทเลือกเลขที่บัญชีที่ต้องการสอบถาม |
| 3 | ระบบทำการบันทึกข้อมูลการทำรายการลง Audit log file และส่งเลขที่บัญชีไปยังระบบ Core Banking |
| 4 | ระบบตอบรับการทำรายการของบริษัท พร้อมกับแสดงข้อมูลยอดคงเหลือของเลขที่บัญชีที่สอบถามทางจอภาพ |
| **Use Case # 2.2 : Inquiry Account Statement** | |
| Main Success Scenario | |
| Step | Action |
| 1 | ระบบแสดงรายการเลขที่บัญชีของบริษัทที่มีอยู่ในระบบ |
| 2 | บริษัทเลือกหมายเลขบัญชีที่ต้องการสอบถาม และระบุช่วงวันที่และเวลาที่ต้องการทราบรายการเคลื่อนไหวของบัญชี |
| 3 | ระบบทำการบันทึกข้อมูลการทำรายการลง Audit log file และส่งเลขที่บัญชีกับช่วงวัน,เวลาที่ต้องการทราบไปยังระบบ Core Banking |
| 4 | ระบบตอบรับการทำรายการของบริษัท พร้อมกับแสดงข้อมูลยอดคงเหลือของเลขที่บัญชีที่สอบถามทางจอภาพ |
| 5 | บริษัทสามารถเลือกทำ ข้อ 6 – 7 ต่อตามต้องการ โดยไม่ต้องเรียงลำดับ |
| 6 | บริษัทสามารถเลือก download statement file ของหมายเลขบัญชีที่สอบถาม |
| 7 | บริษัทสามารถเลือก print ข้อมูลรายการเคลื่อนไหวของหมายเลขบัญชีที่สอบถาม |

3) UC3 : Transfer Funds - ทำรายการเกี่ยวกับการโอนเงินระหว่างบัญชีของบริษัทเอง

3.1) UC4 : Transfer To Other Account - ทำรายการเกี่ยวกับการโอนเงินระหว่างบัญชีบริษัทเองกับบัญชีบุคคลอื่นในธนาคารกรุงไทย

3.2) UC5 : Transfer To Other Bank Account - ทำรายการเกี่ยวกับการโอนเงินระหว่างบัญชีบริษัทกับ บัญชีของธนาคารอื่น

ดังแสดงในรูปที่ 5.7 และมี Scenario ของแผนภาพ Use Case ดังตารางที่ 5.5




รูปที่ 5.7 แผนภาพ Use Case Transfer fund to own account

ตารางที่ 5.5 Scenario ของแผนภาพ Use Case Transfer fund to own account

| Use Case # 3 : Transfer Fund to Own Account | |
|---|---|
| Main Success Scenario | |
| Step | Action |
| 1 | ระบบแสดงรายการเลขที่บัญชีโอนออกของบริษัทที่มีอยู่ในระบบ |
| 2 | บริษัทเลือกเลขที่บัญชีที่ต้องการโอนเงินออก (From Account) |
| 3 | ระบบแสดงรายการเลขที่บัญชีโอนเข้าของบริษัทที่มีอยู่ในระบบ |
| 4 | บริษัทเลือกเลขที่บัญชีของตนเองที่ต้องการโอนเข้า ( To Account ) |
| 5 | บริษัทใส่ รหัสผ่าน |
| 6 | บริษัทใส่จำนวนเงินที่ต้องการโอนเข้า |
| 7 | ระบบจะแสดงข้อมูลการทำรายการ เพื่อการยืนยัน |
| 8 | ระบบทำการบันทึกข้อมูลการทำรายการลง Audit log file พร้อมส่งรายการขออนุมัติจากผู้มีอำนาจ (Authorized User)และส่งรายการที่ทำไปยังระบบ Core Banking |
| 9 | ระบบตอบรับการทำรายการของบริษัท |

| Use Case # 4 : Transfer Fund to Other Account | |
|---|---|
| Main Success Scenario | |
| Step | Action |
| 1 | ระบบแสดงรายการเลขที่บัญชีโอนออกของบริษัทที่มีอยู่ในระบบ |
| 2 | บริษัทเลือกเลขที่บัญชีที่ต้องการโอนเงินออก (From Account) |
| 3 | ระบบแสดงรายการเลขที่บัญชีโอนเข้าของบริษัทที่มีอยู่ในระบบ |
| 4 | บริษัทเลือกเลขที่บัญชีของบุคคลอื่นที่ต้องการโอนเข้า ( To Account ) |
| 5 | บริษัทใส่ รหัสผ่าน |
| 6 | บริษัทใส่จำนวนเงินที่ต้องการโอนเข้า |
| 7 | ระบบจะแสดงข้อมูลการทำรายการ เพื่อการยืนยัน |
| 8 | ระบบทำการบันทึกข้อมูลการทำรายการลง Audit log file พร้อมส่งรายการขออนุมัติจากผู้มีอำนาจ (Authorized User)และส่งรายการที่ทำไปยังระบบ Core Banking |
| 9 | ระบบตอบรับการทำรายการของบริษัท |
| Use Case # 5 : Transfer Fund to Other Bank Account | |
| Main Success Scenario | |
| Step | Action |
| 1 | ระบบแสดงรายการเลขที่บัญชีโอนออกของบริษัทที่มีอยู่ในระบบ |
| 2 | บริษัทเลือกเลขที่บัญชีที่ต้องการโอนเงินออก (From Account) |
| 3 | ระบบแสดงรายการเลขที่บัญชีโอนเข้าของบริษัทที่มีอยู่ในระบบ |
| 4 | บริษัทเลือกเลขที่บัญชีของธนาคารอื่นที่ต้องการโอนเข้า ( To Account ) |
| 5 | บริษัทใส่ รหัสผ่าน |
| 6 | บริษัทใส่จำนวนเงินที่ต้องการโอนเข้า |
| 7 | ระบบจะแสดงข้อมูลการทำรายการ เพื่อการยืนยัน |
| 8 | ระบบทำการบันทึกข้อมูลการทำรายการลง Audit log fileและส่งรายการที่ทำไปยัง Others Gateway ของธนาคารอื่น |
| 9 | ระบบตอบรับการทำรายการของบริษัท |

4) UC6 : Cheque Service - ทำรายการเกี่ยวกับ Cheque ของบริษัท

4.1) UC6.1 : Inquiry Cheque Status - สอบถามสถานะเช็คของบัญชีบริษัท

4.2) UC6.2 : Stop Single Cheque - อายัดเช็ครายฉบับของบัญชีบริษัท

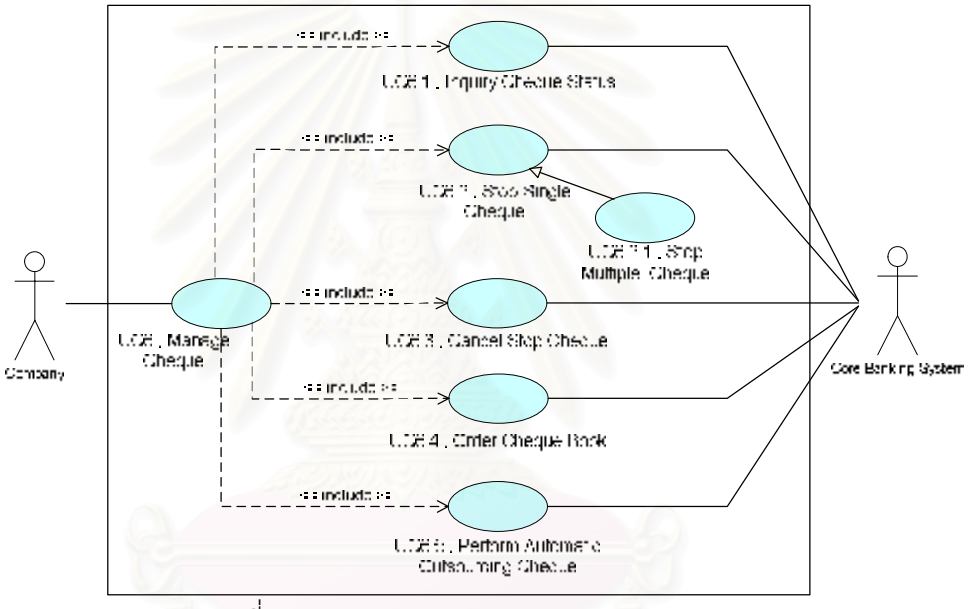
4.2.1) UC6.2.1 : Stop Multiple - อายัดเช็คเป็นช่วงของบัญชีบริษัท

4.3) UC6.3 : Cancel Stop Cheque - ยกเลิกอายัดเช็คของบัญชีบริษัท

4.4) UC6.4 : Order Cheque Book - สั่งซื้อสมุดเช็ค

4.5) UC6.5 : Perform Automatic Outsourcing Cheque - บริษัทสามารถส่งOutsourcing Cheque file มาให้ทางธนาคารออกเช็คแทนบริษัทโดยสั่งจ่ายและดำเนินการตามข้อมูลที่จัดส่งมา

ดังแสดงในรูปที่ 5.8 และมี Scenario ของแผนภาพ Use Case ดังตารางที่ 5.6




รูปที่ 5.8 แผนภาพ Use Case Manage Cheque

ตารางที่ 5.6 Scenario ของแผนภาพ Use Case Manage Cheque

| Use Case # 6 : Manage Cheque | |
|---|---|
| Main Success Scenario | |
| Step | Action |
| 1 | บริษัทสามารถเลือกทำขั้นตอน 2 – 6 โดยไม่ต้องเรียงลำดับ |
| 2 | บริษัททำรายการ UC6.1 : Inquiry Cheque Status |
| 3 | บริษัททำรายการ UC6.2 : Stop Single Cheque |
| 4 | บริษัททำรายการ UC6.3 : Cancel Stop Cheque |
| 5 | บริษัททำรายการ UC6.4 : Order Cheque Book |

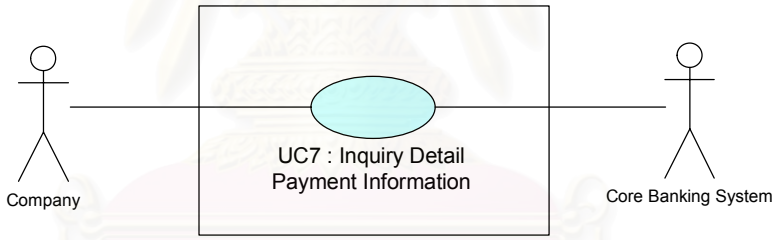
| Step | Action |
|---|---|
| 6 | บริษัททำรายการ UC6.5 : Perform Automatic Outsourcing Cheque |
| **Use Case # 6.1 : Inquiry Cheque Status** | |
| Main Success Scenario | |
| Step | Action |
| 1 | บริษัทป้อนหมายเลขบัญชี |
| 2 | บริษัทใส่ หมายเลขเช็คที่ต้องการสอบถาม โดยระบุ Start กับ End Cheque No |
| 3 | ระบบทำการบันทึกข้อมูลการทำรายการลง Audit log file และส่งหมายเลขที่บัญชี, หมายเลขเช็คที่ระบุ ยังระบบ Core Banking |
| 4 | ระบบแสดงหมายเลขเช็ค (Cheque No.), สถานะของเช็ค (Status of cheque) และ วันที่allocate เช็ค ( Issue Date) ของหมายเลขบัญชีที่สอบถามทางจอภาพ โดยเรียงข้อมูลตามหมายเลขเช็คจากน้อยไปมาก |
| **Use Case # 6.2 : Stop Single Cheque** | |
| Main Success Scenario | |
| Step | Action |
| 1 | บริษัทป้อน หมายเลขบัญชี |
| 2 | บริษัทป้อน หมายเลขเช็คที่ต้องการอายัด |
| 3 | บริษัทป้อนรายละเอียดอื่นๆ ที่ระบบต้องการ ดังนี้<br>o อายัดโดย<br>o รหัสเหตุผลการอายัด<br>o ชื่อผู้รับเงิน<br>o เหตุผลการอายัด |
| 4 | ระบบทำการบันทึกข้อมูลการทำรายการลง Audit log file และส่งหมายเลขบัญชีกับหมายเลขเช็คที่อายัดไปยังระบบ Core Banking |
| 5 | ระบบตอบรับการทำรายการของบริษัท |
| **Use Case # 6.2.1 : Stop Multiple Cheque** | |
| Main Success Scenario | |
| Step | Action |
| 1 | บริษัทป้อน หมายเลขบัญชี |
| 2 | บริษัทป้อนหมายเลขเช็คต่ำสุดในช่วงของเช็คที่จะทำการอายัด และหมายเลขเช็คสูงสุดในช่วงของเช็คที่จะทำการอายัด |

| Step | Action |
|---|---|
| 3 | บริษัทป้อนรายละเอียดอื่นๆ ที่ระบบต้องการ ดังนี้<br>o อายัดโดย<br>o รหัสเหตุผลการอายัด<br>o เหตุผลการอายัด |
| 4 | ระบบทำการบันทึกข้อมูลการทำรายการลง Audit log file และส่งหมายเลขบัญชีกับหมายเลขเช็คที่อายัดไปยังระบบ Core Banking |
| 5 | ระบบตอบรับการทำรายการของบริษัท |
| **Use Case # 6.3 : Cancel Stop Cheque** | |
| **Main Success Scenario** | |
| Step | Action |
| 1 | บริษัทใส่หมายเลขบัญชี |
| 2 | บริษัทใส่หมายเลขเช็คต่ำสุดในช่วงของเช็คที่จะยกเลิกอายัด |
| 3 | หมายเลขเช็คสูงสุดในช่วงของเช็คที่จะทำการยกเลิกรอายัด และบริษัทใส่ชื่อผู้อายัด |
| 4 | บริษัทใส่วันที่อายัดเช็ค (Stop Cheque Date) |
| 5 | ระบบทำการบันทึกข้อมูลการทำรายการลง Audit log file และส่งหมายเลขบัญชีกับช่วงหมายเลขเช็คที่ต้องการยกเลิกการอายัดไปยังระบบ Core Banking |
| 6 | ระบบตอบรับการทำรายการของบริษัท |
| **Use Case # 6.4 : Order Cheque Book** | |
| **Main Success Scenario** | |
| Step | Action |
| 1 | ระบบแสดงรายการเลขที่บัญชีกระแสรายวันของบริษัทที่มีอยู่ในระบบ |
| 2 | บริษัทเลือกหมายเลขบัญชี |
| 3 | บริษัทป้อนจำนวนสมุดเช็คที่ต้องการสั่งซื้อ |
| 4 | บริษัทเลือกช่องทางการจัดส่งสมุดเช็ด โดยเลือกได้ เพียง 1 ช่องทาง<br>o สาขา : เลือกสาขาที่ต้องการไปรับสมุดเช็ค<br>o ไปรษณีย์ : ป้อนที่อยู่ที่ต้องการให้จัดส่ง |
| 5 | ระบบทำการบันทึกข้อมูลการทำรายการลง Audit log file และส่งหมายเลขบัญชีกับจำนวนสมุดเช็คที่ต้องการสั่งซื้อไปยังระบบ Core Banking |
| 6 | ระบบตอบรับการทำรายการของบริษัท |

| Use Case # 6.5 : Perform Automatic Outsourcing Cheque | |
|---|---|
| Main Success Scenario | |
| Step | Action |
| 1 | ระบบ default ค่าของ รหัสบริษัท |
| 2 | บริษัทป้อนวันที่ต้องการให้ออกเช็ค |
| 3 | บริษัทเลือก Outsourcing cheque file ที่ต้องการส่ง |
| 4 | ระบบทำการบันทึกข้อมูลการทำรายการลง Audit log file และส่งข้อมูลรหัสบริษัท, วันที่ต้องการออกเช็คและ Outsourcing cheque file ไปยังระบบ Core Banking |
| 5 | ระบบตอบรับการทำรายการของบริษัท |

5) UC7 : Detail Payment Information Service - เรียกดูรายการที่ลูกค้าของบริษัทมาชำระเงินให้กับบริษัทตามใบเรียกเก็บเงิน

ดังแสดงในรูปที่ 5.9 และมี Scenario ของแผนภาพ Use Case ดังตารางที่ 5.7




รูปที่ 5.9 แผนภาพ Use Case Inquiry Detail Payment Information

ตารางที่ 5.7 Scenario ของแผนภาพ Use Case Inquiry Detail Payment Information

| Use Case # 7 : Inquiry Detail Payment Information | |
|---|---|
| Main Success Scenario | |
| Step | Action |
| 1 | ระบบแสดงรายการ Company Codeของบริษัทที่มีอยู่ในระบบ |
| 2 | บริษัทเลือก Company Code และระบุช่วงวันที่และช่วงเวลาที่ต้องการ |
| 3 | ระบบทำการบันทึกข้อมูลการทำรายการลง Audit log file และส่ง Company Code กับช่วงวันที่ต้องการทราบยังระบบ Core Banking |
| 4 | ระบบตอบรับการทำรายการของบริษัท พร้อมกับแสดงข้อมูลรายการตามวันที่ระบุของ Company Code ที่สอบถามทางหน้าจอ |

| Step | Action |
|---|---|
| 5 | บริษัทสามารถเลือกทำ ข้อ 6 – 7 ต่อตามต้องการ โดยไม่ต้องเรียงลำดับ |
| 6 | บริษัทสามารถเลือก download payment file ของ Company Code ที่สอบถาม |
| 7 | บริษัทสามารถเลือก print ข้อมูลรายการของ Company Code ที่สอบถาม |

6) UC8 : Bulk Payment Service - ทำรายการเกี่ยวกับจ่ายเงินค่าสินค้าและบริการ รวมทั้งการจ่ายเงินเดือนพนักงาน บริการเรียกเก็บเงินโดยที่บริษัทสามารถส่งข้อมูลและรับข้อมูลผ่านระบบ

6.1) UC8.1 : Input Bill Payment - ทำการบันทึกข้อมูล รายละเอียดการชำระเงิน โดยหักบัญชีของ บริษัท เพื่อชำระเงินค่าใช้จ่ายต่างๆ

6.2) UC8.2 : Input Direct Credit - ทำการบันทึกข้อมูล การหักบัญชีของบริษัท เพื่อนำเข้าบัญชีลูกค้าของบริษัท

6.3) UC8.3 : Input Payroll - ทำการบันทึกข้อมูล รายละเอียด บริการหักบัญชีของบริษัทเพื่อนำเงิน เข้าบัญชี ของบุคคล ตามคำสั่งชำระเงินของบริษัท

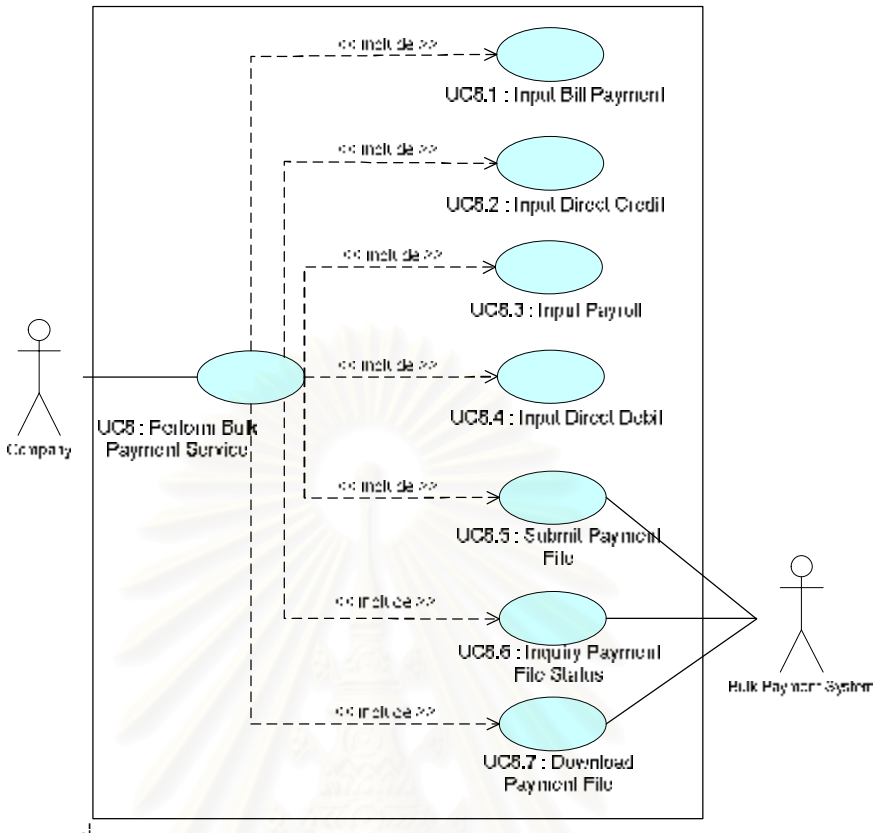
6.4) UC8.4 : Input Direct Debit - ทำการบันทึกข้อมูล รายละเอียดบริการหักบัญชีของลูกค้าของบริษัท รวบรวมเงินเพื่อนำเข้าบัญชีของบริษัท

6.5) UC8.5 : Submit Payment File - ทำรายการส่งข้อมูลให้กับระบบ Bulk Payment System

6.6) UC8.6 : Inquiry Payment File Status - สถานะของแฟ้มข้อมูลที่ส่งให้กับระบบ Bulk Payment System

6.7) UC8.7 : Download Payment File - รับ Output payment fileและ Report payment file ต่างๆ ที่ส่งมาจากระบบ Bulk Payment System

ดังแสดงในรูปที่ 5.10 และมี Scenario ของแผนภาพ Use Case ดังตารางที่ 5.8




รูปที่ 5.10 แผนภาพ Use Case Perform Bulk Payment Service

ตารางที่ 5.8 Scenario ของแผนภาพ Use Case Perform Bulk Payment Service

| Use Case # 8 : Perform Bulk Payment Service | |
|---|---|
| Main Success Scenario | |
| Step | Action |
| 1 | บริษัทสามารถเลือกทำขั้นตอน 2 – 8 โดยไม่ต้องเรียงลำดับ |
| 2 | บริษัททำรายการ UC8.1 : Input Bill Payment |
| 3 | บริษัททำรายการ UC8.2 : Input Direct Credit |
| 4 | บริษัททำรายการ UC8.3 : Input Payroll |
| 5 | บริษัททำรายการ UC8.4 : Input Direct Debit |
| 6 | บริษัททำรายการ UC8.5 : Submit Payment File |
| 7 | บริษัททำรายการ UC8.6 : Inquiry Payment File Status |
| 8 | บริษัททำรายการ UC8.7 : Download Payment File |
| Use Case # 8.1 : Input Bill Payment | |
| Main Success Scenario | |

| Step | Action |
|---|---|
| 1 | บริษัททำขั้นตอน 1- 6 ตามลำดับซ้ำกัน ตามความต้องการ |
| 2 | ระบบแสดงรายการรหัสบริษัทที่มีอยู่ในระบบ |
| 3 | บริษัทเลือกรหัสบริษัทที่ต้องการทำการชำระเงิน |
| 4 | ระบบแสดงรายการหมายเลขบัญชีของบริษัทที่มีอยู่ในระบบ |
| 5 | บริษัทเลือกหมายเลขบัญชีที่ต้องการหักเงิน |
| 6 | บริษัทป้อนข้อมูล<br>o รหัสบริการ (Service code)<br>o Customer No. (Ref #1)<br>o Reference No. (Ref #2)<br>o จำนวนเงินที่ต้องการ<br>o วันที่ที่ต้องการให้หักเงิน |
| 7 | ระบบคำนวณ จำนวนรายการ ยอดรวมรายการชำระเงิน แสดงผลทางหน้าจอ |
| 8 | บริษัทยืนยันการทำรายการ |
| 9 | ระบบทำการบันทึกข้อมูลในรูปแบบของ ไฟล์ข้อมูล เพื่อรอประมวลผล |
| Use Case # 8.2 : Input Direct Credit | |
| Main Success Scenario | |
| Step | Action |
| 1 | ระบบแสดงรายการหมายเลขบัญชีของลูกค้าที่มีอยู่ในระบบ |
| 2 | บริษัทเลือกหมายเลขบัญชีที่ต้องการหักเงิน |
| 3 | บริษัททำข้อ 4 ซ้ำกัน ตามความต้องการ |
| 4 | บริษัทป้อนข้อมูล รายละเอียด<br>o รหัสธนาคาร (Bank ID)<br>o หมายเลขบัญชีที่ต้องการเข้าเงิน<br>o จำนวนเงินที่ต้องการ |
| 5 | ระบบคำนวณ จำนวนรายการ ยอดรวมรายการชำระเงิน แสดงผลทางหน้าจอ |
| 6 | บริษัทยืนยันการทำรายการ |
| 7 | ระบบทำการบันทึกข้อมูลในรูปแบบของ ไฟล์ข้อมูล เพื่อรอประมวลผล |
| Use Case # 8.3 : Input Payroll | |
| Main Success Scenario | |

| Step | Action |
|---|---|
| 1 | ระบบจะแสดงหมายเลขบัญชีของบริษัทที่มีอยู่ในระบบ |
| 2 | บริษัทเลือกหมายเลขบัญชีที่ต้องการให้หักเงินออก |
| 3 | บริษัททำข้อ 4 ซ้ำกัน ตามความต้องการ |
| 4 | บริษัทป้อนข้อมูล<br>o รหัสพนักงาน<br>o รหัสธนาคาร<br>o รหัสประจำตัวผู้เสียภาษี<br>o หมายเลขบัญชีของพนักงานที่จะนำเงินเข้า<br>o จำนวนเงินที่ต้องการนำเข้าในแต่ละบัญชี<br>o ยอดจำนวนเงินทั้งหมดที่จะถูกหักออกจากบัญชีของบริษัท |
| 5 | ระบบคำนวณ จำนวนรายการ ยอดรวมรายการชำระเงิน แสดงผลทางหน้าจอ |
| 6 | ระบบทำการบันทึกข้อมูลในรูปแบบของ ไฟล์ข้อมูล เพื่อรอประมวลผล |
| **Use Case # 8.4 : Input Direct Debit** | |
| **Main Success Scenario** | |
| Step | Action |
| 1 | ระบบจะแสดงหมายเลขบัญชีของบริษัทที่มีอยู่ในระบบ |
| 2 | บริษัทเลือกหมายเลขบัญชีที่ต้องการให้หักเงินออก |
| 3 | บริษัททำข้อ 4 ซ้ำกัน ตามความต้องการ |
| 4 | บริษัทป้อนข้อมูล รายละเอียด<br>o รหัสธนาคาร (Bank ID)<br>o หมายเลขบัญชีที่ถูกหักเงิน<br>o จำนวนเงินที่ต้องการ |
| 5 | ระบบคำนวณ จำนวนรายการ ยอดรวมรายการชำระเงิน แสดงผลทางหน้าจอ |
| 6 | ระบบทำการบันทึกข้อมูลในรูปแบบของ ไฟล์ข้อมูล เพื่อรอประมวลผล |
| **Use Case # 8.5 : Submit Payment File** | |
| **Main Success Scenario** | |
| Step | Action |
| 1 | ระบบแสดง Payment File ของบริษัทที่มีอยู่ในระบบ |
| 2 | บริษัทระบุ ชื่อ Payment File ที่จะส่งให้ระบบ Bulk Payment System |

| Step | Action |
|---|---|
| 3 | ระบบทำการส่ง Payment File ที่ระบุชื่อไว้ และชนิดของ Payment File ให้ระบบ Bulk Payment ได้อย่างสมบรูณ์ |
| Use Case # 8.6 : Inquiry Payment File Status | |
| Main Success Scenario | |
| Step | Action |
| 1 | ระบบแสดง Payment File ของบริษัทที่มีอยู่ในระบบ |
| 2 | บริษัทเลือก Payment File ที่ต้องการตรวจสอบสถานะ |
| 3 | ระบบสั่งให้ระบบ Bulk Payment ส่งกลับรายการพร้อมสถานะของแฟ้มข้อมูลกลับมายังระบบได้อย่างสมบรูณ์ |
| Use Case # 8.7 : Download Payment File | |
| Main Success Scenario | |
| Step | Action |
| 1 | ระบบแสดงรายการ Output Payment File และ Report payment file ต่างๆ ของบริษัทที่มีอยู่ในระบบ |
| 2 | บริษัทเลือก Output Payment File และ Report payment Files ที่ต้องการรับข้อมูล |
| 3 | ระบบรับ Output Payment File และ Report payment file จากระบบ Bulk Payment Systemได้อย่างสมบรูณ์ |

7) UC9 : Maintain Own Company - บริษัทสามารถดูแลและกำหนดสิทธิการใช้ระบบของบริษัทตนเอง

7.1) UC9.1 : Create User Id - สร้าง User ID

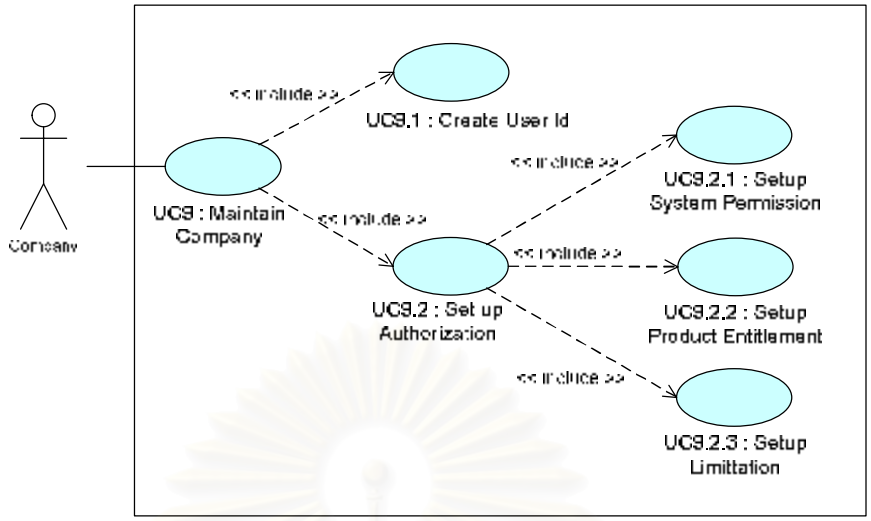
7.2) UC9.2 : Setup Authorization - กำหนดสิทธิการใช้ในแต่ละ User

7.2.1) UC9.2.1 : Setup System Permission - กำหนดสิทธิการใช้ข้อมูลในระบบ

7.2.2) UC9.2.2 : Setup Product Entitlement - กำหนดสิทธิการใช้ บริการทางธุรกิจในระบบ

7.2.3) UC9.2.3 : Setup Limitation - กำหนดวงเงินสูงสุดในแต่ละบริการในระบบ

ดังแสดงในรูปที่ 5.11 และมี Scenario ของแผนภาพ Use Case ดังตารางที่ 5.9




รูปที่ 5.11 แผนภาพ Use Case Company

ตารางที่ 5.9 Scenario ของแผนภาพ Use Case Company

| Use Case # 9 : Maintain Company | |
|---|---|
| Main Success Scenario | |
| Step | Action |
| 1 | บริษัทสามารถเลือกทำขั้นตอน 2 – 3 โดยไม่ต้องเรียงลำดับ |
| 2 | บริษัททำรายการ UC9.1: Create User ID |
| 3 | บริษัททำรายการ UC9.2: Setup Authorization |
| **Use Case # 9.1 : Create User ID** | |
| Main Success Scenario | |
| Step | Action |
| 1 | บริษัทป้อนรายละเอียดที่เกี่ยวกับ User ID<br>o Customer ID และ Company Name<br>o Company Department<br>o User ID และ Password<br>o User Name<br>o Position<br>o E-Mail Address |
| 2 | ระบบตรวจสอบข้อมูลและสร้าง User |
| 3 | ระบบตอบรับการทำรายการ พร้อมกับแสดงข้อความยืนยันการสร้าง User ID |

| Use Case # 9.2 : Setup Authorization | |
|---|---|
| Main Success Scenario | |
| Step | Action |
| 1 | บริษัทสามารถเลือกทำขั้นตอน 2-4 โดยไม่ต้องเรียงลำดับ |
| 2 | บริษัททำรายการ UC9.2.1: Setup System Permission |
| 3 | บริษัททำรายการ UC9.2.2: Setup Product Entitlement |
| 4 | บริษัททำรายการ UC9.2.3: Setup Limitation |
| **Use Case # 9.2.1 : Setup System Permission** | |
| Main Success Scenario | |
| Step | Action |
| 1 | ระบบจะแสดงหน้าจอที่ใช้กำหนดสิทธิ |
| 2 | บริษัททำข้อ 3-4 ซ้ำตามความต้องการ |
| 3 | บริษัทเลือก Function การใช้งาน เพียง 1 function ดังนี้<br>o Company Info<br>o Audit log<br>o Authorization |
| 4 | บริษัทเลือก Action ที่กระทำกับ Function ในข้อ 3<br>o View<br>o Edit |
| 5 | ระบบตอบรับการทำรายการ พร้อมกับแสดงข้อความยืนยันการทำรายการ |
| **Use Case # 9.2.2 : Setup Product Entitlement** | |
| Main Success Scenario | |
| Step | Action |
| 1 | ระบบจะแสดงหน้าจอที่ใช้กำหนดสิทธิ |
| 2 | บริษัททำข้อ 3-4 ซ้ำตามความต้องการ |
| 3 | บริษัทเลือก Function การใช้งาน เพียง 1 function ดังนี้<br>o Direct Debit<br>o Direct Credit<br>o Payroll |

| Step | Action |
|---|---|
| 4 | บริษัทเลือก Action ที่กระทำกับ Function ในข้อ 3<br>o Upload<br>o Edit<br>o Download<br>o Read<br>o Authorized<br>o Cancel<br>o Delete |
| 5 | ระบบตอบรับการทำรายการ พร้อมกับแสดงข้อความยืนยันการทำรายการ |
| **Use Case # 9.2.3 : Setup Limitation** | |
| **Main Success Scenario** | |
| Step | Action |
| 1 | บริษัทป้อนวงเงินสูงสุดแต่ละบริการ<br>o User Limit's Group<br>o Maximum Limit<br>o Fund Transfer Limit<br>o Payment Limit<br>o Total Limit Used |
| 2 | ระบบตอบรับการทำรายการ พร้อมกับแสดงข้อความยืนยันการทำรายการ |

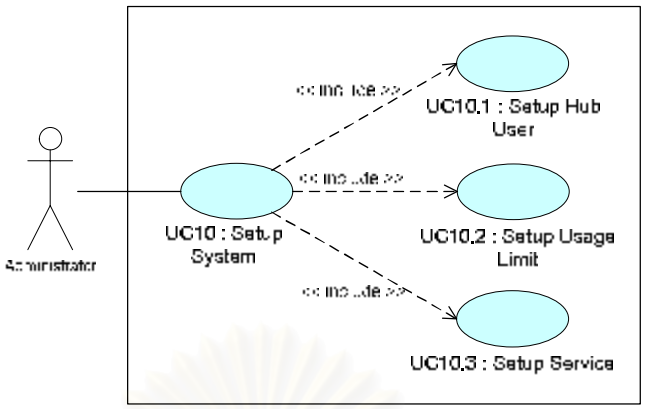
8) UC10 : Prepare Data and Setup - เตรียมข้อมูลรายละเอียดที่ต้องใช้ในระบบ

8.1) UC10.1 : Setup Hub User - กำหนดสิทธิของ Administrator ในการใช้ระบบ

8.2) UC10.2 : Setup Usage Limit - กำหนดวงเงินสูงสุดในแต่ละบริการตามประเภทของบริษัท

8.3) UC10.3 : Setup Service - กำหนดบริการที่เปิดใช้ในระบบ

ดังแสดงในรูปที่ 5.12 และมี Scenario ของแผนภาพ Use Case ดังตารางที่ 5.10




รูปที่ 5.12 แผนภาพ Use Case Setup System

ตารางที่ 5.10 Scenario ของแผนภาพ Use Case Setup System

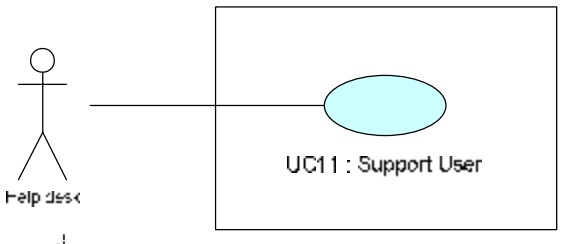
| Use Case # 10 : Prepare Data and Setup | |
|---|---|
| Main Success Scenario | |
| Step | Action |
| 1 | Administrator สามารถเลือกทำขั้นตอน 2 – 4 โดยไม่ต้องเรียงลำดับ |
| 2 | Administrator ทำการ UC10.1: Setup Hub User |
| 3 | Administrator ทำการ UC10.2: Setup Usage Limit |
| 4 | Administrator ทำการ UC10.3: Setup Service |
| **Use Case # 10.1 : Setup Hub User** | |
| Main Success Scenario | |
| Step | Action |
| 1 | Administrator ที่มี Role เป็น Admin ป้อนข้อมูลของเจ้าหน้าที่ที่ต้องรับผิดชอบ<br>o User ID<br>o User Name<br>o Department<br>o Position<br>o User Roles (Manager, Supervisor/Hub Administrator) |
| 2 | Administrator ทำซ้ำข้อ 3-4 ตามต้องการ |
| 3 | Administrator เลือก Function System Permission 1 Function ได้ดังนี้<br>o Hub User<br>o Company Info<br>o Audit log<br>o Authorization |

| Step | Action |
|---|---|
| 4 | Administrator เลือก Action ที่กระทำกับ Function ในข้อ 3 ได้ดังนี้<br>o Add<br>o View<br>o Edit<br>o Approve |
| 5 | Administrator ทำซ้ำข้อ 6-7 ตามต้องการ |
| 6 | Administrator เลือก Function การใช้งานเพียง 1 Function ที่เกี่ยวข้องกับ Account Permission ได้ดังนี้<br>o Fund Transfer<br>o Cheque Inquiry<br>o Stop Cheque |
| 7 | Administrator เลือก Action ที่กระทำกับ Function .ในข้อ 6 ได้ดังนี้<br>o Add<br>o View<br>o Edit<br>o Approve |
| 8 | Administrator ทำซ้ำข้อ 9-10 ตามต้องการ |
| 9 | Administrator เลือก Function การใช้งานเพียง 1 Function ที่เกี่ยวข้องกับ Payment Permission ได้ดังนี้<br>o Bill Payment<br>o Direct Credit<br>o Direct Debit<br>o Payroll |
| 10 | Administrator เลือก Action ที่กระทำกับ Function ในข้อ 9 ได้ดังนี้<br>o View<br>o Edit<br>o Approve<br>o Upload |
| 11 | ระบบตอบรับการทำรายการ พร้อมกับแสดงข้อความยืนยันการทำรายการ |

| Use Case # 10.2 : Setup Usage Limit | |
|---|---|
| Main Success Scenario | |
| Step | Action |
| 1 | ระบบแสดงรายการประเภทบริษัทที่มีอยู่ในระบบ |
| 2 | Administratorเลือก ประเภทบริษัท |
| 3 | Administratorป้อนวงเงินสูงสุดแต่ละบริการตามประเภทของบริษัท<br>o Total Limit<br>o Fund Transfer Limit<br>o Payment Limit |
| 4 | ระบบตอบรับการทำรายการ พร้อมกับแสดงข้อความยืนยันการทำรายการ |
| **Use Case # 10.3 : Setup Service** | |
| Main Success Scenario | |
| Step | Action |
| 1 | ระบบแสดงรายการ บริการ ที่มีอยู่ในระบบ |
| 2 | Administrator เลือก บริการที่เปิดให้ใช้ ดังนี้<br>o Account Information Inquiry<br>o Transfer Funds<br>o Cheque Service<br>o Detail Payment Information Service<br>o Bulk Payment Service |
| 3 | ระบบตอบรับการทำรายการ พร้อมกับแสดงข้อความยืนยันการทำรายการ |

9) UC11 : User Problem - แก้ไขปัญหาให้กับบริษัท กรณีที่บริษัทต้องการความช่วยเหลือ

ดังแสดงในรูปที่ 5.13 และมี Scenario ของแผนภาพ Use Case ดังตารางที่ 5.11



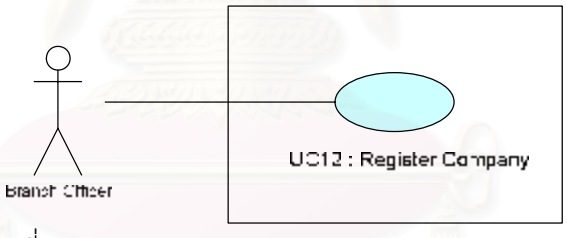

รูปที่ 5.13 แผนภาพ Use Case Support User

ตารางที่ 5.11 Scenario ของแผนภาพ Use Case Support User

| Use Case # 11 : Support User | |
|---|---|
| Main Success Scenario | |
| Step | Action |
| 1 | Help Desk ตรวจสอบการใช้บริการของลูกค้าได้ โดยการค้นหาการทำงานที่ผิดปกติของระบบใน Audit log |
| 2 | Help Desk เลือกค้นหาตามเงื่อนไขที่ต้องการ โดยเลือกจาก keyword เหล่านี้<br>o User ID<br>o Date Time<br>o Service |
| 3 | Help Desk ตอบปัญหาที่บริษัทแจ้งเข้ามา |

10) UC12 : Register Company - ลงทะเบียนการใช้บริการในครั้งแรกให้กับบริษัท

ดังแสดงในรูปที่ 5.14 และมี Scenario ของแผนภาพ Use Case ดังตารางที่ 5.12



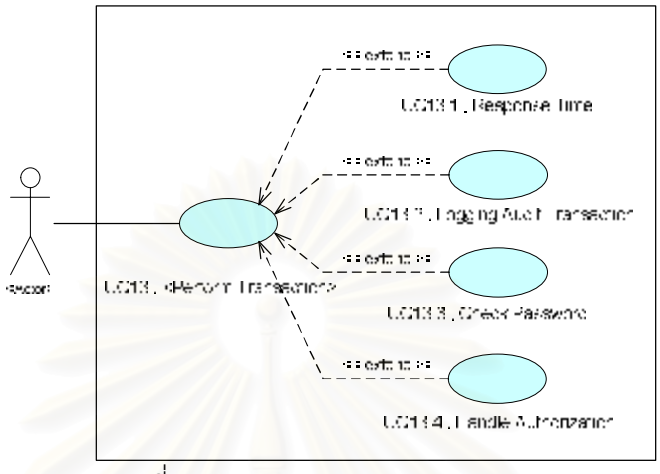

รูปที่ 5.14 แผนภาพ Use Case Register Company

ตารางที่ 5.12 Scenario ของแผนภาพ Use Case Register Company

| Use Case # 12 : Register Company | |
|---|---|
| Main Success Scenario | |
| Step | Action |
| 1 | Branch Officer ป้อน User id ที่มีหน้าที่เป็น Admin ของบริษัทเข้าสู่ระบบ |
| 2 | Branch Officer กำหนดสิทธิการใช้บริการให้กับ User id ที่มี Role เป็น Admin เพื่อให้บริษัท ที่ถือ User id นี้สามารถไปกำหนดสิทธิการใช้ ของ user id อื่น ของบริษัทตัวเองได้ |
| 3 | ระบบบันทึก user id และสิทธิที่กำหนดไว้ |

11) UC13 : Perform Transaction

ดังแสดงในรูปที่ 5.15 และมี Scenario ของแผนภาพ Use Case ดังตารางที่ 5.13



รูปที่ 5.15 แผนภาพ Use Case Infrastructure

ตารางที่ 5.13 Scenario ของแผนภาพ Use Case Infrastructure

| Use Case # 13 : Perform Transaction | |
|---|---|
| Main Success Scenario | |
| Step | Action |
| 1 | ระบบเตรียมพร้อมรับข้อมูลที่ actor ต้องการนำเข้าสู่ระบบ |
| 2 | actor ทำรายการที่ต้องเข้าสู่ระบบ พร้อมทั้งทำการ Submit รายการที่ทำ |
| 3 | ระบบประมวลผลรายการ พร้อมกับแสดงผลที่หน้าจอ |
| 4 | จบการทำงาน |

5.4.3 การกำหนดประเภทของรายการ (Determination of types of the items)

5.4.3.1 Data Functions

1) Internal Logical Files (ILF) ดังตารางที่ 5.14

ตารางที่ 5.14 Internal Logical Files ของระบบงาน CCMS

| ILF Name | Names of RET | Names of DET |
|---|---|---|
| 1. Company Information | 1. บริษัท / ห้างร้าน<br>2. ส่วนราชการ<br>3. รัฐวิสาหกิจ | 1. Company ID<br>2. Company Name<br>3. Company Department<br>4. Company Type |

| ILF Name | Names of RET | Names of DET |
|---|---|---|
| | | 5. E-Mail Address<br>6. Total Limit<br>7. Fund Transfer Limit<br>8. Payment Limit<br>9. Account Information Inquiry Flag<br>10. Transfer Funds Flag<br>11. Cheque Service Flag<br>12. Detail Payment Information Service Flag<br>13. Bulk Payment Service Flag |
| 2. Company Profile | 1. Employee | 1. User ID<br>2. User Name<br>3. Position<br>4. Roles<br>5. Company ID<br>6. Permission Type<br>7. Permission Action<br>8. Product Type<br>9. Product Action<br>10. User Limit's Group<br>11. Maximum Limit<br>12. Fund Transfer Limit<br>13. Payment Limit<br>14. Total Limit Used |
| 3. Internal Officer Profile | 1. Management<br>2. Branch Support<br>3. Electronic Product staff | 1. User ID<br>2. User Name<br>3. Department<br>4. Position |

| ILF Name | Names of RET | Names of DET |
|---|---|---|
| | 4. Auditing Staff<br>5. Legal Staff<br>6. Support Operator | 5. User Roles<br>6. System Permission Type<br>7. System Permission Action<br>8. Account Permission Type<br>9. Account Permission Action<br>10. Payment Permission Type<br>11. Payment Permission Action |
| 4. Account | 1. Saving<br>2. Current | 1. Account Number<br>2. Account Type<br>3. Owner (Company / Customer)<br>5. Company ID<br>6. Bank Of owner account |
| 5. User Profile | 1. Customer<br>2. Officer | 1. User ID<br>2. Password<br>3. Security Type<br>4. User Type |
| 6. Payment File | 1. Bill Payment<br>2. Direct Credit<br>3. Payroll<br>4. Direct Debit | 1. Payment File ID<br>2. Payment File Name<br>3. Payment File Type<br>4. Company ID<br>5. Company Code<br>6. Status |
| 7. Company Type | 1. บริษัท / ห้างร้าน<br>2. ส่วนราชการ<br>3. รัฐวิสาหกิจ | 1. Company Type<br>2. Total Limit<br>3. Fund Transfer Limit<br>4. Payment Limit |

| ILF Name | Names of RET | Names of DET |
|---|---|---|
| 8. Service Type | 1. Account Information Inquiry<br>2. Transfer Funds<br>3. Cheque Service<br>4. Detail Payment Information Service<br>5. Bulk Payment Service | 1. Service Type<br>2. Active/Inactive Flag |
| 9. Outsourcing Cheque | 1. Cheque | 1. Company Id<br>2. Outsourcing file Id<br>3. Outsourcing file Name |
| 10. Company Code | 1. Company | 1. Company ID<br>2. Company Code<br>3. Account Number |
| 11. Audit Log File | 1. Audit Log File | 1. Date<br>2. Time<br>3. Sequence<br>4. Service Type<br>5. Number<br>6. Transaction Amount<br>7. User ID<br>8. EC Flag<br>9. Company ID |

2) External Interface Files (EIF) ดังตารางที่ 5.15

ตารางที่ 5.15 External Interface Files ของระบบงาน CCMS

| ELF Name | Names of RET | Names of DET |
|---|---|---|
| 1. Master Account Information | 1. Deposit<br>2. Loan | 1. Account Number<br>2. Account Name |

| ELF Name | Names of RET | Names of DET |
|---|---|---|
| | | 3. Account Type<br>4. Owner ID<br>5. Ledger Balance |
| 2. Cheque Information | 1. Cheque | 1. Account Number<br>2. Cheque Number<br>3. Status<br>4. Issue Date |
| 3. Cheque Stop | 1. Cheque | 1. Account Number<br>2. Cheque Number<br>3. Stop reason<br>4. Stop By<br>5. Payee Name<br>6. Stop Date |
| 4. Payment Status | 1. Payment file | 1. Payment File<br>2. Status<br>3. Received Date |

5.4.3.2 Transaction Functions ดังตารางที่ 5.16

ตารางที่ 5.16 Transaction Functions ของระบบงาน CCMS

| Use Cases | Transaction |
|---|---|
| 1) UC1: Login<br>o 3 DET (User ID, Password, Security Type)<br>o 1 FTR (User Profile) | EQ |
| 2) UC2.1: Inquiry Account Balance<br>o 3 DET (Company ID, Account Number, Ledger Balance)<br>o 2 FTR (Account, Master Account Information) | EQ |
| o 9 DET (Date, Time, Sequence, Service Type, User ID, EC Flag, Number, Transaction Amount, Company ID)<br>o 1 FTR (Audit Log File) | EI |
| 3) UC2.2: Inquiry Account Statement | EQ |

| Use Cases | Transaction |
|---|---|
| ○ 4 DET (Company ID, Account Number, Date From, Date To)<br>○ 1 FTR (Account) | |
| ○ 9 DET (Date, Time, Sequence, Service Type, User ID, EC Flag, Number, Transaction Amount, Company ID)<br>○ 1 FTR (Audit Log File) | EI |
| 4) UC3: Transfer fund to own Account<br>○ 4 DET (Company ID, Account Number, Amount, Password)<br>○ 2 FTR (Account, User Profile) | EQ |
| ○ 9 DET (Date, Time, Sequence, Service Type, User ID, EC Flag, Number, Transaction Amount, Company ID)<br>○ 1 FTR (Audit Log File) | EI |
| 5) UC6.1: Inquiry Cheque Status<br>○ 4 DET (Account Number, Cheque Number, Issue Date, Status)<br>○ 1 FTR (Cheque Information) | EQ |
| ○ 9 DET (Date, Time, Sequence, Service Type, User ID, EC Flag, Number, Transaction Amount, Company ID)<br>○ 1 FTR (Audit Log File) | EI |
| 6) UC6.2: Stop Single Cheque<br>○ 6 DET (Account Number, Cheque Number, Stop Reason, Stop By, Payee Name, Stop Date)<br>○ 1 FTR (Cheque Stop) | EQ |
| ○ 9 DET (Date, Time, Sequence, Service Type, User ID, EC Flag, Number, Transaction Amount, Company ID)<br>○ 1 FTR (Audit Log File) | EI |
| 7) UC6.3: Cancel Stop Cheque<br>○ 4 DET (Account Number, Cheque Number, Stop By, Stop Date) | EQ |

| Use Cases | Transaction |
|---|---|
| o 1 FTR (Cheque Stop) | |
| o 9 DET (Date, Time, Sequence, Service Type, User ID, EC Flag, Number, Transaction Amount, Company ID)<br>o 1 FTR (Audit Log File) | EI |
| 8) UC6.4: Order Cheque Book<br>o 6 DET (Company Id, Account Number, Account Type, Quantity, Channel, Channel Detail)<br>o 1 FTR (Account) | EQ |
| o 9 DET (Date, Time, Sequence, Service Type, User ID, EC Flag, Number, Transaction Amount, Company ID)<br>o 1 FTR (Audit Log File) | EI |
| 9) UC6.5: Perform Automatic Outsourcing Cheque<br>o 3 DET (Company Id, Outsourcing file Name)<br>o 1 FTR (Outsourcing Cheque, Audit Log File) | EQ |
| o 9 DET (Date, Time, Sequence, Service Type, User ID, EC Flag, Number, Transaction Amount, Company ID)<br>o 1 FTR (Audit Log File) | EI |
| 10) UC7: Inquiry Detail Payment Information<br>o 3 DET (Company Code, Date From, Date To)<br>o 1 FTR (Company Code) | EQ |
| o 9 DET (Date, Time, Sequence, Service Type, User ID, EC Flag, Number, Transaction Amount, Company ID)<br>o 1 FTR (Audit Log File) | EI |
| 11) UC8.1: Input Bill Payment<br>o 12 DET (Company Code, Account Number, Payment File Id, Payment File Name, Payment File Type, Company Id, Status, Reference No, Amount, Date, Total Record, Total payment Amount)<br>o 2 FTR (Company Code, Payment File) | EO |
| 12) UC8.2: Input Direct Credit | EO |

| Use Cases | Transaction |
|---|---|
| o 7 DET ( Account Number, Owner, Bank Id, Account To, Amount, Total Record, Total payment Amount)<br>o 1 FTR (Account) | |
| 13) UC8.3: Input Payroll<br>o 10 DET ( Account Number, Owner, Employee Id, Bank Id, Tax Id, Account To, Amount, Total Amount, Total Record, Total payment Amount)<br>o 1 FTR (Account) | EO |
| 14) UC8.5: Submit Payment File<br>o 7 DET ( Company Id, Company Code, Payment File ID, Payment File Name, Payment File Type, Status)<br>o 1 FTR (Payment File) | EO |
| 15) UC8.6: Inquiry Payment File Status<br>o 3 DET (Company Id, Payment File ID, Status)<br>o 1 FTR (Payment File) | EQ |
| 16) UC8.7: Download Payment File<br>o 5 DET (Company Id, Payment File ID, Payment File Name, Payment File Type, Status)<br>o 1 FTR (Payment File) | EQ |
| 17) UC9.1: Create User ID<br>o 8 DET (Company ID, Company Name, Company Department, User ID, Password, User Name, Position, E-Mail Address)<br>o 3 FTR (Company Information, Company Profile, User Profile ) | EI |
| 18) UC9.2.1: Setup System Permission<br>o 2 DET (Permission Type, Permission Action)<br>o 1 FTR (Company Profile) | EI |
| 19) UC9.2.2: Setup Product Entitlement<br>o 2 DET (Product Type, Product Action) | EI |

| Use Cases | Transaction |
|---|---|
| o 1 FTR (Company Profile) | |
| 20) UC9.2.3: Setup Limitation<br>o 1 DET (Limit service flag )<br>o 1 FTR (Company Information) | EI |
| 21) UC10.1: Setup Hub User<br>o 12 DET (User ID, User Name, Department, Position, User Roles, System Permission Type, System Permission Action, Account Permission Type, Account Permission Action, Payment Permission Type, Payment Permission Action, Password)<br>o 2 FTR (Internal Officer Profile, User Profile) | EI |
| 22) UC10.2: Setup Usage Limit<br>o 3 DET (Total Limit, Fund Transfer Limit, Payment Limit)<br>o 1 FTR (Company Type) | EI |
| 23) UC10.3: Setup Service<br>o 1 DET (Active/Inactive Flag)<br>o 1 FTR (Service Type) | EI |
| 24) UC11: Support User<br>o 9 DET (Date, Time, Sequence, Service Type, User ID, EC Flag, Number, Transaction Amount, Company ID)<br>o 1 FTR (Audit Log File) | EQ |
| 25) UC12: Register Company<br>o 1 DET (Role)<br>o 1 FTR (Company Profile) | EI |
| 26) UC13.2: Logging Audit Transaction<br>o 9 DET (Date, Time, Sequence, Service Type, User ID, EC Flag, Number, Transaction Amount, Company ID)<br>o 1 FTR (Audit Log File) | EI |
| 27) UC13.3: Check Password<br>o 3 DET (User ID, Password, Security Type) | EQ |

| Use Cases | Transaction |
|---|---|
| o 1 FTR (User Profile) | |
| 28) UC13.4: Handle Authorized<br>o 4 DET (User ID, User Type, Roles, User Roles)<br>o 3 FTR (User Profile, Company Profile, Internal Officer Profile) | EQ |

5.4.4 การกำหนดน้ำหนักของปัจจัยสำคัญ (Weighting factors)

5.4.4.1 Data Function ดังตารางที่ 5.17 ในส่วนของ Internal Logical File และดังตารางที่ 5.18 ในส่วนของ External Interface File

ตารางที่ 5.17 การกำหนดน้ำหนักของปัจจัยสำคัญของInternal Logical File (ILF)

| ILF Name | RET | DET | Complexity |
|---|---|---|---|
| 1. Company Information | 3 | 13 | Low |
| 2. Company Profile | 1 | 14 | Low |
| 3. Internal Officer Profile | 6 | 11 | Average |
| 4. Account | 2 | 6 | Low |
| 5. User Profile | 2 | 4 | Low |
| 6. Payment File | 4 | 6 | Low |
| 7. Company Type | 3 | 4 | Low |
| 8. Service Type | 5 | 2 | Low |
| 9. Outsourcing Cheque | 1 | 3 | Low |
| 10. Company Code | 1 | 3 | Low |
| 11. Audit Log File | 1 | 9 | Low |
| | | Total | 10 Low<br>1 Average |

ตารางที่ 5.18 การกำหนดน้ำหนักของปัจจัยสำคัญของExternal Interface File (EIF)

| ELF Name | RET | DET | Complexity |
|---|---|---|---|
| 1. Master Account Information | 2 | 5 | Low |
| 2. Cheque Information | 1 | 4 | Low |
| 3. Cheque Stop | 1 | 6 | Low |
| 4. Payment Status | 1 | 3 | Low |
| | | Total | 4 Low |

5.4.4.2 Transaction Function ดังตารางที่ 5.19

ตารางที่ 5.19 การกำหนดน้ำหนักของปัจจัยสำคัญของ Transaction Function

| Transaction | Complexity | | | Total |
|---|---|---|---|---|
| | Low | Average | High | |
| EI | 16 | 1 | 1 | 18 |
| EO | 3 | 1 | 0 | 4 |
| EQ | 15 | 0 | 0 | 15 |
| | | | Total | 37 |

คำนวณค่า Unadjusted Function Points ของระบบงาน CCMS ได้ดังตารางที่ 5.20

ตารางที่ 5.20 การคำนวณค่า Unadjusted Function Points

| Component Type | Component Complexity | | | |
|---|---|---|---|---|
| | Low | Average | High | Total |
| External Inputs | 16 x 3 = 48 | 1 x 4 = 4 | 1 x 6 = 6 | 58 |
| External Outputs | 3 x 4 = 12 | 1 x 5 = 5 | 0 x 7 = 0 | 17 |
| External Inquiries | 15 x 3 = 45 | 0 x 4 = 0 | 0 x 6 = 0 | 45 |
| Internal Logical Files | 10 x 7 = 70 | 1 x 10 = 10 | 0 x 15 = 0 | 80 |
| External Interface Files | 4 x 5 = 20 | 0 x 7 = 0 | 0 x 10 = 0 | 20 |
| Total Number Of Unadjusted Function Points | | | | 220 |

ดังนั้นระบบงาน CCMS ในเชิง AOP จะมีค่า Function Point เท่ากับ 220 FP

# บทที่ 6

# ผลการวิจัย

ในบทนี้จะกล่าวถึงผลการวิจัยที่ได้ทำการทดสอบโดยนำแผนภาพ Use Case ในเชิงวัตถุของระบบงานที่เป็นกรณีศึกษาจำนวน 12 ระบบ มาแปลงเป็นแผนภาพ Use Case ในเชิงแง่มุมด้วยวิธีการของ Jacobson แล้วมาทำการวิเคราะห์จำนวนฟังก์ชันพอยต์ที่ได้จากแผนภาพทั้งสองตามสมมติฐานการวิจัยที่ได้ตั้งไว้

## 6.1 ผลการนับจำนวนฟังก์ชันพอยต์

ในการวิเคราะห์จำนวนฟังก์ชันพอยต์จากแผนภาพ Use Case จากระบบงาน 12 ระบบ ซึ่งเป็นแผนภาพ Use Case ของระบบงานธนาคารกรุงไทยและบริษัทในเครือ จะทำการนับ 2 แบบโดยการนับแบบที่ 1 จะเป็นการนับจากแผนภาพ Use Case ของระบบงานที่ได้มีการสร้างแผนภาพ Use Case ในเชิง OOP ไว้แล้ว ส่วนการนับแบบที่ 2 จะเป็นการนับจากแผนภาพ Use Case ในเชิง AOP ที่สร้างขึ้นมาใหม่จากข้อมูลความต้องการของระบบเดิมทั้ง 12 ระบบ ซึ่งจะได้ผลดังตารางที่ 6.1

ตารางที่ 6.1 ค่าฟังก์ชันพอยต์ที่นับได้จากแผนภาพ Use Case เชิง OOP และแผนภาพ Use Case เชิง AOP

| Project No. | Total Number Of Unadjusted Function Points | |
|---|---|---|
| | Use Case Diagram based on OOP Technology | Use Case Diagram based on AOP Technology |
| 1 | 211 | 220 |
| 2 | 139 | 142 |
| 3 | 107 | 113 |
| 4 | 110 | 113 |
| 5 | 163 | 169 |
| 6 | 69 | 72 |
| 7 | 79 | 82 |
| 8 | 91 | 94 |
| 9 | 105 | 115 |
| 10 | 92 | 102 |

| Project No. | Total Number Of Unadjusted Function Points | |
|---|---|---|
| | Use Case Diagram based on OOP Technology | Use Case Diagram based on AOP Technology |
| 11 | 91 | 101 |
| 12 | 104 | 107 |

## 6.2 การเปรียบเทียบผลการนับจำนวนฟังก์ชันพอยต์

จากผลการนับฟังก์ชันพอยต์ข้างต้น นำมาเปรียบเทียบกันโดยใช้เทคนิคทางสถิติมาตรวจสอบสมมติฐานการวิจัยดังนี้

1) ค่าฟังก์ชันพอยต์ที่ได้จากการนับแผนภาพ Use Case ด้วยวิธีการของ Jacobson ในเชิง AOP และค่าฟังก์ชันพอยต์ที่ได้จากการนับแผนภาพ Use Case ในเชิง OOP ไม่แตกต่างกัน

สมมติฐานในการวิจัย คือ

$H_0$ : ค่าฟังก์ชันพอยต์เฉลี่ยของแผนภาพ Use Case เชิง OOP และ แผนภาพ Use Case เชิง AOP ไม่แตกต่างกัน

$H_1$ : ค่าฟังก์ชันพอยต์เฉลี่ยของแผนภาพ Use Case เชิง OOP และ แผนภาพ Use Case เชิง AOP แตกต่างกัน

ในการทดสอบนี้ เป็นการเปรียบเทียบความแตกต่างของค่าฟังก์ชันพอยต์ที่ได้จากการนับแผนภาพ Use Case ที่ต่างกันแต่ยังเป็นกลุ่มระบบงานตัวอย่างเดิม จึงใช้สถิติทดสอบ t-test แบบจับคู่

ตารางที่ 6.2 ผลการคำนวณด้วยสถิติทดสอบ t-test แบบจับคู่ของสมมติฐานการวิจัยข้อ 1

| Mean | Std. Deviation | Std. Error Mean | 95% Confidence Interval of the Difference | t | Sig. (2-tailed) |
|---|---|---|---|---|---|
| -5.75 | 3.166 | 0.914 | -7.76 (lower)<br>-3.74 (upper) | -6.292 | 0.000 |

จากตารางที่ 6.2 ค่าความน่าจะเป็น Sig. (2-tailed) มีค่าเท่ากับ 0.000 ซึ่งมีค่าน้อยกว่าระดับนัยสำคัญ $\alpha = 0.05$ ดังนั้นจึงตัดสินใจปฏิเสธสมมติฐาน $H_0$ และสรุปผลได้ว่าค่าฟังก์ชันพอยต์ที่ได้จากแผนภาพ Use Case เชิง OOP และค่าฟังก์ชันพอยต์ที่ได้จากแผนภาพ Use Case เชิง AOP แตกต่างกันที่ระดับนัยสำคัญ 0.05

2) ค่าฟังก์ชันพอยต์ที่ได้จากการนับแผนภาพ Use Case ด้วยวิธีการของ Jacobson ในเชิง AOP จะมีค่ามากกว่าค่าฟังก์ชันพอยต์ที่ได้จากการนับแผนภาพ Use Case ในเชิง OOP

สมมติฐานในการวิจัย คือ

$H_0$ : ค่าฟังก์ชันพอยต์เฉลี่ยของแผนภาพ Use Case เชิง AOP น้อยกว่าหรือเท่ากับค่าฟังก์ชันพอยต์เฉลี่ยของแผนภาพ Use Case เชิง OOP

$H_1$ : ค่าฟังก์ชันพอยต์เฉลี่ยของแผนภาพ Use Case เชิง AOP มากกว่าค่าฟังก์ชันพอยต์เฉลี่ยของแผนภาพ Use Case เชิง OOP

ตารางที่ 6.3 ผลการคำนวณด้วยสถิติทดสอบ t-test แบบจับคู่ของสมมติฐานการวิจัยข้อ 2

| Mean | Std. Deviation | Std. Error Mean | 95% Confidence Interval of the Difference | t | Sig. (1-tailed) |
|---|---|---|---|---|---|
| -5.75 | 3.166 | 0.914 | -7.76 (lower)<br>-3.74 (upper) | -6.292 | 0.000 |

จากตารางที่ 6.3 ค่าความน่าจะเป็น Sig. (1-tailed) มีค่าเท่ากับ 0.000 ซึ่งมีค่าน้อยกว่าระดับนัยสำคัญ $\alpha = 0.05$ ดังนั้นจึงตัดสินใจปฏิเสธสมมติฐาน $H_0$ และสรุปผลได้ว่าค่าฟังก์ชันพอยต์ที่ได้จากแผนภาพ Use Case เชิง AOP มีค่ามากกว่าค่าฟังก์ชันพอยต์ที่ได้จากแผนภาพ Use Case เชิง OOP ที่ระดับนัยสำคัญ 0.05 เมื่อนำความแตกต่างมาทำการคำนวณเพิ่มเติม เพื่อวิเคราะห์หาระดับความแตกต่างนั้น โดยใช้หลักการในการคำนวณดังนี้

$$\frac{\text{Function Points of AOP} - \text{Function Points of OOP}}{\text{Function Points of OOP}} \times 100$$

ซึ่งพบว่าระดับความแตกต่างเพิ่มขึ้นเป็น 5% ของค่าฟังก์ชันพอยต์ที่ได้จากแผนภาพ Use Case เชิง OOP

3) ค่าฟังก์ชันพอยต์ที่ได้จากการนับแผนภาพ Use Case ด้วยวิธีการของ Jacobson ในเชิง AOP โดยผู้วิจัยสร้างขึ้น และค่าฟังก์ชันพอยต์ที่ได้จากการนับแผนภาพ Use Case ด้วยวิธีการของ Jacobson ในเชิง AOP โดยกลุ่มตัวอย่างในสายงานคอมพิวเตอร์สร้างขึ้น ไม่แตกต่างกัน

กลุ่มตัวอย่างในการทดสอบจำนวน 12 คน เพื่อให้แต่ละคนสร้างแผนภาพ Use Case ในเชิง AOP คนละ 1 ระบบ ประกอบด้วย

- Programmer 4 คน
- System Analyst 4 คน
- Project Leader 2 คน
- Business Analyst 2 คน

สมมติฐานในการวิจัย คือ

$H_0$ : ค่าฟังก์ชันพอยต์เฉลี่ยของแผนภาพ Use Case เชิง AOP จากที่ผู้วิจัยสร้างขึ้น และ กลุ่มตัวอย่างสร้างขึ้น ไม่แตกต่างกัน

$H_1$ : ค่าฟังก์ชันพอยต์เฉลี่ยของแผนภาพ Use Case เชิง AOP จากที่ผู้วิจัยสร้างขึ้น และ กลุ่มตัวอย่างสร้างขึ้น แตกต่างกัน

ตารางที่ 6.4 ค่าฟังก์ชันพอยต์ที่นับได้จากแผนภาพ Use Case เชิง AOP ที่ ผู้วิจัยสร้างขึ้น และแผนภาพ Use Case เชิง AOP ที่กลุ่มตัวอย่างสร้างขึ้น

| Project No. | Total Number Of Unadjusted Function Points | |
|---|---|---|
| | Use Case Diagram based on AOP Technology by researcher | Use Case Diagram based on AOP Technology by other |
| 1 | 220 | 243 (System Analyst) |
| 2 | 142 | 151 (Programmer) |
| 3 | 113 | 110 (Project Leader) |
| 4 | 113 | 116 (Programmer) |
| 5 | 169 | 172 (Programmer) |
| 6 | 72 | 91 (System Analyst) |
| 7 | 82 | 85 (Programmer) |
| 8 | 94 | 103 (System Analyst) |
| 9 | 115 | 115 (Business Analyst) |
| 10 | 102 | 102 (Project Leader) |
| 11 | 101 | 101 (Business Analyst) |
| 12 | 107 | 113 (System Analyst) |

จากตาราง 6.4 เป็นค่าฟังก์ชันพอยต์ที่ได้จากการนับแผนภาพ Use Case เชิง AOP จากกลุ่มตัวอย่างและจากผู้วิจัย นำมาวิเคราะห์โดยใช้สถิติทดสอบ t-test แบบจับคู่

ตารางที่ 6.5 ผลการคำนวณด้วยสถิติทดสอบ t-test แบบจับคู่ของสมมติฐานการวิจัยข้อ 3

| Mean | Std. Deviation | Std. Error Mean | 95% Confidence Interval of the Difference | t | Sig. (2-tailed) |
|---|---|---|---|---|---|
| -6.00 | 7.932 | 2.290 | -11.04 (lower)<br>-0.96 (upper) | -2.621 | 0.024 |

จากตารางที่ 6.5 ค่าความน่าจะเป็น Sig. (2-tailed) มีค่าเท่ากับ 0.024 ซึ่งมีค่าน้อยกว่าระดับนัยสำคัญ $\alpha = 0.05$ ดังนั้นจึงตัดสินใจปฏิเสธสมมติฐาน $H_0$ และสรุปผลได้ว่าค่าฟังก์ชันพอยต์ที่ได้จากแผนภาพ Use Case เชิง AOP ที่สร้างจากผู้วิจัยและค่าฟังก์ชันพอยต์ที่ได้จากแผนภาพ Use Case เชิง AOP ที่สร้างจากกลุ่มตัวอย่างแตกต่างกันที่ระดับนัยสำคัญ 0.05

4) ค่าฟังก์ชันพอยต์ที่ได้จากการนับแผนภาพ Use Case ด้วยวิธีการของ Jacobson ในเชิง AOP โดยกลุ่มตัวอย่างในสายงานคอมพิวเตอร์สร้างขึ้น และค่าฟังก์ชันพอยต์ที่ได้จากการนับแผนภาพ Use Case ในเชิง OOP ไม่แตกต่างกัน

สมมติฐานในการวิจัย คือ

$H_0$ : ค่าฟังก์ชันพอยต์เฉลี่ยของแผนภาพ Use Case เชิง OOP และ แผนภาพ Use Case เชิง AOP ที่สร้างโดยกลุ่มตัวอย่างไม่แตกต่างกัน

$H_1$ : ค่าฟังก์ชันพอยต์เฉลี่ยของแผนภาพ Use Case เชิง OOP และ แผนภาพ Use Case เชิง AOP ที่สร้างโดยกลุ่มตัวอย่างแตกต่างกัน

ตารางที่ 6.6 ผลการคำนวณด้วยสถิติทดสอบ t-test แบบจับคู่ของสมมติฐานการวิจัยข้อ 4

| Mean | Std. Deviation | Std. Error Mean | 95% Confidence Interval of the Difference | t | Sig. (2-tailed) |
|---|---|---|---|---|---|
| -11.75 | 7.875 | 2.273 | -16.75 (lower)<br>-6.75 (upper) | -5.168 | 0.000 |

จากตารางที่ 6.6 ค่าความน่าจะเป็น Sig. (2-tailed) มีค่าเท่ากับ 0.000 ซึ่งมีค่าน้อยกว่าระดับนัยสำคัญ $\alpha = 0.05$ ดังนั้นจึงตัดสินใจปฏิเสธสมมติฐาน $H_0$ และสรุปผล

ได้ว่าค่าฟังก์ชันพอยต์ที่ได้จากแผนภาพ Use Case เชิง OOP และค่าฟังก์ชันพอยต์ที่ได้จากแผนภาพ Use Case เชิง AOP ที่สร้างโดยกลุ่มตัวอย่างแตกต่างกันที่ระดับนัยสำคัญ 0.05

5) ในการวัดขนาดซอฟต์แวร์ด้วยวิธี Use Case Points ค่าUse Case Points ที่ได้จากแผนภาพ Use Case ด้วยวิธีการของ Jacobson ในเชิง AOP และค่า Use Case Points ที่ได้จากแผนภาพ Use Case ในเชิง OOP ไม่แตกต่างกัน

ในปัจจุบันนอกจากการวัดขนาดซอฟต์แวร์ด้วยเทคนิคการวิเคราะห์ฟังก์ชันพอยต์ที่นำมาประยุกต์ใช้กับแผนภาพ Use Case แล้ว เทคนิค Use Case Points ก็เป็นอีกวิธีหนึ่งที่ใช้ในการวัดขนาดซอฟต์แวร์ ซึ่งจะทำการวิเคราะห์ในแต่ละ Actor และ use case ของระบบว่ามีความซับซ้อนเพียงใด แล้วจึงกำหนดน้ำหนักในแต่ละ Actor และ use case นั้น เพื่อนำมาคำนวณหาค่า Use Case Points ร่วมกับอีกสองปัจจัย คือ Technical factors และ Environment factors [14] ดังผลที่ได้จากการคำนวณจากแผนภาพ Use Case เชิง OOP กับ แผนภาพ Use Case เชิง AOP ในตารางที่ 6.7

ตารางที่ 6.7 ค่า Use Case Points ของแผนภาพ Use Case เชิง OOP กับ แผนภาพ Use Case เชิง AOP

| Project No. | Total Number Of Use Case Points | |
|---|---|---|
| | Use Case Diagram based on OOP Technology | Use Case Diagram based on AOP Technology |
| 1 | 163 | 172 |
| 2 | 121 | 126 |
| 3 | 73 | 77 |
| 4 | 69 | 73 |
| 5 | 149 | 157 |
| 6 | 56 | 61 |
| 7 | 45 | 49 |
| 8 | 62 | 67 |
| 9 | 94 | 96 |
| 10 | 46 | 50 |

| Project No. | Total Number Of Use Case Points | |
|---|---|---|
| | Use Case Diagram based on OOP Technology | Use Case Diagram based on AOP Technology |
| 11 | 50 | 54 |
| 12 | 84 | 88 |

สมมติฐานในการวิจัย คือ

$H_0$ : ค่า Use Case Points เฉลี่ยของแผนภาพ Use Case เชิง OOP และแผนภาพ Use Case เชิง AOP ไม่แตกต่างกัน

$H_1$ : ค่า Use Case Points เฉลี่ยของแผนภาพ Use Case เชิง OOP และแผนภาพ Use Case เชิง AOP แตกต่างกัน

ตารางที่ 6.8 ผลการคำนวณด้วยสถิติทดสอบ t-test แบบจับคู่ของสมมติฐานการวิจัยข้อ 5

| Mean | Std. Deviation | Std. Error Mean | 95% Confidence Interval of the Difference | t | Sig. (2-tailed) |
|---|---|---|---|---|---|
| -4.833 | 1.898986 | 0.54818 | -6.0399 (lower)<br>-3.6268 (upper) | -8.817 | 0.000 |

จากตารางที่ 6.8 ค่าความน่าจะเป็น Sig. (2-tailed) มีค่าเท่ากับ 0.000 ซึ่งมีค่าน้อยกว่าระดับนัยสำคัญ $\alpha = 0.05$ ดังนั้นจึงตัดสินใจปฏิเสธสมมติฐาน $H_0$ และสรุปผลได้ว่าค่า Use Case Point ที่ได้จากแผนภาพ Use Case เชิง OOP และค่า Use Case Point ที่ได้จากแผนภาพ Use Case เชิง AOP แตกต่างกันที่ระดับนัยสำคัญ 0.05

## 6.3 การวิเคราะห์ผลที่ได้จากการทดสอบสมมติฐาน

เนื่องจากการการสร้างแผนภาพ Use Case ในเชิง AOP โดยใช้แนวความคิดของ Jacobson เป็นการเพิ่ม use case ในส่วนของ Non-Functional Requirements เพิ่มเติมเข้าไปจากแผนภาพ Use Case ปกติ ดังผลที่ได้จากการทดสอบสมมติฐานการวิจัยข้อที่ 1 และ ข้อที่ 2 ซึ่งทำการปฏิเสธ $H_0$ เพราะค่าฟังก์ชันพอยต์ที่ได้จากการนับแผนภาพ Use Case เชิง AOP มีค่ามากกว่าการนับแผนภาพ Use Case เชิง OOP ทุกระบบที่ใช้เป็นกรณีศึกษา โดยมีเปอร์เซ็นต์การเพิ่มขึ้นของค่าฟังก์ชันพอยต์เท่ากับ 5% ถึงแม้ว่าทำการเปลี่ยนวิธีการวัดขนาดซอฟต์แวร์มาเป็นวิธี Use Case Points ดังในสมมติฐานการวิจัยที่ 5 ก็ยังคงได้ผลในแนวทางเดียวกันคือ ค่า Use

Case Points ของทั้งสองแผนภาพมีความแตกต่างกัน เนื่องจากเทคนิค Use Case Points เป็นการวัดขนาดซอฟต์แวร์โดยการนับจำนวนของ actors และ transactions ที่ประกอบอยู่ในแต่ละ use case ดังนั้นเมื่อสร้างแผนภาพ Use Case ในเชิง AOP โดยใช้แนวคิดของ Jacobson ทำให้จำนวน use case เพิ่มขึ้น ค่า Use Case Points จึงเพิ่มขึ้นตามด้วยเช่นกัน

นอกจากนี้ยังได้ทำการทดสอบโดยการให้กลุ่มตัวอย่างทำการทดลองสร้างแผนภาพ Use Case เชิง AOP ตามแนวความคิด AOSD/UC ของ Jacobson แต่เนื่องจากข้อจำกัดในการฝึกอบรมไม่ได้ทำพร้อมกันทั้ง 12 คน อาจทำให้เกิดความคลาดเคลื่อนในการทำความเข้าใจเทคนิค AOSD/UC ของแต่ละบุคคล จึงส่งผลให้แผนภาพ Use Case ในเชิง AOP ที่สร้างขึ้นมีความแตกต่างกันตามความเข้าใจและความคิดเห็นของแต่ละบุคคล เมื่อนำมานับค่าฟังก์ชันพอยต์แล้วเปรียบเทียบกับค่าฟังก์ชันพอยต์ที่นับจากแผนภาพ Use Case เชิง AOP ที่ผู้วิจัยทำการสร้างขึ้นตามสมมติฐานการวิจัยข้อที่ 3 จึงมีความแตกต่างกัน และเมื่อเทียบกับค่าฟังก์ชันพอยต์ที่ได้จากการนับแผนภาพ Use Case เชิง OOP ตามสมมติฐานการวิจัยข้อที่ 4 ผลที่ได้ก็มีความแตกต่างเช่นเดียวกัน ดังนั้นจึงสรุปได้ว่าการสร้างแผนภาพ Use Case ในเชิง AOP ตามแนวคิดของ Jacobson ทำให้ค่าฟังก์ชันพอยต์เปลี่ยนแปลงไป ซึ่งผลการวิจัยนี้เป็นเพียงผลที่ได้จากการวิเคราะห์กลุ่มระบบงานทางด้านการเงินที่มีข้อกำหนดความต้องการของซอฟต์แวร์ ทั้ง Functional Requirements และ Non-Functional Requirements สำหรับงานด้านการเงินโดยเฉพาะ ดังนั้นถ้านำระบบงานทางด้านอื่น ๆ มาทำการวิจัย ผลที่ได้อาจจะแตกต่างจากระบบงานทางด้านการเงิน เนื่องจากข้อกำหนดความต้องการของซอฟต์แวร์ที่แตกต่างกันไปตามลักษณะงาน โดยเฉพาะในส่วนของ Non-Functional Requirements ที่มีผลต่อการสร้างแผนภาพ Use Case ในเชิง AOP ตามแนวคิดของ Jacobson

# บทที่ 7

# สรุปผลการวิจัยและข้อเสนอแนะ

## 7.1 สรุปผลการวิจัย

ผู้วิจัยได้ศึกษาและทำการทดลอง เพื่อตรวจสอบค่าฟังก์ชันพอยต์ที่ได้จากการวัดขนาดซอฟต์แวร์ที่พัฒนาด้วยเทคโนโลยี OOP กับซอฟต์แวร์ที่พัฒนาด้วยเทคโนโลยี AOP จากแผนภาพ Use Case โดยใช้ระบบงานที่เป็นกรณีศึกษาจำนวน 12 ระบบในการทดลอง ซึ่งแบ่งการนับฟังก์ชันพอยต์ออกเป็น 2 แบบ โดยแบบแรกนับฟังก์ชันพอยต์จากแผนภาพ Use Case เชิง OOP ก่อนแล้วจึงมานับฟังก์ชันพอยต์จากแผนภาพ Use Case เชิง AOP ที่สร้างขึ้นมาใหม่ แล้วจึงนำค่าฟังก์ชันพอยต์ทั้งสองมาเปรียบเทียบกันโดยใช้สถิติ t-test ในการทดสอบ ได้ผลสรุปว่าค่าฟังก์ชันพอยต์ที่ได้จากการวัดขนาดซอฟต์แวร์ที่พัฒนาด้วยเทคโนโลยี OOP กับซอฟต์แวร์ที่พัฒนาด้วยเทคโนโลยี AOP มีความแตกต่างกัน ซึ่งการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี AOP จะทำให้ได้ขนาดซอฟต์แวร์ที่เพิ่มขึ้น อาจเนื่องมาจากการประมาณขนาดซอฟต์แวร์ในขั้นตอนการกำหนดความต้องการเป็นเพียงการมองการทำงานของระบบในมุมมองของผู้ใช้งานระบบ แต่เมื่อทำการพัฒนาระบบจริง ๆ โปรแกรมเมอร์ต้องทำการเขียนโค้ดเพิ่มเติมในบางส่วนที่ไม่สามารถแสดงไว้ในแผนภาพ Use Case ได้ Non-Functional Requirements บางอย่างก็เป็นสิ่งจำเป็นที่ต้องทำการเขียนโค้ดเข้าไปในระบบ เช่น การ Logging ของการทำรายการ ดังนั้นการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี AOP ทำให้การประมาณขนาดซอฟต์แวร์ได้ใกล้เคียงกับความเป็นจริงมากยิ่งขึ้น

ผู้วิจัยสามารถสรุปประโยชน์ของงานวิจัยได้ดังต่อไปนี้

1) สามารถนำเทคนิคที่ใช้ในการพิจารณาความต้องการของซอฟต์แวร์ทั้ง Functional Requirements และ Non-Functional Requirements เพื่อสร้างแผนภาพ Use Case ที่ถูกต้องและสมบูรณ์ยิ่งขึ้น เนื่องจากในขั้นตอนของการกำหนดความต้องการของซอฟต์แวร์จะให้ความสำคัญกับความต้องการที่เป็น Functional Requirements มากกว่าความต้องการที่เป็น Non-Functional Requirement ทำให้สร้างแผนภาพ Use Case ขึ้นมาเฉพาะในส่วนของ Functional Requirements ซึ่งจะประสบปัญหาในขั้นตอนของการเขียนโปรแกรมอาจจะทำให้โปรแกรมเมอร์ไม่ได้ให้ความสนใจ Non-Functional Requirement ทั้ง ๆ ที่เป็นความสามารถของระบบที่ควรจะมี ถ้าทำการออกแบบแผนภาพ Use Case ได้ถูกต้องสมบูรณ์เท่าไหร่ การพัฒนาระบบก็จะถูกต้องสมบูรณ์ตามไปด้วย

2) สามารถนำเทคนิคที่ใช้ในการสร้างแผนภาพ Use Case ในเชิง AOP เป็นแนวทางในการสร้างหรือปรับเปลี่ยนแผนภาพ Use Case ที่ใช้ในระบบงานปัจจุบัน เนื่องจากระบบงานที่ใช้งานแล้วบางระบบอาจมีการปรับเปลี่ยนความต้องการของระบบ ซึ่งอาจทำให้ต้องทำการแก้ไขโปรแกรมเกือบทั้งหมด ทำให้เสียค่าใช้จ่ายเพิ่มสูงขึ้นและใช้เวลานานในการพัฒนาใหม่ เนื่องจากแนวความคิด AOP สนับสนุนการนำกลับมาใช้ใหม่และแตกหน่วยย่อยออกมา ดังนั้นการเพิ่มเติมความต้องการใหม่หรือแก้ไขปรับเปลี่ยนการทำงานของระบบ ก็สามารถทำได้ง่ายและรวดเร็วขึ้น

3) สามารถนำเทคนิคการวิเคราะห์ฟังก์ชันพอยต์มาประยุกต์ใช้กับ AOP ได้ เนื่องจากในปัจจุบันแนวโน้มการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี AOP เพิ่มสูงขึ้นจึงจำเป็นต้องมีการประมาณค่าใช้จ่ายก่อนการพัฒนา การนำเทคนิคการวิเคราะห์ฟังก์ชันพอยต์ก็เป็นแนวทางหนึ่งที่สามารถนำมาวัดขนาดซอฟต์แวร์เพื่อประมาณค่าใช้จ่าย

## 7.2 ข้อเสนอแนะ

7.2.1 ในปัจจุบันนักวิจัยทางด้านวิศวกรรมซอฟต์แวร์ (Software Engineering) ได้ทำการค้นคว้าแนวคิดทางด้าน AOP เพิ่มขึ้นเรื่อย ๆ ซึ่งอาจจะมีแนวทางใหม่ที่เหมาะสมกว่าแนวความคิดที่ใช้ในงานวิจัยนี้ และทำให้ได้ผลลัพธ์ที่สมบูรณ์และถูกต้องยิ่งขึ้น

7.2.2 การนำเทคนิคการวิเคราะห์ฟังก์ชันพอยต์มาประยุกต์ใช้กับแผนภาพ Use Case ในงานวิจัยนี้เป็นเพียงแนวทางหนึ่ง ผู้ที่สนใจสามารถนำเสนอแนวทางอื่น ๆ ที่เหมาะสมกับแผนภาพ Use Case มาใช้งานได้

7.2.3 ถึงแม้ว่าการประยุกต์ใช้เทคนิคฟังก์ชันพอยต์กับ AOP จะทำให้ค่าฟังก์ชันพอยต์ที่ได้มีค่าเปลี่ยนแปลงไป แต่การประมาณขนาดซอฟต์แวร์ด้วยวิธีอื่น เช่น การนับ Lines of code ก็น่าจะทำให้ขนาดซอฟต์แวร์ลดลง เนื่องจากแนวความคิดของ AOP แยกเอาส่วนที่เหมือนกันออกมาไปนิยามไว้ที่เดียวกัน เป็นการลดบรรทัดในการเขียนโปรแกรม ก็น่าจะทำให้ขนาดซอฟต์แวร์มีขนาดเล็กลงตามไปด้วย

# รายการอ้างอิง


1. Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, V., Loingtier, J. M., and Irwin, J. Aspect-Oriented Programming. The European Conference on Object-Oriented Programming (ECOOP), Finland. Springer-Verlag LNCS 1241, June 1997.
2. Fetcke, T., Abran, A., and Nguyen, T. Mapping the OO-Jacobson Approach into Function Point Analysis. 1998.
3. Lee, K. An Introduction to Aspect-Oriented programming. COMP 610E Spring Software Development of E-Business Applications, The Hong Kong University of Science and Technology, 2002.
4. IFPUG. Function Point Counting Practices Manual Release 4.1. International Function Point Users Group, Westerville, Ohio, 1999.
5. Laddad, R. Aspect-Oriented Programming Will Improve Quality. IEEE, 2003.
6. Elrad, T., Aksit, M., Kiczales, G., Lieberherr, K., and Ossher, H. Discussing Aspects of AOP. Communications of ACM, October 2001.
7. Abrahao, S., Poels, G., and Pastor, O. Comparative Evaluation of Function Size Measurement Methods: An Experimental Analysis. Faculty of Economics and Business Administration, Ghent University, March 2004.
8. Anda, B., Dreiem, H., Dag, S., and Margne, J. Estimating Software Development Effort based on Use Cases – Experiences from Industry. Department of Informatics, University of Oslo, 2002.
9. Longstreet, D. Use Cases and Function Points [Online]. Available from: http://www.ifpug.com/Use Cases and Function Points.htm [2004, June 19]
10. Pace, D., Calavaro, G. and Cantone, G. Function Point and UML : State of Art and Evaluation Model. SMEF 2004, Rome, Italy, 28 – 30 January 2004.
11. Iorio, T. IFPUG Function Point Analysis in a UML framework. SMEF 2004, Rome, Italy, 28 – 30 January 2004.
12. เมสินี นาคมณี. การวางแผนโครงการพัฒนาซอฟต์แวร์. พิมพ์ครั้งที่1. กรุงเทพมหานคร: สำนักพิมพ์ แว่นแก้ว จำกัด, 2547.

13. Jacobson, I., and Ng, P.-W. Aspect-Oriented Software Development with Use Cases. United States: Addison Wesley Professional, 2005.

14. Carroll, R., E. Estimating Software Based on Use Case Points. OOPSLA'05, California, USA, 16 – 20 October 2005.

15. Mohagheghi, P., Anda, B., and Conradi, R. Effort Estimation of Use Cases for Incremental Large-Scale Software Development. ICSE'05, St Louis, Missouri, USA, 15 – 21 May 2005.

# ภาคผนวก

# ภาคผนวก ก

Requirements Specification Document

**Project Name :** Xxxxxxxxxxxxxxxxx

**Project ID** yynnn (e.g. 04001, 05034)

**Document Name:** Requirements Specification

**Prepared by :** Somchai Charoendee *(position)*

**Reviewed by :** Somsri Meesuk *(position)*

| | S-Date | R-Date | # Pages | # Majors | # Minors | Minutes |
|---|---|---|---|---|---|---|
| 1. | 4 Feb. 2004 | 7 Feb. 2004 | 5 | 2 | 14 | 25 |
| 2. | 10 Feb. 2004 | 15 Feb. 2004 | 6 | 4 | 8 | 15 |
| 3. | 27 Feb. 2004 | 5 Mar. 2004 | 8 | 3 | 5 | 45 |
| . | | | | | | |
| . | | | | | | |
| . | | | | | | |
| . | | | | | | |
| . | | | | | | |
| . | | | | | | |
| . | | | | | | |
| . | | | | | | |
| . | | | | | | |
| . | | | | | | |

***Sign-off by*** _____

***Requirements Specification***

The purpose of Requirements Specification is to set up business cases that can be used to justify the engagement for further development. Note that justification is multi-dimensional although cost is often in focus. To set up business cases one must understand the situation and be able to envision the evolution of work that will bring about explainable added values.

Requirements specification is the input to the analysis and design process. Furthermore it serves as a guideline for producing test documents. Therefore the level of detail must be sufficient to serve those purposes.

Beware of any open-ended descriptions or issues. They are sources of risk that are likely to affect cost, time, and user satisfaction. Try to list, itemize, or tabulate everything. If open-ended issues arise, state clearly who and how to resolve them and what will be the impact.

The audiences of this document are:

1. Business analysts for verifying the correctness and completeness of the system.
2. System analysts and designers for developing the System Design Specification.
3. Quality assurors for developing the Test Specification.

**Table of Contents**

## 1. General Product Description

### 1.1 Purpose

Description of the situation that triggered the project. A few sentences of project definition (goal and boundary). This can be written in an executive summary style. Half a page to one page is fine.

### 1.2 Customer and Stakeholders

Customer is the one who pays for the project. Stakeholders are those who are affected by the project and those whose input is needed for building the system. Typical stakeholders include:

- Project sponsor
- Business domain expert
- Product manager
- Technology people (development, operation,…)
- Tester

List them in a table style indicating names, affiliations, and roles. For example

| Name | Affiliation | Role |
|---|---|---|
| 1. Mr. ABC | EVP | Project sponsor |
| 2. Mrs. PQR | Credit Dept. | Credit evaluation method, Credit checking |
| 3. Mr. XYZ | Compliance Dept. | Regulatory report input. |
| 4. Mr. DEF | Accounting Dept. | Account posting input |
| 5. Mr. JKL | IT dev. manager | System development |
| … | … | … |

Table 1.1 List of Stakeholders

### 1.3 Users

Users or end-users are those who use or operate the system. They are not necessary the same person as the customer. List them by categories and describe their tasks, capabilities, and so on. Also rate or prioritize them into categories

- Key users: frequent users.
- Secondary users: moderate usage users.
- Ternary users: infrequent users.

Tabulation of user groups and their characteristics should be shown.

| User Group | Priority | Usage |
|---|---|---|
| 1. branch teller | key | daily |
| 2. accounting staff | key | cycle |
| 3. retailed banking customer | secondary | daily |
| … | … | … |
| … | … | … |

From the table, describe in more detail of each user group. Details include group size, responsibility or role (how they use the system), IT and language skills.

### 1.3.1 Branch Tellers

There are about 4,500 tellers working in over 620 branches. Their task are to input transaction data (both Thai and English) into the system. Tellers are mostly female with collegiate degree education. They are process-oriented and have skill in basic computer usage. They use the system to …

### 1.3.2 Bank Retailed Customers

There are about 4 million retailed customers using the system via ATM's. Only low percentage of them can understand English. Most have low IT skill. They use the system to …

### 1.3.3 Bank Corporate Customers

There are over 40,000 corporate customers and probably 25 % of them will adopt the system in the first year. Within 3 years after launch most of them will use the system. These group will get services via the Internet. They have medium IT skill and are English literate. They use the system to …

## 1.4 Assumptions

This section describe

1. Constraints: operating environment, workplace environment, time, cost.
2. External factors that may impact the system.
3. Other assumptions on resources, tools, dependencies on other systems, regulation.

**2. Functional Requirements**

This section focuses on the mandatory functional requirements of end users. The summary use case model is used to identify them. The software scope is bound by the model. The detailed use case specifications are presented separately in an appendix so that further adjustment and modification will not affect the main document.

**2.1 Summary Use Case Model**

This model defines the overview of the target software system in terms of interaction dialogs between the actors and software system. A stacked use case is usually used and the primary actors will be included with the top level use cases without any sub use cases.

**2.1.1 Summary Use Cases Diagram**

**2.1.2 Summary Use Case Description**

A narrative context is written to explain the summary use case.

**2.2 Product Scope: Mandatory Functional Requirement List (Feature Lists)**

Provide a paragraph describing the functional structure of the system. This section should be extensive. Subsection should be created to categorize groups of functions. In each subsection, list all the functions as ones can think of.

**2.3 Functional Requirements to Use Cases Mapping**

All of the functional requirements listed previously will be considered and the business use cases will be defined to serve each functional requirements item. A table is simply drawn as follows:

| Functional Requirements | Related Use Case Name | Primary Actors |
|---|---|---|
| 1) The system shall provide a catalog of the products | UC1: Insert product to catalog,<br>UC2: Delete product to catalog,<br>UC3: Update product to catalog | Company Clerk, Sale Manager |
| | | |

**3. Non-functional Requirements**

Non-functional requirements are those that are subjective and not straight forward to test. They must be included for setting a guideline to accept the system.

**3.1 Usability**

This section addresses the look and feel of the system. Ease of use and ease of learning should also be included. Here is an area that is difficult to test but at least should give guidelines for developing the system.

**3.2 Performance**

List measurable performance criteria. This can relates to speed, load, and ratios. For examples

1. Response time of the 80 % of the transactions must be less than 5 seconds on a LAN terminal.
2. Interest posting of 10 million saving accounts must be less than 2 hours.
3. …
4. …
5. …

**3.3 Operational Environment**

Describe physical environment in which the system will operate. Technological environment can also be included; e.g. adjacent interacting devices or system. This will help prepare integrating test.

**3.4 Security**

Specify the security structure of the system; e.g. who authorized the access to what and under what circumstances. What logged information is needed.

**3.5 User Document**

This section list all the documents or manuals that will be delivered as parts of the system. A table showing document names, purposes, intended readers, and content briefs should be sufficient.

| Document Name | Intended User | Purpose | Content Brief |
|---|---|---|---|
| 1. User manual | General users | Reference guide | Synopsis of all functions |
| 2. Tutorial | General users | Self learning | General use of the system |
| 3. Operational guide | IT operators | Operating the system | Monitoring, control, and trouble shooting |
| 4. … | … | … | … |
| 5. … | … | … | … |

**3.6 Maintainability**

Explain the maintenance process; i.e. change triggering and who is doing what.

**4. Project Issues**

**4.1 Time and Schedule**

Break down into tasks and subtasks. Identify steps and responsible persons. The intention here is to roughly show the overall view of activities, time line, and staff involved.

**4.2 Resource and Cost**

Separate the system into development, testing, operation, and backup. Itemized all the components. For man-days break down to what kind of people (consultants, programmers, analysts, …), time spans, and unit cost. Adding contingent reserve of 10-15 % can be done.

**4.3 Cutover and Phasing**

Specify how the system will be delivered. This should include data conversion and roll-out strategy; e.g. big bang or phasing what, where, and when. It is important to identify roll-back strategy.

**4.4 Risk**

Signify any risk that may derail the project or cause deviation of resource consumption. Also rank the risk according to severity and likelihood and identify how to manage them.

# Appendix A

## Existing System

This appendix briefly explains the existising system. If no existing system is in place, state the non-existence. The goal is to list deficiencies/limitations and describe the gap between the existing system and the new system. Details should include operations, software configuration/architecture, and hardware configuration.

# Appendix B

## Use Case Specification

**Use Case Name #1**

***Brief Description***

Flow of Events

***Basic Flow***

***Alternative Flows***

Special Requirements
Preconditions
Postconditions
Extension Points
-

**Use Case Name #2**

.
.
.

**Use Case Name #3**

.
.
.

# Appendix C

## Glossary

This appendix lists all the relevant definitions of technical and business-specific terms used in the document. Terms must be sorted in alphabetical order. All Thai terms precede English terms.

# ประวัติผู้เขียนวิทยานิพนธ์

นางสาวเสาวลักษณ์ รัตนธรรมสกุล เกิดเมื่อวันที่ 5 สิงหาคม พ.ศ. 2519 ที่จังหวัดนครศรีธรรมราช สำเร็จการศึกษาหลักสูตรปริญญาสถิติศาสตร์บัณฑิต (สต.บ.) สาขาเทคโนโลยีสารสนเทศเพื่อธุรกิจ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา 2543 หลังจากนั้นได้ทำงานในบริษัท กรุงไทยคอมพิวเตอร์เซอร์วิสเซส จำกัด (KCS) จนถึงปัจจุบัน (พ.ศ.2549)