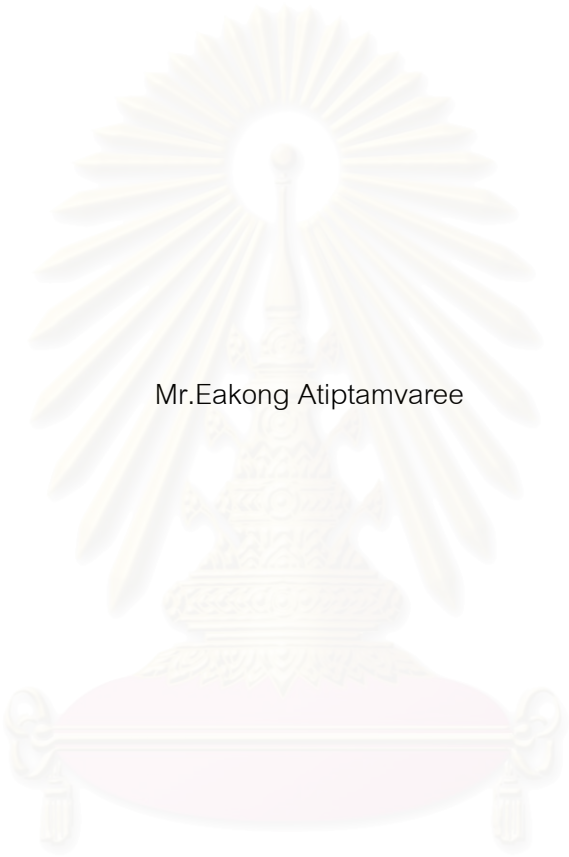
การวัดการออกแบบส่วนประกอบโพรเซสในการบรรลุเป้าหมายด้านการจัดการส่วนประกอบ

นายเอกองค์ อธิปธรรมวารี

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2549
ISBN 974-14-2935-5

MEASURING PROCESS COMPONENT DESIGN ON ACHIEVING COMPONENT MANAGEMENT GOALS

Mr.Eakong Atiptamvaree



A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science Program in Software Engineering
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2006

ISBN 974-14-2935-5

หัวข้อวิทยานิพนธ์	การวัดการออกแบบส่วนประกอบโพรเซสในการบรรลุเป้าหมายด้านการจัดการส่วนประกอบ
โดย	นาย เอกองค์ อธิปธรรมวารี
สาขาวิชา	วิศวกรรมซอฟต์แวร์
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์

(ศาสตราจารย์ ดร.ดิเรก ลาวัณย์ศิริ)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ

(อาจารย์ ดร.ยรรยง เต็งอำนวย)

..... อาจารย์ที่ปรึกษา

(ผู้ช่วยศาสตราจารย์ ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา)

..... กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ)

..... กรรมการ

(อาจารย์ ดร.ดาริชา สุธีวงศ์)

นายเอกองค์ อธิปธรรมวารี : การวัดการออกแบบส่วนประกอบโพรเซสในการบรรลุเป้าหมายด้านการจัดการส่วนประกอบ. (MEASURING PROCESS COMPONENT DESIGN ON ACHIEVING COMPONENT MANAGEMENT GOALS) อ. ที่ปรึกษา : ผศ. ดร.ทวีตีย์ เสนีวงศ์ ณ อยุธยา, 114 หน้า. ISBN 974-14-2935-5.

การพัฒนาซอฟต์แวร์โดยใช้ส่วนประกอบซอฟต์แวร์สนับสนุนแนวคิดการนำซอฟต์แวร์กลับมาใช้ใหม่โดยการนำส่วนประกอบซอฟต์แวร์ที่มีอยู่มาประกอบกันเพื่อสร้างซอฟต์แวร์ใหม่ได้อย่างรวดเร็ว การออกแบบส่วนประกอบซอฟต์แวร์สำหรับโดเมนงานหนึ่ง ๆ จะทำโดยการสร้างแบบจำลองซอฟต์แวร์ในรูปแบบของแผนภาพคลาส แล้วจัดกลุ่มคลาสออกเป็นแผนภาพคลาสย่อย ๆ และนำแต่ละแผนภาพย่อยไปพัฒนาต่อเป็นส่วนประกอบซอฟต์แวร์ เพื่อนำไปใช้ประกอบเป็นส่วนหนึ่งของแอพพลิเคชันใหม่ ๆ ต่อไป ในปัจจุบันการพัฒนาซอฟต์แวร์มีแนวโน้มที่จะใช้การพัฒนาเชิงกระบวนการมากขึ้น โดยพัฒนาซอฟต์แวร์ขึ้นมาจากแบบจำลองกระบวนการทางธุรกิจของโดเมนงานซึ่งกำหนดโดยผู้ใช้ หรือนักวิเคราะห์ธุรกิจ ที่มีความเข้าใจในโดเมนงาน จากนั้นจึงค่อยนำแบบจำลองกระบวนการทางธุรกิจนี้ไปพัฒนาเป็นซอฟต์แวร์ต่อไป ผู้วิจัยจึงเห็นว่าน่าจะสามารถประยุกต์ใช้แนวคิดของส่วนประกอบซอฟต์แวร์กับการพัฒนาเชิงกระบวนการได้

งานวิจัยนี้ใช้แบบจำลองกระบวนการทางธุรกิจของโดเมนงาน ซึ่งอยู่ในรูปของแผนภาพแอคทิวิตีแทนการใช้แบบจำลองในรูปของแผนภาพคลาส แล้วแบ่งแบบจำลองออกเป็นกระบวนการย่อย ๆ เพื่อให้สามารถนำแต่ละกระบวนการย่อยไปพัฒนาต่อเป็นส่วนประกอบโพรเซส สำหรับใช้ประกอบเป็นแอพพลิเคชันใหม่ต่อไปในอนาคต งานวิจัยได้เสนอมาตรวัด เพื่อวัดลักษณะทางเทคนิคของการออกแบบส่วนประกอบโพรเซส พร้อมเสนอแบบจำลองเชิงคุณภาพ เพื่อหาค่าดัชนีการบรรลุเป้าหมายด้านการจัดการซึ่งบ่งบอกว่าการออกแบบบรรลุเป้าหมายด้านการจัดการส่วนประกอบที่กำหนดไว้เพียงใด ซึ่งมาตรวัดและแบบจำลองเชิงคุณภาพนี้ประยุกต์มาจากมาตรวัดและแบบจำลองเชิงคุณภาพที่ใช้ในการออกแบบส่วนประกอบซอฟต์แวร์ นอกจากนี้งานวิจัยได้เสนอเครื่องมือช่วยคำนวณค่ามาตรวัดการออกแบบส่วนประกอบโพรเซส ซึ่งสามารถนำมาใช้ในการเปรียบเทียบว่า การออกแบบส่วนประกอบโพรเซสในลักษณะใด จึงบรรลุเป้าหมายด้านการจัดการส่วนประกอบได้ดีกว่ากัน

ภาควิชา.....วิศวกรรมคอมพิวเตอร์..... ลายมือชื่อนิสิต.....

สาขาวิชา...วิศวกรรมซอฟต์แวร์..... ลายมือชื่ออาจารย์ที่ปรึกษา.....

ปีการศึกษา 2549

4770551721: MAJOR SOFTWARE ENGINEERING

KEY WORD: PROCESS COMPONENT / DESIGN / COMPONENT MANAGEMENT GOALS / BUSINESS PROCESS MODEL / MEASUREMENT

EAKONG ATIPTAMVAREE : MEASURING PROCESS COMPONENT DESIGN ON ACHIEVING COMPONENT MANAGEMENT GOALS. THESIS ADVISOR : ASSIST.PROF. TWITTIE SENIVONGSE, Ph.D., 114 pp. ISBN 974-14-2935-5.

Developing software by software components promotes reuse of software by rapid assembly of available software components into a new software application. Software components for a particular application domain are designed by grouping classes in the class diagram of the domain into subdiagrams. Each subdiagram is used as a model for later development of a software component. At present, software development is moving towards process-oriented development by developing software from business process models, which are defined by users of the domains or business analysts. The author sees a possibility to apply the concept of software components to process-oriented development.

This research proposes metrics for measuring technical features of process components design and determines, by using a quality model, the resulting index which indicates how well the design achieves the predefined managerial goals. These metrics and the quality model are adapted from those used in software components design. This research also presents a software tool that measures the quality of process components design and can help to compare which of the various designs for a particular domain better achieves the managerial goals.

Department.....Computer Engineering...Student's Signature.....

Field of Study.....Software Engineering.....Advisor's Signature.....

Academic Year 2006.....

กิตติกรรมประกาศ

วิทยานิพนธ์นี้สำเร็จลงด้วยความกรุณาเป็นอย่างสูงของผู้ช่วยศาสตราจารย์ ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา อาจารย์ที่ปรึกษาวิทยานิพนธ์ของข้าพเจ้า ที่ให้ความรู้ทั้งหลายมากไปกว่าทางด้านการเรียน อาจารย์ได้ให้คำปรึกษาดี ๆ ที่นำมาใช้ได้ในชีวิต ให้กำลังใจและแก้ปัญหาให้ข้าพเจ้ามาตลอด สิ่งต่าง ๆ ที่ข้าพเจ้าทำผิดพลาดไปข้าพเจ้ากราบขออภัย และกราบขอบพระคุณอาจารย์เป็นอย่างสูงไว้ ณ ที่นี้ด้วย

ขอขอบพระคุณ อาจารย์ ดร.ยรรยง เต็งอำนวย ประธานกรรมการสอบ ผศ.ดร.วิวัฒน์ วัฒนาวุฒิ และ อ.ดร.ดาริชา สุธีวงศ์ กรรมการสอบของข้าพเจ้าที่กรุณาให้คำแนะนำทำให้งานวิจัยชิ้นนี้มีความถูกต้องและสมบูรณ์มากขึ้น และขอขอบพระคุณคำชี้แนะดี ๆ ที่จะเป็นประโยชน์ต่อการดำรงชีวิตของข้าพเจ้าต่อไปในภายภาคหน้า

ขอขอบพระคุณอาจารย์ทุกท่านที่ให้ความรู้แก่ข้าพเจ้าเพื่อใช้ในการทำงานวิจัยนี้ รวมถึงคำแนะนำดี ๆ ที่ให้แก่ข้าพเจ้าตลอดมา

ขอขอบคุณพี่ ๆ น้อง ๆ ที่ห้องปฏิบัติการวิศวกรรมระบบสารสนเทศ ที่ช่วยให้ชีวิตการทำงานวิจัยมีความสุขไม่เงียบเหงาและความอบอุ่นที่มีให้ ขอบคุณสำหรับคำแนะนำดี ๆ และกำลังใจที่ให้มาตลอด

ขอขอบคุณเพื่อน ๆ และพี่ ๆ หลักสูตรวิศวกรรมซอฟต์แวร์ที่ให้คำปรึกษากำลังใจและความรู้แขนงต่าง ๆ ที่ช่วยให้งานวิจัยนี้สมบูรณ์ขึ้น

ขอบคุณที่ให้คำสอนดี ๆ ขอบคุณพี่ชายทั้งสองที่ให้กำลังใจให้คำแนะนำที่ดีในการทำวิทยานิพนธ์และการดำเนินชีวิตที่ถูกต้อง

ขอขอบพระคุณ คุณพ่อ และคุณแม่ที่ให้กำเนิดข้าพเจ้า ให้กำลังใจข้าพเจ้าตลอดมา ให้อภัยข้าพเจ้าตลอดมา คุณความดีของวิทยานิพนธ์ฉบับนี้ ขอมอบเป็นเครื่องบูชาคุณบิดา มารดา ไว้ ณ โอกาสนี้

สารบัญ

หน้า

หน้า

หน้า

สารบัญภาพ

ภาพประกอบ หน้า

สารบัญภาพ

ภาพประกอบ หน้า

สารบัญตาราง

ตาราง หน้า

สารบัญตาราง

ตาราง หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในการพัฒนาแอพพลิเคชันนั้น การออกแบบนิยมใช้แผนภาพคลาส (Class Diagram) [1, 2, 3] เป็นหลักเพื่อใช้วิเคราะห์ปัญหาออกมาเป็นโครงสร้างของแอพพลิเคชันที่จะทำการพัฒนา แต่ในปัจจุบันมีแนวความคิดใหม่ที่จะพัฒนาจากแบบจำลองกระบวนการทางธุรกิจ (Business Process Model) เรียกว่าการพัฒนาเชิงกระบวนการ (Process-Oriented Development) [4] โดยวิเคราะห์ปัญหาในมุมมองของกระบวนการทำงานของแอพพลิเคชันว่าประกอบด้วยกิจกรรมอะไรบ้าง และทำเป็นลำดับก่อนหลัง หรือมีเงื่อนไขในการทำงานอย่างไร การมองปัญหาแบบนี้นักออกแบบและนักพัฒนาซอฟต์แวร์สามารถมองปัญหาได้ง่ายกว่า เพราะสามารถแปลงจากความต้องการของผู้ใช้หรือนักวิเคราะห์ธุรกิจ (Business Analyst) ของโดเมนงานได้โดยตรง ซึ่งการอธิบายความต้องการมักอยู่ในลักษณะของกระบวนการทำงานที่ต้องการให้แอพพลิเคชันทำ การพัฒนาแอพพลิเคชันโดยใช้แบบจำลองกระบวนการทางธุรกิจเป็นหลักนั้น ทุกฝ่ายที่เกี่ยวข้องกับการทำงาน ตั้งแต่ผู้ใช้ นักวิเคราะห์ธุรกิจ นักออกแบบและนักพัฒนาซอฟต์แวร์ ตลอดจนถึงผู้บริหารที่ดูแลจัดการกระบวนการ จะเข้าใจตรงกันได้ง่าย อีกทั้งแบบจำลองกระบวนการทางธุรกิจนี้ยังสนับสนุนการประมวลผลแอพพลิเคชันผ่านภาษาประมวลผลกระบวนการ (Process Execution Language) [5] เช่น บีเพล (BPEL) [6, 7] ดังนั้นการพัฒนาแอพพลิเคชันจึงทำได้อย่างรวดเร็ว การพัฒนาเชิงกระบวนการนี้สอดคล้องกับหลักการของเอสโอซี (SOC: Service Oriented Computing) [8] ซึ่งสนับสนุนการพัฒนาแอพพลิเคชันจากการออเคสเตรชัน (Orchestration) แอพพลิเคชันที่มีอยู่ซึ่งเรียกว่าเซอร์วิส (Service) ให้อยู่ในรูปกระแสของการทำงานร่วมกัน โดยเซอร์วิสจะมีส่วนต่อประสาน (Interface) เพื่อให้เซอร์วิสอื่น ๆ เรียกใช้งานและมีการส่งผ่านข้อมูลหรือกำหนดเงื่อนไขในการทำงาน

การพัฒนาแบบเดิมด้วยแผนภาพคลาสเป็นหลักนั้น ใช้ในแนวคิดการพัฒนาเป็นส่วนประกอบซอฟต์แวร์ (Software Component) โดยจับกลุ่มคลาสภายในโดเมนงานเป็นกลุ่มเพื่อพัฒนาเป็นส่วนประกอบซอฟต์แวร์ต่อไป ซึ่งส่วนประกอบที่ได้ออกมานี้สามารถนำไปใช้ประกอบกับส่วนประกอบซอฟต์แวร์อื่นเพื่อสร้างเป็นซอฟต์แวร์ตัวใหม่ได้ โดยคลาสที่อยู่ในกลุ่มเดียวกันจะมีความสัมพันธ์กัน และทำงานร่วมกันเพื่อบรรลุหน้าที่บางอย่าง ผู้วิจัยเห็นว่าการพัฒนาเชิงกระบวนการก็น่าจะเป็นไปในแนวทางเดียวกันคือน่าจะมีกลุ่มของกิจกรรมหรือแอคชัน (Action) ภายในโดเมนงานที่มีความเกี่ยวข้องกันและสามารถมองเป็นกระบวนการย่อย ๆ ที่ทำหน้าที่บางอย่างได้ การจับกลุ่มของแอคชันเหล่านี้จะเรียกว่าส่วนประกอบโพรเซส (Process Component) [9, 10] จุดมุ่งหมายของส่วนประกอบโพรเซสเป็นไปในทำนองเดียวกับส่วนประกอบซอฟต์แวร์คือ

สามารถนำส่วนประกอบโพรเซสไปประกอบกัน หรือนำไปเป็นส่วนหนึ่งของกระบวนการทางธุรกิจของแอพพลิเคชันตัวใหม่ที่สร้างขึ้นได้ ตัวอย่างเช่น ส่วนประกอบโพรเซสของการจัดการใบสั่งซื้อสามารถนำไปรวมในโพรเซสของธุรกิจต่าง ๆ ที่มีการสั่งซื้อได้ หลักการนี้สอดคล้องกับแนวคิดของโพรเซสแพทเทิร์น (Process Pattern) [11] ซึ่งกล่าวถึงการที่หลายๆ ธุรกิจอาจมีกระบวนการทำงานที่คล้าย ๆ กัน จึงสามารถกำหนดเป็นแพทเทิร์นของกระบวนการทำงานที่สามารถนำไปใช้ซ้ำได้

ส่วนประกอบโพรเซสหนึ่ง ๆ ที่ได้จากการออกแบบสามารถนำไปพัฒนาเป็นซอฟต์แวร์ต่อไปได้ โดยอาจใช้วิธีการพัฒนาซอฟต์แวร์แบบเดิมซึ่งใช้แผนภาพคลาสเป็นหลัก หรือใช้วิธีพัฒนาเชิงกระบวนการก็ได้ หากใช้วิธีการพัฒนาแบบเดิม เราอาจมองกระบวนการทำงานของส่วนประกอบโพรเซสเทียบได้กับเป็นแผนภาพแอคทิวิตี (Activity Diagram) [1, 2, 3] ของส่วนประกอบซอฟต์แวร์ที่จะพัฒนา จากนั้นทำการวิเคราะห์และออกแบบให้ได้แผนภาพคลาส หรือแผนภาพอื่นเพิ่มเติม ก่อนที่จะทำการพัฒนาต่อไป [12] แต่หากใช้วิธีการพัฒนาเชิงกระบวนการ เราสามารถพัฒนาส่วนประกอบโพรเซสโดยใช้เทคโนโลยีเว็บเซอร์วิส (Web services) [13, 14, 15] และการทำออเคสเตรชันด้วยบีเพล [16] ได้

จากการที่ส่วนประกอบโพรเซสมีความคล้ายคลึงกับส่วนประกอบซอฟต์แวร์ ผู้วิจัยจึงเห็นว่าน่าจะสามารถนำวิธีการออกแบบส่วนประกอบซอฟต์แวร์มาประยุกต์ใช้กับการออกแบบส่วนประกอบโพรเซสได้ งานวิจัยนี้จึงได้นำข้อดีของการพัฒนาส่วนประกอบซอฟต์แวร์ที่มีการคำนึงถึงเป้าหมายด้านการจัดการส่วนประกอบ (Component Management Goals) และการวัดข้อมูลทางเทคนิค (Technical Features) [17] มาช่วยในการออกแบบส่วนประกอบโพรเซส โดยเป้าหมายด้านการจัดการส่วนประกอบที่ควรคำนึงถึงนั้นได้แก่ การใช้ต้นทุนอย่างมีประสิทธิผล (Cost Effectiveness) ความง่ายในการประกอบ (Ease of Assembly) ความสามารถในการปรับแต่ง (Customization) ความสามารถในการนำกลับมาใช้ (Reusability) และสภาพบำรุงรักษาได้ (Maintainability) ประเด็นด้านการจัดการเหล่านี้เป็นสิ่งสำคัญเนื่องจากส่วนประกอบโพรเซสนั้นไม่ได้ทำงานอยู่โดดเดี่ยว แต่จะได้รับการพัฒนาขึ้นเพื่อนำไปใช้ต่อในการประกอบเข้าเป็นแอพพลิเคชันที่ใหญ่ขึ้น ดังนั้นนักออกแบบซอฟต์แวร์จึงควรคำนึงถึงประเด็นเหล่านี้ในขณะทำการออกแบบ

งานวิจัย [17] กล่าวถึงการสร้างส่วนประกอบซอฟต์แวร์ (Software Component Fabrication) จากแบบจำลองของโดเมนงานที่อยู่ในรูปของแผนภาพคลาส โดยการจัดแบ่งกลุ่มของคลาสที่ควรอยู่ด้วยกัน เพื่อพัฒนาแต่ละกลุ่มเป็นส่วนประกอบซอฟต์แวร์ จากนั้นทำการวัดข้อมูลลักษณะทางเทคนิคของการออกแบบเพื่อตรวจสอบว่าหากจัดกลุ่มส่วนประกอบซอฟต์แวร์เป็นแบบนี้แล้วทำให้บรรลุเป้าหมายด้านการจัดการส่วนประกอบที่กำหนดไว้เพียงใด

ผู้วิจัยมีแนวคิดที่จะประยุกต์ใช้หลักการของ [17] ในการพิจารณาการออกแบบส่วนประกอบโพรเซสจากแบบจำลองของโดเมนงานที่อยู่ในรูปแบบจำลองกระบวนการทางธุรกิจซึ่งอธิบายโดยแผนภาพแอคทิวิตี โดยหากมีการออกแบบโดยการจัดแบ่งกลุ่มของแอคชันในแผนภาพแอคทิวิตีให้เป็นส่วนประกอบโพรเซสแล้ว ก็น่าจะสามารถใช้วิธีการวัดข้อมูลลักษณะทางเทคนิคของการออกแบบใน [17] มาปรับใช้เพื่อที่จะระบุได้ว่าหากแบ่งกลุ่มส่วนประกอบโพรเซสในลักษณะนี้แล้วจะบรรลุเป้าหมายด้านการจัดการส่วนประกอบที่กำหนดไว้เพียงใด ทั้งนี้เนื่องจากว่าเราสามารถเทียบเคียงแผนภาพคลาสใน [17] กับแผนภาพแอคทิวิตีของกระบวนการทางธุรกิจได้ในแง่องค์ประกอบของแผนภาพ ผู้วิจัยจะนำเสนอวิธีวัดข้อมูลลักษณะทางเทคนิคของการออกแบบส่วนประกอบโพรเซสโดยปรับจาก [17] และทำการพัฒนาเครื่องมือสำหรับวัดข้อมูลดังกล่าว โดยผลจากการวัดจะสามารถใช้เปรียบเทียบการแบ่งกลุ่มส่วนประกอบโพรเซสในหลาย ๆ แบบได้ว่าแบบใดจะบรรลุเป้าหมายด้านการจัดการส่วนประกอบได้ดีกว่ากัน ดังนั้นจะสามารถช่วยนักออกแบบซอฟต์แวร์ในการตัดสินใจออกแบบส่วนประกอบโพรเซสได้อย่างเหมาะสม

1.2 วัตถุประสงค์ของการวิจัย

1.2.1 เพื่อประยุกต์วิธีการออกแบบส่วนประกอบซอฟต์แวร์มาใช้กับการออกแบบส่วนประกอบโพรเซส โดยมีจุดประสงค์เกี่ยวกับเป้าหมายด้านการจัดการส่วนประกอบซึ่งขึ้นกับลักษณะทางเทคนิค

1.2.2 เพื่อพัฒนาเครื่องมือสนับสนุนการประยุกต์แนวคิดดังกล่าว

1.3 ขอบเขตของการวิจัย

1.3.1. ประยุกต์ใช้แนวคิดของการออกแบบส่วนประกอบซอฟต์แวร์โดยมีจุดประสงค์เกี่ยวกับเป้าหมายด้านการจัดการส่วนประกอบซึ่งขึ้นกับลักษณะทางเทคนิค โดยจะได้วิธีการในการวัดการออกแบบส่วนประกอบโพรเซสที่มีความสามารถดังต่อไปนี้

1.3.1.1. สามารถใช้วัดลักษณะทางเทคนิคของการออกแบบส่วนประกอบโพรเซสสำหรับโดเมนงานที่มีการออกแบบโดยใช้แบบจำลองกระบวนการทางธุรกิจที่ออกแบบด้วยแผนภาพแอคทิวิตีได้

1.3.1.2. จะสามารถนำไปเปรียบเทียบกับค่าที่ได้จากการออกแบบอื่น ๆ ที่ขึ้นกับเป้าหมายด้านการจัดการส่วนประกอบที่ต่างกันได้

1.3.2. ข้อจำกัดของขั้นตอนในการวัดมีดังนี้

1.3.2.1. นักออกแบบซอฟต์แวร์สามารถคำนวณค่ามาตรวัดของการออกแบบส่วนประกอบโพรเซสในรูปแผนภาพแอคทิวิตีตามเป้าหมายด้านการจัดการได้

ครั้งละ 1 การออกแบบโดยไม่สามารถหาการออกแบบที่ดีที่สุดได้ นักออกแบบซอฟต์แวร์จะเป็นผู้พิจารณาเลือกการออกแบบที่เหมาะสมเอง

1.3.2.2. นักออกแบบซอฟต์แวร์จะต้องป้อนข้อมูลที่จำเป็นที่ไม่สามารถรวบรวมได้จากแผนภาพ ได้แก่ เมทริกซ์ค่าน้ำหนักของเป้าหมายด้านการจัดการส่วนประกอบ และ เมทริกซ์ค่าความสัมพันธ์ระหว่างเป้าหมายด้านการจัดการส่วนประกอบและข้อมูลทางด้านเทคนิค

1.3.3. เครื่องมือจะมีความสามารถดังนี้

1.3.3.1. สามารถรับข้อมูลแผนภาพแอคทิวิตีที่ทำตามข้อกำหนดของยูเอ็มแอลรุ่น 2.0 ในรูปของแฟ้มข้อมูลเอ็กซ์เอ็มไอรุ่นที่ 2.1

1.3.3.2. สามารถทำการวัดค่ามาตรวัดจากแผนภาพแอคทิวิตีได้

1.3.4. ทดสอบการใช้งานของเครื่องมือที่พัฒนาขึ้น โดยมีรายละเอียดของการทดสอบดังนี้

1.3.4.1. มีการทดสอบกับกรณีศึกษาที่มีกระบวนการทางธุรกิจที่ซับซ้อน

1.3.4.2. มีการทดสอบโดยขึ้นกับเป้าหมายด้านการจัดการที่กำหนดอย่างน้อย 2 สถานการณ์

1.3.5. แสดงการนำส่วนประกอบโพรเซสที่ได้จากตัวอย่างที่ใช้ในการทดสอบไปใช้ในการออกแบบกระบวนการทางธุรกิจอื่นๆ

1.4 ขั้นตอนการวิจัย

1.5.1. สำรวจลักษณะของการออกแบบส่วนประกอบซอฟต์แวร์จากงานวิจัยอื่น

1.5.2. ศึกษาลักษณะของการออกแบบส่วนประกอบซอฟต์แวร์รวมไปถึงการวัดลักษณะทางเทคนิค

1.5.3. ประยุกต์การวัดเข้ากับส่วนประกอบโพรเซส

1.5.4. ศึกษาเครื่องมือสร้างแผนภาพยูเอ็มแอลที่เป็นที่นิยม โดยพิจารณาถึงข้อจำกัดและรูปแบบของแฟ้มข้อมูลที่เครื่องมือนั้นจัดเก็บหรือส่งออก

1.5.5. ศึกษาเทคโนโลยีที่มีความเหมาะสมในการแปลงแผนภาพแพ็กเกจไปเป็นข้อมูลเอ็กซ์เอ็มไอ

1.5.6. พัฒนาเครื่องมือสนับสนุนการวัด

1.5.7. ทดสอบการทำงานของเครื่องมือที่พัฒนาขึ้น

1.5.8. สรุปผลการวิจัยและจัดทำรายงานวิทยานิพนธ์

1.5 ประโยชน์ที่คาดว่าจะได้รับ

จะได้แนวคิดการวัดการออกแบบส่วนประกอบโพรเซส ตามเป้าหมายด้านการจัดการส่วนประกอบ โดยมาตรวัดที่วัดได้จะช่วยบ่งบอกถึงความเหมาะสมของการออกแบบส่วนประกอบโพรเซสในการบรรลุเป้าหมายด้านการจัดการส่วนประกอบ รวมถึงได้เครื่องมือที่สามารถช่วยนักออกแบบซอฟต์แวร์ในการวัดการออกแบบโดยอัตโนมัติ ซึ่งจะทำให้ลดค่าใช้จ่ายและเวลาในการออกแบบส่วนประกอบโพรเซส และยังเป็นแนวทางในการเปรียบเทียบการออกแบบส่วนประกอบโพรเซสได้

1.6 ผลงานตีพิมพ์

ส่วนหนึ่งของวิทยานิพนธ์นี้ได้ตีพิมพ์และนำเสนอในการประชุมวิชาการดังนี้

1. Proceedings 12th International Conference on Computer Science 2006 (ICCS2006), 29-31 March 2006, Vienna, Austria ในบทความเรื่อง Measuring Process Component Design on Achieving Managerial Goals โดยผู้แต่งคือ Eakong Atiptamvaree และ Twittie Senivongse

2. International Journal of Information Technology (IJIT2006), Volume 3 Number 2, 2006 ในบทความเรื่อง A Quantitative Approach to Strategic Design of Component-Based Business Process Models โดยผู้แต่งคือ Eakong Atiptamvaree และ Twittie Senivongse

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

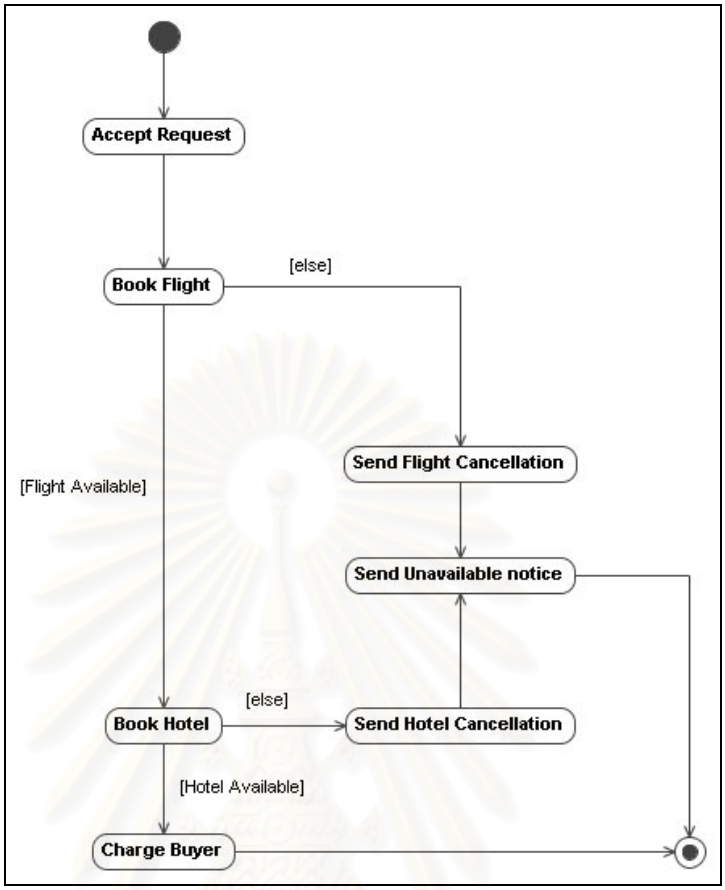
2.1 แนวคิดและทฤษฎี

ทฤษฎีต่าง ๆ ที่เกี่ยวข้องกับการออกแบบส่วนประกอบโพรเซสและการวัดความสามารถด้านการจัดการส่วนประกอบมีดังต่อไปนี้

2.1.1 แบบจำลองกระบวนการทางธุรกิจ (Business Process Model) [18, 19]

แบบจำลองกระบวนการทางธุรกิจเป็นแบบจำลองที่มีเป้าหมายเพื่อให้กลุ่มผู้ใช้ทางธุรกิจหรือนักวิเคราะห์ธุรกิจสามารถออกแบบกระบวนการทำงานของธุรกิจตามโดเมนงานเพื่อให้ผู้พัฒนาซอฟต์แวร์สามารถนำแบบจำลองนั้นไปพัฒนาต่อเป็นซอฟต์แวร์เพื่อรองรับการทำงานของธุรกิจ รวมไปถึงผู้ใช้สามารถตรวจสอบและบริหารกระบวนการทางธุรกิจได้ ดังนั้นแบบจำลองกระบวนการทางธุรกิจนี้จึงสามารถใช้ลดช่องว่างของการออกแบบกระบวนการและการนำกระบวนการไปใช้ต่อได้เป็นอย่างดี

แบบจำลองกระบวนการทางธุรกิจมักแสดงอยู่ในรูปของแผนภาพกระแสงาน ในปัจจุบันมีอยู่หลายมาตรฐาน เช่นแผนภาพแอคทิวิตีของยูเอ็มแอล [1, 2, 3] และแผนภาพบีพีเอ็มเอ็น (BPMN: Business Process Modeling Notation) [20] ในงานวิจัยนี้จะใช้แผนภาพแอคทิวิตีเพื่อแสดงถึงกระบวนการทางธุรกิจดังเช่นรูปที่ 2.1



รูปที่ 2.1 ตัวอย่างกระบวนการทางธุรกิจแสดงโดยแผนภาพแอคทิวิตี

รูปที่ 2.1 แสดงถึงการจองตั๋วเครื่องบินและโรงแรมที่พักซึ่งถ้าจองสำเร็จจะเรียกเก็บค่าใช้จ่ายแต่ถ้าจองไม่สำเร็จจะยกเลิกการจองนั้น

2.1.2 เอ็กซ์เอ็มไอ (XMI: XML Metadata Interchange) [21, 22, 23]

เอ็กซ์เอ็มไอเป็นมาตรฐานที่ออกแบบมาเพื่ออำนวยความสะดวกในการแลกเปลี่ยนเมตาดาตา (Metadata) ระหว่างเครื่องมือที่เกี่ยวข้องกับการทำแบบจำลอง (Modeling) ที่มีพื้นฐานมาจากยูเอ็มแอล กับที่เก็บเมตาดาตา (Metadata Repositories) ที่มีพื้นฐานมาจากเอ็มโอเอฟ (MOF: Meta Object Facility) ในสภาพแวดล้อมแบบกระจายและมีความหลากหลาย โดยเอ็กซ์เอ็มไอเป็นการรวมเอามาตรฐาน 3 มาตรฐานดังต่อไปนี้เข้าไว้ด้วยกัน

1. เอ็กซ์เอ็มแอล (XML: Extensible Markup Language) ซึ่งเป็นมาตรฐานของดับเบิลยูทรีซี (W3C: World Wide Web Consortium)

2. ยูเอ็มแอล (UML: Unified Modeling Language) เป็นมาตรฐานที่กำหนดโดยโอเอ็มจี (OMG: Object Management Group) เพื่อใช้ในการทำแบบจำลองของซอฟต์แวร์

3. เอ็มโอเอฟ (MOF: Meta Object Facility) เป็นมาตรฐานที่กำหนดโดยโอเอ็มจี เพื่อใช้ในการทำเมตาโมเดลและที่เก็บเมตาดาตา

การรวมกันของมาตรฐานทั้งสามนี้ จึงถือเป็นการรวมเอาเทคโนโลยีเกี่ยวกับเมตาดาตาและการทำแบบจำลองของโอเอ็มจีและดับเบิลยูทรีซีเข้าไว้ด้วยกัน ซึ่งทำให้นักพัฒนาสามารถทำการแลกเปลี่ยนแบบจำลองของวัตถุหรือเมตาดาตาชนิดอื่น ๆ ได้สะดวก โดยเฉพาะอย่างยิ่งในการแลกเปลี่ยนกันทางอินเทอร์เน็ต

ในงานวิจัยนี้ จะนำข้อมูลการออกแบบส่วนประกอบโพรเซสตามแผนภาพแอคทิวิตีซึ่งอยู่ในรูปของแฟ้มข้อมูลตามมาตรฐานเอ็กซ์เอ็มไอมาวัดลักษณะทางเทคนิค เพื่อให้ทราบถึงคุณสมบัติของการออกแบบส่วนประกอบโพรเซส ตามเป้าหมายการจัดการส่วนประกอบนั้น

2.2 เอกสารและงานวิจัยที่เกี่ยวข้อง

งานวิจัยที่เกี่ยวข้องและเป็นประโยชน์ต่อการวิจัยนี้ มี 4 งานวิจัย ซึ่งมีรายละเอียดดังนี้

2.2.1การออกแบบเชิงยุทธวิธีเพื่อนำส่วนของโปรแกรมทางธุรกิจกลับมาใช้ซ้ำ (Strategy-Based Design of Reusable Business Components) [17] โดย Padmal Vitharana, Hemant Jain, and Fatemeh “Mariam” Zahedi

ในงานวิจัยนี้ได้นำเสนอระเบียบวิธีการวัดการออกแบบส่วนประกอบซอฟต์แวร์ โดยขึ้นกับเป้าหมายด้านการจัดการและลักษณะทางเทคนิค โดยงานวิจัยนี้ช่วยในการระบุส่วนประกอบซอฟต์แวร์จากแบบจำลองของซอฟต์แวร์ในโดเมนหนึ่ง ๆ ซึ่งเขียนออกแบบอยู่ในรูปของแผนภาพคลาส โดยมีการจัดหมวดหมู่ของกลุ่มคลาสที่ควรอยู่ด้วยกันด้วยแผนภาพแพ็กเกจ แล้ววัดลักษณะทางเทคนิคของแผนภาพคลาสเพื่อมาตัดสินใจว่าคลาสใดควรรวมอยู่ในส่วนประกอบซอฟต์แวร์เดียวกันบ้าง โดยเป้าหมายด้านการจัดการ (Managerial Goals) ประกอบด้วย การใช้ต้นทุนอย่างมีประสิทธิผล (Cost Effectiveness) ความง่ายในการประกอบ (Ease of Assembly) ความสามารถในการปรับแต่ง (Customization) ความสามารถในการนำกลับมาใช้ (Reusability) สภาพบำรุงรักษาได้ (Maintainability) ส่วนลักษณะทางเทคนิค (Technical Features) ที่จะส่งผลต่อเป้าหมายด้านการจัดการข้างต้นได้แก่ การเชื่อมต่อกันระหว่างส่วนประกอบ (Intercomponent Coupling) การเชื่อมติดกันภายในส่วนประกอบ (Intracomponent Cohesion) จำนวนของส่วนประกอบ (Number of Component) ขนาดของส่วนประกอบ (Component Size) และความซับซ้อน (Complexity)

งานวิจัย [17] นี้ได้ทำการสำรวจถึงความสัมพันธ์ระหว่างเป้าหมายด้านธุรกิจและข้อมูลลักษณะทางเทคนิคโดยสร้างแบบสอบถามต่อผู้เชี่ยวชาญทางด้านการพัฒนาแอพพลิเคชันโดย ข้อมูลที่ได้สามารถนำมาปรับแต่งการวัดการออกแบบส่วนประกอบซอฟต์แวร์ได้ นอกจากนี้ได้มีการใช้ซอฟต์แวร์เพื่อการคำนวณสภาพเหมาะสมที่สุด (Optimization Software) ที่ชื่อว่าแกมส์ (GAMS) เพื่อช่วยหาการแบ่งกลุ่มส่วนประกอบซอฟต์แวร์ที่สามารถบรรลุเป้าหมายด้านการจัดการได้ดีที่สุดด้วย

ผู้วิจัยเห็นว่าสามารถนำหลักการใน [17] มาปรับใช้กับการออกแบบส่วนประกอบโพรเซสได้โดยจะให้ความสนใจแบบจำลองรูปนัย (Formal Model) ในการวัดข้อมูลลักษณะทางเทคนิคเพื่อนำมาใช้ในการคิดความสัมพันธ์ระหว่างเป้าหมายด้านการจัดการและลักษณะทางเทคนิคของการออกแบบนั้น

เมื่อศึกษางานวิจัย [17] พบว่ามีข้อดีดังนี้

1. มีการคำนึงถึงเป้าหมายด้านการจัดการในการพัฒนาส่วนประกอบซอฟต์แวร์ ซึ่งเป็นส่วนสำคัญในด้านการพัฒนา
2. สามารถวัดลักษณะทางเทคนิคโดยใช้แบบจำลองรูปนัยเพื่อบอกถึงคุณลักษณะของการออกแบบนั้นว่าตรงตามเป้าหมายด้านการจัดการเพียงใด

งานวิจัยดังกล่าวเป็นการเสนอถึงการระบุว่าคลาสใดควรอยู่ร่วมกันภายในส่วนประกอบซอฟต์แวร์ใด โดยทางผู้วิจัยเห็นว่าน่าจะสามารถนำข้อดีของงานวิจัย [17] นี้มาใช้ในการออกแบบส่วนประกอบโพรเซส ดังนั้นในที่นี้เป็นการระบุว่าแอคชันใดควรจะอยู่ในส่วนประกอบโพรเซสใดนั่นเอง

2.2.2 คุณภาพของการแบ่งส่วนระบบเชิงวัตถุ (Object-oriented System Decomposition Quality) [24] โดย Nejmeddine Tagoug

งานวิจัยนี้กล่าวถึงการแบ่งส่วนระบบเชิงวัตถุภายหลังการออกแบบแผนภาพคลาสโดยรวมของทั้งระบบ และทำการแบ่งระบบออกเป็นระบบย่อยที่มีการเชื่อมโยงระหว่างกัน โดยมีเกณฑ์ในการแบ่งระบบตามเป้าหมายเพื่อเพิ่มการเชื่อมติด (Cohesion) ภายในระบบย่อยที่แบ่งและลดการเชื่อมต่อ (Coupling) ระหว่างระบบย่อยดังนี้

- เกณฑ์ทางข้อมูล (Data Criterion) ทุกคลาสที่มีการใช้ข้อมูลร่วมกันจะถูกจัดกลุ่มให้อยู่ภายในกลุ่มเดียวกัน
- เกณฑ์ทางหน้าที่ทางธุรกิจ (Business Function Criterion) ทุกคลาสที่เกี่ยวข้องกับหน้าที่ทางธุรกิจเดียวกันจะถูกจัดกลุ่มให้อยู่ภายในกลุ่มเดียวกัน

- เกณฑ์ทางเวลา (Time Criterion) ทุกคลาสที่ประมวลผล (Execute) ในช่วงเวลาเดียวกันจะถูกจัดกลุ่มให้อยู่ภายในกลุ่มเดียวกัน
- เกณฑ์ทางโครงสร้างขององค์กร (Organizational Structure Criterion) ทุกคลาสที่ควรทำงานภายใต้หน่วยองค์กรเดียวกันจะถูกจัดกลุ่มให้อยู่ภายในกลุ่มเดียวกัน
- เกณฑ์ทางพฤติกรรม (Behavior Criterion) ทุกคลาสมีพฤติกรรมคล้ายไฟไนต์สเตทแมชีน (Finite State Machine) ถ้าคลาสมีการเปลี่ยนสถานะแล้วมีผลกระทบต่อคลาสอื่นก็จะถูกจัดกลุ่มอยู่ภายในกลุ่มเดียวกัน

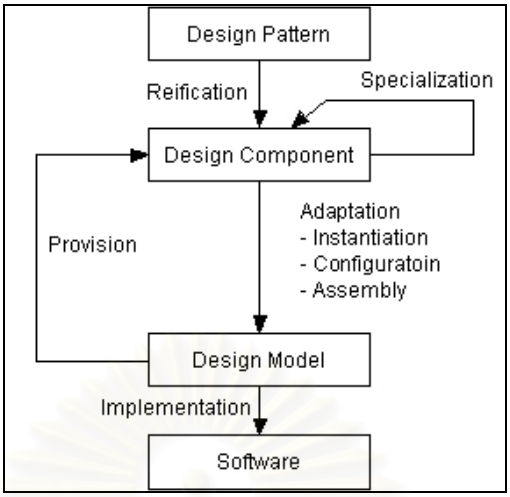
เมื่อกำหนดเกณฑ์ในการแบ่งระบบแล้วจะทำการวัดค่าการเชื่อมต่อและค่าการเชื่อมติดของระบบย่อยที่ได้จากการแบ่ง ซึ่งค่าเหล่านี้จะแสดงถึงคุณภาพของการแบ่งระบบ

งานวิจัยดังกล่าวคล้ายคลึงกับงานวิจัย [17] แต่ในการแบ่งแผนภาพคลาสออกเป็นระบบย่อยไม่ได้คำนึงถึงเป้าหมายด้านการจัดการส่วนประกอบ และลักษณะทางเทคนิคที่สนใจมีเพียงการเชื่อมต่อและการเชื่อมติดเท่านั้น อย่างไรก็ดี ผู้วิจัยสามารถนำเกณฑ์การแบ่งระบบดังกล่าวมาประยุกต์ใช้ในการออกแบบส่วนประกอบโพรเซสโดยจะเป็นแนวทางให้นักออกแบบซอฟต์แวร์ทำการจัดกลุ่มแอคชันในแบบจำลองกระบวนการทางธุรกิจเพื่อนำไปวัดค่าการออกแบบต่อไป

2.2.3 การออกแบบซอฟต์แวร์โดยการประกอบ (A Compositional Approach to Software Design) [25] โดย Rudolf K Keller and Reinhard Schauer

ในงานวิจัยนี้ได้นำเสนอวิธีออกแบบและพัฒนาซอฟต์แวร์โดยใช้ส่วนประกอบซอฟต์แวร์โดยเสนอเป็นวิธีพัฒนาซอฟต์แวร์ตั้งแต่การออกแบบโดยแพทเทิร์นไปจนถึงการพัฒนา

รูปที่ 2.2 แสดงขั้นตอนการพัฒนาซอฟต์แวร์จากการออกแบบโดยแพทเทิร์นแล้วมีการปรับปรุงส่วนประกอบโดยขั้นนี้จะมีการทำซ้ำ และนำมาประกอบพัฒนาเป็นซอฟต์แวร์



รูปที่ 2.2 ขั้นตอนการออกแบบการประกอบซอฟต์แวร์ [25]

งานวิจัยนี้ได้กล่าวถึงสิ่งที่ควรนำมาวัดคุณภาพของการออกแบบโดยมีเกณฑ์ดังนี้

1. ความสามารถในการติดตาม (Traceability)
2. ความสามารถในการนำกลับมาใช้ (Reusability)
3. ความสามารถในการรับรู้และเข้าใจ (Transparency)
4. ความสามารถในการปรับเปลี่ยน (Changeability)
5. ความง่ายในการใช้ (Ease of Use)
6. การมีเครื่องมือสนับสนุน (Tool support)
7. ความสมบูรณ์ (Maturity)
8. ความสามารถในการผลิต (Productivity)

เมื่อศึกษางานวิจัยเกี่ยวกับการออกแบบพัฒนาซอฟต์แวร์ที่มีส่วนประกอบพบว่างานวิจัยนี้มีข้อจำกัดดังนี้

1. ขั้นตอนการออกแบบไม่มีการวัดถึงผลการออกแบบในแต่ละรูปแบบว่ามีข้อแตกต่างกัน
2. การออกแบบไม่ได้ขึ้นกับเป้าหมายด้านการจัดการจึงไม่มีวัตถุประสงค์ในการออกแบบ และพัฒนาส่วนประกอบของซอฟต์แวร์

ข้อจำกัดดังกล่าวของงานวิจัยนี้ ทำให้ยังไม่สามารถนำผลจากการวิจัยนี้ไปประยุกต์ใช้ในการออกแบบส่วนประกอบโพรเซสได้อย่างเหมาะสมนัก

2.2.4 การเสนอรูปแบบและการค้นคืนส่วนประกอบโพรเซสเพื่อนำกลับมาใช้ใหม่ (Reuse-Oriented Process Component Representation and Retrieval) [9] โดย Xu Ru-Zhi, He Tao, Chu Dong-Sheng, Xue Yun-Jiao, Qian Le-Qiu และ แบบจำลองส่วนประกอบโพรเซสเพื่อการนำองค์ความรู้ขององค์กรธุรกิจกลับมาใช้ใหม่ (A Process Component Model for Enterprise Business Knowledge Reuse) [10] โดย Yujie Mou, Jian Cao, Shensheng Zhang

ในงานวิจัย [9] ได้เสนอแนวคิดเพื่อใช้ในการอธิบายและจัดหมวดหมู่ของส่วนประกอบโพรเซสโดยแบ่งข้อมูลเพื่อใช้อธิบายและจัดหมวดหมู่ตามบุคคลที่เกี่ยวข้อง ได้แก่ สำหรับผู้ใช้ จะกำหนดรายละเอียดโดยทั่วไป (General Description) สำหรับวิศวกรโพรเซส จะกำหนดรายละเอียดข้อกำหนดทางคุณลักษณะ (Specification Description) และสำหรับผู้จัดการโครงการซอฟต์แวร์ จะกำหนดรายละเอียดข้อมูล (Data Description) โดยในการนำกลับมาใช้ใหม่นั้น จะมองโพรเซสว่าเป็นองค์ความรู้และประสบการณ์ขององค์กร ซึ่งสามารถนำกลับมาใช้ใหม่ได้ ส่วนงานวิจัย [10] มองโพรเซสว่าเป็นองค์ความรู้และสามารถนำกลับมาใช้ใหม่ได้อีกเช่นกัน แต่มีการอธิบายแบบจำลองของส่วนประกอบโพรเซสโดยออนโทโลจี (Ontology) ซึ่งอธิบายลักษณะต่างๆ ของส่วนประกอบโพรเซส ได้แก่ หน้าที่การทำงาน (Function) กระแสงาน (Workflow) ส่วนต่อประสาน (Interface) คุณภาพการให้บริการ (QoS) และวัฏจักรชีวิต (Life Cycle) ของส่วนประกอบโพรเซส ทั้งสองงานวิจัยดังกล่าวได้เน้นที่การนำกลับมาใช้ใหม่ของส่วนประกอบโพรเซสภายใต้กรอบการทำงาน (Framework) ของตนเอง แต่ไม่ได้กล่าวถึงวิธีการกำหนดหรือออกแบบส่วนประกอบโพรเซสว่ามีที่มาอย่างไร

ผู้วิจัยได้นำนิยามของส่วนประกอบโพรเซสจากงานวิจัยทั้งสองนี้มาประยุกต์ใช้ เนื่องจากมีแนวคิดที่สอดคล้องกันในแง่การนำส่วนประกอบโพรเซสที่มีอยู่กลับมาใช้ใหม่

บทที่ 3
ส่วนประกอบโพรเซสและการบรรลุเป้าหมายด้านการจัดการส่วนประกอบ

แนวทางในการประยุกต์มาตรวัดและแบบจำลองเชิงคุณภาพเพื่อใช้ในการออกแบบส่วนประกอบโพรเซส ได้ประยุกต์มาจากมาตรวัดและแบบจำลองเชิงคุณภาพที่ใช้ในการออกแบบส่วนประกอบซอฟต์แวร์ โดยมีการเทียบเคียงนิยามและปรับค่ามาตรวัดเพื่อความเหมาะสม โดยเริ่มจากการออกแบบส่วนประกอบโพรเซส โดยคำนึงถึงเป้าหมายด้านการจัดการส่วนประกอบ และวัดลักษณะทางเทคนิคของส่วนประกอบโพรเซสที่ออกแบบนั้น ซึ่งท้ายที่สุดจะใช้มาตรวัดเพื่อวัดค่ามาตรวัดก่อนนำค่าที่ได้ไปช่วยในการตัดสินใจพัฒนาส่วนประกอบโพรเซสต่อไป โดยแนวทางในการประยุกต์และออกแบบส่วนประกอบโพรเซสแสดงในรูปที่ 3.1

3.1 การออกแบบส่วนประกอบโพรเซสเพื่อการบรรลุเป้าหมายด้านการจัดการส่วนประกอบ

การออกแบบส่วนประกอบโพรเซสจะอาศัยหลักการของการออกแบบส่วนประกอบซอฟต์แวร์ เนื่องจากส่วนประกอบโพรเซสและส่วนประกอบซอฟต์แวร์มีความคล้ายคลึงกันทั้งในแง่นิยามและในแง่องค์ประกอบของแบบจำลอง ซึ่งการเทียบเคียงนิยามและการเทียบเคียงแบบจำลองเป็นดังนี้

3.1.1 การเทียบเคียงนิยาม

จาก [17] ส่วนประกอบซอฟต์แวร์หมายถึงส่วนของซอฟต์แวร์ที่ทำหน้าที่ทางธุรกิจ และมีการนำไปประกอบเข้ากับส่วนประกอบอื่นเพื่อให้ทำงานเป็นแอพพลิเคชัน โดยไม่อ้างอิงถึงเทคโนโลยีใดเทคโนโลยีหนึ่งในการพัฒนาส่วนประกอบนั้น ส่วนประกอบซอฟต์แวร์มีพื้นฐานอยู่บนการออกแบบเชิงวัตถุซึ่งแบบจำลองมักอยู่ในรูปของแผนภาพคลาสของยูเอ็มแอล ดังนั้นเราสามารถออกแบบส่วนประกอบซอฟต์แวร์สำหรับโดเมนงานหนึ่ง ๆ ด้วยแผนภาพคลาส และทำการแบ่งกลุ่มของคลาสซึ่งทำงานร่วมกันในการให้บริการอย่างใดอย่างหนึ่งเข้าไว้ด้วยกัน แล้วนำแต่ละกลุ่มไปพัฒนาต่อเป็นส่วนประกอบซอฟต์แวร์เพื่อใช้ซ้ำในแอพพลิเคชันอื่นได้

สำหรับส่วนประกอบโพรเซสนั้น มีลักษณะและจุดประสงค์การใช้งานที่เทียบเคียงได้กับส่วนประกอบซอฟต์แวร์ โดยจาก [9, 10] ส่วนประกอบโพรเซสหมายถึงส่วนของซอฟต์แวร์ที่มีความสามารถในการทำงานบางอย่าง โดยมีการห่อหุ้ม (Encapsulation) กระบวนการภายใน และถูกพัฒนาขึ้นเพื่อใช้ประกอบกันเป็นแอพพลิเคชันใหม่โดยไม่ผูกติดกับเทคโนโลยีในการพัฒนาแบบใดแบบหนึ่งโดยเฉพาะเช่นกัน แต่ส่วนประกอบโพรเซสมีพื้นฐานอยู่บนการออกแบบเชิงกระบวนการ ดังนั้นผู้วิจัยสามารถออกแบบส่วนประกอบโพรเซสสำหรับโดเมนงานหนึ่ง ๆ ด้วยแบบจำลองกระบวนการทางธุรกิจ (เช่น แผนภาพแอคทิวิตี) และทำการแบ่งกลุ่มงาน (Task) ในกระบวนการทางธุรกิจซึ่งให้บริการอย่างใดอย่างหนึ่งร่วมกันเข้าไว้ด้วยกัน และนำแต่ละกลุ่มไปใช้

รูปที่ 3.1แนวทางในการประยุกต์และออกแบบส่วนประกอบโพรเซส

แสดงซ้ำในแบบจำลองกระบวนการของแอพพลิเคชันอื่น หรือนำไปพัฒนาเป็นซอฟต์แวร์เพื่อใช้ซ้ำในการพัฒนาแอพพลิเคชันอื่นได้

3.1.2 การเทียบเคียงแบบจำลอง

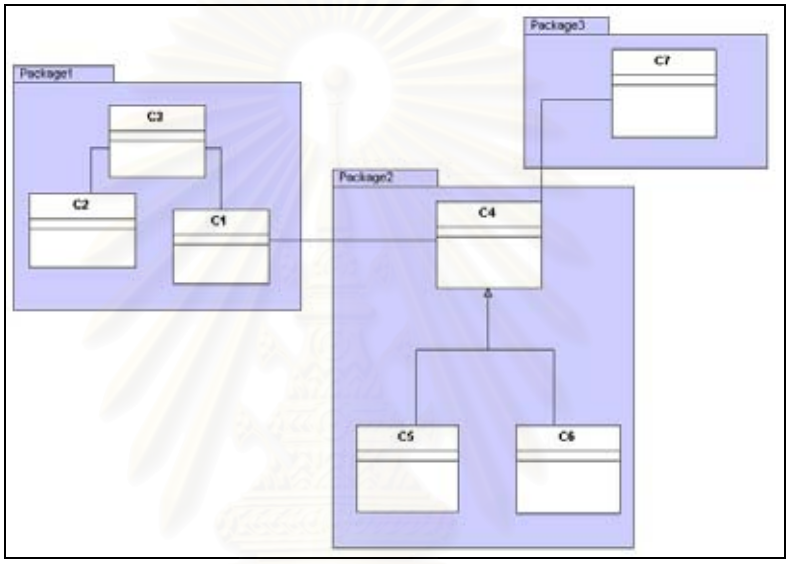
จากงานวิจัย [17] การวัดข้อมูลลักษณะทางเทคนิคของการออกแบบตามเป้าหมายทางด้านการจัดการส่วนประกอบ มีอินพุทเป็นแบบจำลองของโดเมนงานที่อยู่ในรูปของแผนภาพคลาส แล้วแบ่งการจัดกลุ่มของคลาสที่ควรอยู่ด้วยกันเป็นส่วนประกอบซอฟต์แวร์เดียวกันด้วยแผนภาพแพ็กเกจ โดยค่าที่ได้จากการวัดจะใช้บอกได้ว่าการแบ่งกลุ่มของคลาสนั้นเทียบกับการแบ่งกลุ่มของคลาสอีกแบบจะบรรลุเป้าหมายด้านการจัดการส่วนประกอบที่ตั้งไว้แตกต่างกันอย่างไร จากการเทียบเคียงนิยามข้างต้นและการพิจารณาแบบจำลองของโดเมนงานในรูปแผนภาพคลาสใน [17] กับแผนภาพแอคทิวิตีซึ่งแสดงกระบวนการทางธุรกิจตามงานวิจัยนี้แล้ว ผู้วิจัยพบว่าแบบจำลองทั้งสองมีความคล้ายคลึงกันในเชิงองค์ประกอบของแผนภาพและสามารถเทียบเคียงกันได้ ดังนี้

รูปที่ 3.2 แสดงตัวอย่างการออกแบบส่วนประกอบซอฟต์แวร์ตามแนวทางของ [17] โดยเป็นการจัดกลุ่มของคลาสในแผนภาพคลาสของโดเมนงานออกเป็น 3 ส่วนประกอบซอฟต์แวร์ (หรือแพ็กเกจ) คลาสภายในแต่ละส่วนประกอบซอฟต์แวร์จะมีความสัมพันธ์กันโดยแสดงด้วยเส้นเชื่อมระหว่างคลาส อันได้แก่เส้นการสืบทอดคุณลักษณะ (Generalization) หรือเส้นแอสโซสิเอชัน (Association) โดยเส้นแอสโซสิเอชันนั้นแสดงความสัมพันธ์ในหลายลักษณะ เช่น การที่คลาสหนึ่งมีการเรียกใช้เมท็อดของอีกคลาสหนึ่ง การที่คลาสทั้งสองใช้ข้อมูลร่วมกัน หรือ การที่คลาสหนึ่งมีข้อมูลที่มีชนิดเป็นอีกคลาสหนึ่ง ลักษณะความสัมพันธ์เหล่านี้สามารถมองเป็นความสัมพันธ์ดังในหัวข้อที่ 2.2.2 กล่าวคือเป็นความสัมพันธ์ในแง่ข้อมูล (Data Criterion) เนื่องจากคลาสสองคลาสเกี่ยวข้องกันทางข้อมูล หรืออาจมองเป็นความสัมพันธ์ในแง่เวลา (Time Criterion) เนื่องจากคลาสสองคลาสที่มีการเรียกใช้เมท็อดระหว่างกันจะอยู่ในเวลาเดียวกันขณะที่มีการทำงาน นอกจากนี้แต่ละส่วนประกอบซอฟต์แวร์ก็มีความสัมพันธ์ระหว่างกัน โดยแสดงได้ด้วยเส้นแอสโซสิเอชันที่เชื่อมระหว่างคลาสที่อยู่ภายใต้คนละส่วนประกอบซอฟต์แวร์

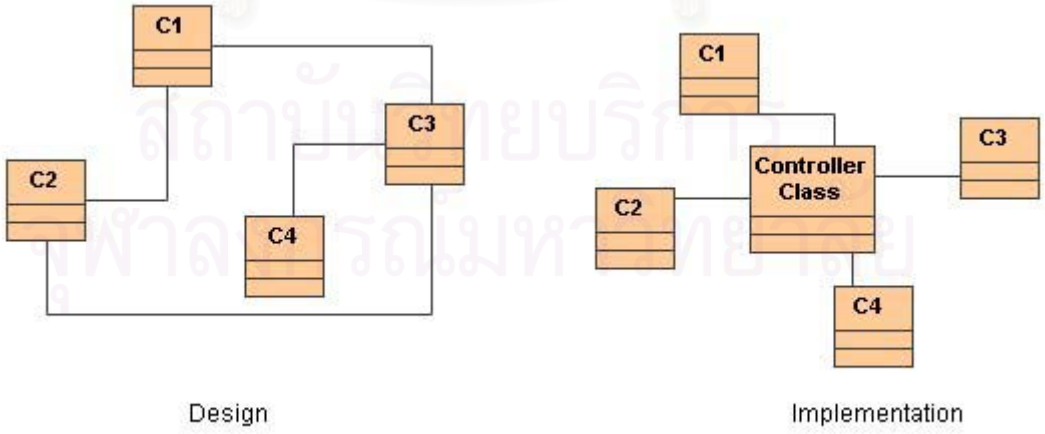
อย่างไรก็ตามแผนภาพคลาสนั้นเป็นเพียงแบบจำลองตามการออกแบบที่วางไว้เท่านั้น ในขั้นตอนการพัฒนาซอฟต์แวร์ นักพัฒนาอาจจะพัฒนาแตกต่างไปได้ ตัวอย่างเช่น คลาสสองคลาสที่มีเส้นแอสโซสิเอชันระหว่างกันตามแผนภาพคลาสนั้น ในขั้นตอนการพัฒนาอาจจะไม่ส่งข้อมูลถึงกันโดยตรงหรือเรียกใช้เมท็อดระหว่างกันโดยตรงก็ได้ หากนักพัฒนาทำการพัฒนาโดยใช้คลาสควบคุม (Controller Class) ดังรูปที่ 3.3 ในการกำกับการเรียกเมท็อดหรือส่งข้อมูลระหว่างคลาส คลาสทั้งสองจึงไม่ได้มีความสัมพันธ์กันโดยตรงแล้วเมื่อพัฒนา แต่จะมีความสัมพันธ์โดยอ้อมผ่านคลาสควบคุม แม้กระนั้นก็ตาม งานวิจัย [17] เน้นที่ขั้นตอนการออกแบบ จึงสนใจแต่เพียง

รายละเอียดที่ปรากฏอยู่ในแบบจำลองในรูปของแผนภาพคลาส และถือว่าเส้นเชื่อมระหว่างคลาสจะบ่งบอกว่าคลาสมีความสัมพันธ์กัน โดยไม่สนใจว่าจะเป็นโดยทางตรงหรือทางอ้อม

การวัดข้อมูลลักษณะทางเทคนิคจากแผนภาพคลาสเพื่อพิจารณาว่าการออกแบบส่วนประกอบซอฟต์แวร์หนึ่ง ๆ เหมาะสมตามเป้าหมายด้านการจัดการส่วนประกอบที่ตั้งไว้หรือไม่นั้น มีพื้นฐานอยู่บนการวัดระดับความสัมพันธ์ระหว่างคลาส จำนวนกลุ่มที่แบ่งและรายละเอียดของคลาสภายในกลุ่มที่แบ่ง ดังจะได้กล่าวต่อไป



รูปที่ 3.2 การจัดกลุ่มของแผนภาพคลาสโดยแบ่งออกเป็นแพ็กเกจ

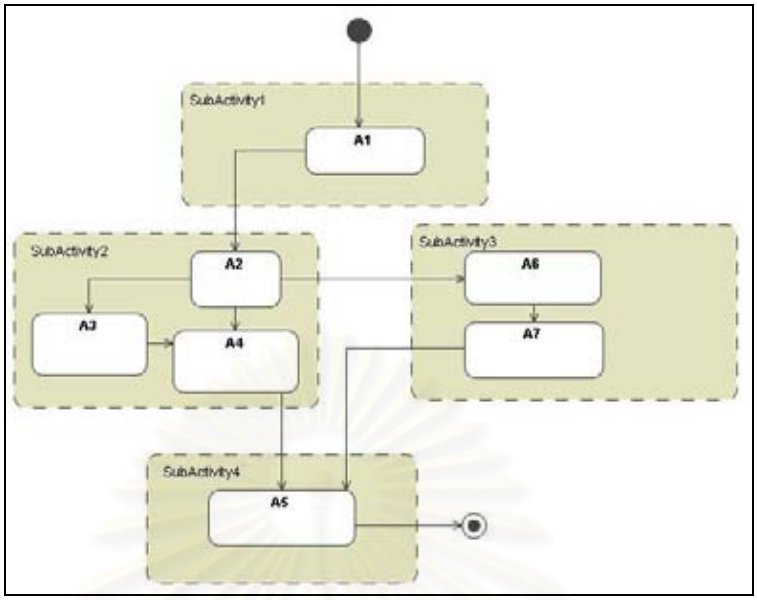


รูปที่ 3.3 ความสัมพันธ์โดยตรงและโดยอ้อมระหว่างคลาส

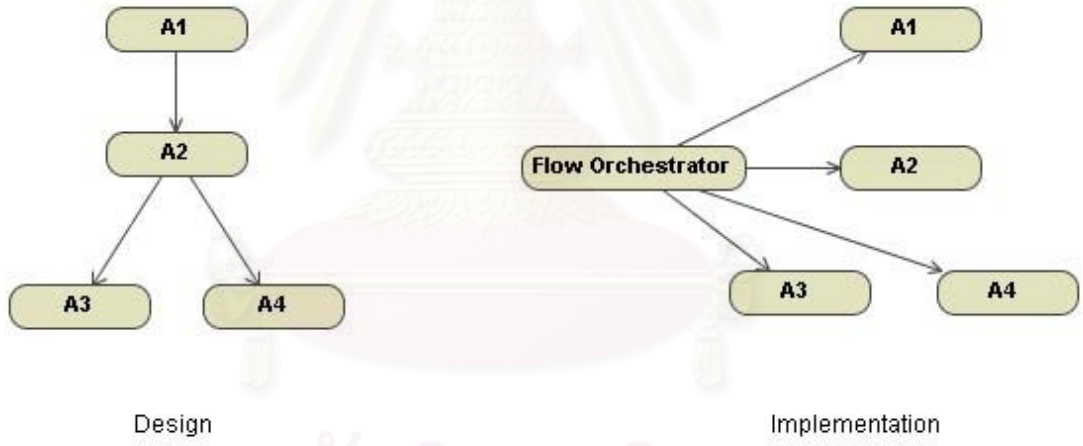
ในทำนองเดียวกัน เมื่อพิจารณาแผนภาพแอคทิวิตีซึ่งแสดงกระบวนการทางธุรกิจของโดเมนงานดังตัวอย่างในรูปที่ 3.4 แล้ว จะเห็นว่าสามารถเทียบเคียงได้กับแผนภาพคลาสในรูปที่ 3.2 ในเชิงองค์ประกอบของแผนภาพ กล่าวคือ เราสามารถจำลองการทำงานของโดเมนงานในลักษณะของกระแสงานของแอคชันต่าง ๆ ซึ่งแอคชันเหล่านี้มีความสัมพันธ์กันแสดงได้ด้วยลูกศรสายงานควบคุมที่เชื่อมระหว่างแอคชัน ลูกศรเหล่านี้แสดงความสัมพันธ์ในแง่ข้อมูล (Data Criterion) กล่าวคือแอคชันหนึ่งใช้ข้อมูลที่ส่งมาจากอีกแอคชันหนึ่งในการทำงาน หรือแสดงความสัมพันธ์ในแง่เวลา (Time Criterion) กล่าวคือแอคชันหนึ่งอยู่ร่วมกับอีกแอคชันหนึ่งในเวลาเดียวกันขณะทำงาน (ดังในหัวข้อที่ 2.2.2) ดังนั้นจึงเทียบเคียงได้กับการที่คลาสสองคลาสในรูปที่ 3.2 มีเส้นเชื่อมแสดงความสัมพันธ์ระหว่างกัน

สำหรับแผนภาพแอคทิวิตีก็เช่นเดียวกับแผนภาพคลาส คือเป็นเพียงแบบจำลองตามการออกแบบที่วางไว้เท่านั้น ในขั้นตอนการนำแบบจำลองไปพัฒนาหรือแปลงเป็นภาษาประมวลผลกระบวนการ (Process Execution Language) เพื่อให้ประมวลผลได้ด้วยเครื่องประมวลผลกระบวนการ (Process Execution Engine) นั้น การประมวลผลตามกระแสงานจากแอคชันหนึ่งไปยังอีกแอคชันหนึ่งก็ทำโดยอาศัยตัวควบคุมกระแสงาน (Flow Orchestrator) ในทำนองเดียวกับกรณีของคลาสควบคุมดังข้างต้น ดังนั้นการส่งผ่านข้อมูลหรือการทำงานในเวลาต่อเนื่องกันนั้นจะไม่ใช่ความสัมพันธ์โดยตรงระหว่างสองแอคชัน แต่เป็นความสัมพันธ์โดยอ้อมผ่านตัวควบคุมกระแสงาน (รูปที่ 3.5) แม้กระนั้นก็ตาม งานวิจัยนี้เน้นที่ขั้นตอนการออกแบบเช่นเดียวกับ [17] จึงสนใจแต่เพียงรายละเอียดที่ปรากฏอยู่ในแบบจำลองในรูปของแผนภาพแอคทิวิตี และถือว่าเส้นลูกศรที่เชื่อมระหว่างแอคชันจะบ่งบอกว่าแอคชันมีความสัมพันธ์กัน โดยไม่สนใจว่าจะเป็นโดยทางตรงหรือทางอ้อม

ในลักษณะเช่นนี้ หากทำการแบ่งกลุ่มของแอคชันในแผนภาพแอคทิวิตีออกเป็นส่วนประกอบโพรเซส (หรือแอคทิวิตีย่อย) ซึ่งในรูปที่ 3.4 แสดง 4 ส่วนประกอบโพรเซส แอคชันต่าง ๆ ในส่วนประกอบโพรเซสเดียวกันจะมีความสัมพันธ์ระหว่างกัน รวมทั้งแต่ละส่วนประกอบโพรเซสก็มีความสัมพันธ์ระหว่างกันด้วยผ่านเส้นลูกศรที่เชื่อมระหว่างแอคชันที่อยู่ภายใต้คนละส่วนประกอบโพรเซส ผู้วิจัยจึงเห็นว่าเราสามารถประยุกต์ใช้การวัดข้อมูลลักษณะทางเทคนิคจากแผนภาพคลาสใน [17] เพื่อพิจารณาว่าการออกแบบส่วนประกอบโพรเซสหนึ่ง ๆ เหมาะสมตามเป้าหมายด้านการจัดการส่วนประกอบที่ตั้งไว้หรือไม่ ซึ่งการพิจารณาดังกล่าวมีพื้นฐานอยู่บนการวัดระดับความสัมพันธ์ระหว่างแอคชัน จำนวนกลุ่มที่แบ่ง และรายละเอียดของแอคชันภายในกลุ่มที่แบ่ง ในทำนองเดียวกัน



รูปที่ 3.4 การจัดกลุ่มของแผนภาพแอคทิวิตีโดยแบ่งออกเป็นแอคทีวิตีย่อย



รูปที่ 3.5 ความสัมพันธ์โดยตรงและโดยอ้อมระหว่างแอคชัน

ในการเทียบเคียงแผนภาพนี้มีข้อสังเกตคือ จากความหมายของแอคชันในแผนภาพแอคทิวิตีของยูเอ็มแอล จะหมายถึงหน่วยงานย่อยที่ไม่สามารถแบ่งย่อยต่อไปได้อีก แต่ในความเป็นจริงแล้ว หน่วยงานย่อยภายในกระบวนการทางธุรกิจมักจะยังเป็นหน่วยงานย่อยที่อยู่ในระดับบนและสามารถแบ่งย่อยต่อไปได้ ดังนั้นการที่ผู้วิจัยใช้แอคชันแทนหน่วยงานย่อยในกระบวนการทางธุรกิจจึงอาจไม่เหมาะสม อย่างไรก็ตามผู้วิจัยจะไม่สนใจว่าหน่วยงานย่อยในกระบวนการทางธุรกิจนั้นจะสามารถแบ่งย่อยออกได้เป็นอีกกี่งาน จึงใช้แอคชันแทนหน่วยงานย่อยเลยเพื่อให้สอดคล้อง

ในการเทียบเคียงกับคลาสในแผนภาพคลาสของส่วนประกอบซอฟต์แวร์ซึ่งแบ่งย่อยไม่ได้อีก แล้วจึงแทนกลุ่มของหน่วยงานย่อย (กลุ่มของแอคชัน) ที่ประกอบกันเป็นส่วนประกอบโพรเซสด้วยแอคทิวิตีย่อยแทน

โดยสรุปแล้ว เราสามารถเทียบเคียงแผนภาพคลาสใน [17] กับแผนภาพแอคทิวิตีที่จะใช้ในงานวิจัยนี้ได้ดังตารางที่ 3.1

ตารางที่ 3.1 การเทียบเคียงแผนภาพคลาสใน [17] กับแผนภาพแอคทิวิตีของงานวิจัยในเชิงองค์ประกอบของแผนภาพ

องค์ประกอบของแผนภาพคลาส	**องค์ประกอบของแผนภาพแอคทีวิตี**
คลาส	แอคชัน
เส้นเชื่อมระหว่างคลาส (การสืบทอด คุณลักษณะหรือแอสโซสิเอชัน)	ลูกศรเชื่อมระหว่างแอคชัน (สายงานควบคุม)
แพ็กเกจ	แอคทิวิตีย่อย

เมื่อเทียบเคียงแผนภาพแล้ว ผู้วิจัยจะเสนอแนวทางในการออกแบบส่วนประกอบโพรเซสโดยทำการประยุกต์จากงานวิจัย [17] โดยมีอินพุทเป็นแบบจำลองกระบวนการทางธุรกิจซึ่งอยู่ในรูปแผนภาพแอคทิวิตี จากนั้นทำการแบ่งกลุ่มของแอคชันออกเป็นแอคทิวิตีย่อย โดยใช้เกณฑ์ตามหัวข้อที่ 2.2.2 [24] เป็นแนวทาง แล้วนำการออกแบบส่วนประกอบโพรเซสที่ได้มาวัดข้อมูลลักษณะทางเทคนิคเพื่อพิจารณาว่าการออกแบบสามารถบรรลุเป้าหมายการจัดการส่วนประกอบที่ได้กำหนดไว้เพียงใด

3.2 เป้าหมายด้านการจัดการส่วนประกอบ

นักออกแบบส่วนประกอบโพรเซสสามารถเลือกเป้าหมายในการออกแบบได้หลายเป้าหมายในลักษณะเดียวกันกับที่นักออกแบบส่วนประกอบซอฟต์แวร์สามารถทำได้ตาม [17, 26] เป้าหมายด้านการจัดการเหล่านี้ได้แก่

1. การใช้ต้นทุนอย่างมีประสิทธิผล (Cost Effectiveness)

เป้าหมายนี้เกี่ยวข้องกับความต้องการต้นทุนในการผลิตส่วนประกอบโพรเซสที่ต่ำ โดยต้องการให้ค่าใช้จ่ายในการพัฒนาต่ำที่สุด และเวลาในการออกแบบและพัฒนาส่วนประกอบลดน้อยที่สุด

2. ความง่ายในการประกอบ (Ease of Assembly)

เป้าหมายนี้เกี่ยวข้องกับความต้องการให้ส่วนประกอบโพรเซสที่ได้สามารถนำไปประกอบกันได้ง่าย โดยการมีส่วนประกอบโพรเซสจำนวนไม่มากนักและแต่ละส่วนประกอบ

สามารถทำงานชิ้นใหญ่ได้จะช่วยให้การประกอบง่ายขึ้น เพราะจำนวนส่วนต่อประสาน (Interface) ที่ต้องพิจารณาเมื่อนำไปประกอบจะลดลง

3. ความสามารถในการปรับแต่ง (Customization)

เป้าหมายนี้เกี่ยวข้องกับความต้องการให้ส่วนประกอบโพรเซสที่ได้สามารถนำไปปรับแต่งโดยผู้ใช้แต่ละรายให้เหมาะสมกับลักษณะงานหรือธุรกิจต่าง ๆ ได้

4. ความสามารถในการนำกลับมาใช้ (Reusability)

เป้าหมายนี้เกี่ยวข้องกับความต้องการให้ส่วนประกอบโพรเซสที่ได้สามารถนำไปใช้ซ้ำได้ในหลาย ๆ ลักษณะงานหรือธุรกิจ โดยไม่ต้องปรับแต่ง

5. สภาพบำรุงรักษาได้ (Maintainability)

เป้าหมายนี้เกี่ยวข้องกับความต้องการให้ส่วนประกอบโพรเซสที่ได้สามารถปรับเปลี่ยนได้ หรือทำการเพิ่มหรือถอดส่วนประกอบโพรเซสได้ง่าย เป้าหมายนี้เกี่ยวเนื่องกับความต้องการต้นทุนที่ต่ำด้วย เพราะการบำรุงรักษาถือเป็นต้นทุนระยะยาวอย่างหนึ่ง

3.3 ลักษณะทางเทคนิคของส่วนประกอบโพรเซส

ลักษณะทางเทคนิคที่มีผลต่อเป้าหมายด้านการจัดการส่วนประกอบในการพัฒนาส่วนประกอบซอฟต์แวร์ สามารถประยุกต์ใช้กับการออกแบบส่วนประกอบโพรเซสได้โดย [17, 26] แบ่งลักษณะทางเทคนิคไว้ดังนี้

1. การเชื่อมต่อกันระหว่างส่วนประกอบ (Intercomponent Coupling)

การเชื่อมต่อกันระหว่างส่วนประกอบซอฟต์แวร์คือ ความสัมพันธ์กันระหว่างคลาสที่อยู่คนละส่วนประกอบกัน เราสามารถแบ่งความสัมพันธ์ระหว่างคลาสโดยขึ้นกับการเรียกใช้คลาสในหลาย ๆ ลักษณะ ได้แก่ การเรียกใช้เมท็อดระหว่างคลาส การเรียกใช้แอททริบิวท์ระหว่างคลาส การเรียกใช้คลาสเป็นพารามิเตอร์อินพุทหรือเอาท์พุทของเมท็อดระหว่างคลาส และการเรียกใช้คลาสเป็นตัวแปรโลคัล ซึ่งความสัมพันธ์เหล่านี้เป็นความสัมพันธ์ระหว่างคลาสในแง่ข้อมูลหรือในแง่เวลา

จากที่ได้ทำการเทียบเคียงแผนภาพคลาสและแผนภาพแอคทิวิตีไว้ในหัวข้อที่ 3.1.1 การพิจารณาการเชื่อมต่อกันระหว่างส่วนประกอบโพรเซสจะพิจารณาความสัมพันธ์ระหว่างแอคชันที่อยู่ภายใต้คนละส่วนประกอบกัน และความสัมพันธ์ระหว่างแอคชันจะพิจารณาจากเส้นลูกศรที่แสดงการส่งผ่านข้อมูลหรือส่งผ่านการควบคุม ซึ่งแสดงความสัมพันธ์ระหว่างแอคชันในแง่ข้อมูลหรือในแง่เวลาเช่นกัน

2. การเชื่อมติดกันภายในส่วนประกอบ (Intracomponent Cohesion)

ส่วนประกอบซอฟต์แวร์ที่มีความสัมพันธ์กันภายในสูงเป็นสิ่งที่ต้องการ เนื่องจากภายในส่วนประกอบนั้นจะสามารถทำงานได้ด้วยตัวเองเป็นหลัก โดยไม่ต้องพึ่งส่วนประกอบซอฟต์แวร์อื่นมากนัก เช่น มีการใช้ข้อมูลอย่างใดอย่างหนึ่ง หรือ ทำงานบางอย่างในเวลาใกล้เคียงกัน การพิจารณาการเชื่อมติดกันภายในส่วนประกอบซอฟต์แวร์จึงมีพื้นฐานอยู่บนการวัดระดับความสัมพันธ์ระหว่างคลาสในทำนองเดียวกับการพิจารณาการเชื่อมต่อกันระหว่างส่วนประกอบซอฟต์แวร์ แต่จะเป็นความสัมพันธ์ระหว่างคลาสที่อยู่ภายใต้ส่วนประกอบซอฟต์แวร์เดียวกัน

ในทำนองเดียวกัน ผู้วิจัยได้กำหนดการวัดการเชื่อมติดกันภายในส่วนประกอบโพรเซสโดยวัดจากความสัมพันธ์ระหว่างแอคชันภายในแอคทิวิตีย่อย

3. จำนวนของส่วนประกอบ (Number of Component)

ในการออกแบบส่วนประกอบซอฟต์แวร์ จำนวนส่วนประกอบซอฟต์แวร์จะหมายถึงจำนวนแพ็กเกจที่ได้จากการแบ่งกลุ่มคลาส

ในทำนองเดียวกัน ผู้วิจัยได้กำหนดจำนวนของส่วนประกอบโพรเซสโดยวัดได้จากจำนวนแอคทิวิตีย่อยที่ได้จากการแบ่งกลุ่มแอคชันในแผนภาพแอคทิวิตี

4. ขนาดของส่วนประกอบ (Component Size)

ในการออกแบบส่วนประกอบซอฟต์แวร์ ขนาดของส่วนประกอบซอฟต์แวร์มีพื้นฐานอยู่บนจำนวนของคลาสภายในส่วนประกอบซอฟต์แวร์

ในทำนองเดียวกัน ผู้วิจัยได้กำหนดขนาดของส่วนประกอบโพรเซสโดยมีพื้นฐานจากการวัดจำนวนของแอคชันภายในแอคทิวิตีย่อย

5. ความซับซ้อนของการออกแบบ (Complexity)

ความซับซ้อนของส่วนประกอบซอฟต์แวร์มีพื้นฐานอยู่บนรายละเอียดของส่วนประกอบซอฟต์แวร์ เช่น จำนวนคลาสที่ประกอบภายในส่วนประกอบซอฟต์แวร์

ในทำนองเดียวกัน การวัดความซับซ้อนของการออกแบบส่วนประกอบโพรเซสสามารถพิจารณาได้จากรายละเอียดของส่วนประกอบโพรเซส ได้แก่ จำนวนของแอคชันภายในแอคทิวิตีย่อย

3.4 การวัดลักษณะทางเทคนิคและการบรรลุเป้าหมายด้านการจัดการส่วนประกอบ

จากแบบจำลองกระบวนการทางธุรกิจที่อยู่ในรูปแผนภาพแอคทิวิตี เมื่อนักออกแบบซอฟต์แวร์ทำการออกแบบส่วนประกอบโพรเซสโดยจัดกลุ่มของแอคชันตามโครงสร้างกลุ่มแอคทิวิตีย่อยที่แสดงด้วยแผนภาพแอคทิวิตีได้แล้ว จะสามารถใช้มาตรวัด (Metrics) วัดลักษณะทาง

เทคนิคเพื่อนำไปใช้ในการเปรียบเทียบการออกแบบว่าแบบใดเหมาะสมกว่าในการบรรลุเป้าหมายทางการจัดการส่วนประกอบ โดยนักออกแบบซอฟต์แวร์ต้องกำหนดเป้าหมายทางการจัดการที่ต้องการก่อน โดยกำหนดสัดส่วนความสำคัญของเป้าหมายด้านการจัดการต่าง ๆ ก่อนจะพิจารณาการออกแบบจากมาตรวัดที่ได้กำหนด

งานวิจัย [17] ได้แสดงความสัมพันธ์ระหว่างเป้าหมายด้านการจัดการและลักษณะทางเทคนิคไว้ดังตารางที่ 3.2 โดยค่า 0 หมายถึงลักษณะทางเทคนิคไม่มีผลต่อเป้าหมายด้านการจัดการ เช่น การเชื่อมต่อกันระหว่างส่วนประกอบไม่มีผลต่อการใช้ต้นทุนอย่างมีประสิทธิผล ค่า – หมายถึงลักษณะทางเทคนิคมีผลในเชิงลบต่อเป้าหมายด้านการจัดการส่วนประกอบ เช่น หากค่าการเชื่อมต่อกันระหว่างส่วนประกอบมีค่ามากจะส่งผลให้การประกอบทำได้ยาก และค่า + หมายถึงลักษณะทางเทคนิคมีผลในเชิงบวกต่อเป้าหมายด้านการจัดการ เช่นหากขนาดของส่วนประกอบมีค่ามากก็จะส่งผลให้การประกอบทำได้ง่าย

ตารางที่ 3.2 ความสัมพันธ์ระหว่างเป้าหมายด้านการจัดการส่วนประกอบและลักษณะทางเทคนิค

	COUPL	**COHES**	**NCOMP**	**CSIZE**	**COMPL**
COST	0	-	-	-	-
ASBL	-	0	0	+	0
CUST	-	+	+	-	0
REUS	-	+	+	-	0
MNTN	-	-	-	-	-

Legend

:

(-) negative impact (+) positive impact (0) no impact

Managerial Goals (Rows)	**Technical Features (Columns)**
COST : Cost effectiveness	COUPL : Coupling
ASBL : Ease of assembly	COHES : Cohesion
CUST : Customization	NCOMP : Number of components
REUS : Reusability	CSIZE : Component size
MNTN : Maintainability	COMPL : Complexity

การประยุกต์ใช้มาตรวัดจาก [17] เพื่อใช้วัดลักษณะทางเทคนิคของส่วนประกอบโพรเซสเป็นดังนี้

3.4.1 การเชื่อมต่อกันระหว่างส่วนประกอบโพรเซส

งานวิจัย [17] เสนอมาตรวัดการเชื่อมต่อกันระหว่างส่วนประกอบซอฟต์แวร์ดังนี้

$$COUPL = \sum_{k=1}^{y} \sum_{i=1}^{n} \sum_{\substack{j=1 \\ i \neq j}}^{n} (x_{ik} * (1 - x_{jk}) * c_{ij})$$

โดยกำหนด

y จำนวนของส่วนประกอบซอฟต์แวร์ในโดเมนของการออกแบบ

n จำนวนรวมของคลาสในโดเมนการออกแบบ

x_{ik} มีค่าเท่ากับ 1 ถ้าคลาส i อยู่ภายในส่วนประกอบซอฟต์แวร์ k

มีค่าเท่ากับ 0 ถ้าคลาส i ไม่ได้อยู่ภายในส่วนประกอบซอฟต์แวร์ k

C_{ij} คือค่าการเชื่อมต่อระหว่างคลาส i และคลาส j

สำหรับค่าการเชื่อมต่อ C_{ij} นั้น งานวิจัย [17] ไม่ได้เจาะจงว่าจะต้องวัดอย่างไร แต่ได้กล่าวถึงวิธีการวัดตามทฤษฎีเชิงวัตถุไว้หลายวิธี เช่น งานวิจัย [27] กล่าวว่าสามารถวัดการเชื่อมต่อได้สองแบบ แบบที่หนึ่งคือวัดจากจำนวนการเรียกใช้เมท็อดที่คลาสหนึ่งเรียกไปยังอีกคลาสหนึ่ง รวมทั้งจำนวนพารามิเตอร์และจำนวนแอททริบิวท์ของคลาสที่มีชนิดเป็นคลาสอื่น แบบที่สองเป็นการวัดคร่าว ๆ จากจำนวนของแอสโซสิเอชัน (จำนวนเส้นเชื่อม) ระหว่างคลาสเท่านั้น แม้กระนั้นก็ตาม จำนวนของแอสโซสิเอชันที่วัดได้ตามแบบที่สองจะมีสหสัมพันธ์ (Correlation) กับค่าที่วัดได้ตามแบบที่หนึ่ง [28] อย่างไรก็ตาม ผู้วิจัยเห็นว่าการวัดการเชื่อมต่อจากแผนภาพคลาสนั้นไม่สามารถทำได้ละเอียดเท่าการวัดจากโค้ดตามทฤษฎีเชิงวัตถุ ตัวอย่างเช่น เราไม่สามารถทราบได้จากแผนภาพคลาสว่าคลาสหนึ่งเรียกใช้เมท็อดของอีกคลาสหนึ่งกี่ครั้ง ดังนั้นการวัดค่าการเชื่อมต่อระหว่างคลาสสามารถทำได้หลายลักษณะ เพียงแต่ค่าที่ได้จะมีความหยาบหรือละเอียดแตกต่างกัน การวัดค่าการเชื่อมต่อในแบบที่สองโดยนับเพียงจำนวนของแอสโซสิเอชันระหว่างคลาส ถึงแม้ว่าจะให้ค่าที่หยาบ แต่เนื่องจากมีสหสัมพันธ์กับการวัดแบบละเอียด จึงสามารถใช้เป็นตัวแทนในการบอกระดับความสัมพันธ์ระหว่างคลาสซึ่งจะส่งผลต่อระดับความสัมพันธ์ที่ส่วนประกอบซอฟต์แวร์จะเชื่อมต่อกันได้

จากการเทียบเคียงแผนภาพ การเชื่อมต่อกันระหว่างส่วนประกอบโพรเซสในงานวิจัยนี้จะพิจารณาจากความสัมพันธ์ระหว่างแอคชันที่อยู่ภายใต้คนละส่วนประกอบกัน โดยจะวัดความสัมพันธ์โดยใช้การวัดการเชื่อมติดแบบที่สอง ซึ่งเป็นการวัดจำนวนเส้นเชื่อมระหว่างแอคชัน

มาตรวัดการเชื่อมต่อกันระหว่างส่วนประกอบโพรเซสจะเทียบเคียงจากมาตรวัดของส่วนประกอบซอฟต์แวร์ได้ดังนี้

$$COUPL = \sum_{k=1}^{y} \sum_{i=1}^{n} \sum_{\substack{j=1 \\ i \neq j}}^{n} (x_{ik} * (1 - x_{jk}) * c_{ij})$$

โดยกำหนด

y จำนวนของส่วนประกอบโพรเซสในโดเมนของการออกแบบ

n จำนวนรวมของแอคชันในโดเมนการออกแบบ

x_{ik} มีค่าเท่ากับ 1 ถ้าแอคชัน i อยู่ภายในส่วนประกอบโพรเซส k
มีค่าเท่ากับ 0 ถ้าแอคชัน i ไม่ได้อยู่ภายในส่วนประกอบโพรเซส k

C_{ij} คือค่าการเชื่อมต่อ โดยค่านี้ได้จากการนับจำนวนลูกศรสายงานควบคุมที่ติดต่อระหว่างแอคชัน i และแอคชัน j

3.4.2 การเชื่อมติดกันภายในส่วนประกอบโพรเซส

งานวิจัย [17] เสนอมาตรวัดการเชื่อมติดกันภายในส่วนประกอบซอฟต์แวร์ดังนี้

$$COHES = \sum_{k=1}^{y} \sum_{i=1}^{n} \sum_{\substack{j=1 \\ i \neq j}}^{n} ((x_{ik} * x_{jk}) * c_{ij})$$

โดยกำหนด

y จำนวนของส่วนประกอบซอฟต์แวร์ในโดเมนของการออกแบบ

n จำนวนรวมของคลาสในโดเมนการออกแบบ

x_{ik} มีค่าเท่ากับ 1 ถ้าคลาส i อยู่ภายในส่วนประกอบซอฟต์แวร์ k
มีค่าเท่ากับ 0 ถ้าคลาส i ไม่ได้อยู่ภายในส่วนประกอบซอฟต์แวร์ k

C_{ij} คือค่าการเชื่อมต่อระหว่างคลาส i และคลาส j

มาตรวัดการเชื่อมติดกันภายในส่วนประกอบซอฟต์แวร์มีพื้นฐานอยู่บนการวัดการเชื่อมต่อระหว่างคลาสเช่นเดียวกับมาตรวัดการเชื่อมต่อระหว่างส่วนประกอบซอฟต์แวร์ข้างต้น ดังนั้นผู้วิจัยสามารถกำหนดมาตรวัดการเชื่อมติดกันภายในส่วนประกอบโพรเซสได้ในทำนองเดียวกัน โดยมีพื้นฐานอยู่บนการวัดการเชื่อมต่อระหว่างแอคชัน

มาตรวัดการเชื่อมติดกันภายในส่วนประกอบโพรเซสจะเทียบเคียงจากมาตรวัดของส่วนประกอบซอฟต์แวร์ได้ดังนี้

$$COHES = \sum_{k=1}^{y} \sum_{i=1}^{n} \sum_{\substack{j=1 \\ i \neq j}}^{n} ((x_{ik} * x_{jk}) * c_{ij})$$

โดยกำหนด

y จำนวนของส่วนประกอบโพรเซสในโดเมนของการออกแบบ

n จำนวนรวมของแอคชันในโดเมนการออกแบบ

x_{ik} มีค่าเท่ากับ 1 ถ้าแอคชัน i อยู่ภายในส่วนประกอบโพรเซส k
มีค่าเท่ากับ 0 ถ้าแอคชัน i ไม่ได้อยู่ภายในส่วนประกอบโพรเซส k

C_{ij} คือค่าการเชื่อมต่อ โดยค่านี้ได้จากการนับจำนวนลูกศรสายงานควบคุมที่ติดต่อระหว่างแอคชัน i และแอคชัน j

3.4.3 จำนวนของส่วนประกอบโพรเซส

งานวิจัย [17] เสนอมาตรวัดจำนวนของส่วนประกอบซอฟต์แวร์ดังนี้

$$NCOMP = y$$

โดยกำหนด

y จำนวนของส่วนประกอบซอฟต์แวร์ในโดเมนของการออกแบบ โดยค่า y จะมีค่าน้อยกว่าหรือเท่ากับ n ซึ่งเป็นจำนวนรวมของคลาสในโดเมนของการออกแบบ

การวัดจำนวนส่วนประกอบซอฟต์แวร์จากแผนภาพคลาสจะวัดได้จากจำนวนของแพ็คเกจที่ได้จากการแบ่งกลุ่มคลาส จากการเทียบเคียงแผนภาพ การวัดจำนวนส่วนประกอบโพรเซสก็สามารถวัดได้จากจำนวนแอคทิวิตีย่อยที่ได้จากการแบ่งกลุ่มแอคชันในแผนภาพแอคทิวิตีเช่นกัน

มาตรวัดจำนวนของส่วนประกอบโพรเซสจะเทียบเคียงจากมาตรวัดของส่วนประกอบซอฟต์แวร์ได้ดังนี้

$$NCOMP = y$$

โดยกำหนด

y จำนวนของส่วนประกอบโพรเซสในโดเมนของการออกแบบ โดยค่า y จะมีค่าน้อยกว่าหรือเท่ากับ n ซึ่งเป็นจำนวนรวมของแอคชันในโดเมนของการออกแบบ

3.4.4 ขนาดของส่วนประกอบโพรเซส

งานวิจัย [17] เสนอมาตรวัดขนาดของส่วนประกอบซอฟต์แวร์ดังนี้

$$CSIZE = \sqrt{\frac{\sum_{j=1}^{y}\left(\sum_{i=1}^{n} x_{ij}\right)^2}{y}}$$

โดยกำหนด

y จำนวนของส่วนประกอบซอฟต์แวร์ในโดเมนของการออกแบบ

n จำนวนรวมของคลาสในโดเมนของการออกแบบ

x_{ij} มีค่าเท่ากับ 1 ถ้าคลาส i อยู่ภายในส่วนประกอบซอฟต์แวร์ j

มีค่าเท่ากับ 0 ถ้าคลาส i ไม่ได้อยู่ภายในส่วนประกอบซอฟต์แวร์ j

มาตรวัดขนาดของส่วนประกอบซอฟต์แวร์นี้ได้จากการวัดจำนวนคลาสภายในส่วนประกอบซอฟต์แวร์ต่าง ๆ แล้วทำนอร์มอลไลเซชัน (Normalization) จากการเทียบเคียงแผนภาพ การวัดขนาดของส่วนประกอบโพรเซสก็สามารถวัดได้จากจำนวนแอคชันภายในส่วนประกอบโพรเซสแล้วทำนอร์มอลไลเซชันเช่นกัน

มาตรวัดขนาดของส่วนประกอบโพรเซสจะเทียบเคียงจากมาตรวัดของส่วนประกอบซอฟต์แวร์ได้ดังนี้

$$CSIZE = \sqrt{\frac{\sum_{j=1}^{y}\left(\sum_{i=1}^{n} x_{ij}\right)^2}{y}}$$

โดยกำหนด

y จำนวนของส่วนประกอบโพรเซสในโดเมนของการออกแบบ

n จำนวนรวมของแอคชันในโดเมนของการออกแบบ

x_{ij} มีค่าเท่ากับ 1 ถ้าแอคชัน i อยู่ภายในส่วนประกอบโพรเซส j

มีค่าเท่ากับ 0 ถ้าแอคชัน i ไม่ได้อยู่ภายในส่วนประกอบโพรเซส j

3.4.5 ความซับซ้อนของการออกแบบ

งานวิจัย [17] เสนอมาตรวัดความซับซ้อนของการออกแบบส่วนประกอบซอฟต์แวร์ดังนี้

$$COMPL = \sum_{j=1}^{y}\left(\sum_{i=1}^{n}\left(\left(w_{mcx} * m_i\right) + \left(w_{pcx} * \left(\sum_{k=1}^{m_i}\sum_{l=1}^{p_{ik}} p_{ikl}\right)\right)\right) * x_{ij}\right)^2$$

โดยกำหนด

y จำนวนของส่วนประกอบซอฟต์แวร์ในโดเมนของการออกแบบ

n จำนวนรวมของคลาสในโดเมนของการออกแบบ

x_{ij} มีค่าเท่ากับ 1 ถ้าคลาส i อยู่ภายในส่วนประกอบซอฟต์แวร์ j หรือ มีค่าเท่ากับ 0 ถ้าคลาส i ไม่ได้อยู่ภายในส่วนประกอบซอฟต์แวร์ j

W_{mcx} ค่าความสำคัญสัมพัทธ์ (Relative Importance) ของความซับซ้อนของเมท็อดภายในคลาส โดยมีค่ามากกว่าหรือเท่ากับ 0 แต่น้อยกว่าหรือเท่ากับ 1

m_i จำนวนของเมท็อดภายในคลาส i

W_{pcx} ค่าความสำคัญสัมพัทธ์ (Relative Importance) ของความซับซ้อนของพารามิเตอร์ของเมท็อดภายในคลาส โดยมีค่ามากกว่าหรือเท่ากับ 0 แต่น้อยกว่าหรือเท่ากับ 1

P_{ik} จำนวนของพารามิเตอร์ของเมท็อดภายในคลาส i โดยมีค่ามากกว่าหรือเท่ากับ 0

P_{ikl} ค่าความซับซ้อนสัมพัทธ์ (Relative Complexity) ของพารามิเตอร์ l ภายในเมท็อด k ของคลาส i โดยมีค่ามากกว่าหรือเท่ากับ 0 แต่น้อยกว่าหรือเท่ากับ 1

มาตรวัดนี้มีพื้นฐานอยู่บนการวัดจำนวนคลาสภายในส่วนประกอบซอฟต์แวร์ และเพื่อให้ได้ค่าการวัดที่ละเอียดขึ้น จึงมีการพิจารณารายละเอียดภายในคลาสด้วย ได้แก่การนับจำนวนเมท็อดในทุก ๆ คลาส และการนับจำนวนพารามิเตอร์ (ทั้งอินพุทและเอาท์พุท) ของทุก ๆ เมท็อด อีกทั้งเพื่อให้ค่าที่จะวัดได้สะท้อนความซับซ้อนได้ดียิ่งขึ้น จึงกำหนดค่าถ่วงน้ำหนัก (Weight) ไว้ด้วยตามความสำคัญของเมท็อด (W_{mcx}) ความสำคัญของพารามิเตอร์ (W_{pcx}) และความซับซ้อนของพารามิเตอร์แต่ละตัว (P_{ikl}) โดยนักออกแบบจะเป็นผู้ระบุค่าเหล่านี้ ตามทฤษฎีเชิงวัตถุนั้น การวัดความซับซ้อนของวัตถุจะสามารถวัดได้จากความซับซ้อนของเมท็อด และความซับซ้อนของพารามิเตอร์ [28] ดังนั้นหากโดยรวมแล้ว เมท็อดต่าง ๆ มีการคำนวณที่ยุ่งยากหรือมีการเรียกใช้เมท็อดกันเป็นจำนวนมาก ก็จะให้ค่าความสำคัญของเมท็อดมีค่ามาก ถ้าพารามิเตอร์ที่ใช้ในการทำงานมีความยุ่งยากซับซ้อนถูกเข้าถึงมาก หรือมีชื่อเดียวกันปรากฏในหลายคลาสมาก ก็จะให้ค่าความสำคัญของพารามิเตอร์มีค่ามาก สำหรับค่าความซับซ้อนของพารามิเตอร์แต่ละตัว ก็จะพิจารณาจากชนิดข้อมูลของพารามิเตอร์ เช่น ชนิดข้อมูลอาเรย์จะมีความซับซ้อนมากกว่าชนิดข้อมูลจำนวนเต็ม

เนื่องจากแผนภาพแอคทิวิตีเป็นแบบจำลองที่อยู่ในระดับบน (High Level) แต่ละแอคชันเป็นการทำงานที่เป็นนามธรรม (Abstract Function) โดยไม่มีรายละเอียดภายในว่าการทำงานนั้นจริง ๆ แล้วทำอย่างไร ผู้วิจัยเห็นว่าสามารถประยุกต์มาตรวัดความซับซ้อนของการออกแบบส่วนประกอบโพรเซสจากมาตรวัดของส่วนประกอบซอฟต์แวร์ข้างต้นได้ โดยลดทอนการวัดรายละเอียดภายในคลาสและการกำหนดค่าถ่วงน้ำหนักที่เกี่ยวข้อง ซึ่งไม่มีรายละเอียดที่เทียบเคียงได้ในแอคชัน ให้เหลือเป็นมาตรวัดความซับซ้อนอย่างหยาบ ซึ่งมาตรวัดนี้สอดคล้องกับการวัดความซับซ้อนแบบหยาบตามที่กล่าวถึงใน [17] ซึ่งมีพื้นฐานอยู่บนการวัดจำนวนคลาสในส่วนประกอบซอฟต์แวร์ ดังนั้นมาตรวัดที่ประยุกต์ได้จะมีพื้นฐานอยู่บนการวัดจำนวนแอคชันในส่วนประกอบโพรเซส

มาตรวัดความซับซ้อนของการออกแบบส่วนประกอบโพรเซสจะลดทอนจากมาตรวัดของส่วนประกอบซอฟต์แวร์ได้ดังนี้

$$COMPL = \sum_{j=1}^{y} \left(\sum_{i=1}^{n} x_{ij} \right)^2$$

โดยกำหนด

y จำนวนของส่วนประกอบโพรเซสในโดเมนของการออกแบบ

n จำนวนรวมของแอคชันในโดเมนของการออกแบบ

x_{ij} มีค่าเท่ากับ 1 ถ้าแอคชัน i อยู่ภายในส่วนประกอบโพรเซส j หรือ มีค่าเท่ากับ 0 ถ้าแอคชัน i ไม่ได้อยู่ภายในส่วนประกอบโพรเซส j

3.4.6 การวัดการบรรลุเป้าหมายด้านการจัดการส่วนประกอบ

เมื่อวัดลักษณะทางเทคนิคจากการออกแบบส่วนประกอบโพรเซสแล้ว จึงนำมาเข้าสูตรแบบจำลองคุณภาพ [17] เพื่อคำนวณค่าดัชนีการบรรลุเป้าหมายด้านการจัดการ (Achieving Managerial Goals Index: AMGI)

$$AMGI = R' * W * D$$

โดยกำหนด

R' เป็นเมทริกซ์ที่กำหนดสัดส่วนของเป้าหมายด้านการจัดการส่วนประกอบว่าเน้นเป้าหมายใด โดยกำหนดผลรวมของค่ามีค่าเท่ากับ 1 โดยระบุ R' ดังนี้

$$R' = [COST \quad ASBL \quad CUST \quad REUS \quad MNTN]$$

แล้วกำหนดสัดส่วนของเป้าหมายแต่ละตัว อาทิเช่น [0.8 0.05 0.05 0.05 0.05]

การกำหนดเป้าหมายด้านการจัดการส่วนประกอบดังข้างต้นนี้จะเน้นการใช้ต้นทุนอย่างมีประสิทธิผล มากถึง 8 ส่วนและเน้นเป้าหมายด้านการจัดการอื่น ๆ ที่เหลือเท่า ๆ กัน

W คือเมทริกซ์ของค่าความสัมพันธ์ระหว่างเป้าหมายด้านการจัดการส่วนประกอบ (แถว) และลักษณะทางเทคนิค (สดมภ์)

เป้าหมายด้านการจัดการส่วนประกอบเรียงจากบนลงล่างคือ การใช้ต้นทุนอย่างมีประสิทธิผล ความง่ายในการประกอบ ความสามารถในการปรับแต่ง ความสามารถในการนำกลับมาใช้ สภาพบำรุงรักษาได้

ลักษณะทางเทคนิคเรียงจากซ้ายไปขวาคือ การเชื่อมต่อกันระหว่างส่วนประกอบโพรเซส การเชื่อมติดกันภายในส่วนประกอบโพรเซส จำนวนของส่วนประกอบโพรเซส ขนาดของส่วนประกอบโพรเซส ความซับซ้อนของการออกแบบ

$$W = \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} & w_{15} \\ w_{21} & w_{22} & w_{23} & w_{24} & w_{25} \\ w_{31} & w_{32} & w_{33} & w_{34} & w_{35} \\ w_{41} & w_{42} & w_{43} & w_{44} & w_{45} \\ w_{51} & w_{52} & w_{53} & w_{54} & w_{55} \end{bmatrix}$$

ค่าอิลิเมนต์แต่ละตัวที่ใส่ใน **W** เป็นค่าความสัมพันธ์ระหว่างเป้าหมายด้านการจัดการส่วนประกอบและลักษณะทางเทคนิค โดยนักออกแบบซอฟต์แวร์จะเป็นผู้กำหนดความสัมพันธ์เหล่านี้ การกำหนดค่าให้กับแต่ละอิลิเมนต์จะสอดคล้องกับตารางที่ 2.2 โดยจะบ่งบอกปริมาณผลกระทบที่ลักษณะทางเทคนิคหนึ่ง ๆ จะมีต่อเป้าหมายด้านการจัดการแต่ละแบบกล่าวคือ

$$W = \begin{bmatrix} 0 & - & - & - & - \\ - & 0 & 0 & + & 0 \\ - & + & + & - & 0 \\ - & + & + & - & 0 \\ - & - & - & - & - \end{bmatrix}$$

โดยอิลิเมนต์ที่ระบุจะเป็นไปตามแนวทางด้านบนคือ

ตำแหน่งที่เป็น 0 ให้ระบุเป็น 0 (หมายถึงไม่มีผลกระทบ)

ตำแหน่งที่เป็น – ให้ระบุจำนวนเต็มลบมีค่าตั้งแต่ -1 ถึง -10 (หมายถึงมีผลกระทบในเชิงลบ)

ตำแหน่งที่เป็น + ให้ระบุจำนวนเต็มบวกที่มีค่าตั้งแต่ 1 ถึง 10 (หมายถึงมีผลกระทบในเชิงบวก)

D คือเมทริกซ์ของค่าที่วัดได้จากลักษณะทางเทคนิคดังนี้

$$D = \begin{bmatrix} COUPL \\ COHES \\ NCOMP \\ CSIZE \\ COMPL \end{bmatrix}$$

ค่าดัชนีการบรรลุเป้าหมายด้านการจัดการจะบ่งบอกได้ว่าการออกแบบส่วนประกอบโพรเซสแบบหนึ่ง ๆ สามารถบรรลุเป้าหมายด้านการจัดการได้ดีเพียงใดโดยเปรียบเทียบกับดัชนีที่วัดได้จากการออกแบบแบบอื่น ๆ กล่าวคือ การออกแบบที่ได้ค่าดัชนีมากกว่าค่าดัชนีที่ได้จากการออกแบบแบบอื่น จะสามารถบรรลุเป้าหมายด้านการจัดการได้ดีกว่า

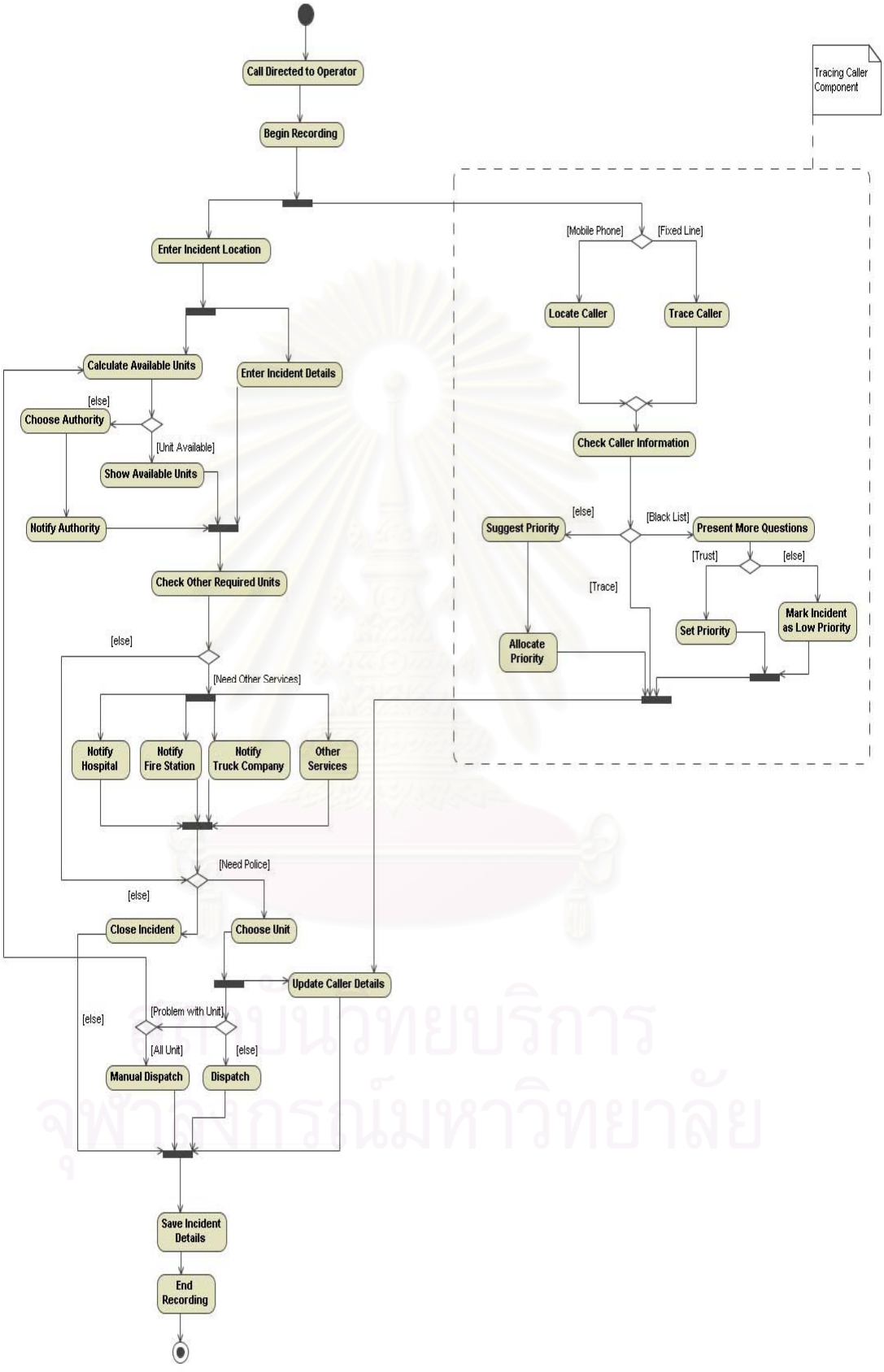
บทที่ 4
แนวคิดการนำส่วนประกอบโพรเซสกลับมาใช้ใหม่

เนื่องจากค่าใช้จ่ายในการออกแบบแบบจำลองกระบวนการทางธุรกิจเป็นส่วนที่ต้องคำนึงถึงในค่าใช้จ่ายโดยรวมของการพัฒนาแอพลิเคชัน รวมไปถึงปัญหาความน่าเชื่อถือของการออกแบบใหม่โดยปราศจากการทดสอบและการนำไปใช้จริงมาก่อน การนำส่วนประกอบโพรเซสกลับมาใช้ใหม่จึงเป็นแนวทางที่เป็นประโยชน์ต่อการออกแบบแบบจำลองกระบวนการทางธุรกิจสำหรับแอพพลิเคชันใหม่ ๆ เนื่องจากสามารถลดค่าใช้จ่ายในการออกแบบ และเพิ่มความน่าเชื่อถือให้แก่การออกแบบ ในงานวิจัยนี้ได้นำเสนอการวัดการออกแบบส่วนประกอบโพรเซส ซึ่งมีการจัดกลุ่มส่วนประกอบโพรเซสซึ่งสามารถนำกลับมาใช้ใหม่ได้ ดังนั้นในบทนี้ผู้วิจัยจะแสดงแนวคิดการนำส่วนประกอบโพรเซสกลับมาใช้ใหม่ในโดเมนงานที่ต่างกันสองโดเมนงานคือ ระบบตอบรับโทรศัพท์ของตำรวจ และระบบตอบรับโทรศัพท์ของสถานีดับเพลิง ซึ่งโดเมนระบบตอบรับโทรศัพท์ของตำรวจได้ประยุกต์จาก [29] ส่วนโดเมนสถานีดับเพลิงได้นำส่วนประกอบโพรเซสบางส่วนของโดเมนงานแรกมาใช้

4.1 ระบบตัวอย่าง

4.1.1. ระบบตอบรับโทรศัพท์ของตำรวจ

ระบบตอบรับโทรศัพท์ของตำรวจ แสดงดังรูปที่ 4.1 โดยเริ่มต้นจากมีผู้แจ้งเหตุร้ายโทรมายังโอเปอเรเตอร์ของระบบ โอเปอเรเตอร์จะเริ่มทำการบันทึกข้อมูลผู้โทรโดยสอบถามสถานที่เกิดเหตุในขณะเดียวกับที่ทำการตรวจสอบตำแหน่งของผู้โทรว่าโทรมาจากโทรศัพท์มือถือ หรือโทรศัพท์บ้านซึ่งอยู่ที่ใด จากนั้นทำการตรวจสอบข้อมูลของผู้โทรว่าเชื่อถือได้หรือไม่ หากผู้โทรไม่มีประวัติเสียมาก่อน ก็จะกำหนดระดับความสำคัญให้กับการแจ้งเหตุครั้งนี้ แต่ถ้าผู้โทรมีประวัติเสียก็จะสอบถามข้อมูลเพิ่มเติม ถ้าน่าเชื่อถือก็จะกำหนดระดับความสำคัญให้กับการแจ้งเหตุ แต่ถ้าไม่น่าเชื่อถือจะกำหนดระดับความสำคัญให้ต่ำสุดและจะมีการบันทึกข้อมูลของผู้โทรเก็บไว้ด้วย ในระหว่างที่ทำการตรวจสอบความน่าเชื่อถือของการแจ้งเหตุ ระบบจะตรวจสอบกำลังตำรวจที่ว่างอยู่ในขณะนั้นด้วย แล้วทำการแสดงหน่วยที่ว่าง แต่หากไม่มีหน่วยใดว่างก็จะทำการแจ้งผู้บังคับบัญชา จากนั้นจึงทำการตรวจสอบว่าต้องการกำลังจากหน่วยอื่นหรือไม่ได้แก่ โรงพยาบาล สถานีดับเพลิง รถบรรทุกลาก หรือบริการอื่น ๆ หลังจากนั้นหากไม่ต้องการกำลังตำรวจในที่เกิดเหตุ ก็จะปิดการแจ้งเหตุ แต่หากต้องการกำลังตำรวจ ระบบก็จะพยายามเลือกหน่วยที่สามารถไปปฏิบัติงานในที่เกิดเหตุได้ และก่อนจบการแจ้งเหตุจะมีการเก็บบันทึกข้อมูลเหตุร้ายไว้

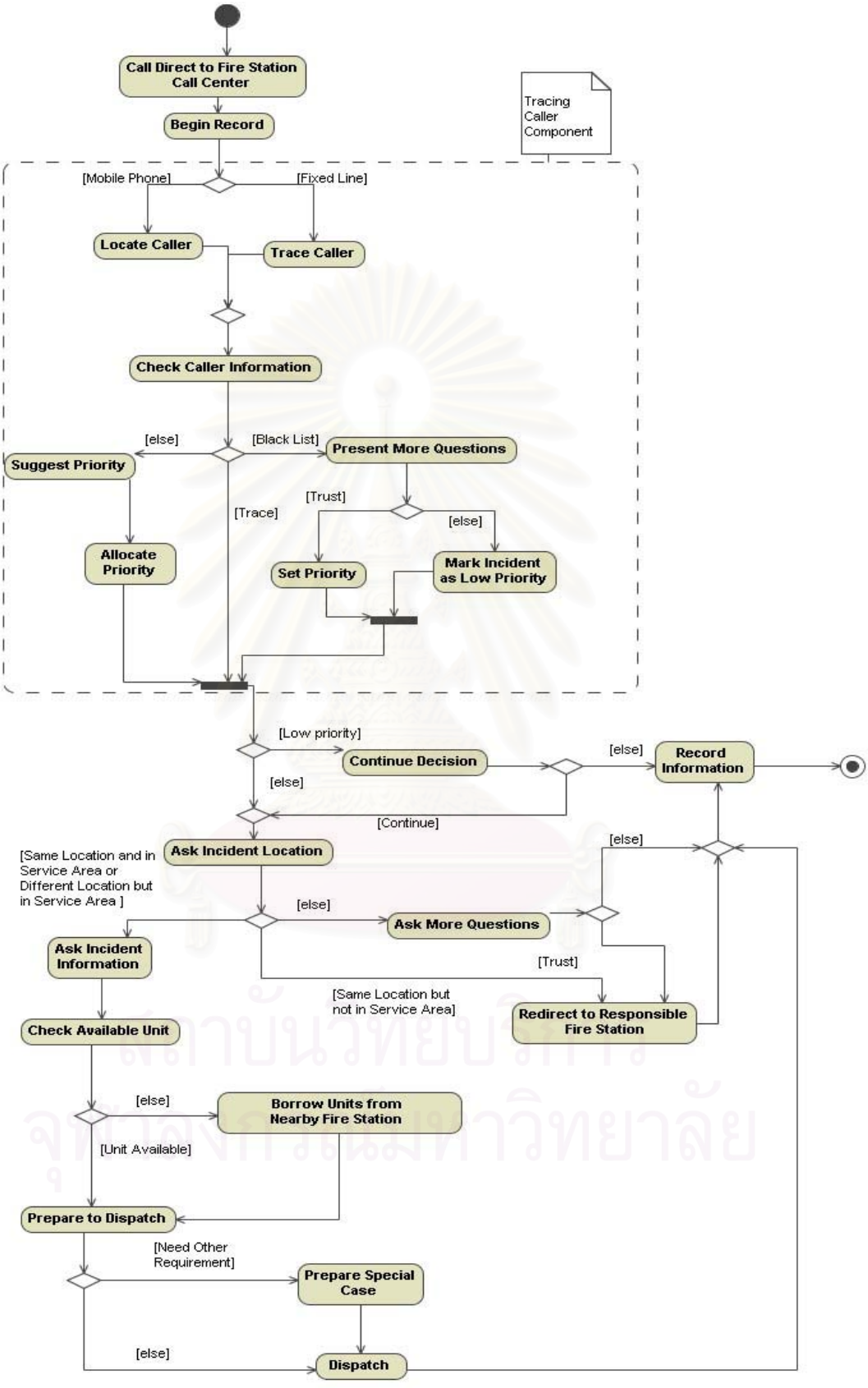


รูปที่ 4.1 ระบบตอบรับโทรศัพท์ของตำรวจ

4.1.2. ระบบตอบรับโทรศัพท์ของสถานีดับเพลิง

ระบบตอบรับโทรศัพท์ของสถานีดับเพลิงมีขั้นตอนการทำงานที่คล้ายกับระบบตอบรับโทรศัพท์ของตำรวจจากโดเมนแรกดังนั้นผู้วิจัยจึงได้ดัดแปลงกระบวนการทางธุรกิจบางอย่าง แต่ก็จะมีรายละเอียดปลีกย่อยต่างกันที่ระบบนี้ไม่มีเบอร์โทรกลางของสถานีดับเพลิง เมื่อเกิดเหตุผู้แจ้งเหตุต้องโทรไปแจ้งยังแต่ละสถานีดับเพลิงเอง ทำให้ต้องมีการตรวจสอบพื้นที่ของการโทรและมีการโอนสายไปยังสถานีดับเพลิงใกล้เคียงในกรณีที่อยู่นอกพื้นที่ หรือมีจำนวนอุปกรณ์หรือเจ้าหน้าที่ไม่เพียงพอ

จากรูปที่ 4.2 ระบบตอบรับโทรศัพท์ของสถานีดับเพลิงเริ่มที่มีผู้แจ้งเหตุอัคคีภัยมายังโอเปอเรเตอร์ของสถานีดับเพลิง โอเปอเรเตอร์เริ่มบันทึกเหตุการณ์และตรวจสอบการแจ้งเหตุว่ามาจากที่ใด จากนั้นจึงตรวจสอบข้อมูลผู้โทรแจ้ง หากผู้โทรไม่มีประวัติเสียมาก่อน ก็จะกำหนดระดับความสำคัญให้กับการแจ้งเหตุครั้งนี้ แต่ถ้าผู้โทรมีประวัติเสียก็จะสอบถามข้อมูลเพิ่มเติม ถ้าน่าเชื่อถือก็จะกำหนดระดับความสำคัญให้กับการแจ้งเหตุ แต่ถ้าไม่น่าเชื่อถือจะกำหนดระดับความสำคัญให้ต่ำสุดและจะมีการบันทึกข้อมูลของผู้โทรเก็บไว้ด้วย ในกรณีที่การแจ้งเหตุมีระดับความสำคัญสูงจะสอบถามสถานที่เกิดเหตุอัคคีภัย ถ้าการแจ้งเหตุมาจากสถานที่เกิดเหตุเลยแต่ไม่ได้อยู่ในพื้นที่ให้บริการของสถานีดับเพลิงระบบจะโอนสายไปยังสถานีดับเพลิงที่มีพื้นที่ให้บริการครอบคลุมสถานที่เกิดเหตุอัคคีภัยแล้วบันทึกข้อมูลและจบการแจ้งเหตุ แต่ถ้าการแจ้งเหตุไม่ได้มาจากสถานที่เกิดเหตุระบบจะสอบถามข้อมูลเพิ่มเติม ในกรณีที่ไม่น่าเชื่อถือจะทำการบันทึกข้อมูลแล้วจบการแจ้งเหตุ แต่ถ้าน่าเชื่อถือจะทำการโอนสายไปให้สถานีดับเพลิงที่ใกล้เคียงกับที่เกิดเหตุบันทึกข้อมูล และจบการแจ้งเหตุอัคคีภัย ณ สถานีดับเพลิงนั้น ถ้าการแจ้งเหตุมาจากสถานที่เกิดเหตุและอยู่ในพื้นที่ให้บริการของสถานีดับเพลิงหรือไม่ได้แจ้งมาจากสถานทีเกิดเหตุแต่สถานที่เกิดเหตุอยู่ในพื้นที่ให้บริการ ระบบจะสอบถามเหตุการณ์และตรวจสอบหน่วยดับเพลิงที่ต้องส่งไปในที่เกิดเหตุ ในกรณีที่หน่วยดับเพลิงไม่เพียงพอทำการ ระบบจะแจ้งไปยังสถานีดับเพลิงใกล้เคียงเพื่อขอกำลังเสริม ถ้ามีความต้องการอื่นเป็นพิเศษเช่น ต้องทำการดับเพลิงในตึกสูง หรือเกิดอัคคีภัยจากสารเคมีอันตราย ระบบจะต้องเตรียมเครื่องมือพิเศษ ในที่สุดระบบจะส่งหน่วยดับเพลิงออกไปและทำการบันทึกข้อมูลแล้วจบการแจ้งเหตุอัคคีภัย



รูปที่ 4.2 ระบบตอบรับโทรศัพท์ของสถานีดับเพลิง

4.2. การนำส่วนประกอบโพรเซสกลับมาใช้ใหม่

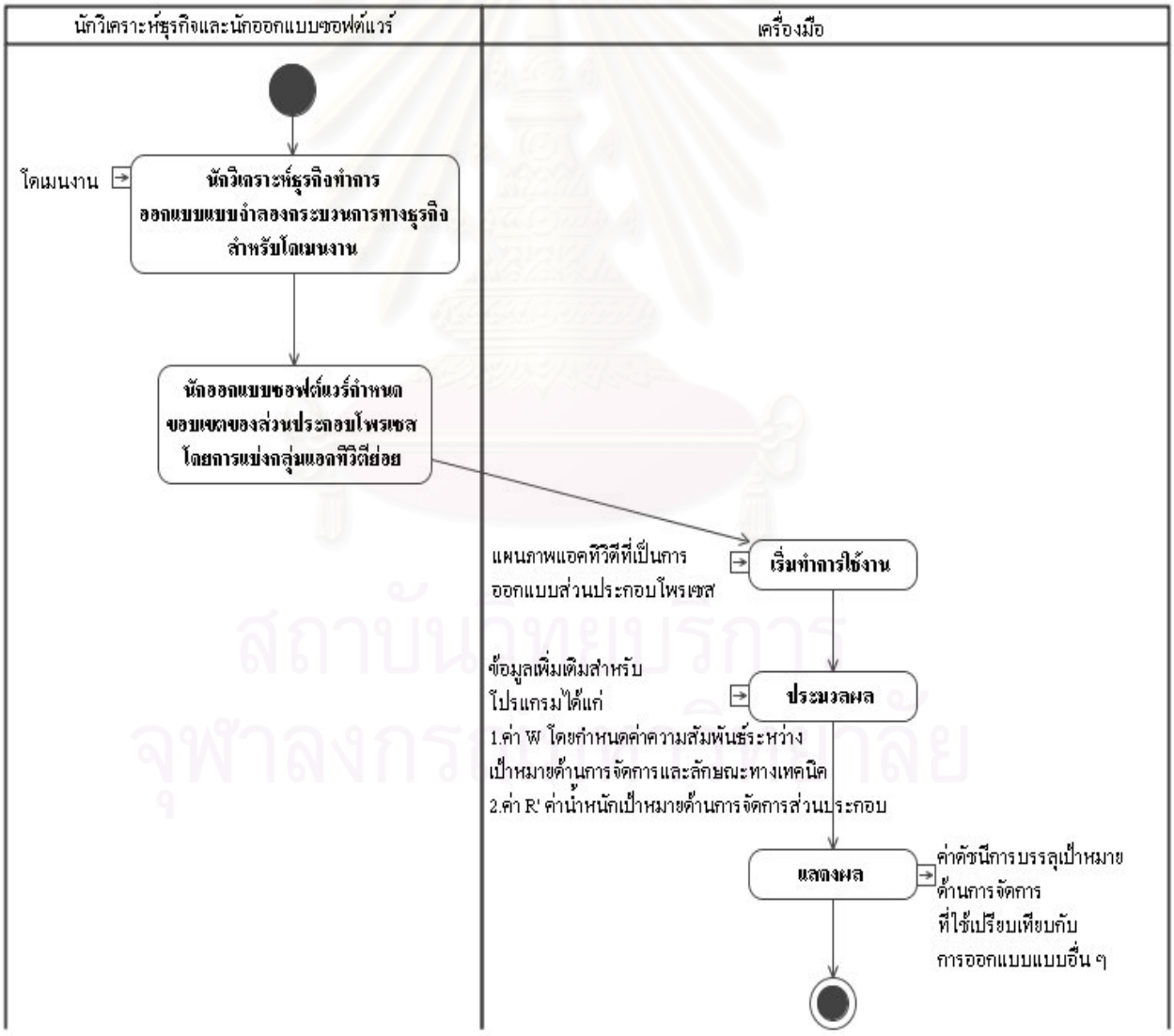
ในงานวิจัยนี้ได้มีการจัดกลุ่มส่วนประกอบโพรเซสตามความเหมาะสมจากนักออกแบบซอฟต์แวร์ และมีการนำส่วนประกอบโพรเซสกลับมาใช้ใหม่ จากรูปที่ 4.1 มีการจับกลุ่มของแอคชันที่มีหน้าที่ต่อเนื่องหรือใกล้เคียงกันและทำงานร่วมกันให้เป็นแอคทิวิตีย่อยซึ่งในที่นี้คือ ส่วนประกอบโพรเซสที่มีชื่อว่า Trace Caller Component ซึ่งทำการตรวจสอบผู้โทรแจ้งเหตุ เมื่อมีการออกแบบระบบใหม่ได้แก่ ระบบตอบรับโทรศัพท์ของสถานีดับเพลิงซึ่งมีความใกล้เคียงกับระบบตอบรับโทรศัพท์ของตำรวจ ดังนั้นจึงสามารถนำส่วนประกอบที่ได้ทำการออกแบบไว้แล้วนำกลับมาใช้ดังรูปที่ 4.2 จากการที่สามารถนำกลับมาใช้ใหม่ทำให้ลดค่าใช้จ่ายในการออกแบบพร้อมทั้งยังเพิ่มความน่าเชื่อถือให้กับระบบที่ทำการออกแบบใหม่

ตัวอย่างของระบบที่กล่าวถึงในบทนี้จะนำไปเป็นกรณีศึกษาในการทดสอบเครื่องมือในบทที่ 7 ต่อไปด้วย

บทที่ 5
การออกแบบเครื่องมือวัดการออกแบบส่วนประกอบโพรเซส

ในบทนี้เป็นการกล่าวถึงการออกแบบเครื่องมือที่ช่วยสนับสนุนการวัดการออกแบบส่วนประกอบโพรเซส ตามเป้าหมายด้านการจัดการส่วนประกอบ โดยเครื่องมือที่ทำการออกแบบนี้จะสามารถช่วยให้นักออกแบบซอฟต์แวร์ตัดสินใจในการออกแบบได้ดียิ่งขึ้น โดยขั้นตอนการออกแบบและการทำงานของเครื่องมือจะต้องสามารถคำนวณค่ามาตรวัดได้ตามขั้นตอนดังรูปที่ 5.1 ดังมาตรวัดที่ได้กำหนดไว้ในบทที่ 3

5.1 การออกแบบการทำงานของเครื่องมือ



รูปที่ 5.1 ภาพรวมของกระบวนการวัด

5.1.1. นักวิเคราะห์ธุรกิจทำการออกแบบแบบจำลองกระบวนการทางธุรกิจสำหรับโดเมนงาน

ในขั้นตอนนี้ นักวิเคราะห์ธุรกิจทำการออกแบบกระบวนการทางธุรกิจด้วยแผนภาพแอคทิวิตี โดยในขั้นตอนนี้มีอินพุทของขั้นตอนคือโดเมนงาน

5.1.2. นักออกแบบซอฟต์แวร์กำหนดขอบเขตของส่วนประกอบโพรเซสโดยการแบ่งกลุ่มแอคทิวิตีย่อย

ในขั้นตอนนี้นักออกแบบซอฟต์แวร์ทำการจัดกลุ่มกระบวนการทางธุรกิจที่ออกแบบโดยแผนภาพแอคทิวิตี โดยจัดกลุ่มของแอคชันที่มีหน้าที่การทำงานใกล้เคียงกันหรือระยะเวลาประมวลผลที่ใกล้เคียงกันไว้ด้วยกัน โดยกำหนดขอบเขตของแอคทิวิตีย่อยและระบุชื่อของแต่ละแอคทิวิตีย่อยนั้น

5.1.3. เริ่มการทำงานของเครื่องมือ

ในขั้นตอนนี้นักออกแบบซอฟต์แวร์ต้องป้อนไฟล์ข้อมูลการออกแบบในรูปแบบเอ็กซ์เอ็มไอให้กับเครื่องมือเพื่อให้ทำการประมวลผลในขั้นตอนถัดไป

ในการที่จะได้ข้อมูลของการออกแบบในรูปเอ็กซ์เอ็มไอมานั้นจะต้องมีการแลกเปลี่ยนข้อมูลกับเครื่องมือแผนภาพยูเอ็มแอล โดยจากการสำรวจเครื่องมือสร้างแผนภาพยูเอ็มแอลต่าง ๆ พบว่าส่วนใหญ่สามารถส่งออกหรือมีการจัดเก็บแบบจำลองยูเอ็มแอลในรูปของแฟ้มข้อมูลชนิดเอ็กซ์เอ็มแอลตามมาตรฐานเอ็กซ์เอ็มไอซึ่งเป็นมาตรฐานที่สามารถทำให้การแลกเปลี่ยนข้อมูลแบบจำลองยูเอ็มแอลระหว่างเครื่องมือต่าง ๆ เป็นไปได้โดยสะดวก ดังนั้นผู้วิจัยจึงได้เลือกใช้แฟ้มข้อมูลชนิดนี้เป็นข้อมูลเข้าของเครื่องมือที่จะพัฒนาขึ้นมา อย่างไรก็ตามโครงสร้างแฟ้มข้อมูลดังกล่าวจะมีรายละเอียดที่แตกต่างกันไปตามแต่ผู้ผลิต ซึ่งผู้วิจัยเลือกที่จะพัฒนาเครื่องมือที่สามารถทำงานกับแฟ้มข้อมูลที่ส่งออกจากเมจิกดรอว์ (Magic Draw) รุ่น 10.0

5.1.4. เครื่องมือทำการประมวลผล

ในขั้นตอนนี้เครื่องมือจะทำการประมวลผลตามที่ได้กำหนดไว้ในมาตรวัดและคำนวณค่าดัชนีการบรรลุเป้าหมายด้านการจัดการส่วนประกอบดังที่อธิบายรายละเอียดไว้ในบทที่ 3 โดยจะต้องกำหนดข้อมูลเพิ่มเติมภายในขั้นตอนนี้ได้แก่เมทริกซ์แสดงค่าความสัมพันธ์ระหว่างข้อมูลทางเทคนิค และเป้าหมายทางด้านการจัดการส่วนประกอบให้กับเครื่องมือ

5.1.5. เครื่องมือทำการแสดงผลค่าดัชนีการบรรลุเป้าหมายด้านการจัดการ

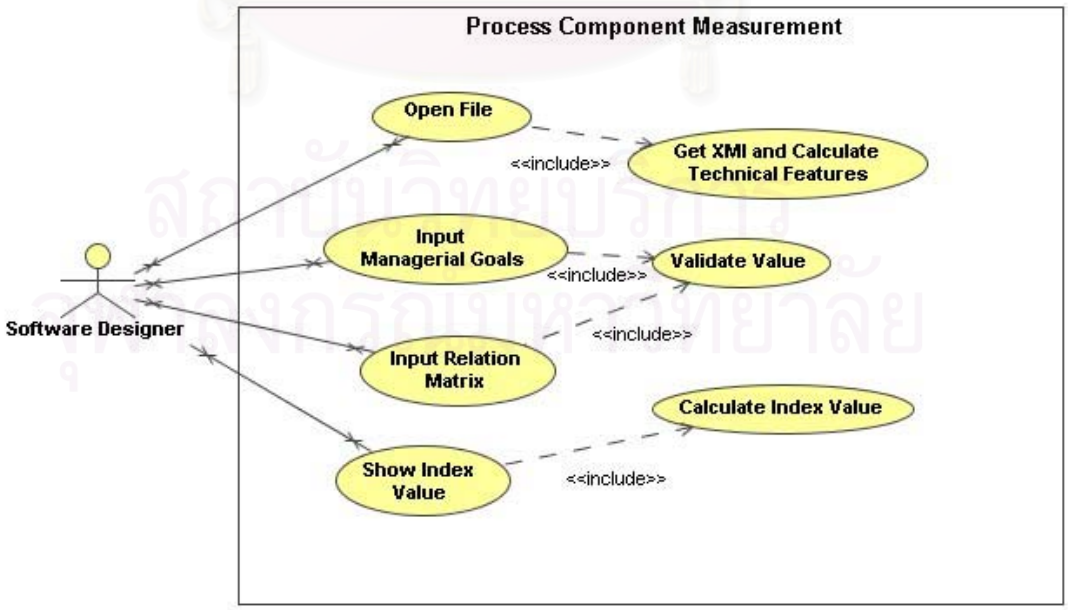
เครื่องมือจะแสดงค่าดัชนีการบรรลุเป้าหมายด้านการจัดการออกมาในขั้นตอนนี้ นักออกแบบซอฟต์แวร์จะเป็นผู้ตัดสินใจว่าค่าที่ได้ออกมานั้นตรงตามเป้าหมายด้านการจัดการส่วนประกอบเพียงใด โดยเปรียบเทียบกับค่าดัชนีที่ได้จากการออกแบบก่อนหน้านี้ ถ้าผู้ออกแบบยังไม่พอใจค่าดัชนีที่ได้ก็จะสามารถออกแบบส่วนประกอบโพรเซสใหม่แล้วนำมาเปรียบเทียบกับผลที่ได้จากการออกแบบครั้งก่อนหน้าอีกครั้ง

5.2 การออกแบบเครื่องมือ

ภายหลังจากที่ได้อธิบายภาพรวมของเครื่องมือแล้วผู้วิจัยจะนำเสนอการออกแบบการทำงานภายในเครื่องมือดังนี้

5.2.1. การออกแบบการใช้งานเครื่องมือ

รูปที่ 5.2 แสดงแผนภาพยูสเคสของเครื่องมือสนับสนุนการวัดค่าการออกแบบ ที่ได้ทำการพัฒนาขึ้นมา โดยคำอธิบายยูสเคสของเครื่องมือแสดงดังตาราง ก.1-ก.7 ภาคผนวก ก โดยนักออกแบบซอฟต์แวร์บันทึกข้อมูลแผนภาพจากแผนภาพแอคทิวิตีที่อยู่ในรูปของแฟ้มข้อมูลเอ็กซ์เอ็มไอ โดยรายละเอียดของกิจกรรมต่าง ๆ ที่เครื่องมือดังกล่าวจะดำเนินการเพื่อให้ได้ค่าดัชนีที่นักออกแบบซอฟต์แวร์สามารถนำไปเปรียบเทียบได้ จะได้กล่าวในหัวข้อต่อไป



รูปที่ 5.2 แผนภาพยูสเคสของเครื่องมือวัดการออกแบบส่วนประกอบโพรเซส

5.2.2. การออกแบบกิจกรรมต่าง ๆ ในการวัดการออกแบบ

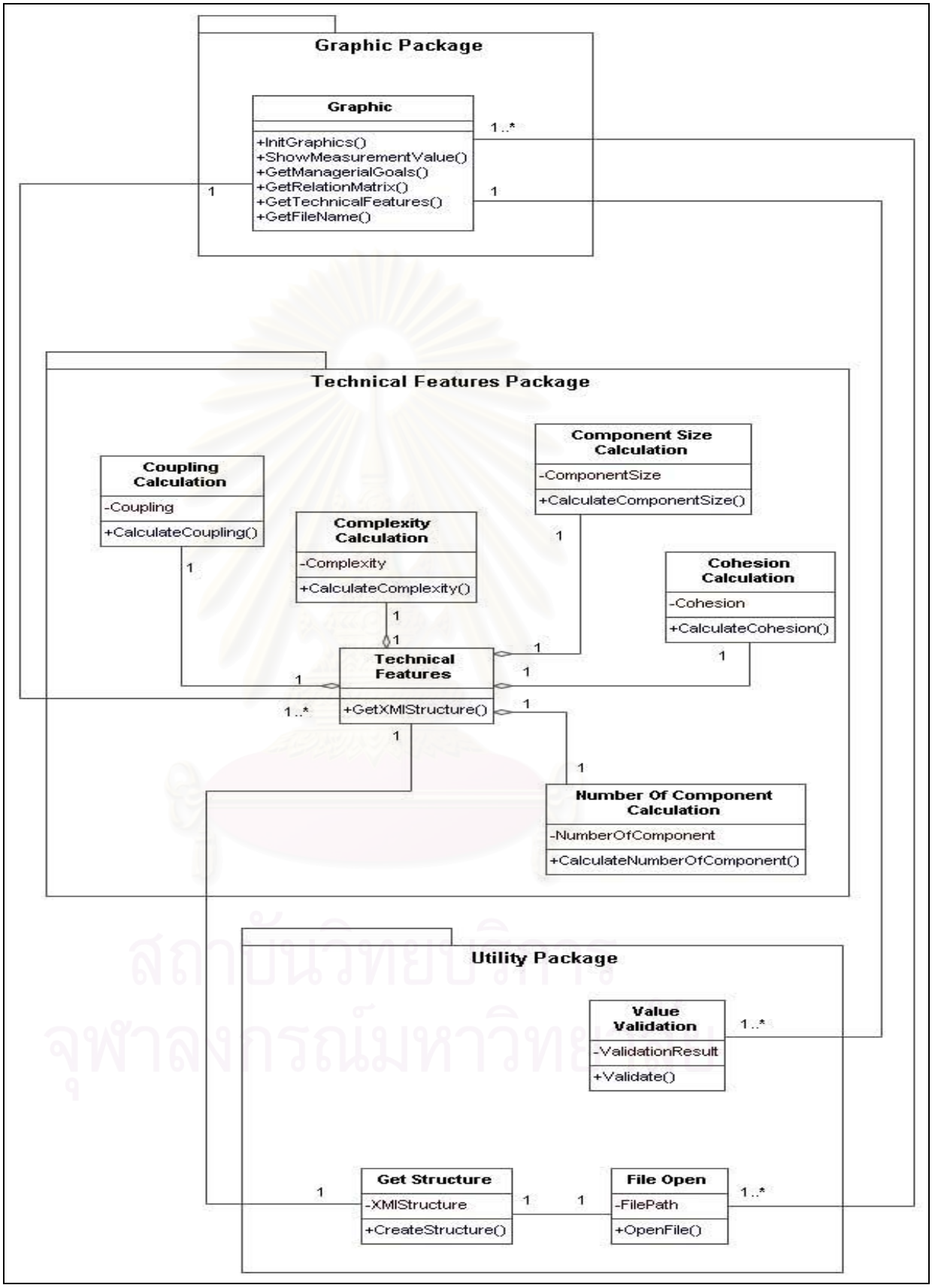
รูปที่ 5.3 แสดงกิจกรรมต่าง ๆ ในขั้นตอนการวัดค่าการออกแบบของเครื่องมือ โดยกิจกรรมแรกเป็นการอ่านค่าจากแฟ้มข้อมูลเอ็กซ์เอ็มไอซึ่งได้มาจากเครื่องมือสร้างยูเอ็มแอลเพื่อดึงข้อมูลโครงสร้างออกมา (รายละเอียดของโครงสร้างแฟ้มข้อมูลเอ็กซ์เอ็มไอจะกล่าวถึงในหัวข้อที่ 5.2.3) ในกิจกรรมที่ 2 เป็นการจัดเก็บข้อมูลในรูปแบบโครงสร้างเพื่อให้สามารถดึงมาใช้คำนวณตามมาตรวัดได้อย่างสะดวกและรวดเร็วกว่าไปทำการอ่านข้อมูลจากแฟ้มข้อมูลเอ็กซ์เอ็มไอโดยตรง กิจกรรมที่ 3 เป็นการคำนวณค่าลักษณะทางเทคนิคจากโครงสร้างที่ได้อ่านมาจากแฟ้มข้อมูลเอ็กซ์เอ็มไอแล้ว โดยรายละเอียดเป็นดังรูปที่ 5.4 โดยมีลำดับการคำนวณลักษณะทางเทคนิคได้แก่ การเชื่อมต่อกันระหว่างส่วนประกอบ การเชื่อมติดกันภายในส่วนประกอบ จำนวนของส่วนประกอบ ขนาดของส่วนประกอบ และความซับซ้อนของการออกแบบ กิจกรรมที่ 4 เป็นการคำนวณค่าดัชนีการบรรลุเป้าหมายด้านการจัดการ โดยอาศัยค่าลักษณะทางเทคนิคที่วัดได้จากกิจกรรมที่ 3 และข้อมูลเพิ่มเติมที่ได้จากนักออกแบบซอฟต์แวร์ได้แก่ข้อมูลเป้าหมายด้านการจัดการส่วนประกอบและเมทริกซ์ความสัมพันธ์ระหว่างลักษณะทางเทคนิคกับเป้าหมายด้านการจัดการส่วนประกอบ กิจกรรมที่ 5 แสดงค่าดัชนีจากการออกแบบ โดยนักออกแบบซอฟต์แวร์สามารถใช้งานเครื่องมือในการวัดการออกแบบส่วนประกอบโพรเซสในแบบต่าง ๆ เพื่อนำค่าดัชนีที่วัดได้มาเปรียบเทียบกันและตัดสินใจเลือกการออกแบบที่จะใช้ในการพัฒนาส่วนประกอบโพรเซสต่อไป

รูปที่ 5.3 ภาพรวมกิจกรรมของเครื่องมือที่ใช้ทำการวัด

รูปที่ 5.4 กิจกรรมในขั้นตอนการคำนวณลักษณะทางเทคนิค

5.2.3. แผนภาพคลาสของเครื่องมือ

ในส่วนนี้เป็นการออกแบบเครื่องมือในมุมมองของแผนภาพคลาสโดยในการออกแบบนี้แบ่งแผนภาพออกเป็นสามแพ็กเกจคือ แพ็กเกจกราฟิก แพ็กเกจลักษณะทางเทคนิค และแพ็กเกจอรรถประโยชน์โดยแผนภาพนี้ประกอบไปด้วย 10 คลาสดังรูปที่ 5.5 โดยแพ็กเกจกราฟิกมีหน้าที่แสดงผลและรับข้อมูลเข้าจากนักออกแบบซอฟต์แวร์ แพ็กเกจลักษณะทางเทคนิคมีหน้าที่คำนวณค่าลักษณะทางเทคนิค ส่วนแพ็กเกจอรรถประโยชน์มีหน้าที่เปิดแฟ้มข้อมูลโครงสร้างเอ็กซ์เอ็มไอสร้างโครงสร้างของข้อมูลเพื่อให้แพ็กเกจลักษณะทางเทคนิคนำไปใช้คำนวณ และตรวจสอบความถูกต้องของข้อมูลที่นักออกแบบซอฟต์แวร์ป้อนเข้าสู่เครื่องมือก่อนนำไปคำนวณค่าดัชนีการบรรลุเป้าหมายด้านการจัดการและแสดงผลต่อไป



รูปที่ 5.5 แผนภาพคลาสของเครื่องมือ

5.2.4. โครงสร้างของแฟ้มข้อมูลเอ็กซ์เอ็มไอที่ส่งเข้าเครื่องมือ

ข้อมูลเชิงโครงสร้างของแผนภาพแอคทิวิตีที่อยู่ในรูปของแฟ้มข้อมูลเอ็กซ์เอ็มไอ จะถูกใช้เป็นหลักในการคำนวณลักษณะทางเทคนิคเพื่อนำไปใช้ในการคำนวณค่าดัชนีการบรรลุเป้าหมายด้านการจัดการต่อไป ตารางที่ 5.1 ถึง 5.8 แสดงรายละเอียดของแต่ละส่วนย่อยในโครงสร้างเอ็กซ์เอ็มแอลสำหรับข้อมูลเชิงโครงสร้าง

ตารางที่ 5.1 รายละเอียดส่วนย่อย “Activity”

ชื่อส่วนย่อย	Activity
คำอธิบาย	ทำหน้าที่เป็นราก (Root) ของข้อมูลเชิงโครงสร้าง
แอททริบิวต์	
xmi:id	รหัสเพื่อใช้ระบุตัวตนของส่วนย่อย
name	ชื่อของแอคทิวิตี
visibility	ประเภทของขอบเขตการมองเห็น โดยมีค่าได้ 3 แบบ ได้แก่ Private, Protected และ Public
owningPackage	รหัสแพ็กเกจที่เป็นเจ้าของแอคทิวิตีนี้

ตารางที่ 5.2 รายละเอียดส่วนย่อย “StructuredActivityNode”

ชื่อส่วนย่อย	StructuredActivityNode
คำอธิบาย	เก็บข้อมูลแอคทิวิตีย่อย
แอททริบิวต์	
xmi:id	รหัสเพื่อใช้ระบุตัวตนของส่วนย่อย
visibility	ประเภทของขอบเขตการมองเห็น โดยมีค่าได้ 3 แบบ ได้แก่ Private, Protected และ Public
activity	รหัสแอคทิวิตีที่แอคทิวิตีย่อยนี้บรรจุอยู่

ตารางที่ 5.3 รายละเอียดส่วนย่อย “CallBehaviorAction”

ชื่อส่วนย่อย	CallBehaviorAction
คำอธิบาย	เก็บข้อมูลของแอคชัน
แอททริบิวต์	
xmi:id	รหัสเพื่อใช้ระบุตัวตนของส่วนย่อย
name	ชื่อของแอคชัน
visibility	ประเภทของขอบเขตการมองเห็น โดยมีค่าได้ 3 แบบ ได้แก่ Private, Protected และ Public
activity	รหัสแอคทิวิตีที่แอคชันนี้บรรจุอยู่
outgoing	รหัสของสายงานควบคุมที่ออกจากแอคชันนี้
incoming	รหัสของสายงานควบคุมที่เข้าสู่แอคชันนี้

ตารางที่ 5.4 รายละเอียดส่วนย่อย “InitialNode”

ชื่อส่วนย่อย	InitialNode
คำอธิบาย	เก็บข้อมูลของโหนดเริ่มต้น
แอททริบิวต์	
xmi:id	รหัสเพื่อใช้ระบุตัวตนของส่วนย่อย
visibility	ประเภทของขอบเขตการมองเห็น โดยมีค่าได้ 3 แบบ ได้แก่ Private, Protected และ Public
activity	รหัสแอคทิวิตีที่โหนดเริ่มต้นนี้บรรจุอยู่
outgoing	รหัสของสายงานควบคุมที่ออกจากโหนดเริ่มต้นนี้

ตารางที่ 5.5 รายละเอียดส่วนย่อย “DecisionNode”

ชื่อส่วนย่อย	DecisionNode
คำอธิบาย	เก็บข้อมูลโหนดตัดสินใจ
แอททริบิวต์	
xmi:id	รหัสเพื่อใช้ระบุตัวตนของส่วนย่อย
visibility	ประเภทของขอบเขตการมองเห็น โดยมีค่าได้ 3 แบบ ได้แก่ Private, Protected และ Public
activity	รหัสแอคทิวิตีที่โหนดเริ่มต้นนี้บรรจุอยู่
outgoing	รหัสของสายงานควบคุมที่ออกจากโหนดนี้
incoming	รหัสของสายงานควบคุมที่เข้าสู่โหนดนี้

ตารางที่ 5.6 รายละเอียดส่วนย่อย “ForkNode”

ชื่อส่วนย่อย	ForkNode
คำอธิบาย	เก็บข้อมูลการแยกออก
แอททริบิวต์	
xmi:id	รหัสเพื่อใช้ระบุตัวตนของส่วนย่อย
visibility	ประเภทของขอบเขตการมองเห็น โดยมีค่าได้ 3 แบบ ได้แก่ Private, Protected และ Public
activity	รหัสแอคทิวิตีที่การแยกออกนี้บรรจุอยู่
outgoing	รหัสของสายงานควบคุมที่ออกจากโหนดนี้
incoming	รหัสของสายงานควบคุมที่เข้าสู่โหนดนี้

ตารางที่ 5.7 รายละเอียดส่วนย่อย “ActivityFinalNode”

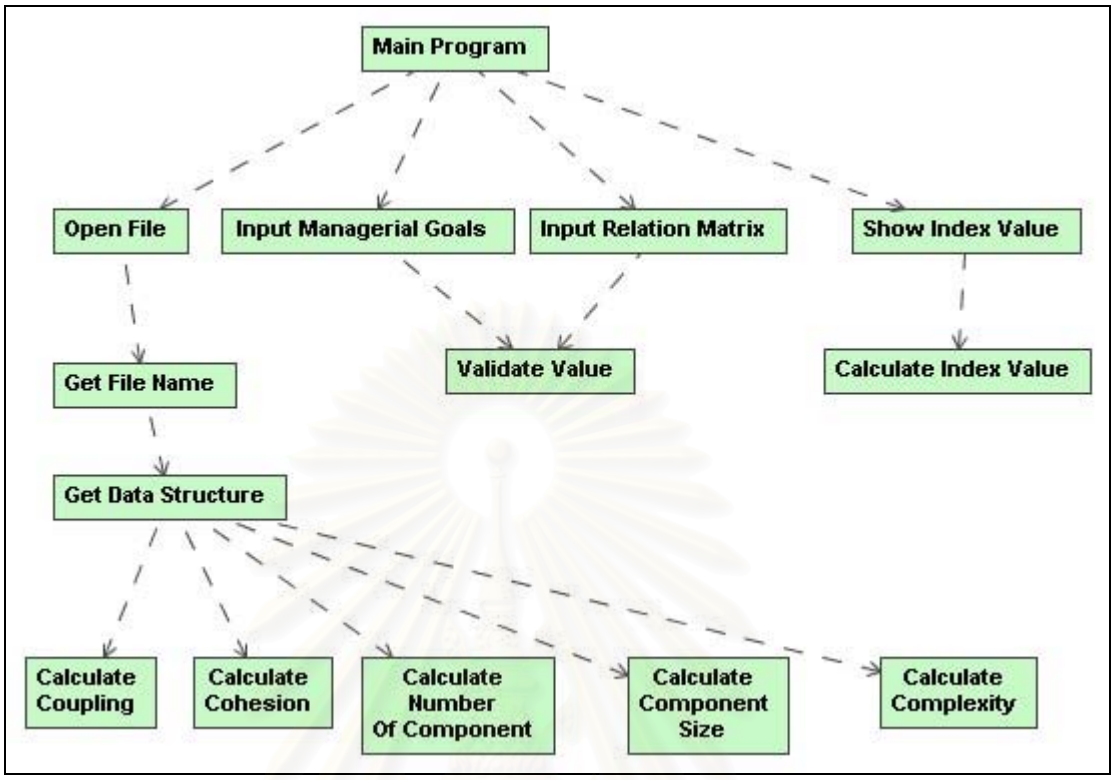
ชื่อส่วนย่อย	ActivityFinalNode
คำอธิบาย	เก็บข้อมูลของโหนดสุดท้าย
แอททริบิวต์	
xmi:id	รหัสเพื่อใช้ระบุตัวตนของส่วนย่อย
visibility	ประเภทของขอบเขตการมองเห็น โดยมีค่าได้ 3 แบบ ได้แก่ Private, Protected และ Public
activity	รหัสแอคทิวิตีที่โหนดเริ่มต้นนี้บรรจุอยู่
incoming	รหัสของสายงานควบคุมที่เข้าสู่โหนดนี้

ตารางที่ 5.8 รายละเอียดส่วนย่อย “ControlFlow”

ชื่อส่วนย่อย	ControlFlow
คำอธิบาย	เก็บข้อมูลของสายงานควบคุม
แอททริบิวต์	
xmi:id	รหัสเพื่อใช้ระบุตัวตนของส่วนย่อย
visibility	ประเภทของขอบเขตการมองเห็น โดยมีค่าได้ 3 แบบ ได้แก่ Private, Protected และ Public
activity	รหัสแอคทิวิตีที่สายงานควบคุมนี้บรรจุอยู่
source	รหัสเพื่อใช้ระบุตัวตนของต้นกำเนิดของสายงานนี้
target	รหัสเพื่อใช้ระบุตัวตนของปลายทางของสายงานนี้

5.2.5. การออกแบบโครงสร้างของเครื่องมือ

รูปที่ 5.6 แสดงโครงสร้างของเครื่องมือ โดยหน้าที่ของแต่ละส่วนแสดงอยู่ในตารางที่ 5.9 โดยลูกศรเส้นประหมายถึงส่วนย่อยนั้นถูกเรียกใช้งานโดยส่วนย่อยอื่นที่เป็นผู้เรียก และหัวลูกศรอยู่ที่ส่วนย่อยใดหมายถึงส่วนย่อยนั้นถูกเรียกใช้งานโดยส่วนย่อยที่ปลายลูกศร



รูปที่ 5.6โครงสร้างหลักของเครื่องมือ

ตารางที่ 5.9 หน้าที่ของแต่ละส่วนย่อยของเครื่องมือ

ส่วนของเครื่องมือ	หน้าที่
Main Program	เป็นหน้าแรกของเครื่องมือที่เป็นตัวหลักสู่การทำงานของทุกส่วนย่อย
Open File	ทำหน้าที่เปิดแฟ้มข้อมูล และตรวจสอบความถูกต้องของแฟ้มข้อมูลในด้านชนิดของแฟ้มข้อมูลที่ป้อนเข้า
Input Managerial Goals	ทำหน้าที่เป็นส่วนที่รับเป้าหมายด้านการจัดการโพรเซสจากนักออกแบบซอฟต์แวร์
Input Relation Matrix	ทำหน้าที่เป็นส่วนที่รับเมทริกซ์ความสัมพันธ์ระหว่างข้อมูลลักษณะทางเทคนิคและเป้าหมายด้านการจัดการโพรเซส
Show Index Value	ทำหน้าที่ตรวจสอบค่าดัชนีที่วัดได้จากการออกแบบและแสดงแก่ผู้ใช้
Get File Name	ทำหน้าที่ดึงแฟ้มข้อมูลและตรวจสอบความถูกต้องก่อนส่งต่อให้ส่วนที่ทำการสร้างข้อมูลเชิงโครงสร้าง

ตารางที่ 5.9 หน้าที่ของแต่ละส่วนย่อยของเครื่องมือ (ต่อ)

ส่วนของเครื่องมือ	หน้าที่
Get Data Structure	ทำหน้าที่สร้างข้อมูลเชิงโครงสร้างในรูปแบบต่าง ๆ เพื่อให้สามารถคำนวณค่ามาตรวัดลักษณะทางเทคนิคได้
Calculate Coupling	ทำหน้าที่ดึงข้อมูลเชิงโครงสร้างมาประมวลผลตามค่ามาตรวัดส่วนเชื่อมต่อระหว่างส่วนประกอบโพรเซส
Calculate Cohesion	ทำหน้าที่ดึงข้อมูลเชิงโครงสร้างมาประมวลผลตามค่ามาตรวัดส่วนเชื่อมติดภายในส่วนประกอบโพรเซส
Calculate Number Of Component	ทำหน้าที่ดึงข้อมูลเชิงโครงสร้างมาประมวลผลตามค่ามาตรวัดจำนวนส่วนประกอบโพรเซส
Calculate Component Size	ทำหน้าที่ดึงข้อมูลเชิงโครงสร้างมาประมวลผลตามค่ามาตรวัดขนาดของส่วนประกอบโพรเซส
Calculate Complexity	ทำหน้าที่ดึงข้อมูลเชิงโครงสร้างมาประมวลผลตามค่ามาตรวัดความซับซ้อนของส่วนประกอบโพรเซส
Validate Value	ทำหน้าที่ตรวจสอบข้อมูลที่นักออกแบบซอฟต์แวร์ป้อนเข้ามา
Calculate Index Value	ทำหน้าที่คำนวณค่าดัชนีการออกแบบก่อนแสดงผลให้กับนักออกแบบซอฟต์แวร์

บทที่ 6
การพัฒนาเครื่องมือวัดการออกแบบส่วนประกอบโพรเซส

ในการวัดการออกแบบตามเป้าหมายด้านการจัดการส่วนประกอบโพรเซสนั้น ถ้าปราศจากเครื่องมือสนับสนุนการวัด นักออกแบบซอฟต์แวร์จะต้องคำนวณค่ามาตรวัดด้วยมือจึงอาจทำให้เกิดข้อผิดพลาด หรือเกิดความคลาดเคลื่อนในการวัดซึ่งทำให้ไม่สะดวกที่จะนำแนวคิดในการวัดการออกแบบนี้ไปใช้ ดังนั้นผู้วิจัยจึงทำการสร้างเครื่องมือตามแนวคิดจากบทที่ 5 เพื่อให้เกิดความสะดวกและถูกต้องในการวัดค่ามาตรวัดที่ได้กำหนดไว้ โดยเครื่องมือที่ได้ทำการสร้างนั้นได้ใช้เครื่องมือในการพัฒนาหลายส่วนดังจะกล่าวในหัวข้อที่ 6.1 และข้อจำกัดของการพัฒนาจะกล่าวในหัวข้อที่ 6.2

6.1 เครื่องมือที่ใช้ทำการพัฒนา

ในการพัฒนาเครื่องมือวัดการออกแบบส่วนประกอบโพรเซสตามเป้าหมายด้านการจัดการส่วนประกอบนั้น ผู้วิจัยได้ใช้เครื่องมือในการพัฒนาดังนี้

1. จาวาเอสดีเค (Java SDK) รุ่น 1.4.2 เพื่อใช้เป็นตัวแปลโปรแกรม

2. อิคลิพซ์ (Eclipse) รุ่น 3.1 สำหรับเป็นเครื่องมือและสภาพแวดล้อมสำหรับการพัฒนาเครื่องมือ

3. จิ๊กกลูปลั๊กอินสำหรับอิคลิพซ์ (Jigloo plug-in for Eclipse) รุ่น 3.5.2 เป็นส่วนเสริมซอฟต์แวร์ที่ช่วยในการออกแบบและพัฒนาแบบส่วนต่อประสานกับผู้ใช้

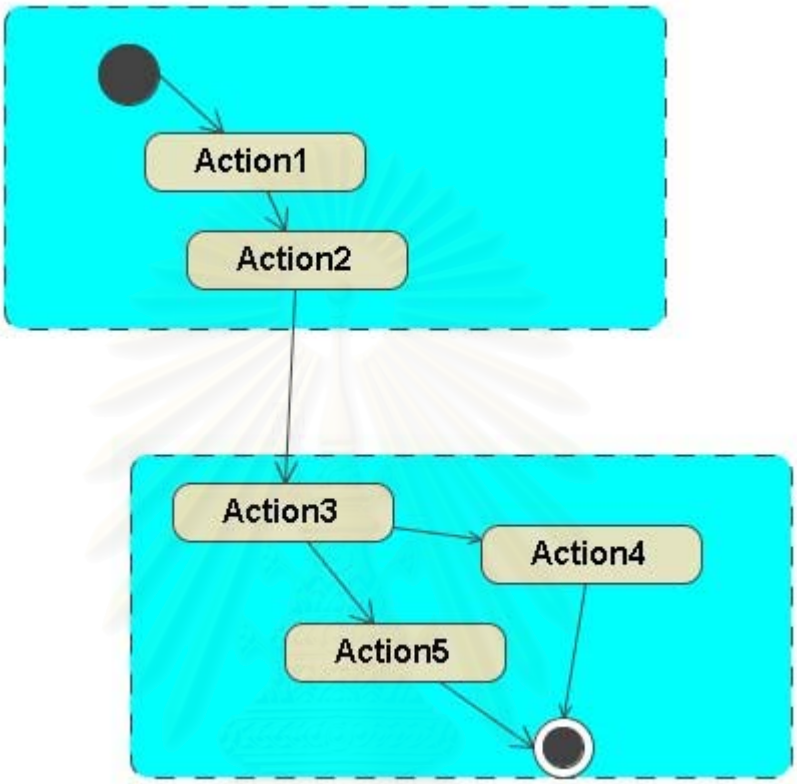
4. เมจิกดรอว์ (Magic Draw) รุ่น 10 เพื่อใช้เป็นตัวออกแบบยูเอ็มแอลรุ่น 2.0 และส่งออกแฟ้มข้อมูลเอ็กซ์เอ็มไอรุ่น 2.1

6.2 ข้อจำกัดของการพัฒนา

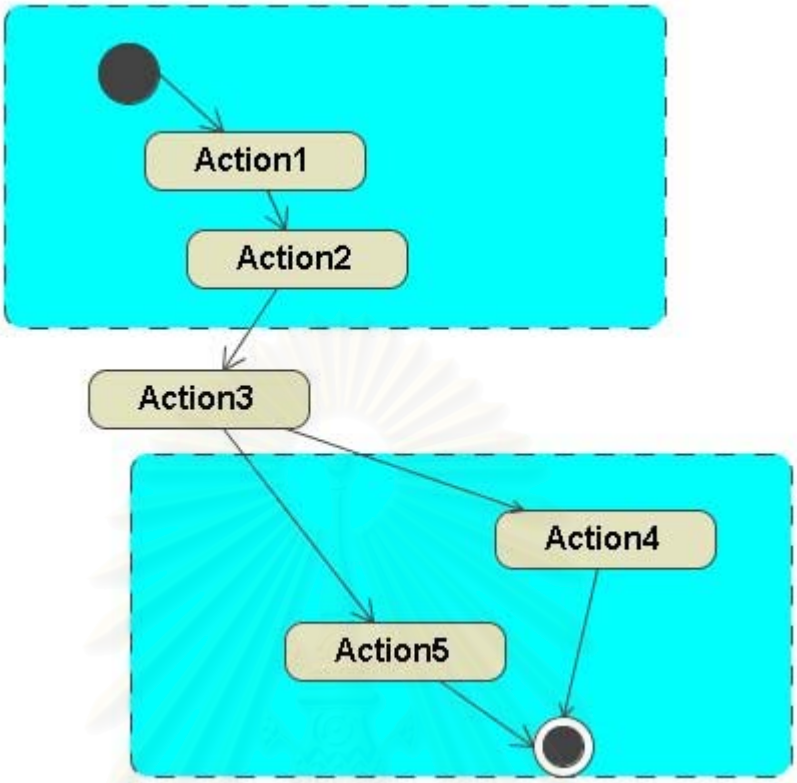
1. เนื่องจากรูปแบบของแฟ้มข้อมูลเอ็กซ์เอ็มไอที่แต่ละบริษัทผู้ผลิตเครื่องมือวาดแผนภาพยูเอ็มแอลใช้ในการส่งออก (Export) ข้อมูลแผนภาพนั้นมีความหลากหลาย ผู้วิจัยได้เลือกรูปแบบของแฟ้มข้อมูลเอ็กซ์เอ็มไอที่ส่งออกมาจากเครื่องมือเมจิกดรอว์ รุ่น10 เนื่องจากสามารถส่งออกแฟ้มข้อมูลเอ็กซ์เอ็มไอรุ่น 2.1 ได้ ซึ่งเครื่องมือนี้มีความสามารถในการส่งออกแฟ้มข้อมูลเอ็กซ์เอ็มไอของแผนภาพแอคทิวิตีที่มีการระบุแอคชันอยู่ภายในแอคทิวิตีย่อยได้

2. แต่ละแอคชันในแผนภาพแอคทิวิตีจะต้องได้รับการบรรจุอยู่ภายในหนึ่งแอคทิวิตีย่อย ตัวอย่างเช่นรูปที่ 6.1 ทั้งนี้เพื่อให้เครื่องมือสามารถคำนวณค่าลักษณะทางเทคนิคตามมาตรวัดได้ มิ

เช่นนั้นจะเกิดข้อผิดพลาดขณะทำการประมวลผลหากไม่ได้จับกลุ่มแอคชันให้อยู่ภายในแอคทิวิตีย่อยทั้งหมด (เช่นรูปที่ 6.2)



รูปที่ 6.1 แอคชันบรรจุภายในแอคทิวิตีย่อย



รูปที่ 6.2 แอคชันที่ไม่ได้บรรจุในแอคทิวิตีย่อยทั้งหมด

จากรูปที่ 6.2 แอคชัน 3 ไม่ได้ถูกบรรจุอยู่ภายในแอคทิวิตีย่อยดังนั้นเครื่องมือจะไม่สามารถทราบได้ว่าแผนภาพแอคทิวิตีนี้แบ่งกลุ่มของส่วนประกอบโพรเซสอย่างไรทำให้เกิดข้อผิดพลาดขณะคำนวณค่ามาตรวัด

บทที่ 7
การทดสอบเครื่องมือด้วยกรณีศึกษา

ในบทนี้ผู้วิจัยใช้ตัวอย่างจากบทที่ 4 ในการทดสอบเครื่องมือวัดการออกแบบส่วนประกอบโพรเซส โดยตัวอย่างที่นำมาทดสอบนี้คือ ระบบตอบรับโทรศัพท์ของตำรวจ และระบบตอบรับโทรศัพท์ของสถานีดับเพลิง ซึ่งทั้งสองระบบที่เลือกมาทดสอบนี้ได้ดัดแปลงจาก [29] โดยจะมีการจัดกลุ่มแอคชันให้เป็นส่วนประกอบโพรเซสในรูปแบบต่าง ๆ และวัดค่าดัชนีการบรรลุเป้าหมายด้านการจัดการส่วนประกอบ โดยแต่ละตัวอย่างนั้นจะเน้นเป้าหมายด้านการจัดการที่ต่างกันและนำผลที่ได้มาสรุปผลการทดสอบ

7.1 ระบบตอบรับโทรศัพท์ของตำรวจ

ในการทดสอบเครื่องมือโดยใช้กรณีศึกษาของระบบตอบรับโทรศัพท์ของตำรวจนี้จะแบ่งหัวข้อออกเป็น การกำหนดค่าเป้าหมายด้านการจัดการเพื่อทำการคำนวณค่าดัชนี การกำหนดเมทริกซ์ของค่าความสัมพันธ์ระหว่างเป้าหมายด้านการจัดการส่วนประกอบและลักษณะทางเทคนิค การจัดกลุ่มของแอคชันเป็นส่วนประกอบโพรเซส และการสรุปผลสำหรับกรณีศึกษา

7.1.1. การกำหนดค่าเป้าหมายด้านการจัดการ

ในหัวข้อนี้จะทำการกำหนดเป้าหมายด้านการจัดการเพื่อทดสอบเครื่องมือ โดยในกรณีศึกษานี้ผู้วิจัยจะกำหนดเป้าหมายด้านการจัดการโพรเซสเป็นสองแบบเพื่อเปรียบเทียบ โดยเป้าหมายด้านการจัดการแบบแรกจะแสดงในตารางที่ 7.1 ส่วนเป้าหมายด้านการจัดการแบบที่สองจะแสดงในตารางที่ 7.2

ตารางที่ 7.1 เป้าหมายด้านการจัดการส่วนประกอบโพรเซสแบบที่หนึ่ง

เป้าหมายด้านการจัดการส่วนประกอบ	น้ำหนัก
การใช้ต้นทุนอย่างมีประสิทธิผล	0.05
ความง่ายในการประกอบ	0.05
ความสามารถในการปรับแต่ง	0.05
ความสามารถในการนำกลับมาใช้	0.8
สภาพบำรุงรักษาได้	0.05

ตารางที่ 7.2 เป้าหมายด้านการจัดการส่วนประกอบโพรเซสแบบที่สอง

เป้าหมายด้านการจัดการส่วนประกอบ	น้ำหนัก
การใช้ต้นทุนอย่างมีประสิทธิผล	0.05
ความง่ายในการประกอบ	0.05
ความสามารถในการปรับแต่ง	0.05
ความสามารถในการนำกลับมาใช้	0.05
สภาพบำรุงรักษาได้	0.8

จากตารางที่ 7.1 เป็นการกำหนดเป้าหมายด้านการจัดการโดยเน้นที่ความสามารถในการนำกลับมาใช้ใหม่และส่วนที่เหลือจะเน้นความสำคัญเท่า ๆ กัน ส่วนตารางที่ 7.2 เป็นการกำหนดเป้าหมายด้านการจัดการโดยเน้นที่สภาพบำรุงรักษาได้และส่วนที่เหลือจะเน้นความสำคัญเท่า ๆ กัน หลักเกณฑ์ในการกำหนดค่าได้กล่าวไว้ในหัวข้อที่ 3.4.6

7.1.2. การกำหนดเมทริกซ์ของค่าความสัมพันธ์ระหว่างเป้าหมายด้านการจัดการส่วนประกอบและลักษณะทางเทคนิค

จากตารางที่ 3.2 ได้กำหนดแนวทางสำหรับการกำหนดค่าความสัมพันธ์ระหว่างเป้าหมายด้านการจัดการส่วนประกอบและลักษณะทางเทคนิค ในส่วนนี้ได้กำหนดค่าสำหรับกรณีทดสอบไว้ดังตารางที่ 7.3 โดยค่าในตารางได้มาจาก [17] ซึ่งได้ทำการสำรวจจากกลุ่มผู้เชี่ยวชาญและมีประสบการณ์ในการออกแบบพัฒนาแอพพลิเคชัน โดยให้ระบุความสัมพันธ์ระหว่างเป้าหมายด้านการจัดการส่วนประกอบและลักษณะทางเทคนิค เมทริกซ์นี้สามารถใช้เป็นค่าโดยปริยาย (Default) ในเครื่องมือที่ทำการพัฒนา แต่ก็สามารถเปลี่ยนค่าได้ตามแต่นักออกแบบซอฟต์แวร์จะเห็นว่าเหมาะสม

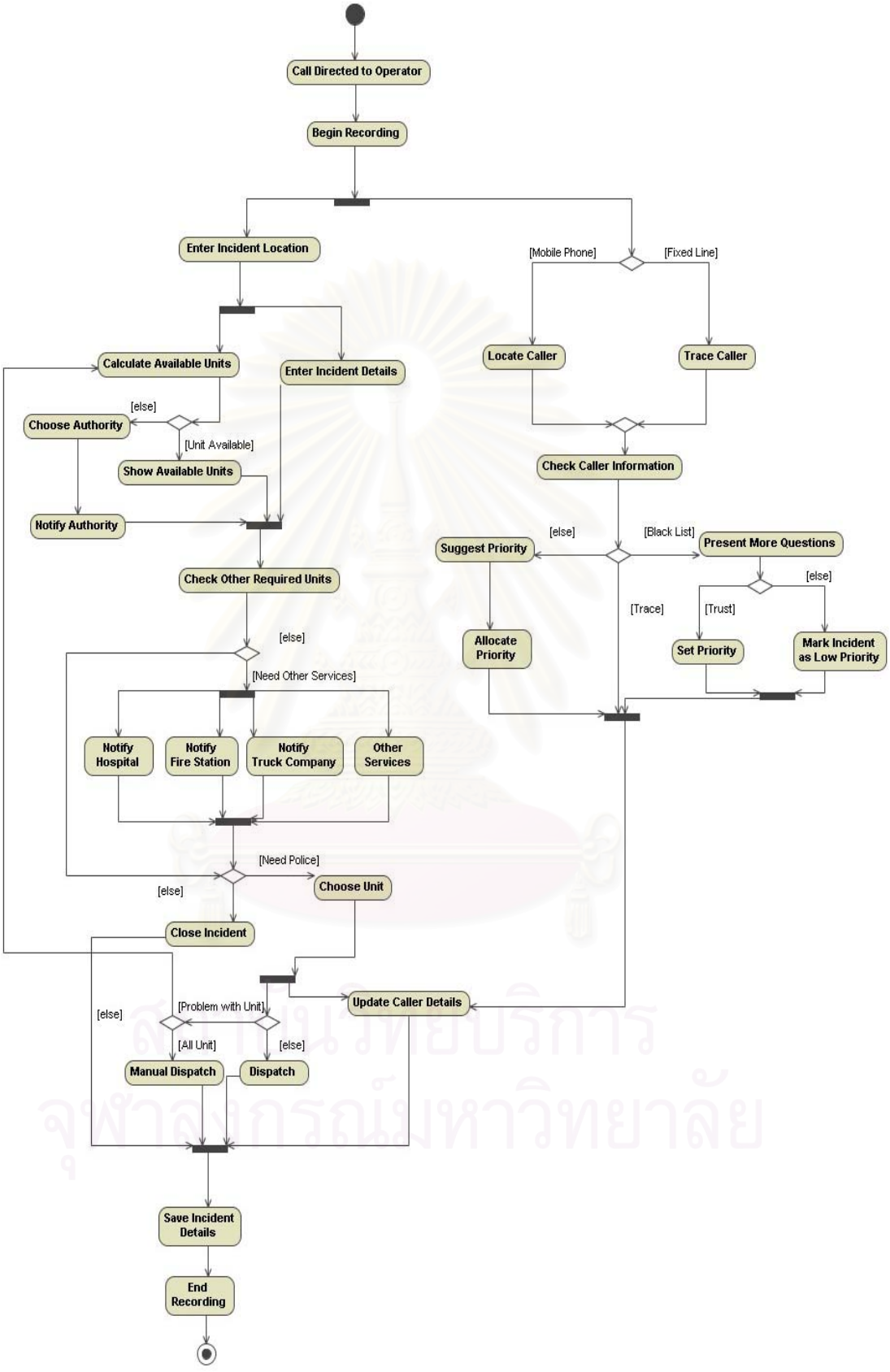
7.1.3. การออกแบบส่วนประกอบโพรเซส

รูปที่ 7.1 แสดงแบบจำลองกระบวนการทางธุรกิจของระบบตอบรับโทรศัพท์ของตำรวจ ซึ่งผู้วิจัยจะทำการออกแบบส่วนประกอบโพรเซสโดยจัดกลุ่มแอคทิวิตีย่อย ซึ่งในที่นี้จะทำการจัดกลุ่มตามเกณฑ์ทางหน้าที่ทางธุรกิจ (หัวข้อที่ 2.2.2) ในส่วนนี้ผู้วิจัยได้ทำการจัดกลุ่มออกเป็นสี่แบบ โดยแบบแรกนั้นจะแบ่งออกเป็นหนึ่งส่วนประกอบโพรเซสดังรูปที่ 7.2 แบบที่สองนั้นจะแบ่งออกเป็นสองส่วนประกอบโพรเซสดังรูปที่ 7.3 แบบที่สามนั้นจะแบ่งออกเป็นสามส่วนประกอบ

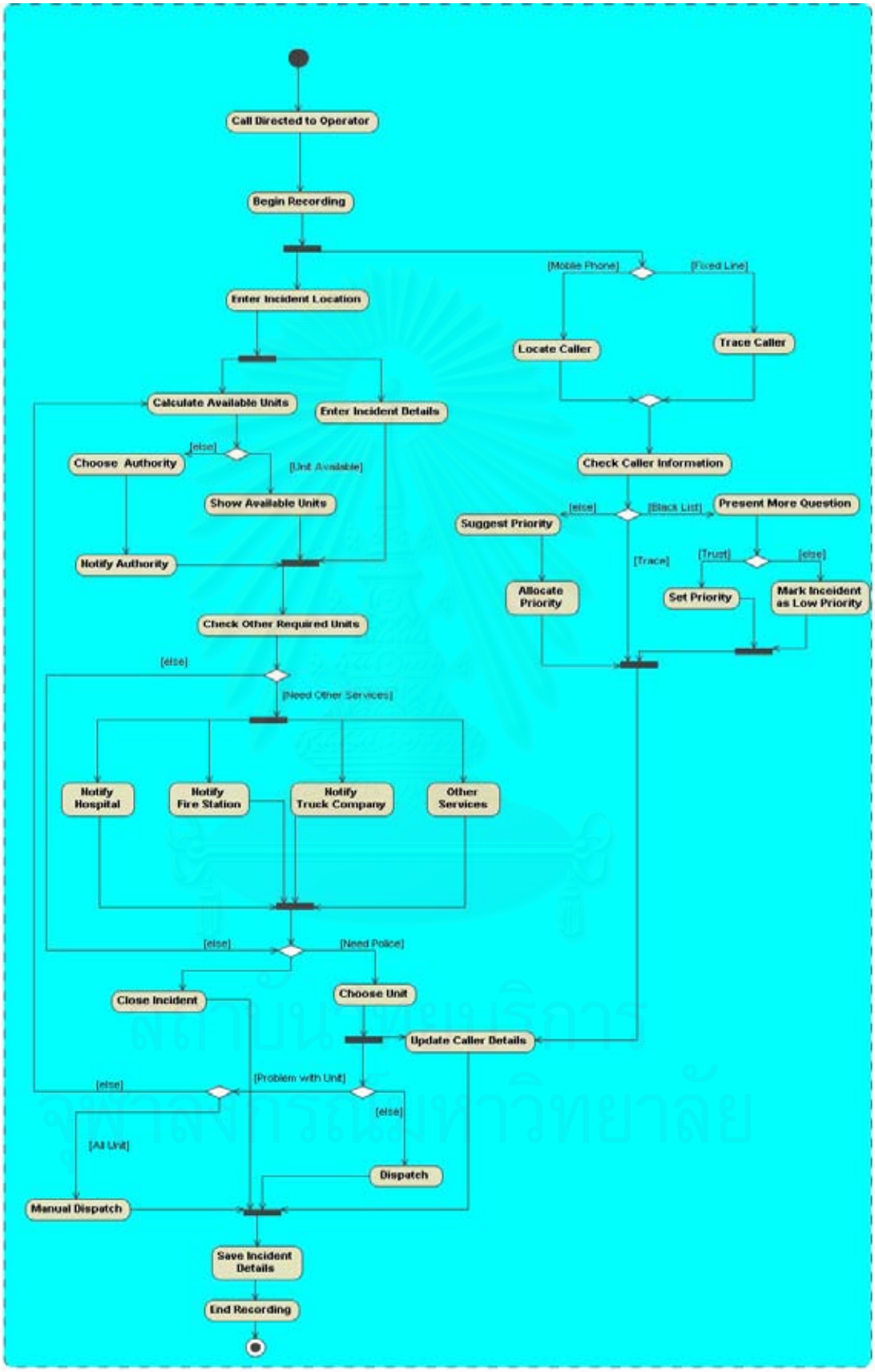
โพรเซสดังรูปที่ 7.4 ส่วนแบบสุดท้ายแบ่งออกเท่ากับจำนวนของแอคชันคือ ยี่สิบแปดส่วนประกอบโพรเซส จากนั้นได้นำทั้งสี่การออกแบบไปคำนวณค่ามาตรวัดโดยใช้เครื่องมือ

ตารางที่ 7.3 เมทริกซ์ของค่าความสัมพันธ์ระหว่างเป้าหมายด้านการจัดการส่วนประกอบ และลักษณะทางเทคนิคที่กำหนดขึ้นเพื่อทดสอบเครื่องมือ

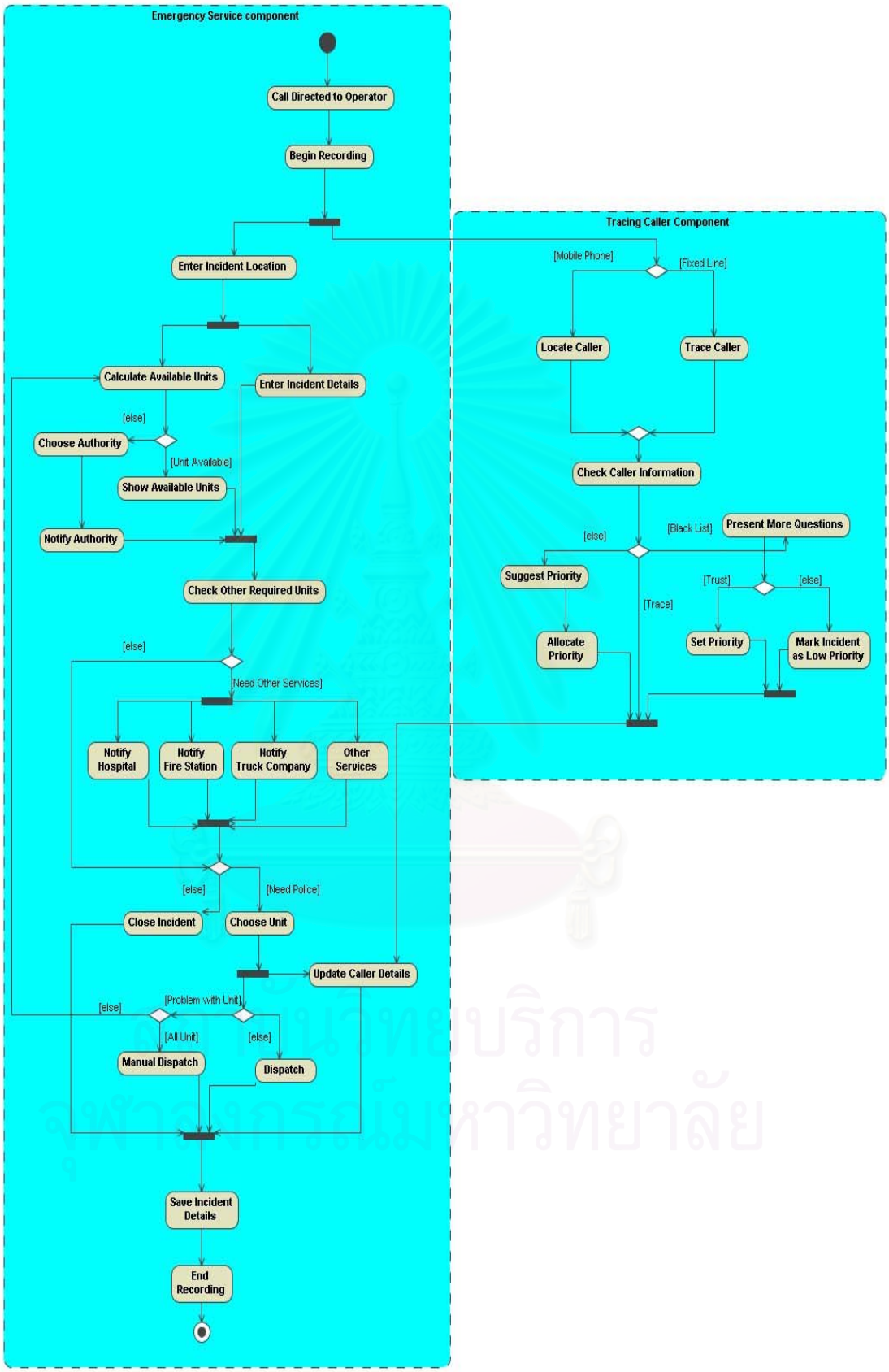
	การเชื่อมต่อกันระหว่างส่วนประกอบ	การเชื่อมติดกันภายในส่วนประกอบ	จำนวนของส่วนประกอบ	ขนาดของส่วนประกอบ	ความซับซ้อนของการออกแบบ
การใช้ต้นทุนอย่างมีประสิทธิผล	0	-8	-6	-8	-7
ความง่ายในการประกอบ	-5	0	0	8	0
ความสามารถในการปรับแต่ง	-6	5	5	-5	0
ความสามารถในการนำกลับมาใช้	-8	8	6	-7	0
สภาพบำรุงรักษาได้	-7	-6	-5	-6	-8



รูปที่ 7.1 ระบบตอบรับโทรศัพท์ของตำรวจก่อนการจัดกลุ่ม



รูปที่ 7.2 ระบบตอบรับโทรศัพท์ของตำรวจที่แบ่งออกเป็นหนึ่งส่วนประกอบโพรเซส



รูปที่ 7.3 ระบบตอบรับโทรศัพท์ของตำรวจที่แบ่งออกเป็นสองส่วนประกอบโพรเซส

รูปที่ 7.4 ระบบตอบรับโทรศัพท์ของตำรวจที่แบ่งออกเป็นสามส่วนประกอบโพรเซส

รูปที่ 7.5 ระบบตอบรับโทรศัพท์ของตำรวจที่แบ่งออกเป็นยี่สิบแปดส่วนประกอบโพรเซส

7.1.4. การสรุปผล

เมื่อนำการออกแบบส่วนประกอบโพรเซสที่สร้างขึ้นไปคำนวณลักษณะทางเทคนิคแล้วจะได้ค่าดังตารางที่ 7.4 สำหรับการออกแบบเป็นหนึ่งส่วนประกอบโพรเซส ตารางที่ 7.5 สำหรับการออกแบบเป็นสองส่วนประกอบโพรเซส ตารางที่ 7.6 สำหรับการออกแบบเป็นสามส่วนประกอบโพรเซส และตารางที่ 7.7 สำหรับการออกแบบเป็น 28 ส่วนประกอบโพรเซส

ตารางที่ 7.4 ค่าลักษณะทางเทคนิคของระบบตอบรับโทรศัพท์ของตำรวจเมื่อแบ่งออกเป็นหนึ่งส่วนประกอบโพรเซส

ลักษณะทางเทคนิค	ค่ามาตรวัด
การเชื่อมต่อกันระหว่างส่วนประกอบโพรเซส	0
การเชื่อมติดกันภายในส่วนประกอบโพรเซส	46
จำนวนของส่วนประกอบโพรเซส	1
ขนาดของส่วนประกอบโพรเซส	28
ความซับซ้อนของการออกแบบ	784

ตารางที่ 7.5 ค่าลักษณะทางเทคนิคของระบบตอบรับโทรศัพท์ของตำรวจเมื่อแบ่งออกเป็นสองส่วนประกอบโพรเซส

ลักษณะทางเทคนิค	ค่ามาตรวัด
การเชื่อมต่อกันระหว่างส่วนประกอบโพรเซส	6
การเชื่อมติดกันภายในส่วนประกอบโพรเซส	40
จำนวนของส่วนประกอบโพรเซส	2
ขนาดของส่วนประกอบโพรเซส	15.23
ความซับซ้อนของการออกแบบ	464

ตารางที่ 7.6 ค่าลักษณะทางเทคนิคของระบบตอบรับโทรศัพท์ของตำรวจเมื่อแบ่งออกเป็นสามส่วนประกอบโพรเซส

ลักษณะทางเทคนิค	ค่ามาตรวัด
การเชื่อมต่อกันระหว่างส่วนประกอบโพรเซส	10
การเชื่อมติดกันภายในส่วนประกอบโพรเซส	36
จำนวนของส่วนประกอบโพรเซส	3
ขนาดของส่วนประกอบโพรเซส	9.52
ความซับซ้อนของการออกแบบ	272

ตารางที่ 7.7 ค่าลักษณะทางเทคนิคของระบบตอบรับโทรศัพท์ของตำรวจเมื่อแบ่งออกเป็น 28 ส่วนประกอบโพรเซส

ลักษณะทางเทคนิค	ค่ามาตรวัด
การเชื่อมต่อกันระหว่างส่วนประกอบโพรเซส	46
การเชื่อมติดกันภายในส่วนประกอบโพรเซส	0
จำนวนของส่วนประกอบโพรเซส	28
ขนาดของส่วนประกอบโพรเซส	1
ความซับซ้อนของการออกแบบ	28

เมื่อได้ค่ามาตรวัดลักษณะทางเทคนิคสำหรับแต่ละการออกแบบส่วนประกอบโพรเซสแล้ว เครื่องมือจะรับค่าเป้าหมายด้านการจัดการส่วนประกอบและเมทริกซ์ความสัมพันธ์ระหว่างเป้าหมายด้านการจัดการส่วนประกอบและค่าลักษณะทางเทคนิคดังในตารางที่ 7.1, 7.2 และ 7.3 มาจากผู้วิจัย จากนั้นเครื่องมือจะคำนวณค่าดัชนีการบรรลุเป้าหมายด้านการจัดการส่วนประกอบซึ่งได้ผลดังตารางที่ 7.8

ตารางที่ 7.8 ค่าดัชนีการบรรลุเป้าหมายด้านการจัดการของระบบตอบรับโทรศัพท์ของตำรวจ

	เป้าหมายด้านการจัดการแบบที่หนึ่ง	เป้าหมายด้านการจัดการแบบที่สอง
หนึ่งส่วนประกอบ	-482	-5656.25
สองส่วนประกอบ	-238.47	-3443.05
สามส่วนประกอบ	-107.9	-2127.97
ยี่สิบแปดส่วนประกอบ	-236.95	-600.7

จากตารางที่ 7.8 เมื่อกำหนดเป้าหมายแบบที่หนึ่งโดยเน้นที่การนำกลับมาใช้ใหม่และแบ่งส่วนประกอบโพรเซสออกเป็นหนึ่งส่วนประกอบ สองส่วนประกอบ สามส่วนประกอบ และยี่สิบแปดส่วนประกอบนั้น จะได้ว่าการแบ่งออกเป็นสามส่วนประกอบจะได้ค่าดัชนีมากที่สุดและบรรลุเป้าหมายด้านการจัดการซึ่งเน้นการนำกลับมาใช้ใหม่ได้ดีกว่าเมื่อแบ่งออกเป็นแบบอื่น ๆ เนื่องจากเมื่อแบ่งออกเป็นหลายส่วนประกอบนั้นจะสามารถนำส่วนประกอบโพรเซสที่แบ่งไว้ไปใช้ในโดเมนงานอื่นได้หลากหลายมากกว่า แต่อย่างไรก็ตามเมื่อแบ่งจำนวนส่วนประกอบมากเท่ากับจำนวนแอคชันก็จะทำให้ได้ค่าดัชนีที่ด้อยกว่า

สำหรับเป้าหมายด้านการจัดการแบบที่สองซึ่งเน้นที่สภาพบำรุงรักษาได้ เมื่อแบ่งส่วนประกอบโพรเซสออกเป็นยี่สิบแปดส่วนประกอบจะได้ค่าดัชนีที่มากที่สุด สาเหตุที่ค่าดัชนีในกรณีนี้ติดลบมากเพราะว่าลักษณะทางเทคนิคของค่าความซับซ้อนมีค่ามากกว่าลักษณะทางเทคนิคอื่น ๆ มาก นั่นคือ เมื่อแบ่งออกเป็นหนึ่งส่วนประกอบโพรเซสนั้นจะมีความซับซ้อนเท่ากับ 784 ถ้าแบ่งออกเป็นสองสวนประกอบจะได้ค่าความซับซ้อนเท่ากับ 464 เมื่อแบ่งออกเป็นสามส่วนประกอบจะได้ค่าความซับซ้อนเท่ากับ 272 และถ้าแบ่งออกเป็นยี่สิบแปดส่วนประกอบจะมีความซับซ้อนเท่ากับ 28 อีกทั้งเมทริกซ์ค่าเป้าหมายด้านการจัดการนั้นเน้นที่ค่าสภาพบำรุงรักษาได้และค่าน้ำหนักของเมทริกซ์แสดงความสัมพันธ์ก็เน้นที่สภาพบำรุงรักษาได้เช่นกัน ดังนั้นค่าที่ได้ออกมาจึงมีค่าติดลบมาก จากค่าดัชนีที่ได้เมื่อกำหนดเป้าหมายด้านการจัดการที่เน้นสภาพบำรุงรักษาได้ เมื่อแบ่งจำนวนส่วนประกอบโพรเซสยิ่งมากจะได้ค่าดัชนีที่สูงขึ้น

7.2 ระบบตอบรับโทรศัพท์ของสถานีดับเพลิง

เช่นเดียวกับหัวข้อที่ 7.1 การทดสอบเครื่องมือโดยใช้กรณีศึกษาของระบบตอบรับโทรศัพท์ของสถานีดับเพลิง จะแบ่งออกเป็น การกำหนดค่าเป้าหมายด้านการจัดการ การกำหนดเมทริกซ์ของค่าความสัมพันธ์ระหว่างเป้าหมายด้านการจัดการส่วนประกอบและลักษณะทางเทคนิค การจัดกลุ่มของแอคชันเป็นส่วนประกอบโพรเซส และการสรุปผลสำหรับกรณีศึกษานี้

7.2.1. การกำหนดเป้าหมายด้านการจัดการ

ในกรณีศึกษานี้ได้ใช้เป้าหมายด้านการจัดการเดียวกับหัวข้อที่ 7.1 โดยใช้ค่าจากตารางที่ 7.1 และ ตารางที่ 7.2 ซึ่งได้กำหนดให้เน้นที่การนำกลับมาใช้ใหม่ และสภาพบำรุงรักษาได้

7.2.2. การกำหนดเมทริกซ์ของค่าความสัมพันธ์ระหว่างเป้าหมายด้านการจัดการ ส่วนประกอบและลักษณะทางเทคนิค

การกำหนดเมทริกซ์นี้ใช้ตามหัวข้อที่ 7.1 ดังค่าในตารางที่ 7.3 เช่นกัน

7.2.3. การออกแบบส่วนประกอบโพรเซส

รูปที่ 7.6 แสดงแบบจำลองกระบวนการทางธุรกิจของระบบตอบรับโทรศัพท์ของสถานีดับเพลิงซึ่งผู้วิจัยจะทำการออกแบบส่วนประกอบโพรเซสโดยจัดกลุ่มแอคทิวิตีย่อย ซึ่งในที่นี้จะทำการจัดกลุ่มตามเกณฑ์หน้าที่ทางธุรกิจ (หัวข้อที่ 2.2.2) ในส่วนนี้ผู้วิจัยได้ทำการจัดกลุ่มออกเป็นสี่แบบโดยแบบแรกจะแบ่งออกเป็นหนึ่งส่วนประกอบโพรเซสดังรูปที่ 7.7 แบบที่สองจะแบ่งออกเป็นสองส่วนประกอบดังรูปที่ 7.8 แบบที่สามจะแบ่งออกเป็นสามส่วนประกอบโพรเซสดังรูปที่ 7.9 ส่วนแบบสุดท้ายจะแบ่งออกเท่ากับจำนวนแอคชันของโดเมนงานคือยี่สิบเอ็ดส่วนประกอบโพรเซส

7.2.4. การสรุปผล

เมื่อนำการออกแบบส่วนประกอบโพรเซสไปคำนวณลักษณะทางเทคนิคแล้วจะได้ค่าดังตารางที่ 7.9 สำหรับการออกแบบเป็นหนึ่งส่วนประกอบ ดังตารางที่ 7.10 สำหรับการออกแบบเป็นสองส่วนประกอบ ดังตารางที่ 7.11 สำหรับการออกแบบเป็นสามส่วนประกอบ และดังตารางที่ 7.12 สำหรับการออกแบบเป็นยี่สิบเอ็ดส่วนประกอบ

ตารางที่ 7.9 ค่าลักษณะทางเทคนิคของระบบตอบรับโทรศัพท์ของสถานีดับเพลิงเมื่อแบ่งออกเป็นหนึ่งส่วนประกอบโพรเซส

ลักษณะทางเทคนิค	ค่ามาตรวัด
การเชื่อมต่อกันระหว่างส่วนประกอบโพรเซส	0
การเชื่อมติดกันภายในส่วนประกอบโพรเซส	34
จำนวนของส่วนประกอบโพรเซส	1
ขนาดของส่วนประกอบโพรเซส	21
ความซับซ้อนของการออกแบบ	441

ตารางที่ 7.10 ค่าลักษณะทางเทคนิคของระบบตอบรับโทรศัพท์ของสถานีดับเพลิงเมื่อแบ่งออกเป็นสองส่วนประกอบโพรเซส

ลักษณะทางเทคนิค	ค่ามาตรวัด
การเชื่อมต่อกันระหว่างส่วนประกอบโพรเซส	2
การเชื่อมติดกันภายในส่วนประกอบโพรเซส	32
จำนวนของส่วนประกอบโพรเซส	2
ขนาดของส่วนประกอบโพรเซส	11.42
ความซับซ้อนของการออกแบบ	261

ตารางที่ 7.11 ค่าลักษณะทางเทคนิคของระบบตอบรับโทรศัพท์ของสถานีดับเพลิงเมื่อแบ่งออกเป็นสามส่วนประกอบโพรเซส

ลักษณะทางเทคนิค	ค่ามาตรวัด
การเชื่อมต่อกันระหว่างส่วนประกอบโพรเซส	10
การเชื่อมติดกันภายในส่วนประกอบโพรเซส	24
จำนวนของส่วนประกอบโพรเซส	3
ขนาดของส่วนประกอบโพรเซส	7.33
ความซับซ้อนของการออกแบบ	161

ตารางที่ 7.12 ค่าลักษณะทางเทคนิคของระบบตอบรับโทรศัพท์ของสถานีดับเพลิงเมื่อแบ่งออกเป็นยี่สิบเอ็ดส่วนประกอบโพรเซส

ลักษณะทางเทคนิค	ค่ามาตรวัด
การเชื่อมต่อกันระหว่างส่วนประกอบโพรเซส	34
การเชื่อมติดกันภายในส่วนประกอบโพรเซส	0
จำนวนของส่วนประกอบโพรเซส	21
ขนาดของส่วนประกอบโพรเซส	1
ความซับซ้อนของการออกแบบ	21

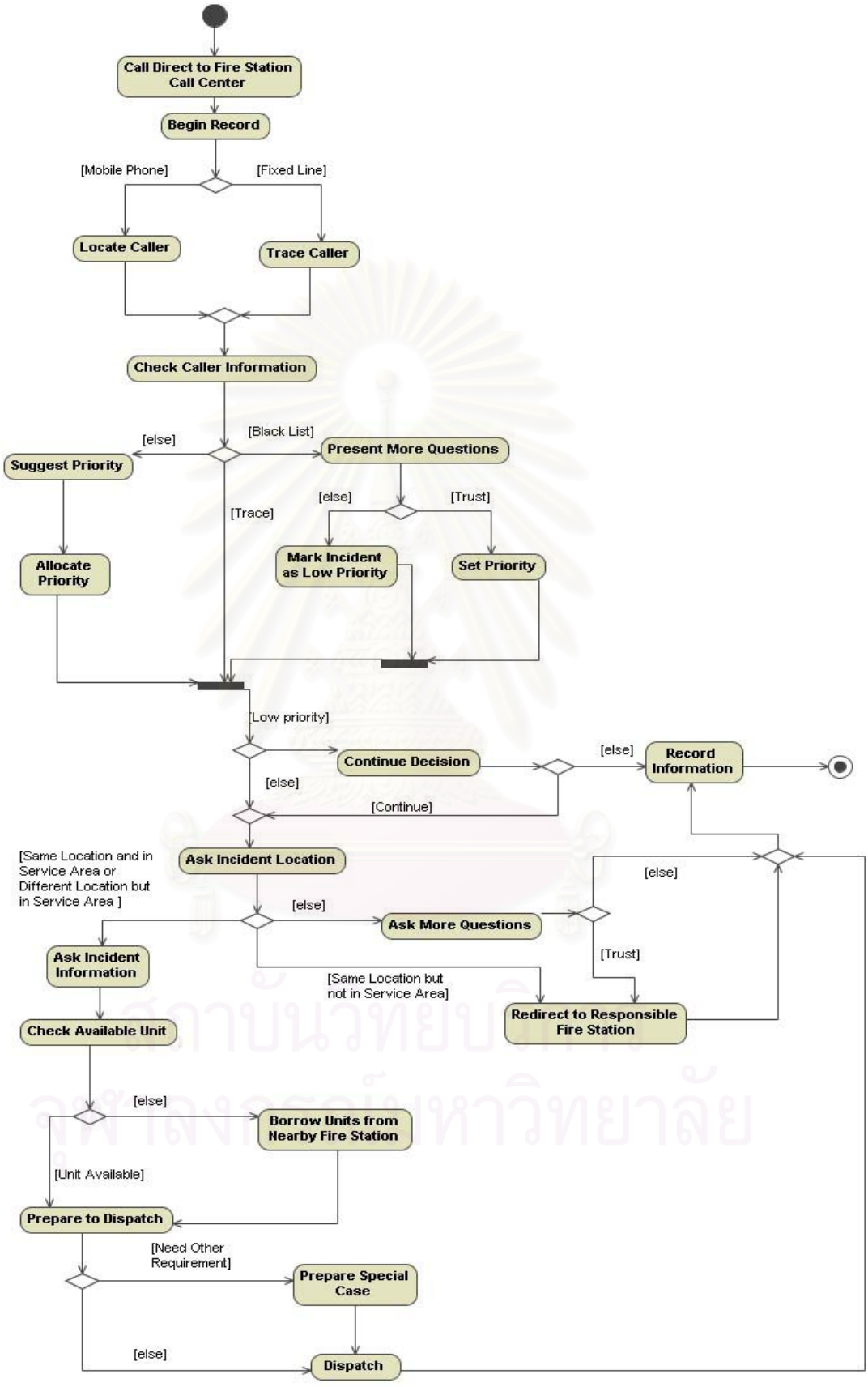
เมื่อได้ค่ามาตรวัดลักษณะทางเทคนิคสำหรับแต่ละการออกแบบส่วนประกอบโพรเซสแล้ว เครื่องมือจะรับค่าเป้าหมายด้านการจัดการและเมทริกซ์ความสัมพันธ์ระหว่างเป้าหมายด้านการจัดการส่วนประกอบโพรเซสและค่าลักษณะทางเทคนิคดังในตารางที่ 7.1, 7.2 และ 7.3 มาจากผู้วิจัย จากนั้นเครื่องมือจะคำนวณค่าดัชนีการบรรลุเป้าหมายด้านการจัดการส่วนประกอบซึ่งได้ผลดังตารางที่ 7.13

ตารางที่ 7.13 ค่าดัชนีการบรรลุเป้าหมายด้านการจัดการของระบบตอบรับโทรศัพท์ของสถานีดับเพลิง

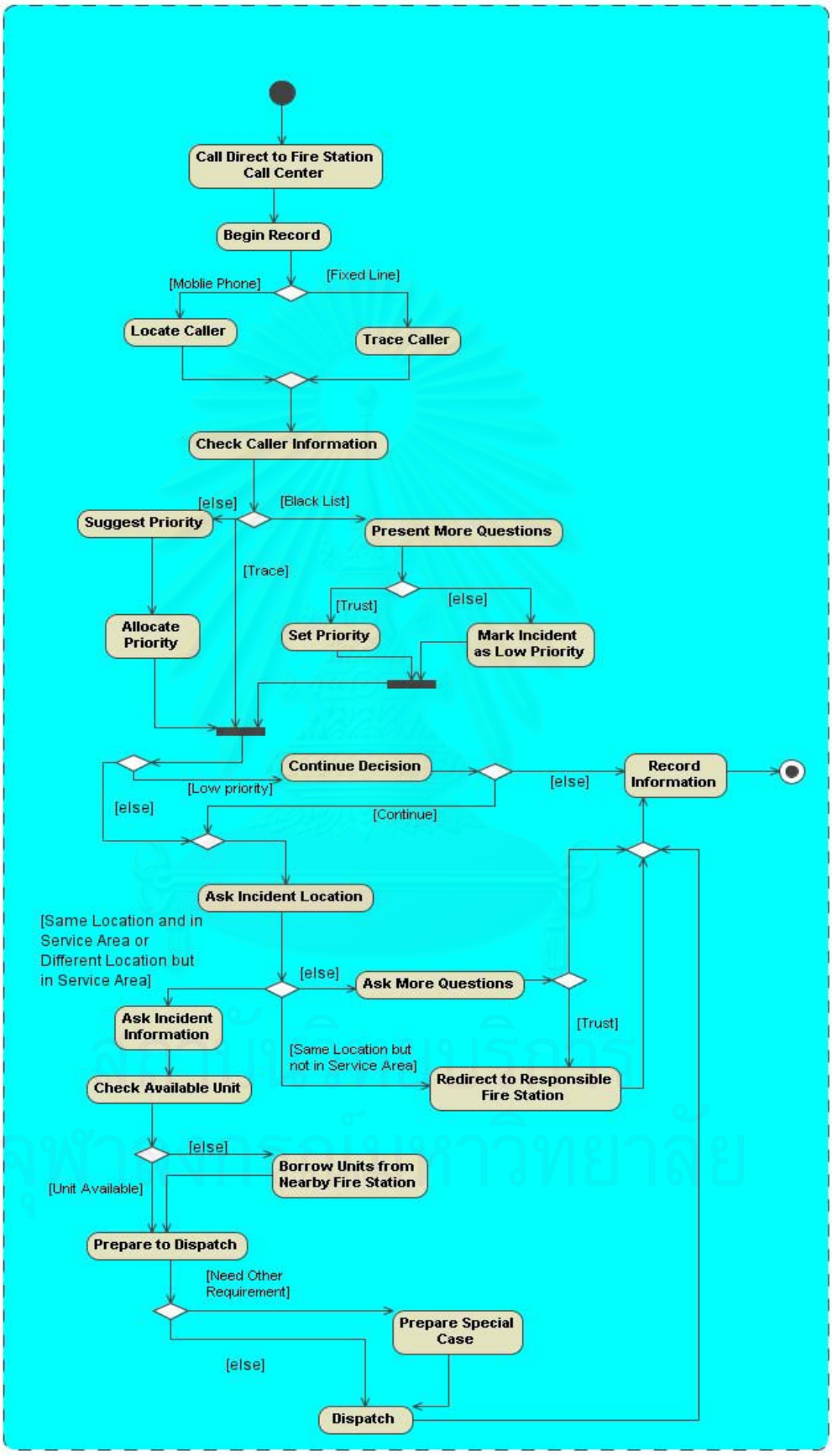
	เป้าหมายด้านการจัดการแบบที่หนึ่ง	เป้าหมายด้านการจัดการแบบที่สอง
หนึ่งส่วนประกอบ	-253.1	-3248.6
สองส่วนประกอบ	-81.21	-1989.64
สามส่วนประกอบ	-82.5	-1312.26
ยี่สิบเอ็ดส่วนประกอบ	-175.6	-448.6

จากตารางที่ 7.13 สามารถสรุปได้ว่าเมื่อกำหนดเป้าหมายด้านการจัดการที่เน้นการนำกลับมาใช้ใหม่ การแบ่งออกเป็นสองส่วนประกอบโพรเซสจะได้ค่าดัชนีมากกว่าเมื่อแบ่งออกเป็นหนึ่งส่วนประกอบ สามส่วนประกอบ และยี่สิบเอ็ดส่วนประกอบ โดยค่าที่ได้เมื่อแบ่งออกเป็นสองส่วนประกอบจะบรรลุเป้าหมายด้านการนำกลับมาใช้ใหม่ได้ดีกว่าแบ่งออกเป็นสามส่วนประกอบเล็กน้อย

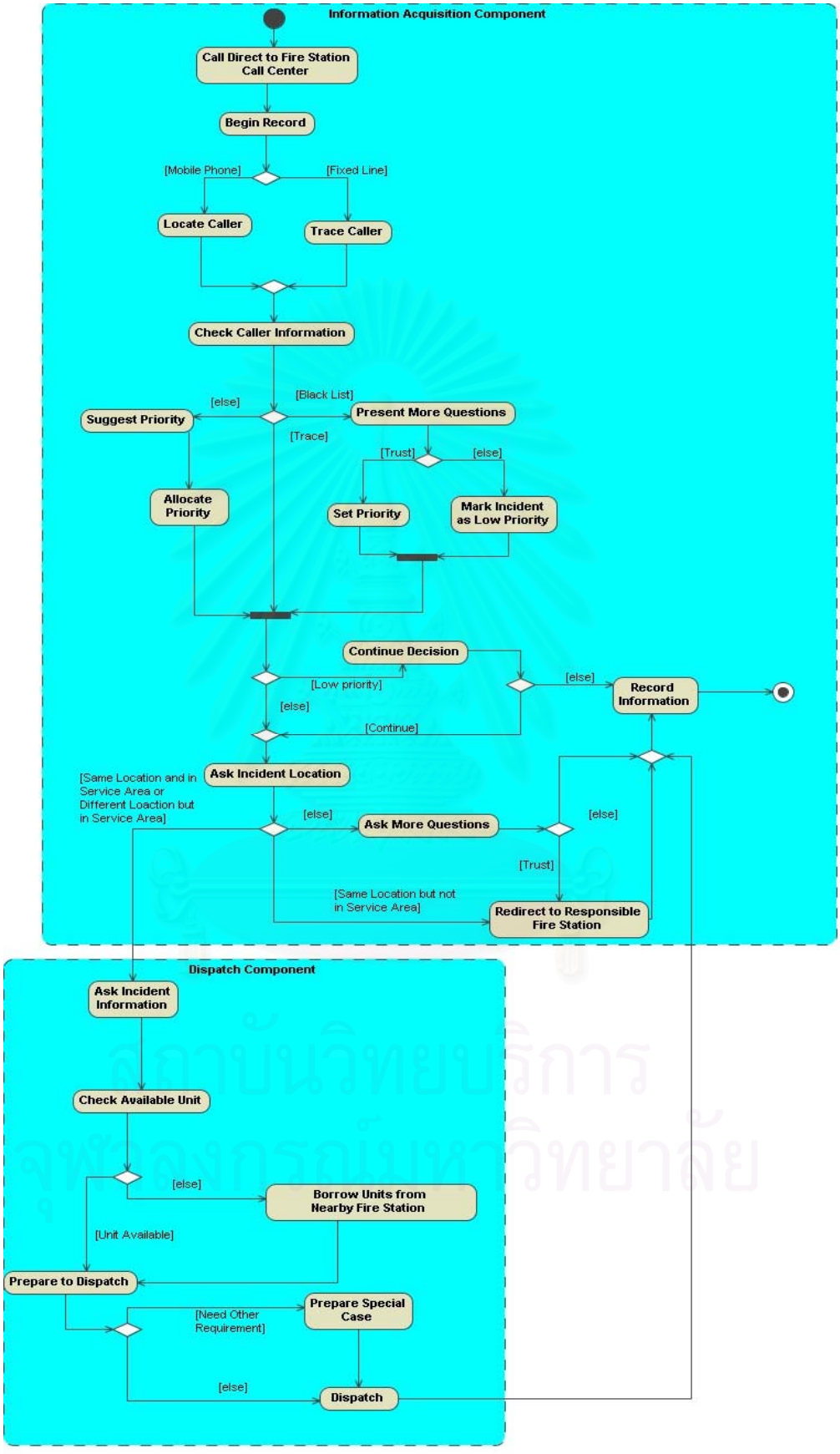
ส่วนเมื่อกำหนดเป้าหมายด้านการจัดการแบบที่สองซึ่งเน้นสภาพบำรุงรักษาได้ การแบ่งส่วนประกอบออกเป็นยี่สิบเอ็ดส่วนประกอบจะได้ค่าดัชนีที่มากกว่าการแบ่งส่วนประกอบออกเป็นแบบอื่น และค่าที่ได้มีค่าติดลบค่อนข้างมากด้วยเหตุผลเช่นเดียวกับตัวอย่างที่แล้วคือค่าน้ำหนักของเป้าหมายด้านการจัดการจะเน้นไปที่สภาพบำรุงรักษาได้ในด้านลบ รวมไปถึงค่าความซับซ้อนเมื่อแบ่งออกเป็นหนึ่งส่วนประกอบโพรเซสจะมีค่ามากกว่าเมื่อแบ่งออกเป็นแบบอื่น ดังนั้นเมื่อคำนวณตามค่ามาตรวัดแล้วจึงทำให้ค่าติดลบค่อนข้างมาก จากค่าดัชนีที่ได้แสดงว่าการแบ่งออกเป็นยี่สิบเอ็ดส่วนประกอบโพรเซส จะบรรลุเป้าหมายด้านการจัดการซึ่งเน้นที่สภาพบำรุงรักษาได้ดีกว่าเมื่อแบ่งออกเป็นแบบอื่น



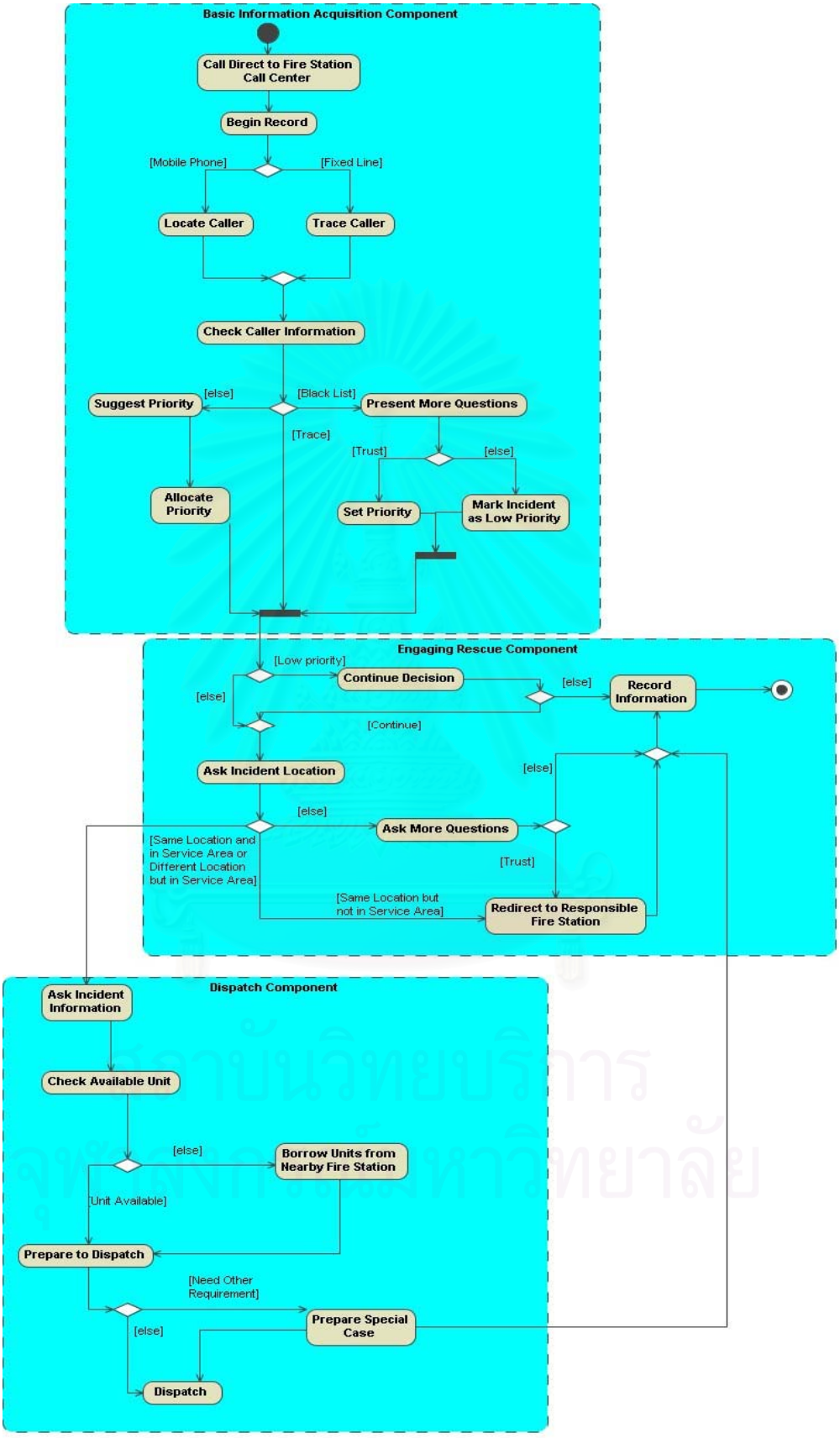
รูปที่ 7.6 ระบบตอบรับโทรศัพท์ของสถานีดับเพลิงก่อนการจัดกลุ่ม



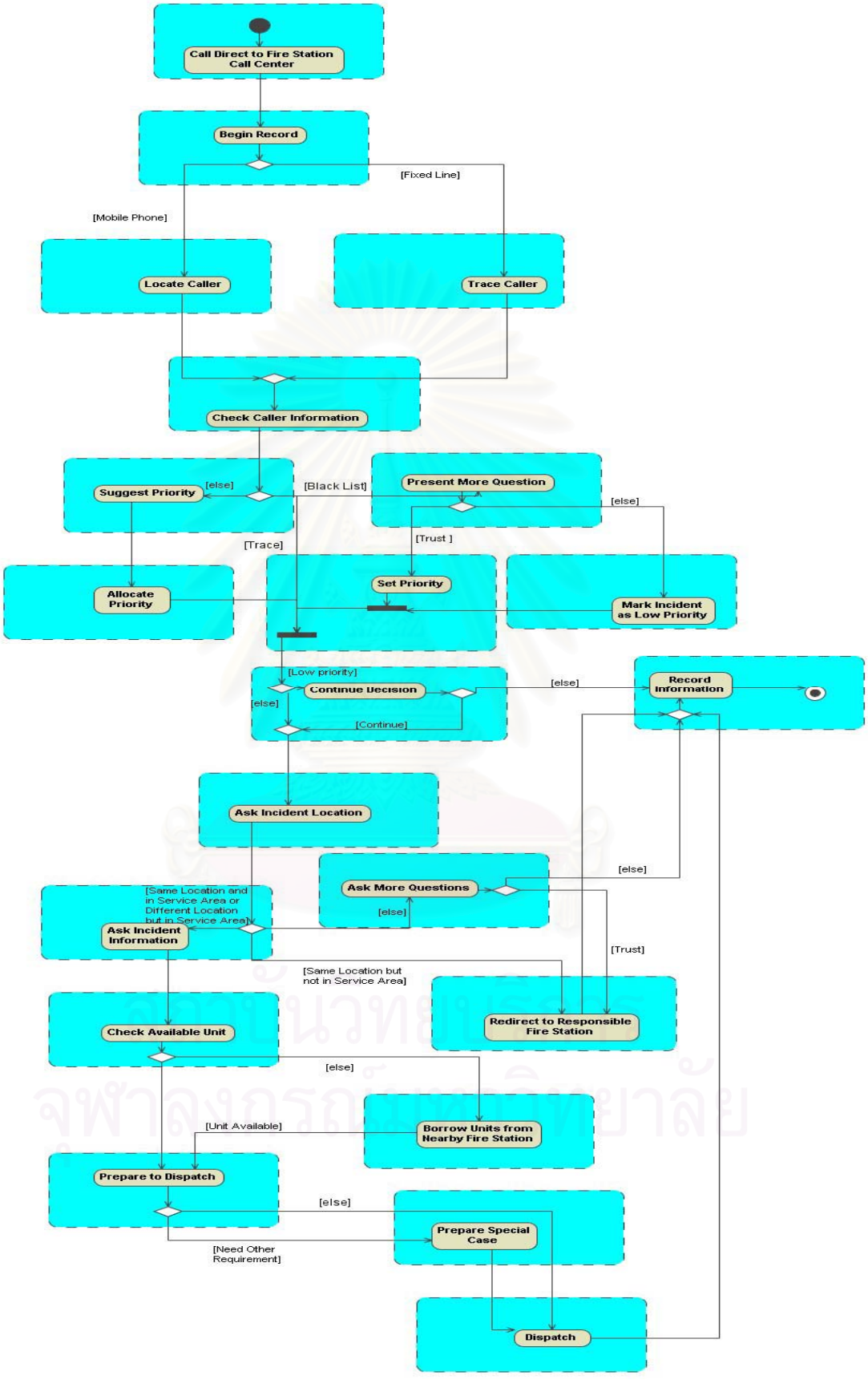
รูปที่ 7.7 ระบบตอบรับโทรศัพท์ของสถานีดับเพลิงที่แบ่งออกเป็นหนึ่งส่วนประกอบโพรเซส



รูปที่ 7.8 ระบบตอบรับโทรศัพท์ของสถานีดับเพลิงที่แบ่งออกเป็นสองส่วนประกอบโพรเซส



รูปที่ 7.9 ระบบตอบรับโทรศัพท์ของสถานีดับเพลิงที่แบ่งออกเป็นสามส่วนประกอบโพรเซส



รูปที่ 7.10 ระบบตอบรับโทรศัพท์ของสถานีดับเพลิงที่แบ่งออกเป็นยี่สิบเอ็ดส่วนประกอบโพรเซส

7.3. สรุปผลการทดสอบ

จากผลการทดสอบ ค่าดัชนีที่ได้นั้นจะขึ้นกับการแบ่งส่วนประกอบโพรเซส และการกำหนดค่าเป้าหมายด้านการจัดการ ซึ่งค่าที่ได้นั้นสามารถช่วยบอกถึงการบรรลุเป้าหมายด้านการจัดการว่าดีเพียงใด เมื่อกำหนดเป้าหมายด้านการจัดการที่เน้นการนำกลับมาใช้ใหม่นั้น เมื่อแบ่งจำนวนของส่วนประกอบโพรเซสมากเกินไปก็จะทำให้ค่าดัชนีที่ได้นั้นลดลงมากกว่าแบ่งในจำนวนที่พอเหมาะ และเมื่อกำหนดเป้าหมายด้านการจัดการที่เน้นสภาพบำรุงรักษาได้นั้น เมื่อจำนวนของส่วนประกอบโพรเซสยิ่งมากก็จะทำให้ได้ค่าดัชนีการบรรลุเป้าหมายด้านการจัดการสูงขึ้นเนื่องจากค่าดัชนีความซับซ้อนของการออกแบบลดลง

จากการทดสอบเครื่องมือด้วยกรณีศึกษา ระบบตอบรับโทรศัพท์ของตำรวจ และระบบตอบรับโทรศัพท์ของสถานีดับเพลิง เครื่องมือสามารถทำการคำนวณค่ามาตรวัดและดัชนีได้ตรงตามข้อกำหนดขอบเขตการวิจัยในบทที่ 3 ทั้งในด้านตรงตามมาตรวัด และสามารถอ่านแฟ้มข้อมูลเอ็กซ์เอ็มไอที่เก็บข้อมูลของแผนภาพแอคทิวิตีได้

บทที่ 8
สรุปผลการวิจัย

8.1 สรุปผลการวิจัย

ในงานวิจัยนี้ได้ประยุกต์แนวความคิดของการออกแบบแอพพลิเคชันด้วยส่วนประกอบซอฟต์แวร์ โดยนำมาตรวัดที่ใช้วัดค่าการออกแบบของส่วนประกอบซอฟต์แวร์ตามเป้าหมายด้านการจัดการส่วนประกอบมาใช้กับการออกแบบส่วนประกอบโพรเซส ซึ่งค่าดัชนีที่ได้สามารถช่วยให้นักออกแบบซอฟต์แวร์มีข้อมูลในการตัดสินใจที่จะเลือกการออกแบบที่เหมาะสมเพื่อนำไปพัฒนาส่วนประกอบโพรเซสต่อไป งานวิจัยนี้ได้ใช้มุมมองใหม่ของการออกแบบแอพพลิเคชันโดยเปลี่ยนจากมุมมองแบบเดิมซึ่งใช้แผนภาพคลาส มาเป็นการใช้แบบจำลองกระบวนการทางธุรกิจ ซึ่งในที่นี้ผู้วิจัยได้เลือกแผนภาพแอคทิวิตีมาช่วยในการออกแบบ โดยมีเหตุผลเนื่องจากแผนภาพแอคทิวิตีของยูเอ็มแอลนั้นสนับสนุนการแลกเปลี่ยนเมตาดาตาระหว่างเครื่องมือ ดังนั้นผู้วิจัยสามารถนำแฟ้มข้อมูลเอ็กซ์เอ็มไอที่ส่งออกจากเครื่องมือวาดแผนภาพแอคทิวิตีไปทำการวัดค่าลักษณะทางเทคนิคของการออกแบบ และคำนวณค่าดัชนีในการบรรลุเป้าหมายด้านการจัดการส่วนประกอบได้

ในงานวิจัยนี้ได้ใช้เป้าหมายด้านการจัดการเพื่อใช้ในการวัดการออกแบบได้แก่ การใช้ต้นทุนอย่างมีประสิทธิผล ความง่ายในการประกอบ ความสามารถในการปรับแต่ง ความสามารถในการนำกลับมาใช้ และสภาพบำรุงรักษาได้ ส่วนลักษณะทางเทคนิคที่ทำการวัดจากแฟ้มข้อมูลเอ็กซ์เอ็มไอนั้นได้แก่ การเชื่อมต่อกันระหว่างส่วนประกอบ การเชื่อมติดกันภายในส่วนประกอบ จำนวนของส่วนประกอบ ขนาดของส่วนประกอบ และความซับซ้อนของการออกแบบ ในงานวิจัยนี้นักออกแบบซอฟต์แวร์จะต้องป้อนข้อมูลเป้าหมายด้านการจัดการโพรเซส และเมทริกซ์ของค่าความสัมพันธ์ระหว่างเป้าหมายด้านการจัดการส่วนประกอบและลักษณะทางเทคนิคเพื่อใช้ในการคำนวณค่าดัชนีการออกแบบ

ผู้วิจัยได้พัฒนาเครื่องมือเพื่อช่วยในการวัดค่ามาตรวัดต่าง ๆ เพื่อช่วยเพิ่มความถูกต้องและลดความซับซ้อนในการคำนวณ โดยได้ใช้กรณีศึกษาสองกรณีศึกษาคือระบบตอบรับโทรศัพท์ของตำรวจ และระบบตอบรับโทรศัพท์ของสถานีดับเพลิง เพื่อทดสอบเครื่องมือ โดยผลที่ได้นั้นเป็นไปตามความสัมพันธ์ระหว่างเป้าหมายด้านการจัดการโพรเซสและลักษณะทางเทคนิคของการออกแบบนั้น

ค่าดัชนีที่วัดได้จากการออกแบบส่วนประกอบโพรเซสจะช่วยให้นักออกแบบซอฟต์แวร์ที่ออกแบบส่วนประกอบโพรเซสด้วยแบบจำลองกระบวนการทางธุรกิจสามารถตัดสินใจเลือกการ

ออกแบบที่เหมาะสมกับการพัฒนาได้สะดวกยิ่งขึ้น และงานวิจัยนี้ยังสนับสนุนการนำส่วนประกอบโพรเซสกลับมาใช้ใหม่ในการออกแบบแอพพลิเคชันสำหรับธุรกิจที่ใกล้เคียงกันทำให้ลดเวลาและค่าใช้จ่ายในการเริ่มต้นออกแบบใหม่ทั้งหมด

8.2 สรุปการประยุกต์ใช้วิธีการวัด

จากผลการทดลองในบทที่ 7 ค่าดัชนีที่ได้จะขึ้นกับ การกำหนดค่าเป้าหมายด้านการจัดการส่วนประกอบโพรเซส เมทริกซ์ค่าความสัมพันธ์ระหว่างเป้าหมายด้านการจัดการส่วนประกอบและลักษณะทางเทคนิค และค่าลักษณะทางเทคนิค โดยค่าเป้าหมายด้านการจัดการนั้นเป็นการแบ่งน้ำหนักเป้าหมายด้านการจัดการออกเป็นห้าส่วนโดยมี อัตราส่วนของการกำหนดที่แน่นอน แต่ค่าลักษณะทางเทคนิคที่คำนวณได้นั้น ขึ้นอยู่กับแต่ละการออกแบบ และมีหน่วยของการวัดไม่เหมือนกัน รวมทั้งค่าที่ได้อาจจะต่างกันมากด้วย ตัวอย่างเช่น ความซับซ้อนกับจำนวนส่วนประกอบ เมื่อแบ่งออกเป็นหนึ่งส่วนประกอบนั้น จะได้ค่าความซับซ้อนที่สูงมาก แต่จะได้จำนวนของส่วนประกอบเพียงหนึ่ง ดังนั้นเมื่อนำมาคำนวณดัชนี ค่าความซับซ้อนจะส่งผลต่อค่าดัชนีสูงมาก ปัญหานี้อาจแก้ได้ด้วยการปรับเมทริกซ์ค่าความสัมพันธ์ระหว่างเป้าหมายด้านการจัดการส่วนประกอบและลักษณะทางเทคนิค ให้มีผลในแง่ที่เกื้อหนุนต่อจำนวนของส่วนประกอบให้มากขึ้น และลดความสัมพันธ์ของความซับซ้อนลงในทางตรงกันข้าม ซึ่งการแก้ปัญหาในลักษณะนี้จะขึ้นกับประสบการณ์ของนักออกแบบซอฟต์แวร์ผู้กำหนดค่าเมทริกซ์ หรือตามความพอใจในค่าดัชนีที่วัดได้ ดังนั้นจึงไม่มีแบบแผนที่แน่ชัด การแก้ปัญหาแบบนี้จึงอาจจะไม่เหมาะสมนัก ดังนั้นผู้วิจัยจึงเสนอว่าควรจะมีการปรับฐานค่าลักษณะทางเทคนิคต่าง ๆ ที่วัดได้ให้ตรงกันก่อนที่จะใช้ในการคำนวณดัชนี อีกทั้งเมทริกซ์ค่าความสัมพันธ์ระหว่างเป้าหมายด้านการจัดการส่วนประกอบและลักษณะทางเทคนิคที่จะนำมาใช้นั้น ควรจะมีการหาค่ากลางที่เหมาะสมกับสภาพแวดล้อมการพัฒนาซอฟต์แวร์ในประเทศไทย เนื่องจากความสัมพันธ์ระหว่างเป้าหมายด้านการจัดการกับลักษณะทางเทคนิคนี้อาจจะต่างกับต่างประเทศ ตัวอย่างเช่น ค่าแรงในประเทศไทยจะถูกกว่าต่างประเทศ ดังนั้นการใช้ต้นทุนอย่างมีประสิทธิผลกับลักษณะทางเทคนิคอื่น ๆ อาจจะมีค่าต่างไปจากค่ามาตรฐานจากต่างประเทศได้

8.3 ปัญหาและอุปสรรค

ถึงแม้ว่าแฟ้มข้อมูลเอ็กซ์เอ็มไอจะเป็นที่แพร่หลายและเป็นมาตรฐานในการส่งออกข้อมูลยูเอ็มแอล แต่บริษัทที่พัฒนาเครื่องมือสำหรับการออกแบบยูเอ็มแอลแล้วทำการส่งออกแฟ้มข้อมูลนั้นไม่ได้ทำตามมาตรฐานของเอ็กซ์เอ็มไออย่างเคร่งครัด บางบริษัทก็ได้เติมลักษณะพิเศษลงไปในเครื่องมือตัวเอง ดังนั้นเมื่อนำแฟ้มข้อมูลที่ส่งออกมาจากเครื่องมือหนึ่งไปใช้กับเครื่องมือหนึ่งจึงเกิด

ปัญหาในการนำเข้าแฟ้มข้อมูล ผู้วิจัยจึงต้องเลือกเครื่องมือที่ทำการส่งออกเอ็กซ์เอ็มไอที่ตรงกับความต้องการในการนำแฟ้มข้อมูลเอ็กซ์เอ็มไอที่ส่งออกมาไปคำนวณต่อได้อย่างสะดวกที่สุด

8.4 แนวทางการวิจัยต่อไป

ประเด็นที่งานวิจัยนี้ยังไม่ได้ศึกษา และสามารถทำการวิจัยเพิ่มเติมได้ในอนาคตได้แก่

1. การหาการออกแบบส่วนประกอบโพรเซสที่ดีที่สุด เนื่องจากในงานวิจัยนี้ นักออกแบบซอฟต์แวร์ต้องกำหนดรูปแบบให้กับเครื่องมือก่อนว่าจะออกแบบโดยแบ่งกลุ่มส่วนประกอบโพรเซสในลักษณะใด งานวิจัยนี้สามารถพัฒนาต่อไปได้โดยให้เครื่องมือสามารถหารูปแบบการออกแบบส่วนประกอบโพรเซสที่เหมาะสมที่สุดตามเป้าหมายด้านการจัดการที่กำหนด

2. ในงานวิจัยนี้ค่าเมทริกซ์ความสัมพันธ์ระหว่างเป้าหมายด้านการจัดการกับลักษณะทางเทคนิคนั้นได้อิงตามงานวิจัย [17] ดังนั้นค่าเมทริกซ์จะเป็นไปตามสภาพการพัฒนาซอฟต์แวร์ในต่างประเทศ ซึ่งการนำมาใช้โดยตรงอาจจะยังไม่เหมาะกับประเทศไทย งานวิจัยนี้จึงสามารถพัฒนาให้สมบูรณ์ต่อไปได้โดยหาค่าเมทริกซ์ความสัมพันธ์ที่เหมาะสมกับการพัฒนาซอฟต์แวร์ในประเทศไทยเพื่อนำมาคำนวณค่าดัชนีแทน

3. เนื่องจากค่าลักษณะทางเทคนิคของการออกแบบนั้น มีหน่วยในการวัดต่างกันและค่าที่ได้ก็ต่างกันค่อนข้างมาก ดังนั้นจึงควรมีการปรับฐานค่าลักษณะทางเทคนิคให้เป็นไปในแนวทางเดียวกันก่อนที่จะนำมาใช้คำนวณดัชนี

4. ในการกำหนดค่ามาตรวัดความซับซ้อนยังเป็นแบบคร่าว ๆ เนื่องจากเป็นการประยุกต์มาใช้เพื่อให้เหมาะสมกับแผนภาพแอคทิวิตี โดยอิงค่าความซับซ้อนจากจำนวนของแอคชันภายในแอคทิวิตี ดังนั้นสามารถทำการวิจัยต่อไปได้โดยการเลือกมาตรวัดใหม่ที่เหมาะสม เช่นความซับซ้อนที่วัดจากโครงสร้างภายในของแอคทิวิตีแทน

รายการอ้างอิง

[1] OMG. “Unified Modeling Language (UML)” [online] Available from http://www.omg.org/technology/uml

[2] Grady Booch, James Rumbaugh and Ivar Jacobson. The Unified Modeling Language User Guide. Massachusetts: Addison-Wesley, 1999.

[3] Martin Fowler. UML Distilled third edition A Brief Guide to the Standard Object Modeling Language. Boston: Addison-Wesley, 2004.

[4] Laurent Lachel. “BPM-The promises and the practice Part 1” [Online].Available from http://www.kmworld.com/Articles/PrintArticle.aspx?ArticleID=9538

[5] S. A. White “Intro to Business Process Model Notation” [online] Available from http://www.bptrends.com [2004, Jul]

[6] IBM. “BPEL4WS Version 1.1 specification” [online] Available from http://www-128.ibm.com/developerworks/library/specification/ws-bpel/. [2003, May 5]

[7] J. Gortmaker,M. Jansen, R. W. Wagnearr “The Advantages of Web Service Orchestration in Perspective” Proceeding of 6th International Conference on Electronic Commerce(ICEC04), 2004.

[8] M. N. Huhns, M. P. Singh “Service-Oriented Computing: Key Concepts and Principles" Internet computing: Key concepts and principle Vol. 9, Issue 1, 2005.

[9] X. Ru-Zhie, H. Tao, C. Dong-Sheng, X. Yun-Jiao, Q. Le-Qiu “Reused-Oriented Process Component Representation and Retrieval” Proceeding of 5th International Conference on Computer and Information Technology (CIT05), 2005.

[10] Y. Mou, J. Cao, S. Zhang “A Process Component Model for Enterprise Business Knowledge Reuse” Proceeding IEEE International Conference on Services Computing (SCC04), 2004.

[11] O. H. Barros. “Business Information System Design Based on Process Pattern and Frameworks” Industrial Engineering Department, University of Chile [online] Available from http://www.BPtrends.com [2004, Sep]

[12] W. Rungworawut, T. Senivongse “A Guideline to Mapping Business Process to UML Class Diagrams” WSEAS Transactions on Computers Vol. 4, November, 2005.

[13] A. Bosworth Crossgain Corporation. “Developing Web Services”, W3C Workshop on Web services: Position papers 11-12 Apr, 2001.

[14] Rational Software Corporation, “Guidelines Designing Web Services in .NET” [online] Available from http://www-306.ibm.com/software/rational [2003, Jun 12]

[15] V. Tosic and B. Pagurek “On Comprehensive Contractual Descriptions of Web Services” Proceedings of Technology e-Technology, e-Commerce and e-Service (IEEE 05) The 2005 IEEE International Conference 29 Mar-1 Apr, pp. 444–449, 2005.

[16] J. Koehler, R. Hauser, S. Kapoor, F. Y. Wu and S.KumaranA “Model-Driven Transformation Method” Proceedings 7th IEEE International Enterprise Distributed Object Computing Conference (EDOC03), 2003.

[17] P. Vitharana, H. Jain, F. M. Zahedi. “Strategy-Based Design of Reusable Business Components”, IEEE Transactions on systems. Man, And Cybernetics Part C: Applications and Reviews .Vol. 34, No. 4, November, 2004.

[18] A. Lonjon “Business Process Modeling and Standardization” [online] Available from http://www.bptrends.com. [2004, Dec]

[19] H. Smith “BPM and MDA: Competitors, Alternatives or Complementary” [online] Available from http://www.bptrends.com. [2003, Jul]

[20] BPMI.org. Business Process Modeling Notation (BPMN) Version 1.0 May 3, 2004.

[21]T. J. Grose, G. C. Doney and S. A. Brodsky. Mastering XMI Java Programming with XMI, XML, and UML. OMG press, 2001.

[22]OMG. XML Metadata Interchange (XMI) [online] Available from http://www.omg.org/technology/xml

[23]OMG. “OMG XML Metadata Interchange (XMI) Specification Version 1.2” [Online].Available from http://www.omg.org/uml. [2003, July 10]

[24] N. Tagoug “Object-oriented System Decomposition Quality” Proceeding of 7th International Symposium on High Assurance Systems Engineering (HASE02), 2002.

[25]R. K. Keller and R. Schauer. “A Compositional Approach to Software Design“, Proceedings 31st Annual Hawaii International Conference on System Sciences, 1998.

[26] P. Vitharana, H. Jain and F. M. Zahedi. “Design, Retrieval and Assembly in Component-based software Development” Communications of the ACM November 2003 Vol 46 No 11, 2003.

[27] S. R. Chidamber and C. F. Kemerer “A Metrics Suite for Object Oriented Design” IEEE Transactions of software engineering Vol. 20, No. 6, June, 1994.

[28] R. Harison, S. Counsell, R. Nithi “Coupling Metrics for Object-Oriented Design” Proceeding of 5th IEEE International Software Metrics Symposium (METRICS98), 1998.

[29] A, Rotem Gal Oz “Methodology for developing Use Cases for large systems” [online] Available from http://www.rgoarchitects.com/blog/

ภาคผนวก

ภาคผนวก ก

คำอธิบายยูสเคสของเครื่องมือ

ในส่วนนี้จะแสดงคำอธิบายยูสเคสแสดงการทำงานของเครื่องมือดังตารางที่ ก.1 ถึง ก.7

ตาราง ก.1 คำอธิบายยูสเคสเปิดแฟ้มข้อมูล

ชื่อยูสเคส :	เปิดแฟ้มข้อมูล	**รหัส :** 1	**ระดับ** สูง **ความสำคัญ :**
ผู้กระทำหลัก :	นักออกแบบซอฟต์แวร์	**ชนิดยูสเคส :**	ละเอียด, สาระสำคัญ
ผู้มีส่วนเกี่ยวข้องและการใช้ประโยชน์ : นักออกแบบซอฟต์แวร์ – ต้องการวัดค่าดัชนีเพื่อช่วยในการตัดสินใจในการออกแบบส่วนประกอบโพรเซส			
รายละเอียดยูสเคส :	ผู้ใช้เรียกใช้เครื่องมือและเปิดแฟ้มข้อมูลเอ็กซ์เอ็มไอ		
สิ่งกระตุ้น :	ผู้ใช้กดปุ่ม "Open" ในแท็บ Open File		
ชนิด :	ภายนอก		
ความสัมพันธ์ : **ความเกี่ยวเนื่อง :** นักออกแบบซอฟต์แวร์ **การรวม :** อ่านและคำนวณแฟ้มข้อมูลเอ็กซ์เอ็มไอ **การขยาย :** **การสืบทอด :**			
ภาวะก่อนทำงาน :	เครื่องมือแสดงหน้าจอรอรับแฟ้มข้อมูลเอ็กซ์เอ็มไอ		
สายงานปกติ : 1. ผู้ใช้เปิดเครื่องมือรอรับแฟ้มข้อมูลเอ็กซ์เอ็มไอ 2. ผู้ใช้ป้อนข้อมูลเอ็กซ์เอ็มไอ ถ้าพบแฟ้มข้อมูล เครื่องมือคำนวณและแสดงข้อมูลลักษณะทางเทคนิคของแฟ้มข้อมูลเอ็กซ์เอ็มไอ ถ้าไม่สามารถอ่านแฟ้มข้อมูลได้ ระบบแสดงข้อผิดพลาดแล้วกลับไปสถานะรอรับแฟ้มข้อมูลเอ็กซ์เอ็มไอ ถ้าไม่ เครื่องมือกลับไปสถานะรอรับแฟ้มข้อมูลเอ็กซ์เอ็มไอ			
ภาวะหลังทำงาน :	เครื่องมือแสดงข้อมูลลักษณะทางเทคนิค		
สายงานย่อย :			
สายงานทางเลือก / สายงานพิเศษ :			

ตาราง ก.2 คำอธิบายยูสเคสอ่านแฟ้มข้อมูลเอ็กซ์เอ็มไอและคำนวณลักษณะทางเทคนิค

ชื่อยูสเคส :	อ่านและคำนวณแฟ้มข้อมูลเอ็กซ์เอ็มไอ	**รหัส :** 2	**ระดับความสำคัญ :** สูง
ผู้กระทำหลัก :	นักออกแบบซอฟต์แวร์	**ชนิดยูสเคส :**	ละเอียด, สาระสำคัญ

ผู้มีส่วนเกี่ยวข้องและการใช้ประโยชน์ :

นักวิเคราะห์ทางธุรกิจ – ต้องการวัดค่าดัชนีเพื่อช่วยในการตัดสินใจในการออกแบบส่วนประกอบโพรเซส

รายละเอียดยูสเคส :	อ่านแฟ้มข้อมูลไฟล์เอ็กซ์เอ็มไอ และคำนวณลักษณะทางเทคนิค
สิ่งกระตุ้น :	ยูสเคสเปิดแฟ้มข้อมูลเรียกใช้งาน
ชนิด :	ภายนอก

ความสัมพันธ์ :

ความเกี่ยวเนื่อง :

การรวม :

การขยาย :

การสืบทอด :

ภาวะก่อนทำงาน : เครื่องมือรอชื่อแฟ้มข้อมูลก่อนนำไปประมวลผล

สายงานปกติ :

1. รอรับชื่อแฟ้มข้อมูลเอ็กซ์เอ็มไอก่อนจะนำไปประมวลผล
2. อ่านค่าแฟ้มข้อมูลเก็บข้อมูลไว้เป็นโครงสร้าง
3. คำนวณลักษณะทางเทคนิค"การเชื่อมต่อกันระหว่างส่วนประกอบ"
4. คำนวณลักษณะทางเทคนิค"การเชื่อมติดกันภายในของส่วนประกอบ"
5. คำนวณลักษณะทางเทคนิค"จำนวนของส่วนประกอบ"
6. คำนวณลักษณะทางเทคนิค"ขนาดของส่วนประกอบ"
7. คำนวณลักษณะทางเทคนิค"ความซับซ้อนของการออกแบบ"

ภาวะหลังทำงาน : เครื่องมือแสดงข้อมูลลักษณะทางเทคนิค

สายงานย่อย :

สายงานทางเลือก / สายงานพิเศษ :

ตาราง ก.3 คำอธิบายยูสเคสป้อนข้อมูลเป้าหมายด้านการจัดการ

ชื่อยูสเคส :	ป้อนข้อมูลเป้าหมายด้านการจัดการ	**รหัส :** 3	**ระดับความสำคัญ :** สูง
ผู้กระทำหลัก :	นักออกแบบซอฟต์แวร์	**ชนิดยูสเคส :**	ละเอียด, สาระสำคัญ
ผู้มีส่วนเกี่ยวข้องและการใช้ประโยชน์ : นักออกแบบซอฟต์แวร์ – ต้องการวัดค่าดัชนีเพื่อช่วยในการตัดสินใจในการออกแบบส่วนประกอบโพรเซส			
รายละเอียดยูสเคส :	ผู้ใช้ป้อนเป้าหมายด้านการจัดการเข้าสู่เครื่องมือ		
สิ่งกระตุ้น :	ผู้ใช้เลือกแท็บ "Input Managerial Goals"		
ชนิด :	ภายนอก		
ความสัมพันธ์ : **ความเกี่ยวเนื่อง :** นักออกแบบซอฟต์แวร์ **การรวม :** ตรวจสอบค่า **การขยาย :** **การสืบทอด :**			
ภาวะก่อนทำงาน :	เครื่องมือแสดงหน้าจอเป้าหมายด้านการจัดการ		
สายงานปกติ : 1. เครื่องมือรอรับค่าเป้าหมายด้านการจัดการ 2. ผู้ใช้ป้อนค่าเป้าหมายด้านการจัดการ ถ้าป้อนครบทั้งห้าเป้าหมายแล้วไม่มีปัญหา เครื่องมือจะแสดงค่าเป้าหมายด้านการจัดการพร้อมและพร้อมทำงานในแท็บต่อไป ถ้าไม่ เครื่องมือจะไปที่ตำแหน่งกล่องข้อมูลเริ่มต้น หรือตำแหน่งกล่องข้อมูลที่มีปัญหา			
ภาวะหลังทำงาน :	เครื่องมือแสดงข้อมูลเป้าหมายด้านการจัดการ		
สายงานย่อย :			
สายงานทางเลือก / สายงานพิเศษ :			

ตาราง ก.4 คำอธิบายยูสเคสป้อนข้อมูลเมทริกซ์ความสัมพันธ์

ชื่อยูสเคส :	ป้อนข้อมูลเมทริกซ์ความสัมพันธ์	**รหัส :** 4	**ระดับความสำคัญ :** สูง
ผู้กระทำหลัก :	นักออกแบบซอฟต์แวร์	**ชนิดยูสเคส :**	ละเอียด, สาระสำคัญ

ผู้มีส่วนเกี่ยวข้องและการใช้ประโยชน์ :

นักออกแบบซอฟต์แวร์ – ต้องการวัดค่าดัชนีเพื่อช่วยในการตัดสินใจในการออกแบบส่วนประกอบโพรเซส

รายละเอียดยูสเคส :	ผู้ใช้ป้อนข้อมูลเมทริกซ์ความสัมพันธ์
สิ่งกระตุ้น :	ผู้ใช้เลือกแท็บ "Input Relation Matirx"
ชนิด :	ภายนอก

ความสัมพันธ์ :

ความเกี่ยวเนื่อง :	นักออกแบบซอฟต์แวร์
การรวม :	ตรวจสอบค่า
การขยาย :	
การสืบทอด :	

ภาวะก่อนทำงาน : เครื่องมือแสดงหน้าจอเป้าหมายด้านการจัดการ

สายงานปกติ :

1. เครื่องมือรอรับค่าเมทริกซ์ความสัมพันธ์
2. ผู้ใช้ป้อนค่าเมทริกซ์ความสัมพันธ์หรือเลือกค่ามาตรฐาน

ถ้าป้อนครบแล้วไม่มีปัญหา

เครื่องมือจะแสดงข้อมูลค่าเมทริกซ์ความสัมพันธ์และพร้อมทำงานในแท็บต่อไป

ถ้าไม่

เครื่องมือจะไปที่ตำแหน่งกล่องข้อมูลเริ่มต้น หรือตำแหน่งกล่องข้อมูลที่มีปัญหา

ภาวะหลังทำงาน : เครื่องมือแสดงข้อมูลค่าเมทริกซ์ความสัมพันธ์

สายงานย่อย :

สายงานทางเลือก / สายงานพิเศษ :

ตาราง ก.5 คำอธิบายยูสเคสตรวจสอบค่า

ชื่อยูสเคส : ตรวจสอบค่า	**รหัส :** 5	**ระดับความสำคัญ :** สูง
ผู้กระทำหลัก : นักออกแบบซอฟต์แวร์	**ชนิดยูสเคส :**	ละเอียด, สาระสำคัญ
ผู้มีส่วนเกี่ยวข้องและการใช้ประโยชน์ : นักออกแบบซอฟต์แวร์ – ต้องการวัดค่าดัชนีเพื่อช่วยในการตัดสินใจในการออกแบบส่วนประกอบโพรเซส		
รายละเอียดยูสเคส :	ตรวจสอบค่าที่ผู้ใช้ป้อนเข้ามาว่าอยู่ในช่วงที่กำหนดไว้	
สิ่งกระตุ้น :	ยูสเคสป้อนค่าเป้าหมายด้านการจัดการหรือเมทริกซ์ความสัมพันธ์	
ชนิด :	ภายนอก	
ความสัมพันธ์ : **ความเกี่ยวเนื่อง :** **การรวม :** **การขยาย :** **การสืบทอด :**		
ภาวะก่อนทำงาน :	เครื่องมือรอการป้อนข้อมูลของผู้ใช้	
สายงานปกติ : 1. เครื่องมือรอการป้อนข้อมูลจากผู้ใช้ 2. เครื่องมือตรวจสอบข้อมูลที่ผู้ใช้ป้อน ถ้าผิดจากค่าที่กำหนดไว้ แสดงผลที่ตำแหน่งที่ป้อนข้อมูลผิดพลาด ถ้าไม่ เครื่องมือรอการตรวจสอบจากผู้ใช้ต่อไป		
ภาวะหลังทำงาน :	เครื่องมือแสดงผลการตรวจสอบการป้อนข้อมูล	
สายงานย่อย :		
สายงานทางเลือก / สายงานพิเศษ :		

ตาราง ก.6 คำอธิบายยูสเคสแสดงค่าดัชนี

ชื่อยูสเคส : แสดงค่าดัชนี		**รหัส :** 6	**ระดับ** สูง **ความสำคัญ :**
ผู้กระทำหลัก : นักออกแบบซอฟต์แวร์		**ชนิดยูสเคส :** ละเอียด, สาระสำคัญ	
ผู้มีส่วนเกี่ยวข้องและการใช้ประโยชน์ : นักออกแบบซอฟต์แวร์ – ต้องการวัดค่าดัชนีเพื่อช่วยในการตัดสินใจในการออกแบบส่วนประกอบโพรเซส			
รายละเอียดยูสเคส :	แสดงค่าภายหลังจากการคำนวณค่าดัชนีของเครื่องมือ		
สิ่งกระตุ้น :	ผู้ใช้กดปุ่ม "Calculate" ในแท็บ "Measurement Value"		
ชนิด :	ภายนอก		
ความสัมพันธ์ : **ความเกี่ยวเนื่อง :** นักออกแบบซอฟต์แวร์ **การรวม :** คำนวณค่าดัชนี **การขยาย :** **การสืบทอด :**			
ภาวะก่อนทำงาน :	เครื่องมือแสดงหน้าจอแสดงผลค่าดัชนี		
สายงานปกติ : 1. เครื่องมือรอแสดงผลค่าดัชนี 2. ผู้ใช้กดปุ่ม "Calculate" 3. เครื่องมือประมวลผลตามสูตรที่ได้กำหนดไว้ 4. เครื่องมือแสดงค่าดัชนี			
ภาวะหลังทำงาน :	เครื่องมือแสดงค่าดัชนีในหน้าจอแสดงผล		
สายงานย่อย :			
สายงานทางเลือก / สายงานพิเศษ :			

ตาราง ก.7 คำอธิบายยูสเคสคำนวณค่าดัชนี

ชื่อยูสเคส : คำนวณค่าดัชนี	**รหัส :** 7	**ระดับความสำคัญ :** สูง
ผู้กระทำหลัก : นักออกแบบซอฟต์แวร์	**ชนิดยูสเคส :**	ละเอียด, สาระสำคัญ
ผู้มีส่วนเกี่ยวข้องและการใช้ประโยชน์ : นักออกแบบซอฟต์แวร์ – ต้องการวัดค่าดัชนีเพื่อช่วยในการตัดสินใจในการออกแบบส่วนประกอบโพรเซส		
รายละเอียดยูสเคส :	คำนวณค่าดัชนีจากสูตรที่ได้กำหนดไว้	
สิ่งกระตุ้น :	ยูสเคสแสดงค่าดัชนีสั่งให้คำนวณและส่งกลับค่าดัชนี	
ชนิด :	ภายนอก	
ความสัมพันธ์ : **ความเกี่ยวเนื่อง :** **การรวม :** **การขยาย :** **การสืบทอด :**		
ภาวะก่อนทำงาน :	เครื่องมือรอคำสั่งคำนวณค่าดัชนี	
สายงานปกติ : 1. เครื่องมือรอคำสั่งการคำนวณค่าดัชนี 2. เครื่องมือคำนวณค่าดัชนีตามสูตรที่ได้กำหนดไว้ 3. เครื่องมือส่งค่าดัชนีเพื่อแสดงผล		
ภาวะหลังทำงาน :	เครื่องมือส่งกลับค่าดัชนีไปยังหน้าจอแสดงผล	
สายงานย่อย :		
สายงานทางเลือก / สายงานพิเศษ :		

ภาคผนวก ข

แฟ้มข้อมูลเอ็กซ์เอ็มไอที่นำมาทำการวัดค่ามาตรวัด

ในภาคผนวกนี้จะแสดงโครงสร้างและตัวอย่างของแฟ้มข้อมูลเอ็กซ์เอ็มไอที่เครื่องมือที่พัฒนาขึ้นมาในงานวิจัยนี้นำมาทำการวัดค่ามาตรวัด โดยแฟ้มข้อมูลดังกล่าวเป็นแฟ้มข้อมูลเอ็กซ์เอ็มไอรุ่น 2.1 ที่ถูกส่งออกจากเมจิกดรอว์รุ่น 10.0 และเนื่องจากโครงสร้างเอ็กซ์เอ็มไอมีความซับซ้อนและมีส่วนย่อยอยู่มาก ในที่นี้จึงขอเลือกแสดงเฉพาะส่วนที่เครื่องมือดังกล่าวใช้ในการวัดค่ามาตรวัดเท่านั้น

```
uml:Model xmi:id='eee_1045467100313_135436_1' name='Data' visibility='public'>

<xmi:Extension xmi:Extender='MagicDraw UML 10.0'
xmi:ExtenderID='MagicDraw UML 10.0'>

<moduleExtension ignoredInModule='true'/>

</xmi:Extension>

<ownedComment xmi:type='uml:Comment'
xmi:id='_10_0_24400562_1151307371781_743419_1' body='Author:Administrator.&#10;Created:26/6/2549,
14:36 น..&#10;Title:.&#10;Comment:.&#10;' annotatedElement='eee_1045467100313_135436_1'
owningElement='eee_1045467100313_135436_1'/>

<ownedMember xmi:type='uml:Package'
href='UML_Standard_Profile.xml|magicdraw_uml_standard_profile_v_0001'>

<xmi:Extension xmi:Extender='MagicDraw UML 10.0'
xmi:ExtenderID='MagicDraw UML 10.0'>

<referenceExtension referentPath='UML Standard
Profile' referentType='Package'/>

</xmi:Extension>

</ownedMember>

<ownedMember xmi:type='uml:Activity'
xmi:id='_10_0_24400562_1151307404921_565410_3' name='Untitled' visibility='public'
owningPackage='eee_1045467100313_135436_1'>

<node xmi:type='uml:CallBehaviorAction'
xmi:id='_10_0_24400562_1151307511140_746731_6' name='Accept Request' visibility='public'
activity='_10_0_24400562_1151307404921_565410_3'
outgoing='_10_0_24400562_1151307867015_934557_99'
incoming='_10_0_24400562_1151307863750_572807_85'/>

<node xmi:type='uml:CallBehaviorAction'
xmi:id='_10_0_24400562_1151307513578_486981_17' name='Book Flight' visibility='public'
activity='_10_0_24400562_1151307404921_565410_3'
incoming='_10_0_24400562_1151307867015_934557_99'>

<outgoing
xmi:idref='_10_0_24400562_1151307944234_343401_124'/>

<outgoing
xmi:idref='_10_0_24400562_1151307955437_393001_166'/>

</node>
```

รูปที่ ข.1 ส่วนของแฟ้มข้อมูลที่สอดคล้องกับแผนภาพแอคทิวิตีตามรูปที่ 2.1

```
                    <node xmi:type='uml:CallBehaviorAction'
xmi:id='_10_0_24400562_1151307749359_66353_28' name='Send Flight Cancellation' visibility='public'
activity='_10_0_24400562_1151307404921_565410_3'
outgoing='_10_0_24400562_1151307972187_442722_180'
incoming='_10_0_24400562_1151307955437_393001_166'/>

                                        <node xmi:type='uml:CallBehaviorAction'
xmi:id='_10_0_24400562_1151307751640_368337_39' name='Send Unavailable notice ' visibility='public'
activity='_10_0_24400562_1151307404921_565410_3'
outgoing='_10_0_24400562_1151308034484_580478_250'>

                                                <incoming
xmi:idref='_10_0_24400562_1151307972187_442722_180'/>

                                                <incoming
xmi:idref='_10_0_24400562_1151308002890_552201_222'/>

                                        </node>

                                        <node xmi:type='uml:CallBehaviorAction'
xmi:id='_10_0_24400562_1151307752953_930717_50' name='Send Hotel Cancellation' visibility='public'
activity='_10_0_24400562_1151307404921_565410_3'
outgoing='_10_0_24400562_1151308002890_552201_222'
incoming='_10_0_24400562_1151307978187_214441_208'/>

                                        <node xmi:type='uml:CallBehaviorAction'
xmi:id='_10_0_24400562_1151307754421_49135_61' name='Book Hotel' visibility='public'
activity='_10_0_24400562_1151307404921_565410_3'
incoming='_10_0_24400562_1151307944234_343401_124'>

                                                <outgoing
xmi:idref='_10_0_24400562_1151307978187_214441_208'/>

                                                <outgoing
xmi:idref='_10_0_24400562_1151308012796_156125_236'/>

                                        </node>

                                        <node xmi:type='uml:InitialNode'
xmi:id='_10_0_24400562_1151307763406_61666_72' visibility='public'
outgoing='_10_0_24400562_1151307863750_572807_85'
activity='_10_0_24400562_1151307404921_565410_3'/>

                                        <node xmi:type='uml:ActivityFinalNode'
xmi:id='_10_0_24400562_1151307768437_548058_78' visibility='public'
activity='_10_0_24400562_1151307404921_565410_3'>

                                                <incoming
xmi:idref='_10_0_24400562_1151308034484_580478_250'/>

                                                <incoming
xmi:idref='_10_0_24400562_1151308037531_300566_264'/>

                                        </node>
```

รูปที่ ข.1 ส่วนของแฟ้มข้อมูลที่สอดคล้องกับแผนภาพแอคทิวิตีตามรูปที่ 2.1(ต่อ)

```
                    <node xmi:type='uml:CallBehaviorAction'
xmi:id='_10_0_24400562_1151307895265_620660_112' name='Charge Buyer' visibility='public'
activity='_10_0_24400562_1151307404921_565410_3'
outgoing='_10_0_24400562_1151308037531_300566_264'
incoming='_10_0_24400562_1151308012796_156125_236'/>

                                        <edge xmi:type='uml:ControlFlow'
xmi:id='_10_0_24400562_1151307863750_572807_85' visibility='public'
activity='_10_0_24400562_1151307404921_565410_3' source='_10_0_24400562_1151307763406_61666_72'
target='_10_0_24400562_1151307511140_746731_6'>

                                            <weight xmi:type='uml:LiteralInteger'
xmi:id='_10_0_24400562_1151307863750_571738_86' value='1' visibility='public'/>

                                        </edge>

                                        <edge xmi:type='uml:ControlFlow'
xmi:id='_10_0_24400562_1151307867015_934557_99' visibility='public'
activity='_10_0_24400562_1151307404921_565410_3' source='_10_0_24400562_1151307511140_746731_6'
target='_10_0_24400562_1151307513578_486981_17'>

                                            <weight xmi:type='uml:LiteralInteger'
xmi:id='_10_0_24400562_1151307867015_348030_100' value='1' visibility='public'/>

                                        </edge>

                                        <edge xmi:type='uml:ControlFlow'
xmi:id='_10_0_24400562_1151307944234_343401_124' visibility='public'
activity='_10_0_24400562_1151307404921_565410_3' source='_10_0_24400562_1151307513578_486981_17'
target='_10_0_24400562_1151307754421_49135_61'>

                                            <weight xmi:type='uml:LiteralInteger'
xmi:id='_10_0_24400562_1151307944234_718574_125' value='1' visibility='public'/>

                                        </edge>

                                        <edge xmi:type='uml:ControlFlow'
xmi:id='_10_0_24400562_1151307955437_393001_166' visibility='public'
activity='_10_0_24400562_1151307404921_565410_3' source='_10_0_24400562_1151307513578_486981_17'
target='_10_0_24400562_1151307749359_66353_28'>

                                            <weight xmi:type='uml:LiteralInteger'
xmi:id='_10_0_24400562_1151307955437_102714_167' value='1' visibility='public'/>

                                        </edge>

                                        <edge xmi:type='uml:ControlFlow'
xmi:id='_10_0_24400562_1151307972187_442722_180' visibility='public'
activity='_10_0_24400562_1151307404921_565410_3' source='_10_0_24400562_1151307749359_66353_28'
target='_10_0_24400562_1151307751640_368337_39'>

                                            <weight xmi:type='uml:LiteralInteger'
xmi:id='_10_0_24400562_1151307972187_318247_181' value='1' visibility='public'/>

                                    </edge>
```

รูปที่ ข.1 ส่วนของแฟ้มข้อมูลที่สอดคล้องกับแผนภาพแอคทิวิตีตามรูปที่ 2.1 (ต่อ)

```
                    <edge xmi:type='uml:ControlFlow' xmi:id='_10_0_24400562_1151307978187_214441_208'
visibility='public' activity='_10_0_24400562_1151307404921_565410_3'
source='_10_0_24400562_1151307754421_49135_61' target='_10_0_24400562_1151307752953_930717_50'>

                                                <weight xmi:type='uml:LiteralInteger'
xmi:id='_10_0_24400562_1151307978187_162487_209' value='1' visibility='public'/>

                                            </edge>

                                            <edge xmi:type='uml:ControlFlow'
xmi:id='_10_0_24400562_1151308002890_552201_222' visibility='public'
activity='_10_0_24400562_1151307404921_565410_3' source='_10_0_24400562_1151307752953_930717_50'
target='_10_0_24400562_1151307751640_368337_39'>

                                                <weight xmi:type='uml:LiteralInteger'
xmi:id='_10_0_24400562_1151308002890_21249_223' value='1' visibility='public'/>

                                            </edge>

                                            <edge xmi:type='uml:ControlFlow'
xmi:id='_10_0_24400562_1151308012796_156125_236' visibility='public'
activity='_10_0_24400562_1151307404921_565410_3' source='_10_0_24400562_1151307754421_49135_61'
target='_10_0_24400562_1151307895265_620660_112'>

                                                <weight xmi:type='uml:LiteralInteger'
xmi:id='_10_0_24400562_1151308012796_240723_237' value='1' visibility='public'/>

                                            </edge>

                                            <edge xmi:type='uml:ControlFlow'
xmi:id='_10_0_24400562_1151308034484_580478_250' visibility='public'
activity='_10_0_24400562_1151307404921_565410_3' source='_10_0_24400562_1151307751640_368337_39'
target='_10_0_24400562_1151307768437_548058_78'>

                                                <weight xmi:type='uml:LiteralInteger'
xmi:id='_10_0_24400562_1151308034484_943791_251' value='1' visibility='public'/>

                                            </edge>

                                            <edge xmi:type='uml:ControlFlow'
xmi:id='_10_0_24400562_1151308037531_300566_264' visibility='public'
activity='_10_0_24400562_1151307404921_565410_3' source='_10_0_24400562_1151307895265_620660_112'
target='_10_0_24400562_1151307768437_548058_78'>

                                                <weight xmi:type='uml:LiteralInteger'
xmi:id='_10_0_24400562_1151308037531_737690_265' value='1' visibility='public'/>

                                            </edge>

                                        </ownedMember>

            </uml:Model>
```

รูปที่ ข.1 ส่วนของแฟ้มข้อมูลที่สอดคล้องกับแผนภาพแอคทิวิตีตามรูปที่ 2.1 (ต่อ)

ภาคผนวก ค

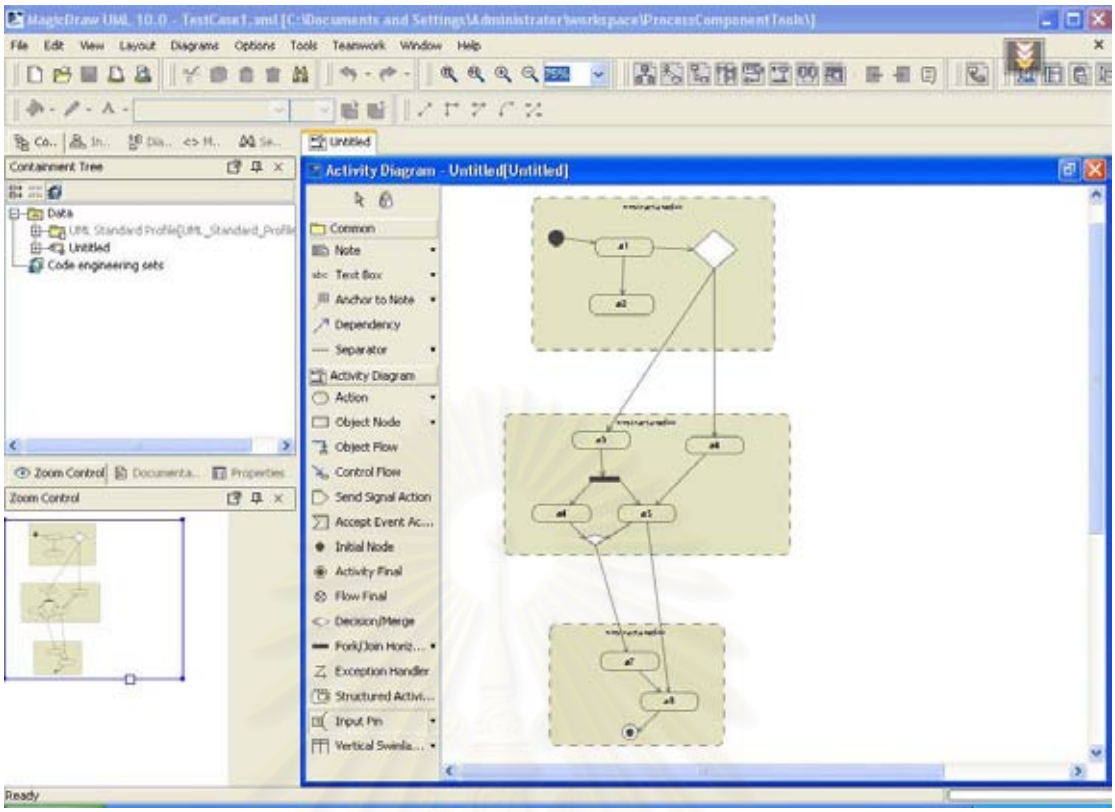
การใช้งานเครื่องมือ

ในภาคผนวกนี้จะกล่าวถึงการใช้งานเครื่องมือสนับสนุนการวัดค่ามาตรวัดจากแผนภาพแอคทิวิตีในรูปแบบแฟ้มข้อมูลเอ็กซ์เอ็มไอ โดยจะแบ่งออกเป็นสองส่วนคือ การใช้เครื่องมืออื่น ๆ ที่ใช้ร่วมกับเครื่องมือที่พัฒนาขึ้น และการใช้เครื่องมือที่พัฒนาขึ้น

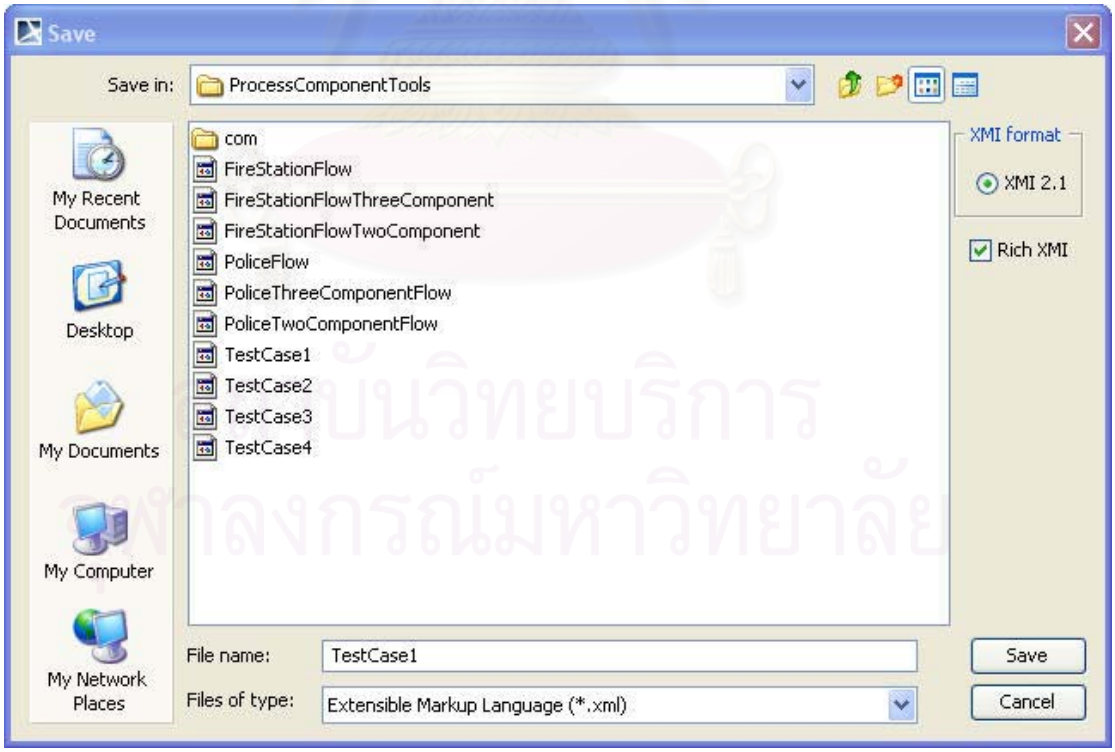
ค.1 การใช้เครื่องมืออื่น ๆ ที่ใช้ร่วมกับเครื่องมือที่พัฒนาขึ้น

ในการใช้เครื่องมือนี้ ผู้ใช้จะต้องสร้างแผนภาพแอคทิวิตีด้วยเมจิกดรอว์ รุ่น 10.0 และทำการส่งออกข้อมูลแผนภาพดังกล่าวด้วยรูปแบบแฟ้มข้อมูลเอ็กซ์เอ็มไอ รุ่น 2.1 เครื่องมือจะทำการอ่านแฟ้มข้อมูลและประมวลผลในขั้นต่อไป การใช้เครื่องมือเพื่อทำการส่งออกแฟ้มข้อมูลมีขั้นตอนดังต่อไปนี้

1. เริ่มทำการออกแบบส่วนประกอบโพรเซสในเมจิกดรอว์รุ่น 10.0 โดยจัดกลุ่มแผนภาพแอคทิวิตีออกเป็นแผนภาพย่อยเมื่อทำการออกแบบเสร็จจะได้ดังรูปที่ ค.1
2. ไปที่เมนู “File” แล้วเลือกเมนูย่อย “Save” จะปรากฏหน้าต่าง “Save” ดังรูปที่ ค.2
3. ในหัวข้อ “XMI Format” เลือก รุ่น 2.1
4. ในกรอบ “File Name” พิมพ์ชื่อของแฟ้มข้อมูลที่ต้องการบันทึก
5. กดปุ่ม “Save”



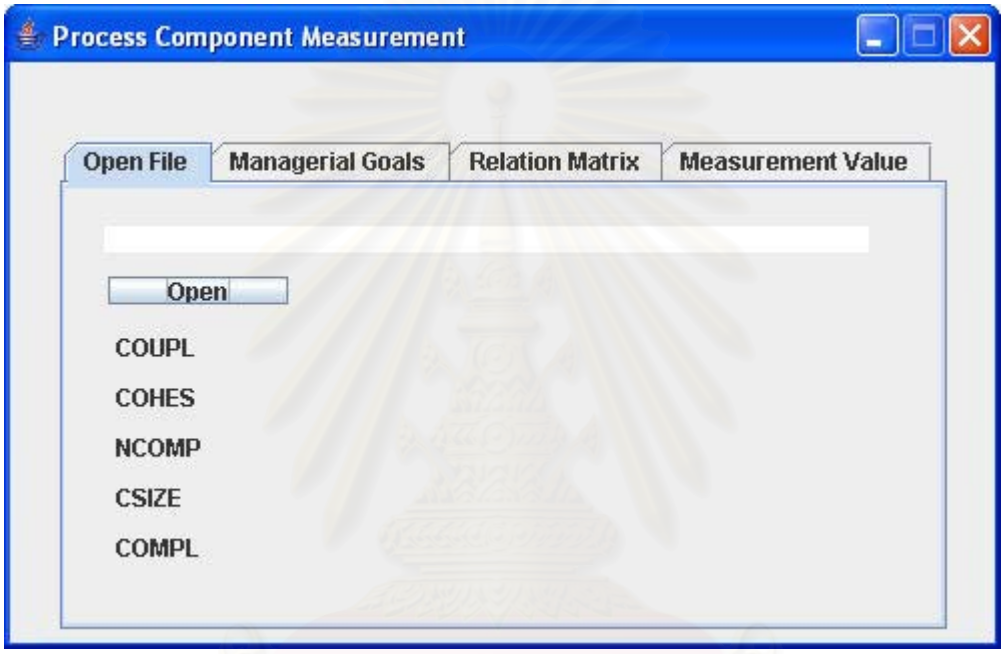
รูปที่ ค.1 หน้าจอเมื่อออกแบบแผนภาพแอคทิวิตีด้วยเมจิกดรอว์รุ่น 10.0



รูปที่ ค.2 หน้าจอ “Save” และ เลือก “XMI Format” ในหน้าต่าง “Save”

ค.2 การใช้เครื่องมือที่พัฒนาขึ้น

ในส่วนนี้จะอธิบายการใช้เครื่องมือและแท็บของเครื่องมือโดยเครื่องมือจะแบ่งออกเป็นสี่แท็บซึ่งประกอบด้วย “Open File” ซึ่งมีหน้าที่เปิดแฟ้มข้อมูลเอ็กซ์เอ็มไอ แท็บ “Managerial Goals” เพื่อใช้ป้อนค่าน้ำหนักของเป้าหมายการจัดการส่วนประกอบ แท็บ “Relation Matrix” เพื่อใช้ป้อนค่าความสัมพันธ์ระหว่างเป้าหมายด้านการจัดการส่วนประกอบและลักษณะทางเทคนิคของส่วนประกอบ แท็บ “Measurement Value” เพื่อคำนวณค่าดัชนี แท็บทั้งหมดนี้แสดงดังรูปที่ ค.3



ค.3 หน้าจอเครื่องมือเมื่อทำการเลือกแท็บ “Openfle”

โดยขั้นตอนการใช้เครื่องมือมีดังนี้

1. เมื่อเปิดเครื่องมือจะพบหน้าจอดังรูปที่ ค.3
2. กดปุ่ม “Open” เพื่อเปิดแฟ้มข้อมูลเอ็กซ์เอ็มไอ
3. เครื่องมือจะแสดงหน้าจอ “Open” ดังรูปที่ ค.4
4. เลือกแฟ้มข้อมูลเอ็กซ์เอ็มไอดังรูปที่ ค.5 แล้วกด “Open”
5. เมื่อเปิดไฟล์สำเร็จจะปรากฏข้อมูลของลักษณะทางเทคนิคดังรูปที่ ค.6
6. เลือกแท็บ “Managerial Goals” ป้อนข้อมูลลักษณะทางเทคนิคดังรูปที่ ค.7

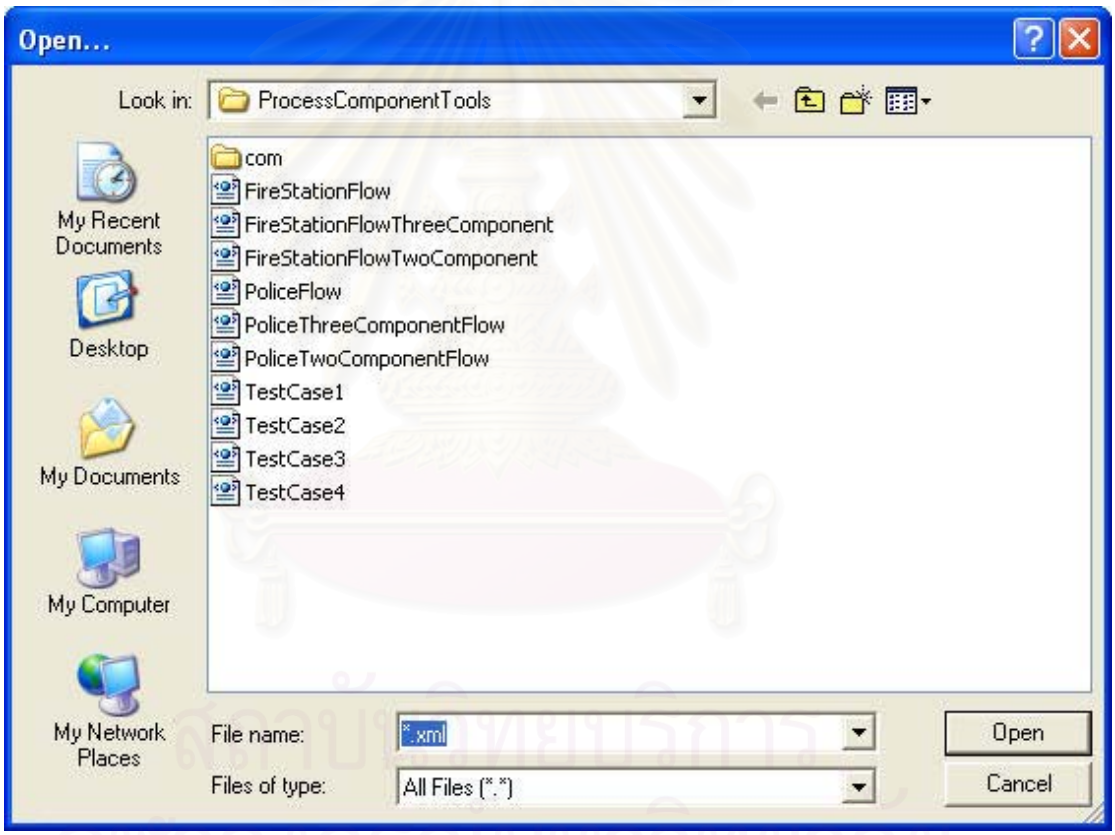
7. เลือกแท็บ “Relation Matrix” ดังรูปที่ ค.8

8. สามารถป้อนข้อมูลความสัมพันธ์เองหรือเลือกค่ามาตรฐานโดยกดปุ่ม “Default” ดังรูปที่ ค.9

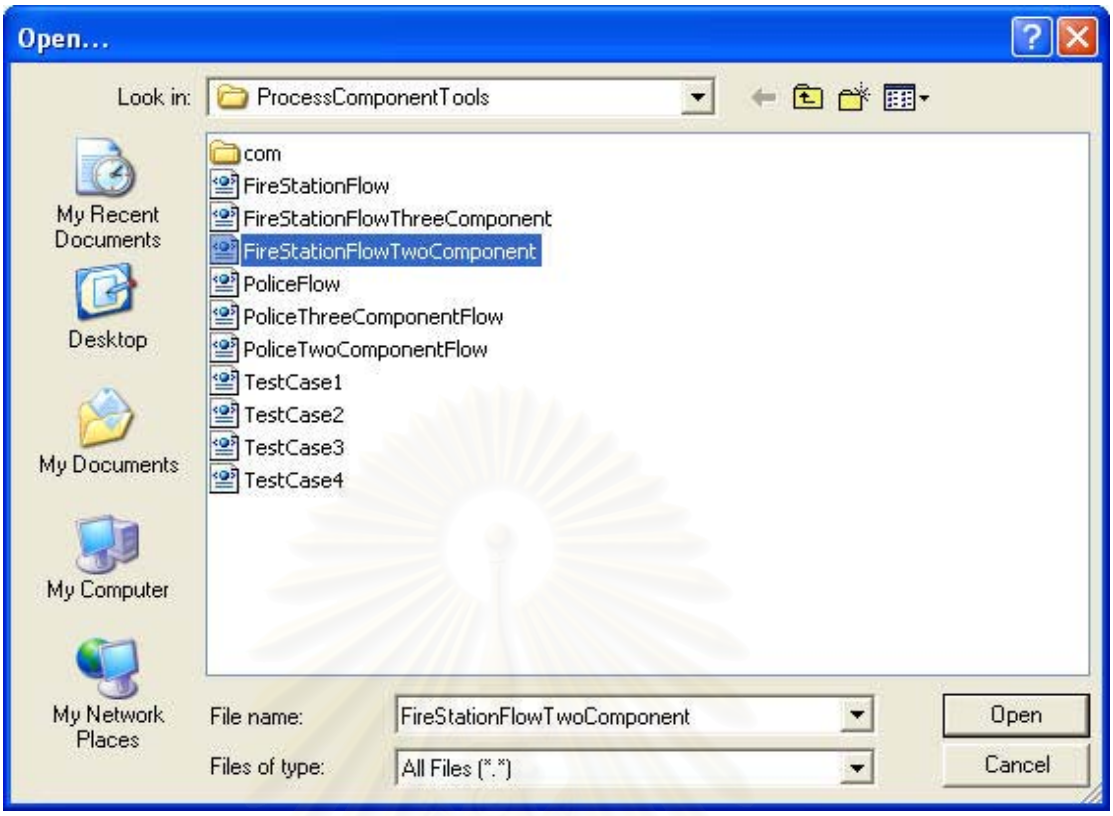
9. เลือกแท็บ “Measurement Value” ดังรูปที่ ค.10

10. กด “Calculate” เพื่อคำนวณค่าดัชนีดังรูปที่ ค.11

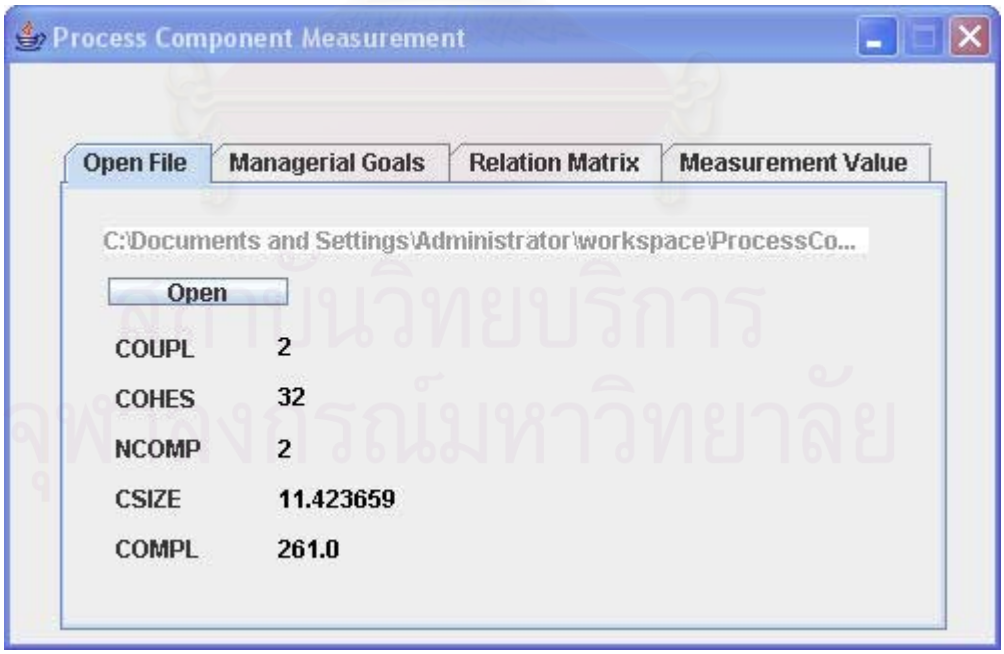
11. นำค่าที่ได้ไปเปรียบเทียบกับค่าจากการออกแบบในลักษณะอื่น



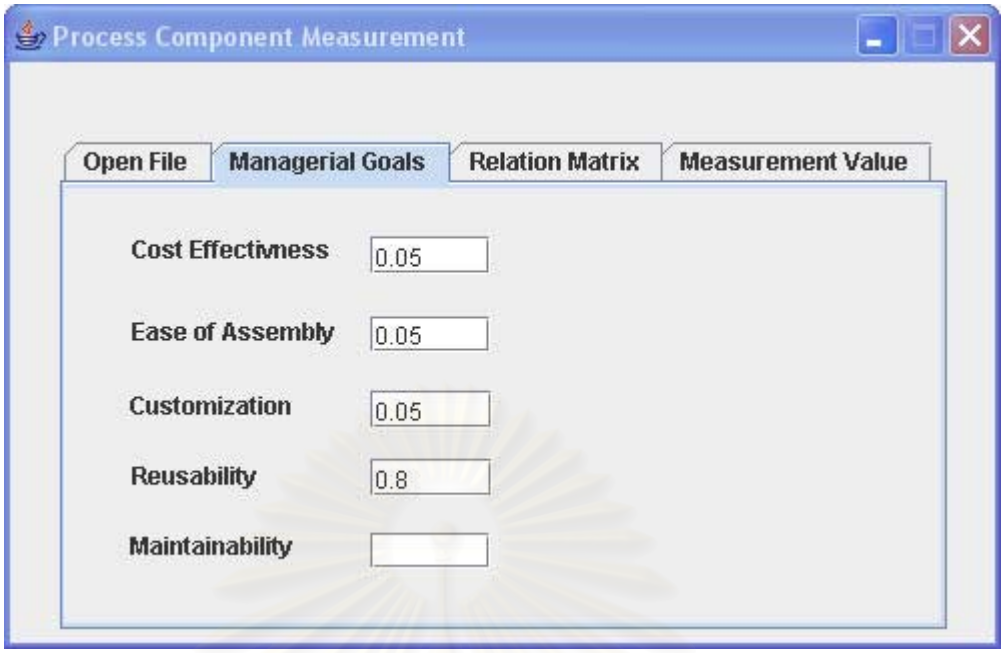
รูปที่ ค.4 หน้าจอ”Open”



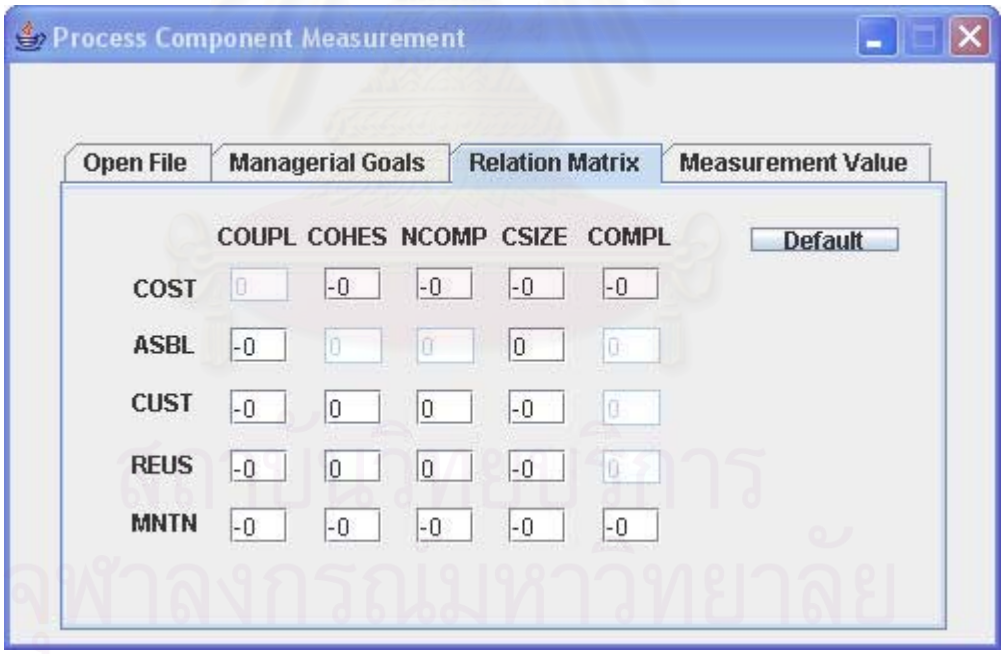
รูปที่ ค.5 หน้าจอ “Open” เมื่อเลือกแฟ้มข้อมูลเอ็กซ์เอ็มไอ



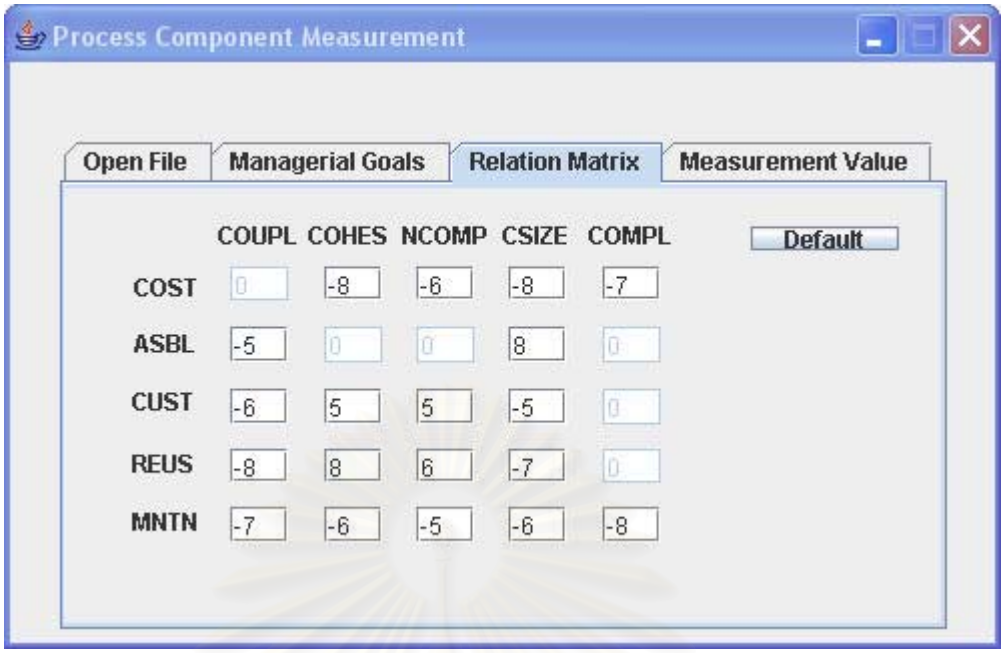
รูปที่ ค.6 หน้าจอเครื่องมือเมื่อแสดงลักษณะทางเทคนิค



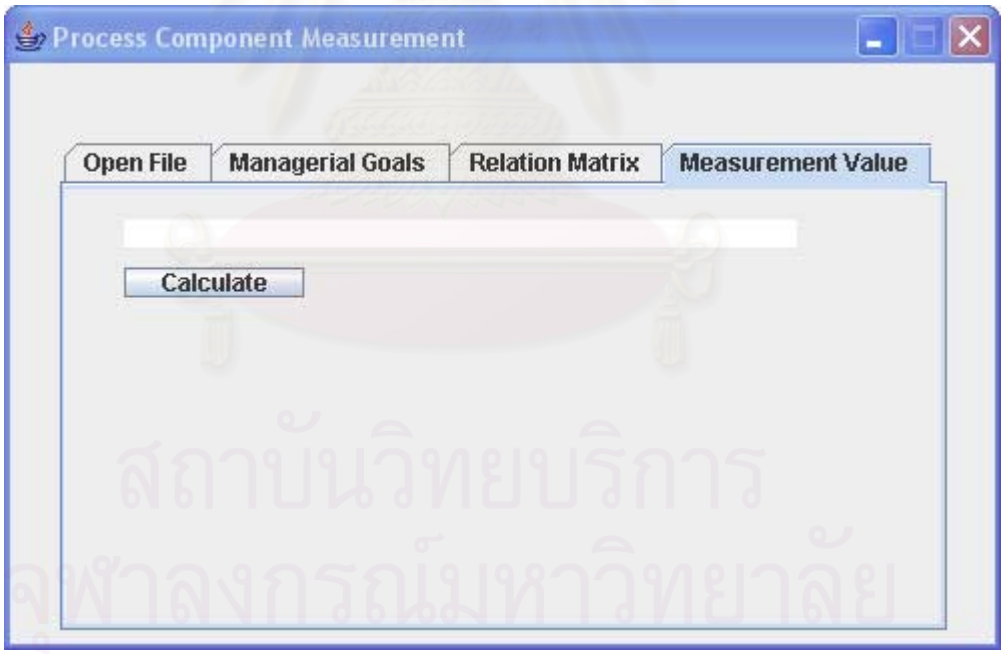
รูปที่ ค.7 หน้าจอเครื่องมือเมื่อเลือกแท็บ “Managerial Goals”



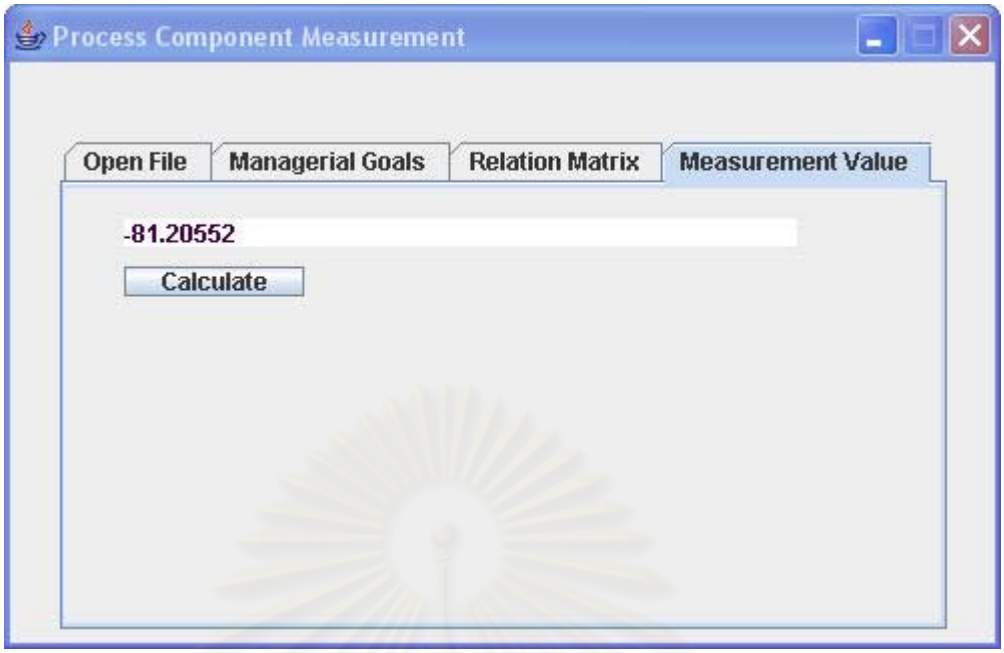
รูปที่ ค.8 หน้าจอเครื่องมือเมื่อเลือกแท็บ “Relation Matrix”



รูปที่ ค.9 หน้าจอเครื่องมือเมื่อเลือกแท็บ “Relation Matrix” และกดปุ่ม “Default”



รูปที่ ค.10 หน้าจอเครื่องมือเมื่อเลือกแท็บ “Measurement Value”



รูปที่ ค.11 หน้าจอเครื่องมือเมื่อแสดงค่าดัชนี

ภาคผนวก ง

ผลงานตีพิมพ์

ผลงานตีพิมพ์ซึ่งเป็นส่วนหนึ่งของงานวิจัยมีดังนี้

1. Proceedings 12th International Conference on Computer Science 2006 (ICCS2006), 29-31 March 2006, Vienna, Austria ในบทความเรื่อง Measuring Process Component Design on Achieving Managerial Goals โดยผู้แต่งคือ Eakong Atiptamvaree และ Twittie Senivongse

2. International Journal of Information Technology (IJIT2006), Volume 3 Number 2, 2006 ในบทความเรื่อง A Quantitative Approach to Strategic Design of Component-Based Business Process Models โดยผู้แต่งคือ Eakong Atiptamvaree และ Twittie Senivongse

Measuring Process Component Design on Achieving Managerial Goals

Eakong Atiptamvaree and Twittie Senivongse

***Abstract*—**Process-oriented software development is a new software development paradigm in which software design is modeled by a business process which is in turn translated into a process execution language for execution. The building blocks of this paradigm are software units that are composed together to work according to the flow of the business process. This new paradigm still exhibits the characteristic of the applications built with the traditional software component technology. This paper discusses an approach to apply a traditional technique for software component fabrication to the design of process-oriented software units, called process components. These process components result from decomposing a business process of a particular application domain into subprocesses, and these process components can be reused to design the business processes of other application domains. The decomposition considers five managerial goals, namely cost effectiveness, ease of assembly, customization, reusability, and maintainability. The paper presents how to design or decompose process components from a business process model and measure some technical features of the design that would affect the managerial goals. A comparison between the measurement values from different designs can tell which process component design is more appropriate for the managerial goals that have been set. The proposed approach can be applied in Web Services environment which accommodates process-oriented software development.

***Keywords*—**Business Process Model, Managerial Goals, Process Component.

I. Introduction

PROCESS-ORIENTED software development is a new software development paradigm in which the application domain is modeled as a business process model (BPM) [1]. A BPM describes the control flow of the operational process of the business with business rules incorporated. Users or business analysts who are familiar with the nature of the business can easily model their business with BPMs. This is opposed to modeling with software models such as UML [2] which requires expertise of software designers. Since BPMs correspond to process flow, software designers can rapidly develop applications by mapping BPMs to a workflow execution language and have them run with a supporting execution engine. The building blocks of this paradigm are software units that are composed together to work according to the BPM. Current software technology such as Web Services technology [3] realizes this paradigm with software building blocks called Web Services that can be composed by using a process execution language called BPEL [4].

By promoting reuse of software units across the design and development of different application domains, process-oriented paradigm is analogous to building applications with traditional software component technology [5]. We therefore see a potential to apply a traditional software component technique to process-oriented software development.

In this paper, we look from the component supplier's point of view and focus on a software component fabrication technique (i.e., how to develop components). Traditionally, component-based software is modeled by using UML class diagram and the model is decomposed into small units in order to implement them as reusable software components. Analogously in process-oriented paradigm, an application domain is modeled by a BPM which can be decomposed into subprocesses called process components [6]. These process components can be reused in the design of the BPMs of other application domains, and can also be implemented into reusable software units. This idea corresponds to the concept of process patterns [7].

This paper applies a software component fabrication technique presented in [8] to fabricate process components. The technique in [8] starts by modeling an application domain with a UML class diagram and dividing the classes within the diagram into groups, each group referring to a software component (to be implemented). Such grouping can be seen as one way to design software components for the application domain. Technical features are measured from the design, namely intercomponent coupling, intracomponent cohesion, number of components, component size, and complexity. The resulting measurement values are applied onto a mathematical model, called the Business Strategy-Based Component Design (BusCod) model, to determine how well such a software component design can achieve predefined managerial goals (i.e., cost effectiveness, ease of assembly, customization, reusability, and maintainability). We are particularly interested in this technique because it can give qualitative measurement that reflects quality of the design while also considering business strategies.

In this paper, we will model a particular application domain with a BPM and decompose the BPM into process components so that the technique in [8] can be applied. Software designers can try to design process components for a

E. Atiptamvaree is with the Information Systems Engineering Laboratory, Department of Computer Engineering, Chulalongkorn University, Bangkok 10330 Thailand (phone: +66 2 2186991; fax: +66 2 2186955; e-mail: Eakong.A@student.chula.ac.th).

T. Senivongse is with the Information Systems Engineering Laboratory, Department of Computer Engineering, Chulalongkorn University, Bangkok 10330 Thailand (phone: +66 2 2186996; fax: +66 2 2186955; e-mail: twittie.s@chula.ac.th).

particular domain in different ways (i.e., with different grouping) and compare the measurement values given by the BusCod model to determine which design (i.e., which grouping) better suits the managerial goals that have been set.

This paper has the following organization. Section II discusses some related work. Section III presents our approach to apply the technique in [8] to a BPM of a particular application domain. A flight reservation system is the case study in Section IV and we give two designs of process components for this domain as an example. The paper concludes with some discussion in Section V.

II. Related Work

Process-oriented software development tends to focus on the approaches to map BPMs to a process execution language for a particular software technology. Reference [1] addresses a correspondence between BPM and Web Services technology. The work in [9] realizes this approach by modeling a BPM with ADF or UML activity diagrams and transforming them to BPEL for execution. However such an approach attacks the implementation issue of the BPM, not the design and reuse issue.

A number of researches have addressed the concept of reusable business processes. In [6], a model for reusable business processes is proposed. A business process will be associated with information such as process function, process interface, and quality of service. Such information is for component cataloguing and assembly purposes. In [7], a process pattern is used in the design of a software application but the work does not address how the pattern is designed.

For software component fabrication, the technique in [10] is close to the one in [8] which we will adopt. Software in [10] is also modeled using a UML class diagram and the model is decomposed according to different criteria such as data usage and business functions. Metrics based on cohesion and coupling of software components are proposed to measure the quality of the decomposition. However, the technique does not take managerial goals into account and consider less technical features than [8].

III. Measuring Process Component Design

We apply the technique in [8] to a BPM of an application domain as follows.

A. Design of Process Components

In [8], an application domain is modeled with a UML class diagram. The classes link with each other by association lines which represent relationships or connectivity that a class has with other classes. We can design software components by grouping together the classes that exhibit high degree of relationship or association with each other. Analogously for a BPM such as a UML activity diagram in Fig. 1, the actions link with each other by control flow arrows. The arrows represent relationships or association between actions in terms of data coupling (i.e., data that are passed through) or time requirement (i.e., actions occur at the same instant of time) [10]. Therefore we can design process components by grouping together the actions that exhibit high degree of relationship with each other.

A software designer will try to identify which part of the business process should be together as a process component for the domain. In Fig. 1, a software designer divides the actions in the activity diagram of the business process into three groups; each group is referred as an activity or a process component. Note that this is only one design for process components; the software designer can group the actions differently to create other designs.

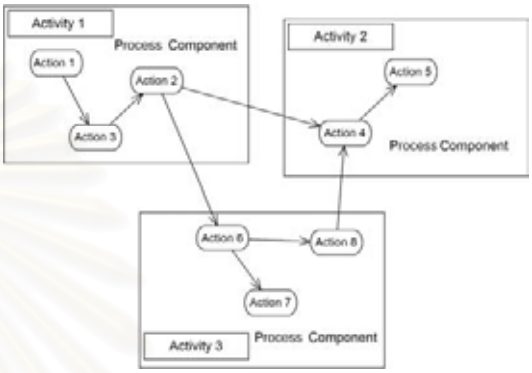


Fig. 1 Business process designed with three process components

B. Managerial Goals

Five managerial goals for the design of software components [8] can be adopted for the design of process components:

1) Cost Effectiveness (COST)

Cost effectiveness encompasses minimal component development cost and reduction in design and development time. This goal is important for achieving low cost business strategy. In component fabrication, costs depend on the actions included in the process component and the relationships among these actions.

2) Ease of Assembly (ASBL)

Ease of assembly refers to the ease with which components can be assembled. Reduction in the number of components required to assemble an application can enhance the assembly process since larger components will incorporate more functionality while complexities remain internal to the components. This goal is important for serving application developers who do not expect technical complexity at assembly.

3) Customization (CUST)

Customization is the ability to allow the application developers or assemblers to fit and alter solutions for a large variety of business applications using the components. This goal is important if the business competes in the market of customizable applications.

4) Reusability (REUS)

Reusability is the ability to reuse a component as is, without

modification, in the development of various applications. Reusability also implies quality (i.e., conformance to requirements) and reliability (i.e., the ability to be depended on to correctly perform the function). The goal is important if the application developers are to be provided with components that can be used in many applications.

5) Maintainability (MNTN)

Maintainability is the ease with which the components can be added, deleted, or modified. This goal is important to low cost business strategy as maintenance may represent a long-term cost.

C. Technical Features

Five technical features for the design of software components [8] can be adapted for the design of process components:

1) Intercomponent Coupling (COUPL)

Intercomponent coupling is the strength of relationship between different components and low coupling is desired. In [8] where the application is designed with a UML class model, coupling is defined as the extent to which classes within a component relate in any way to other classes that are not in that component (e.g., by method invocation or by having the other classes as data types for attributes or method parameters of the class). For process components, actions in the activity diagram corresponds to classes in [8], and therefore coupling is defined as data coupling in terms of a control flow that carries data from one action in one component to the other action in the other component [11], or as time coupling by which two actions will occur together at an instant of time [10]. The measure for intercomponent coupling is as follows:

$$COUPL = \sum_{k=1}^{y} \sum_{i=1}^{n} \sum_{\substack{j=1 \\ i \neq j}}^{n} (x_{ik} * (1 - x_{jk}) * c_{ij}) \quad (1)$$

where

y number of process components for the domain;
n total number of actions in the domain model;
x_{ik} 1 if action *i* is placed in process component *k*;
0 if action *i* is not placed in process component *k*;
c_{ij} coupling between action *i* and *j*, ($i,j \geq 0$; $i \neq j$).
Coupling between two actions can be measured from the number of control flow between them.

2) Intracomponent Cohesion (COHES)

Intracomponent cohesion is the strength of relationship within the component and high cohesion is desired. In [8], cohesion is defined as the extent to which classes within a component relate in any way to other classes within that component. For process components, cohesion is defined in terms of the control flow between actions of the same component. The measure for intracomponent cohesion is as follows:

$$COHES = \sum_{k=1}^{y} \sum_{i=1}^{n} \sum_{\substack{j=1 \\ i \neq j}}^{n} ((x_{ik} * x_{jk}) * c_{ij}) \quad (2)$$

where

y number of process components for the domain;
n total number of actions in the domain model;
x_{ik} 1 if action *i* is placed in process component *k*;
0 if action *i* is not placed in process component *k*;
c_{ij} coupling between action *i* and *j*, ($i,j \geq 0$; $i \neq j$).

3) Number of Components (NCOMP)

Number of components represents the number of interface between components and complexity of that design, but a large number of components also give the application developers more choices in selecting the component that will closely satisfy the user requirement. The measure for number of components is as follows:

$$NCOMP = y \quad (3)$$

where

y number of process components for the domain.

4) Component Size (CSIZE)

Component size represents the granularity of the component set of the design. In [8], the measure for component size uses statistical standard deviation, instead of average size, to take into account the variability in the number of classes in different components. For process components, a similar normalized measure can be adopted as follows:

$$CSIZE = \sqrt{\sum_{j=1}^{y} \left(\sum_{i=1}^{n} x_{ij} \right)^2 / y} \quad (4)$$

where

y number of process components, $y > 0$;
n total number of actions, $n \geq 0$;
x_{ij} 1 if action *i* is placed in process component *j*;
0 if action *i* is not placed in process component *j*.

5) Complexity (COMPL)

The number of actions in a process component can provide a coarse-level complexity measure. In [8], complexity of a component is determined by the number of public methods and method parameters of classes within the component. Instead of a simple addition of the number of methods and parameter complexities across all components, the measure takes into account the variability of complexity between different designs by which the classes are distributed differently among the components. For process components, a similar measure can be adopted by looking at each action in the activity diagram as a single abstract operation that may take some input parameter data from the previous action in the process flow and produce some output parameter data to be passed onto the next action in the flow. This agrees with standard UML which allows a class method to associate with an action in an activity diagram [2]. However, the software designer will have to provide such operation details for the business process. The measure is as follows:

$$COMPL = \sum_{j=1}^{y} \left(\sum_{i=1}^{n} \left((w_{mcx} * m_i) + \left(w_{pcx} * \left(\sum_{k=1}^{m_i} \sum_{l=1}^{p_{ik}} p_{ikl} \right) \right) \right) * x_{ij} \right)^2 \quad (5)$$

where

y number of process components, $y > 0$;
n total number of actions, $n \geq 0$;

x_{ij} 1 if action i is placed in process component j;
0 if action i is not placed in process component j;
w_{mcx} relative importance of operation complexity, $(0 \geq w_{mcx} \leq 1)$;
m_i number of operation in action i, $(m_i = 1)$;
w_{pcx} relative importance of parameter complexity, $(0 \geq w_{pcx} \leq 1)$;
p_{ik} number of parameters in operation k in action i, $(p_{ik} \geq 0)$;
p_{ikl} relative complexity of the parameter l in operation k in action i, $(0 \geq p_{ikl} \leq 1)$.

The values for w_{mcx}, w_{pcx}, and p_{ikl} are subjective and assigned by the software designer during the design process. w_{mcx} and w_{pcx} can be assigned based on the complexity the software designer expects for the operations and parameters respectively. If the computation of the operations is likely to be complex in order to serve purpose of the actions, w_{mcx} may have high value. If it is necessary that the operations need a lot of complex parameters (e.g., those of complex data types) for their computation, then w_{pcx} may have high value. p_{ikl} is the degree of complexity of a parameter of an operation compared to that of other parameters of the same operation. A parameter is complex if, for example, it is of a complex data type.

Table I is a summary of a literature survey on the impact the technical features may have on the managerial goals [8]. Positive impact (+) means the higher the technical feature value is, the better the goal is achieved. Negative impact (-) means the lower the technical feature is, the better the goal is achieved. No impact (0) means the technical feature has no relation to achieving the goal.

TABLE I
IMPACT OF TECHNICAL FEATURES ON MANAGERIAL GOALS

	COUPL	COHES	NCOMP	CSIZE	COMPL
COST	0	-	-	-	-
ASBL	-	0	0	+	0
CUST	-	+	+	-	0
REUS	-	+	+	-	0
MNTN	-	-	-	-	-

D. Applying BusCod Model

We can adopt the BusCod model in [8]:

$$R' * W * D \quad (6)$$

Each part of the model is as follows.

1) $R' = [R_{COST} \quad R_{ASBL} \quad R_{CUST} \quad R_{REUS} \quad R_{MNTN}]$

This is a vector representing relative importance of all managerial goals for the design where

R_{COST} relative importance of cost effectiveness;
R_{ASBL} relative importance of ease of assembly;
R_{CUST} relative importance of customization;
R_{REUS} relative importance of reusability;
R_{MNTN} relative importance of maintainability;
and $R_{COST} + R_{ASBL} + R_{CUST} + R_{REUS} + R_{MNTN} = 1$.

2)

$$W = \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} & w_{15} \\ w_{21} & w_{22} & w_{23} & w_{24} & w_{25} \\ w_{31} & w_{32} & w_{33} & w_{34} & w_{35} \\ w_{41} & w_{42} & w_{43} & w_{44} & w_{45} \\ w_{51} & w_{52} & w_{53} & w_{54} & w_{55} \end{bmatrix}$$

This is a matrix representing the strength of association between managerial goals and technical features where

w_{ij} strength of the association between i^{th} managerial goal and j^{th} technical feature in W. The value is either 0 or 1-10 with the sign (+ or -) as in Table I.

3)

$$D = \begin{bmatrix} D_{COUPL} \\ D_{COHES} \\ D_{NCOMP} \\ D_{CSIZE} \\ D_{COMPL} \end{bmatrix}$$

This is a vector representing measurement of various technical features of the design where

D_{COUPL} intercomponent coupling;
D_{COHES} intracomponent cohesion;
D_{NCOMP} number of components;
D_{CSIZE} component size;
D_{COMPL} complexity.

E. Design Process

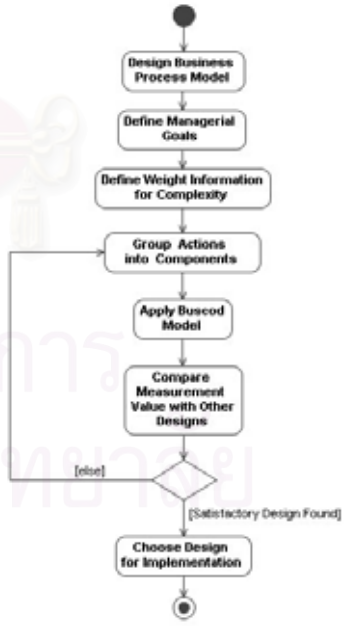


Fig. 2 Process for designing process component

Fig. 2 describes the process a software designer takes to design process components. This process will be supported by a software tool.The software designer obtains a BPM from a business analyst and defines relative importance of each managerial goal for the design as well as weight information

for complexity calculation. Then, the BPM – a UML activity diagram in this case – will be decomposed into groups of actions; each group corresponds to a package. The packages will be processed by the tool to apply the BusCod model. The software designer can repeat this process with different designs and compare their measurement values from the BusCod model. The design with maximum measurement value will best achieve the managerial goals that have been set and can be chosen for implementation.

IV. Case Study

An example business process model used to demonstrate the design of process components is of the flight reservation domain, involving an airline (Fig. 3). We define this model by adapting from the Open Travel Alliance (OTA) specification [12] and the case study of [13]. The flow begins with checking a flight for a particular trip. Other details (i.e., schedule, arrival time, seat availability, and price) are checked subsequently. If seats are available and price is in budget, the flight is booked, the seat numbers are confirmed, and the itinerary is produced. In the case that no seats are available, this failed booking can be recorded for administrative purpose (e.g., to increase the number of flights during some period of the year).

Fig. 3 Business process of flight reservation domain

According to the BusCod model, a software designer may want to design process components for this process flow with an emphasis on reusability. The managerial goals may be set as

$$R' = \begin{bmatrix} 0.05 & 0.05 & 0.05 & 0.8 & 0.05 \end{bmatrix}$$

For this example, we use the matrix W below:

$$W = \begin{bmatrix} 0 & -8 & -6 & -8 & -7 \\ -5 & 0 & 0 & +8 & 0 \\ -6 & +5 & +5 & -5 & 0 \\ -8 & +8 & +6 & -7 & 0 \\ -7 & -6 & -5 & -6 & -8 \end{bmatrix}$$

This matrix is derived from a survey on the strength of relationship between the managerial goals and the technical features, conducted on industry experts [8].

Suppose the software designer decomposes the above process flow into three process components (Fig. 4). We may look at them as flight inquiry component, flight booking component, and failed booking component. For later presentation purpose, we also annotate each action with a symbol A1-A9. This design then has its technical features calculated. We discuss the calculation of some values below.

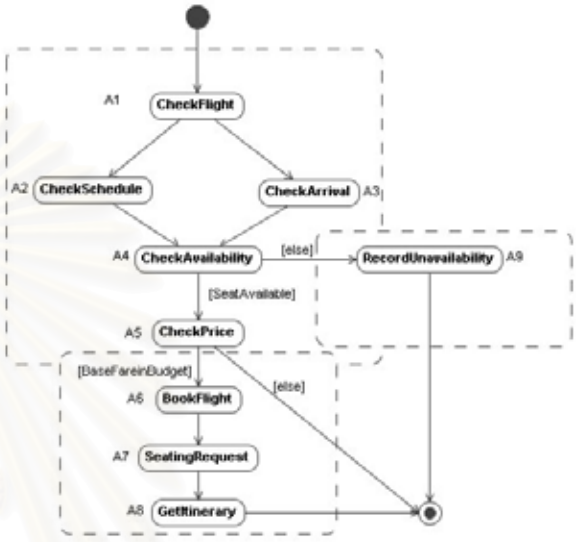


Fig. 4 A design with three process components

This design has 3 process components and 9 actions in total.

For intercomponent coupling and intracomponent cohesion, the value c_{ij} which is the coupling between any two actions in the process model is summarized in Table II.

TABLE II
Coupling between Each Action

	A1	A2	A3	A4	A5	A6	A7	A8	A9
A1	0								
A2	1	0							
A3	1	0	0						
A4	0	1	1	0					
A5	0	0	0	1	0				
A6	0	0	0	0	1	0			
A7	0	0	0	0	0	1	0		
A8	0	0	0	0	0	0	1	0	
A9	0	0	0	1	0	0	0	0	0

For complexity, the software designer specifies operation detail for each action (Table III). As mentioned in Section III.C, we can see each action as a single abstract operation. The design of the operations here is based on the technique to design service interfaces in [13] which considers elementary business functions and applies data normalization to interface parameters. This leads to minimization of coupling and maximization of cohesion of the operations.

In this example, the software designer also assigns a value 0.5 for the relative importance of operation complexity w_{mcx} and a value 0.5 for the relative importance of parameter complexity w_{pcx}. For each parameter in each operation, a value 1 is assigned for the relative complexity p_{ikl} as the parameters are equally complex (i.e., all are simple parameters).

The values of all technical features are:

$$D = \begin{bmatrix} 2 \\ 7 \\ 3 \\ 3.416 \\ 318.5 \end{bmatrix}$$

The calculation of the BusCod model $R^{'} * W * D$ for this three-component design gives the value -219.33. If we repeat the design process to calculate the BusCod value for a one-component design (i.e., the whole process flow is seen as one component), the BusCod value is -544.05. Therefore, the three-component design better achieves the managerial goals that have been set. With an emphasis on the reusability goal, we may reason that the one-component design is less appropriate for reuse because it carries too much functionality.

TABLE III
ACTIONS AND THEIR PARAMETERS

Action/Operation (No. of Parameters)	Input Parameters	Output Parameters
CheckFlight (4)	OriginalLocation DestinationLocation DepartureDate	FlightNumber
CheckSchedule (5)	FlightNumber	DepartureAirport DepartureTime ArrivalAirport ArrivalTime
CheckArrival (3)	FlightNumber DepartureDate	ArrivalDate
CheckAvailability (4)	FlightNumber DepartureDate CabinType	NoOfSeats
CheckPrice (6)	FlightNumber DepartureDate CabinType Budget	FareBasisCode BaseFare
BookFlight (5)	FlightNumber DepartureDate TravelerName CabinType	BookingReferenceID
SeatingRequest (3)	BookingReferenceID SeatPreference	SeatNumber
GetItinerary (12)	BookingReferenceID	BookingReferenceID FlightNumber DapartureAirport DepartureDate DepartureTime ArrivalAirport ArrivalDate ArrivalTime CabinType BookingStatus JourneyDuration
RecordUnavailability (3)	FlightNumber DepartureDate CabinType	

V. CONCLUSION

We discuss a possibility to apply a software component fabrication technique to design process components for an application domain which is modeled by a business process model. The paper relies on the BusCod model, the efficiency of which has been evaluated by industry experts as reported in [8]. Once a satisfactory design is found, the software designer can reuse each process component in the design of other business processes, and can have it implemented into a software unit. To implement a process component, we can follow the process-oriented paradigm and map each process component into BPEL [9], or we may follow the traditional paradigm and map each process component into a UML class diagram for component development [14].

This technique requires to a certain extent the skill of the software designer to group process components and to assign weight information for complexity. Other guidelines can help the software designer to decide which actions should be in the same process component (e.g., to reduce coupling and increase cohesion) [10].

The supporting tool for this technique is still at its early stage; it can only give a BusCod value of a particular design. An enhancement is expected such that the tool can give an optimal design for a particular business process model.

REFERENCES

[1] BPMI.org. (2004, May, 3). *Business Process Modeling Notation (BPMN)* Version 1.0. Availalble: http://www.bpmi.org

[2] G. Booch, J. Rumbaugh, and I Jacobson, *The Unified Modeling Language User Guide*, Massachusetts: Addison-Wesley, 1999.

[3] E. Newcomer, Understanding Web Services. Indianapolis: Addison-Wesley, 2002.

[4] IBM. (2003, May, 5) *BPEL4WS Version 1.1 specification* Available: http://www-128.ibm.com/developerworks/library/specification/ws-bpel/

[5] C. Szyperski, *Component Software: Beyond Object-Oriented Programming*, 2nd ed. New York: Addison-Wesley, 2002.

[6] Y. Mou, J. Cao, and S. Zhang, "A process component model for enterprise business knowledge reuse," in *Proc. IEEE International Conference on Services Computing (SCC04),* 2004.

[7] O. H. Barros. (2004, September). *Business Information System Design Based on Process Pattern and Frameworks*. Industrial Engineering Department, University of Chile. Available: http://www.BPtrends.com

[8] P. Vitharana, H. Jain, and F. M. Zahedi, "Strategy-based design of reusable business components," *IEEE Transactions on Systems, Man, and Cybernetics–Part C: Applications and Reviews*, vol. 34, No. 4, pp. 460-474, November 2004.

[9] J. Koehler, R. Hauser, S. Kapoor, F. Y. Wu, and S. Kumaran, "A model-driven transformation method," in *Proc. 7th IEEE International Enterprise Distributed Object Computing Conference*, 16-19 September 2003, pp. 186-197.

[10] N. Tagoug, "Object-oriented system decomposition quality", in *Proc. 7th International Symposium on High Assurance Systems Engineering (HASE02)*, 2002.

[11] B. Husslage, E. van Dam, D. den Hertog, P. Stehouwer, and E. Stinstra, *Coordination of coupled black box simulations in the construction of metamodels*, Discussion Paper 2, Center for Economic Research, Tilburg University, 2003.

[12] Open Travel Alliance (OTA). (2005, December, 05). *OTA Specification 2005B*. Available: http://www.opentravel.org

[13] G. Feuerlicht and S. Meesathit, "Design framework for interoperable service interfaces," in *Proc. 2nd International Conference on Service Oriented Computing (ICSOC'04)*, New York, 2004, pp. 299-307.

[14] W. Rungworawut and T. Senivongse, "A guideline to mapping business process to UML class diagrams," *WSEAS Transactions on Computers,* vol. 4, November 2005.

A Quantitative Approach to Strategic Design of Component-Based Business Process Models

Eakong Atiptamvaree and Twittie Senivongse

***Abstract*—**A new paradigm for software design and development models software by its business process, translates the model into a process execution language, and has it run by a supporting execution engine. This process-oriented paradigm promotes modeling of software by less technical users or business analysts as well as rapid development. Since business process models may be shared by different organizations and sometimes even by different business domains, it is interesting to apply a technique used in traditional software component technology to design reusable business processes. This paper discusses an approach to apply a technique for software component fabrication to the design of process-oriented software units, called process components. These process components result from decomposing a business process of a particular application domain into subprocesses with an aim that the process components can be reusable in different process-based software models. The approach is quantitative because the quality of process component design is measured from technical features of the process components. The approach is also strategic because the measured quality is determined against business-oriented component management goals. A software tool has been developed to measure how good a process component design is, according to the required managerial goals and comparing to other designs. We also discuss how we benefit from reusable process components.

***Keywords*—**Business Process Model, Process Component, Component Management Goals, Measurement.

I. Introduction

PROCESS-ORIENTED software development is a new software development paradigm in which the application domain is modeled as a business process model (BPM) [1]. A BPM describes the control flow of the operational process of the business with business rules incorporated. Users or business analysts who are familiar with the nature of the business can easily model their business with BPMs. This is opposed to modeling with software models such as UML [2] which requires expertise of software designers. Since BPMs correspond to process flow, software designers can rapidly develop applications by mapping BPMs to a workflow execution language and have them run with a supporting execution engine. The building blocks of this paradigm are software units that are composed together to work according to the BPM. Current software technology such as Web Services technology [3] realizes this paradigm with software building blocks called Web Services that can be composed by using a process execution language called BPEL [4].

By promoting reuse of software units across the design and development of different application domains, process-oriented paradigm is analogous to building applications with traditional software component technology [5]. We therefore see a potential to apply a traditional software component technique to process-oriented software development.

In this paper, we look from the component supplier's point of view and focus on a software component fabrication technique (i.e., how to develop components). Traditionally, component-based software is modeled by using UML class diagram and the model is decomposed into small units in order to implement them as reusable software components. Analogously in process-oriented paradigm, an application domain is modeled by a BPM which can be decomposed into subprocesses called process components [6]. These process components can be reused in the design of the BPMs of other businesses or application domains, and can also be implemented into reusable software units. This idea corresponds to the concept of process patterns [7].

This paper applies a software component fabrication technique presented in [8] to fabricate process components. The technique in [8] starts by modeling an application domain with a UML class diagram and dividing the classes within the diagram into groups, each group referring to a software component (to be implemented). Such grouping can be seen as one way to design software components for the application domain. Technical features are measured from the design, namely intercomponent coupling, intracomponent cohesion, number of components, component size, and complexity. The resulting measurement values are applied onto a mathematical model, called the Business Strategy-Based Component Design (BusCod) model, to determine how well such a software component design can achieve predefined managerial goals (i.e., cost effectiveness, ease of assembly, customization, reusability, and maintainability). We are particularly interested in this technique because it can give quantitative measurement that reflects quality of the design while also considering business strategies.

In this paper, we will model a particular application domain with a BPM and decompose the BPM into process components so that the technique in [8] can be applied. Software designers can try to design process components for a particular domain in different ways (i.e., with different

E. Atiptamvaree is with the Information Systems Engineering Laboratory, Department of Computer Engineering, Chulalongkorn University, Bangkok 10330 Thailand (phone: +66 2 2186991; fax: +66 2 2186955; e-mail: Eakong.A@student.chula.ac.th).

T. Senivongse is with the Information Systems Engineering Laboratory, Department of Computer Engineering, Chulalongkorn University, Bangkok 10330 Thailand (phone: +66 2 2186996; fax: +66 2 2186955; e-mail: twittie.s@chula.ac.th; corresponding author).

grouping) and compare the measurement values given by the BusCod model to determine which design (i.e., which grouping) better suits the component management goals that have been set.

This paper has the following organization. Section II discusses some related work. Section III presents our approach to apply the technique in [8] to a BPM of a particular application domain. A flight reservation system is the case study in Section IV and we give two designs of process components for this domain as an example. A discussion about the benefit of reusable process components is in Section V. Section VI presents the supporting tool for component measurement. The paper concludes and discusses future research directions in Section VII.

II. Related Work

Process-oriented software development tends to focus on the approaches to map BPMs to a process execution language for a particular software technology. Reference [1] addresses a correspondence between BPM and Web Services technology. The work in [9] realizes this approach by modeling a BPM with ADF or UML activity diagrams and transforming them to BPEL for execution. However such an approach attacks the implementation issue of the BPM, not the design and reuse issue.

A number of researches have addressed the concept of reusable business processes. In [6], a model for reusable business processes is proposed. A business process will be associated with information such as process function, process interface, and quality of service. Such information is for component cataloging and assembly purposes. In [7], a process pattern is used in the design of a software application but the work does not address how the pattern is designed.

For software component fabrication, the technique in [10] is close to the one in [8] which we will adopt. Software in [10] is also modeled using a UML class diagram and the model is decomposed according to different criteria such as data usage and business functions. Metrics based on cohesion and coupling of software components are proposed to measure the quality of the decomposition. However, the technique does not take managerial goals into account and consider less technical features than [8].

III. Measuring Process Component Design

We apply the technique in [8] to a BPM of an application domain as follows.

A. Design of Process Components

In [8], an application domain is modeled with a UML class diagram. The classes link with each other by association lines which represent relationships or connectivity that a class has with other classes. We can design software components by grouping together the classes that exhibit high degree of relationship or association with each other (e.g., Fig. 1). Analogously for a BPM such as a UML activity diagram in Fig. 2, the actions link with each other by control flow arrows. The arrows represent relationships or association between actions in terms of data coupling (i.e., data that are passed through) or time requirement (i.e., actions occur at the same instant of time) [10]. Therefore we may design process components by grouping together the actions that exhibit high degree of relationship with each other. Table I summarizes the correspondence between class diagram structure and BPM structure.

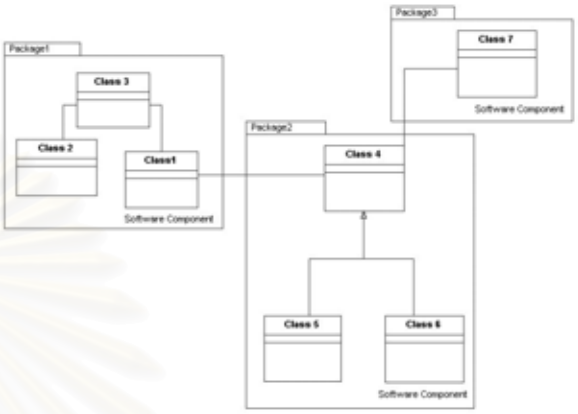


Fig. 1 Class model designed with three software components

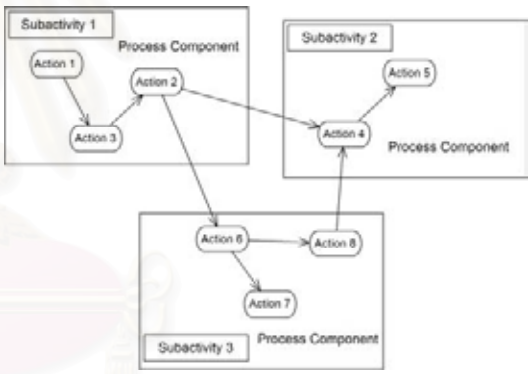


Fig. 2 Business process designed with three process components

TABLE I
Correspondence between structures of class diagram and BPM

Class diagram	BPM (activity diagram)
Class	Action
Link (inheritance or association)	Arrow (control flow)
Package	Subactivity

To design process components, a software designer will identify which part of the business process should be together as a process component for the domain. In Fig. 2, a software designer divides the actions in the activity diagram of the business process into three groups; each group is referred as a subactivity or a process component. Note that this is only one design for process components; the software designer can group the actions differently to create other designs.

B. Component Management Goals

Five component management goals for the design of software components [8] can be adopted for the design of

process components:

1) Cost Effectiveness (COST)

Cost effectiveness encompasses minimal component development cost and reduction in design and development time. This goal is important for achieving low cost business strategy. In component fabrication, costs depend on the actions included in the process component and the relationships among these actions.

2) Ease of Assembly (ASBL)

Ease of assembly refers to the ease with which components can be assembled. Reduction in the number of components required to assemble an application can enhance the assembly process since larger components will incorporate more functionality while complexities remain internal to the components. This goal is important for serving application developers who do not expect technical complexity at assembly.

3) Customization (CUST)

Customization is the ability to allow the application developers or assemblers to fit and alter solutions for a large variety of business applications using the components. This goal is important if the business competes in the market of customizable applications.

4) Reusability (REUS)

Reusability is the ability to reuse a component as is, without modification, in the development of various applications. Reusability also implies quality (i.e., conformance to requirements) and reliability (i.e., the ability to be depended on to correctly perform the function). The goal is important if the application developers are to be provided with components that can be used in many applications.

5) Maintainability (MNTN)

Maintainability is the ease with which the components can be added, deleted, or modified. This goal is important to low cost business strategy as maintenance may represent a long-term cost.

C. Technical Features

Five technical features for the design of software components [8] can be adapted for the design of process components:

1) Intercomponent Coupling (COUPL)

Intercomponent coupling is the strength of relationship between different components and low coupling is desired. In [8] where the application is designed with a UML class model, coupling is defined as the extent to which classes within a component relate in any way to other classes that are not in that component (e.g., by method invocation or by having the other classes as data types for attributes or method parameters of the class). For process components, actions in the activity diagram corresponds to classes in [8], and therefore coupling is defined as data coupling in terms of a control flow that carries data from one action in one component to the other action in the other component [11], or as time coupling by which two actions will occur together at an instant of time [10]. The measure for intercomponent coupling is as follows:

$$COUPL = \sum_{k=1}^{y} \sum_{i=1}^{n} \sum_{\substack{j=1 \\ i \neq j}}^{n} (x_{ik} * (1 - x_{jk}) * c_{ij}) \quad (1)$$

where

y number of process components for the domain;
n total number of actions in the domain model;
x_{ik} 1 if action *i* is placed in process component *k*;
0 if action *i* is not placed in process component *k*;
c_{ij} coupling between action *i* and *j*, ($i,j \geq 0$; $i \neq j$).
Coupling between two actions can be measured from the number of control flow between them.

2) Intracomponent Cohesion (COHES)

Intracomponent cohesion is the strength of relationship within the component and high cohesion is desired. In [8], cohesion is defined as the extent to which classes within a component relate in any way to other classes within that component. For process components, cohesion is defined in terms of the control flow between actions of the same component. The measure for intracomponent cohesion is as follows:

$$COHES = \sum_{k=1}^{y} \sum_{i=1}^{n} \sum_{\substack{j=1 \\ i \neq j}}^{n} ((x_{ik} * x_{jk}) * c_{ij}) \quad (2)$$

where

y number of process components for the domain;
n total number of actions in the domain model;
x_{ik} 1 if action *i* is placed in process component *k*;
0 if action *i* is not placed in process component *k*;
c_{ij} coupling between action *i* and *j*, ($i,j \geq 0$; $i \neq j$).

3) Number of Components (NCOMP)

Number of components represents the number of interface between components and complexity of that design, but a large number of components also give the application developers more choices in selecting the component that will closely satisfy the user requirement. The measure for number of components is as follows:

$$NCOMP = y \quad (3)$$

where

y number of process components for the domain.

4) Component Size (CSIZE)

Component size represents the granularity of the component set of the design. In [8], the measure for component size uses statistical standard deviation, instead of average size, to take into account the variability in the number of classes in different components. For process components, a similar normalized measure can be adopted as follows:

$$CSIZE = \sqrt{\sum_{j=1}^{y} \left(\sum_{i=1}^{n} x_{ij} \right)^2 / y} \quad (4)$$

where

y number of process components, $y > 0$;
n total number of actions, $n \geq 0$;

x_{ij} 1 if action *i* is placed in process component *j*;
0 if action *i* is not placed in process component *j*.

5) Complexity (COMPL)

The number of actions in a process component can provide a coarse-level complexity measure. In [8], complexity of a component is determined by the number of public methods and method parameters of classes within the component. Instead of a simple addition of the number of methods and parameter complexities across all components, the measure takes into account the variability of complexity between different designs by which the classes are distributed differently among the components. For process components, we may also adopt similar measurement. Since an action in the activity diagram corresponds to a UML class (c.f., Table I), we may look at each action as corresponding to a class with a single abstract operation which may take some input parameter data from the previous action in the process flow and produce some output parameter data to be passed onto the next action in the flow. This agrees with standard UML which allows a class method to associate with an action in an activity diagram [2]. However, the software designer will have to provide the details of such abstract operations for the business process. It is seen that the software designer can analyze the business process and can identify details (e.g., data that flow between actions). The measure is as follows:

$$COMPL = \sum_{j=1}^{y}\left(\sum_{i=1}^{n}\left(\left(w_{mcx} * m_i\right)+\left(w_{pcx} * \left(\sum_{k=1}^{m_i}\sum_{l=1}^{p_{ik}} p_{ikl}\right)\right)\right) * x_{ij}\right)^2 \quad (5)$$

where
y number of process components, $y > 0$;
n total number of actions, $n \geq 0$;
x_{ij} 1 if action *i* is placed in process component *j*;
0 if action *i* is not placed in process component *j*;
w_{mcx} relative importance of operation complexity, $(0 \geq w_{mcx} \leq 1)$;
m_i number of operation in action *i*, $(m_i = 1)$;
w_{pcx} relative importance of parameter complexity, $(0 \geq w_{pcx} \leq 1)$;
p_{ik} number of parameters in operation *k* in action *i*, $(p_{ik} \geq 0)$;
p_{ikl} relative complexity of the parameter *l* in operation *k* in action *i*, $(0 \geq p_{ikl} \leq 1)$.

The values for w_{mcx}, w_{pcx}, and p_{ikl} are subjective and assigned by the software designer during the design process. w_{mcx} and w_{pcx} can be assigned based on the complexity the software designer expects for the operations and parameters respectively. If the computation of the operations is likely to be complex in order to serve purpose of the actions, w_{mcx} may have high value. If it is necessary that the operations need a lot of complex parameters (e.g., those of complex data types) for their computation, then w_{pcx} may have high value. p_{ikl} is the degree of complexity of a parameter of an operation compared to that of other parameters of the same operation. A parameter is complex if, for example, it is of a complex data type.

By looking at an action as a class with an abstract operation, complexity measurement is refined. However, in most cases, an action in the activity diagram is modeled at high level and the software designer may find it inconvenient to analyze the details of the abstract operation and the complexity weights. In such cases, we propose a simplified formula for complexity based on the number of actions. This is analogous to the coarse-grained complexity measurement mentioned in [8] which is based on the number of classes in the class diagram. The simplified measure is as follows:

$$COMPL = \sum_{j=1}^{y}\left(\sum_{i=1}^{n} x_{ij}\right)^2 \quad (6)$$

where
y number of process components, $y > 0$;
n total number of actions, $n \geq 0$;
x_{ij} 1 if action *i* is placed in process component *j*;
0 if action *i* is not placed in process component *j*.

Table II is a summary of a literature survey on the impact that all the technical features may have on the component management goals [8]. Positive impact (+) means the higher the technical feature value is, the better the goal is achieved. Negative impact (-) means the lower the technical feature is, the better the goal is achieved. No impact (0) means the technical feature has no relation to achieving the goal.

TABLE II
IMPACT OF TECHNICAL FEATURES ON COMPONENT MANAGEMENT GOALS

	COUPL	COHES	NCOMP	CSIZE	COMPL
COST	0	-	-	-	-
ASBL	-	0	0	+	0
CUST	-	+	+	-	0
REUS	-	+	+	-	0
MNTN	-	-	-	-	-

D. Applying BusCod Model

We can adopt the BusCod model in [8]:

$$R' * W * D \quad (7)$$

Each part of the model is as follows.

1) $R' = \left[R_{COST} \quad R_{ASBL} \quad R_{CUST} \quad R_{REUS} \quad R_{MNTN}\right]$

This is a vector representing relative importance of all component management goals for the design where
R_{COST} relative importance of cost effectiveness;
R_{ASBL} relative importance of ease of assembly;
R_{CUST} relative importance of customization;
R_{REUS} relative importance of reusability;
R_{MNTN} relative importance of maintainability;
and $R_{COST} + R_{ASBL} + R_{CUST} + R_{REUS} + R_{MNTN} = 1$.

2)

$$W = \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} & w_{15} \\ w_{21} & w_{22} & w_{23} & w_{24} & w_{25} \\ w_{31} & w_{32} & w_{33} & w_{34} & w_{35} \\ w_{41} & w_{42} & w_{43} & w_{44} & w_{45} \\ w_{51} & w_{52} & w_{53} & w_{54} & w_{55} \end{bmatrix}$$

This is a relation matrix representing the strength of association between managerial goals and technical features where

w_{ij} strength of the association between i^{th} managerial goal and j^{th} technical feature in W. The value is either 0 or 1-10 with the sign (+ or -) as in Table II.

3)

$$D = \begin{bmatrix} D_{COUPL} \\ D_{COHES} \\ D_{NCOMP} \\ D_{CSIZE} \\ D_{COMPL} \end{bmatrix}$$

This is a vector representing measurement of various technical features of the design where

D_{COUPL} intercomponent coupling;
D_{COHES} intracomponent cohesion;
D_{NCOMP} number of components;
D_{CSIZE} component size;
D_{COMPL} complexity.

E. Design Process

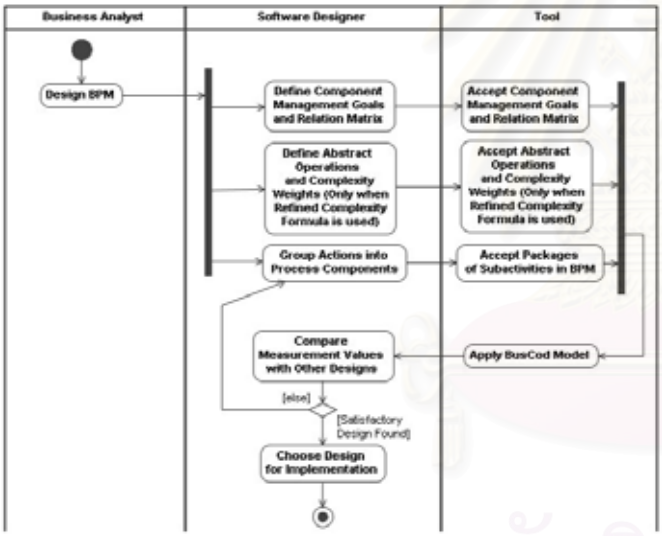


Fig. 3 Process for designing process components

Fig. 3 describes the process taken to design process components. This process will be supported by a software tool (see Section VI). The software designer obtains a BPM from a business analyst and defines relative importance of each managerial goal for the design. If the refined complexity formula is used, the software designer will also define details of abstract operations and complexity weights. Then, the BPM – a UML activity diagram in this case – will be decomposed into groups of actions; each group corresponds to a package. The packages will be processed by the tool to apply the BusCod model. The software designer can repeat this process with different designs and compare their measurement values from the BusCod model. The design with maximum measurement value will best achieve the managerial goals that have been set and can be chosen for implementation.

IV. CASE STUDY

An example business process model used to demonstrate the design of process components is of the flight reservation domain, involving an airline (Fig. 4). We define this model by adapting from the Open Travel Alliance (OTA) specification [12] and the case study of [13]. The flow begins with checking a flight for a particular trip. Other details (i.e., schedule, arrival time, seat availability, and price) are checked subsequently. If seats are available and price is in budget, the flight is booked, the seat numbers are confirmed, and the itinerary is produced. In the case that no seats are available, this failed booking can be recorded for administrative purpose (e.g., to increase the number of flights during some period of the year).

Fig. 4 Business process of flight reservation domain

According to the BusCod model, a software designer may want to design process components for this process flow with an emphasis on reusability. The component management goals may be set as

$$R' = \begin{bmatrix} 0.05 & 0.05 & 0.05 & 0.8 & 0.05 \end{bmatrix}$$

For this example, we use the matrix W below:

$$W = \begin{bmatrix} 0 & -8 & -6 & -8 & -7 \\ -5 & 0 & 0 & +8 & 0 \\ -6 & +5 & +5 & -5 & 0 \\ -8 & +8 & +6 & -7 & 0 \\ -7 & -6 & -5 & -6 & -8 \end{bmatrix}$$

This matrix is derived from a survey on the strength of relationship between the managerial goals and the technical features, conducted on industry experts [8].

Suppose the software designer decomposes the above process flow into three process components (Fig. 5). We may look at them as a flight inquiry component, a flight booking component, and a failed booking component. For later presentation purpose, we also annotate each action with a symbol A1-A9. This design then has its technical features calculated. We discuss the calculation of some values below.

Fig. 5 A design with three process components

This design has 3 process components and 9 actions in total.

For intercomponent coupling and intracomponent cohesion, the value c_{ij} which is the coupling between any two actions in the process model is summarized in Table III.

TABLE III
COUPLING BETWEEN EACH ACTION

	A1	A2	A3	A4	A5	A6	A7	A8	A9
A1	0								
A2	1	0							
A3	1	0	0						
A4	0	1	1	0					
A5	0	0	0	1	0				
A6	0	0	0	0	1	0			
A7	0	0	0	0	0	1	0		
A8	0	0	0	0	0	0	1	0	
A9	0	0	0	1	0	0	0	0	0

Suppose the refined formula for complexity is used, the software designer specifies the operation detail for each action (Table IV). As mentioned in Section III.C, we can see each action as a single abstract operation. The design of the operations here is based on the technique to design service interfaces in [13] which considers elementary business functions and applies data normalization to interface parameters. This leads to minimization of coupling and maximization of cohesion of the operations.

In this example, the software designer also assigns a value 0.5 for the relative importance of operation complexity w_{mcx} and a value 0.5 for the relative importance of parameter complexity w_{pcx}. For each parameter in each operation, a value 1 is assigned for the relative complexity p_{ikl} as the parameters are equally complex (i.e., all are simple parameters).

The values of all technical features are:

$$D = \begin{bmatrix} 2 \\ 7 \\ 3 \\ 3.416 \\ 318.5 \end{bmatrix}$$

The calculation of the BusCod model $R' * W * D$ for this three-component design gives the value -219.33. If we repeat the design process to calculate the BusCod value for a one-component design (i.e., the whole process flow is seen as one component), the BusCod value is -544.05. Therefore, the three-component design better achieves the component management goals that have been set. With an emphasis on the reusability goal, we may reason that the one-component design is less appropriate for reuse because it carries too much functionality.

TABLE IV
ACTIONS AS CLASSES WITH ABSTRACT OPERATIONS AND THEIR PARAMETERS

Abstract Operation of Action (No. of Parameters)	Input Parameters	Output Parameters
CheckFlight (4)	OriginalLocation DestinationLocation DepartureDate	FlightNumber
CheckSchedule (5)	FlightNumber	DepartureAirport DepartureTime ArrivalAirport ArrivalTime
CheckArrival (3)	FlightNumber DepartureDate	ArrivalDate
CheckAvailability (4)	FlightNumber DepartureDate CabinType	NoOfSeats
CheckPrice (6)	FlightNumber DepartureDate CabinType Budget	FareBasisCode BaseFare
BookFlight (5)	FlightNumber DepartureDate TravelerName CabinType	BookingReferenceID
SeatingRequest (3)	BookingReferenceID SeatPreference	SeatNumber
GetItinerary (12)	BookingReferenceID	BookingReferenceID FlightNumber DapartureAirport DepartureDate DepartureTime ArrivalAirport ArrivalDate ArrivalTime CabinType BookingStatus JourneyDuration
RecordUnavailability (3)	FlightNumber DepartureDate CabinType	

For this example, if the simplified formula is used for complexity, the calculation of the BusCod model for the three-component design gives the value -6.706 while the one-component design yields -58.05. The result still corresponds to the case where the refined formula is used.

V. Process Component Reuse

The quantitative measurement above can help give the software designer some confidence over the process component design. Process components from one business process model may be applied in the design of other business processes. For example, the flight inquiry component (top component) in Fig. 5 can be reused in the business process of another organization (Fig. 6). The organization views the reused flight inquiry process component as a single abstract action and composes it with organization's own process to additionally set booking information, buy and print flight tickets, and handle other unsuccessful booking incidents. Unlike the process in Fig. 5, ticket budget, seat assignment at the time of ticket purchase, and itinerary are not of concern to this business process.

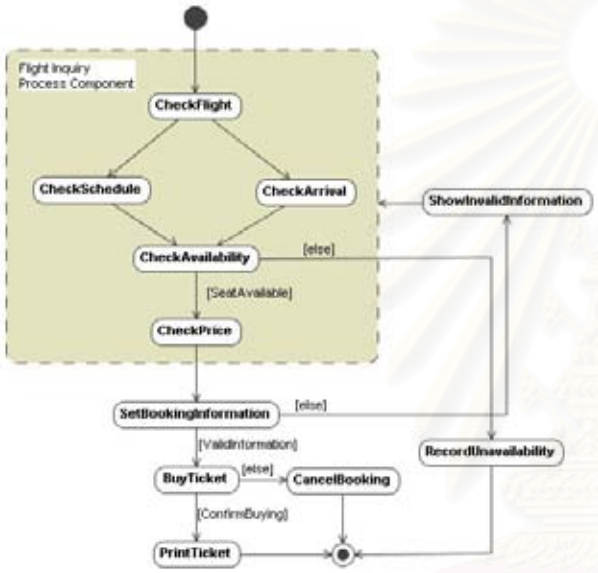


Fig. 6 Reusable process component in another business process

VI. Process Component Measuring Tool

A supporting tool has been developed to support the software designer to measure the quality of process components. According to Fig. 3, it is assumed that the business analyst will first use a modeling tool to design the activity diagram of the business process. The software designer will also use a modeling tool to define groups of subactivities. The result is an XMI-based file with packages of subactivities; the file will be an input to the tool.

In Fig. 7, the software designer can specify the input activity diagram file in the open file tab. Fig. 8 and Fig. 9 show respectively the managerial goals tab and the relation matrix tab where the software designer can fill in appropriate values. The measurement value tab in Fig. 10 calculates the BusCod value of the three-component model in Fig. 5 where a simplified complexity formula is used. The software designer will use this value to compare with those of other designs to determine the quality of the designs.

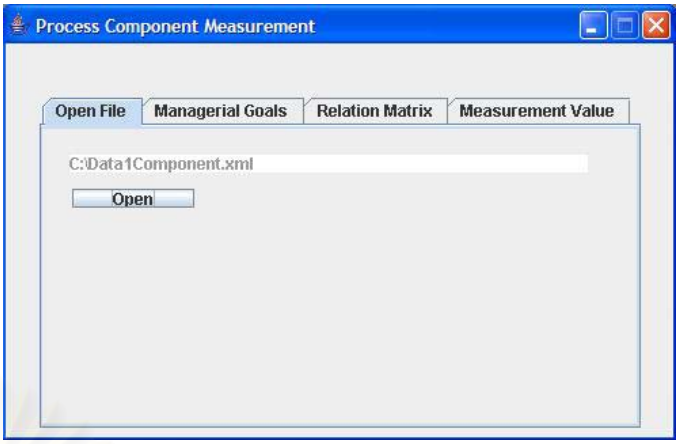


Fig. 7 Open file tab of the tool

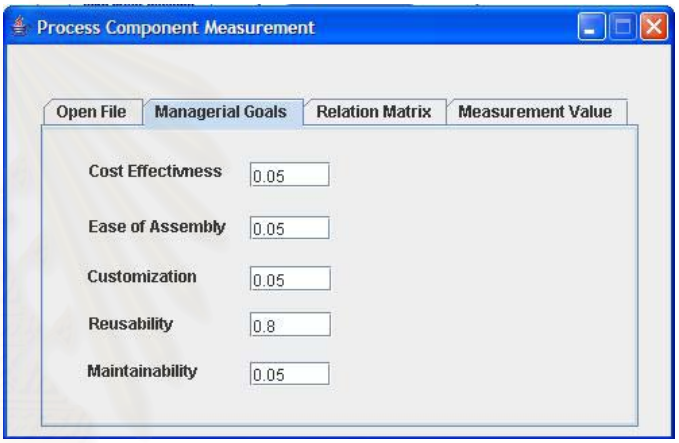


Fig. 8 Managerial goals tab of the tool

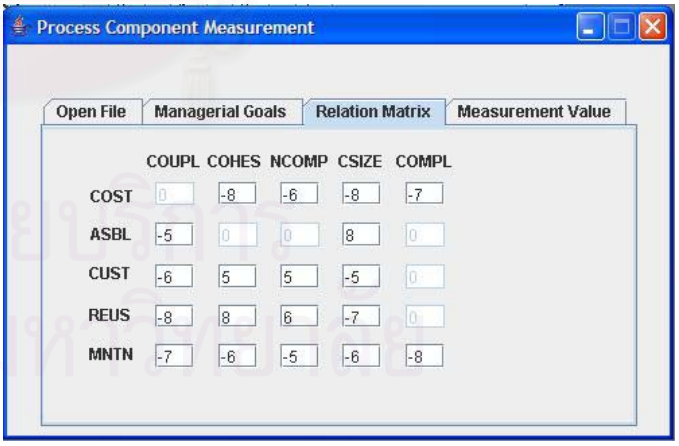


Fig. 9 Relation matrix tab of the tool

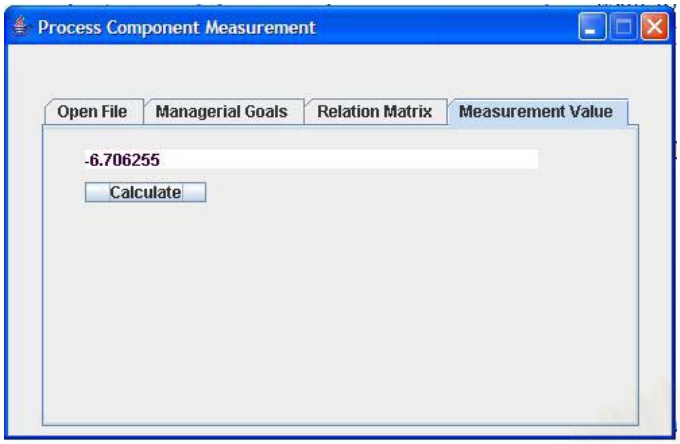


Fig. 10 Measurement value tab of the tool

VII. CONCLUSION

We discuss a possibility to apply a software component fabrication technique to design process components for an application domain which is modeled by a business process model. The paper relies on the BusCod model, the efficiency of which has been evaluated by industry experts as reported in [8]. Once a satisfactory design is found, the software designer can reuse each process component in the design of other business processes, and can have it implemented into a software unit. To implement a process component, we can follow the process-oriented paradigm and map each process component into BPEL [9], or we may follow the traditional paradigm and map each process component into a UML class diagram for component development [14].

This technique requires to a certain extent the skill of the software designer to group process components and to assign weight information for complexity. Other guidelines can help the software designer to decide which actions should be in the same process component (e.g., to reduce coupling and increase cohesion) [10].

The current version of the supporting tool can only give a BusCod value of a particular design. An enhancement is expected such that the tool can give an optimal design for a particular business process model.

ACKNOWLEDGMENT

The authors would like to thank Prof. Padmal Vitharana for guidance on the use of the BusCod model.

REFERENCES

[1] BPMI.org. (2004, May, 3). *Business Process Modeling Notation (BPMN)* Version 1.0. Availalble: http://www.bpmi.org

[2] G. Booch, J. Rumbaugh, and I Jacobson, *The Unified Modeling Language User Guide*, Massachusetts: Addison-Wesley, 1999.

[3] E. Newcomer, Understanding Web Services. Indianapolis: Addison-Wesley, 2002.

[4] IBM. (2003, May, 5) *BPEL4WS Version 1.1 specification* Available: http://www-128.ibm.com/developerworks/library/specification/ws-bpel/

[5] C. Szyperski, *Component Software: Beyond Object-Oriented Programming*, 2nd ed. New York: Addison-Wesley, 2002.

[6] Y. Mou, J. Cao, and S. Zhang, "A process component model for enterprise business knowledge reuse," in *Proc. IEEE International Conference on Services Computing (SCC04),* 2004.

[7] O. H. Barros. (2004, September). *Business Information System Design Based on Process Pattern and Frameworks*. Industrial Engineering Department, University of Chile. Available: http://www.BPtrends.com

[8] P. Vitharana, H. Jain, and F. M. Zahedi, "Strategy-based design of reusable business components," *IEEE Transactions on Systems, Man, and Cybernetics–Part C: Applications and Reviews*, vol. 34, No. 4, pp. 460-474, November 2004.

[9] J. Koehler, R. Hauser, S. Kapoor, F. Y. Wu, and S. Kumaran, "A model-driven transformation method," in *Proc. 7th IEEE International Enterprise Distributed Object Computing Conference*, 16-19 September 2003, pp. 186-197.

[10] N. Tagoug, "Object-oriented system decomposition quality", in *Proc. 7th International Symposium on High Assurance Systems Engineering (HASE02)*, 2002.

[11] B. Husslage, E. van Dam, D. den Hertog, P. Stehouwer, and E. Stinstra, *Coordination of coupled black box simulations in the construction of metamodels*, Discussion Paper 2, Center for Economic Research, Tilburg University, 2003.

[12] Open Travel Alliance (OTA). (2005, December, 05). *OTA Specification 2005B*. Available: http://www.opentravel.org

[13] G. Feuerlicht and S. Meesathit, "Design framework for interoperable service interfaces," in *Proc. 2nd International Conference on Service Oriented Computing (ICSOC'04)*, New York, 2004, pp. 299-307.

[14] W. Rungworawut and T. Senivongse, "A guideline to mapping business process to UML class diagrams," *WSEAS Transactions on Computers,* vol. 4, November 2005.

ประวัติผู้เขียนวิทยานิพนธ์

นายเอกองค์ อธิปธรรมวารี เกิดเมื่อวันที่ 14 มีนาคม พ.ศ. 2525 ที่จังหวัดกรุงเทพฯ สำเร็จการศึกษาระดับปริญญาบัณฑิต หลักสูตรวิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์ จาก จุฬาลงกรณ์มหาวิทยาลัย เมื่อ พ.ศ. 2547 และ สำเร็จการศึกษาระดับปริญญาบัณฑิต สาขาวิชาบริหารการจัดการทั่วไป จากมหาวิทยาลัยสุโขทัยธรรมาธิราช เมื่อ ปี พ.ศ. 2547 และได้เข้าศึกษาต่อในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมซอฟต์แวร์ ณ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปี พ.ศ. 2547