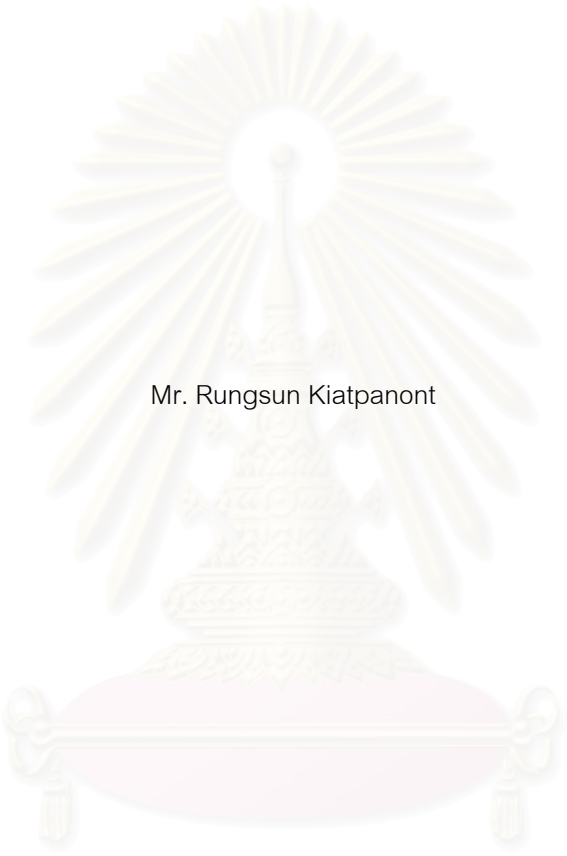
วิธีการออกแบบและพัฒนาระบบงานสำหรับคอมพิวเตอร์มือถือโดยใช้สถาปัตยกรรมอิงแบบจำลอง

นายรังสรรค์ เกียรติ์ภานนท์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2546
ISBN 974-17-4178-2

A SYSTEM DESIGN AND DEVELOPMENT METHODOLOGY FOR HANDHELD COMPUTERS USING MODEL DRIVEN ARCHITECHTURE

Mr. Rungsun Kiatpanont



A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Computer Science
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2003

ISBN 974-17-4178-2

หัวข้อวิทยานิพนธ์	วิธีการออกแบบและพัฒนาระบบงานสำหรับคอมพิวเตอร์มือถือโดยใช้สถาปัตยกรรมอิงแบบจำลอง
โดย	นายรังสรรค์ เกียรติ์ภานนท์
สาขาวิชา	วิทยาศาสตร์คอมพิวเตอร์
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
(อาจารย์ ดร.ณัฐวุฒิ หนูไพโรจน์)

…… กรรมการ
(อาจารย์ จารุมาตร ปิ่นทอง)

รังสรรค์ เกียรติ์ภานนท์ : วิธีการออกแบบและพัฒนาระบบงานสำหรับคอมพิวเตอร์มือถือโดยใช้สถาปัตยกรรมอิงแบบจำลอง (A SYSTEM DESIGN AND DEVELOPMENT METHODOLOGY FOR HANDHELD COMPUTERS USING MODEL DRIVEN ARCHITECHTURE) อ. ที่ปรึกษา : ผู้ช่วยศาสตราจารย์ ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา,117 หน้า. ISBN 974-17-4178-2.

ระบบสารสนเทศในปัจจุบันมีแนวโน้มที่จะมีลักษณะเป็นระบบกระจายมากขึ้น การพัฒนาระบบจึงมีความซับซ้อนมากขึ้น และยังต้องสามารถรองรับการทำงานร่วมกันของส่วนต่างๆ ของระบบที่อยู่บนแพลตฟอร์ม หรือใช้เทคโนโลยีที่แตกต่างกัน และมีการเปลี่ยนแปลงอย่างรวดเร็ว

เพื่อสนับสนุนความต้องการดังกล่าว โอเอ็มจีจึงได้เสนอเอ็มดีเอซึ่งเป็นแนวคิดการพัฒนาระบบสารสนเทศ ซึ่งใช้แบบจำลองในระดับต่างๆ ของระบบเป็นตัวขับเคลื่อนให้เกิดการสร้างโปรแกรมของระบบโดยอัตโนมัติ แต่เนื่องจากเอ็มดีเอยังเป็นแนวคิดที่ค่อนข้างใหม่ จึงทำให้การพัฒนาระบบสารสนเทศตามแนวคิดนี้ยังมีภาพที่ไม่ชัดเจนนัก และยังไม่สามารถนำแนวคิดนี้ไปใช้ได้อย่างเต็มที่ในปัจจุบัน

วิทยานิพนธ์นี้มีจุดประสงค์ที่จะศึกษาหลักการของแนวคิดเอ็มดีเอ โดยทดลองพัฒนาระบบสารสนเทศขนาดเล็กที่มีตรรกะทางธุรกิจไม่ซับซ้อน เพื่อพิจารณาความเป็นไปได้ และความซับซ้อนในการใช้เอ็มดีเอมาพัฒนาระบบจริงอย่างอัตโนมัติ โดยระบบสารสนเทศตัวอย่างที่ใช้เป็นระบบลอตเตอรีออนไลน์สำหรับเครื่องคอมพิวเตอร์มือถือปาล์ม ซึ่งใช้เจทูเอ็มอีในการพัฒนา และมีการติดต่อกับเซิร์ฟเวอร์บนพีซีซึ่งใช้เจทูอีอี/เซิร์ฟเล็ตในการพัฒนา นอกจากนี้ ได้ทำการสำรวจมาตรฐานและเครื่องมือในการพัฒนาระบบซึ่งมีอยู่ในท้องตลาดว่าได้รองรับเอ็มดีเอแล้วมากน้อยเพียงใด จากการทดลองพัฒนาระบบตัวอย่างและการสำรวจเครื่องมือ พบว่า การพัฒนาระบบโดยอัตโนมัติด้วยแนวคิดเอ็มดีเอนั้นจะมีความซับซ้อน เนื่องจากแบบจำลองที่สร้างให้กับระบบงาน จะต้องมีรายละเอียดและสะท้อนพฤติกรรมต่างๆ ของระบบเพียงพอ และแบบจำลองต่างๆ จะต้องสอดคล้องและสามารถผนวกรวมเป็นภาพรวมของระบบงานที่ชัดเจนให้ได้ อีกทั้งเอ็มดีเอจะกลายเป็นแนวคิดหลักสำหรับการพัฒนาระบบสารสนเทศในอนาคตได้ จำเป็นต้องมีเครื่องมือช่วยในการพัฒนาระบบ ซึ่งมีความสามารถในการรองรับความซับซ้อนจากแบบจำลองในหลายแง่มุม รวมทั้งจะต้องมีมาตรฐานการแปลงแบบจำลองในระดับต่างๆ และมาตรฐานโค้ดที่ชัดเจนสำหรับหลากหลายแพลตฟอร์ม

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

สาขาวิชา.....วิทยาศาสตร์คอมพิวเตอร์ .... ลายมือชื่ออาจารย์ที่ปรึกษา.....

ปีการศึกษา 2546

4470484021 : MAJOR COMPUTER SCIENCE

KEY WORD: MDA / MODELING TECHNOLOGY / UML / PALM / CODE GENERATION

RUNGSUN KIATPANONT : A SYSTEM DESIGN AND DEVELOPMENT METHODOLOGY FOR HANDHELD COMPUTERS USING MODEL DRIVEN ARCHITECHTURE. THESIS ADVISOR : ASSIST.PROF TWITTIE SENIVONGSE, Ph.D, 117 pp. ISBN 974-17-4178-2.

Information systems today are likely to have distributed systems characteristics. As a result, application development is getting complicated and requires parts of the system components on different platforms and with different technologies to interoperate and be able to handle changes. To answer this requirement, OMG has proposed MDA which is an application development concept by which the construction of the application is driven by several levels of system models. Since it is relatively new, application development by MDA does not impose a clear picture and the concept is not fully exploited in application development at present.

The objective of this thesis is to study MDA concept through the development of a pilot application of small size and uncomplicated business logic, in order to see the possibility and complexity when using MDA to automatically implement a real system. The pilot application is an online lottery system for palm PDA using J2ME and connecting to a J2EE/Servlet server on a PC. This thesis also surveys some application development standards and tools in the market to check their maturity to support MDA. The pilot development and the survey show that automatic application development by MDA concept is complicated since several of system models are required and each of them needs to represent details and behavior of the application very clearly. Different models needs to be consistent and able to integrate well to represent a clear picture of the overall system. For MDA to be the mainstream concept for information systems development in the future, development tools that can maintain the complexity of the models and standard mappings for system models and code for various platforms are essential.

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

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

Academic year 2003

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จได้ด้วยความกรุณา และความช่วยเหลืออย่างดียิ่งจาก ผศ.ดร. ทวิตีย์ เสนีวงศ์ ณ อยุธยา อาจารย์ที่ปรึกษาวิทยานิพนธ์ ซึ่งท่านได้สละเวลาในการให้คำปรึกษา คำแนะนำและข้อคิดเห็นต่างๆ ตลอดจนช่วยตรวจแก้ไขวิทยานิพนธ์ด้วยความทุ่มเท เสียสละ และเอาใจใส่อย่างดียิ่ง

ขอขอบคุณ อ.ดร.ยรรยง เต็งอำนวย ซึ่งช่วยแนะนำแนวคิดต่างๆ ที่เป็นประโยชน์อย่างมาก จนกลายเป็นแนวคิดหลักที่ประยุกต์ใช้ในวิทยานิพนธ์นี้ รวมถึง อ.ดร. ณัฐวุฒิ หนูไพโรจน์ และ อ. จารุมาตร ปิ่นทอง ซึ่งเป็นกรรมการสอบวิทยานิพนธ์นี้

ขอขอบคุณ นายศรัณย์ ชัยวรวิทยกุล และนายชยันต์ เทพบุตร ที่ช่วยอยู่เป็นเพื่อนทำงานด้วยกันจนดึกเสมอๆ

ขอขอบคุณ นางสาวรัศมีทิพย์ วิตา นางสาวณิศรา ศิริพรกุลทรัพย์ นายกัน อุตะเดช และนายกิตติพิชญ์ คุปตะวาณิช ซึ่งเป็นเพื่อนร่วมกลุ่มการวิจัยเกี่ยวกับระบบสารสนเทศของตลาดหลักทรัพย์ และคอยช่วยข้าพเจ้าในทุกๆ เรื่อง ตลอดการทำวิทยานิพนธ์นี้

ขอขอบคุณ นายชาติชาย ดวงสอาด และนางสาวนงเยาว์ จินดาสวัสดิ์ ซึ่งคอยให้ความช่วยเหลือตอบข้อซักถามต่างอยู่เสมอๆ

ขอขอบคุณ เพื่อนๆ พี่ๆ น้องๆ ในห้องปฏิบัติการวิศวกรรมสารสนเทศ และห้องปฏิบัติการวิศวกรรมซอฟต์แวร์ ทุกท่านที่ให้ความช่วยเหลือข้าพเจ้าอย่างดีมาตลอด

ขอขอบคุณ นางสาวกัลย์สุดา ปลั่งศิริ ถึงแม้เธออาจจะไม่ทราบ แต่เธอช่วยให้ชีวิตการเรียนปริญญาโทของผมมีความหมายมากขึ้น

สุดท้ายนี้ ขอกราบขอบพระคุณ คุณพ่อ คุณแม่ รวมถึงน้องๆ ของข้าพเจ้าที่ช่วยเป็นกำลังใจ และให้การสนับสนุนข้าพเจ้า จนสำเร็จการศึกษา

สารบัญ

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญตาราง

ตาราง หน้า

สารบัญรูปภาพ

ภาพประกอบ หน้า

สารบัญรูปภาพ(ต่อ)

ภาพประกอบ หน้า

สารบัญรูปภาพ(ต่อ)

ภาพประกอบ หน้า

สารบัญรูปภาพ(ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

แอพพลิเคชันในปัจจุบันมีแนวโน้มเป็นระบบกระจายเชิงวัตถุมากขึ้น สาเหตุเนื่องมาจากความนิยมของการออกแบบระบบโดยแนวคิดเชิงวัตถุที่มีเพิ่มมากขึ้น และงานที่สามารถทำโดยเครื่องคอมพิวเตอร์เพียงเครื่องเดียวได้มีน้อยลง ปัญหาที่ตามมาของการทำงานแบบกระจายคือปัญหาเรื่องความสามารถในการทำงานร่วมกัน (Interoperability) เนื่องจากมีความเป็นไปได้อย่างมากว่าเครื่องคอมพิวเตอร์ที่ทำงานร่วมกันนั้น อาจมีแพลตฟอร์มที่ต่างกันเช่น มีระบบปฏิบัติการที่ต่างกัน ถูกเขียนขึ้นโดยคนละภาษาโปรแกรมกัน อยู่บนเครือข่ายคนละประเภทกัน เป็นมิดเดิลแวร์คนละชนิดกัน หรือใช้โพรโทคอลที่ไม่เหมือนกัน นอกจากนี้ยังมีปัญหาเกี่ยวกับการบูรณาการ (Integration) เช่นเมื่อต้องการเชื่อมต่อการทำงานของสองระบบให้ทำงานร่วมกัน โดยปัญหาจะมากขึ้นในกรณีที่ระบบทั้งสอง ไม่ได้เป็นแพลตฟอร์มเดียวกัน

โอเอ็มจี (OMG: Object Management Group) เป็นองค์กรที่ก่อตั้งโดยความร่วมมือขององค์กรชั้นนำทางด้านเทคโนโลยีสารสนเทศหลายบริษัท มีจุดมุ่งหมายเพื่อกำหนดมาตรฐานที่เกี่ยวกับความสามารถในการทำงานร่วมกันของระบบกระจายเชิงวัตถุ โดยมีผลงานที่เป็นที่รู้จักเช่นคอร์บา (CORBA) ซึ่งเป็นมาตรฐานของมิดเดิลแวร์ที่มีลักษณะพิเศษคือ เป็นมาตรฐานที่ไม่อิงกับผู้ผลิต หรือภาษาโปรแกรมใดๆ (Vendor- , Language-Independent Interoperability Standard) โดยคอร์บาสามารถจัดการกับปัญหาของการทำงานร่วมกันระหว่างวัตถุ ที่เขียนขึ้นโดยคนละภาษากัน หรืออยู่ในสภาวะแวดล้อมที่ต่างกัน เช่น ระบบปฏิบัติการที่ต่างกัน ซึ่งปัจจุบันก็มีมิดเดิลแวร์ที่ทำหน้าที่เช่นนี้หลายตัวเช่น อีเจบี (EJB) หรือดอตเนต (.NET) เป็นต้น

โอเอ็มจี ได้พัฒนาแนวคิดเรื่องการทำงานร่วมกันต่อไป กล่าวคือระบบควรมีความสามารถในการทำงานร่วมกันได้อย่างแท้จริง โดยไม่มีข้อจำกัดแม้จะใช้แพลตฟอร์มของมิดเดิลแวร์ที่ต่างกัน ซึ่งแนวคิดนี้ทำได้ โดยแยกการออกแบบระบบออกเป็นสองส่วนเรียกว่าพีไอเอ็ม (PIM: Platform Independent Model) และพีเอสเอ็ม (PSM: Platform Specific Model) ซึ่งพีไอเอ็มนั้นเป็นแบบจำลองของธุรกิจแต่เพียงอย่างเดียวโดยมิได้คำนึงว่าจะนำไปพัฒนาอย่างไร บนแพลตฟอร์มใด เมื่อได้พีไอเอ็มที่เสถียรแล้ว ก็จะนำพีไอเอ็มไปแปลงเป็นพีเอสเอ็ม ซึ่งเป็นแบบจำลองของระบบบนมิดเดิลแวร์แพลตฟอร์มที่ใช้พัฒนา และจากพีเอสเอ็มนี้จะสามารถนำไปอิมพลีเมนต์เป็นโค้ดของระบบงานได้อย่างอัตโนมัติ แนวคิดการพัฒนาระบบงานจากแบบจำลองระดับต่างๆ ของระบบ

เพื่อนำไปสู่การสร้างโปรแกรมต้นฉบับนี้เรียกว่าสถาปัตยกรรมอิงแบบจำลอง หรือเอ็มดีเอ (MDA: Model Driven Architecture) [1]

แนวคิดเรื่องเอ็มดีเอนี้สามารถแก้ปัญหาเรื่องความสามารถในการทำงานร่วมกัน และการบูรณาการของระบบที่กล่าวไปในตอนต้นได้เนื่องจาก การออกแบบระบบโดยแบบจำลองพีไอเอ็มจะไม่พิจารณารายละเอียดของแพลตฟอร์ม กล่าวคือไม่ได้สนใจว่าพีไอเอ็มที่ได้จะถูกนำไปพัฒนาบนแพลตฟอร์มใด ส่วนการแปลงพีไอเอ็มเป็นพีเอสเอ็มนั้น ทำได้โดยการกำหนดมาตรฐานการแปลง (Standard Mapping) ซึ่งจะแปลงแบบจำลองพีไอเอ็มไปเป็นแบบจำลองพีเอสเอ็มสำหรับแต่ละแพลตฟอร์มที่ใช้ได้ นอกจากนี้ยังเป็นการเพิ่มความยืดหยุ่น (Flexibility) ในการพัฒนาโปรแกรม เพราะสามารถนำแบบจำลองพีไอเอ็มเดิมมาสร้างเป็นแบบจำลองพีเอสเอ็มของแพลตฟอร์มใหม่ได้เมื่อเกิดการเปลี่ยนแปลงของเทคโนโลยี ทำให้ค่าใช้จ่ายในการพัฒนาจะน้อยลง เนื่องจากแบบจำลองพีไอเอ็มซึ่งแสดงถึงความต้องการเชิงธุรกิจ (Business Requirement) ที่แท้จริงนั้น สามารถนำกลับมาใช้ใหม่ได้เสมอ ตราบเท่าที่ความต้องการเชิงธุรกิจยังไม่เปลี่ยนไป

อย่างไรก็ตาม การนำเอาแนวคิดเอ็มดีเอมาใช้ในปัจจุบันนั้น ยังมีความไม่พร้อมในหลายด้าน เช่น ความไม่เฉพาะเจาะจงของการสื่อความหมายด้วยภาษายูเอ็มแอล [2] ซึ่งเป็นภาษาที่ใช้อธิบายแบบจำลองของระบบ ทำให้ไม่สามารถนำไปสร้างพีเอสเอ็ม หรือโปรแกรมต้นฉบับได้อย่างสมบูรณ์ อีกทั้งในปัจจุบันมาตรฐานการแปลงพีไอเอ็มเป็นพีเอสเอ็มสำหรับแพลตฟอร์มต่างๆ รวมทั้งการแปลงพีเอสเอ็มไปเป็นโปรแกรมต้นฉบับ ยังอยู่ในขั้นตอนการกำหนดเป็นมาตรฐานอยู่ [1, 3] ผลที่ตามมาคือยังไม่มีเครื่องมือตัวใดที่รองรับการออกแบบและพัฒนาระบบด้วยแนวคิดเอ็มดีเอได้อย่างสมบูรณ์ ซึ่งนับเป็นอุปสรรคที่สำคัญในการนำแนวคิดเอ็มดีเอไปใช้จริง

จากปัญหาดังกล่าว งานวิจัยนี้จึงมีแนวคิดที่จะนำเอ็มดีเอมาใช้กับการพัฒนาระบบงานสำหรับเครื่องคอมพิวเตอร์มือถือ (Handheld Computer) ซึ่งนับวันจะเป็นที่แพร่หลายมากขึ้นในแอพพลิเคชันสำหรับระบบกระจาย โดยงานวิจัยจะทำการศึกษาและพัฒนาระบบงานบนคอมพิวเตอร์มือถือตามกระบวนการของเอ็มดีเอ โดยมีการกำหนดกฎการแปลงพีไอเอ็มเป็นพีเอสเอ็มและการแปลงพีเอสเอ็มเป็นโปรแกรมต้นฉบับ นอกจากนี้งานวิจัยจะศึกษาถึงปัญหาที่อาจเกิดขึ้นเมื่อใช้เอ็มดีเอพัฒนาระบบ รวมทั้งสำรวจความพร้อมของทรัพยากร หรือเครื่องมือต่างๆ ที่จำเป็นต่อการพัฒนาระบบด้วยเอ็มดีเอ

1.2 วัตถุประสงค์ของการวิจัย

1. เพื่อกำหนดวิธีการพัฒนาแอพพลิเคชันสำหรับเครื่องปาล์มโดยเอ็มดีเอ
2. เพื่อศึกษาปัญหาและความเป็นไปได้ในการนำแนวคิดเอ็มดีเอมาใช้ในการพัฒนาระบบ

1.3 ขอบเขตของการวิจัย

1. เครื่องคอมพิวเตอร์มือถือที่เป็นแพลตฟอร์มของการพัฒนาระบบด้วยแนวคิดเอ็มดีเอคือเครื่องปาล์ม
2. แบบจำลองพีไอเอ็ม และพีเอสเอ็ม เป็นไปตามมาตรฐานยูเอ็มแอลรุ่น 1.4 เป็นอย่างน้อย
3. การแปลงจากพีไอเอ็มเป็นพีเอสเอ็ม ไม่จำเป็นต้องทำโดยเครื่องมืออัตโนมัติ (สามารถทำด้วยมือได้) แต่สามารถทำโดยเครื่องมือได้หากมีเครื่องมือพร้อมใช้งานอยู่แล้ว
4. การแปลงพีเอสเอ็มเป็นโค้ด ไม่จำเป็นต้องทำโดยเครื่องมืออัตโนมัติ (สามารถทำด้วยมือได้) แต่สามารถทำโดยเครื่องมือได้หากมีเครื่องมือพร้อมใช้งานอยู่แล้ว
5. โปรแกรมที่ได้สามารถทำงานบนเครื่องปาล์มจริง หรือบนโปรแกรมเครื่องปาล์มจำลอง อย่างใดอย่างหนึ่งได้

1.4 ขั้นตอนการดำเนินงาน

1. ศึกษาความรู้เบื้องต้นที่เกี่ยวกับเอ็มดีเอ
2. ศึกษาทฤษฎีเกี่ยวกับภาษายูเอ็มแอล และการเขียนโปรแกรมบนเครื่องปาล์ม
3. สร้างแบบจำลองพีไอเอ็มของการส่งข้อมูลทางเดียว จากเครื่องปาล์มไปยังเครื่องพีซี
4. เขียนโปรแกรมเพื่อการส่งข้อมูลทางเดียวจากเครื่องปาล์มไปยังเครื่องพีซี
5. สร้างแบบจำลองพีเอสเอ็มโดยการทำวิศวกรรมย้อนกลับจากโค้ดที่ได้เขียนขึ้น
6. กำหนดกฎการแปลงจากพีไอเอ็มเป็นพีเอสเอ็ม
7. กำหนดกฎการแปลงจากพีเอสเอ็มเป็นโค้ด
8. แก้ไขรูปแบบของโค้ดให้เป็นไปตามกฎการแปลงที่กำหนดไว้
9. ตรวจสอบความสอดคล้องของการพัฒนาในแต่ละขั้นตอนจากการใช้กฎการแปลงที่กำหนดขึ้น
10. สรุปผลการวิจัย และจัดทำรายงานวิทยานิพนธ์

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้ตัวอย่างในการพัฒนาระบบทั้งระบบด้วยแนวคิดเอ็มดีเอ
2. ได้ศึกษาปัญหาที่อาจพบในการพัฒนาระบบด้วยเอ็มดีเอ เพื่อนำไปสู่แนวทางแก้ไขปัญหาหรืองานวิจัยอื่นๆ ต่อไป
3. ได้เข้าใจถึงสภาพความพร้อมของการพัฒนาระบบในปัจจุบันว่ารองรับแนวคิดเอ็มดีเอมากน้อยเพียงใด

1.6 โครงสร้างวิทยานิพนธ์

วิทยานิพนธ์ฉบับนี้ประกอบด้วยการอธิบายถึงทฤษฎีและงานวิจัยที่เกี่ยวข้องในบทที่ 2 ยกตัวอย่างเช่น ภาษายูเอ็มแอลและยูเอ็มแอลโปรไฟล์ เทคโนโลยีมิดเดิลแวร์ หลักการของเอ็มดีเอ การจัดการแบบจำลองด้วยเอ็มโอเอฟ การพัฒนาโปรแกรมบนเครื่องคอมพิวเตอร์มือถือ จากนั้นในบทที่ 3 จะกล่าวถึงวิธีการพัฒนาระบบด้วยแนวคิดเอ็มดีเอโดยจะอธิบายตามลำดับขั้นของการพัฒนา ได้แก่ การสร้างแบบจำลองพีไอเอ็ม การสร้างแบบจำลองพีเอสเอ็ม และการสร้างโค้ดตามลำดับ ส่วนในบทที่ 4 นั้นเป็นการแสดงการพัฒนาระบบตัวอย่างตามวิธีการที่ได้อธิบายไว้ในบทที่ 3 และแสดงตัวอย่างผลลัพธ์ของการออกแบบในแต่ละขั้น โดยจะมีการวิเคราะห์ผลการทดลองพัฒนาระบบและนำเสนอข้อคิดเห็นที่ได้ทั้งจากการพัฒนาระบบ และการสำรวจเครื่องมือที่มีอยู่ในบทที่ 5 ตามด้วยการสรุปผลการวิจัยในบทที่ 6

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 แนวคิดและทฤษฎี

2.1.1 ภาษายูเอ็มแอล (Unified Modeling Language) [4]

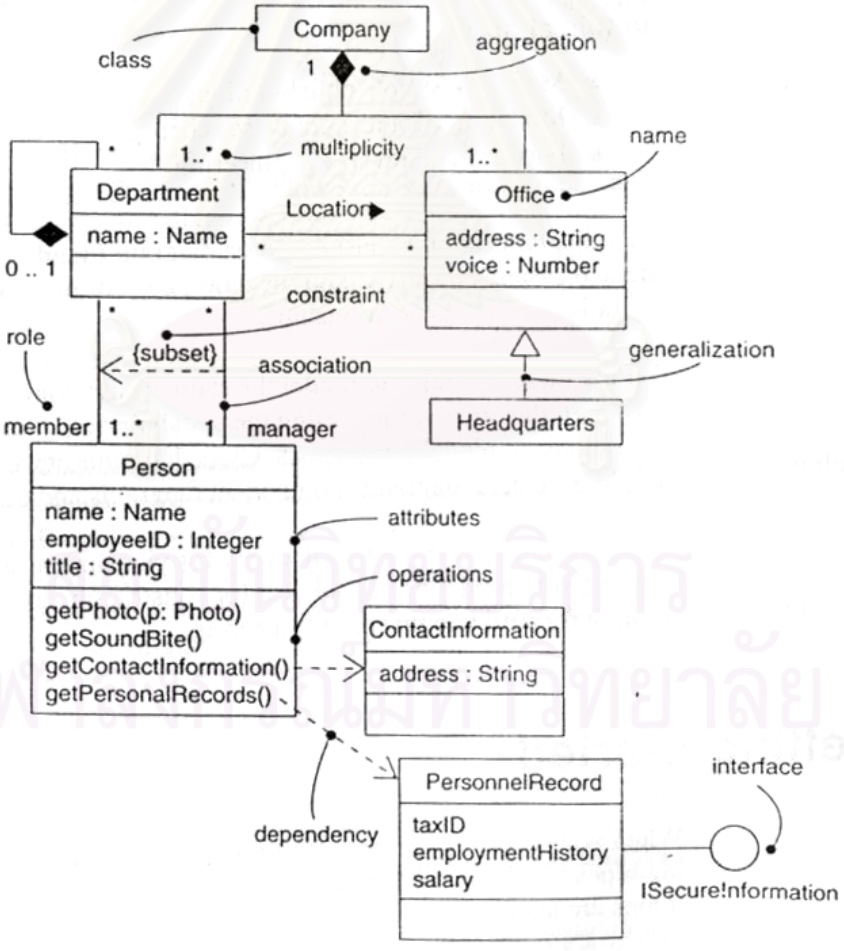
ยูเอ็มแอลเป็นภาษาโมเดลมาตรฐาน (Standard Modeling Language) ซึ่งใช้แสดงแบบจำลองของระบบ (System Model) ภาษายูเอ็มแอลเกิดจากการรวมแนวทางการออกแบบเชิงวัตถุ (Object-oriented design) ซึ่งมีหลากหลายแนวทางให้เป็นหนึ่งเดียว และผลักดันให้เป็นมาตรฐาน ดังนั้นภาษายูเอ็มแอลจึงมีการแสดงคุณลักษณะของระบบตามแนวคิดเชิงวัตถุเช่นมีการห่อหุ้ม (Encapsulation) ทั้งสถานะ (State) และพฤติกรรม (Behavior) ไว้ภายในวัตถุ และถึงแม้ว่ายูเอ็มแอลจะไม่ใช่ภาษาโปรแกรม แต่ในการเชื่อมโยงแบบจำลองที่สร้างขึ้นด้วยภาษายูเอ็มแอลกับภาษาโปรแกรมต่างๆ ก็ทำได้ง่ายเนื่องจากมีแนวคิดเชิงวัตถุเหมือนกัน ดังนั้นแบบจำลองของระบบที่สร้างด้วยภาษายูเอ็มแอล จึงมีคุณค่ามากกว่าการเป็นเพียงเอกสารการออกแบบ (Design Document) เนื่องจากแบบจำลองดังกล่าวสามารถช่วยในการอิมพลีเมนต์เป็นโปรแกรมได้จริง โดยความสามารถในการแสดงออก (Expressiveness) ของยูเอ็มแอลเกิดจากการมองระบบจากหลายมุมมอง ผ่านการแสดงออกจากหลายแผนภาพ (Diagram) ต่างๆ ดังแสดงในตารางที่ 2.1

ตารางที่ 2.1 ชื่อมุมมอง และแผนภาพต่างๆ ในการวิเคราะห์ระบบด้วยภาษายูเอ็มแอล

Main Area	View	Diagram	Main Concept
Structural	Static View	Class Diagram	Class, Association, Generalization, Dependency, Realization, Interface
	Use Case View	Use Case Diagram	Use Case, Actor, Association, Extend, Include, Use Case Generalization
	Implementation View	Component Diagram	Component, Interface, Dependency, Realization
	Deployment View	Deployment Diagram	Node, Component, Dependency, Location
Dynamic	State Machine View	State Chart Diagram	State, Event, Transition, Action
	Activity View	Activity Diagram	State, Activity, Completion Transition, Fork, Join
	Interaction View	Sequence Diagram	Interaction, Object, Message, Activation
		Collaboration Diagram	Collaboration, Interaction, Collaboration Role, Message
Model Management	Model Management View	Class Diagram	Package, Subsystem, Model

ในทางปฏิบัตินั้น การสร้างแบบจำลองของระบบอาจไม่จำเป็นต้องใช้ทุกแผนภาพพร้อมกันก็ได้ โดยผู้ออกแบบเป็นผู้มีหน้าที่เลือกแผนภาพที่คิดว่าเหมาะสมในการแสดงลักษณะของระบบ ซึ่งโดยทั่วไปนั้นควรมีอย่างน้อย 2 แผนภาพเพื่อใช้ในการแสดงมุมมองเชิงโครงสร้าง และเชิงพฤติกรรมของระบบ สำหรับมุมมองเชิงโครงสร้างนั้นแผนภาพคลาสเป็นแผนภาพที่จำเป็นที่นักออกแบบโดยทั่วไปนิยมใช้ในการแสดงมุมมองเชิงโครงสร้างของระบบเสมอ แต่สำหรับการแสดงมุมมองเชิงพฤติกรรมนั้น นักออกแบบแต่ละคนมักมีแนวทางการเลือกแผนภาพที่ใช้แสดงพฤติกรรมของระบบแตกต่างกันออกไป นอกจากนั้นอาจใช้หลายแผนภาพร่วมกันแสดงพฤติกรรมของระบบได้เช่นกัน สำหรับในงานวิจัยชิ้นนี้เลือกใช้แผนภาพ 3 แผนภาพได้แก่ แผนภาพคลาส ในการแสดงลักษณะเชิงโครงสร้างของระบบ ส่วนแผนภาพแอคทิวิตีและแผนภาพซีเควนซ์ใช้แสดงลักษณะเชิงพฤติกรรมของระบบ

2.1.1.1 แผนภาพคลาส



รูปที่ 2.1 ตัวอย่างแผนภาพคลาส [5]

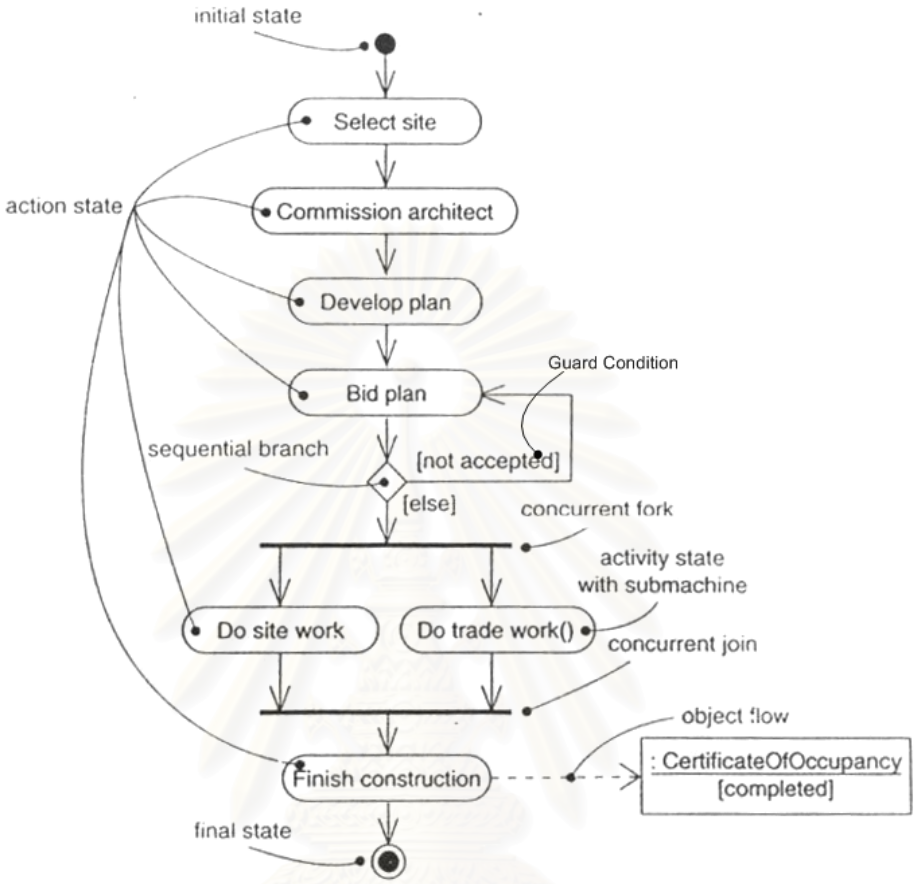
แผนภาพคลาสเป็นแผนภาพที่ใช้แสดงส่วนประกอบเชิงโครงสร้างของระบบโดยโครงสร้างแต่ละส่วนจะแสดงออกในรูปของคลาส โดยแต่ละคลาสก็จะมีรายละเอียดภายในคลาสนั้นๆ นอกจากนั้นแผนภาพคลาสยังแสดงถึงความสัมพันธ์ระหว่างคลาสอีกด้วย และจากรูปที่ 2.1 แสดงตัวอย่างของแผนภาพคลาสพร้อมทั้งส่วนประกอบต่างๆ ของแผนภาพคลาส โดยส่วนประกอบหลักๆ ของแผนภาพคลาสได้แก่

1. คลาส (Class) เป็นส่วนประกอบพื้นฐานที่สำคัญที่สุดของแผนภาพคลาส ใช้ในการแสดง (Represent) ตัวตนของคอนเซปต์ต่างๆ ซึ่งอาจจะเป็นสิ่งที่มีลักษณะทางกายภาพอยู่แล้ว (Physical Things) เช่น เครื่องบิน ตู้เย็น ลูกค้า ใบสั่งสินค้า เป็นต้น หรือเป็นสิ่งที่เป็นคอนเซปต์เชิงนามธรรม (Abstract Things) เช่น สตรีม (Stream) หรือ สแตก (Stack) เป็นต้น โดยในหนึ่งคลาสจะประกอบด้วย ชื่อ (Name) แอททริบิวท์ (Attributes) และ โอเปอร์เรชัน (Operation)
2. แอททริบิวท์ (Attributes) เป็นส่วนหนึ่งของคลาสเพื่อบอกความรู้ของคลาสว่า คลาสดังกล่าวมีความรู้ (Knowledge) อะไรบ้างโดยความรู้แต่ละชิ้นจะแสดงด้วยตัวแปรซึ่งจะมีชนิด (Type) ที่แตกต่างกัน
3. โอเปอร์เรชัน (Operations) เป็นส่วนหนึ่งของคลาสเพื่อบอกความสามารถของคลาสว่า คลาสดังกล่าวสามารถทำอะไรได้บ้าง โดยแต่ละความสามารถจะแสดงด้วยโอเปอร์เรชันซึ่งอาจต้องการข้อมูลนำเข้า (Input Parameter) หรือไม่ก็ได้ รวมทั้งอาจมีการคืนค่าผลลัพธ์ (Return Value) ออกมาหรือไม่ก็ได้เช่นกัน
4. แอสโซซิเอชัน (Association) เป็นส่วนประกอบของแผนภาพคลาสที่ทำหน้าที่แสดงความสัมพันธ์ระหว่างคลาส (Relationship) ว่ามีความเกี่ยวข้องกันอย่างไร ซึ่งมีส่วนประกอบย่อยๆ ดังนี้
   4.1. ชื่อ (Name) ใช้แสดงชื่อของความสัมพันธ์นั้นๆ เพื่อให้สามารถอ้างอิงได้ถูกต้อง
   4.2. มัลติพลิซิตี (Multiplicity) ใช้แสดงรายละเอียดเชิงจำนวนของความสัมพันธ์ระหว่างคลาสคู่ที่มีความสัมพันธ์กันเช่น 0..1, *, 1..1, 1..*
   4.3. บทบาท (Role) ใช้แสดงถึงบทบาทของคลาสซึ่งถูกมองเห็นจากอีกคลาสหนึ่ง ยกตัวอย่างเช่น คลาสชื่อไก่ อาจมีบทบาทเป็นอาหารเมื่อมองจากมุมมองของคลาสผู้บริโภค ในขณะเดียวกันอาจมีบทบาทเป็นสัตว์เลี้ยงเมื่อมองจากมุมมองของคลาสคนรักสัตว์ เป็นต้น
   4.4. นาวิกาบิลิตี (Navigability) ใช้แสดงถึงทิศทางของความสัมพันธ์ระหว่างคลาสในความสัมพันธ์บางประเภทที่ต้องมีทิศทาง เช่น ความสัมพันธ์ระหว่างคลาส

สมาชิกกับคลาสหนังสือ จะมีทิศทางจากคลาสสมาชิกไปยังคลาสหนังสือ กล่าวคือข้อมูลของสมาชิกจะนำไปสู่ข้อมูลของหนังสือที่สมาชิกคนนั้นเช่าไป

5. แอกกรีเกชัน (Aggregation) เป็นรูปแบบหนึ่งของแอสโซซิเอชันในลักษณะของการแสดงส่วนประกอบยกตัวอย่างเช่น คลาสโรงเรียนประกอบด้วยคลาสครู คลาสนักเรียน คลาสสถานที่เรียน คลาสตารางเรียน และคลาสวิชาเรียน เป็นต้น
6. คอมโพซิชัน (Composition) เป็นรูปแบบหนึ่งของแอกกรีเกชันซึ่งมีเงื่อนไขพิเศษคือ อินสแตนซ์หนึ่งของคลาสใดๆ ก็ตาม สามารถเป็นส่วนประกอบของคลาสประกอบ (Composite Class) อื่นๆ ได้เพียงหนึ่งคลาสในเวลาหนึ่งๆ เท่านั้น ยกตัวอย่างเช่น อินสแตนซ์หนึ่งของคลาสเครื่องยนต์ สามารถเป็นส่วนประกอบของอินสแตนซ์ของคลาสรถยนต์ได้เพียงหนึ่งคลาสในเวลาหนึ่งๆ เท่านั้น (ไม่มีเครื่องยนต์เครื่องใด อยู่ในรถยนต์ได้ทั้ง 2 คัน)
7. เจเนอรัลไลเซชัน (Generalization) เป็นการแสดงความสัมพันธ์ระหว่างคลาสในเชิงของการสืบทอดลักษณะ โดยถือว่าคลาสที่เป็นคลาสลูก (Child) จะมีลักษณะทุกประการของคลาสที่เป็นแม่ (Parent)
8. ดีเพนเดนซี (Dependency) เป็นการแสดงความสัมพันธ์ระหว่างคลาสรูปแบบหนึ่ง กล่าวคือถ้าคลาสสองคลาสใดมีดีเพนเดนซีกัน นั่นคือการเปลี่ยนแปลงภายในคลาสหนึ่งจะส่งผลกระทบต่อการทำงานของอีกคลาสหนึ่งนั่นเอง

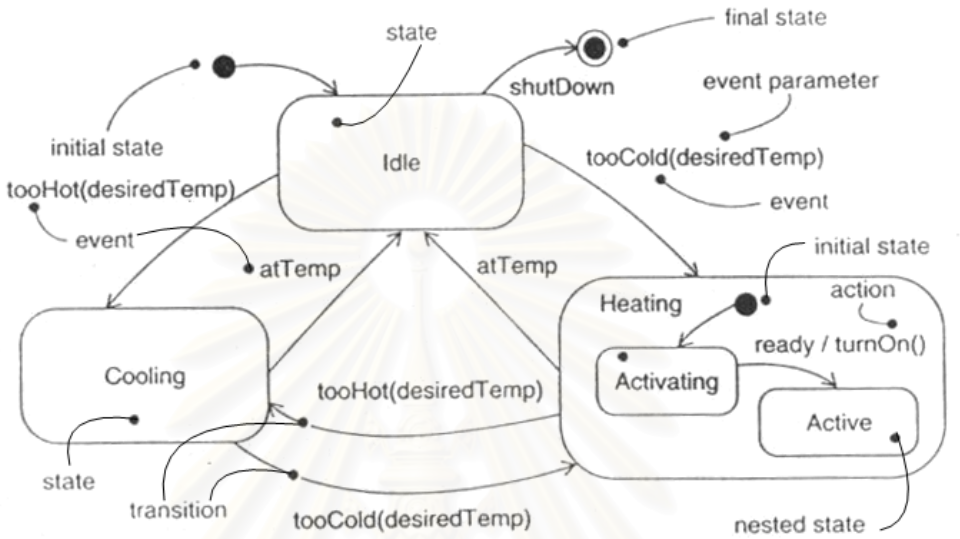
2.1.1.2 แผนภาพแอคทิวิตี



รูปที่ 2.2 ตัวอย่างแผนภาพแอคทิวิตี [5]

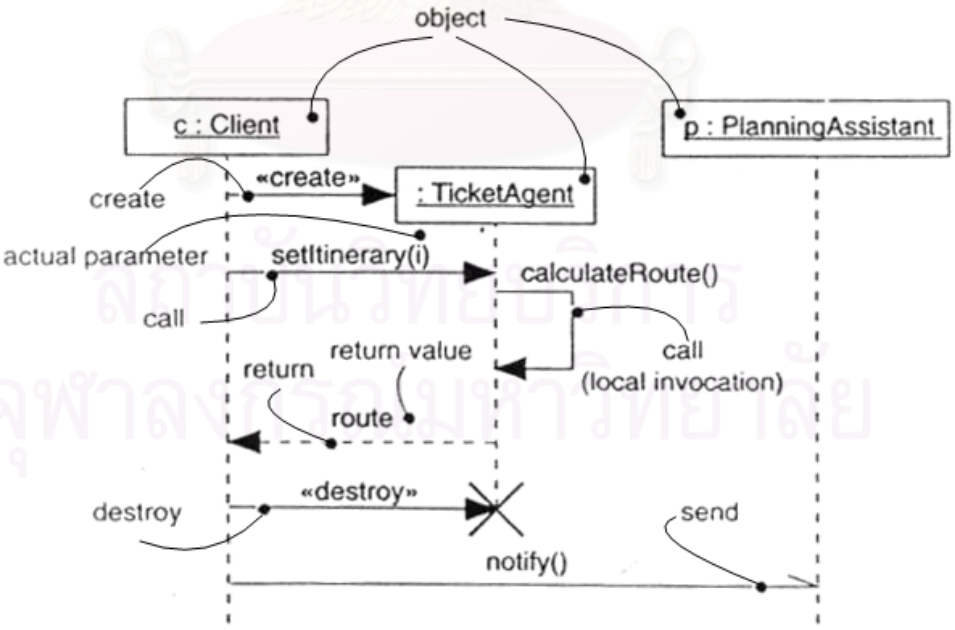
แผนภาพแอคทิวิตีเป็นแผนภาพที่ใช้แสดงลำดับการทำงานของระบบ ยกตัวอย่างดังรูปที่ 2.2 โดยแต่ละแผนภาพจะเริ่มต้นที่สถานะจุดเริ่มต้นการทำงาน (Initial State) และจบที่สถานะสุดท้าย (Final State) เสมอ โดยแต่ละลำดับขั้นของการทำงานนั้นจะเรียกว่า แอคชันสเตท (Action State) ซึ่งในการจัดลำดับการทำงานนั้นสามารถกำหนดเงื่อนไขการทำงานได้โดยตั้งเงื่อนไขการควบคุมการทำงาน (Guard Condition) และกำหนดจุดแยก/เชื่อมการทำงานของแอคชันสเตทที่สามารถทำงานไปพร้อมๆ กันได้ (Concurrent Fork/Join) นอกจากนั้นแผนภาพแอคทิวิตียังมีคุณสมบัติที่มีทั้งหมดของแผนภาพสเตทชาร์ตอีกด้วย ทั้งนี้เนื่องจากในข้อกำหนดของภาษายูเอ็มแอล [4] ระบุว่าแผนภาพแอคทิวิตีนั้นเป็นกรณีพิเศษของแผนภาพสเตทชาร์ต ดังนั้นจึงสามารถใช้แสดงความสามารถของแผนภาพสเตทชาร์ตทั้งหมดได้ด้วย ยกตัวอย่างแผนภาพสเตทชาร์ตดังรูปที่ 2.3 โดยแผนภาพสเตทชาร์ตหนึ่งแผนภาพจะแทนสเตทแมชชีน (State Machine)

หนึ่งตัว ซึ่งแต่ละแผนภาพจะประกอบด้วย สเตท (State) ทรานซิชัน (Transition) อีเวนต์ (Event) และ แอคทิวิตี (Activity) ดังนั้นแผนภาพแอคทิวิตีในฐานะที่เป็นกรณีพิเศษของแผนภาพสเตทชาร์ตก็สามารถใช้แสดงคุณสมบัติเหล่านี้ได้เช่นกัน



รูปที่ 2.3 ตัวอย่างแผนภาพสเตทชาร์ต [5]

2.1.1.3 แผนภาพซีเควนซ์



รูปที่ 2.4 ตัวอย่างแผนภาพซีเควนซ์ [5]

แผนภาพซีเควนซ์เป็นแผนภาพที่ใช้แสดงพฤติกรรมของระบบ ในลักษณะของการแสดงการโต้ตอบ (Interaction) ระหว่างวัตถุ (Object) ยกตัวอย่างเช่นรูปที่ 2.4 โดยทุกแผนภาพจะประกอบไปด้วยแกน 2 แกน แกนแนวนอนคือแกนของวัตถุซึ่งใช้แสดงวัตถุทั้งหมดที่โต้ตอบกัน ส่วนแกนแนวตั้งคือแกนของเวลาโดยไล่จากก่อน (ด้านบน) ไปสู่หลัง (ด้านล่าง) รูปแบบของการโต้ตอบนั้นมีทั้งการเรียก (Call) และการส่งข้อความ (Message)

การพิจารณาระบบจากหลายมุมมอง และสร้างแบบจำลองเป็นหลายแผนภาพเป็นการแยกแยะความซับซ้อนของแต่ละมุมมองออกจากกัน โดยแบ่งเป็น 2 ประเภทหลักๆ ได้แก่ แบบจำลองเชิงโครงสร้าง (Structural Model) และแบบจำลองเชิงพฤติกรรม (Behavioral Model) ทำให้การพิจารณารายละเอียดในมุมมองใดมุมมองหนึ่งทำได้ละเอียดมากขึ้น เนื่องจากความซับซ้อนน้อยลง อย่างไรก็ตามเนื่องจากแต่ละแผนภาพเป็นการมองจากมุมมองที่ต่างกันของระบบเดียวกัน ดังนั้นโดยอุดมคติแล้วทุกแผนภาพต้องสื่อความเข้าใจที่สอดคล้องกันต่อระบบเดียวกัน และต้องสามารถเชื่อมโยงกันได้ ซึ่งในประเด็นความเชื่อมโยงระหว่างแผนภาพนี้ภาษายูเอ็มแอลเวอร์ชันปัจจุบัน (เวอร์ชัน 1.4) ไม่ได้กล่าวถึง โดยมีแนวโน้มที่จะกล่าวถึงเรื่องนี้ในอนาคตในเวอร์ชัน 2.0

2.1.2 เทคโนโลยีมิดเดิลแวร์ (Middleware Technology)

การสร้างระบบซอฟต์แวร์ใหม่ๆ มีแนวโน้มที่จะมีขนาดใหญ่ขึ้น และมีความซับซ้อนเพิ่มมากขึ้นทุกที ซึ่งสิ่งที่ตามมาคือการอิมพลีเมนต์ด้วยการเขียนโค้ดจำนวนมหาศาล ถึงแม้ว่าจะมีการนำแนวคิดของการใช้ใหม่ (Reuse) หรือการจัดทำไลบรารีมาใช้ก็ตาม ขนาดและความซับซ้อนของระบบที่มากขึ้นทำให้เกินความสามารถของคนที่จะโค้ดโปรแกรมคนเดียวให้มีคุณภาพได้

แนวคิดของของเทคโนโลยีมิดเดิลแวร์คือ การแยกส่วนงานที่จำเป็นที่ต้องใช้งานอยู่เสมอๆ ในหลายแอพพลิเคชันมาทำให้เป็นบริการมาตรฐาน เพื่อให้ผู้สร้างระบบไม่ต้องสนใจการเขียนโค้ดสำหรับงานส่วนนี้ เพียงแค่เรียกใช้บริการมาตรฐานต่างๆ ที่มีให้ใช้ในแพลตฟอร์มนั้นๆ ให้ถูกต้อง ทำให้ส่วนของโปรแกรมที่โปรแกรมเมอร์ต้องโค้ดเองมีน้อยลง นอกจากนั้นโปรแกรมที่ได้ยังมีคุณภาพมากขึ้นเนื่องจากบริการมาตรฐานเหล่านี้จะถูกอิมพลีเมนต์ด้วยบริษัทที่เชี่ยวชาญเฉพาะ ทำให้มีระดับความเหมาะสมที่สุด (Optimization) สูง

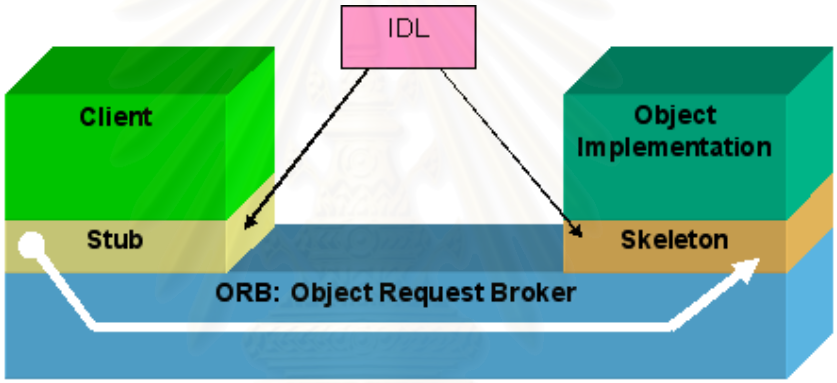
เพื่อให้เข้าใจถึงแนวคิดของเทคโนโลยีมิดเดิลแวร์มากขึ้น ซึ่งจะเป็นพื้นฐานที่สำคัญของแนวคิดเอ็มดีเอ จึงขอยกตัวอย่างเทคโนโลยีมิดเดิลแวร์ 2 ตัวที่เป็นที่นิยมในปัจจุบันได้แก่ คอร์บา ซึ่งพัฒนาโดยโอเอ็มจี และอีเจบีซึ่งพัฒนาโดยผู้พัฒนาภาษาจาวา

2.1.2.1 คอร์บา (CORBA: Common Object Request Broker Architecture) [6]

เป็นมิดเดิลแวร์แพลตฟอร์ม ที่มีจุดเด่นมากในเรื่องความไม่ขึ้นกับภาษาโปรแกรมใด (Language Independent) และถูกพัฒนาในลักษณะของข้อกำหนดมาตรฐานโดยองค์กรที่มีความเป็นกลางได้แก่ โอเอ็มจี ซึ่งข้อกำหนดดังกล่าวจะถูกผู้ผลิตนำไปผลิตเป็นผลิตภัณฑ์อีกทีหนึ่ง ทำให้มีภาพพจน์ของแพลตฟอร์มที่เป็นกลางไม่ยึดติดกับยี่ห้อผลิตภัณฑ์ใดๆ

หลักการสำคัญของคอร์บาคือการกำหนดมาตรฐานการเชื่อมต่อระหว่างคอมโพเนนท์โดยแสดงในรูปแบบของภาษาทางการ (Formal Language) ภาษาหนึ่งเรียกว่าภาษาไอดีแอล (IDL: Interface Definition Language) เป็นการประกาศหน้าตาของส่วนเชื่อมต่อ และวิธีการเรียกใช้งานคอมโพเนนท์ให้เป็นมาตรฐาน ส่วนการอิมพลีเมนต์คอมโพเนนท์นั้นสามารถอิมพลีเมนต์โดยใช้ภาษาโปรแกรมใดๆ ก็ได้ที่คอร์บารองรับ ยกตัวอย่างเช่น ภาษาซี ซีพลัสพลัส หรือจาวา เป็นต้น

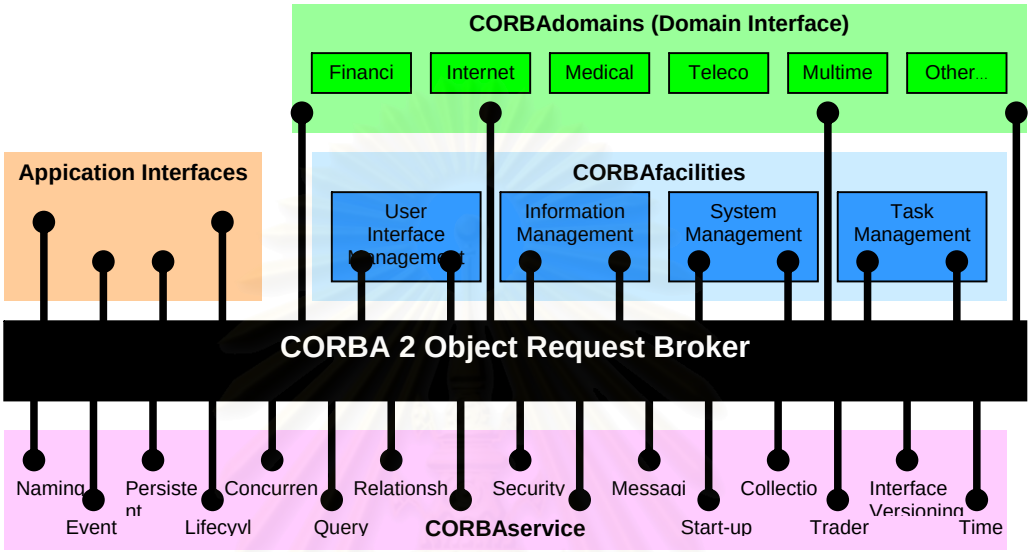
ข้อกำหนดส่วนเชื่อมต่อของคอมโพเนนท์ที่เขียนด้วยภาษาไอดีแอลที่ได้ จะถูกนำไปคอมไพล์เป็นโค้ดส่วนเชื่อมต่อ 2 ส่วนได้แก่ สตับ (Stub) ทางฝั่งผู้ให้บริการ และ สเกเลตัน (Skeleton) ทางฝั่งผู้เรียกใช้บริการ ดังแสดงในรูปที่ 2.5 การที่แต่ละคอมโพเนนท์ที่อิมพลีเมนต์ด้วยคนละภาษาโปรแกรมกันแต่สามารถทำงานด้วยกันได้นั้น เนื่องจากคอมโพเนนท์ผู้เรียกใช้บริการจะมองเห็นคอมโพเนนท์ผู้ให้บริการผ่านสตับ ซึ่งเป็นภาษาโปรแกรมเดียวกับผู้เรียกใช้บริการ จากนั้น สเกเลตันจะสื่อสารกับสตับของผู้ให้บริการผ่าน ออร์บ (ORB: Object Request Broker) ซึ่งสเกเลตันจะไปเรียกใช้โค้ดส่วนอิมพลีเมนต์อีกทีหนึ่ง ดังนั้นออร์บจึงทำหน้าที่เป็นตัวให้บริการสื่อสารระหว่างคอมโพเนนท์ และเป็นตัวแปลภาษาระหว่างสตับกับสเกเลตัน



รูปที่ 2.5 การเรียกใช้บริการของมอดูลผ่านออร์บ

นอกจากนั้นโอเอ็มจียังมีความพยายามกำหนดมาตรฐานของระบบงานในสายงานแต่ละประเภท (Vertical Market) ผ่านคณะทำงานสำหรับสายงาน (Domain Task Force) เนื่องจากเห็นว่าธุรกิจต่างๆ ที่อยู่ในสายงานแต่ละประเภท มักมีความต้องการใช้บริการพื้นฐานบางอย่างที่เฉพาะเจาะจงสำหรับสายงานนั้นๆ เหมือนกัน โดยโอเอ็มจีได้สนับสนุนแนวความคิดดังกล่าวนี้โดยได้สร้างเป็นสถาปัตยกรรมการจัดการวัตถุ (OMA: Object Management Architecture) โดยมีการแบ่งกลุ่มของวัตถุตามรูปที่ 2.6 ซึ่งแสดงถึงการทำงานของแอพพลิเคชันระบบกระจายว่าประกอบขึ้นจากข้อกำหนดของอินเทอร์เฟสต่างๆ (Application Interface) ซึ่งบางอินเทอร์เฟสได้มีการกำหนดไว้แล้วเป็นมาตรฐาน ได้แก่ คอร์บาเซอร์วิส (CORBAservice) ซึ่งเป็นอินเทอร์เฟสของบริการพื้นฐานที่ใช้กันทั่วไปในแอพพลิเคชันใดๆ คอร์บาฟาซิลิตี (CORBAfacilities) ซึ่งเป็นอินเทอร์เฟสสำหรับการทำงานเฉพาะอย่าง ซึ่งไม่ใช่การทำงานพื้นฐานแต่ก็มีที่ใช้ในหลายแอพ

พลิเคชัน และคอร์บาโดเมน (CORBAdomains) ซึ่งเป็นอินเทอร์เฟสสำหรับแอพพลิเคชันเฉพาะสายงานหนึ่งๆ โดยอินเทอร์เฟสเหล่านี้จะทำงานร่วมกันและสื่อสารระหว่างกันโดยใช้ออร์บ



รูปที่ 2.6 สถาปัตยกรรมการจัดการวัตถุ [6]

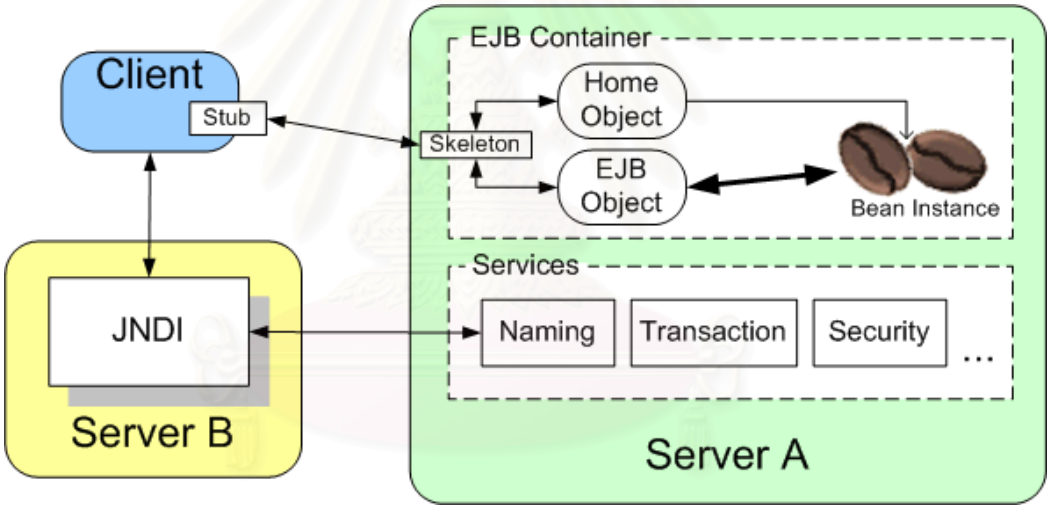
2.1.2.2 อีเจบี (EJB: Enterprise Java Bean) [7]

เป็นมิดเดิลแวร์แพลตฟอร์มของค่ายจาวา ดังนั้นทำให้ภาษาโปรแกรมที่ใช้อิมพลีเมนต์อีเจบีจึงเป็นภาษาจาวาเท่านั้น (ต่างจากคอร์บาที่สามารถเลือกภาษาโปรแกรมใดๆ ก็ได้) อย่างไรก็ตามแพลตฟอร์มอีเจบีมีความได้เปรียบเหนือคอร์บา จากจุดเด่นของภาษาจาวาได้แก่ ความสามารถของการเขียนโปรแกรมครั้งเดียวแต่นำไปทำงานที่ไหนก็ได้ (Write once Run anywhere) รวมทั้งข้อได้เปรียบเรื่องการมีเอพีไอและคอมโพเนนท์สำเร็จรูปสำหรับแพลตฟอร์มจาวาอยู่เป็นจำนวนมาก (ต่างจากคอร์บาที่เน้นเฉพาะการกำหนดอินเทอร์เฟสโดยไม่สนใจการอิมพลีเมนต์)

การพัฒนามอดูลโดยใช้แพลตฟอร์มอีเจบีนั้น มีหลักการที่คล้ายกับคอร์บาเป็นอย่างมาก ดังแสดงในรูปที่ 2.7 โดยคอมโพเนนท์ในแพลตฟอร์มอีเจบีนั้นจะเรียกว่า บีน (Bean) ซึ่งการเรียกใช้งานคอมโพเนนท์บีนนี้จำเป็นต้องเรียกผ่านอินเทอร์เฟสซึ่งมี 2 ชนิดได้แก่ โฮมอินเทอร์เฟส (Home Interface) ซึ่งทำหน้าที่เกี่ยวกับการควบคุมวงจรชีวิตของบีนอินแสตนซ์ (Bean Instance) และรีโมทอินเทอร์เฟส (Remote Interface) ซึ่งเป็นอินเทอร์เฟสของการเรียกใช้งานที่เกี่ยวกับ

ตรรกะเชิงธุรกิจ โดยอีเจบีคอนเทนเนอร์ (EJB Container) จะนำรีโมทอินเทอร์เฟสไปสร้างเป็นสตับ (Stub) และสเกเลตัน (Skeleton) คล้ายกับไอดีแอลของคอร์บา ทั้งนี้ส่วนประกอบทั้งหมดของระบบจะต้องทำงานอยู่ในสภาพแวดล้อมของอีเจบีคอนเทนเนอร์ ซึ่งจะจัดการสร้างสภาพแวดล้อมที่จำเป็นของแพลตฟอร์มอีเจบีให้แก่ บีนอินสแตนซ์

ขั้นตอนของการทำงานเริ่มจากเครื่องไคลเอนต์ ติดต่อไปยังเครื่องเซิร์ฟเวอร์เครื่องหนึ่งในที่นี้กำหนดให้เป็นเซิร์ฟเวอร์บี (Server B) ซึ่งมีบริการเจเอ็นดีไอ (JNDI: Java Naming and Directory Interface) เพื่อหาเรฟเฟอเรนซ์ของโฮมอ็อบเจกต์ (ซึ่งเครื่องเซิร์ฟเวอร์บีจะไปขอมาจากบริการเนมมิงของเครื่องเซิร์ฟเวอร์เออีกทีหนึ่ง) เมื่อไคลเอนต์ทราบเรฟเฟอเรนซ์ของโฮมอ็อบเจกต์แล้ว จะสั่งให้โฮมอ็อบเจกต์สร้างบีนอินสแตนซ์ขึ้นมา และส่งเรฟเฟอเรนซ์ของอีเจบีอ็อบเจกต์ให้ไคลเอนต์ จากนั้นไคลเอนต์จึงจะสามารถเรียกใช้งานบีนอินสแตนซ์ได้ โดยต้องเรียกผ่านอีเจบีอ็อบเจกต์เสมอ

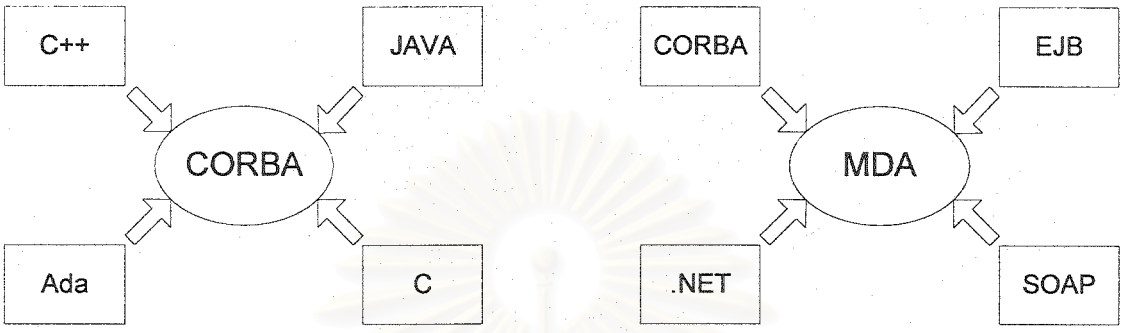


รูปที่ 2.7 ตัวอย่างการสร้างมอดูลด้วยแพลตฟอร์มอีเจบี [7]

2.1.3 เอ็มดีเอ (MDA: Model Driven Architecture) [3]

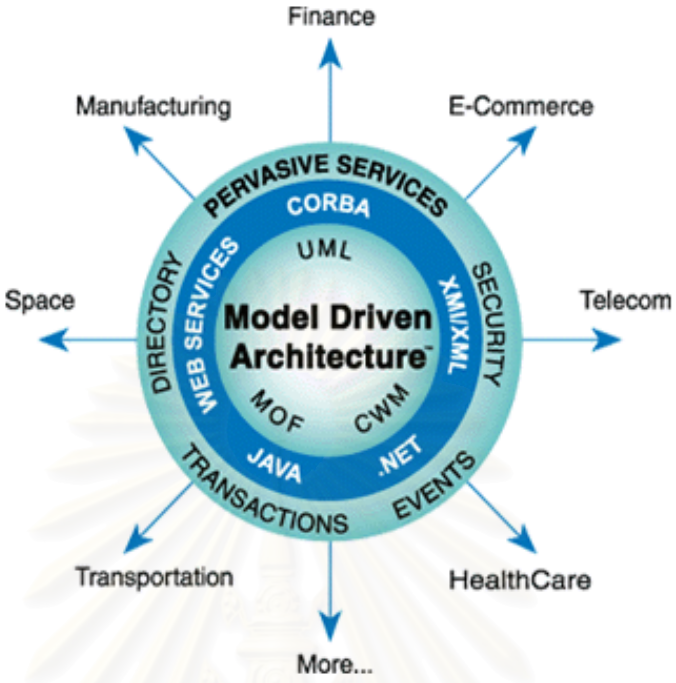
นอกจากคอร์บาและมาตรฐานอื่นในการทำงานร่วมกันของระบบคอมพิวเตอร์เช่น ซีดับเบิลยูเอ็ม (CWM: Common Warehouse Metamodel) แล้ว โอเอ็มจียังได้พัฒนาแนวคิดเอ็มดีเอขึ้นมาด้วย โดยเอ็มดีเอนั้นจะกล่าวถึงการทำงานร่วมกันของระบบที่มีมิดเดิลแวร์แพลตฟอร์มที่ต่างกัน หรือการเปลี่ยนแปลงระบบเดิมไปใช้แพลตฟอร์มใหม่ที่แตกต่างกันออกไป จึงเป็นแนวคิดที่มีการทำงานคนละระดับกับคอร์บาซึ่งจะพูดถึงการทำงานร่วมกันของคอมโพเนนท์ ที่เขียนขึ้นจาก

คนละภาษาโปรแกรมกัน หรืออยู่บนระบบปฏิบัติการที่ต่างกัน โดยแสดงการเปรียบเทียบดังรูปที่ 2.8



รูปที่ 2.8 เปรียบเทียบความคล้ายคลึงกันของคอร์บาและเอ็มดีเอ

เอ็มดีเอเป็นแนวคิดที่เกี่ยวกับการเขียนข้อกำหนด และพัฒนาระบบ โดยเริ่มจากการสร้างแบบจำลองของธุรกิจที่ไม่ขึ้นกับแพลตฟอร์ม หรือพีไอเอ็ม (PIM: Platform Independent Model) ซึ่งแยกออกจากอีกส่วนหนึ่งคือแบบจำลองที่ขึ้นกับแพลตฟอร์มหรือพีเอสเอ็ม (PSM: Platform Specific Model) การแยกแบบจำลองออกเป็นสองระดับนั้น มีวัตถุประสงค์เพื่อรองรับการเปลี่ยนแปลงของมิดเดิลแวร์แพลตฟอร์ม และการทำงานร่วมกับแพลตฟอร์มที่ต่างกัน เนื่องจากพีไอเอ็มเป็นแบบจำลองของระบบที่จะไม่เปลี่ยนแปลง ตราบเท่าที่ความต้องการเชิงธุรกิจไม่เปลี่ยนไป ซึ่งถ้าแบบจำลองพีไอเอ็มของระบบมีความเสถียรเพียงพอ การสร้างแบบจำลองพีเอสเอ็มสำหรับแพลตฟอร์มใหม่ๆ นั้นจะสามารถทำได้ผ่านมาตรฐานการแปลงไปยังแพลตฟอร์มนั้นๆ ซึ่งจะถูกกำหนดขึ้นในอนาคต ดังนั้นจึงกล่าวได้ว่าการสร้างแบบจำลองพีไอเอ็ม และแบบจำลองพีเอสเอ็มที่แยกจากกันนั้น ช่วยลดความยุ่งยากในการทำงานร่วมกันระหว่างหลายระบบที่มีหลายแพลตฟอร์มได้



รูปที่ 2.9 องค์ประกอบของเอ็มดีเอ [3]

องค์ประกอบของเอ็มดีเอตามรูปที่ 2.9 ประกอบด้วย 4 ส่วนได้แก่

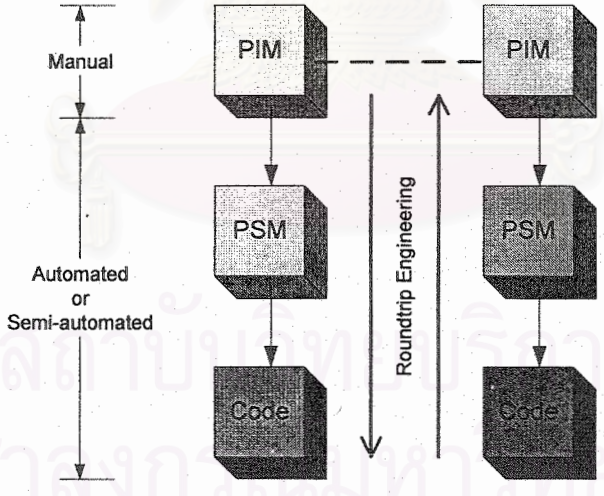
1. แบบจำลองที่ไม่ขึ้นกับแพลตฟอร์ม หรือพีไอเอ็ม (Platform Independent Model) แบ่งเป็น 2 ระดับ ได้แก่
   1.1. พีไอเอ็มฐาน (Base PIM) เป็นแบบจำลองที่แสดงเฉพาะฟังก์ชันงานทางธุรกิจ (Business Functionality) ซึ่งเป็นฐานความต้องการของระบบ โดยไม่คำนึงถึงเทคโนโลยีที่จะใช้ในการพัฒนาระบบ
   1.2. พีไอเอ็มระดับล่าง (Next Level PIM) เป็นแบบจำลองที่มีการให้รายละเอียดเกี่ยวกับความต้องการเชิงเทคนิค แต่ไม่ขึ้นกับแพลตฟอร์ม เช่น ความต้องการเกี่ยวกับ ความคงสภาพ (Persistence) การทำรายการเปลี่ยนแปลง (Transactionality) ความมั่นคง (Security) หรือโครงแบบ (Configuration) ของระบบเป็นต้น
2. แบบจำลองที่ขึ้นกับแพลตฟอร์ม หรือพีเอสเอ็ม (Platform Specific Model) เป็นแบบจำลองที่มีการเพิ่มเติมรายละเอียดการพัฒนาระบบตามแพลตฟอร์มเป้าหมาย เช่น คอร์บา เว็บเซอร์วิส อีเจบี เป็นต้น ซึ่งแต่ละแพลตฟอร์ม จะมีการกำหนดมาตรฐานของการแสดงความหมายที่เฉพาะเจาะจงสำหรับแพลตฟอร์มนั้นในรูปของ

ยูเอ็มแอลโปรไฟล์ (UML Profile) ซึ่งเป็นชุดของมาตรฐานการเพิ่มขยายสำหรับภาษายูเอ็มแอล เพื่อใช้ในงานอย่างใดอย่างหนึ่ง โดยในปัจจุบันมีมาตรฐานยูเอ็มแอลโปรไฟล์ที่ขึ้นกับแพลตฟอร์มเพียงตัวเดียว ได้แก่มาตรฐานยูเอ็มแอลโปรไฟล์สำหรับคอร์บารุ่น 1.0 [8]

3. เพอร์วาซีฟเซอร์วิส (Pervasive Service) เป็นการกำหนดแบบจำลองในระดับพีไอเอ็ม ของบริการ (Service) พื้นฐานในการพัฒนาระบบแอพพลิเคชัน ซึ่งโดยทั่วไปแอพพลิเคชันมักมีความต้องการใช้บริการพื้นฐานบางอย่างร่วมกัน เช่น บริการไดเรกทอรี (Directory Service) บริการด้านความมั่นคง (Security Service) เป็นต้น ดังนั้นเพอร์วาซีฟเซอร์วิสจึงเทียบได้กับแบบจำลองของคอร์บาเซอร์วิสนั่นเอง โดยผู้ออกแบบสามารถออกแบบแอพพลิเคชันของตนโดยนำแบบจำลองของเพอร์วาซีฟเซอร์วิสไปใช้ได้ และสามารถอิมพลีเมนต์เครื่องมือแมปปิงมาช่วย
4. โดเมนฟาซิลิตี (Domain Facilities) เป็นการกำหนดแบบจำลองมาตรฐานสำหรับงานเฉพาะอย่างหรือธุรกิจแต่ละสายงาน โดยจะออกมาในรูปของแบบจำลองพีไอเอ็ม ซึ่งมีครั้งโครงสร้างและพฤติกรรมของระบบมาตรฐาน นอกจากนั้นยังมีการกำหนดเอ็กซ์เอ็มแอลสกีมา เพื่อใช้เป็นมาตรฐานคำศัพท์ในวงการนั้นๆ อีกด้วย ทั้งนี้เพื่อให้หน่วยงานต่างๆ ในสายงานเดียวกันมีการยึดแบบจำลองมาตรฐานร่วมกัน และใช้คำศัพท์มาตรฐานในวงการนั้นๆ ในการสื่อสาร ทำให้ระบบของแต่ละองค์กรมีความเป็นไปได้ที่จะสามารถทำงานร่วมกันมากขึ้น โดเมนฟาซิลิตีจึงเทียบได้กับคอร์บาฟาซิลิตี หรือคอร์บาโดเมน

เอ็มดีเอไม่ได้ถูกสร้างขึ้นเพื่อแทนที่เทคโนโลยีใด ในทางตรงข้าม เอ็มดีเอเป็นการรวมกันของเทคโนโลยีต่างๆ ให้สามารถทำงานร่วมกันได้ หากเปรียบเทียบกระบวนการพัฒนาระบบด้วยเอ็มดีเอ กับแนวทางการพัฒนาอื่นๆ เช่น ซีบีดี (CBD: Component Based Design) ดีไซน์แพทเทิร์น (Design Pattern) คอนแทรกเบสดีไซน์ (Contract Based Design) อาร์ยูพี (RUP: Rational Unified Process) จะพบว่าเอ็มดีเอกับวิธีการออกแบบเหล่านี้อยู่กันคนละระดับจึงเปรียบเทียบกันไม่ได้ โดยเอ็มดีเอจะอยู่ในระดับของเฟรมเวิร์ก (Framework) และไม่ได้กำหนดกระบวนการพัฒนาอย่างชัดเจนนัก ดังนั้นจึงสามารถประยุกต์วิธีการออกแบบต่างๆ ข้างต้น มาใช้ในกระบวนการพัฒนาระบบด้วยเอ็มดีเอได้ นอกจากนี้เอ็มดีเอยังเป็นเฟรมเวิร์กที่มีความเป็นมาตรฐานสูง เนื่องจากการใช้เทคโนโลยีเกี่ยวกับการออกแบบที่เป็นมาตรฐานอื่นๆ เป็นพื้นฐาน เช่น

1. ยูเอ็มแอล (Unified Modeling Language) เป็นภาษามาตรฐานที่เอ็มดีเอ ใช้ในการอธิบายแบบจำลองของระบบในลักษณะรูปภาพ ซึ่งยูเอ็มแอลมีจุดเด่นที่สามารถอธิบายระบบได้จากหลายมุมมองผ่านการสื่อความหมายของแผนภาพ 9 ชนิด ที่สามารถเลือกใช้งานได้ตามความเหมาะสม นอกจากนี้ยังมีกลไกที่รองรับการเพิ่มขยายความหมายใหม่ๆ ทำให้สามารถประยุกต์ใช้ในการสร้างแบบจำลองได้อย่างกว้างขวาง ซึ่งได้กล่าวรายละเอียดแล้วในหัวข้อ 2.1.1
2. เอ็มโอเอฟ (Meta-Object Facility) เป็นสถาปัตยกรรมมาตรฐาน ในการจัดการกับเมตาดาตาทุกชนิดด้วยมาตรฐานเดียวกัน ทำให้เกิดความสามารถทำงานร่วมกันระหว่างเมตาเดตาคนละชุดได้ เนื่องจากสามารถกำหนดกฎการแปลงระหว่างชุดของเมตาดาตาที่ต่างกันได้ โดยรายละเอียดในประเด็นนี้จะกล่าวถัดไปในหัวข้อ 2.1.4
3. ซีดับเบิลยูเอ็ม (Common Warehouse Metamodel) เป็นมาตรฐานการแสดงแบบจำลองของคลังข้อมูลแบบต่างๆ ซึ่งอยู่นอกเหนือขอบเขตของงานวิจัยชิ้นนี้
4. เอ็กซ์เอ็มไอ (XML Metadata Interchange) เป็นมาตรฐานการแลกเปลี่ยน (Interchange) ของแบบจำลองในรูปแบบของภาษาเอ็กซ์เอ็มแอล



รูปที่ 2.10 การพัฒนาระบบด้วยแนวคิดเอ็มดีเอ

จากแนวคิดเอ็มดีเอ ทำให้เห็นแนวทางการพัฒนาระบบที่เป็นไปได้ในอนาคต ดังที่แสดงในรูปที่ 2.10 โดยการพัฒนาโดยส่วนใหญ่ จะเป็นการออกแบบตรรกะเชิงธุรกิจที่ระดับพีไอเอ็ม ส่วนในระดับล่างๆ นั้นจะพัฒนาโดยใช้เครื่องมือกึ่งอัตโนมัติช่วย ทำให้ปริมาณงานที่ต้องทำในการ

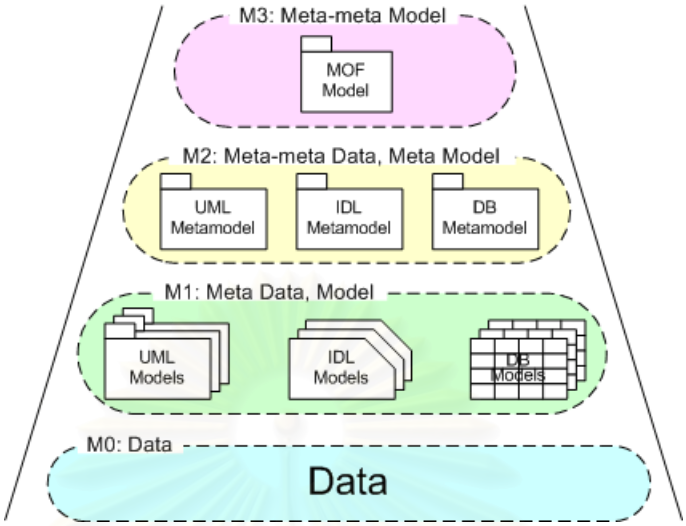
พัฒนาระบบมีน้อยลง และระบบที่ได้จะมีการตอบสนองต่อการเปลี่ยนแปลงของเทคโนโลยีมากขึ้น รวมทั้งสามารถเชื่อมต่อระบบอื่นๆ โดยการเชื่อมโยงแบบจำลองที่ระดับพีไอเอ็ม ส่วนการเชื่อมต่อในระดับล่างนั้นจะเกิดเองโดยอัตโนมัติ ผ่านการจัดการของเครื่องมือกึ่งอัตโนมัติที่รองรับการพัฒนาแบบเอ็มดีเอ รวมทั้งอาจสามารถส่งต่อการเปลี่ยนแปลง (Change Propagation) ที่เกิดขึ้นในแต่ละระดับไปยังระดับอื่นในทั้ง 2 ทิศทางเรียกว่าวิศวกรรมไปกลับ (Roundtrip Engineering)

ประโยชน์ที่ได้จากการพัฒนาระบบด้วยแนวคิดเอ็มดีเอได้แก่

1. ในการพัฒนาแอพพลิเคชัน องค์กรจะมุ่งไปที่การวิเคราะห์และออกแบบระบบตามความต้องการของธุรกิจเพื่อสร้างพีไอเอ็ม จึงทำให้เกิดความเข้าใจในระบบงานอย่างลึกซึ้ง ทำให้ได้แอพพลิเคชันที่ตรงกับความต้องการและมีข้อผิดพลาดน้อยลง
2. แอพพลิเคชันสามารถรองรับการเปลี่ยนแปลง ทั้งในแง่การเปลี่ยนแปลงของเทคโนโลยี หรือการบูรณาการกับระบบอื่นๆ ซึ่งจะทำให้ค่าใช้จ่ายเกี่ยวกับการทำให้ระบบสามารถทำงานร่วมกันได้ และการบูรณาการลดลง
3. นักพัฒนาระบบจะมีความยืดหยุ่นในการเลือกแพลตฟอร์มมากขึ้น เนื่องจากสามารถสร้างแบบจำลองพีเอสเอ็ม และโค้ดใหม่ บนแพลตฟอร์มใหม่ ที่ได้จากแบบจำลองพีไอเอ็มเดิมที่มีเสถียรภาพโดยการกำหนดมาตรฐานการแปลงใหม่ ซึ่งเป็นการเพิ่มคุณค่าและความสำคัญของแบบจำลองและกระบวนการพัฒนาแบบจำลองของระบบ
4. แอพพลิเคชันดั้งเดิม (Legacy Application) หลังจากผ่านกระบวนการทำแรปปิง (Wrapping) และสร้างแบบจำลองของระบบแล้วก็สามารถจะทำการบูรณาการกับระบบอื่นๆ ได้
5. วงการธุรกิจต่างๆ จะมีระบบที่สามารถทำงานร่วมกับระบบภายนอกได้มากขึ้น เนื่องจากมีการกำหนดมาตรฐานร่วมของแต่ละวงการธุรกิจ ทำให้ระบบงานต่างๆ เชื่อมต่อกันได้ง่ายขึ้น และมีความเป็นมาตรฐานมากขึ้น

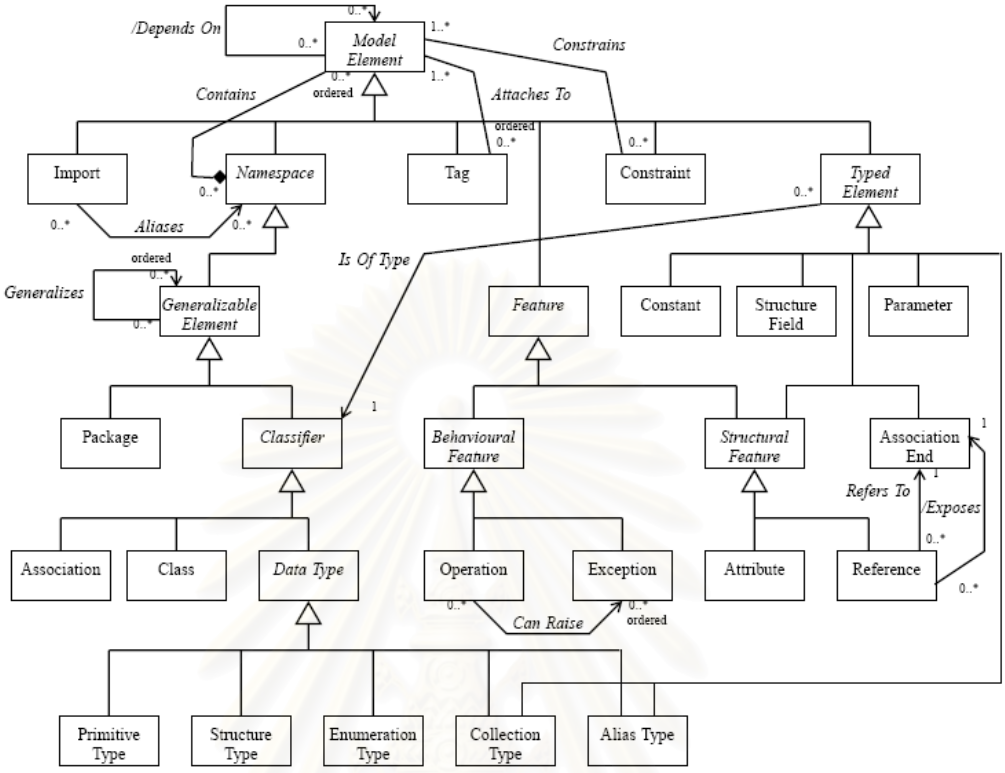
2.1.4 เอ็มโอเอฟ (MOF: Meta Object Facility) [9]

เป็นมาตรฐานของระบบการจัดการเมตาโมเดลซึ่งกำหนดโดยโอเอ็มจี การจัดการเมตาโมเดลในที่นี้ รวมถึงทั้งวิธีการสร้างเมตาโมเดล ภาษาที่ใช้สร้างเมตาโมเดล การจัดเก็บเมตาโมเดล และการเชื่อมโยงกันระหว่างเมตาโมเดล กลไกที่สำคัญของเอ็มโอเอฟในการจัดการกับเมตาโมเดล คือ การกำหนดมาตรฐานของสถาปัตยกรรมเอ็มโอเอฟดังรูปที่ 2.11



รูปที่ 2.11 สถาปัตยกรรมเอ็มโอเอฟ [9]

สถาปัตยกรรมเอ็มโอเอฟกำหนดกรอบของวิธีการอธิบายสิ่งต่างๆ โดยมีการแบ่งเป็นชั้นๆ โดยเริ่มจากระดับล่างสุดได้แก่ ระดับเอ็มศูนย์ (M0) ซึ่งหมายถึงข้อมูล (Data) และข่าวสาร (Information) ต่างๆ ซึ่งจะถูกอธิบายด้วยระดับเอ็มหนึ่ง (M1) ที่เรียกว่า เมตาดาตา (Metadata) หรือโมเดล (Model) ซึ่งเป็นการอธิบายถึงลักษณะเชิงโครงสร้าง หรือลักษณะเชิงพฤติกรรมของข้อมูลนั้นๆ ระดับถัดมาคือระดับเอ็มสอง (M2) เรียกว่า เมตาเมตาดาตา (Meta-meta data) หรือเมตาโมเดล (Metamodel) ซึ่งจะอธิบายถึงวิธีการสร้างโมเดล การกำหนดโครงสร้างโมเดล (Model Construct) และความหมายของแต่ละชิ้นส่วนซึ่งใช้ในการประกอบขึ้นมาเป็นโมเดล และสุดท้ายคือระดับบนสุดหรือ ระดับเอ็มสาม (M3) เรียกว่าเมตาเมตาโมเดล (Meta-meta model) ซึ่งจะอธิบายถึงวิธีการสร้างเมตาโมเดล ซึ่งเอ็มโอเอฟกำหนดว่าระดับเอ็มสามนี้ต้องใช้ภาษาเอ็มโอเอฟเท่านั้น ดังนั้นเอ็มโอเอฟจึงเป็นภาษาที่อยู่ในระดับบนสุดสำหรับใช้ในการอธิบายเมตาโมเดลทุกประเภท ดังแสดงตัวอย่างของส่วนหนึ่งของเอ็มโอเอฟโมเดลดังรูปที่ 2.12 ซึ่งจะสังเกตได้ว่าเอ็มโอเอฟโมเดลนี้มีความคล้ายคลึงกับยูเอ็มแอลเมตาโมเดลคอร์แพคเกจ (Core Package) เป็นอย่างมาก ทั้งนี้เนื่องจากเอ็มโอเอฟนำโนเทชันต่างๆ ของภาษายูเอ็มแอลมาใช้ในการอธิบายเมตาโมเดลต่างๆ ดังนั้นจึงสามารถเขียนอธิบายเมตาโมเดลด้วยคลาสไดอะแกรมได้ ซึ่งคลาสไดอะแกรมนี้จริงๆ แล้วเป็น เอ็มโอเอฟคลาสไดอะแกรม ไม่ใช่ยูเอ็มแอลคลาสไดอะแกรม เนื่องจากสร้างขึ้นโดยอาศัยเอ็มโอเอฟโมเดล มิใช่ยูเอ็มแอลเมตาโมเดล

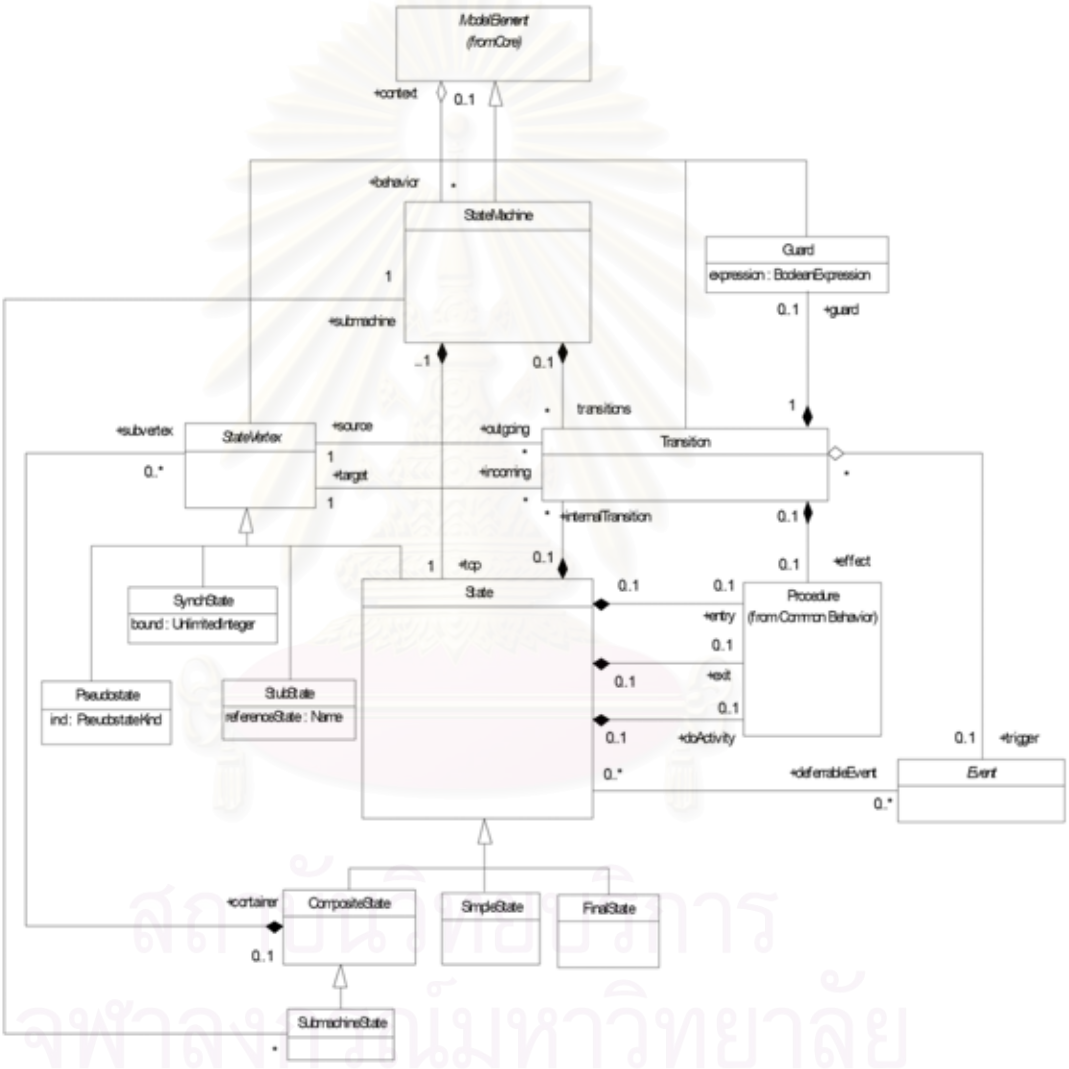


รูปที่ 2.12 ส่วนหนึ่งของเอ็มโอเอฟโมเดล [9]

การกำหนดให้ระดับเอ็มสามจำเป็นต้องเป็นเอ็มโอเอฟเท่านั้นเนื่องจากสถาปัตยกรรมเอ็มโอเอฟมีสมมติฐานว่าทุกเมตาโมเดลสามารถถูกอธิบายความหมายด้วยภาษาเอ็มโอเอฟได้ทั้งหมด และเหตุที่ไม่ต้องมีระดับเอ็มสี่หรือเอ็มห้าต่อขึ้นไปเรื่อยๆ นั้นเนื่องจากภาษาเอ็มโอเอฟสามารถอธิบายตัวเองได้ (Self-describe) ทั้งนี้ผู้วิจัยใช้คำว่าภาษาแทนคำว่าเมตาเมตาโมเดลนั้นเนื่องจากเมตาเมตาโมเดลใช้เขียนแสดงความหมายของเมตาโมเดล จึงมองเป็นภาษาในการเขียนเมตาโมเดล (ซึ่งในหัวข้อถัดๆ ไปก็จะใช้คำว่าภาษาแทนคำว่าเมตาโมเดลเนื่องจากมองว่าเป็นการเขียนอธิบายโมเดล)

ตัวอย่างในการใช้เอ็มโอเอฟในการอธิบายเมตาโมเดลนั้นแสดงดังรูปที่ 2.13 และ 2.14 โดยในรูปที่ 2.13 เป็นการแสดงส่วนหนึ่งของเมตาโมเดลของแผนภาพสเตทชาร์ต ซึ่งสิ่งที่แสดงในเมตาโมเดลดังกล่าวทั้งหมดนั้นอาศัยกลไกที่ได้กำหนดขึ้นในเอ็มโอเอฟโมเดลทั้งสิ้น ยกตัวอย่าง เช่น สี่เหลี่ยมแต่ละรูปแสดงเมตาคลาส (ในรูปที่ 2.12 ใช้คำว่า Class) ซึ่งแต่ละเมตาคลาสมีคุณสมบัติเป็นอีลีเมนต์ที่มีการสืบทอดได้ (Generalizable Element) เนื่องจากเมตาคลาสสืบทอด

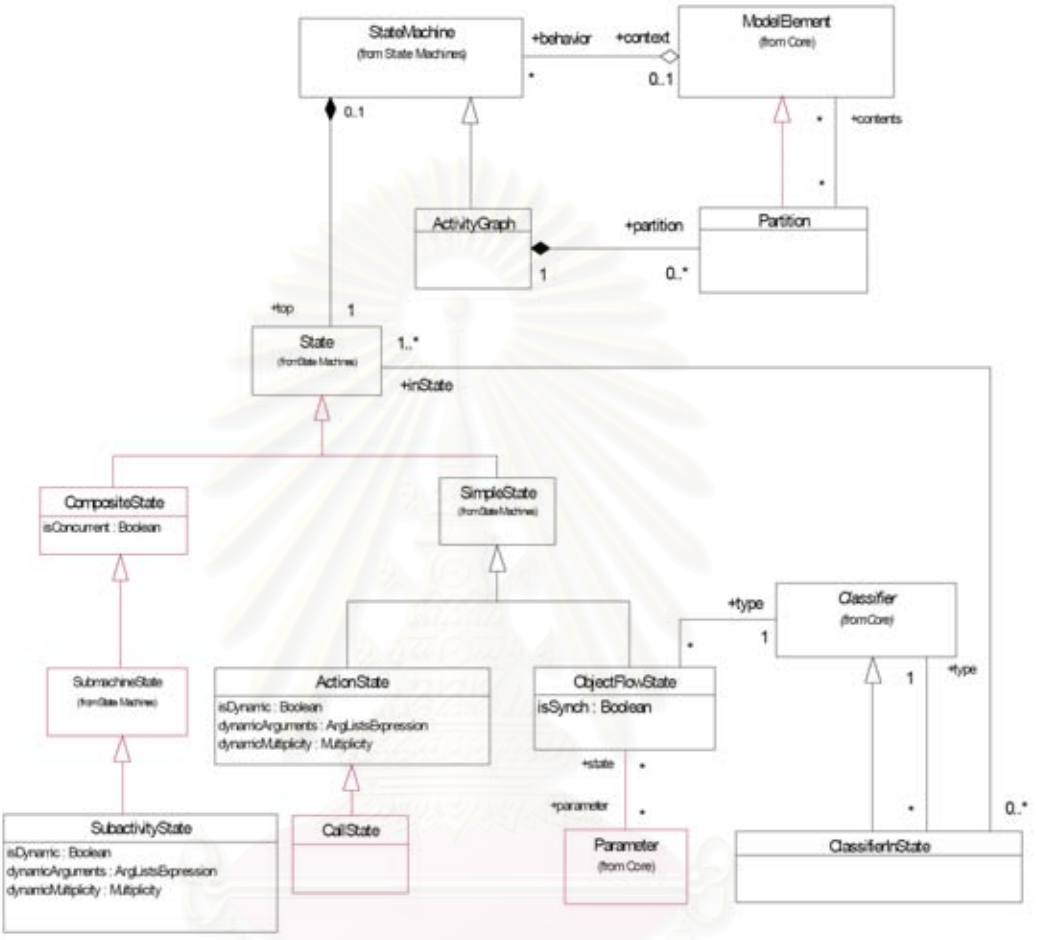
มาจากคลาสสิฟายเออร์ (Classifier) ซึ่งสืบทอดมาจากอีลีเมนต์ที่มีการสืบทอดได้ อีกทีหนึ่ง นอกจากนั้นยังมีการระบุความสัมพันธ์ระหว่างคลาสได้ (Association) เนื่องจากในเมตาโมเดลระบุไว้ว่าคลาสสิฟายเออร์ใดๆ สามารถมีความสัมพันธ์กับคลาสสิฟายเออร์อื่นได้ (รายละเอียดดังกล่าวอยู่ในส่วนของเมตาโมเดลส่วนอื่นซึ่งไม่ได้แสดงในรูปที่ 2.12) กล่าวโดยสรุปคือการสร้างเมตาโมเดลนั้นจำเป็นต้องอ้างอิงการแสดงความหมายตามที่เอ็มโอเอฟโมเดลกำหนดไว้



รูปที่ 2.13 ส่วนหนึ่งของเมตาโมเดลของแผนภาพสเตทชาร์ต [4]

ตัวอย่างการอ่านความหมายที่กำหนดในเมตาโมเดลรูปที่ 2.13 เช่น StateMachine เป็น ModelElement ชนิดหนึ่งซึ่งประกอบด้วย State และ Transition ซึ่งแต่ละ Transition อาจจะมีอีเวนต์หรือไม่มีก็ได้ อาจจะมี Guard หรือไม่มีก็ได้ (พิจารณาจากมัลติพลิซิตี) นอกจากนั้นยัง

สามารถมี CompositeState ซึ่งประกอบด้วยสเตทย่อยภายในได้ โดย CompositeState นั้นก็ถือเป็น State ชนิดหนึ่งด้วยเช่นกัน



รูปที่ 2.14 ส่วนหนึ่งของเมตาโมเดลของแผนภาพแอคทิวิตี [4]

อีกตัวอย่างได้แก่การอ่านเมตาโมเดลจากรูปที่ 2.14 เช่น ActivityGraph เป็น StateMachine ชนิดหนึ่งซึ่งมีการกำหนดคอนเซปต์ของ ActionState ซึ่งเป็น State ชนิดหนึ่งขึ้น นอกจากนั้นแต่ละสเตทยังอาจเลือกที่จะทำตัวเป็น CompositeState โดยในแผนภาพแอคทิวิตีจะเรียกว่า SubactivityState เป็นต้น

การกำหนดเมตาโมเดลโมเดลนั้นมีความสำคัญมากต่อเทคโนโลยีของเอ็มดีเอ เนื่องจากเมตาโมเดลให้รายละเอียดของโมเดล 2 ประการได้แก่ วายสัมพันธ์เชิงนามธรรม (Abstract Syntax) และ ความหมาย (Semantic) ของส่วนโครงสร้างโมเดล (Model Construct) ซึ่งการแยกวาย

สัมพันธ์เชิงนามธรรมออกจากเชิงรูปธรรมนั้นทำให้โครงสร้างโมเดลนั้นๆ ไม่จำเป็นต้องมีรูป หรืออาจอาศัยรูปของโครงสร้างโมเดลอื่นๆ เช่นยูเอ็มแอลโปรไฟล์ต่างๆ ก็อาศัยรูปของภาษายูเอ็มแอลเพื่อสื่อความหมายที่เฉพาะเจาะจงลงไปตามที่เมตาโมเดลกำหนด

2.1.5 ยูเอ็มแอลโปรไฟล์สำหรับอีด็อก (UML Profile for EDOC) [10]

จากข้อกำหนดของภาษายูเอ็มแอลได้ให้ความหมายของ ยูเอ็มแอลโปรไฟล์ไว้ว่าเป็นเซตของส่วนต่อขยาย (Extension) ของภาษายูเอ็มแอล เพื่อให้ภาษายูเอ็มแอลสามารถแสดงความหมายที่เฉพาะเจาะจง หรือแตกต่างไปจากความหมายปกติที่ยูเอ็มแอลสามารถแสดงได้ โดยแต่ละโปรไฟล์ จะมีจุดประสงค์ในการทำให้ยูเอ็มแอลสามารถสื่อความทีเฉพาะเจาะจงในประเด็นนั้นๆ แตกต่างกันไป

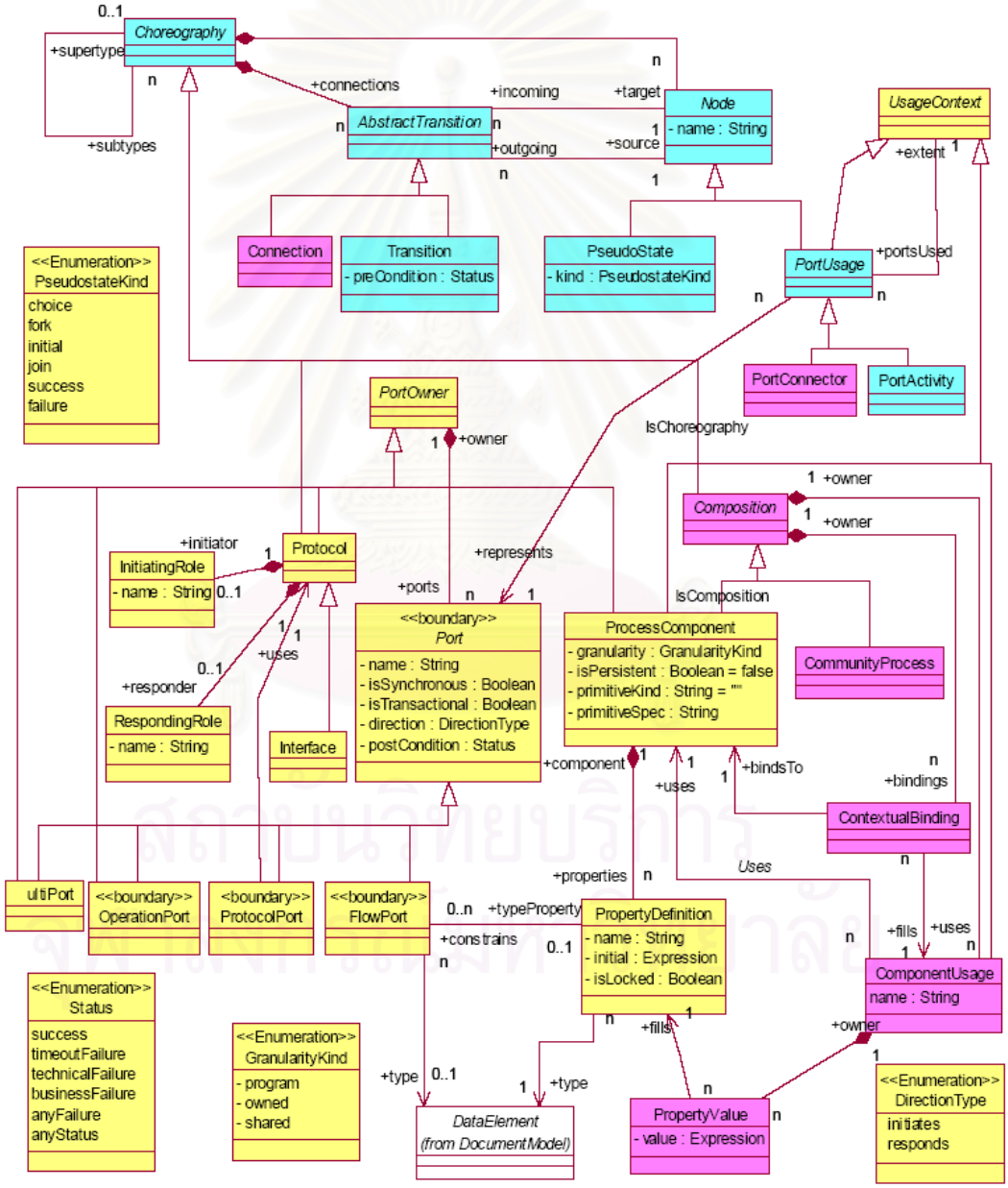
ยูเอ็มแอลโปรไฟล์สำหรับอีด็อก เป็นโปรไฟล์ชนิดหนึ่งซึ่งกำหนดเป็นมาตรฐานโดยโอเอ็มจี มีจุดประสงค์ในการทำให้ยูเอ็มแอลสามารถสื่อความหมาย ที่เกี่ยวข้องกับการประมวลผลของระบบกระจายเชิงวัตถุระดับวิสาหกิจ (EDOC: Enterprise Distributed Object Computing) โดยประกอบด้วยโปรไฟล์ย่อยถึง 5 ชนิด ได้แก่

1. ซีซีเอโปรไฟล์ (CCA: Component Collaboration Architecture)
2. เอนทิตีโปรไฟล์ (Entity Profile)
3. อีเวนต์โปรไฟล์ (Event Profile)
4. บิซิเนสโปรเซสโปรไฟล์ (Business Process Profile)
5. รีเลชันชิพโปรไฟล์ (Relationship Profile)

ในทีนี้จะขออธิบายเฉพาะซีซีเอโปรไฟล์เท่านั้น เนื่องจากในงานวิจัยชิ้นนี้เลือกใช้งานเฉพาะโปรไฟล์นี้เท่านั้น โดยนำซีซีเอโปรไฟล์ไปสร้างแบบจำลองของระบบที่ไม่ขึ้นกับแพลตฟอร์มหรือพีไอเอ็ม เหตุผลที่เลือกเพียงหนึ่งโปรไฟล์จากทั้งหมดเนื่องจากไม่ต้องการให้แบบจำลองที่ได้มีความซับซ้อนมากเกินไป โดยได้ยกตัวอย่างการใช้งานซีซีเอโปรไฟล์ไว้ในหัวข้อที่ 4.3

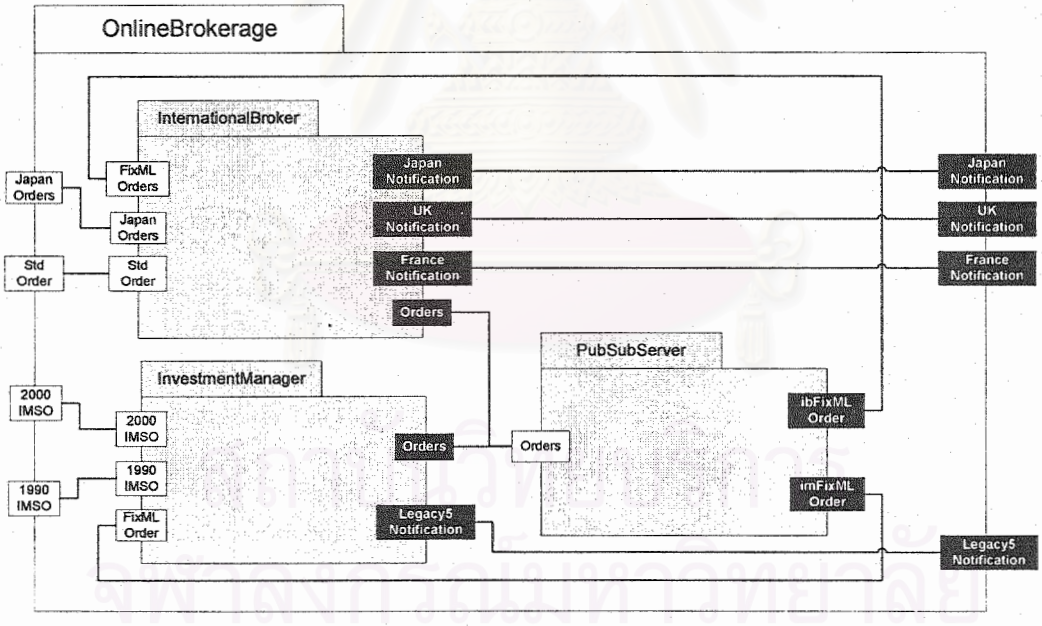
ซีซีเอโปรไฟล์เป็นยูเอ็มแอลโปรไฟล์ ซึงถูกสร้างขึ้นเพื่อช่วยให้ยูเอ็มแอลสามารถสื่อความหมายที่เฉพาะเจาะจง เกี่ยวกับการอธิบายระบบที่ประกอบด้วยการทำงานเชื่อมโยงกันระหว่างคอมโพเนนท์ ซึ่งจะเห็นได้ว่ามีความคล้ายคลึงกับแผนภาพอินเทอร์แอคชันของภาษายูเอ็มแอล อย่างไรก็ตามจุดแตกต่างที่ชัดเจนระหว่างซีซีเอโปรไฟล์กับแผนภาพอินเทอร์แอคชันนั้น ได้แก่ การให้ความสำคัญกับส่วนเชื่อมต่อระหว่างคอมโพเนนท์มากกว่ารายละเอียดภายในคอมโพเนนท์

โดยซีซีเอโปรไฟล์จะเรียกส่วนการเชื่อมต่อดังกล่าวว่าพอร์ต นอกจากนั้น ซีซีเอโปรไฟล์ยังรองรับการอธิบายพฤติกรรมที่ซับซ้อนเกี่ยวกับการติดต่อเช่นการสื่อสารตอบโต้ระหว่างกันได้ โดยสามารถกำหนดเป็นโพรโทคอลของการสื่อสารระหว่างคอมโพเนนท์ได้ ข้อแตกต่างอีกประการคือความสามารถในการแสดงส่วนประกอบของคอมโพเนนท์โดยการลงรายละเอียดของส่วนประกอบภายในคอมโพเนนท์เป็นคอมโพเนนท์ย่อยภายใน โดยสามารถมีจำนวนลำดับชั้นของการแตกรายละเอียดของคอมโพเนนท์ย่อยได้ไม่จำกัดอีกด้วย

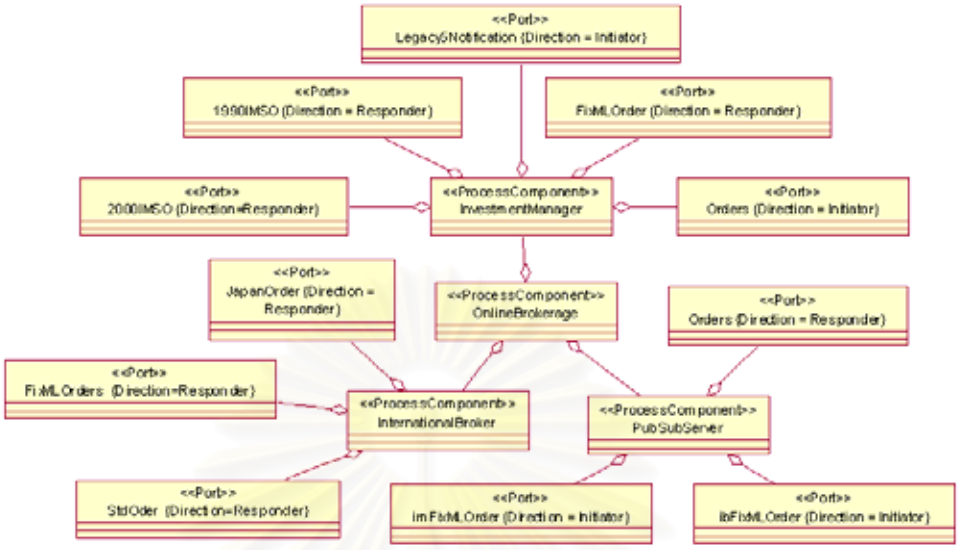


รูปที่ 2.15 ส่วนหนึ่งของเมตาโมเดลของซีซีเอ [10]

คุณสมบัติและความสามารถในการสื่อความหมายต่างๆ ของซีซีเอโปรไฟล์มิได้ถูกกำหนดขึ้นลอยๆ หากแต่มีการสร้างเป็นข้อกำหนดอย่างเป็นทางการ (Formal Specification) ในระดับเมตาโมเดล โดยเมตาโมเดลดังกล่าวถูกเขียนโดยใช้ภาษาเอ็มโอเอฟและสอดคล้องกับสถาปัตยกรรมเอ็มโอเอฟ ตัวอย่างเช่นส่วนหนึ่งของเมตาโมเดลของซีซีเอดังรูปที่ 2.15 ซึ่งมีคอนเซปต์ที่สำคัญในการอธิบายโครงสร้างของระบบในซีซีเอได้แก่ โปรเซสคอมโพเนนท์ (ProcessComponent) ซึ่งเป็นชนิดหนึ่งของคอมโพซิชัน (Composition) ซึ่งสามารถมีคอมโพเนนท์ย่อยได้ (สังเกตจากคอมโพซิชันมีคุณสมบัติ ComponentUsage) นอกจากนี้โปรเซสคอมโพเนนท์ยังมีคุณสมบัติการเป็นเจ้าของพอร์ต (PortOwner) ซึ่งพอร์ต (Port) ในที่นี้คือข้อกำหนดส่วนเชื่อมต่อของระบบโดยจะมีการแบ่งชนิดของพอร์ตเป็น 4 ชนิด ส่วนการแสดงคุณสมบัติเชิงพฤติกรรมของการเชื่อมต่อระหว่างพอร์ตนั้นในภาษาซีซีเอใช้คอนเซปต์ของโพรโทคอล (Protocol) ซึ่งเป็นชนิดหนึ่งของ คอรีโอกราฟฟี (Choreography) ซึ่งเป็นสเตทแมชชีนชนิดหนึ่งและแสดงพฤติกรรมของระบบในลักษณะของแผนภาพสเตทชาร์ต



รูปที่ 2.16 แบบจำลองของระบบตัวอย่างแสดงด้วยซีซีเอโนเทชัน [11]



รูปที่ 2.17 บางส่วนของแบบจำลองของระบบตัวอย่างแสดงด้วยยูเอ็มแอลโนเทชัน

นอกจากมีการกำหนดเมตาโมเดลของซีซีเอแล้ว ในข้อกำหนดของซีซีเอโปรไฟล์ยังมีการกำหนดโนเทชันสำหรับการสร้างโมเดลของซีซีเอด้วย ซึ่งการกำหนดโนเทชันดังกล่าวช่วยให้การสร้างแบบจำลองสามารถทำได้ง่ายขึ้น และสามารถทำความเข้าใจได้ง่ายกว่าการใช้โนเทชันของภาษายูเอ็มแอลตามปกติ โดยแสดงการเปรียบเทียบแบบจำลองของระบบตัวอย่างระบบหนึ่งซึ่งแสดงด้วยซีซีเอโนเทชันดังรูปที่ 2.16 และแสดงด้วยแผนภาพคลาสดังรูปที่ 2.17 ซึ่งจะเห็นได้ว่าระบบที่แสดงด้วยซีซีเอโนเทชันมีความสามารถในการสื่อความหมายที่ดีกว่ามาก

2.1.6 การพัฒนาโปรแกรมสำหรับเครื่องคอมพิวเตอร์มือถือด้วยแพลตฟอร์มเจทูเอ็มอี

เครื่องคอมพิวเตอร์มือถือ (Handheld Computer) เป็นเครื่องคอมพิวเตอร์ขนาดเล็กกะทัดรัด สะดวกต่อการพกพา มีประสิทธิภาพที่ต่ำกว่าเครื่องคอมพิวเตอร์ส่วนบุคคล ทั้งทางด้านพลังการคำนวณและขนาดของหน่วยความจำ มีจุดประสงค์การใช้งานที่ต่างไปจากเครื่องคอมพิวเตอร์ส่วนบุคคลทั่วไป โดยหน้าที่หลักคือการทำงานเป็นออร์แกไนเซอร์ (Organizer) ช่วยจดบันทึก และจดจำนัดหมายต่างๆ นอกจากนั้นยังมีความสามารถพิเศษอื่นๆ ขึ้นอยู่ความสามารถของฮาร์ดแวร์ที่เพิ่มขึ้นจากปกติ และโปรแกรมที่ถูกติดตั้งภายในเครื่องนั้นเช่น การดูหนัง ฟังเพลง อัดเสียง ดิกชันนารี ถ่ายภาพ หรือเป็นโทรศัพท์มือถือในตัวเป็นต้น โดยเครื่องคอมพิวเตอร์มือถือในปัจจุบันจะมีเพียง 2 แพลตฟอร์มหลักได้แก่ เครื่องปาล์ม (Palm) ซึ่งใช้ระบบปฏิบัติการ ปาล์มโอเอส (Palm OS) และเครื่องพอคเกตพีซี (Pocket PC) ซึ่งใช้ระบบปฏิบัติการ ไมโครซอฟต์วินโดวส์

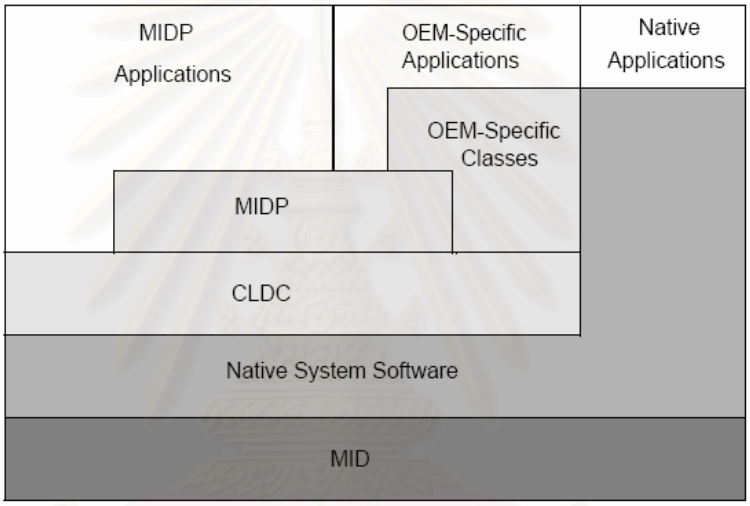
โมบายล์ (Microsoft Windows Mobile) ซึ่งชื่อเดิมคือ วินโดวส์ซีอี (Windows CE) โดยแสดงตัวอย่างเครื่องปาล์ม และพอคเกตพีซีในรูปที่ 2.18

รูปที่ 2.18 ตัวอย่างเครื่องปาล์ม (ซ้าย) และเครื่องพอคเกตพีซี (ขวา)

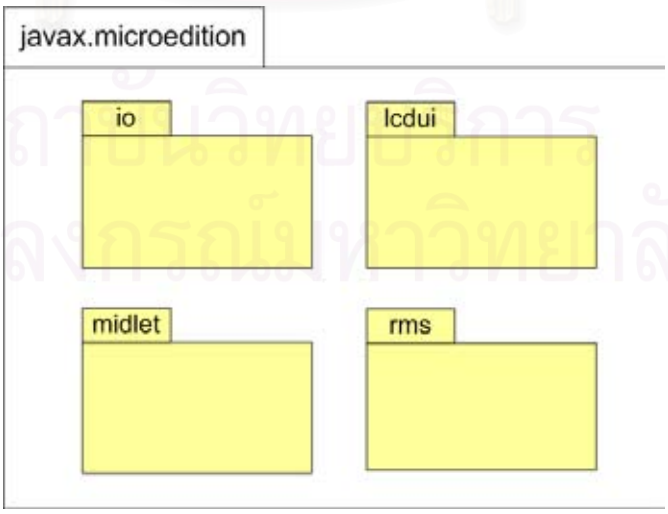
สำหรับการพัฒนาโปรแกรมบนเครื่องคอมพิวเตอร์มือถือนั้น ในงานวิจัยชิ้นนี้เลือกใช้แพลตฟอร์มปาล์มในการอิมพลีเมนต์ระบบฝั่งไคลเอนต์ และเลือกใช้แพลตฟอร์มเจทูเอ็มอี (J2ME: Java2 Mobile Edition) [12] ในการพัฒนาโปรแกรมโดยแสดงสถาปัตยกรรมของแพลตฟอร์มเจทูเอ็มอีดังรูปที่ 2.19 แนวคิดสำคัญของแพลตฟอร์มเจทูเอ็มอีนี้คือการกำหนดแบบจำลองของอุปกรณ์ฮาร์ดแวร์ของระบบฝังตัว (Embedded System) ให้เป็นอุปกรณ์เสมือนเรียกว่า ซีแอลดีซี (CLDC: Connected Limited Device Configuration) และซีดีซี (CDC: Connected Device Configuration) ซึ่งทำหน้าที่เป็นจาวาเวอร์ชวลแมชชีนด้วย นอกจากนี้อาจมีการกำหนดรายละเอียดที่เฉพาะเจาะจงเพิ่มเติมเช่น อุปกรณ์พกพาทั้งหลายเช่นโทรศัพท์มือถือหรือเครื่องคอมพิวเตอร์มือถือจะถูกโมเดลเป็นเอ็มไอดีพี (MIDP: Mobile Information Device Profile) ซึ่งในเชิงของการพัฒนาโปรแกรมแล้ว แพลตฟอร์มเจทูเอ็มอีได้เตรียมเอพีไอไลบรารีจำเป็นไว้จำนวนหนึ่งดังแสดงในรูปที่ 2.20 โดยไลบรารีดังกล่าวแบ่งเป็น 4 แพคเกจได้แก่

1. javax.microedition.io บรรจุคลาสและอินเทอร์เฟสที่จำเป็นเกี่ยวกับการติดต่อกับเครือข่ายไอพี (IP)

2. javax.microedition.midlet บรรจุคลาสและอินเทอร์เฟสที่จำเป็นเกี่ยวกับการควบคุมวงจรชีวิตของแอพพลิเคชัน รวมทั้งทำหน้าเป็นส่วนรับคำสั่งจากระบบปฏิบัติการของแอพพลิเคชัน
3. javax.microedition.lcdui บรรจุคลาสและอินเทอร์เฟสที่จำเป็นเกี่ยวกับการสร้างและควบคุมการแสดงผลบนหน้าจอของอุปกรณ์พกพา
4. javax.microedition.rms บรรจุคลาสและอินเทอร์เฟสที่จำเป็นเกี่ยวกับการจัดเก็บและการดึงข้อมูลจากหน่วยความจำถาวร (Persistent Data Storage)



รูปที่ 2.19 สถาปัตยกรรมของแพลตฟอร์มเจทูเอ็มอี



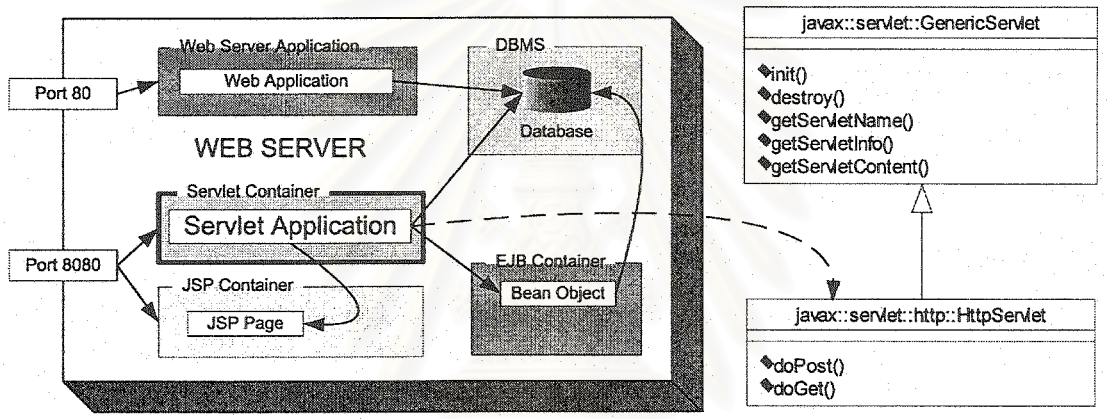
รูปที่ 2.20 ไลบรารีที่เจทูเอ็มอีเตรียมไว้สำหรับการพัฒนาโปรแกรมบนแพลตฟอร์มเจทูเอ็มอี

ตัวอย่างเอพีไอที่ใช้ในงานวิจัยชิ้นนี้ได้แก่

1. javax.microedition.midlet.MIDlet เป็นคลาสหลักซึ่งเจทูเอ็มอีแอพพลิเคชันทุกอันจำเป็นต้องมี หน้าที่หลักของคลาสนี้ได้แก่ การควบคุมวงจรชีวิต (Life Cycle) ของแอพพลิเคชัน คอยติดต่อรับเหตุการณ์และคำสั่งต่างๆ จากระบบปฏิบัติการ
2. javax.microedition.lcdui.Display เป็นคลาสที่จัดการเกี่ยวกับการแสดงผลบนหน้าจอ โดยสิ่งที่จะแสดงผลบนหน้าจอได้นั้นได้แก่คลาส Displayable เท่านั้น (ซึ่งรวมถึงคลาสลูกของ Displayable ด้วย) เมธอดที่สำคัญได้แก่ getDisplay() ซึ่งใช้ในการขอสิทธิการแสดงผลบนหน้าจอ (ซึ่งสิทธิการแสดงผลก็คืออินสแตนซ์ของคลาส Display นี้นั่นเองโดยหนึ่งแอพพลิเคชันจะมีเพียงหนึ่งสิทธิแสดงผลเท่านั้น) และในขณะหนึ่งๆนั้นจะมีคลาส Displayable เพียงคลาสเดียวซึ่งได้สิทธิดังกล่าว
3. javax.microedition.io.Connector เป็นคลาสที่จัดการเกี่ยวกับการสร้างการเชื่อมต่อ (Connection) กับอุปกรณ์อื่นๆ ในเครือข่าย
4. javax.microedition.io.HttpConnection เป็นอินเทอร์เฟสที่จัดการเกี่ยวกับการเชื่อมต่อกับเครื่องอื่นๆ ในเครือข่ายด้วยโพรโทคอล เอชทีทีพี (Http) โดยมีเมธอดหลักได้แก่ openInputStream() สำหรับเปิดช่องทางการสื่อสารสำหรับการส่งข้อมูลไปยังเครื่องอื่น และ openOutputStream() ใช้สำหรับเปิดช่องทางการสื่อสารเพื่อรับข้อมูลจากเครื่องอื่นๆ
5. javax.microedition.rms.RecordStore เป็นคลาสที่จัดการเกี่ยวกับการเก็บข้อมูลลงที่เก็บข้อมูลของอุปกรณ์ โดยจะเก็บในลักษณะเดียวกับการเก็บฐานข้อมูล 1 ตารางซึ่งประกอบด้วยเรคคอร์ด (Record) หลายๆ เรคคอร์ดซึ่งแต่ละเรคคอร์ดก็จะประกอบด้วยฟิลด์ (field) หลายๆ ฟิลด์ โดยในอุปกรณ์หนึ่งสามารถเก็บข้อมูลหลายๆ ตารางได้ โดยมีข้อแม้ว่าแต่ละตารางต้องมีชื่อไม่ซ้ำกัน
6. org.jdom.Document เป็นคลาสที่ใช้แสดง (Represent) เอกสารเอ็กซ์เอ็มแอลโดยยึดตามแบบจำลองของดอม (DOM: Document Object Model)
7. org.jdom.input.DOMBuilder เป็นคลาสที่ช่วยในการสร้างเอกสารเอ็กซ์เอ็มแอลจากแหล่งข้อมูลต้นทางเช่น ไฟล์ ยูอาร์แอล หรือสตรีมเป็นต้น
8. org.jdom.output.XMLOutputter เป็นคลาสที่ช่วยในการเขียนเอกสารเอ็กซ์เอ็มแอลไปยังแหล่งข้อมูลปลายทางซึ่งอาจเป็น ไฟล์ หรือสตรีม เป็นต้น

2.1.7 การพัฒนาโปรแกรมฝั่งเซิร์ฟเวอร์ด้วยแพลตฟอร์มเจทูอีอี/เซิร์ฟเล็ต เพื่อทำงานบนโปรแกรมอาปาเช่ทอมแคท

การพัฒนาเว็บเทคโนโลยีในปัจจุบัน ทำให้รูปแบบเครื่องเซิร์ฟเวอร์มีแนวโน้มจะเป็นแบบเว็บเซิร์ฟเวอร์มากขึ้น ซึ่งในการสร้างระบบที่ทำหน้าที่เป็นเว็บเซิร์ฟเวอร์นั้นสามารถทำได้หลายวิธี แต่โดยหลักการแล้วมีลักษณะที่คล้ายกัน คือสามารถทำการตอบสนองการร้องขอที่ติดต่อเข้ามาผ่านโพรโทคอลเอชทีทีพี (HTTP Protocol) ซึ่งสำหรับในแพลตฟอร์มจาวาแล้ว คอมโพเนนท์ที่ถูกออกแบบขึ้นเพื่อทำหน้าที่เป็นเว็บเซิร์ฟเวอร์ได้แก่ เจทูอีอี/เซิร์ฟเล็ต (J2EE/Servlet) ซึ่งมีคลาสหลักที่เกี่ยวข้องได้แก่คลาส javax::servlet::http::HttpServlet ซึ่งมีหลักการทำงานดังรูปที่ 2.21



รูปที่ 2.21 หลักการทำงานของเซิร์ฟเล็ต

รูปที่ 2.21 แสดงตัวอย่างการทำงานของเซิร์ฟเล็ต โดยเซิร์ฟเล็ตแอพพลิเคชันจำเป็นต้องทำงานอยู่ในสภาพแวดล้อมของเซิร์ฟเล็ตคอนเทนเนอร์ ซึ่งในวิทยานิพนธ์นี้เลือกใช้โปรแกรมอาปาเช่ทอมแคท (Apache Tomcat) [13] ทำหน้าที่เป็นเซิร์ฟเล็ตคอนเทนเนอร์โดยที่โปรแกรมจะใช้พอร์ตหมายเลข 8080 จึงไม่ซ้ำซ้อนกับพอร์ตปกติของเอชทีทีพี (พอร์ต 80) ซึ่งในการใช้งานจริงสามารถปรับแต่งหมายเลขพอร์ตดังกล่าวได้อย่างอิสระ

การเขียนเซิร์ฟเล็ตแอพพลิเคชันทำโดยการสืบทอด (Inherit) คลาส HttpServlet ซึ่งมีเมธอด ที่สำคัญ 2 เมธอดได้แก่ doGet() ซึ่งจะถูกเรียกเมื่อแอพพลิเคชันได้รับคำร้องขอเอชทีทีพีชนิดเก็ต (GET) และ doPost() ซึ่งจะถูกเรียกเมื่อแอพพลิเคชันได้รับคำร้องขอเอชทีทีพีชนิดโพส (POST) สำหรับการอิมพลีเมนต์เมธอดทั้ง 2 นั้นมีลักษณะเหมือนกับการเขียนโปรแกรมภาษาจาวาทั่วไป โดยสามารถนำเข้าไลบรารีและใช้งานคลาสเอพีไอต่างๆ ได้ตามปกติ สำหรับการส่งผลลัพธ์กลับไปยังไคลเอนต์ อาจทำโดยส่งต่อคำร้องขอ (redirect) ไปยังหน้าเจเอสพี (JSP Page)

เพื่อแสดงผลลัพธ์ที่เบราเซอร์ของไคลเอนต์ หรือใช้วิธีการเขียนข้อมูลในลักษณะสตรีมกลับไปยังเครื่องไคล์เอนนต์ก็ได้ โดยการเรียกเมธอด getOutputStream() หรือเมธอด getWriter() จากอินสแตนซ์ของคลาส HttpServletResponse (เป็นพารามิเตอร์ของเมธอ doGet() และ doPost()) ในทำนองเดียวกัน เซิร์ฟเล็ตสามารถอ่านข้อมูลที่ส่งมาจากไคลเอนต์ในรูปแบบของสตรีมโดยเรียกเมธอด getInpuStream() และ getReader() จากอินสแตนซ์ของคลาส HttpServletRequest ได้เช่นกัน

2.2 งานวิจัยที่เกียวข้อง

2.2.1 งานวิจัย “Transformation: The Missing Link of MDA” [14]

งานวิจัยนี้กล่าวถึงการแปลงระหว่างแบบจำลอง (Model-to-Model Transformation) ซึ่งงานวิจัยนี้กล่าวว่าเป็นสิ่งที่ขาดหายไปของเอ็มดีเอ (The Missing Link of MDA) โดยกล่าวถึงวิธีการแปลงทั่วไปที่มีอยู่ในปัจจุบัน และนำหลักการดังกล่าวมาใช้แปลงแบบจำลองอีด็อก-บีพี (EDOC-BP: Enterprise Distributed Object Computing – Business Process Model) ไปเป็นแบบจำลองการไหลของงานของบรีส (Breeze Workflow Model) โดยหลังจากทำการทดลองการแปลงด้วยวิธีต่างๆ แล้วจึงได้สรุปเสนอเป็นรายการความต้องการของภาษาการแปลงของเอ็มดีเอ (MDA Mapping Language Requirement)

ในงานวิจัยที่จะพัฒนา จะได้นำรายการความต้องการของภาษาการแปลงเอ็มดีเอซึ่งงานวิจัยนี้ได้ให้ไว้ มาใช้เป็นแนวทางในการสร้างกฎการแปลงระหว่างพีไอเอ็มกับพีเอสเอ็ม

2.2.2 งานวิจัย “Agile Model Driven Development Is Good Enough” [15]

งานวิจัยนี้กล่าวถึงจุดด้อยของเอ็มดีเอในประเด็นต่างๆเช่น การสร้างแบบจำลองของระบบเป็นงานที่ซับซ้อนเกินไปสำหรับนักออกแบบระบบทั่วไปเนื่องจาก ต้องมีความเข้าใจในภาษาที่ใช้ในการสร้างแบบจำลองสูง โดยเอ็มดีเอคาดหวังให้นักออกแบบมีความรู้และทักษะที่ดีเกียวกับการสร้างแบบจำลองซึ่งในความเป็นจริงมักไม่เป็นเช่นนั้น นอกจากนั้นระบบงานจริงนั้นมักมีส่วนที่ไม่สามารถอธิบายได้ด้วยภาษาโมเดลในปัจจุบัน อีกทั้งขาดความพร้อมของเครื่องมือพัฒนาระบบด้วยแนวคิดของเอ็มดีเอ ทำให้การนำเอ็มดีเอมาใช้จริงนั้นเป็นเรืองยากและยังต้องการความพยายามในการวิจัยและพัฒนาอีกมาก

2.2.3 งานวิจัย “MDA-based Development of E-Learning System” [16]

งานวิจัยนี้แสดงตัวอย่างการพัฒนาระบบสารสนเทศสำหรับระบบงานอีเลิร์นนิง โดยใช้แนวคิดเอ็มดีเอ โดยเริ่มจากการสร้างแบบจำลองพีไอเอ็มโดยการอ้างอิงแบบจำลองของแอลทีเอสเอ (LTSA: Learning Technology System Architecture) ในการกำหนดองค์ประกอบของระบบ จากนั้นจึงสร้างแผนภาพคลาสเพื่อแสดงแบบจำลองเชิงโครงสร้างของระบบ ส่วนในระดับพีเอสเอ็มนั้นงานวิจัยนี้เลือกแพลตฟอร์มปลายทางเป็นแพลตฟอร์มอีเจบี และใช้ยูเอ็มแลโปรไฟล์สำหรับอีเจบี เป็นภาษาที่ใช้แสดงแบบจำลองพีเอสเอ็ม โดยแสดงในรูปแผนภาพคลาสเช่นกัน

2.2.4 งานวิจัย “Design of Rules for Transforming UML Sequence Diagrams into Java Code” [17]

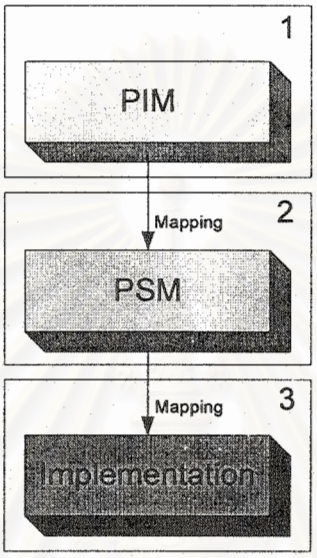
งานวิจัยนี้กล่าวถึงการแปลงแผนภาพซีเควนซ์ (Sequence Diagram) และแผนภาพคลาส (Class Diagram) ไปเป็นโปรแกรมภาษาจาวา ด้วยการกำหนดกฎการแปลงสำหรับแต่ละรูปแบบที่เป็นไปได้ในแผนภาพซีเควนซ์ โดยมีการเชื่อมโยงข้อมูลของแผนภาพทั้งสองในระดับเมตาโมเดล (Metamodel)

ในงานวิจัยที่จะพัฒนาขึ้นจะได้นำกฎการแปลงเหล่านี้มาใช้เป็นแนวทางในการเติมส่วนของโปรแกรมเพื่อให้เป็นโปรแกรมที่สามารถทำงานจริงได้ จากพีเอสเอ็มในส่วนที่แสดงพฤติกรรมของระบบ (System Behavior)

บทที่ 3
วิธีดำเนินการวิจัย

3.1 การแบ่งส่วนงานวิจัย

จากการศึกษาขั้นต้นเกี่ยวกับแนวคิดเอ็มดีเอ พบว่ากระบวนการพัฒนาระบบด้วยแนวคิดเอ็มดีเอประกอบไปด้วยขั้นตอนหลักๆ 3 ขั้นตอนได้แก่ (ในรูปที่ 3.1)



รูปที่ 3.1 ขั้นตอนการพัฒนาระบบ

1. การสร้างแบบจำลองพีไอเอ็ม
2. การสร้างแบบจำลองพีเอสเอ็มจากแบบจำลองพีไอเอ็ม
3. การสร้างโค้ดจากแบบจำลองพีเอสเอ็ม

ดังนั้น ในงานวิจัยชิ้นนี้ จะมุ่งประเด็นศึกษาถึงรายละเอียดในแต่ละขั้นตอน โดยการศึกษาถึงนิยาม (Definition) และข้อกำหนด (Specification) ต่างๆ ที่เกี่ยวข้อง เพื่อให้เกิดภาพที่ชัดเจนยิ่งขึ้นในแต่ละขั้นตอน โดยจะกล่าวรายละเอียดแต่ละขั้นตอนในหัวข้อถัดไป รวมทั้งแสดงตัวอย่างการพัฒนาระบบตัวอย่างในบทที่ 4

3.2 การสร้างแบบจำลองพีไอเอ็ม

3.2.1 คำจำกัดความของแบบจำลองพีไอเอ็ม

โอเอ็มจี ในฐานะองค์กรที่เป็นผู้ออกข้อกำหนดมาตรฐานต่างๆ ได้ให้คำจำกัดความของแบบจำลองพีไอเอ็มไว้ว่า "เป็นแบบจำลองที่แสดงเฉพาะหน้าที่และพฤติกรรมในเชิงธุรกิจ

(Business Functionality and Behavior)" เท่านั้น [3] โดยในพีไอเอ็มเองอาจจะประกอบด้วยหลายๆ แบบจำลองซึ่งเกิดจากหลายมุมมองรวมทั้งสามารถมีการบอกรายละเอียดในหลายระดับได้ ซึ่งโอเอ็มจีได้ระบุในเอกสาร [3] ว่าแบบจำลองพีไอเอ็มนั้นสามารถถูกแบ่งได้อีกเป็น 2 ระดับ ได้แก่ พีไอเอ็มฐาน(Base PIM) และพีไอเอ็มระดับล่าง (Next Level PIM) ซึ่งได้กล่าวรายละเอียดไว้บางส่วนในหัวข้อ 2.1.3

แบบจำลองพีไอเอ็มฐานจะมีระดับความเป็นนามธรรมที่สูงกว่าแบบจำลองพีไอเอ็มระดับล่างซึ่งจะมีระดับความเป็นนามธรรมที่สูงกว่าแบบจำลองพีเอสเอ็ม จุดนี้เองอาจเป็นที่สงสัยถึงข้อแตกต่างระหว่างแบบจำลองพีไอเอ็มระดับล่างกับแบบจำลองพีเอสเอ็มว่าแตกต่างกันอย่างไร ซึ่งคำตอบน่าจะเป็นการตอบคำถามว่ารายละเอียดเชิงเทคนิคในแบบจำลองนั้น ขึ้นกับแพลตฟอร์มหรือไม่ ยกตัวอย่างรายละเอียดเชิงเทคนิคที่ไม่ขึ้นกับแพลตฟอร์มเช่น การเรียกใช้บริการ (Service) ต่างๆ ของระบบ ซึ่งบริการเหล่านี้ในแง่ของแบบจำลอง สามารถระบุเป็นเพียงข้อกำหนดโดยไม่ต้องลงรายละเอียดในขั้นตอนของการปฏิบัติก็ได้ ดังนั้นการสร้างแบบจำลองระดับล่าง จึงเป็นการเพิ่มรายละเอียดทางเทคนิคบางส่วนให้กับแบบจำลองพีไอเอ็มฐาน โดยอาศัยกลไกของการเรียกใช้บริการ ซึ่งแต่ละบริการจะมีการกำหนดแบบจำลองการทำงานของบริการนั้นๆ ไว้เป็นมาตรฐานอยู่แล้ว ดังนั้นหากกล่าวในบริบทของเอ็มดีเอ ขั้นตอนนี้ก็คือการนำแบบจำลองพีไอเอ็มฐานไปเชื่อมต่อกับแบบจำลองมาตรฐานของบริการต่างๆ เพื่อนำไปสร้างเป็นโค้ดในขั้นตอนต่อไป

ลักษณะการยึดแนวความคิดจากเทคโนโลยีมิดเดิลแวร์ (โดยเฉพาะอย่างยิ่ง คอร์บา) จะพบได้อย่างสม่ำเสมอในเอกสารต่างๆ ที่อธิบายถึงแนวคิดเอ็มดีเอ ทั้งนี้เนื่องจากวัตถุประสงค์ของแนวคิดเอ็มดีเอจะคล้ายกับของเทคโนโลยีมิดเดิลแวร์กล่าวคือจะเน้น ความสามารถในการทำงานร่วมกันได้ (Interoperability) และความสามารถในการบูรณาการ (Integration) ของระบบ โดยเอ็มดีเอมุ่งประเด็นไปที่ระดับที่สูงกว่ามิดเดิลแวร์ ดังที่ได้กล่าวรายละเอียดไปแล้วในหัวข้อ 2.1.3

เนื่องจากนิยามของพีไอเอ็มที่โอเอ็มจีกำหนดมานั้นเป็นการให้ความหมายกว้างๆ ในเชิงหลักการเท่านั้น งานวิจัยชิ้นนี้ได้พยายามศึกษาถึงการหาคำจำกัดความของแบบจำลองพีไอเอ็ม ที่สามารถช่วยให้เห็นภาพได้ชัดเจนยิ่งขึ้น แต่จากการศึกษาพบว่า จนถึงปัจจุบันนี้ ยังไม่มีข้อกำหนดมาตรฐานใดๆ ที่ระบุถึงลักษณะที่ชัดเจนของแบบจำลองพีไอเอ็มได้ อย่างไรก็ตามมิได้หมายความว่าเราจะไม่สามารถสร้างแบบจำลองพีไอเอ็มได้ เพราะจากการศึกษางานวิจัยหลายชิ้นที่เกี่ยวข้องทำให้สามารถสรุปลักษณะที่แบบจำลองพีไอเอ็มควรจะมีซึ่งจะกล่าวถึงในหัวข้อถัดไป

3.2.2 ลักษณะที่ควรจะเป็นของแบบจำลองพีไอเอ็ม

หากเปรียบเทียบเอ็มดีเอกับคอร์บาซึ่งมีการกำหนดส่วนต่อประสาน (Interface) ของทุกคอมโพเนนท์ด้วยภาษาไอดีแอล (IDL: Interface Definition Language) ซึ่งช่วยทำให้เกิดความสามารถในการทำงานร่วมกันระหว่างคนละภาษาโปรแกรมแล้ว วิธีการที่เอ็มดีเอใช้ในการทำให้เกิดความสามารถในการทำงานร่วมกันถึงแม้จะเป็นมิดเดิลแวร์คนละชนิด ก็คือการกำหนดให้ระบบทุกระบบที่จะทำงานร่วมกันต้องถูกอธิบายด้วยแบบจำลอง (หมายรวมทั้งพีไอเอ็มและพีเอสเอ็ม) โดยการกำหนดการเชื่อมต่อของระบบต่างๆ ก็จะทำในระดับแบบจำลอง (ไม่ใช่ในระดับของการเขียนโค้ดเหมือนเช่นในคอร์บา) จากนั้นจึงนำแบบจำลองดังกล่าวไปสร้างโค้ดด้วยเครื่องมืออัตโนมัติ ซึ่งโปรแกรมที่ได้จะรวมส่วนของโค้ดที่ทำให้ระบบทำงานร่วมกันได้ไว้แล้ว

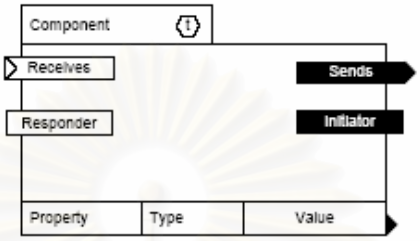
อย่างไรก็ตาม ปัญหาที่ตามมาคือ ในการทำงานร่วมกันนั้นแบบจำลองของระบบหนึ่งจะเข้าใจข้อกำหนดต่างๆ ที่กำหนดไว้ในแบบจำลองของอีกระบบได้อย่างไร โอเอ็มจีได้ให้คำตอบในเรื่องนี้ โดยการกำหนดมาตรฐานที่เกี่ยวข้องกับการสร้างแบบจำลองขึ้นมา ได้แก่ ยูเอ็มแอล เอ็มโอเอฟ และซีดับเบิลยูเอ็ม ซึ่งทั้งสามมาตรฐานนี้เองเป็นกลไกสำคัญสำหรับสร้างแบบจำลองของระบบที่มีความสามารถในการทำงานร่วมกันได้ จากการศึกษาพบว่าแบบจำลองของระบบ (หมายรวมทั้งพีไอเอ็มและพีเอสเอ็ม) ควรมีลักษณะที่สอดคล้องกับข้อกำหนด 2 ข้อ เพื่อใช้รองรับการทำงานร่วมกันระหว่างระบบที่มีแบบจำลองคนละแบบจำลองดังนี้

1. แบบจำลองนั้น ในที่สุดแล้วต้องสามารถแสดงออกในรูปของภาษายูเอ็มแอลได้
2. แบบจำลองนั้นต้องมีการกำหนดเมตาโมเดลซึ่งสอดคล้องกับข้อกำหนดของสถาปัตยกรรมเอ็มโอเอฟ

3.2.2.1 แบบจำลองนั้นต้องสามารถแสดงออกในรูปของภาษายูเอ็มแอลได้

ในขั้นตอนของการสร้างแบบจำลองนั้น ไม่มีข้อกำหนดว่าจะต้องใช้ภาษาใดในการอธิบายหรือแสดงแบบจำลอง ผู้สร้างแบบจำลองสามารถเลือกภาษาใดก็ได้ แต่ถ้าไม่ใช่ภาษายูเอ็มแอลต้องสร้างชุดของกฎการแปลงภาษานั้นๆ ให้เป็นภาษายูเอ็มแอลไว้ด้วย เนื่องจากภาษายูเอ็มแอลเป็นภาษาโมเดลมาตรฐานสำหรับเอ็มดีเอ ดังนั้นทุกเครื่องมือที่สร้างขึ้นรองรับมาตรฐานเอ็มดีเอจำเป็นต้องสามารถเข้าใจภาษายูเอ็มแอลเป็นพื้นฐาน (อาจเข้าใจภาษาอื่นด้วยก็ได้) การนำเข้าแบบจำลอง (Import) สำหรับเครื่องมือดังกล่าวจึงจำเป็นต้องแสดงแบบจำลองในรูปของภาษายูเอ็มแอลเสียก่อน ยกตัวอย่างเช่นโอเอ็มจีได้กำหนดภาษาเฉพาะสำหรับการสร้างแบบจำลองของระบบที่เป็นระบบประมวลผลแบบกระจายเชิงวัตถุหรืออีด็อก โดยการทำงานร่วมกันระหว่างส่วน

ต่างๆ ของระบบสามารถแสดงด้วยซีซีเอ ซึ่งมีการกำหนด เมตาโมเดล ดังแสดงในรูปที่ 2.14 และกำหนดโนเทชันพิเศษขึ้น ดังแสดงในรูปที่ 3.2 นอกจากนี้ยังมีการสร้างกฎการแปลงขึ้นเพื่อให้สามารถแสดงแบบจำลองซึ่งอยู่ในรูปโนเทชันพิเศษนี้ด้วยภาษายูเอ็มแอลได้ โดยส่วนหนึ่งของกฎการแปลงอยู่ในรูปที่ 3.3



รูปที่ 3.2 ตัวอย่างของซีซีเอโนเทชัน [10]

Metamodel element name	Stereotype name	UML base Class	Parent	Tags	Constraints
ProcessComponent	ProcessComponent	Classifier	N/A	granularity isPersistent primitiveKind primitiveSpec	
Port	Port	Class	N/A	isSynchronous isTransactional direction postCondition	
FlowPort	FlowPort	Class	Port	typeProperty	
ProtocolPort	ProtocolPort	Class	Port	uses	
MultiPort	MultiPort	Class	Port		
OperationPort	N/A	Operation	Port		
Protocol	Protocol	Class	N/A		
Interface	N/A	Classifier	N/A		
InitiatingRole	InitiatingRole	Class	N/A		
RespondingRole	InitiatingRole	Class	N/A		
PropertyDefinition	PropertyDefinition	Attribute	N/A		
«enumeration» DirectionKind	DirectionKind	Enumeration			
«enumeration» GranularityKind	GranularityKind	Enumeration	N/A		
Direction (value)	initiates	Association	N/A		
Direction (value)	responds	Association	N/A		

รูปที่ 3.3 ตัวอย่างกฎการแปลงภาษาซีซีเอเป็นภาษายูเอ็มแอล [10]

การใช้ภาษาโมเดลอื่นที่ไม่ใช่ยูเอ็มแอลในการออกแบบระบบนั้นเป็นเรื่องปกติที่เกิดขึ้นได้ ในบางกรณีผู้ออกแบบอาจเห็นว่าภาษาอื่นมีความเหมาะสมมากกว่า ใช้อธิบายแบบจำลองของงานนั้นๆ ได้ง่ายชัดเจนและสะดวกกว่า การเลือกภาษาโมเดลที่ดีที่สุดสำหรับงานหนึ่งๆ คือเลือกภาษาที่สามารถแสดงรายละเอียดของระบบได้ง่าย และถูกพัฒนาได้อย่างไม่ซับซ้อน ยกตัวอย่าง

เช่น ในการออกแบบอาคาร หรือระบบโรงงานอุตสาหกรรม ผู้ออกแบบย่อมเลือกที่จะใช้ภาษาที่มีแผนภาพที่เหมาะสมและเฉพาะเจาะจงกับงานวิเคราะห์และออกแบบระบบนั้นๆ ซึ่งแต่ละงานย่อมใช้ภาษาโมเดลที่แตกต่างกัน ดังนั้นการที่จะใช้ภาษาโมเดลเดียวที่สามารถสร้างแบบจำลองได้ทุกประเภทจึงเป็นเรื่องที่ปฏิบัติจริงได้ยาก

3.2.2.2 แบบจำลองนั้นต้องมีการกำหนดเมตาโมเดลซึ่งสอดคล้องกับข้อกำหนดของสถาปัตยกรรมเอ็มโอเอฟ

จากที่ได้กล่าวมาแล้วในหัวข้อที่ผ่านมา เอ็มดีเออนุญาตให้ใช้ภาษาโมเดลใดๆ ก็ได้ในการสร้างแบบจำลอง อย่างไรก็ตามภาษาที่ใช้นั้นจำเป็นต้องเป็นภาษาที่มีการกำหนดโครงสร้างโมเดล (Model Construct) อย่างชัดเจนในระดับเมตาโมเดล กล่าวคือ ภาษาโมเดลดังกล่าวต้องมีการกำหนดเมตาโมเดลของภาษาขึ้นด้วย นอกจากนั้นเมตาโมเดลดังกล่าวยังต้องสอดคล้องกับสถาปัตยกรรมเอ็มโอเอฟซึ่งได้กล่าวไว้ในหัวข้อที่ 2.1.4 การกำหนดให้การสร้างเมตาโมเดลของทุกภาษาโมเดล ต้องมีมาตรฐานเดียวกัน (ตามมาตรฐานเอ็มโอเอฟ) เป็นการเพิ่มความเป็นไปได้ที่เครื่องมือออกแบบจะสามารถเข้าใจภาษาโมเดลใหม่ๆ ได้อย่างอัตโนมัติ และเข้าใจถึงความหมาย (Semantics) ของภาษาโมเดลนั้นได้ ซึ่งทำให้สถาปัตยกรรมเอ็มดีเอรองรับภาษาโมเดลใหม่ๆ ในอนาคตได้

3.2.3 เทคนิคการคิดย้อนกลับ (Reverse Projection Technique)

จากที่ได้กล่าวไปแล้วว่า แบบจำลองที่ได้ควรเป็นแบบจำลองที่อธิบายถึงหน้าที และพฤติกรรมของระบบ แต่ในขณะเดียวกันแบบจำลองดังกล่าวก็ควรมีความยืดหยุ่นเพียงพอ และไม่ยึดติดกับรายละเอียดทางเทคนิคในระดับอิมพลีเมนต์ อย่างไรก็ตามพบว่าในทางปฏิบัติแล้วค่อนข้างทำได้ยาก เนื่องจากปกติแล้วนักออกแบบระบบมักจะคิดถึงลักษณะของระบบที่เสร็จสิ้นไว้อยู่แล้วว่าจะออกมาหน้าตาเป็นอย่างไร เช่นใช้ภาษาโปรแกรมอะไร หรือทำงานบนระบบปฏิบัติการใด ซึ่งบ่อยครั้งลักษณะของระบบที่เสร็จสิ้นดังกล่าวมีผลต่อการสร้างแบบจำลองทำให้แบบจำลองที่ได้มีลักษณะเฉพาะเจาะจงเกินไป ยกตัวอย่างเช่น เมื่อนักออกแบบมีความคิดที่จะให้ระบบเมื่อเสร็จสมบูรณ์ถูกเขียนด้วยภาษาจาวาทำงานบนวินโดวส์ ก็อาจจะสร้างแบบจำลองของระบบโดยใช้คลาสของภาษาจาวา และมีการกำหนดการใช้เอพีไอต่างๆ ของวินโดวส์ไว้ในแบบจำลองเลย ซึ่งถึงแม้ระบบที่ได้จะทำงานได้อย่างถูกต้องและตรงตามความต้องการ แต่ระบบก็จะถูกผูกติดกับแพลตฟอร์มจาวาบนวินโดวส์เสมอเนื่องจากถูกออกแบบให้เป็นเช่นนั้นตั้งแต่ต้น

เพื่อแก้ปัญหาดังกล่าวในงานวิจัยชิ้นนี้ได้นำเสนอแนวคิดในการสร้างแบบจำลองเพื่อให้ได้แบบจำลองที่มีลักษณะทั่วไปไม่เฉพาะเจาะจง แต่สามารถอธิบายถึงหน้าที และพฤติกรรมของระบบได้ เรียกว่าเทคนิคการคิดย้อนกลับ (Reverse Projection Technique) กล่าวคือการคิดถึงความเป็นไปได้ในการอิมพลีเมนต์ระบบด้วยวิธีอื่นๆ แต่ยังคงมีตรรกะทางธุรกิจที่เหมือนกัน

เพื่อให้เข้าใจในแนวคิดนี้มากขึ้นจะขอยกตัวอย่างเชิงรูปธรรมประกอบเช่น ในการสร้างพาหนะทางบกลำหนึ่ง ถ้าผู้ออกแบบยึดติดว่าพาหนะดังกล่าวต้องเป็นรถยนต์ ก็อาจจะทำให้การพิจารณาถึงความต้องการเชิงหน้าที พฤติกรรม ยึดติดกับความเป็นรถยนต์ไปด้วยได้แก่

- ต้องสามารถให้คนโดยสารได้
- ต้องบรรทุกของได้
- ต้องมีระบบระบายความร้อน
- ต้องมีเครื่องยนต์ขับเคลื่อน
- ต้องบังคับทิศทางได้
- ต้องมีล้อ 4 ล้อ
- ต้องเร่งความเร็วได้

แต่หากคิดย้อนกลับถึงข้อกำหนดเริมต้นซึ่งระบุเพียงพาหนะทางบกเท่านั้น แล้วคิดถึงความเป็นไปได้อื่นๆ เช่น เกวียน รถไฟ หรือ เลื่อนหิมะ มาพิจารณาด้วย ก็จะทำให้ข้อกำหนดดังกล่าวกว้างขึ้นเหลือเพียง ต้องสามารถให้คนโดยสารได้ ส่วนประเด็นในการมีเครื่องยนต์ก็ต้องขยายความให้กว้างขึ้นเป็น ต้องมีแหล่งพลังงานในการขับเคลื่อน เป็นต้น

การที่ข้อกำหนดกว้างขึ้น และมีลักษณะที่เฉพาะเจาะจงน้อยลง หมายถึงมีทางเลือกในการอิมพลีเมนต์มากขึ้น ซึ่งตรงกับแนวคิดของแบบจำลองพีไอเอ็มซึ่งต้องการให้คงเหลือเฉพาะข้อกำหนดเชิงหน้าที และพฤติกรรมของระบบทีจำเป็นจริงๆ เท่านั้น โดยไม่มีรายละเอียดของการอิมพลีเมนต์อยู่เลย

3.3 การสร้างแบบจำลองพีเอสเอ็มจากแบบจำลองพีไอเอ็ม

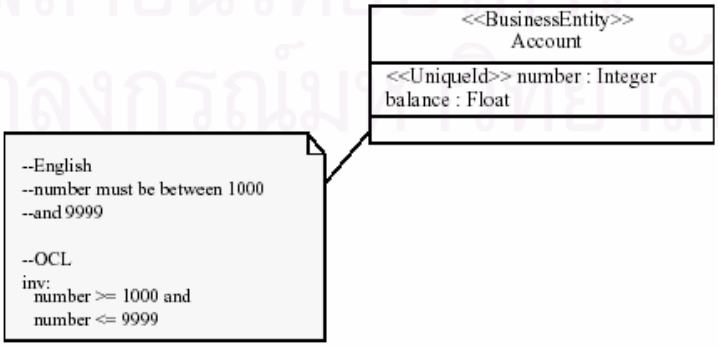
3.3.1 คำจำกัดความของแบบจำลองพีเอสเอ็ม

โอเอ็มจีได้ให้คำจำกัดความของพีเอสเอ็มในความหมายกว้างๆ ว่าพีเอสเอ็มหมายถึงแบบจำลองที่ยึดติดกับแพลตฟอร์ม โดยแพลตฟอร์มในที่นี้หมายถึงรายละเอียดเชิงเทคโนโลยี และ

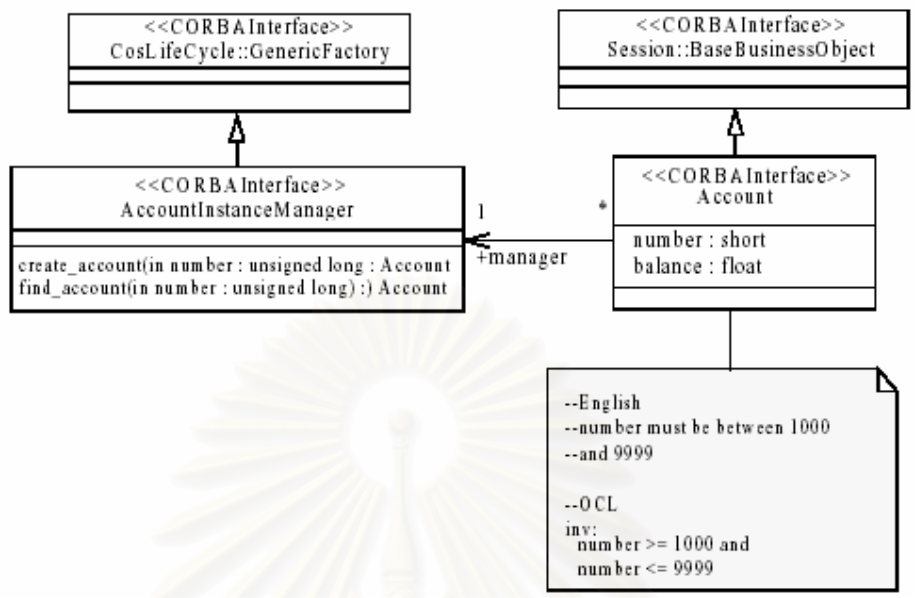
เชิงวิศวกรรมของการอิมพลีเมนต์ กล่าวคือแบบจำลองพีเอสเอ็มเป็นแบบจำลองของระบบที่สามารถให้รายละเอียดของการอิมพลีเมนต์ได้ โดยจุดประสงค์ที่แบ่งแบบจำลองออกเป็นสองระดับนั้น เพื่อแบ่งแยกรายละเอียดในการอิมพลีเมนต์ออกจากข้อกำหนดเชิงหน้าที่ และพฤติกรรมของระบบ ทำให้สามารถเลือกรูปแบบการอิมพลีเมนต์ในภายหลังได้โดยไม่ต้องสร้างแบบจำลองของระบบใหม่

โอเอ็มจีได้ให้แนวทางในการแสดงแบบจำลองพีเอสเอ็มด้วยภาษายูเอ็มแอล ซึ่งปกติเป็นภาษาที่ใช้ในการแสดงความหมายที่ไม่เฉพาะเจาะจง การที่จะนำมาใช้แสดงแบบจำลองพีเอสเอ็มนั้นทำได้โดยใช้ความสามารถของยูเอ็มแอลที่สามารถขยายต่อได้ (Extensibility) โดยแสดงในรูปแบบของยูเอ็มแอลโปรไฟล์ที่ได้อธิบายไว้ในหัวข้อ 2.1.5 ซึ่งเป็นการต่อขยายภาษายูเอ็มแอลทำให้ภาษาสามารถแสดงความหมายที่เฉพาะเจาะจงได้มากขึ้น

ยกตัวอย่างดังรูปที่ 3.4 และ 3.5 ซึ่งเป็นการเปรียบเทียบแบบจำลองพีไอเอ็มกับแบบจำลองพีเอสเอ็มของระบบธนาคารที่แสดงเอนทิตีของบัญชีลูกค้า (Account) ซึ่งมีหมายเลขลูกค้า (Number) และยอดเงินฝาก (Balance) โดยมีข้อกำหนดเชิงพฤติกรรมเขียนด้วยภาษาโอซีแอล (OCL: Object Constraint Language) ซึ่งจากแบบจำลองในรูปที่ 3.4 นี้จะเห็นว่าไม่มีรายละเอียดในการอิมพลีเมนต์เอนทิตีบัญชีลูกค้านี้อยู่เลย มีแต่ข้อมูลของข้อกำหนดเชิงธุรกิจเท่านั้น โดยข้อมูลในการอิมพลีเมนต์นั้นจะถูกเพิ่มเข้าไปในแบบจำลองพีเอสเอ็มในรูปที่ 3.5 โดยระบุว่าการอิมพลีเมนต์เป็นแพลตฟอร์มคอร์บา (สังเกตจาก การที่แบบจำลองมีโครงสร้างที่ใช้องค์ประกอบของแพลตฟอร์มคอร์บา) โดยเอนทิตีบัญชีลูกค้าในตอนนี้จะถูกอิมพลีเมนต์เป็นคอร์บาอินเทอร์เฟสชื่อว่า Account ซึ่งเป็นชนิดหนึ่งของ (Inheritance) BaseBusinessEntity โดยทุก Account จะถูกจัดการโดยผู้จัดการบัญชีหนึ่งตัวชื่อ AccountManager ทำหน้าที่ในการสร้าง Account ใหม่และหาอินสแตนซ์ของ Account ใดๆ โดยอาศัย number เป็นตัวสืบค้น



รูปที่ 3.4 ตัวอย่างแบบจำลองพีไอเอ็ม [3]

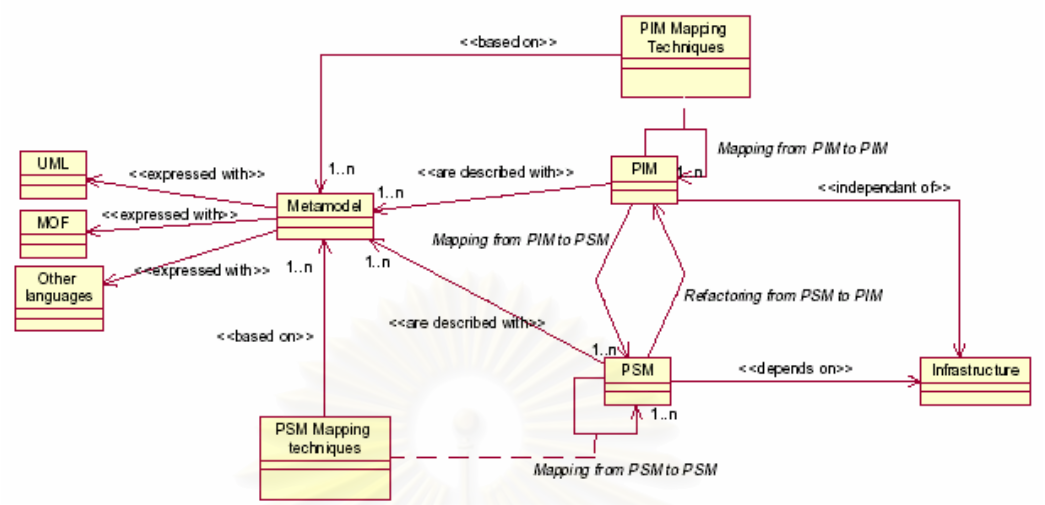


รูปที่ 3.5 ตัวอย่างแบบจำลองพีเอสเอ็ม [3]

ในงานวิจัยชิ้นนี้ใช้การแสดงแบบจำลองพีเอสเอ็มด้วยภาษายูเอ็มแอลเช่นกัน โดยแพลตฟอร์มที่ใช้อิมพลีเมนต์ระบบนั้นมี 2 แพลตฟอร์มหลักได้แก่ แพลตฟอร์มเจทูเอ็มอี (J2ME) ซึ่งทำหน้าที่เป็นฝั่งไคลเอนต์ทำงานบนเครื่องปาล์ม และ เจทูอีอีเซิร์ฟเล็ต (J2EE/Servlet) ซึ่งทำหน้าที่เป็นฝั่งเซิร์ฟเวอร์ทำงานบนเครื่องพีซี แต่เนื่องจากในขณะนี้ยังไม่มีมาตรฐานยูเอ็มแอลโปรไฟล์สำหรับแพลตฟอร์มทั้งสองออกมา ในงานวิจัยชิ้นนี้จึงจะนำเสนอโนเทชันซึ่งกำหนดขึ้นโดยอาศัยความสามารถในการต่อขยายได้ของภาษายูเอ็มแอล ในการทำให้ภาษายูเอ็มแอลสามารถสื่อความหมายที่เฉพาะเจาะจงของแพลตฟอร์มทั้งสองได้

3.3.2 การแปลงแบบจำลองพีไอเอ็มไปเป็นแบบจำลองพีเอสเอ็ม

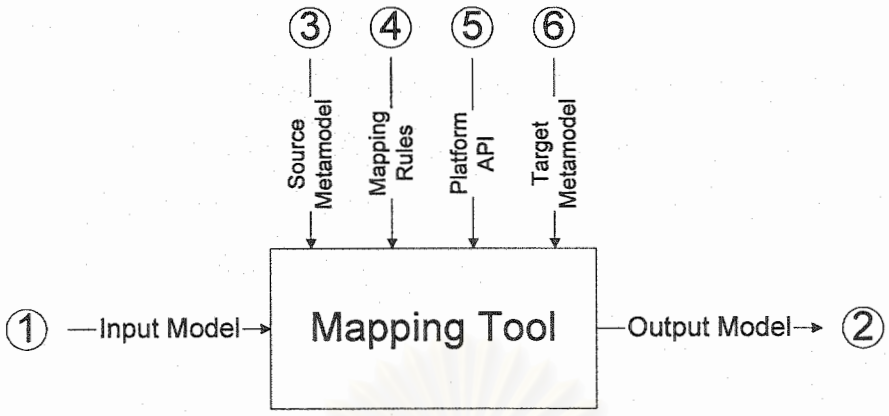
โดยหลักการแล้วแบบจำลองพีเอสเอ็มเกิดจากแบบจำลองพีไอเอ็มที่เพิ่มรายละเอียดในการอิมพลีเมนต์เข้าไป ซึ่งในบริบทของเอ็มดีเอจะเรียกว่าการแมปปิง (Mapping) โดยมีความสัมพันธ์ระหว่างพีไอเอ็มและพีเอสเอ็มดังรูปที่ 3.6 สาเหตุที่มองเป็นการแมปปิงนั้นเนื่องจากเอ็มดีเอมองว่าในพีไอเอ็มมีเอนทิตีที่เป็นข้อกำหนดเชิงหน้าที่ของระบบอยู่แล้ว ส่วนพีเอสเอ็มก็มีเอนทิตีพื้นฐานสำหรับการอิมพลีเมนต์ระบบบนแพลตฟอร์มนั้นๆ ดังนั้นการแปลงพีไอเอ็มเป็นพีเอสเอ็มจึงเหมือนเป็นการเลือกว่า เอนทิตีแต่ละตัวในพีไอเอ็มนั้นจะถูกอิมพลีเมนต์ด้วยเอนทิตีใดในพีเอสเอ็มนั่นเอง



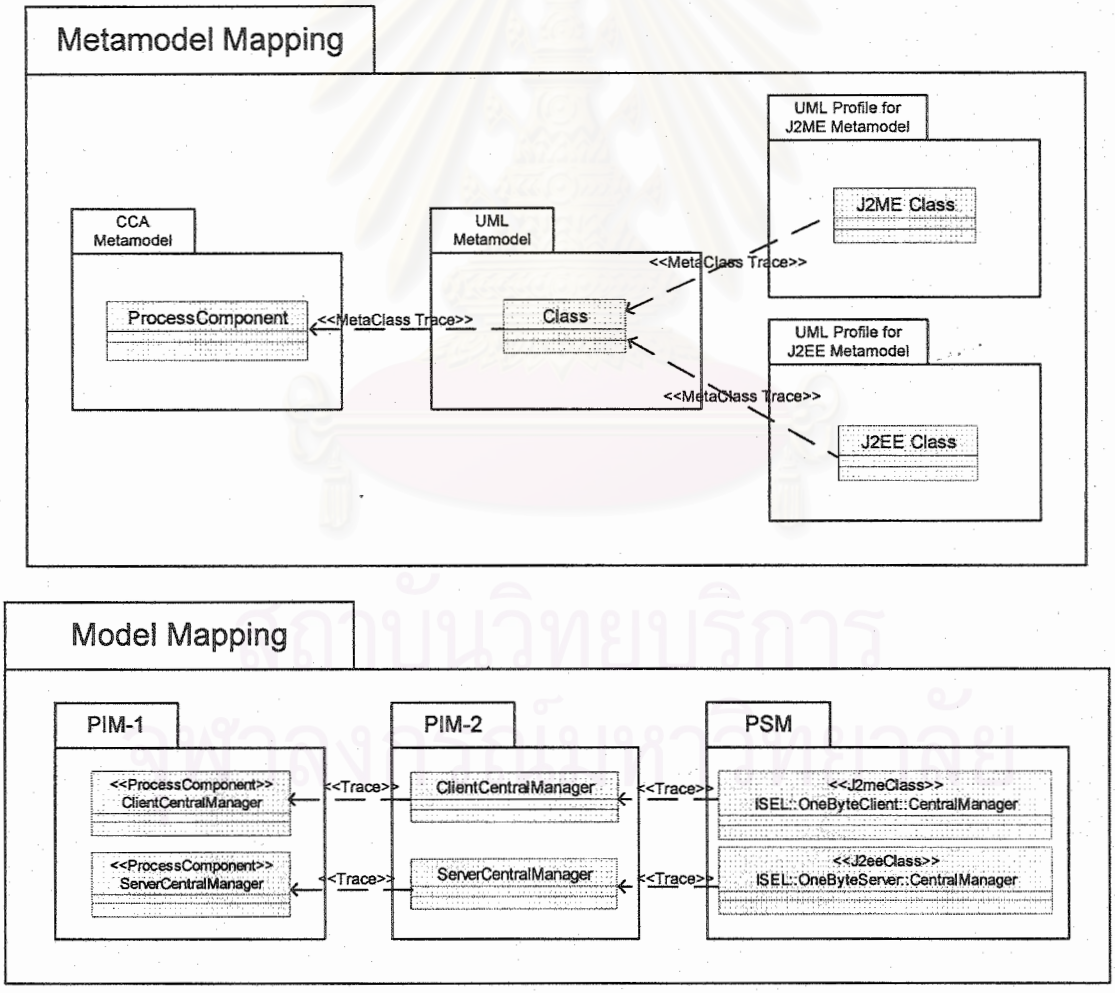
รูปที่ 3.6 ความสัมพันธ์ระหว่างพีไอเอ็ม และพีเอสเอ็ม [3]

นอกจากนั้น โอเอ็มจียังให้แนวคิดเกี่ยวกับการนำกลับมาใช้ใหม่ของกระบวนการแมปปิงได้ กล่าวคือ ในเมื่อพีไอเอ็มถูกสร้างขึ้นจากภาษาโมเดลซึ่งเป็นมาตรฐาน (เช่น ภาษายูเอ็มแอล) และแพลตฟอร์มที่ถูกนำไปอิมพลีเมนต์นั้น แม้จะมีหลายแพลตฟอร์มได้แต่ทุกแพลตฟอร์มก็เป็นแพลตฟอร์มมาตรฐาน (เช่น คอร์บา อีเจบี ดอตเนต และ เว็บเซอร์วิส เป็นต้น) ดังนั้นกระบวนการแมปปิงระหว่างสองสิ่งที่เป็นมาตรฐานจึงน่าจะสามารถใช้ซ้ำได้ ไม่ต้องกำหนดใหม่ในทุกครั้ง หรืออาจทำเป็นมาตรฐานแมปปิงได้ ซึ่งถ้ามาตรฐานมีการกำหนดขั้นตอนที่ชัดเจนมากเพียงพอก็น่าจะสามารถทำโดยอาศัยเครื่องมือแบบอัตโนมัติได้

แนวคิดเรื่องเครื่องมือในการแมปปิงแบบอัตโนมัตินี้ ในขณะนี้ยังค่อนข้างห่างไกลจากการนำไปปฏิบัติจริง เนื่องจากยังขาดความชัดเจนในการกำหนดกระบวนแมปปิง แต่จากการศึกษางานวิจัย [18-20] ทำให้ทราบลักษณะโดยคร่าวของเครื่องมือแมปปิงดังกล่าว โดยแสดงในรูปที่ 3.7 โดยเครื่องมือแมปปิงนี้ต้องสามารถรับแบบจำลองพีไอเอ็มของระบบเป็นอินพุท (1) โดยผลิตแบบจำลองพีเอสเอ็มเป็นเอาท์พุท (2) เนื่องจากแบบจำลองพีไอเอ็มที่เป็นอินพุทอาจแสดงได้หลากหลายรูปแบบ ดังนั้นเพื่อให้เครื่องมือแมปปิงเข้าใจสิ่งที่แบบจำลองพีไอเอ็มต้องการแสดงข้อมูลเกี่ยวกับเมตาโมเดลของภาษาที่ใช้แสดงแบบจำลองพีไอเอ็ม (3) จึงเป็นสิ่งที่เครื่องมือต้องทราบ นอกจากนั้นเครื่องมือต้องมีข้อมูลของกฎการแปลง (4) ซึ่งจะกำหนดการแปลงระหว่างเอนทิตีในแบบจำลองพีไอเอ็มเป็นเอนทิตีในแบบจำลองพีเอสเอ็ม ซึ่งมีลักษณะคล้ายกับตารางในรูปที่ 3.3 ดังนั้นเครื่องมือจึงต้องทราบข้อมูลของโมเดลเอนทิตีทั้งหมดที่มีในแพลตฟอร์มนั้น (5) รวมทั้งทราบเมตาโมเดลของแพลตฟอร์มนั้นๆ ด้วย (6) ตัวอย่างการแมปปิงแสดงไว้ในรูปที่ 3.8



รูปที่ 3.7 ลักษณะโดยคร่าวของเครื่องมือแมปปิง



รูปที่ 3.8 ตัวอย่างการแมปปิงในระดับเมตาโมเดล (บน) และระดับโมเดล (ล่าง)

จากรูปที่ 3.8 ทำให้อาจกล่าวได้ว่าการแมปปิงนั้นมี 2 มิติ คือในมิติของเมตาโมเดล และมิติของโมเดล ในมิติของเมตาโมเดลนั้นยกตัวอย่างเช่น การแปลงจากโปรเซสคอมโพเนนท์ไปเป็นยูเอ็มแอลคลาสและเป็นเจทูเอ็มอีคลาสตามลำดับ ซึ่งหลังจากกำหนดแมปปิงในมิติของเมตาโมเดลแล้วจึงมากำหนดแมปปิงในมิติของโมเดลอีกทีหนึ่งเช่น การแปลงจากโปรเซสคอมโพเนนท์ตัวจัดการฝั่งไคลเอนต์ไปเป็นยูเอ็มแอลคลาสและเจทูเอ็มอีคลาสของตัวจัดการฝั่งไคลเอนต์

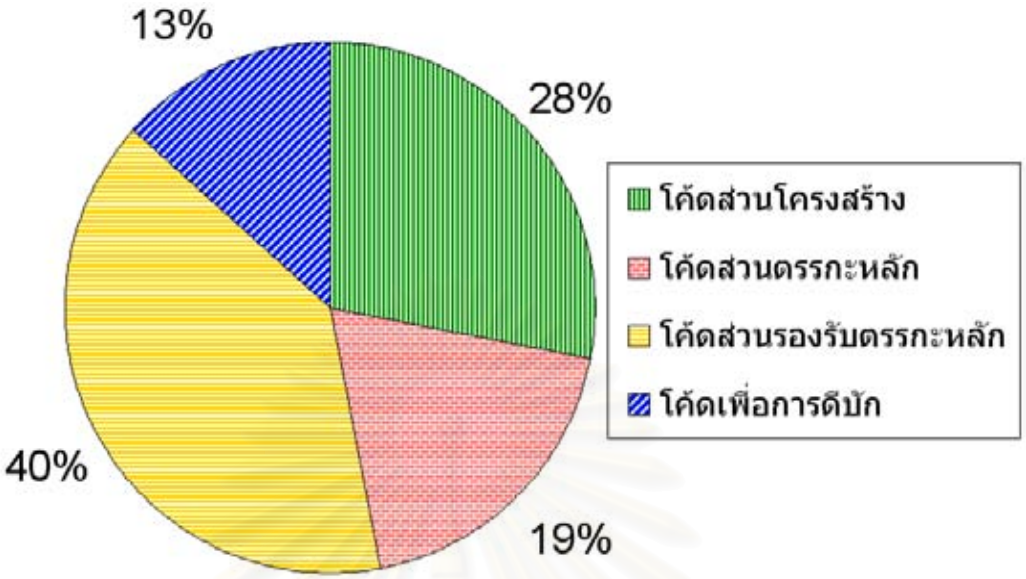
จากการทดลองพัฒนาระบบ ผู้วิจัยพบว่า การแปลงแบบจำลองโดยอัตโนมัติทั้งหมดโดยอาศัยเครื่องมือแมปปิงนั้นทำได้ยาก ทั้งนี้เนื่องจากในการทำงานจริงมักมีทางเลือกในการอิมพลีเมนต์ได้หลายทาง กล่าวคือมีแบบจำลองพีเอสเอ็มทีเป็นไปได้หลายแบบ ซึ่งเครื่องมือไม่น่าจะสามารถทราบได้ว่าแบบใดเป็นแบบที่เหมาะสม ยกตัวอย่างเช่น ถ้าในแบบจำลองพีเอสเอ็มต้องการเก็บข้อมูลเข้าที่เก็บ แล้วเรียกมาใช้ใหม่ได้เมื่อต้องการ ในการอิมพลีเมนต์ความต้องการเช่นนี้ก็มีทางเลือกได้หลายทาง เช่นการใช้ฐานข้อมูล การเก็บลงไฟล์ การเก็บลงหน่วยความจำในรูปแบบโครงสร้างข้อมูลในลักษณะต่างๆ เช่น ลิสต์ หรือ ทรี เป็นต้น

ดังนั้นผู้วิจัยจึงเห็นว่าแนวทางที่น่าจะเป็นไปในระยะอันใกล้ของเครื่องมือแมปปิงนี้คือ การช่วยคัดกรองการอิมพลีเมนต์ที่เป็นไปได้มาให้ผู้ใช้เลือก มากกว่าการให้เครื่องมือทำการเลือกเองแบบอัตโนมัติให้ผู้ใช้ ยกตัวอย่างเช่น หากผู้ออกแบบต้องการอิมพลีเมนต์ระบบบนแพลตฟอร์มทีเป็นอีเจบี เครื่องมือต้องมีหน้าในการคัดกรองเอาเอพีไอบนแพลตฟอร์มอีเจบีที่น่าจะเหมาะสมและตรงกับความต้องการของแต่ละคอมโพเนนท์ในแบบจำลองพีไอเอ็ม จากนั้นจึงแสดงรายการเอพีไอดังกล่าวมาเพื่อให้ผู้ใช้งานเลือกอีกทีหนึ่งเป็นต้น

3.4 การสร้างโค้ดจากแบบจำลองพีเอสเอ็ม

3.4.1 การแยกส่วนของโค้ด

ในขั้นตอนของการสร้างโค้ดจากแบบจำลองนั้น พบว่าแต่ละส่วนของโค้ดมีที่มาจากส่วนต่างๆ กันของแบบจำลอง โดยในงานวิจัยชิ้นนี้ได้จำแนกประเภทของโค้ดไว้เป็น 4 ส่วนได้แก่ โค้ดส่วนโครงร่าง โค้ดส่วนตรรกะหลัก โค้ดส่วนรองรับตรรกะหลัก และโค้ดสำหรับการดีบักโปรแกรม ซึ่งโค้ดแต่ละส่วนจะมีปริมาณที่แตกต่างกัน โดยในงานวิจัยนี้ได้ทำการวัดสัดส่วนโค้ดประเภทต่างๆ จากโปรแกรมตัวอย่างที่ได้อธิบายไว้ในหัวข้อที่ 4.5 ได้ผลดังแสดงในรูปที่ 3.9 ซึ่งเป็นสัดส่วนโดยเฉลี่ยของจำนวนบรรทัดของโค้ดแต่ละประเภทใน 1 มอดูล



รูปที่ 3.9 ค่าเฉลี่ยของสัดส่วนของโค้ดแต่ละประเภทของระบบตัวอย่าง

ประเด็นที่น่าสนใจจากผลที่ปรากฏในรูปที่ 3.9 ได้แก่สัดส่วนของโค้ดส่วนรองรับตรรกะหลัก และโค้ดเพื่อการดีบักรวมกันมีสัดส่วนมากกว่าครึ่งหนึ่งของปริมาณโค้ดทั้งหมด ในขณะที่โค้ดส่วนโครงร่างและโค้ดส่วนตรรกะหลัก ซึ่งมีความสำคัญต่อตรรกะการทำงานของระบบมากกว่า มีปริมาณโค้ดรวมกันคิดเป็นสัดส่วนเพียงร้อยละ 47 เท่านั้น ดังนั้นหากมีวิธีการใดซึ่งช่วยลดภาระของโปรแกรมเมอร์ในการสร้างโค้ดส่วนรองรับตรรกะหลัก และโค้ดเพื่อการดีบักได้ ก็น่าที่จะช่วยลดปริมาณงานที่โปรแกรมเมอร์ต้องโค้ด และประหยัดเวลาในการพัฒนาโปรแกรมเป็นอย่างมาก

3.4.1.1 โค้ดส่วนโครงร่าง

เป็นส่วนของการประกาศโครงสร้างของโค้ดตามหลักเกณฑ์เชิงวากยสัมพันธ์ (Syntax) ของภาษานั้นซึ่งในที่นี้คือภาษาจาวา ประกอบด้วย 3 ส่วนหลักดังแสดงในรูปที่ 3.10 ได้แก่

1. การประกาศชื่อคลาส แพคเกจ และโครงสร้างของคลาส
2. การประกาศชื่อเมธอด และชื่อและชนิดของพารามิเตอร์
3. การประกาศชื่อและชนิดแอททริบิวท์

นอกจากนี้ในเทคโนโลยีมิดเดิลแวร์สมัยใหม่ ก็จะมีการกำหนดโค้ดส่วนโครงร่างเหล่านี้เช่นกัน ถึงแม้หลักการจะเหมือน 3 ส่วนข้างต้นแต่ก็มีรายละเอียดที่ต่างออกไปเช่น คอร์บาจะมีการ

กำหนดเป็น คำนิยามอินเทอร์เฟส (Interface Definition) ส่วนในอีเจบีจะต้องมีการสร้างโฮมอินเทอร์เฟส และรีโมทอินเทอร์เฟสเสมอทุกครั้งที่สร้างวัตถุบีน เป็นต้น

```
package th.ac.chula.eng.cp.ISEL.LittleRockProject1

public Class DummyClass
{
      Type1 attribute1;
      Type2 attribute2;

      Type3 method1(){
      }
      Type4 method2(Type5 parameter1, Type6 parameter2){
      }
}
```

รูปที่ 3.10 ตัวอย่างโค้ดส่วนโครงร่าง

3.4.1.2 โค้ดส่วนตรรกะหลัก

เป็นส่วนของโค้ดที่บรรจุตรรกะของการทำงานหลักของระบบไว้ ซึ่งเป็นการอิมพลีเมนต์หน้าที่ของเมธอดตามข้อกำหนดของเมธอดนั้น การแบ่งแยกว่าส่วนใดเป็นส่วนตรรกะหลักนั้นคงต้องพิจารณาถึงจุดประสงค์ หรือหน้าที่หลักของเมธอดนั้นๆ ว่ามีหน้าที่อะไร ซึ่งส่วนใหญ่แล้วตรรกะในส่วนนี้เกิดจากการเรียกใช้คำสั่งไม่กี่คำสั่ง หรือการเรียกใช้เอพีไอไม่กี่ตัวเท่านั้น ยกตัวอย่างเช่น โปรแกรมนับคำในไฟล์ ในรูปที่ 3.11 ซึ่งส่วนตรรกะหลักนั้นเป็นการอ่านไฟล์จากนั้นแปลงข้อมูลไบนารีของไฟล์เป็นสตริงแล้วใช้เอพีไอ StringTokenizer นับจำนวนคำออกมา

```
public int countToken(File f){

      int returnValue;

      charASCII = theFileReader.read();
      while (charASCII != -1)
      {
            theStringWriter.write(charASCII);
            charASCII = theFileReader.read();
      }
      theStringOfFile = theStringWriter.toString();
      returnValue = theStringTokenizer.countTokens();

      return returnValue;

      }
}
```

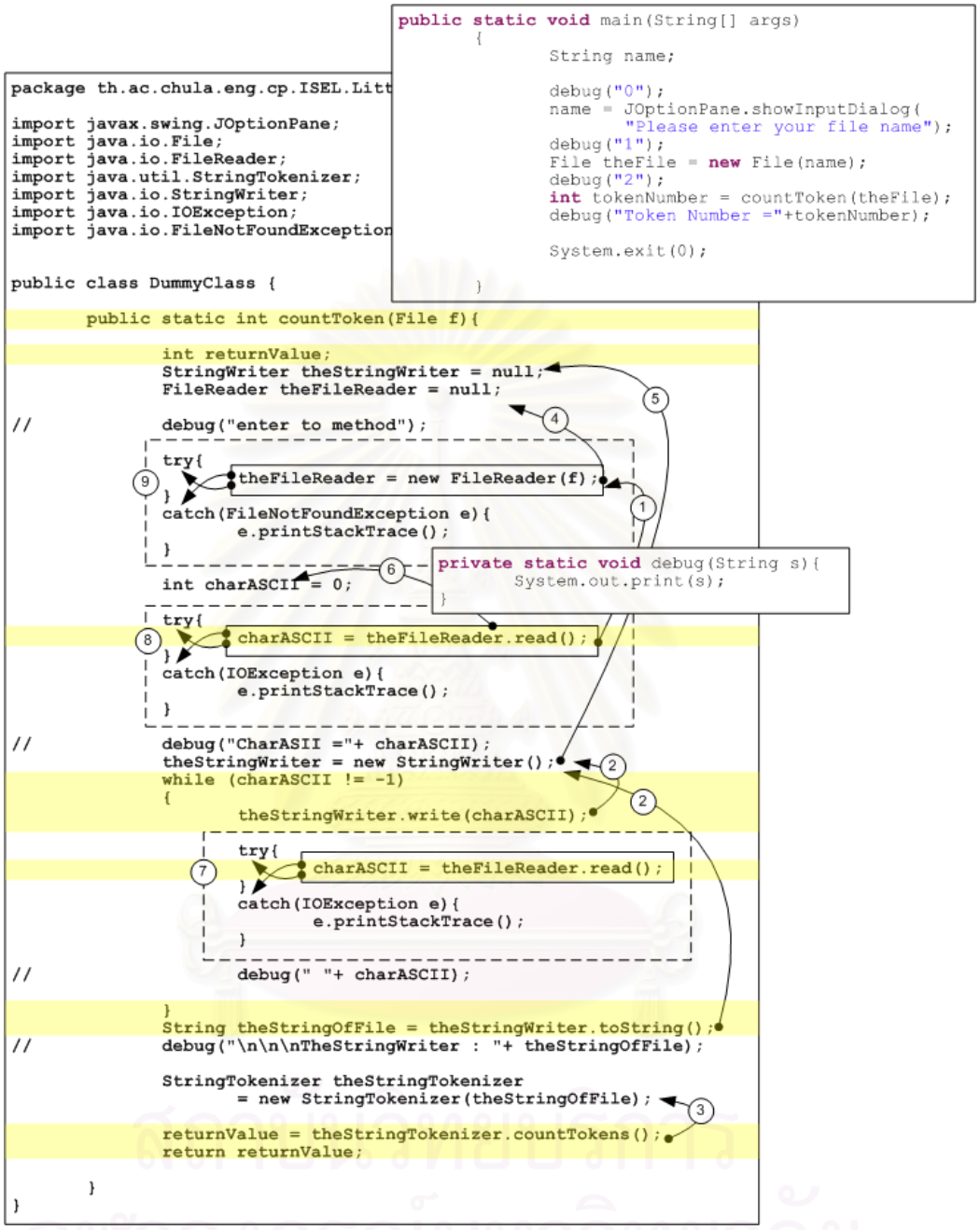
รูปที่ 3.11 ตัวอย่างของโค้ดส่วนตรรกะหลัก

3.4.1.3 โค้ดส่วนรองรับตรรกะหลัก

เป็นส่วนของโค้ดที่ไม่ได้เป็นตรรกะของการทำงานหลักของเมธอด แต่จำเป็นต้องมีเพื่อให้โค้ดส่วนตรรกะหลักสามารถทำงานได้โดยสมบูรณ์ ยกตัวอย่างเช่นในรูปที่ 3.12 ซึ่งเป็นโค้ดของเมธอดเดียวกับในรูปที่ 3.11 แต่ต่างกันที่รูปที่ 3.12 มีการเพิ่มโค้ดส่วนรองรับตรรกะหลักเข้าไปเพื่อให้โค้ดส่วนตรรกะหลักในรูปที่ 3.12 สามารถทำงานได้จริง โดยบรรทัดที่มีการแรเงาคือโค้ดส่วนตรรกะหลักซึ่งมีอยู่เดิม ส่วนบรรทัดที่ไม่ได้แรเงาและไม่ใช่โค้ดส่วนโครงร่าง คือโค้ดส่วนรองรับตรรกะหลัก ซึ่งเส้นลูกศรจากบรรทัดโค้ดส่วนตรรกะหลักจะเชื่อมโยงไปยังบรรทัดโค้ดที่เพิ่มขึ้นเพื่อรองรับการทำงานของโค้ดส่วนตรรกะหลักนั้นๆ สำหรับวิธีการสร้างโค้ดส่วนรองรับตรรกะหลักนั้นจะกล่าวรายละเอียดในหัวข้อถัดไป

ในงานวิจัยนี้ได้แบ่งโค้ดส่วนรองรับตรรกะหลักออกเป็น 4 ประเภทได้แก่

1. ส่วนจัดการเอ็กซ์เซปชัน (Exception Handling) เกิดจากการที่เอพีไอที่ใช้ถูกกำหนดให้สามารถโยนเอกซ์เซปชันบางชนิดออกมาได้ ดังนั้นจึงต้องมีโค้ดส่วนหนึ่งถูกเขียนขึ้นเพื่อดูแลในกรณีที่เกิดเอ็กซ์เซปชันดังกล่าวในขณะทำงาน (Run Time)
2. ส่วนนำเข้าไลบรารี (Import) ซึ่งปกติแล้วการใช้เอพีไอหรือไลบรารีใดๆ ทุกครั้งจำเป็นต้องบอกที่มาหรือแหล่งอ้างอิงว่านำเข้ามาจากไลบรารีชุดใด ทั้งนี้เนื่องจากมีความเป็นไปได้ที่เอพีไอจากคนละไลบรารีกันจะมีชื่อซ้ำกันได้ ดังนั้นจึงต้องมีการระบุที่มาของเอพีไอที่ใช้ทุกครั้ง
3. การเตรียมค่าอาร์กิวเมนต์สำหรับการเรียกใช้เมธอด และการจัดการกับค่าที่รีเทิร์นเป็นส่วนที่ขึ้นกับชนิดของเอพีไอที่เลือกใช้ เพราะแต่ละเอพีไอจะมีความต้องการข้อมูลนำเข้า (Input) ที่ไม่เหมือนกัน โค้ดส่วนนี้เป็นส่วนที่เป็นการประกาศให้ชัดขึ้นว่าอาร์กิวเมนต์แต่ละตัวที่ใช้นั้น ได้มาอย่างไร
4. การเพิ่มตรรกะเนื่องจากเป็นไปตามรูปแบบการเรียกใช้เอพีไอ (API Usage Pattern) เกิดจากการสังเกตการเขียนโปรแกรมในทางปฏิบัติเมื่อเรียกใช้เอพีไอบางตัวว่า ในการเรียกใช้งานเอพีไอตัวนั้นในแต่ละครั้งจะมีส่วนโค้ดรองรับตรรกะที่เหมือนกันหรือคล้ายกันมาก ทำให้สามารถพิจารณาได้ว่า แท้จริงแล้วการใช้งานเอพีไอนั้นๆ น่าจะมีรูปแบบการใช้งานที่ชัดเจน ซึ่งเครื่องมือสามารถเรียนรู้ได้ อันจะช่วยให้สามารถสร้างโค้ดส่วนนี้อย่างอัตโนมัติในอนาคต



รูปที่ 3.12 ตัวอย่างของโค้ดส่วนรองรับตรรกะหลัก

3.4.1.4 โค้ดสำหรับการดีบักโปรแกรม

เป็นส่วนของโค้ดที่ไม่เกียวข้องใดๆ กับการทำงานของระบบเลย แต่จำเป็นมากในขั้นตอนของการพัฒนา เนื่องจากแทบไม่มีครั้งใดที่โปรแกรมที่พัฒนาได้จะปราศจากข้อผิดพลาดตั้งแต่การคอมไพล์ครั้งแรก ส่วนของโค้ดสำหรับการดีบักแล้วโดยทั่วไป ก็คือการพิมพ์ผลลัพธ์ในขั้นตอนต่างๆ

ออกมานั่นเอง ทั้งนี้เพื่อตรวจสอบว่าผลลัพธ์ในขั้นตอนนั้นๆ ตรงกับผลลัพธ์ที่โปรแกรมเมอร์คาดการณ์ไว้หรือไม่ นอกจากนั้นยังสามารถตรวจสอบการทำงานว่าได้ทำงานถึงขั้นตอนที่ต้องการหรือไม่ ยกตัวอย่างของโค้ดสำหรับการดีบักแสดงในรูปที่ 3.12 ซึ่งจะเห็นได้ว่ามีเครื่องหมาย // นำหน้าบรรทัดที่เป็นโค้ดสำหรับการดีบักทุกบรรทัด ทั้งนี้เนื่องจากในทางปฏิบัติเมื่อโปรแกรมเสร็จสมบูรณ์แล้ว ผู้พัฒนามักจะลบโค้ดสำหรับการดีบักนี้ออกจากโปรแกรมเพราะไม่มีส่วนเกี่ยวข้องกับการทำงานตรรกะหลักเลย

3.4.2 การสร้างโค้ดส่วนโครงร่าง

ในภาษายูเอ็มแอลนั้นมีการแบ่งมุมมองของระบบผ่านไดอะแกรมต่างๆ ซึ่งสำหรับมุมมองที่เกี่ยวกับโครงสร้างของระบบในระดับโมเดลนั้นจะแสดงด้วยคลาสไดอะแกรม (ไม่พูดถึงโครงสร้างในระดับอิมพลีเมนต์ ที่ใช้พวกคอมโพเนนท์ไดอะแกรม) ดังนั้นการสร้างโค้ดส่วนโครงร่างจากแบบจำลองนั้นสามารถทำได้โดยตรงจากแผนภาพคลาส [21, 22] โดยมีหลักเกณฑ์ดังแสดงในตารางที่ 3.1

ตารางที่ 3.1 การแปลงโค้ดส่วนโครงร่างจากแผนภาพคลาส

เอนทิตีในแผนภาพคลาส	ส่วนของโค้ด
ชื่อคลาส	การประกาศคลาสใช้ชื่อตามแผนภาพ
ชื่อเมธอด และชนิดของพารามิเตอร์	การประกาศเมธอด และชนิดของพารามิเตอร์
ชื่อและชนิดของแอททริบิวท์	การประกาศชื่อและชนิดของแอททริบิวท์
แอสโซซิเอชัน	การประกาศชื่อและชนิดของแอททริบิวท์ตามคลาสอื่นที่มีแอสโซซิเอชันกับคลาสที่สนใจ

อย่างไรก็ตาม ยังคงมีส่วนรายละเอียดของแผนภาพคลาสที่ไม่ได้ถูกนำมาสร้างเป็นโค้ดส่วนโครงร่าง ยกตัวอย่างเช่น คอมโพซิชัน แอกกรีเกชัน ดีเพนเดนซี มัลติพลิซิตี และโน้ต ทั้งนี้เนื่องจากยังคงมีความไม่ชัดเจนในการแมปปิงรายละเอียดดังกล่าว ซึ่งในงานวิจัยนี้จะละเลยการแมปปิงส่วนรายละเอียดเหล่านี้ไปเป็นโค้ดด้วย แต่ในขณะเดียวกันไม่มีส่วนของโค้ดส่วนโครงร่างที่ไม่สามารถสร้างได้จากแผนภาพคลาส (ไม่รวม Comment ซึ่งในที่นี้ไม่ถือเป็นส่วนหนึ่งของโค้ดเพราะเป็นเพียงการช่วยให้โปรแกรมเมอร์เข้าใจโค้ดได้มากขึ้น แต่ไม่มีผลกับการทำงานของโค้ดแต่อย่างใด) ดังนั้นจึงอาจกล่าวได้ว่าแผนภาพคลาส ให้ข้อมูลของระบบมากกว่าที่จำเป็นต้องใช้ในการสร้างโค้ดส่วนโครงร่าง

การสร้างโค้ดส่วนโครงร่างนี้ถือเป็นการแมปปิงขั้นที่ง่ายที่สุด ซึ่งเครื่องมือส่วนใหญ่ในท้องตลาดสามารถทำได้อยู่แล้ว ยกตัวอย่างเช่น เรชันนอลโรส (Rational Rose) โพไซดน (Poseidon) หรือ แมจิกดรอว์ (Magic Draw) เป็นต้น

3.4.3 การสร้างโค้ดส่วนตรรกะหลัก

โค้ดส่วนตรรกะหลัก เป็นส่วนของการอิมพลีเมนต์พฤติกรรมของเมธอดตามข้อกำหนดเชิงพฤติกรรมของเมธอดนั้นๆ ดังนั้นโค้ดส่วนตรรกะหลักจึงต้องถูกสร้างมาจากส่วนของโมเดลที่อธิบายถึงพฤติกรรมของระบบ ซึ่งในภาษายูเอ็มแอลนั้นมีแผนภาพที่ใช้อธิบายพฤติกรรมถึง 4 แผนภาพได้แก่

1. แผนภาพซีเควนซ์ (Sequence Diagram)
2. แผนภาพคอลแลบอเรชัน (Collaboration Diagram)
3. แผนภาพแอคทิวิตี (Activity Diagram)
4. แผนภาพสเตทชาร์ต (State Chart Diagram)

จุดประสงค์ของการมีแผนภาพถึง 4 แผนภาพในการอธิบายพฤติกรรมของระบบนั้นเพื่อให้สามารถเห็นภาพจากมุมมองที่ต่างกันได้ โดยแต่ละแผนภาพจะมีความเหมาะสมในการอธิบายพฤติกรรมในแต่ละโอกาสที่ต่างกัน อย่างไรก็ตามการมีแผนภาพถึง 4 แผนภาพในการอธิบายพฤติกรรมของระบบก็เป็นปัญหาอย่างหนึ่งเกี่ยวกับการสร้างโค้ดจากแผนภาพเหล่านี้ เพราะยูเอ็มแอลไม่ได้บังคับว่าต้องใช้แผนภาพใดในการอธิบายพฤติกรรมของแบบจำลอง กล่าวคือผู้ออกแบบสามารถใช้แผนภาพใดก็ได้ที่คิดว่าเหมาะสมต่อการอธิบาย นอกจากนี้บางแผนภาพยังเป็นการอธิบายพฤติกรรมของระบบในเชิงนามธรรม (Abstract Behavioral Model) ได้แก่ แผนภาพแอคทิวิตี และแผนภาพสเตทชาร์ต ซึ่งยอมให้เขียนคำอธิบายสถานะ (State) และการกระทำ (Action) ในลักษณะของการสื่อความหมายเชิงนามธรรม โดยอาจไม่ต้องระบุการเชื่อมโยงไปยังเมธอดของคลาสก็ได้ ทำให้การอธิบายแบบจำลองทำได้สะดวกเพราะเขียนอธิบายด้วยภาษามนุษย์ได้ แต่ในขณะเดียวกันก็ทำให้การเทียบเคียงแบบจำลองกับโค้ดโดยตรงเป็นไปได้ยากขึ้น และแม้ในแผนภาพซีเควนซ์และแผนภาพคอแลบอเรชันจะมีการอธิบายพฤติกรรมโดยการเรียกใช้งานเมธอดของคลาสอื่น ซึ่งส่วนนี้สามารถเทียบเคียงโดยตรงกับโค้ดได้ แต่ในขณะเดียวกันกลับอนุญาตให้มีการสื่อสารระหว่างคลาสในลักษณะที่เป็นการส่งเมสเสจ (Message Passing) ซึ่งยังไม่มีวิธีการมาตรฐานใดในการแปลงโมเดลส่วนนี้เป็นโค้ด

ในแวดวงของการวิจัยนั้นมีงานวิจัยที่กล่าวถึงเรื่องนี้เป็นจำนวนมาก แต่โดยส่วนใหญ่แล้วเป็นการแปลงเฉพาะแผนภาพคลาสไปเป็นโค้ด หรือรวมการแปลงแผนภาพเชิงพฤติกรรมเพียงหนึ่งหรือสองแผนภาพไปเป็นโค้ด [17, 21, 23]

สำหรับในงานวิจัยชิ้นนี้เลือกใช้แผนภาพสองแผนภาพในการอธิบายพฤติกรรมของระบบ ได้แก่

1. แผนภาพแอคทิวิตี ใช้อธิบายพฤติกรรมโดยรวมในระดับของระบบ หรือ ระดับของคลาส
2. แผนภาพซีเควนซ์ ใช้อธิบายพฤติกรรมในระดับของเมธอด โดยแสดงในลักษณะของการปฏิสัมพันธ์ (Interaction) ระหว่างคลาส

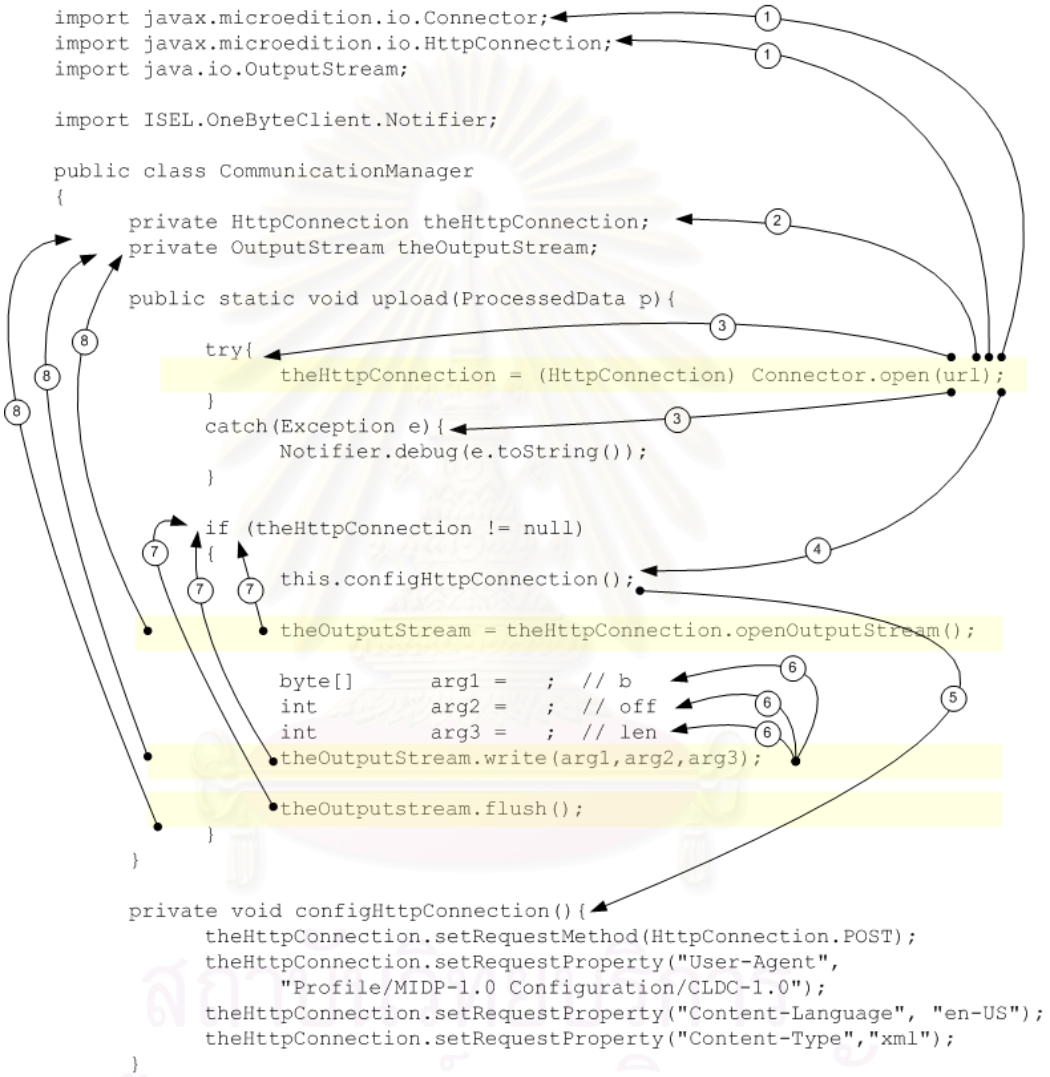
3.4.4 การสร้างโค้ดส่วนรองรับตรรกะหลัก

โดยทั่วไปแล้วโค้ดส่วนรองรับตรรกะหลักนั้น ไม่ได้ถูกกำหนดอยู่ในแบบจำลองใด เนื่องจากเป็นส่วนที่ไม่ได้เกี่ยวข้องโดยตรงกับหน้าที่ของโปรแกรม แต่จำเป็นต้องมีเพื่อให้โค้ดส่วนตรรกะหลักสามารถทำงานได้อย่างถูกต้อง ดังนั้นงานในการสร้างโค้ดส่วนนี้จึงต้องเป็นงานที่อาศัยความฉลาดของเครื่องมือ โดยเครื่องมือต้องทราบรายละเอียดการใช้งานของเอพีไอแต่ละตัว (ซึ่งอาจจะมีในเอกสารคู่มือการใช้งานเอพีไอเช่น จาวาด็อก) จากนั้นจึงนำข้อมูลดังกล่าวไปสร้างโค้ดส่วนนี้ขึ้นมา

จากการแบ่งประเภทของโค้ดส่วนรองรับตรรกะหลักในหัวข้อ 3.4.1.3 ในงานวิจัยนี้ได้เลือกใช้รูปแบบอย่างง่ายในการสร้างโค้ดส่วนนี้โดยแยกเป็นประเภทได้ดังนี้

1. โค้ดส่วนนำเข้าไลบรารีจะเพิ่มการ import ทุกครั้งที่มีการเรียกใช้คลาสใดๆ โดยจะ import หนึ่งครั้งต่อหนึ่งคลาสที่เรียกใช้
2. โค้ดส่วนจัดการเอ็กซ์เซปชัน ใช้ try คร่อมบล็อกของโค้ดที่สามารถโยนเอ็กซ์เซปชันออกมาได้ แล้วทำการ catch แต่ละเอ็กซ์เซปชันแล้วทำการพิมพ์ชนิดของเอ็กซ์เซปชันออกมา
3. การเตรียมค่าอาร์กิวเมนต์ และการจัดการกับค่าที่รีเทิร์น จะตั้งชื่อตัวแปรอาร์กิวเมนต์ตามข้อตกลงการตั้งชื่อตัวแปรสำหรับงานวิจัยนี้ซึ่งจะกล่าวรายละเอียดในบทถัดไป โดยจะมีส่วนของโค้ดเพื่อการรับค่าอาร์กิวเมนต์ ก่อนการเรียกใช้เอพีไอเสมอ ส่วนค่ารีเทิร์นนั้นจะมีตัวแปรรับค่ารีเทิร์น ซึ่งก่อนออกจากเมธอดจะมีโค้ดการรีเทิร์นตัวแปรดังกล่าว

4. โค้ดส่วนที่เป็นไปตามรูปแบบการใช้งานเอพีไอ จะสมมติว่าเอพีไอทุกตัวมีเอกสารการใช้งานที่ระบุรูปแบบการใช้งานดังกล่าวให้อยู่แล้ว โดยเครื่องมือต้องสามารถอ่านข้อมูลดังกล่าวได้แล้วนำมาสร้างเป็นโค้ดได้



รูปที่ 3.13 การสร้างโค้ดส่วนรองรับตรรกะหลัก

จากรูปที่ 3.13 เป็นการอิมพลีเมนต์เมธอดชื่อ upload ซึ่งเป็นการส่งข้อมูลจากเครื่องไคลเอนต์ไปยังเครื่องเซิร์ฟเวอร์ ซึ่งในกรณีนี้ผู้ออกแบบเลือกที่จะอิมพลีเมนต์โดยใช้การเชื่อมต่อ

แบบ HttpConnection ซึ่งแท้จริงแล้วส่วนของโค้ดส่วนตรรกะหลักเป็นการเรียกใช้เอพีไอของแพลตฟอร์มเจทูเอ็มอี ซึ่งน่าจะสามารถเขียนได้ในบรรทัดเดียวคือ

```
theHttpConnection.openOuputStream().write();
```

แต่ในทางปฏิบัติไม่สามารถทำเช่นนั้นได้ เนื่องจากการใช้งานเอพีไอทุกเอพีไอมักมีความต้องการบางประการก่อนใช้งานเสมอ ยกตัวอย่างในกรณีข้างต้นคือ เมธอด write() เป็นเมธอดของคลาส OutputStream ซึ่งมีอาร์กิวเมนต์ของการเรียกใช้สามตัว (6) ซึ่งจะเห็นได้ว่า byte[23] ซึ่งเป็นข้อมูลที่จะถูกเขียนไปยังเครื่องเซิร์ฟเวอร์ยังไม่สอดคล้องกับ processedData ซึ่งเป็นข้อมูลรับเข้าของเมธอดนี้ ทำให้ต้องมีขั้นตอนของการเตรียมค่าอาร์กิวเมนต์ ซึ่งในกรณีนี้คือการแปลง processedData ให้เป็น byte[23] นั่นเอง (แต่ในตัวอย่างข้างต้นไม่ได้ระบุรายละเอียดวิธีการแปลงไว้) อีกทั้งการเรียกเมธอด write() ยังไม่สามารถเรียกโดยลำพังได้ จำเป็นต้องมีการเรียกเมธอด flush() ปิดท้ายการเรียกเมธอด write() ครั้งสุดท้ายทุกครั้งเสมอ นอกจากนั้น ก่อนการใช้งานต้องมีการสร้างอินสแตนซ์ของคลาส OutputStream ขึ้นเสียก่อน โดยการเรียกเมธอดของ openOutputStream() ของอินสแตนซ์ของคลาส HttpConnection (7)-(8) ส่วนการสร้างอินสแตนซ์ของคลาส HttpConnection ก็จะต้องถูกสร้างจากการเรียกสแตติกเมธอด open ของคลาส Connection เสมอ (2)-(3) และมีความเป็นไปได้ทีจะติดต่อไม่ได้และเกิดเอ็กซ์เซปชันขึ้น ดังนั้นจึงต้องมีโค้ดส่วนจัดการเอ็กซ์เซปชันเกิดขึ้น

รายละเอียดของข้อกำหนดการใช้งานเอพีไอ และรูปแบบการใช้งานเอพีไอเหล่านี้โปรแกรมเมอร์สามารถทำความเข้าใจได้จากการศึกษาคู่มือการใช้งานเอพีไอ ซึ่งมักจะมีตัวอย่างการใช้งานมาให้ แต่คำถามคือหากต้องการให้เครื่องมือทราบความรู้นี้ เพื่อให้สร้างโค้ดทีสมบูรณ์มาให้อย่างอัตโนมัติ เครื่องมือจะสามารถเรียนรู้ข้อมูลเหล่านี้ได้อย่างไร อย่างไรก็ตามงานวิจัยชิ้นนี้ก็มิได้ตอบคำถามดังกล่าว โดยได้อนุมานว่าในอนาคตคำถามนี้จะสามารถมีคำตอบได้ โดยในงานวิจัยชิ้นนี้จะได้นำคำตอบของคำถามนี้ไปใช้ในการสร้างโค้ดเลย

3.4.5 การสร้างโค้ดสำหรับการดีบัก

ดังที่ได้กล่าวไปแล้วในหัวข้อ 3.4.1.4 ว่าโค้ดสำหรับการดีบักนั้นไม่มีความเกียวข้องกับการทำงานของโปรแกรมเลย ซึ่งโดยทั่วไปแล้วโค้ดส่วนนี้จะถูกลบออกเมือโปรแกรมเสร็จสมบูรณ์ แต่ในขณะเดียวกันโค้ดส่วนนี้กลับจำเป็นมากในขั้นตอนของการพัฒนาโปรแกรม

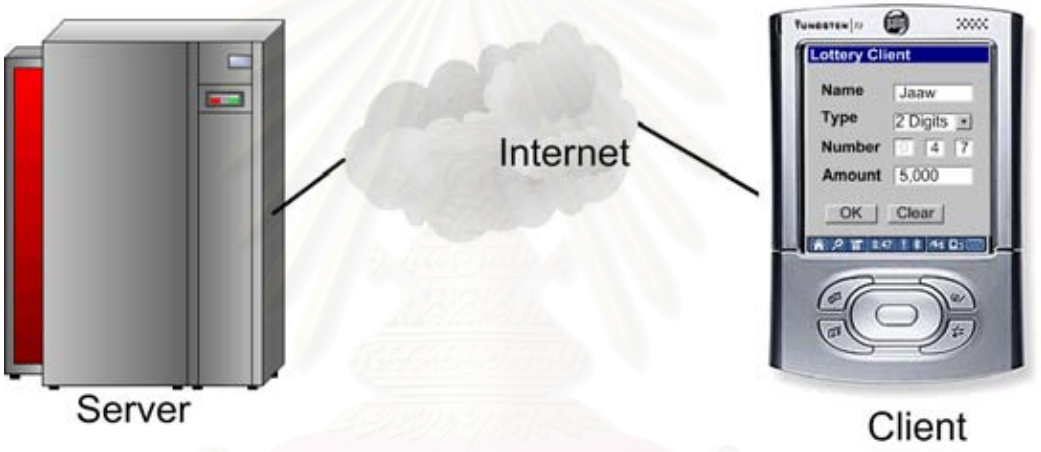
การสร้างโค้ดในส่วนนี้นั้นผู้วิจัยเห็นว่าไม่มีความจำเป็นที่เครื่องมือสร้างโค้ดต้องเป็นผู้สร้างให้ แต่ควรเป็นการสร้างหลังจากที่โค้ดสามส่วนแรกถูกสร้างมาแล้ว โดยควรอยู่ในขั้นตอนของการทดสอบโปรแกรมที่สร้างได้มากกว่า

อย่างไรก็ตามเนื่องจากในงานวิจัยชิ้นนี้โค้ดทั้งหมดถูกเขียนขึ้นด้วยมือ (ไม่ได้ใช้เครื่องมือสร้างให้) ดังนั้นจึงยังปรากฏโค้ดสำหรับการดีบักอยู่ โดยเป็นลักษณะของการเรียกใช้คลาสชื่อ Notifier ซึ่งจะพิมพ์ผลลัพธ์ของขั้นตอนต่างๆ ออกมาให้ และเมื่อโปรแกรมเสร็จสมบูรณ์แล้วจะลบโค้ดส่วนของการดีบักออกโดยแก้ไขคลาส Notifier ให้ไม่ต้องทำงานใดๆ

บทที่ 4
ตัวอย่างการพัฒนาระบบด้วยเอ็มดีเอ

4.1 ความต้องการของระบบตัวอย่าง

ในงานวิจัยชิ้นนี้ได้มีการทดลองพัฒนาระบบด้วยเอ็มดีเอ โดยมีขั้นตอนตั้งแต่การออกแบบระบบ สร้างแบบจำลอง และอิมพลีเมนต์ระบบจนสามารถทำงานได้จริง โดยมีวัตถุประสงค์เพื่อศึกษาถึงปัญหาที่เกิดขึ้นในการพัฒนาระบบจริง อย่างไรก็ตามเพื่อหลีกเลี่ยงความซับซ้อนเชิงตรรกะทางธุรกิจของระบบโดยมุ่งเป้าเพื่อศึกษาความซับซ้อนของกระบวนการพัฒนาระบบเท่านั้น ดังนั้นในงานวิจัยนี้จึงได้มีการกำหนดโจทย์ตัวอย่างขึ้นโดยกำหนดให้เป็นระบบที่มีความต้องการเชิงธุรกิจ (Business Requirement) ที่ง่าย ไม่ซับซ้อน



รูปที่ 4.1 ระบบตัวอย่าง

จากรูปที่ 4.1 แสดงลักษณะของระบบตัวอย่างซึ่งเป็นระบบลอตเตอรีออนไลน์ โดยตัวแทนขายจะทำการเก็บข้อมูลการซื้อลอตเตอรีจากลูกค้า ซึ่งลูกค้ามีทางเลือกในการซื้อได้สามแบบได้แก่ เลขท้าย 2 ตัว เลขท้าย 3 ตัวเต็ง (ตัวเลขสลับที่กันไม่ได้) และเลขท้าย 3 ตัวโต๊ด (ตัวเลขสลับที่กันได้) โดยรายละเอียดของข้อมูลที่ต้องการในการสั่งซื้อได้แก่ ชื่อผู้ซื้อ ชนิดของลอตเตอรีที่ซื้อ เลขที่ซื้อ จำนวนเงินที่ซื้อ โดยข้อมูลดังกล่าวจะเก็บลงในเครื่องคอมพิวเตอร์มือถือปาล์มของตัวแทนขาย โดยจะมีฟอร์มในการกรอกข้อมูลดังกล่าวทั้งหมด จากนั้นเมื่อสามารถทำการเชื่อมต่อกับเครือข่ายได้จึงอัปโหลด (Upload) ข้อมูลทั้งหมดไปยังเครื่องเซิร์ฟเวอร์ จากนั้นจึงลบทิ้งข้อมูลที่ส่งไปแล้ว โดยรูปแบบของข้อมูลที่ส่งไปยังเครื่องเซิร์ฟเวอร์นั้นเป็นข้อมูลเอ็กซ์เอ็มแอล (XML) จำนวนเครื่องไคลเอนต์ซึ่งเป็นเครื่องปาล์มนั้น สามารถมีมากกว่าหนึ่งเครื่องได้ นอกจากนั้น ทุกครั้งที่เครื่อง

เซิร์ฟเวอร์รับข้อมูลจากเครื่องไคลเอนต์ควรมีการจดบันทึกการรับเข้าเพื่อใช้เป็นหลักฐานในภายหลัง

จากความต้องการเชิงธุรกิจข้างต้น ขั้นตอนต่อไปคือการวิเคราะห์และออกแบบระบบ โดยการพัฒนาระบบนั้นจะยึดตามแนวทางการพัฒนาด้วยเอ็มดีเอ โดยเริ่มจากการสร้างแบบจำลองพีไอเอ็ม และแบบจำลองพีเอสเอ็ม จนถึงการแปลงเป็นโค้ดโปรแกรมตามลำดับ โดยจะนำเสนอในหัวข้อถัดไป

4.2 การวิเคราะห์ระบบ

จากลักษณะของระบบที่ได้กล่าวถึงในหัวข้อที่ 4.1 ทำให้สามารถบอกได้ว่าระบบมีลักษณะเป็นระบบกระจายอย่างหนึ่งโดยมีรูปแบบเป็นแบบไคลเอนต์-เซิร์ฟเวอร์ การเชื่อมต่อระหว่างกันมีลักษณะไม่ถาวรกล่าวคือ จะมีเพียงบางเวลาเท่านั้นที่มีการเชื่อมต่อกัน โดยจำนวนของเครื่องไคลเอนต์นั้นมีได้มากกว่าหนึ่งเครื่อง ในขณะที่เครื่องเซิร์ฟเวอร์นั้นมีเพียงเครื่องเดียวเท่านั้น นอกจากนั้นการส่งข้อมูลจากเครื่องไคลเอนต์ไปยังเครื่องเซิร์ฟเวอร์นั้นจำเป็นต้องมีการแปลงข้อมูลให้อยู่ในรูปแบบของภาษาเอ็กซ์เอ็มแอลเสียก่อน เนื่องจากการสร้างแบบจำลองพีไอเอ็มจากข้อกำหนดของระบบดังกล่าว เป็นการวิเคราะห์เพื่อหาความต้องการเชิงธุรกิจของระบบอย่างแท้จริงโดยไม่สนใจรายละเอียดของการอิมพลีเมนต์ ในงานวิจัยชิ้นนี้ในระดับพีไอเอ็ม ผู้วิจัยจึงมองระบบดังกล่าวเป็นเพียงการรับส่งข้อมูลของอุปกรณ์ใดๆ 2 อุปกรณ์เท่านั้น

สำหรับการทำงานของเครื่องไคลเอนต์นั้นผู้วิจัยมองระบบเป็นอุปกรณ์ใดๆ 1 ตัวซึ่งมีตัวจัดการฝั่งไคลเอนต์ (ClientCentralManager) ทำหน้าที่ในการควบคุมลำดับการทำงานทั้งหมดของไคลเอนต์ ส่วนงานในการรับข้อมูลจากผู้ใช้งานผ่านส่วนติดต่อผู้ใช้งานนั้นได้มอบหมายให้มีตัวจัดการแหล่งข้อมูล (DataSourceManager) เป็นตัวจัดการโดยมีหน้าที่ในการจัดการงานที่เกี่ยวข้องกับการปฏิสัมพันธ์กับผู้ใช้งานทั้งหมด นอกจากนั้นหน้าที่ในการส่งข้อมูลที่เก็บไว้ไปยังเซิร์ฟเวอร์นั้นเป็นหน้าที่ของตัวจัดการการสื่อสารฝั่งไคลเอนต์ (ClientCommunicationManager) โดยมีหน้าที่ในการแปลงรูปแบบข้อมูล และส่งข้อมูลไปยังเครื่องเซิร์ฟเวอร์นั่นเอง ซึ่งขั้นตอนในส่วนนี้เรียกว่าการประมวลผลข้อมูล (ProcessData) นอกจากนี้จากการพิจารณาการใช้งานจริงนั้นทุกๆ ครั้งที่มีข้อมูลใหม่มา ตัวจัดการแหล่งข้อมูลควรสามารถรับข้อมูลได้ทันที โดยไม่ต้องคอยจัดการข้อมูลที่รับมาก่อนหน้า ดังนั้นจึงเกิดเป็นคอนเซปต์ของการแฮนเดิลข้อมูล (HandleData) โดยที่ตัวจัดการแหล่งข้อมูลจะยกภาระในการประมวลผลข้อมูลที่ได้มา ให้กับตัวจัดการฝั่งไคลเอนต์ และเนื่องจากการเชื่อมต่อระหว่างเครื่องไคลเอนต์และเซิร์ฟเวอร์เป็นแบบไม่ถาวร ดังนั้น

เครื่องไคลเอนต์ควรมีความสามารถในการเก็บข้อมูลที่ได้รับมาแล้วจากผู้ใช้งานพักไว้ที่ที่เก็บข้อมูลเป็นการชั่วคราวก่อน ซึ่งข้อมูลที่พักไว้อาจถูกประมวลผลบางอย่าง (ProcessData) หรือไม่ก็ได้ก่อนที่จะอัปโหลดไปยังเครื่องเซิร์ฟเวอร์ โดยในงานวิจัยนี้ การประมวลผลข้อมูลที่ได้มาเป็นจะเป็นเพียงการบันทึกข้อมูล (SaveData) ลงที่เก็บข้อมูลทันทีเท่านั้น

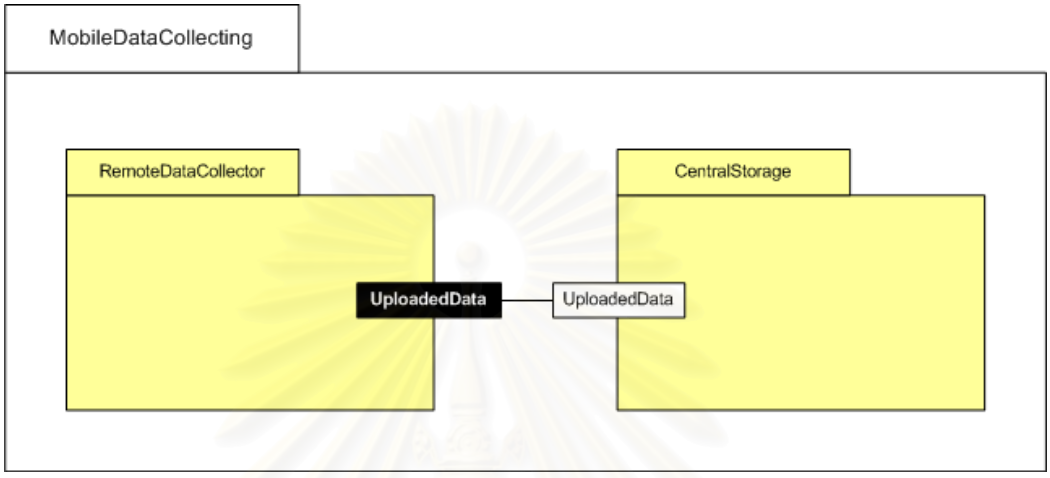
การทำงานของเครื่องเซิร์ฟเวอร์นั้นก็มีลักษณะเช่นเดียวกันกับฝั่งไคลเอนต์ กล่าวคือมีตัวจัดการฝั่งเซิร์ฟเวอร์หนึ่งตัว (ServerCentralManager) ทำหน้าที่ควบคุมลำดับการทำงานของเซิร์ฟเวอร์ โดยเซิร์ฟเวอร์มีหน้าที่รับการร้องขอ (Request) จากฝั่งไคลเอนต์ โดยมองว่าเครื่องไคลเอนต์ที่ติดต่อเข้ามานั้นก็เพื่อเรียกใช้บริการบันทึกข้อมูลของเครื่องเซิร์ฟเวอร์ ดังนั้นทุกการติดต่อที่ส่งมาจากไคลเอนต์จึงมองเป็นการร้องขอใช้บริการ ซึ่งตัวที่มีหน้าที่ในการแฮนเดิลการร้องขอ (HandleRequest) คือตัวจัดการการสื่อสารฝั่งเซิร์ฟเวอร์ (ServerCommunicationManager) ซึ่งจะส่งคำร้องขอที่ได้ไปให้ตัวจัดการฝั่งเซิร์ฟเวอร์ทำการดึงข้อมูล (ExtractData) เพื่อที่ตัวเองจะได้สามารถคอยรับการร้องขอถัดไปได้ในทันที ซึ่งเมื่อตัวจัดการฝั่งเซิร์ฟเวอร์ดึงข้อมูลสำเร็จแล้วจะบันทึกข้อมูลที่ได้ลงที่เก็บข้อมูลฝั่งเซิร์ฟเวอร์ ซึ่งงานส่วนนี้มอบหมายให้ตัวจัดการการเก็บข้อมูล (ServerStorageManager) เป็นผู้ดูแล ซึ่งในงานวิจัยนี้เลือกใช้การเก็บเป็นไฟล์อักขระ (Text File) โดยข้อมูลที่เก็บก็จะมีรูปแบบเป็นภาษาเอ็กซ์เอ็มแอลด้วยเช่นกัน และเพื่อเป็นการยืนยันถึงการรับข้อมูลเข้าจากไคลเอนต์ ทุกๆ ครั้งที่มีการรับคำร้องขอจะมีการจดเวลาที่คำร้องขอนั้นมาถึงเซิร์ฟเวอร์ พร้อมทั้งบันทึกเวลาดังกล่าวลงในไฟล์เอ็กซ์เอ็มแอลพร้อมกับข้อมูลเรคคอร์ดนั้นๆ อีกด้วย

4.3 การพัฒนาและสร้างแบบจำลองพีไอเอ็มของระบบตัวอย่าง

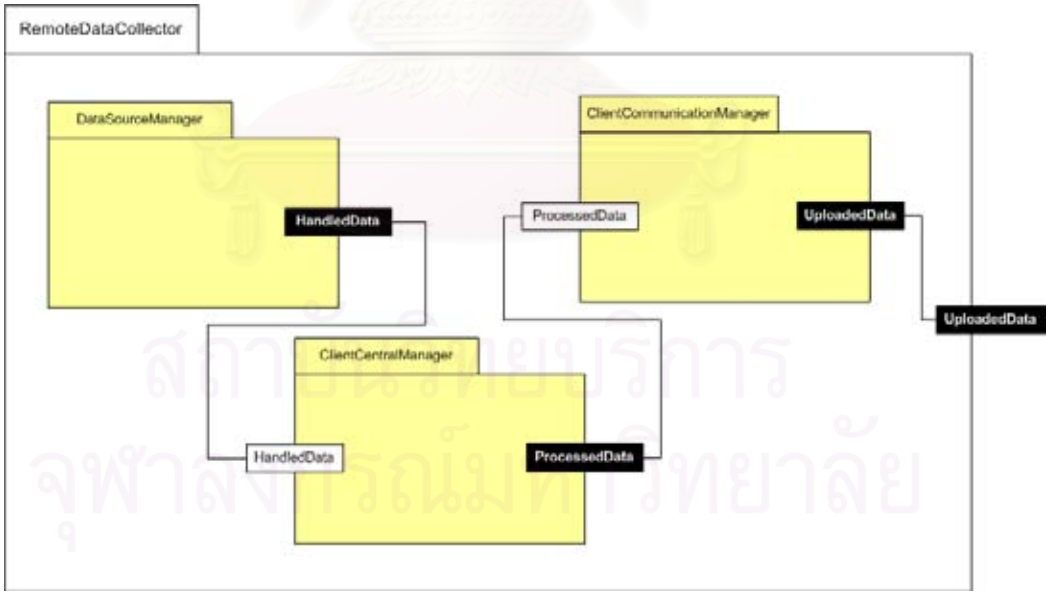
จากที่กล่าวถึงหลักการสร้างแบบจำลองพีไอเอ็มในหัวข้อที่ 3.2 แบบจำลองพีไอเอ็มต้องสามารถอธิบายถึง หน้าที่ และพฤติกรรมของระบบได้ แต่ในขณะเดียวกัน แบบจำลองต้องยืดหยุ่นเพียงพอและไม่ยึดติดกับแพลตฟอร์มอันใดอันหนึ่ง นอกจากนั้นเพื่อให้แบบจำลองมีคุณภาพที่ดีการใช้ภาษาโมเดลมาตรฐานจึงเป็นสิ่งที่ควรคำนึงถึงในการออกแบบ

ระบบตัวอย่างเป็นการส่งข้อมูลระหว่างไคลเอนต์ที่เป็นเครื่องปาล์ม ไปเก็บยังเครื่องเซิร์ฟเวอร์ที่เป็นเครื่องพีซี ซึ่งลักษณะระบบเช่นนี้สามารถมองเป็นระบบการประมวลผลแบบกระจายได้ ดังนั้นในงานวิจัยชิ้นนี้จึงได้เลือกภาษายูเอ็มแอล (หัวข้อที่ 2.1.1) และยูเอ็มแอลโปรไฟล์สำหรับอีด็อก (หัวข้อที่ 2.1.5) ในการสร้างแบบจำลองพีไอเอ็ม

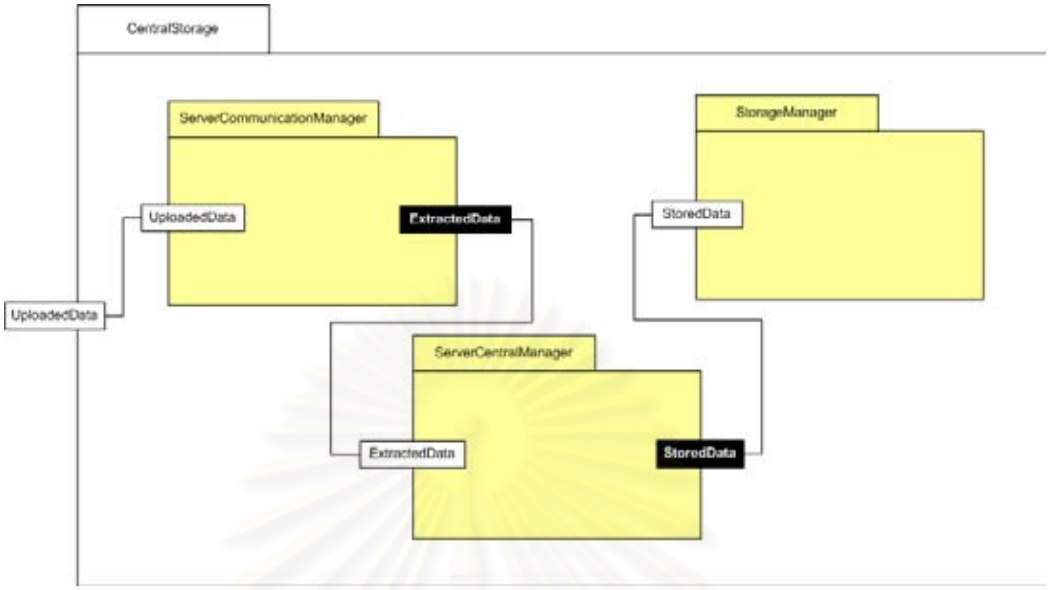
4.3.1 แบบจำลองเชิงธุรกิจระดับสูง (High Level Business Model)



รูปที่ 4.2 แบบจำลองเชิงธุรกิจระดับสูงของระบบตัวอย่างแสดงด้วยภาษาซีซีเอ



รูปที่ 4.3 ส่วนประกอบของตัวรวบรวมข้อมูลระยะไกลแสดงด้วยภาษาซีซีเอ



รูปที่ 4.4 ส่วนประกอบของตัวเก็บข้อมูลส่วนกลางแสดงด้วยภาษาซีซีเอ

แบบจำลองเชิงธุรกิจระดับสูงของระบบตัวอย่างที่แสดงด้วยภาษาซีซีเอดังรูปที่ 4.2 เป็นการมองระดับบนสุดโดยแยกระบบออกเป็นสองส่วนย่อยได้แก่ ตัวรวบรวมข้อมูลระยะไกล (RemoteDataCollector) และ ตัวเก็บข้อมูลส่วนกลาง (CentralStorage) โดยมีการส่งข้อมูล (Data) ซึ่งมีการตกลงข้อกำหนดการเชื่อมต่อไว้ล่วงหน้าซึ่งจะได้กล่าวในรายละเอียดต่อไป

รูปที่ 4.3 แสดงส่วนประกอบของตัวรวบรวมข้อมูลระยะไกล ซึ่งประกอบไปด้วยตัวจัดการแหล่งข้อมูล (DataSourceManager) มีหน้าที่ในการควบคุมการติดต่อกับแหล่งข้อมูลและดึงข้อมูลที่ระบบสนใจจากแหล่งข้อมูล ตัวจัดการฝั่งไคลเอนต์ (ClientCentralManager) จะเก็บตรรกะเกี่ยวกับการประมวลผลข้อมูลที่ได้มาและควบคุมลำดับการทำงานของไคลเอนต์ ส่วนตัวจัดการการสื่อสารฝั่งไคลเอนต์ (ClientCommunicationManager) ทำหน้าที่ในการส่งข้อมูลที่ประมวลผลแล้วไปยังเครื่องเซิร์ฟเวอร์ จะสังเกตได้ว่าการเชื่อมต่อระหว่างโปรเซสคอมโพเนนท์จะทำผ่านพอร์ต (Port) เสมอ ซึ่งแต่ละพอร์ตจะแสดงข้อมูลรูปแบบต่างๆ ที่ส่งผ่านมา

รูปที่ 4.4 แสดงส่วนประกอบของตัวรวบรวมเก็บข้อมูลส่วนกลาง ซึ่งประกอบไปด้วยตัวจัดการการสื่อสารฝั่งเซิร์ฟเวอร์ (ServerCommunicationManager) ทำหน้าที่ในการรับข้อมูลจากเครื่องไคลเอนต์ซึ่งมีได้มากกว่าหนึ่งเครื่อง ตัวจัดการฝั่งเซิร์ฟเวอร์ (ServerCentralManager) จะเก็บตรรกะเกี่ยวกับการประมวลผลข้อมูลที่รับมาจากเครื่องลูกข่าย และตัวจัดการการเก็บข้อมูล (StorageManager) ทำหน้าที่ในการเก็บข้อมูลลงที่เก็บข้อมูล

4.3.2 ความยืดหยุ่นของแบบจำลอง

เมื่อพิจารณาแบบจำลองที่ได้ข้างต้น อาจมีคำถามว่าเป็นแบบจำลองที่ไม่ให้ภาพที่ชัดเจนของระบบเลย ไม่สามารถบอกได้ว่าระบบที่เสร็จสมบูรณ์แล้วจะหน้าตาเป็นอย่างไร แล้วจะเป็นแบบจำลองที่สมบูรณ์ได้อย่างไร คำตอบคือ ถูกต้องแล้วที่เมื่อพิจารณาแบบจำลองข้างต้นแล้วยังไม่สามารถเห็นภาพของระบบที่เสร็จสมบูรณ์ได้ เนื่องจากแบบจำลองดังกล่าวเป็นแบบจำลองพีไอเอ็มจึงยังไม่มีรายละเอียดการอิมพลีเมนต์ใดๆ อยู่เลย ทำให้ไม่สามารถเห็นภาพของระบบที่เสร็จสมบูรณ์ได้ แต่ในทางกลับกันแบบจำลองดังกล่าวก็สามารถนำไปอิมพลีเมนต์ในรูปแบบที่แตกต่างกันหลายรูปแบบได้เช่นกัน

จากที่ได้เคยกล่าวในข้างต้นว่า แบบจำลองพีไอเอ็มต้องสามารถอธิบายถึง หน้าที่ และพฤติกรรมของระบบได้ แต่ในขณะเดียวกัน แบบจำลองต้องยืดหยุ่นเพียงพอและไม่ยึดติดกับแพลตฟอร์มอันใดอันหนึ่ง ในที่นี้จะขอยกตัวอย่างระบบอื่นๆ 2 ระบบเพื่อให้เข้าใจถึงหลักการนี้มากขึ้น โดยระบบเหล่านี้สามารถนำแบบจำลองซีซีเอของตัวรวบรวมข้อมูลระยะไกลไปใช้ได้

1. ระบบเครื่องตรวจวัดอุณหภูมิ

ระบบจะเลือกอ่านเฉพาะค่าอุณหภูมิผ่านตัวรับรู้ (Sensor) (อิมพลีเมนต์ DataSourceManager) โดยไม่สนใจข้อมูลอื่นเช่นแสงหรือเสียงในสภาวะแวดล้อม โดยตัวควบคุมจะสั่งให้อ่านค่า 30 ครั้งต่อวินาที แล้วทำการหาค่าเฉลี่ยภายใน 1 วินาทีนั้น (ตัวควบคุมอิมพลีเมนต์ ClientCentralManager) จากนั้นตัวควบคุมจะส่งค่าเฉลี่ยอุณหภูมิไปยังวงจรส่งข้อมูลซึ่งจะแปลงข้อมูลดังกล่าวจากแบบขนาน เป็นแบบอนุกรมของบิต และใช้มาตรฐานการส่งแบบ RS-478 ซึ่งใช้สายไฟเพียง 2 เส้น (วงจรส่งข้อมูลอิมพลีเมนต์ ClientCommunicationManager) ส่วนในฝั่งเซิร์ฟเวอร์ ก็จะมีลักษณะของการอิมพลีเมนต์แบบจำลองของตัวเก็บข้อมูลส่วนกลางในทำนองเดียวกัน

2. ระบบการจัดทำผลสำรวจประชามติ

พนักงานเก็บข้อมูล (อิมพลีเมนต์ RemoteDataCollector) ออกสำรวจประชามติโดยแต่ละคนจะมีแบบสอบถามติดตัวไปด้วย พนักงานเก็บข้อมูลอาจอ่านแบบสอบถามให้ฟังหรือให้ผู้ให้ข้อมูลอ่านเอง (แบบสอบถามอิมพลีเมนต์ DataSourceManager) เมื่อได้แบบสอบถามที่กรอกข้อมูลมาแล้วก็จะทำการสรุปข้อมูลเช่น ทำการจัดเรียงแบบสอบถามตามอายุของผู้ให้ข้อมูล โดยแบ่งแยกเพศด้วย (อิมพลีเมนต์ ClientCentralManager) จากนั้นจะโทรศัพท์ไปที่ศูนย์รวบรวม

ข้อมูลกลางเพื่อแจ้งให้คนรับเอกสารมารับแบบสอบถามดังกล่าวไป (อิมพลีเมนต์ ClientCommunicationManager)

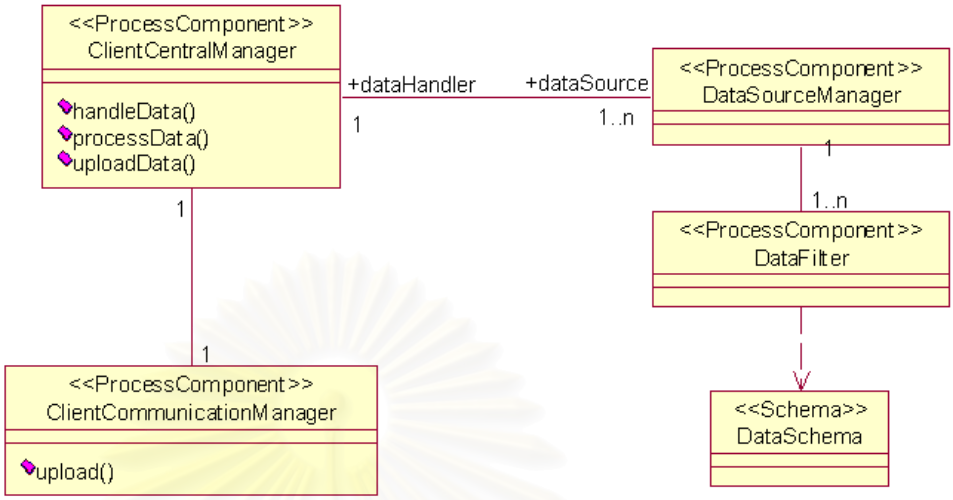
จากตัวอย่างจะสังเกตว่าระบบทั้งสองข้างต้นนี้เป็นระบบที่ต่างกันอย่างสิ้นเชิงแต่ทั้งคู่กลับถูกอธิบายได้ด้วยแบบจำลองเดียวกัน ทั้งนี้เนื่องจากแบบจำลองดังกล่าวไม่ได้ระบุเจาะจงรายละเอียดของการอิมพลีเมนต์ บอกแต่เพียงตรรกะทางธุรกิจ เท่านั้น ซึ่งเป็นไปตามคอนเซปต์ของแบบจำลองพีไอเอ็มนั่นเอง

4.3.3 แบบจำลองพีไอเอ็มของตัวรวบรวมข้อมูลระยะไกล

เอ็มดีเอให้แนวคิดของการพัฒนาระบบ โดยใช้แบบจำลองเป็นตัวขับเคลื่อนโดยสามารถสร้าง เปลี่ยนแปลง แก้ไข และเชื่อมต่อระบบในระดับแบบจำลองซึ่งจะส่งผลต่อไปยังระดับปฏิบัติงานได้ แต่เงื่อนไขที่สำคัญอย่างหนึ่งคือแบบจำลองต้องมีความสมบูรณ์เพียงพอ ความสมบูรณ์ของแบบจำลองในที่นี้หมายถึง แบบจำลองต้องมีความถูกต้อง มีรายละเอียดของระบบที่ครบถ้วนเพียงพอที่จะนำไปอิมพลีเมนต์ได้ รวมทั้งต้องเขียนด้วยภาษาโมเดลที่เป็นมาตรฐานด้วย อย่างไรก็ตาม ในปัจจุบันยังไม่มีวิธีการมาตรฐานใดในการตรวจสอบความสมบูรณ์ของแบบจำลอง

ดังที่ได้กล่าวในหัวข้อที่ 4.3.2 ว่าแบบจำลองที่ได้ในหัวข้อที่ 4.3.1 ยังไม่มีความสมบูรณ์เพียงพอเนื่องจากขาดรายละเอียดในการอิมพลีเมนต์ ดังนั้นจึงต้องเพิ่มรายละเอียดให้แบบจำลองเพื่อให้แบบจำลองมีความสมบูรณ์มากยิ่งขึ้น อย่างไรก็ตามภาษาซีซีเอมีข้อจำกัดในการอธิบายแบบจำลองเนื่องจากไม่สามารถบอกรายละเอียดเชิงข้อกำหนด (Specification Details) ของแต่ละโปรเซสคอมโพเนนท์ได้ โดยจะบอกได้แต่เพียงส่วนประกอบ (Composition) เท่านั้น ดังนั้นจึงต้องใช้ภาษาโมเดลอื่นในการอธิบายแบบจำลองในระดับที่มีรายละเอียดมากขึ้น

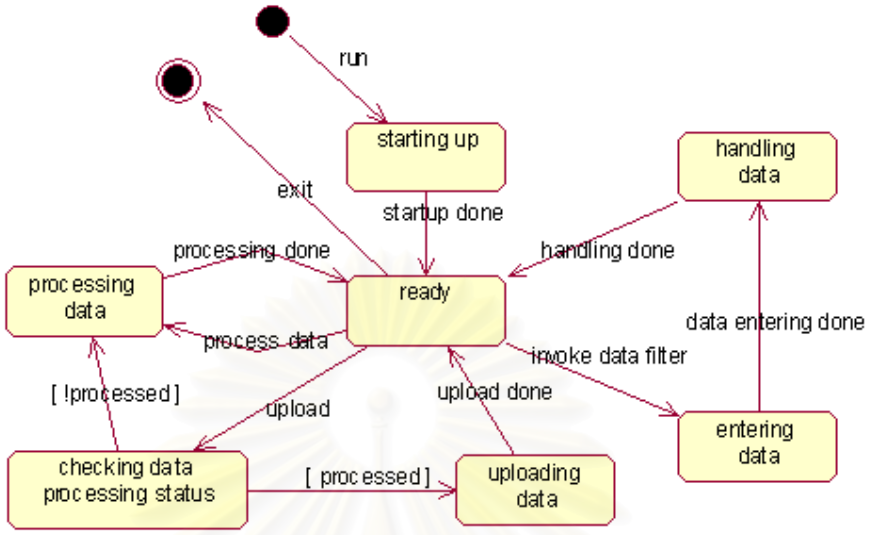
เนื่องจากในงานวิจัยนี้เลือกที่จะอิมพลีเมนต์ระบบในขั้นตอนสุดท้ายด้วยซอฟต์แวร์ ดังนั้นภาษาโมเดลที่น่าจะเหมาะสมที่สุดคือภาษายูเอ็มแอล ซึ่งเป็นภาษาที่นิยมใช้ในการออกแบบระบบซอฟต์แวร์เชิงวัตถุโดยทั่วไป โดยจากแบบจำลองที่เขียนด้วยภาษาซีซีเอในหัวข้อที่ 4.3.1 สามารถเขียนในรูปแบบของยูเอ็มแอลโปรไฟล์ได้โดยอาศัยกฎการแปลง (Mapping Rules) ระหว่างภาษาซีซีเอ ไปยังยูเอ็มแอลโปรไฟล์ (ดูหัวข้อที่ 3.2.2.1)



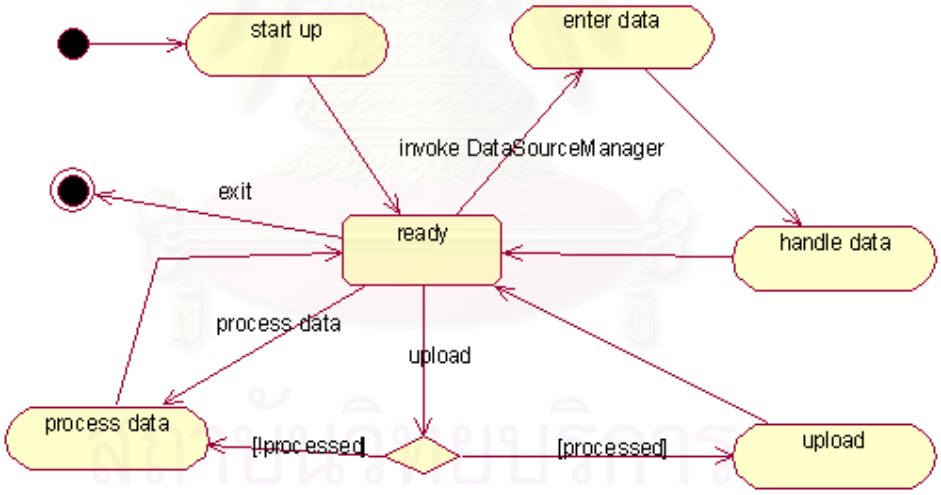
รูปที่ 4.5 แบบจำลองตัวรวบรวมข้อมูลระยะไกลแสดงด้วยยูเอ็มแอลโปรไฟล์สำหรับอีด็อก

แบบจำลองรูปที่ 4.5 เกิดจากการแปลงแบบจำลองในรูปที่ 4.3 ด้วยกฎการแปลงที่ได้กล่าวถึงข้างต้น ซึ่งจะทำให้ได้โปรเซสคอมโพเนนท์ที่เขียนในลักษณะของแผนภาพคลาส ส่วนโอเปอร์เรชันในโปรเซสคอมโพเนนท์ เกิดจากการแปลงโอเปอร์เรชันพอร์ตไปเป็นโอเปอร์เรชันในคลาส โดยต้องกำหนดชื่อโอเปอร์เรชันด้วยตัวเองภายหลัง เพราะไม่สามารถทราบชื่อเหล่านี้จากกฎการแปลงได้ จากนั้นจึงได้เพิ่มรายละเอียดให้กับการทำงานของตัวจัดการแหล่งข้อมูลโดยการเพิ่มคอนเซปต์ของตัวกรองข้อมูล (DataFilter) ซึ่งไม่ปรากฏในแบบจำลองก่อนหน้า โดยแต่ละตัวกรองจะทำการคัดข้อมูลเฉพาะที่ระบบต้องการจากแหล่งข้อมูล (ข้อมูลที่ต้องการกำหนดไว้ในสกีมาข้อมูล (DataSchema)) แล้วส่งข้อมูลไปให้ตัวจัดการแหล่งข้อมูล

หลังจากได้แบบจำลองเชิงโครงสร้างของตัวรวบรวมข้อมูลระยะไกลแล้ว ขั้นตอนต่อไปเป็นการเพิ่มรายละเอียดของแบบจำลองเชิงพฤติกรรมของระบบ (สร้างเพิ่มขึ้นทั้งหมดเพราะไม่มีรายละเอียดเช่นนี้อยู่ในแบบจำลองก่อนหน้า) โดยในที่นี้จะมองตัวรวบรวมข้อมูลเป็นไฟไนท์สเตทแมชชีนหนึ่งตัว ดังนั้นจึงเลือกที่จะโมเดลด้วยแผนภาพสเตทชาร์ต ดังรูปที่ 4.6 แต่จากที่กล่าวไว้ในหัวข้อที่ 3.4.3 ว่าในงานวิจัยนี้เลือกจะแสดงพฤติกรรมของระบบด้วยแผนภาพแอคทิวิตี และแผนภาพซีเควนซ์ ดังนั้นหากเขียนแบบจำลองเชิงพฤติกรรมของตัวรวบรวมข้อมูลระยะไกลด้วยแผนภาพแอคทิวิตีแทนแผนภาพสเตทชาร์ตจะได้ดังรูปที่ 4.7

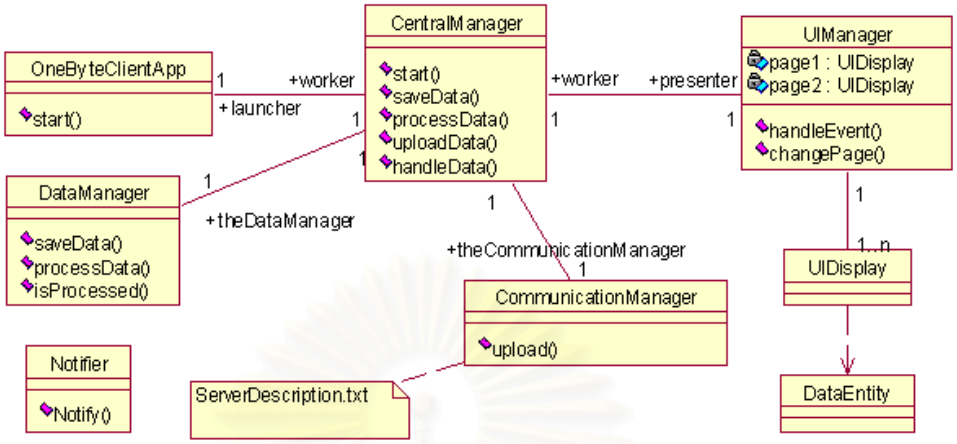


รูปที่ 4.6 แบบจำลองเชิงพฤติกรรมของตัวรวบรวมข้อมูลระยะไกลแสดงด้วยแผนภาพสเตทชาร์ต



รูปที่ 4.7 แบบจำลองเชิงพฤติกรรมของตัวรวบรวมข้อมูลระยะไกลแสดงด้วยแผนภาพแอคทิวิตี

แบบจำลองในรูปที่ 4.5 นั้นยังไม่ได้เป็นแบบจำลองของระบบซอฟต์แวร์เนื่องจากยังคงแสดงในรูปของโปรเซสคอมโพเนนท์อยู่ ซึ่งในโลกของซอฟต์แวร์ไม่มีคอนเซปต์ตรงของโปรเซสคอมโพเนนท์ ดังนั้นจึงต้องเพิ่มรายละเอียดให้เป็นแบบจำลองของระบบซอฟต์แวร์เชิงวัตถุ (Object Oriented Software) โดยจะได้แบบจำลองดังรูปที่ 4.8



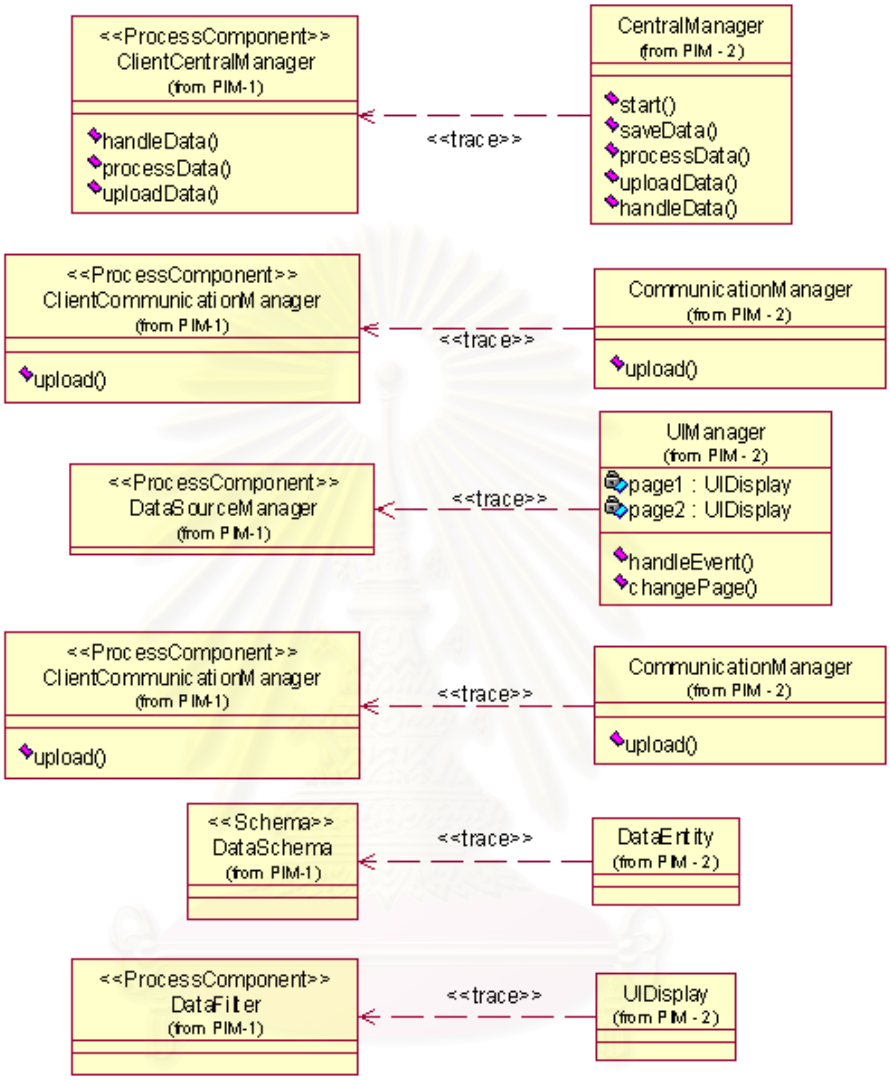
รูปที่ 4.8 แบบจำลองของตัวรวบรวมข้อมูลระยะไกลที่เพิ่มรายละเอียดเชิงเทคนิคของระบบซอฟต์แวร์

แบบจำลองที่ได้ในรูปที่ 4.8 นี้มีรายละเอียดเชิงเทคนิคของระบบซอฟต์แวร์เนื่องจากโครงสร้างโมเดลถูกกำหนดด้วยคลาส ซึ่งเป็นโครงสร้างพื้นฐานของโมเดลระบบซอฟต์แวร์ อย่างไรก็ตามแบบจำลองนี้ยังถือเป็นแบบจำลองพีไอเอ็มเนื่องจากไม่บอกรายละเอียดการอิมพลีเมนต์ (ยังไม่สามารถบอกได้ว่าจะนำแบบจำลองนี้ไปอิมพลีเมนต์อย่างไร) การแปลงแบบจำลองในรูปที่ 4.5 มาเป็นรูปที่ 4.8 เป็นการเชื่อมโยงแบบจำลองเชิงโครงสร้างระหว่างสองระดับนามธรรม (Level of Abstraction) ซึ่งโดยส่วนใหญ่จะเป็นการจับคู่ระหว่างโปรเซสคอมโพเนนท์ไปเป็นคลาสโดยตรง ส่วนสกีมาข้อมูลจะถูกแปลงเป็นเอนทิตีคลาส ความเชื่อมโยงระหว่างแบบจำลองสองระดับนี้แสดงในรูปที่ 4.9

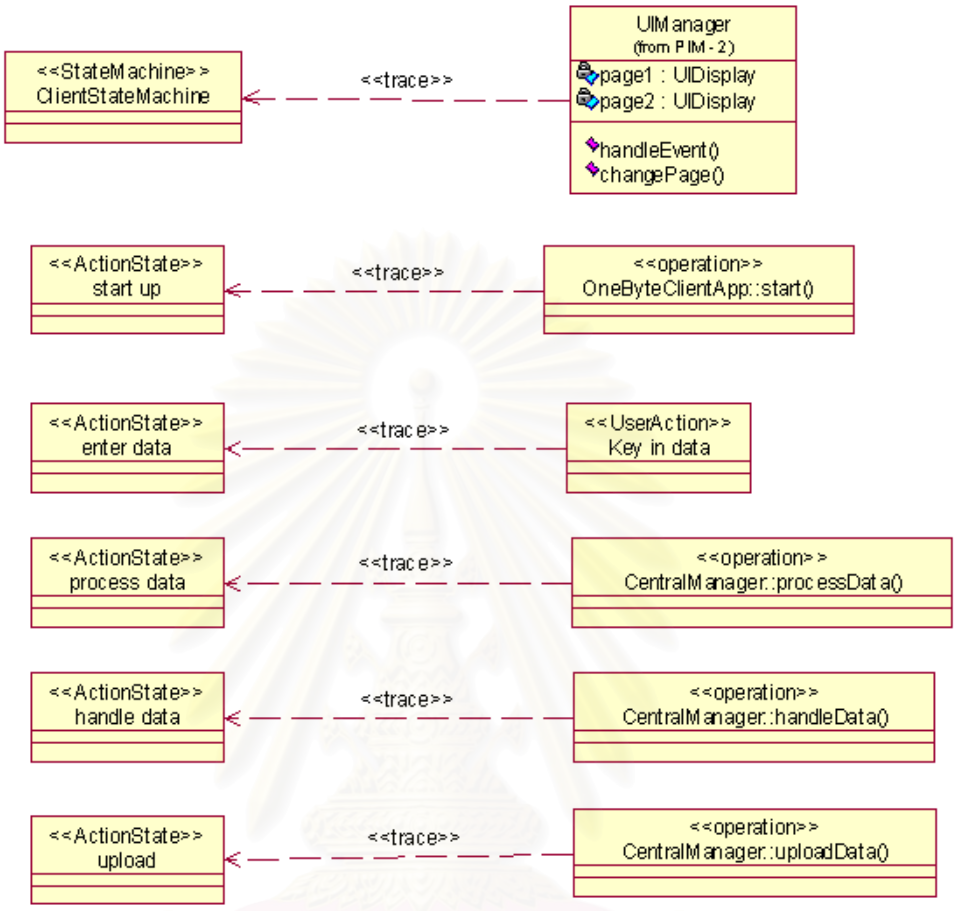
จะสังเกตได้ว่ารูปที่ 4.5 กับรูปที่ 4.8 มีความแตกต่างกันบางประการได้แก่ มีคลาสบางคลาสเพิ่มขึ้นมาเช่น OneByteClientApp ซึ่งเป็นคลาสที่ทำหน้าควบคุมวงจรชีวิตของแอพพลิเคชันและติดต่อกับระบบปฏิบัติการ ส่วนคลาส DataManager ทำหน้าที่ในการจัดการเก็บข้อมูลลงที่เก็บข้อมูลของเครื่องคอมพิวเตอร์มือถือ ซึ่งงานส่วนนี้เป็นงานที่ได้รับมอบหมาย (Delegation) มาจากคลาส CentralManager ส่วนคลาส Notifier ทำหน้าที่แสดงการแจ้งเตือนความผิดพลาดต่างๆ ของระบบ นอกจากนั้นยังมีคลาสบางคลาสซึ่งเป็นการอิมพลีเมนต์คลาสเดิมในแบบจำลองรูปที่ 4.5 ยกตัวอย่างเช่น UIManager เป็นการอิมพลีเมนต์ DataSourceManager (โดยจัดการอีเวนต์ที่ได้รับจากผู้ใช้ระบบ (handleEvent()) และควบคุมการแสดงหน้าจอของระบบ changePage()) คลาส UIDisplay เป็นการอิมพลีเมนต์คลาส DataFilter ส่วนคลาส DataEntity เป็นการอิมพลี

เมนต์คลาส DataSchema ในขณะที่บางคลาสนั้นเป็นเพียงการเปลี่ยนชื่อเพื่อความเหมาะสมเท่านั้น ยกตัวอย่างเช่น CentralManager มาจาก ClientCentralManager และ CommunicationManager มาจาก ClientCommunicationManager แต่เนื่องจากความกระชับในการเขียนโค้ดจึงตัดคำว่า “Client” ทิ้งไปเพราะได้ระบุในชื่อแพคเกจอยู่แล้ว

การจะกล่าวว่าแบบจำลองที่ได้นั้นมีความสมบูรณ์หรือไม่นั้น มักเป็นเรื่องยากในการตัดสิน เนื่องจากในทางปฏิบัติมักมีระดับของความละเอียดของแบบจำลองที่ยอมรับได้ในหลายระดับ วิธีการตรวจสอบแบบไม่เป็นทางการที่ใช้ในงานวิจัยนี้คือ การทดลองตั้งคำถามกับแบบจำลองที่ได้ว่าทำให้เข้าใจการทำงานของระบบได้มากพอหรือไม่ หากไม่มากพอแสดงว่าแบบจำลองนั้นต้องการการอธิบายเพิ่มเติม

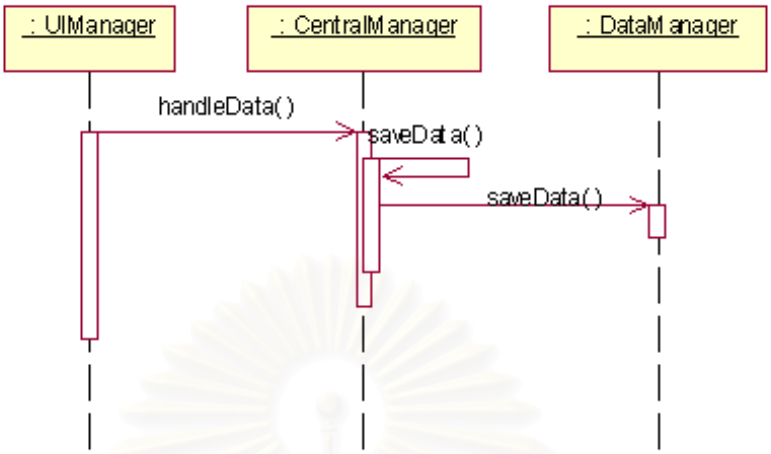


รูปที่ 4.9 การเชื่อมโยงของแบบจำลองเชิงโครงสร้างระหว่างสองระดับ

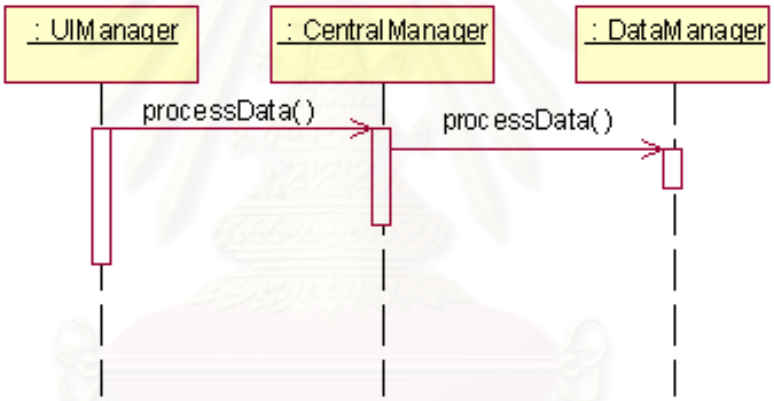


รูปที่ 4.10 ส่วนหนึ่งของการเชื่อมโยงแบบจำลองเชิงโครงสร้างกับแบบจำลองเชิงพฤติกรรม

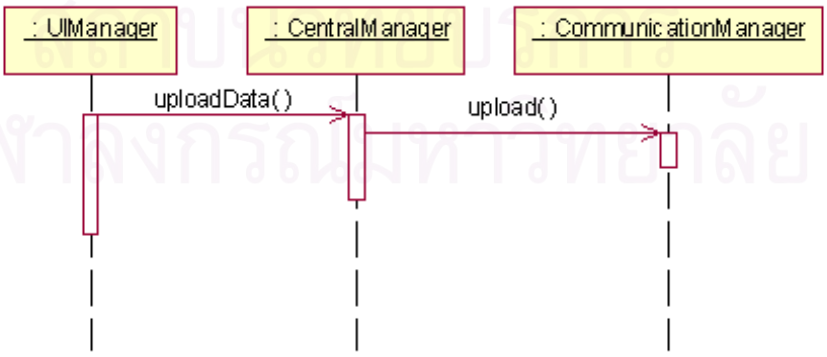
รูปที่ 4.10 แสดงตัวอย่างการเชื่อมโยงแบบจำลองเชิงโครงสร้าง (รูปที่ 4.8) กับแบบจำลองเชิงพฤติกรรม (รูปที่ 4.7) โดยการแยกพิจารณาส่วนของแบบจำลองเชิงพฤติกรรมทีละส่วน จากนั้นจึงกำหนดคอมโพเนนท์ที่อิมพลีเมนต์ส่วนของแบบจำลองพฤติกรรมนั้น ยกตัวอย่างเช่นสเตทแมชชีน ในรูปที่ 4.6 หรือ 4.7 จะถูกอิมพลีเมนต์ด้วยคลาส UIManager ทั้งนี้เนื่องจากในทางปฏิบัตินั้นอีเวนต์ทั้งหมดเป็นอีเวนต์ที่เกิดจากผู้ใช้งานเป็นหลัก ดังนั้นการเลือกคลาสที่ทำหน้าที่ควบคุมส่วนติดต่อผู้ใช้งานมาอิมพลีเมนต์สเตทแมชชีนนี้ จึงน่าจะเป็นทางเลือกที่สมเหตุสมผล ส่วน ActionState ต่างๆ นั้นจะถูกอิมพลีเมนต์ด้วยโอเปอร์เรชันของของคลาสต่างๆ ดังแสดงด้วยแผนภาพซีเควนซ์ในรูปที่ 4.11 ถึง 4.13 ส่วน enter data ซึ่งอิมพลีเมนต์ด้วยการกระทำของผู้ใช้งานนั้น ในที่นี้แสดงด้วยสัญลักษณ์ <<UserAction>>



รูปที่ 4.11 แบบจำลองพีไอเอ็มเชิงพฤติกรรมของกิจกรรม handleData



รูปที่ 4.12 แบบจำลองพีไอเอ็มเชิงพฤติกรรมของกิจกรรม processData

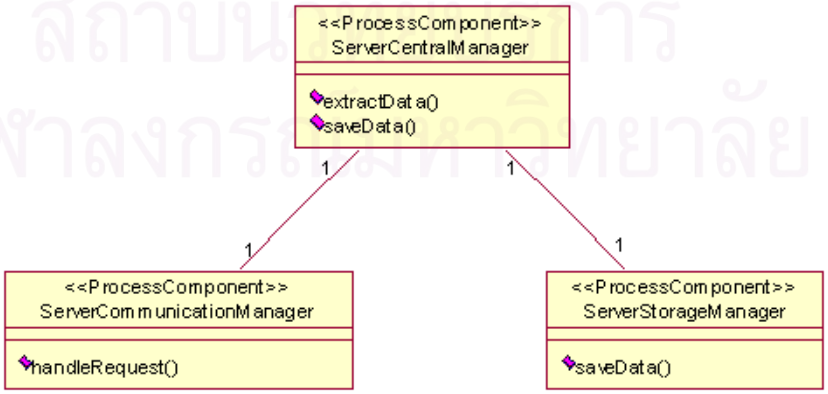


รูปที่ 4.13 แบบจำลองพีไอเอ็มเชิงพฤติกรรมของกิจกรรม uploadData

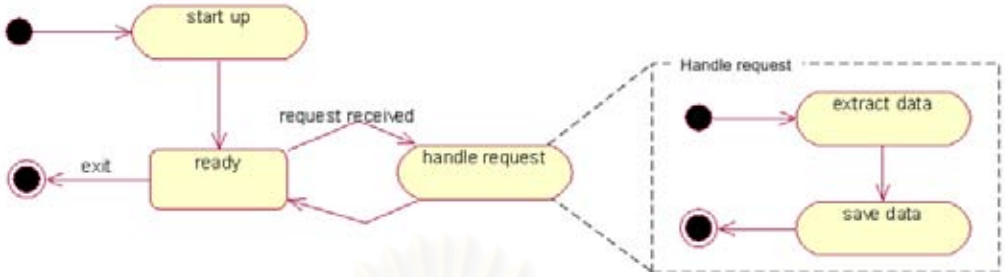
4.3.4 แบบจำลองพีไอเอ็มของตัวเก็บข้อมูลส่วนกลาง

ในทำนองเดียวกัน จากแบบจำลองตัวเก็บข้อมูลส่วนกลางในรูปที่ 4.4 เมื่อเปลี่ยนภาษาโมเดลเป็นภาษายูเอ็มแอลจะได้เป็นแบบจำลองในรูปที่ 4.14 จากนั้นจึงออกแบบแบบจำลองเชิงพฤติกรรมของตัวเก็บข้อมูลส่วนกลางดังแสดงในรูปที่ 4.15 แล้วจึงเพิ่มรายละเอียดเชิงเทคนิคโดยการกำหนดให้เป็นระบบซอฟต์แวร์ ทำให้ได้แบบจำลองดังรูปที่ 4.16 โดยได้เพิ่มเติมคลาสบางคลาสได้แก่ OneByteServerApp ทำหน้าที่คล้ายกับ OneByteClientApp ของฝั่งไคลเอนต์ กล่าวคือ ทำหน้าที่คอยควบคุมวงจรชีวิตของแอพพลิเคชันฝั่งเซิร์ฟเวอร์ และติดต่อกับระบบปฏิบัติการ คลาส DocumentManager เป็นคลาสที่ทำหน้าที่จัดการประมวลผลเอกสารเอ็กซ์เอ็มแอลซึ่งในแบบจำลองก่อนหน้าเป็นหน้าที่ของตัวจัดการฝั่งเซิร์ฟเวอร์นั่นเอง นอกจากนั้นยังมีการเปลี่ยนแปลงชื่อคลาสบางคลาสได้แก่ CentralManager เป็นการอิมพลีเมนต์คลาส ServerCentralManager ส่วนคลาส CommunicationManager เป็นการอิมพลีเมนต์คลาส ServerCommunicationManager และคลาส StorageManager เป็นการอิมพลีเมนต์คลาส ServerStorageManager ซึ่งสาเหตุที่ตัดคำว่า Server ออกนั้นเนื่องจากชื่อแพคเกจสามารถระบุขอบเขตของระบบได้ชัดเจนอยู่แล้ว จึงควรตัดทิ้งเพื่อประโยชน์ในการเขียนโค้ดในภายหลัง ยกตัวอย่างการมอบหมายงาน (Delegation) เช่นรูปที่ 4.17 ซึ่งเป็นแบบจำลองพีไอเอ็มของกิจกรรม saveData ของฝั่งเซิร์ฟเวอร์โดยแสดงด้วยแผนภาพซีเควนซ์ ส่วนกิจกรรม extractData ไม่มีการมอบหมายงานเนื่องจากเมธอด extractData() ของ CentralManager เป็นผู้จัดการเอง

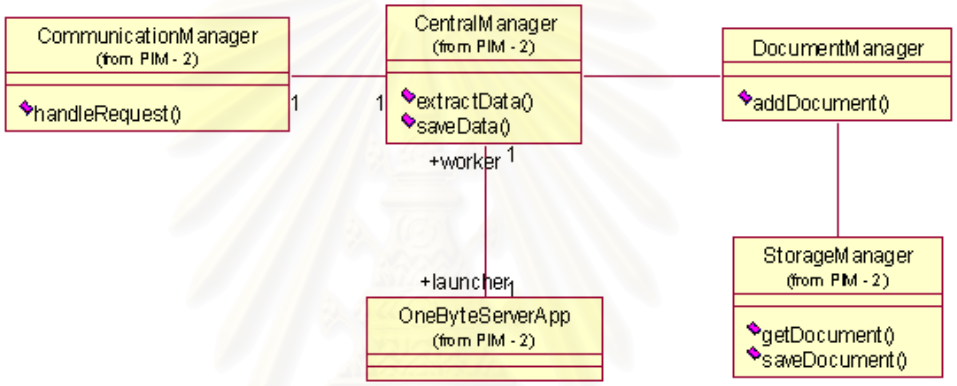
การเชื่อมโยงแบบจำลองเชิงโครงสร้างทั้ง 2 ระดับ (รูปที่ 4.14 และรูปที่ 4.16) แสดงไว้ในรูปที่ 4.18 และการเชื่อมโยงแบบจำลองเชิงโครงสร้างกับแบบจำลองเชิงพฤติกรรม (รูปที่ 4.15 และรูปที่ 4.16) แสดงไว้ในรูปที่ 4.19



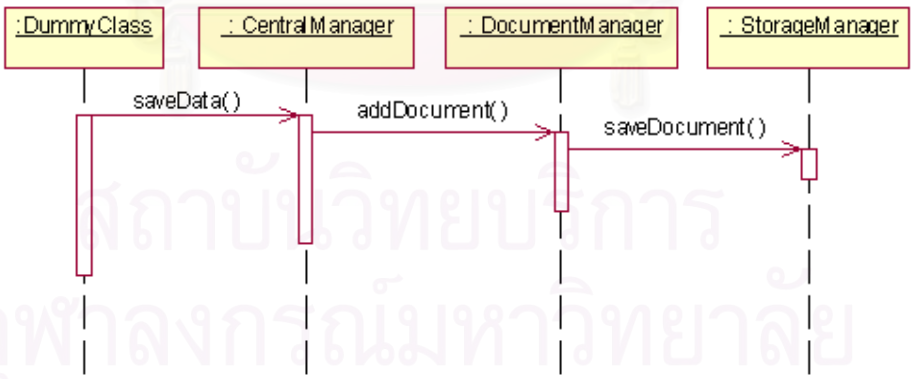
รูปที่ 4.14 แบบจำลองพีไอเอ็มตัวเก็บข้อมูลส่วนกลางแสดงด้วยยูเอ็มแอลโปรไฟล์สำหรับอีด็อก



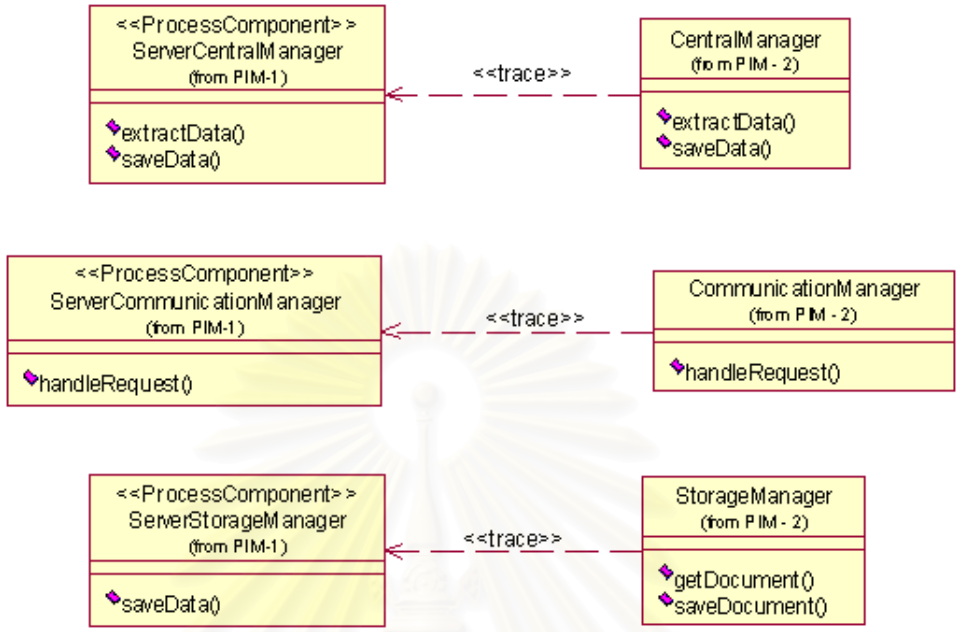
รูปที่ 4.15 แบบจำลองเชิงพฤติกรรมของตัวเก็บข้อมูลส่วนกลาง



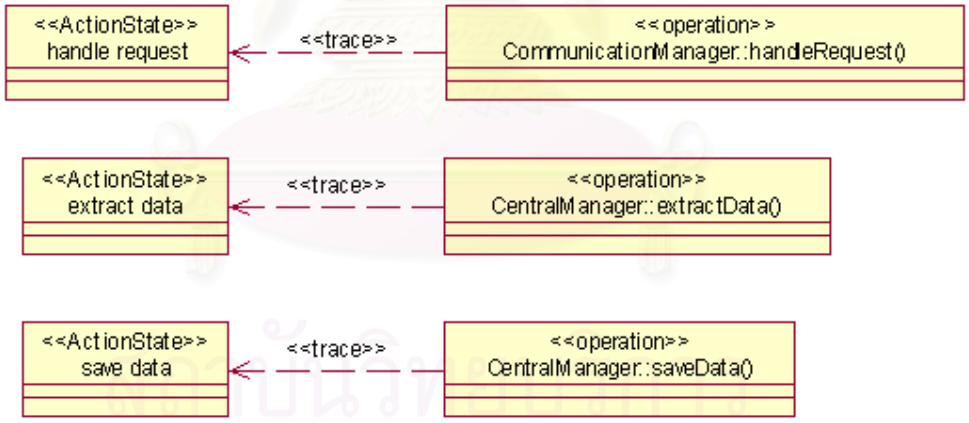
รูปที่ 4.16 แบบจำลองเชิงโครงสร้างของตัวเก็บข้อมูลส่วนกลางที่เพิ่มรายละเอียดเชิงเทคนิค



รูปที่ 4.17 แบบจำลองพีไอเอ็มเชิงพฤติกรรมของกิจกรรม saveData



รูปที่ 4.18 การเชื่อมโยงระหว่างแบบจำลองเชิงโครงสร้างทั้ง 2 ระดับ



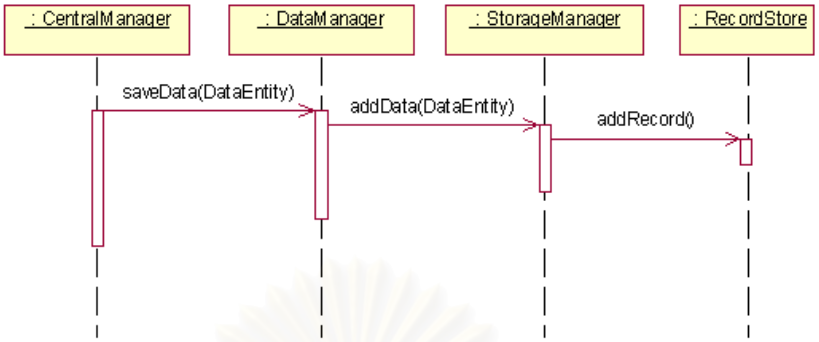
รูปที่ 4.19 การเชื่อมโยงแบบจำลองเชิงโครงสร้างกับแบบจำลองเชิงพฤติกรรม

4.4 การสร้างแบบจำลองพีเอสเอ็มของระบบตัวอย่าง

4.4.1 แบบจำลองพีเอสเอ็มของตัวรวบรวมข้อมูลระยะไกล

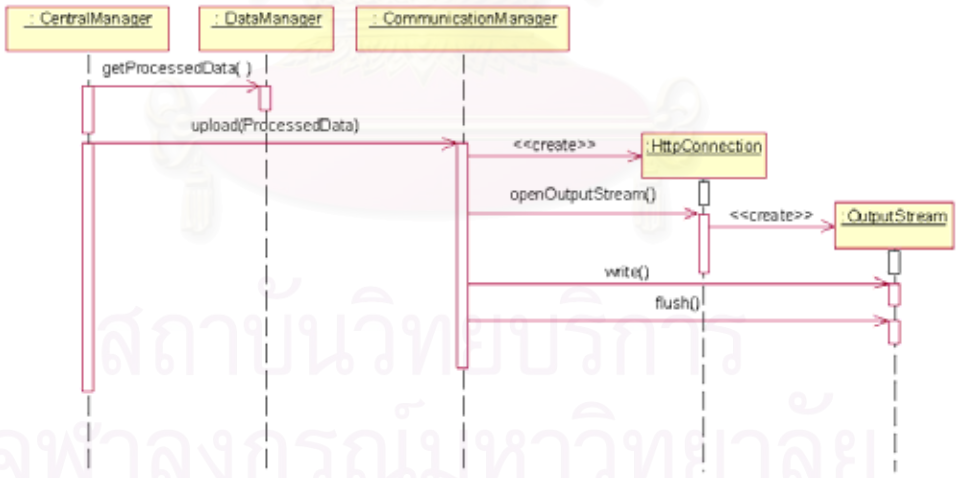
จากหลักการแปลงแบบจำลองพีไอเอ็มไปเป็นแบบจำลองพีเอสเอ็ม ที่ได้กล่าวถึงในหัวข้อที่ 3.3.2 ดังนั้นแบบจำลองพีเอสเอ็มเกิดจากการนำแบบจำลองพีไอเอ็มของตัวรวบรวมข้อมูลระยะไกลที่ได้ในหัวข้อที่ 4.3.3 มาเพิ่มรายละเอียดเชิงเทคนิค และเชิงวิศวกรรมในการอิมพลีเมนต์ ซึ่งในงานวิจัยชิ้นนี้ได้เลือกที่จะอิมพลีเมนต์ตัวรวบรวมข้อมูลระยะไกลบนเครื่องปาล์ม ซึ่งในการพัฒนาระบบนั้นเลือกใช้ภาษาจาวา ซึ่งมีเอพีไอที่เกี่ยวข้องกับการพัฒนาโปรแกรมบนระบบฝังตัว (Embedded System) เรียกว่าแพลตฟอร์มเจทูเอ็มอี (J2ME: Java2 Mobile Edition) ซึ่งช่วยจัดการเกี่ยวกับการทำงานระดับล่างของเครื่องปาล์ม ทำให้การพัฒนาไม่ต้องเกี่ยวข้องกับรายละเอียดเชิงฮาร์ดแวร์ของเครื่องปาล์มโดยตรงในการพัฒนาระบบ โดยได้อธิบายรายละเอียดไว้ในหัวข้อที่ 2.1.6

การเพิ่มรายละเอียดเชิงเทคนิคในการอิมพลีเมนต์ให้แบบจำลองพีไอเอ็มนั้น ในงานวิจัยนี้ใช้หลักการพิจารณาเลือกเอพีไอของแพลตฟอร์มปลายทางที่เหมาะสมต่อการทำงานหนึ่งๆ ก่อนจากนั้นจึงกำหนดแบบจำลองให้สอดคล้องกับการใช้งานเอพีไอนั้น อย่างไรก็ตาม จากการทดลองพบว่าโดยส่วนใหญ่แล้วจะไม่สามารถทำการจับคู่คอมโพเนนท์ในระดับพีไอเอ็มกับเอพีไอได้อย่างตรงไปตรงมา ในลักษณะของการจับคู่แบบหนึ่งต่อหนึ่งได้ เนื่องจากในทางปฏิบัติแล้วการอิมพลีเมนต์คอมโพเนนท์ในพีไอเอ็มมักมีความซับซ้อนในการอิมพลีเมนต์มากกว่านั้น ยกตัวอย่างเช่น ต้องมีการส่งมอบหน้าที่ระหว่างวัตถุ (Delegation) การตรวจสอบเงื่อนไข (Condition) การทำซ้ำ (Iteration) และการแปลงชนิดของข้อมูลให้เหมาะสมกับการเรียกใช้เอพีไอ ดังนั้นระหว่างขั้นตอนการแปลงแบบจำลองพีไอเอ็ม ไปยังการอิมพลีเมนต์ด้วยคอมโพเนนท์เอพีไอของแพลตฟอร์มปลายทาง ควรจะมีขั้นตอนตรงกลางได้แก่ การกำหนดอัลกอริทึม (Algorithm) ให้แก่คอมโพเนนท์ โดยตัวอย่างของอัลกอริทึมสำหรับงานวิจัยนี้แสดงไว้ในรูปที่ 4.20 ถึง 4.23 ซึ่งบางคลาสในรูปอาจไม่เคยมีมาก่อนในแบบจำลองก่อนหน้าเนื่องจากส่วนหนึ่งเป็นคลาสซึ่งเป็นเอพีไอของแพลตฟอร์ม และอีกส่วนเป็นคลาสซึ่งเกิดจากการแตกส่วนงานบางส่วนมาอิมพลีเมนต์ต่างหากในอีกคลาสหนึ่ง ซึ่งการแตกคลาสเช่นนี้มีประโยชน์ในการแยกส่วนของงานให้มีความชัดเจนยิ่งขึ้นทำให้การดูแลแก้ไข ปรับปรุง เปลี่ยนแปลงโค้ดในภายหลังนั้น น่าจะสามารถทำได้ง่ายขึ้น

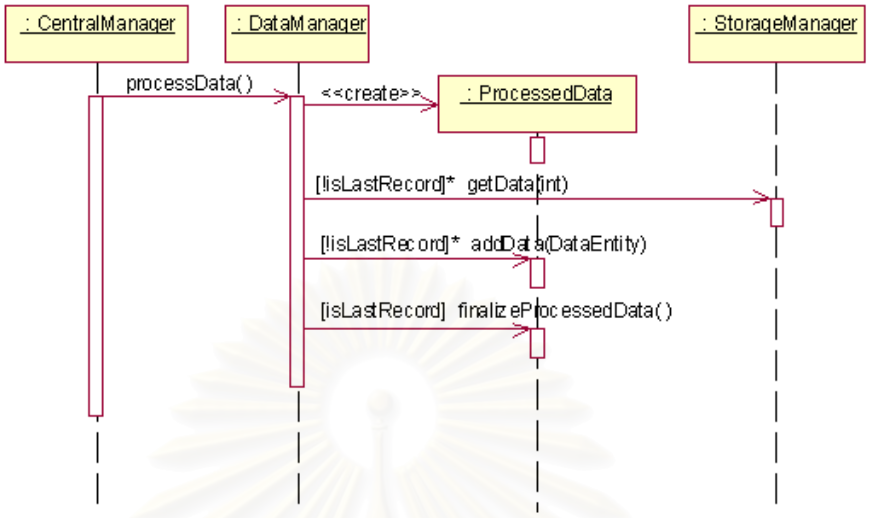


รูปที่ 4.20 แบบจำลองพีเอสเอ็มเชิงพฤติกรรมของกิจกรรม saveData

ในรูปที่ 4.20 เป็นการอธิบายอัลกอริทึมของกิจกรรม saveData ของคลาส CentralManager ซึ่งจะมอบหมายงานให้คลาส DataManager เป็นผู้ทำงานที่แท้จริง ซึ่งในกรณีนี้ผู้วิจัยเห็นว่าควรแยกงานการประมวลผลเอกสารกับงานการจัดการที่เก็บข้อมูลออกจากกัน ดังนั้นจึงสร้างคลาสเพิ่มขึ้นมาอีกหนึ่งคลาสได้แก่คลาส StorageManager มีหน้าที่จัดการที่เก็บข้อมูลซึ่งในระดับการอิมพลีเมนต์จะไปเรียกใช้งานคลาส RecordStore ซึ่งเป็นเอพีไอบนแพลตฟอร์มเจทูเอ็มอีอีกทีหนึ่ง



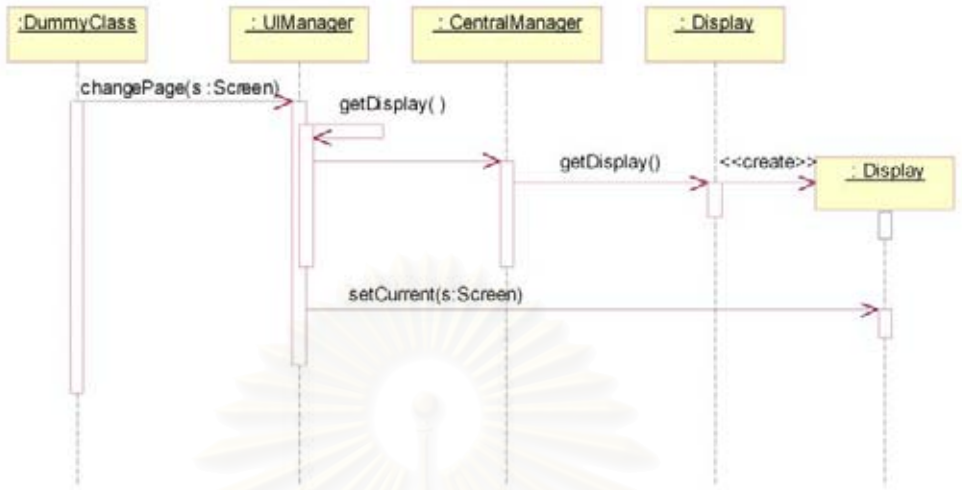
รูปที่ 4.21 แบบจำลองพีเอสเอ็มเชิงพฤติกรรมของกิจกรรม uploadData



รูปที่ 4.22 แบบจำลองพีเอสเอ็มเชิงพฤติกรรมของกิจกรรม processData

ในรูปที่ 4.21 เป็นการอธิบายอัลกอริทึมของกิจกรรม uploadData ของคลาส CentralManager ซึ่งขอข้อมูลที่ประมวลผลแล้วโดยเรียกเมธอด getProcessedData() ของคลาส DataManager จากนั้นจึงส่งข้อมูลดังกล่าวไปให้คลาส CommunicationManager เพื่ออัปโหลดข้อมูล โดยเรียกเมธอด uploadProcessedData() โดยในระดับของการอิมพลีเมนต์นั้นจะมีการเรียกใช้คลาสเอพีไอหลายคลาสเช่น มีการสร้าง HttpConnection เพื่อเชื่อมการติดต่อไปยังเครื่องเซิร์ฟเวอร์ จากนั้นจึงเปิดสตรีมด้วยเมธอด openOutputStream() แล้วจึงเขียนข้อมูลลงสตรีมดังกล่าวด้วยเมธอด write() ซึ่งเมื่อเขียนข้อมูลจนหมดแล้วต้องมีการฟลัชข้อมูลออกจากบัฟเฟอร์โดยเรียกเมธอด flush()

ในรูปที่ 4.22 เป็นการอธิบายอัลกอริทึมของกิจกรรม processData ของคลาส CentralManager ซึ่งจะมอบหมายงานไปให้คลาส DataManager อีกทีหนึ่ง โดยในระดับอิมพลีเมนต์จะวนลูปขอข้อมูลจากคลาส StorageManager ทีละเรคคอร์ดโดยเรียกเมธอด getData() แล้วจึงนำข้อมูลดังกล่าวมาประมวลผลพร้อมเก็บพักไว้ในคลาส ProcessedData ซึ่งเป็นคลาสที่ทำหน้าที่เป็นโครงสร้างข้อมูล (Data Structure) ของข้อมูลที่ประมวลผลแล้ว โดยเรียกเมธอด addData() และเมื่อวนลูปจนครบทุกเรคคอร์ดแล้วจะต้องเรียกเมธอด finalizeProcessedData() จึงจะเสร็จสมบูรณ์ (ซึ่งก็คือการเขียนแท็กปิดลงในเอกสารเอ็กซ์เอ็มแอล ซึ่งจำเป็นต้องทำท้ายสุด)

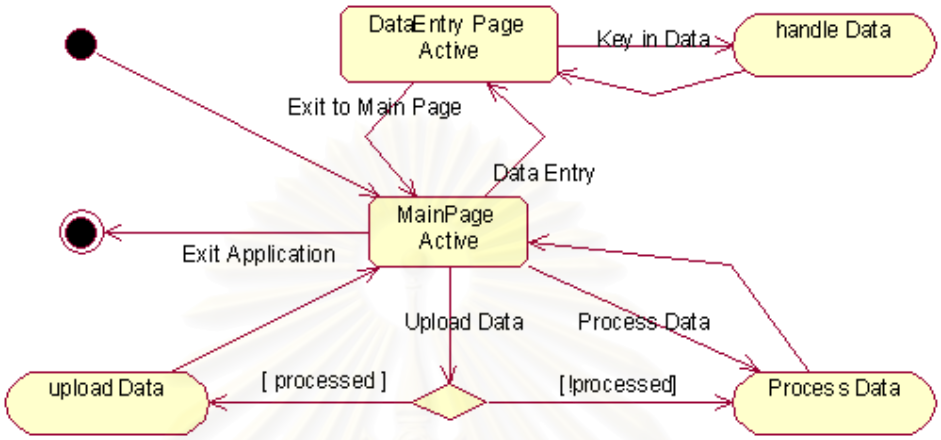


รูปที่ 4.23 ตัวอย่างแบบจำลองพีเอสเอ็มเชิงพฤติกรรมของกิจกรรม changePage

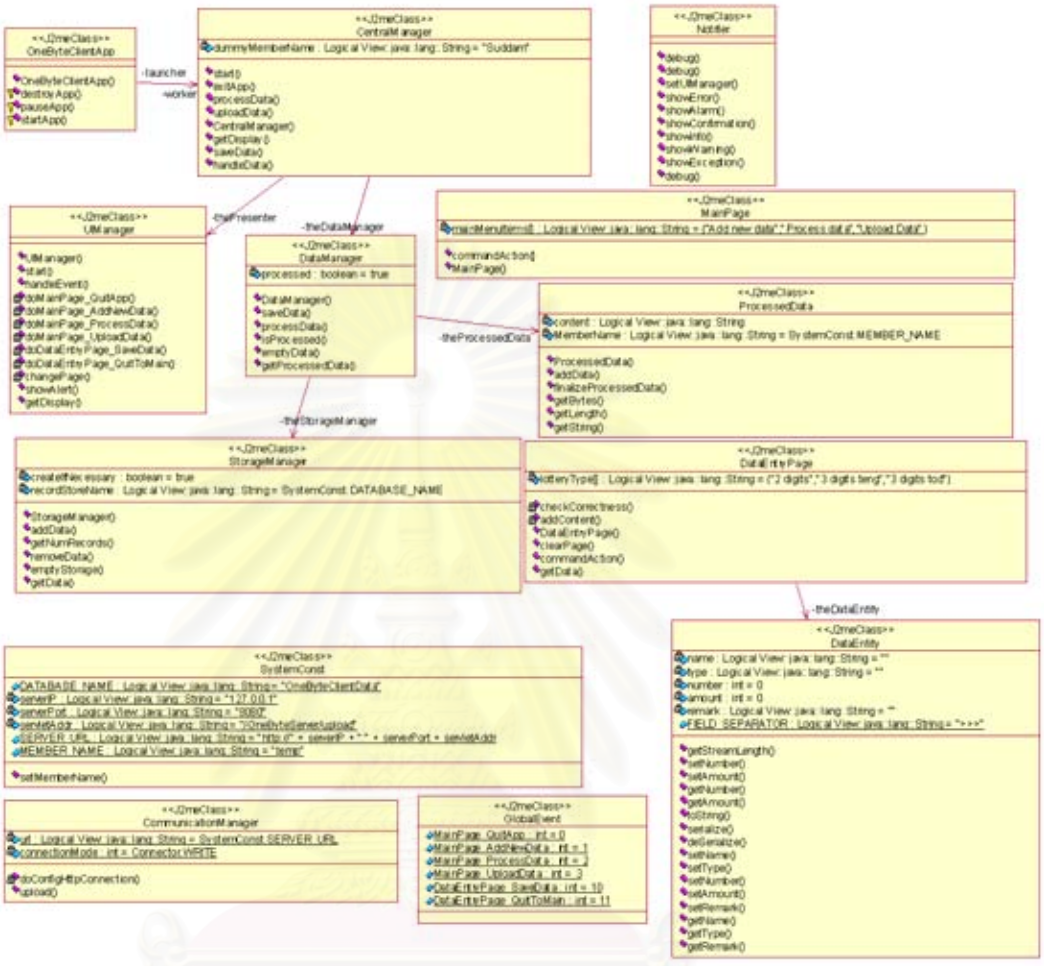
รูปที่ 4.23 เป็นการอธิบายอัลกอริทึมของกิจกรรม changePage ของคลาส UIManager โดยเริ่มจากการขอสิทธิการแสดงผลบนหน้าจอด้วยการเรียกเมธอด getDisplay() ซึ่งในระดับอิม-พลีเมนต์นั้นต้องขอไปยังคลาส CentralManager ซึ่งจะเรียกสแตติกเมธอด (Static Method) getDisplay() ของคลาส Display อีกทีหนึ่ง เพื่อขอสิทธิการแสดงผล ซึ่งก็คืออินสแตนซ์ของคลาส Display โดยในหนึ่งแอพพลิเคชันจะมีเพียงหนึ่งอินสแตนซ์ของคลาส Display นี้เท่านั้น และเมื่อได้สิทธิการแสดงผลมาแล้ว จึงเปลี่ยนหน้าจอโดยการเรียกเมธอด setCurrent()

สำหรับแบบจำลองพีไอเอ็มเชิงพฤติกรรมของทั้งระบบนั้น (ดูรูปที่ 4.7) ในระดับของแบบจำลองพีเอสเอ็มจะถูกอิมพลีเมนต์ด้วยคลาส UIManager เนื่องจากในการอิมพลีเมนต์ระบบบนแพลตฟอร์มปาล์มนั้น ความสามารถในการรับรู้เหตุการณ์ (Event) ที่มาจากผู้ใช้งานนั้น มาจากคลาสที่ทำหน้าที่เป็นส่วนติดต่อกับผู้ใช้งาน (User Interface) ดังนั้นการอิมพลีเมนต์การเปลี่ยนแปลงสเตทต่อเหตุการณ์ของระบบ ซึ่งส่วนใหญ่เป็นเหตุการณ์ที่มาจากผู้ใช้งาน ด้วยคลาสที่ทำหน้าที่ในการควบคุมการติดต่อกับผู้ใช้งาน (UIManager) จึงน่าจะมีความเหมาะสม ดังแสดงแบบจำลองเชิงพฤติกรรมของคลาสตัวควบคุมการติดต่อกับผู้ใช้งานดังรูปที่ 4.24 ซึ่งสังเกตได้ว่ามีการเปลี่ยนจากรูปที่ 4.7 โดยการแยกสเตท ready ออกเป็นสองสเตทย่อยทั้งนี้เนื่องจากในทางปฏิบัตินั้นระบบจะรับรู้อีเวนต์แต่ละอย่างผ่านทางหน้าจอ ซึ่งในกรณีนี้ผู้วิจัยเลือกที่จะอิมพลีเมนต์เป็นสองหน้าจอโดยหน้าจอแรกเป็นเมนูเพื่อรับคำสั่งต่างๆ จากผู้ใช้งาน เรียกว่า MainPage ส่วนอีกหน้าจอนั้นเป็นหน้าจอสำหรับการกรอกข้อมูลการซื้อลอตเตอรี่จึงมีชื่อเรียกว่า DataEntryPage

ซึงการทีระบบแสดงผลในหน้าจอใดนั้น คือการทีหน้าจอนั้นเป็นหน้าจอทีแอคทีฟอยู่นันเอง ทำให้สามารถแบ่งสถานะออกเป็นสองสถานะตามการแอคทีฟของสองหน้าจอได้



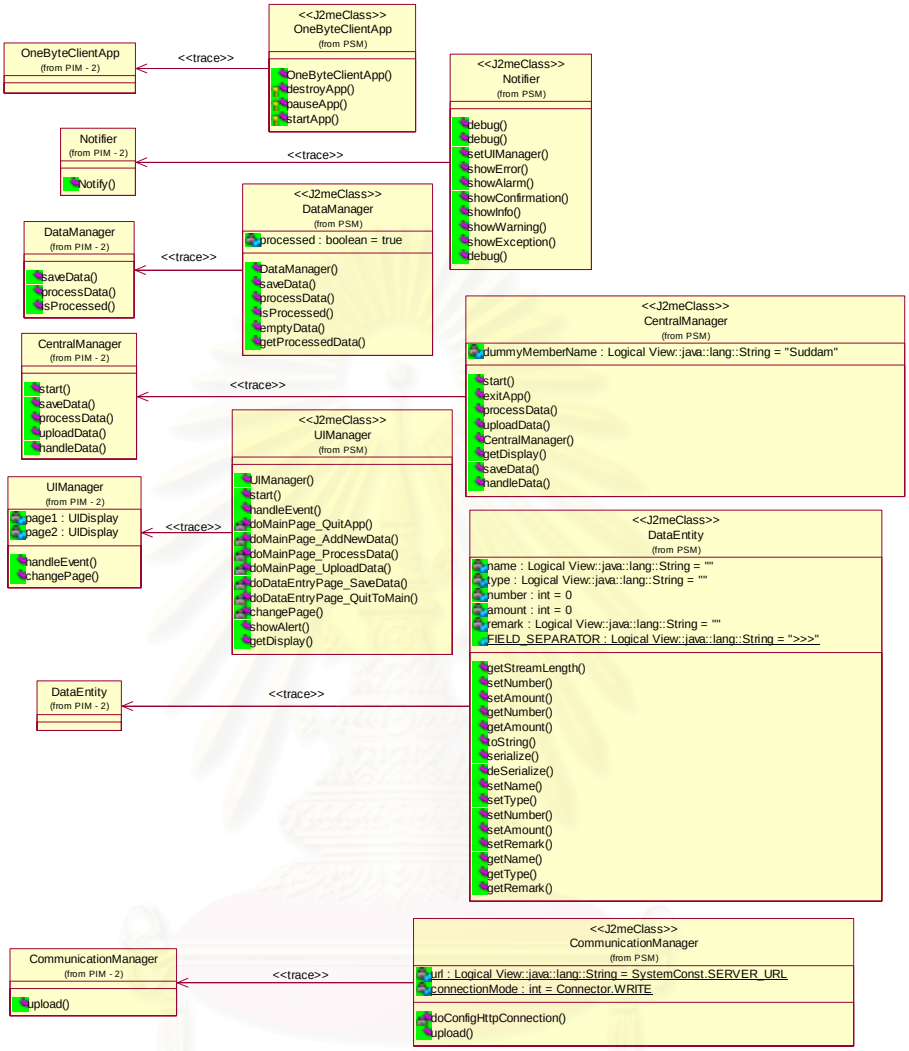
รูปที่ 4.24 แบบจำลองพีเอสเอ็มเชิงพฤติกรรมของคลาส UIManager



รูปที่ 4.25 แบบจำลองพีเอสเอ็มเชิงโครงสร้างของตัวรวบรวมข้อมูลระยะไกล

หลังจากการกำหนดอัลกอริทึมให้กับคอมโพเนนท์ต่างๆ จะทำให้ทราบถึงรายละเอียดการอิมพลีเมนต์ของแต่ละคอมโพเนนท์มากขึ้น ซึ่งจะนำมาสู่การสร้างแบบจำลองเชิงโครงสร้างดังรูปที่ 4.25 ซึ่งส่วนหนึ่งมาจากความต้องการที่เกิดขึ้นในขั้นตอนของการกำหนดอัลกอริทึม แต่อีกส่วนก็มาจากการกำหนดเพิ่มเติมขึ้นเอง ยกตัวอย่างเช่นคลาส SystemConst ซึ่งเป็นคลาสที่ใช้เป็นที่รวบรวมค่าคงที่ของระบบทำให้การแก้ไขหรือปรับปรุงค่าต่างๆ ในอนาคตสามารถทำได้สะดวกมากขึ้นเนื่องจากสามารถจัดการแก้ไขจากที่เดียว คลาส GlobalEvent นั้นเป็นการกำหนดชื่อของอีเวนต์ ซึ่งใช้ในการจำลองคลาส UIManager ให้สามารถทำงานในลักษณะของสเตทแมชชีนได้ ส่วนคลาส MainPage และ DataEntryPage นั้นเป็นคลาสที่กำหนดรายละเอียดเชิงโครงสร้าง

และพฤติกรรมของแต่ละหน้าจอ (Screen) คลาส Notifier ทำหน้าที่ในการแจ้งเตือนข้อผิดพลาดต่างๆ ที่เกิดขึ้นในระบบ คลาส ProcessedData เป็นโครงสร้างข้อมูล (Data Structure) ซึ่งใช้เก็บข้อมูลในรูปแบบของเอกสารเอ็กซ์เอ็มแอล ส่วนชื่อเมธอดที่เพิ่มขึ้นมานั้นส่วนหนึ่งเป็นเมธอดแบบไพรเวท (Private Method) ซึ่งเป็นเทคนิคในการแยกส่วนของโค้ดแต่ละส่วนมาเขียนเป็นไพรเวทเมธอดย่อยๆ ต่างหาก บางส่วนเป็นเมธอด เซท/เก็ต (set/get) ใช้สำหรับการอ่านและเขียนแอททริบิวท์ของคลาส บางส่วนก็เป็นเมธอดที่ได้รับการถ่ายทอดมาจากคลาสแม่เช่น commandAction() ของคลาส MainPage และ DataEntryPage ซึ่งได้รับการถ่ายทอดมาจากคลาส Display ของเจทูเอ็มอี ส่วนเมธอด startApp(), pauseApp() และ destroyApp() ของคลาส OneByteClientApp ได้รับการถ่ายทอดมาจากคลาส MIDlet ของเจทูเอ็มอี การเชื่อมโยงแบบจำลองในระดับพีไอเอ็ม (รูปที่ 4.8) กับระดับพีเอสเอ็ม (รูปที่ 4.25) แสดงไว้ในรูปที่ 4.26

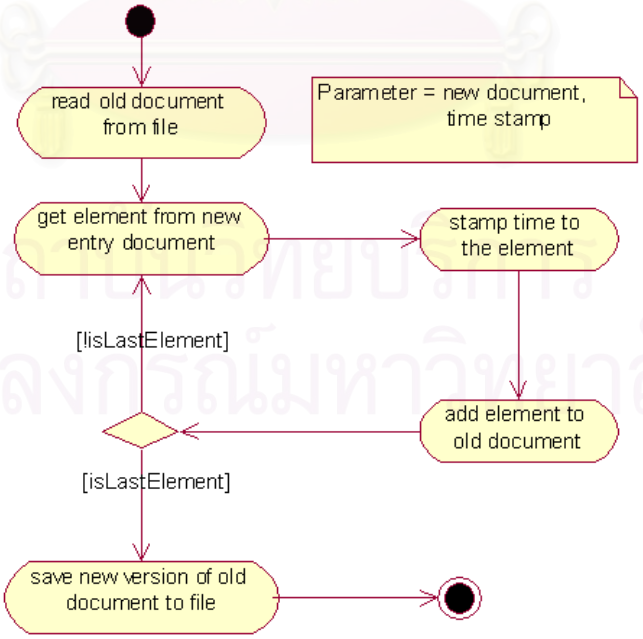


รูปที่ 4.26 การเชื่อมโยงระหว่างแบบจำลองเชิงโครงสร้างระดับพีไอเอ็มและพีเอสเอ็ม

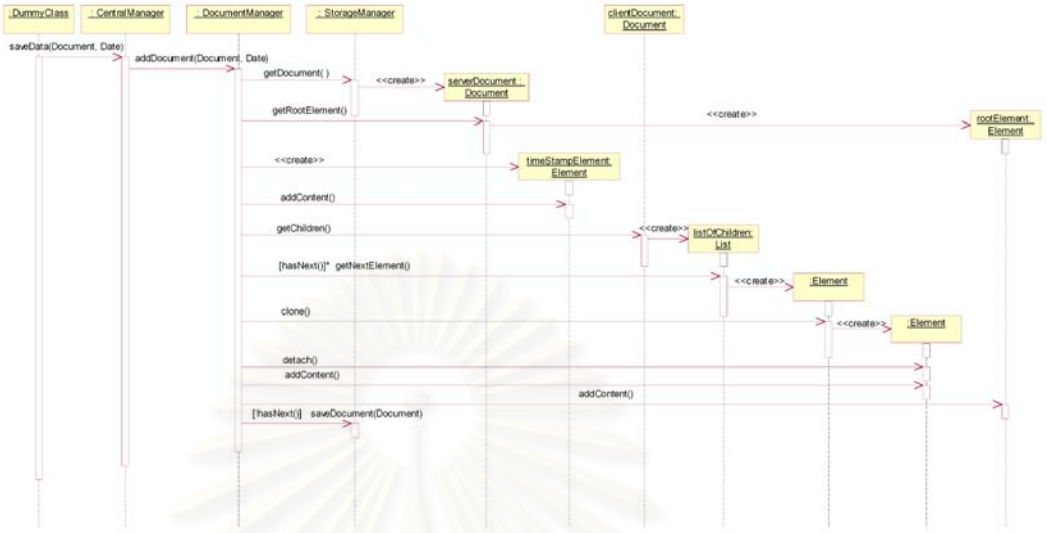
4.4.2 แบบจำลองพีเอสเอ็มของตัวเก็บข้อมูลส่วนกลาง

สำหรับการอิมพลีเมนต์ตัวเก็บข้อมูลส่วนกลางนั้น ในงานวิจัยชิ้นนี้เลือกแพลตฟอร์มเจทูอีอี/เซิร์ฟเล็ต (J2EE / Servlet) ในการอิมพลีเมนต์ เนื่องจากมีความสะดวกในการจัดการการติดต่อจากเครื่องไคลเอนต์ซึ่งเขียนด้วยภาษาจาวาเช่นกัน โดยการสร้างแบบจำลองพีเอสเอ็มของตัวเก็บข้อมูลส่วนกลางนี้นั้น ก็มีหลักการเดียวกับการสร้างแบบจำลองพีเอสเอ็มของตัวรวบรวมข้อมูลระยะไกล โดยการแมปปิงคอมโพเนนท์ของแบบจำลองพีไอเอ็ม กับเอพีไอคอมโพเนนท์ที่เหมาะสม

บนแพลตฟอร์มเจทูอีอี/เซิร์ฟเล็ต อย่างไรก็ตาม จากการทดลองพบว่าไม่สามารถทำการจับคู่คอมโพเนนท์ทั้งสองระดับแบบหนึ่งต่อหนึ่งได้โดยตรง เนื่องจากพบว่าระบบมีความซับซ้อนเชิงเทคนิคของพฤติกรรมของระบบมากกว่าที่รูปที่ 4.15 ได้อธิบายไว้ ทำให้ต้องมีการอธิบายรายละเอียดเชิงพฤติกรรมเพิ่มเติมเสียก่อน ดังแสดงในรูปที่ 4.27 ซึ่งเป็นการเพิ่มรายละเอียดของกิจกรรม saveData ในรูปที่ 4.15 โดยสังเกตได้ว่า แบบจำลองที่ได้มีหน้าตาคล้ายกับโค้ดเทียม (Pseudo Code) หรือผังงาน (Flowchart) กล่าวคือแบบจำลองในรูปที่ 4.27 นี้เป็นการกำหนดอัลกอริทึมให้กับกิจกรรม saveData นั่นเอง ประเด็นที่น่าสนใจคือแบบจำลองในรูปที่ 4.27 นี้ควรจัดเป็นแบบจำลองพีไอเอ็มหรือพีเอสเอ็ม สำหรับความเห็นของผู้วิจัยเห็นว่าควรจัดเป็นแบบจำลองพีไอเอ็มเนื่องจาก เห็นว่าแบบจำลองนี้ไม่ขึ้นกับแพลตฟอร์มที่อิมพลีเมนต์ แต่ก็อาจไม่สามารถสรุปว่าเป็นพีไอเอ็มได้อย่างเต็มที่นักเนื่องจาก จากการทดลองพัฒนาระบบตัวอย่างนี้พบว่าคลาสเอพีไอที่เลือกใช้นั้น อนุญาตให้ทำการเพิ่มข้อมูลในเอกสารเอ็กซ์เอ็มแอลได้ทีละหนึ่งอีลีเมนต์เท่านั้น ทำให้แบบจำลองเชิงพฤติกรรมของกิจกรรม saveData ในรูปที่ 4.17 นั้นไม่ถูกต้องนัก เนื่องจากแบบจำลองดังกล่าวสมมติว่าการเพิ่มข้อมูลสามารถทำได้ทั้งเอกสาร (ทุกอีลีเมนต์) พร้อมกัน ดังนั้นจึงต้องมีการเพิ่มอัลกอริทึมของการวนลูปเพื่อเพิ่มข้อมูลเข้าเอกสารทีละอีลีเมนต์ ซึ่งจากเหตุผลดังกล่าวนี้เองทำให้ผู้วิจัยต้องเลือกเอพีไอที่สามารถใช้อิมพลีเมนต์ระบบได้เสียก่อน จึงจะสามารถกำหนดอัลกอริทึมที่เหมาะสมได้ แสดงว่าอัลกอริทึมก็น่าจะมีความขึ้นกับเอพีไอที่จะเลือกใช้ในระดับหนึ่ง ดังนั้นจึงอาจมองแบบจำลองส่วนนี้ว่ามีบางส่วนเป็นพีเอสเอ็มได้เช่นกัน



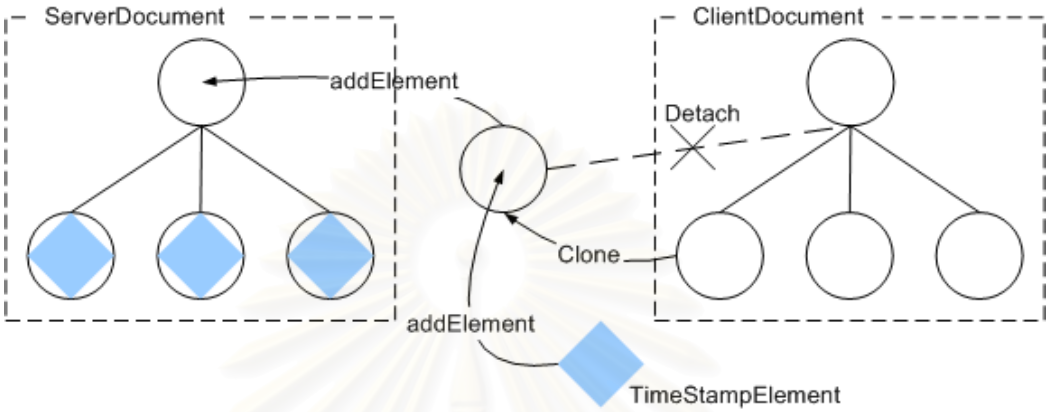
รูปที่ 4.27 แบบจำลองเชิงพฤติกรรมของกิจกรรม saveData ของตัวเก็บข้อมูลส่วนกลาง



รูปที่ 4.28 แบบจำลองเชิงพฤติกรรมระดับล่างของกิจกรรม saveData

หลังจากกำหนดอัลกอริทึมของกิจกรรม saveData ดังรูปที่ 4.27 ขั้นตอนต่อไปคือการกำหนดรายละเอียดการทำงานโต้ตอบกันระหว่างคลาสซึ่งแสดงด้วยแผนภาพซีเควนซ์ดังรูปที่ 4.28 ซึ่งในกรณีนี้เป็นตัวอย่างที่ดีของระบบงานที่มีตรรกะที่ซับซ้อน สังเกตได้จากจำนวนคลาสและลำดับขั้นของการเรียกใช้งานเมธอดที่มีหลายขั้นตอน เฉพาะกิจกรรม saveData ต้องใช้คลาสในการอิมพลีเมนต์ถึง 2 คลาสได้แก่ DocumentManager รับผิดชอบการจัดการกับเอกสารเอ็กซ์เอ็มแอล และ StorageManager ซึ่งทำหน้าที่ในการอ่านและเขียนเอกสารเอ็กซ์เอ็มแอลไปยังที่เก็บข้อมูลซึ่งในกรณีนี้เป็นไฟล์อักขระ (Text File) นอกจากนั้นยังมีการเรียกใช้คลาสเอพีไอถึง 8 คลาส ได้แก่ DOMBuilder, XMLOutputter, List, LinkedList, ListIterator, File, FileWriter และ Date และในบางครั้งอาจมีรายละเอียดเชิงข้อจำกัดของการเรียกใช้งานเอพีไอบางตัว ยกตัวอย่างเช่น การเก็บข้อมูลซึ่งฝั่งไคลเอนต์ส่งมาในรูปแบบเอ็กซ์เอ็มแอลลงในไฟล์ข้อมูลเอ็กซ์เอ็มแอลที่ฝั่งเซิร์ฟเวอร์นั้น จะต้องเพิ่มโหนดอีลีเมนต์จากข้อมูลในฝั่งไคลเอนต์ลงในไฟล์เอ็กซ์เอ็มแอลที่ฝั่งเซิร์ฟเวอร์ แต่เนื่องจากจะไม่สามารถเพิ่มโหนดอีลีเมนต์ได้ เพราะโหนดนั้นมีโหนดแม่อยู่แล้วในข้อมูลทางฝั่งไคลเอนต์ ดังนั้นดังในรูปที่ 4.29 จึงต้องทำการคัดลอก (Clone) โหนดอีลีเมนต์ลูกดังกล่าวออกมาก่อน จากนั้นทำการปลดความสัมพันธ์ของโหนดที่คัดลอกมาออกจากโหนดแม่ทางฝั่งข้อมูลไคลเอนต์ (Detach) แล้วจึงค่อยบันทึกเวลาประทับ (Timestamp) ของการเก็บข้อมูลลงในโหนดลูกที่คัดลอกมา แล้วเพิ่มโหนดลูกดังกล่าวลงในไฟล์เอ็กซ์เอ็มแอลของฝั่งเซิร์ฟเวอร์ เป็นต้น ซึ่งรายละเอียดเหล่านี้นั้นเป็นรายละเอียดเชิงพฤติกรรมที่จำเป็นของระบบ ซึ่งไม่มีทางทราบได้ใน

ระดับพีไอเอ็มเนื่องจากต้องมีการกำหนดเอพีไอที่จะเลือกใช้เสียก่อน ในขณะเดียวกันเราวางหลักการของการสร้างแบบจำลองพีเอสเอ็ม จากการแมปปิงระหว่างคอมโพเนนท์ในสองระดับ ดังนั้นการแมปปิงจึงไม่จำเป็นต้องเป็นแบบหนึ่งต่อหนึ่งเท่านั้นขึ้นอยู่กับเอพีไอทีเลือกใช้

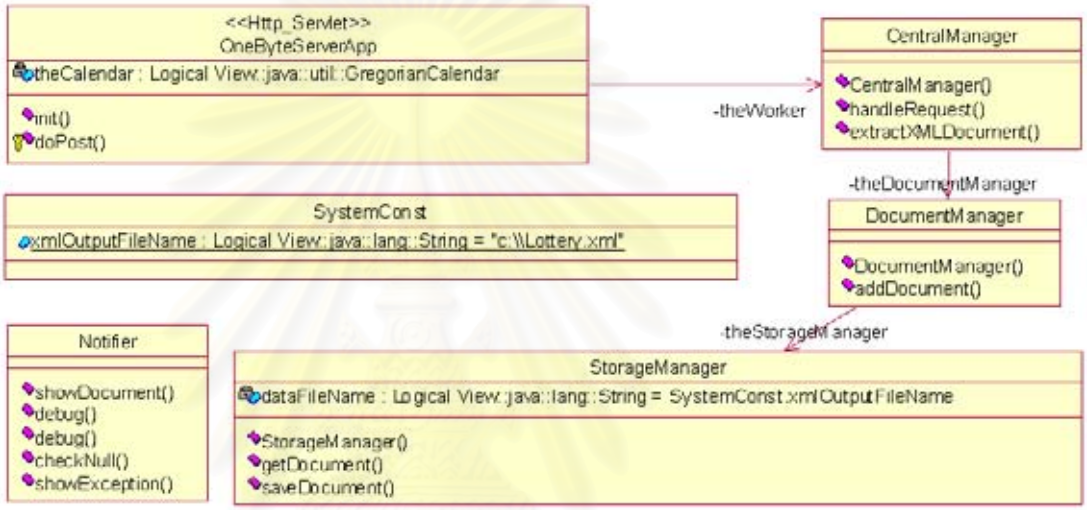


รูปที่ 4.29 กระบวนการเพิ่มโหนดอีลีเมนต์ให้เอกสารเอ็กซ์เอ็มแอล

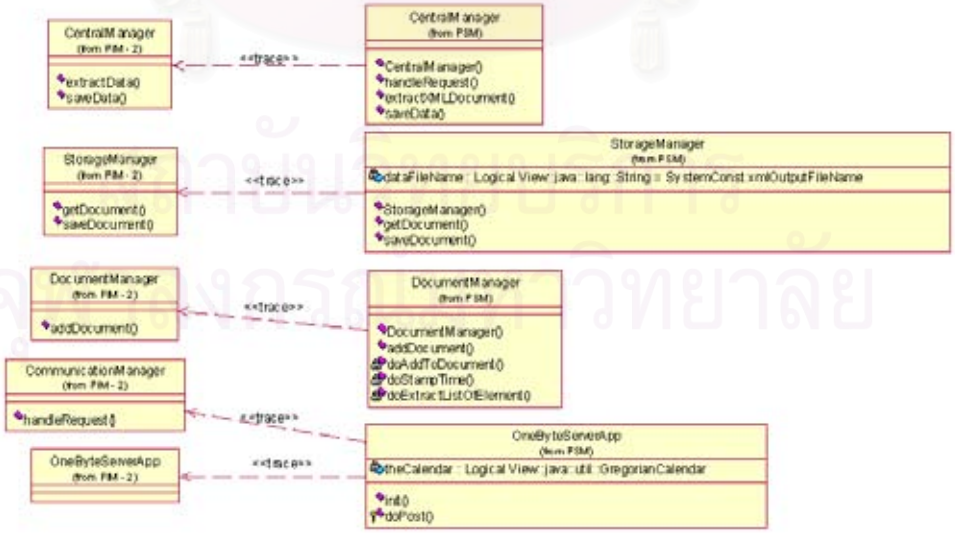
ปัญหาที่น่าสนใจคือ เราจะสามารถสร้างมาตรฐานการแมปปิงที่มีความซับซ้อนมากๆ เช่นนี้ได้หรือไม่ ยกตัวอย่างความต้องการของมาตรฐานการแมปปิงเช่น สามารถแมปปิงจากแบบจำลองพีไอเอ็มซึ่งมีทั้งส่วนโครงสร้างและส่วนพฤติกรรม ซึ่งแต่ละส่วนมีจำนวนแผนภาพได้ไม่จำกัด ไปเป็นแบบจำลองพีเอสเอ็มซึ่งก็มีทั้งส่วนโครงสร้างและส่วนพฤติกรรม แต่ละส่วนมีจำนวนแผนภาพได้ไม่จำกัด และไม่จำเป็นต้องมีจำนวนแผนภาพหรือจำนวนคอมโพเนนท์เท่ากับที่ได้ออกแบบไว้ในแบบจำลองพีไอเอ็มก็ได้ นอกจากนั้นมาตรฐานการแมปปิงยังต้องอนุญาตให้มีการเพิ่มเติมรายละเอียดบางอย่างระหว่างทางของการแปลงได้ (เช่นในรูปที่ 4.28) ต้องสามารถทำให้มาตรฐานการแปลงนี้ทำงานแบบอัตโนมัติได้ ต้องสามารถนำกลับมาใช้ใหม่ได้สำหรับการแปลงครั้งต่อไปจากแบบจำลองพีไอเอ็มที่ต่างออกไป ซึ่งจากการศึกษาพบว่าประเด็นความซับซ้อนต่างๆ เหล่านี้ยังไม่มีคำตอบที่ชัดเจนในเวลานี้

หลังจากมีการเพิ่มเติมรายละเอียดเชิงพฤติกรรมของระบบ จะทำให้สามารถระบุขอบเขตการทำงานที่ต้องการให้คลาสแต่ละคลาสทำได้ และกำหนดเป็นความสัมพันธ์เชิงโครงสร้างออกมา ดังแสดงในรูปที่ 4.30 ซึ่งบางคลาสในรูปที่ 4.30 อาจไม่เคยเห็นมาก่อนยกตัวอย่างเช่น SystemConst ซึ่งในที่นี้ทำหน้าที่เป็นที่เก็บข้อมูลค่าคงที่ของทั้งระบบช่วยให้การเปลี่ยนแปลงค่าต่างๆ เหล่านี้ในภายหลังสามารถทำได้ง่าย คลาส Notifier ทำหน้าที่ในการจัดการฟ้องเตือนเกี่ยวกับข้อผิดพลาดต่างๆ ของระบบที่อาจจะเกิดขึ้น ในขณะที่บางคลาสหายไปเช่น

CommunicationManager เนื่องจากในแพลตฟอร์มเจทูอีอี/เซิร์ฟเล็ตนั้น คลาส HttpServlet ทุกคลาสจะทำหน้าเป็นตัวรับคำร้องขอซึ่งส่งมาจากเครื่องไคลเอนต์ โดยจะเรียกผ่านเมธอดชื่อ doPost() หรือ doGet() แล้วแต่ชนิดของคำร้องขอ ดังนั้นสามารถกล่าวได้ว่าคลาส HttpServlet อิมพลีเมนต์คลาส CommunicationManager เรียบร้อยแล้วจึงไม่จำเป็นต้องมีคลาสดังกล่าวอีกในระดับอิมพลีเมนต์ โดยรายละเอียดการเชื่อมโยงแบบจำลองพีไอเอ็ม (รูปที่ 4.16) กับแบบจำลองพีเอสเอ็ม (รูปที่ 4.30) แสดงไว้ในรูปที่ 4.31



รูปที่ 4.30 แบบจำลองพีเอสเอ็มเชิงโครงสร้างของตัวเก็บข้อมูลส่วนกลาง



รูปที่ 4.31 การเชื่อมโยงแบบจำลองเชิงโครงสร้างระดับพีไอเอ็มและพีเอสเอ็ม

4.5 การสร้างโค้ดของระบบตัวอย่าง

ในหัวข้อนี้จะแสดงถึงการแปลงแบบจำลองเป็นโค้ด โดยจะแสดงการเปรียบเทียบความหมายระหว่างแบบจำลองและโค้ดว่ามีความสอดคล้องกันอย่างไร โดยลำดับของการสร้างโค้ดจะสร้างทีละส่วนตามการแบ่งส่วนของโค้ดเป็น 3 ส่วนดังที่ได้เคยกล่าวไว้ในหัวข้อที่ 3.4.1

4.5.1 รูปแบบการตั้งชื่อตัวแปรที่ใช้ในการสร้างโค้ด

ในงานวิจัยชิ้นนี้ มีการกำหนดรูปแบบการตั้งชื่อตัวแปรในขั้นตอนของการสร้างโค้ดขึ้น เนื่องจากหากต้องการอิมพลีเมนต์เป็นเครื่องมืออัตโนมัติจริงๆ นั้น ประเด็นของการตั้งชื่อตัวแปรก็จะเป็นประเด็นหนึ่งซึ่งสำคัญมากเพราะโค้ดที่มีมาตรฐานการตั้งชื่อตัวแปรที่ดีจะทำให้โค้ดอ่านง่ายขึ้น และเข้าใจโค้ดได้ง่ายขึ้น ทำให้โค้ดมีคุณภาพมากขึ้น โดยการตั้งชื่อตัวแปรนั้นจะแยกตามชนิดของตัวแปร ซึ่งในงานวิจัยชิ้นนี้แบ่งชนิดตัวแปรเป็น 6 ชนิด ได้แก่

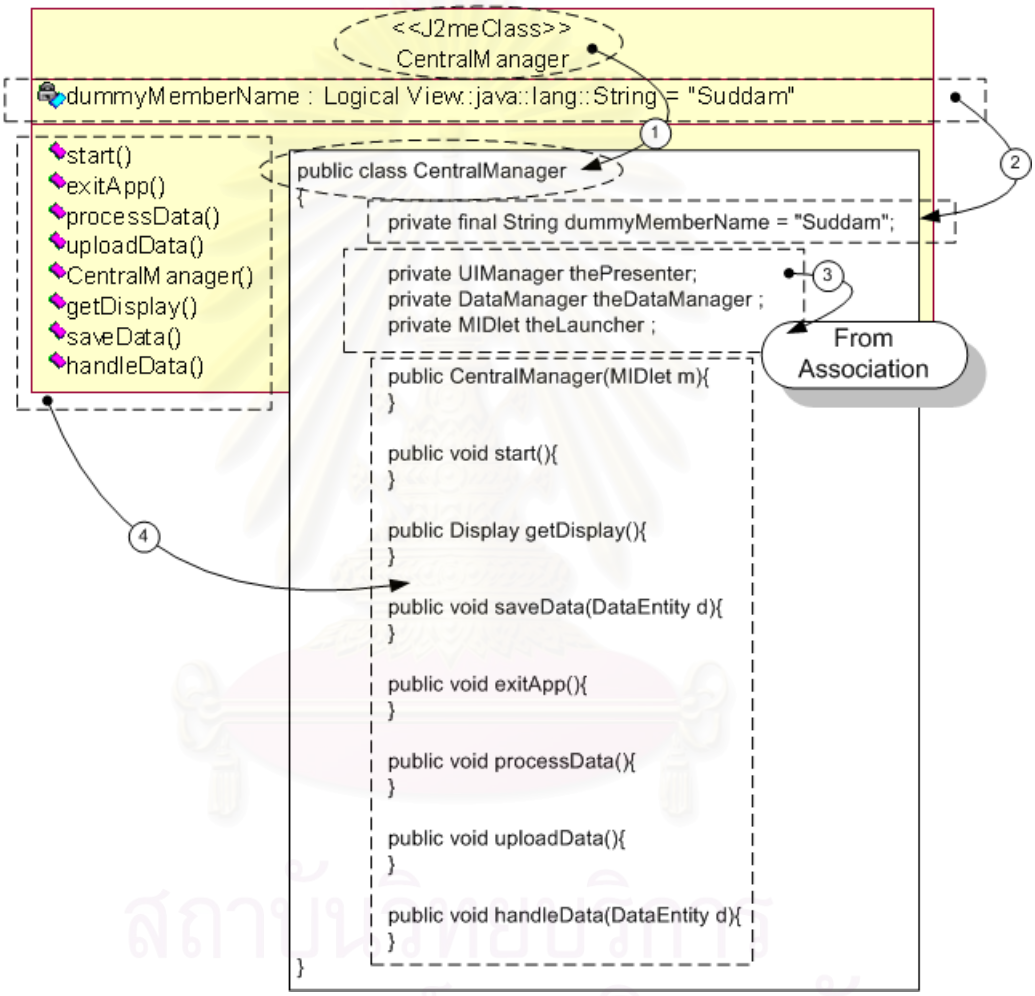
1. ตัวแปรพารามิเตอร์ (Parameter Variable)ใช้ตัวอักษรหนึ่งตัวซึ่งมักเป็นตัวย่อของชื่อคลาสของพารามิเตอร์นั้นๆ
2. ตัวแปรอินสแตนซ์ (Instance Variable) ใช้ “the” นำหน้าชื่อคลาสหรือชื่อบทบาท
3. ตัวแปรนับเลข (Counter Variable) ใช้ตัวอักษรตัวใดตัวหนึ่งในเซต {i,j,k,n}
4. ตัวแปรชั่วคราว (Temporary Variable) ใช้คำว่า dummy หรือคำว่า temp นำหน้าชื่อคลาสของตัวแปรนั้นๆ
5. ตัวแปรอาร์กิวเมนต์ (Argument Variable) ใช้คำว่า arg ตามด้วยตัวเลขแสดงลำดับของอาร์กิวเมนต์นั้นๆ เช่น arg1 แทนอาร์กิวเมนต์ตัวที่หนึ่งเป็นต้น
6. ตัวแปรคืนค่า (Return Variable) ใช้คำว่า return ตามด้วยชื่อคลาสของค่าที่คืนกลับ หรือใช้คำว่า returnValue ในกรณีที่ค่าที่รีเทิร์นนั้นเป็นชนิดพริมิทีฟ (Primitive Type)

อย่างไรก็ตาม การตั้งชื่อตัวแปรในโค้ดก็อาจไม่ต้องปฏิบัติตามแนวปฏิบัตินี้เสมอไป เช่น ถ้าใช้ชื่ออื่นแล้วสื่อความหมายได้ดีกว่าก็สามารถเปลี่ยนชื่อได้ตามสมควร แนวปฏิบัติดังกล่าวนี้เป็นเพียงแนวทางให้เครื่องมือสร้างโค้ดโดยอัตโนมัติมีความเป็นไปได้เพิ่มมากขึ้นเท่านั้น

4.5.2 การสร้างโค้ดส่วนโครงร่าง

จากแบบจำลองพีเอสเอ็มเชิงโครงสร้างของตัวรวบรวมข้อมูลระยะไกล ซึ่งแสดงในรูปที่ 4.25 จะสามารถนำมาสร้างโค้ดส่วนโครงร่างได้ตามหลักเกณฑ์ที่ได้กล่าวในหัวข้อที่ 3.4.2 โดยแสดงตัวอย่างการสร้างโค้ดของตัวรวบรวมข้อมูลระยะไกลเปรียบเทียบกับแผนภาพคลาส ดังรูปที่

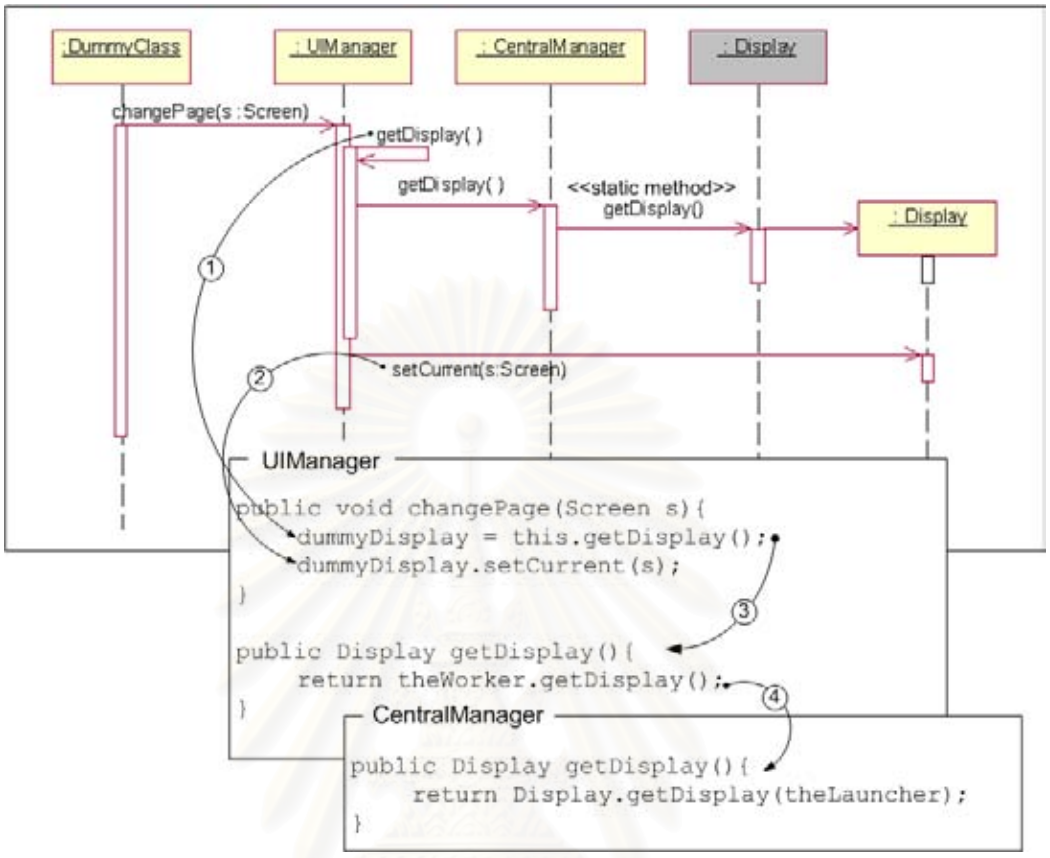
4.32 ซึ่งโค้ดส่วนที่ (1) นั้นเป็นการกำหนดโครงสร้างพร้อมชื่อของคลาส โค้ดส่วนที่ (2) และ (3) เป็นการกำหนดชื่อและชนิดของแอททริบิวท์ของคลาส แต่แตกต่างกันตรงที่โค้ดในส่วนที่ (3) เกิดจากความสัมพันธ์ระหว่างคลาส CentralManager กับคลาสอื่นๆ ซึ่งจะสังเกตได้ว่าชื่อของแอททริบิวท์นั้นกำหนดโดยใช้ชื่อของบทบาท (Role) ซึ่งปรากฏอยู่ในแผนภาพคลาส สำหรับโค้ดในส่วนที่ (4) นั้นเป็นการสร้างโอเปอร์เรชันพร้อมทั้งชื่อและชนิดของพารามิเตอร์ และค่ารีเทิร์นด้วย



รูปที่ 4.32 การเปรียบเทียบแผนภาพคลาสกับโค้ดส่วนโครงร่างที่ถูกสร้างขึ้น

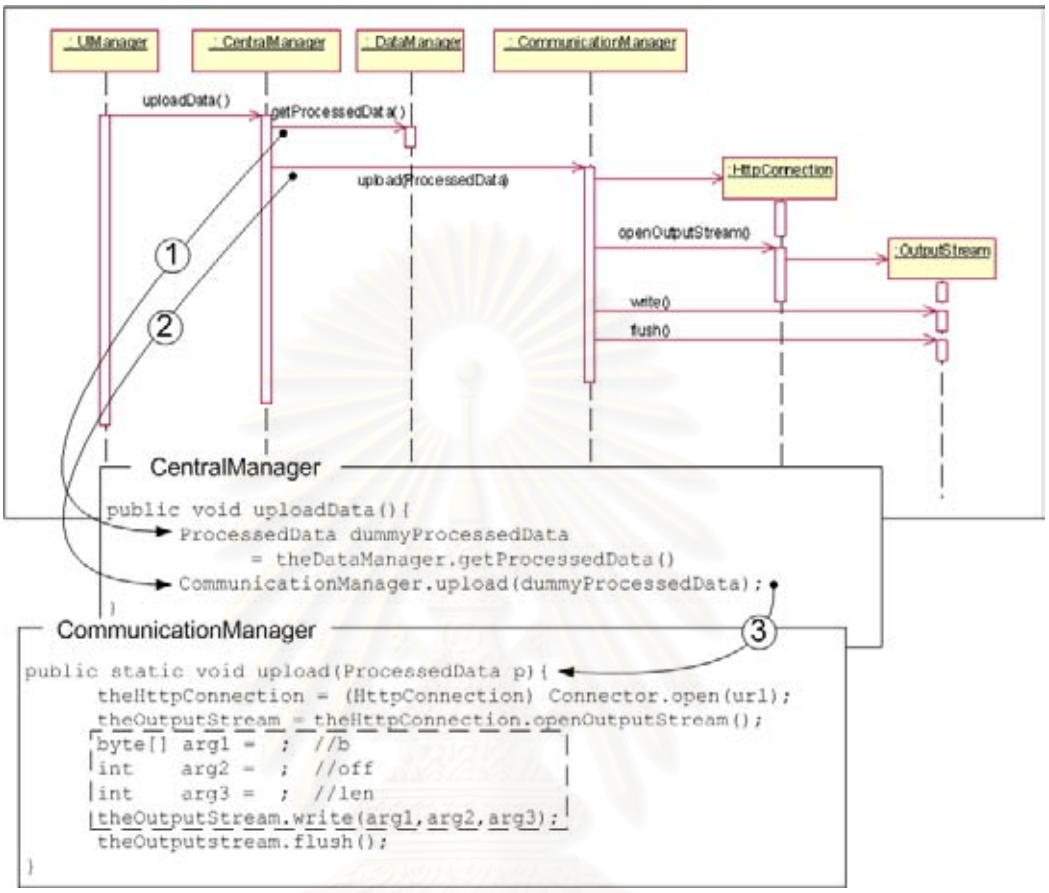
4.5.3 การสร้างโค้ดส่วนตรรกะหลัก

การสร้างโค้ดส่วนตรรกะหลักนั้นจะอาศัยหลักการที่ได้อธิบายไปแล้วในหัวข้อที่ 3.4.3 โดยตัวอย่างของโค้ดส่วนตรรกะหลักที่สร้างได้จากแบบจำลองพีเอสเอ็มแสดงได้ดังรูปที่ 4.33 ถึงรูปที่ 4.35



รูปที่ 4.33 การเชื่อมโยงระหว่างแบบจำลองพีเอสเอ็มเชิงพฤติกรรมกับโค้ดที่สร้างได้ของกิจกรรม ChangePage ของตัวรวบรวมข้อมูลระยะไกล

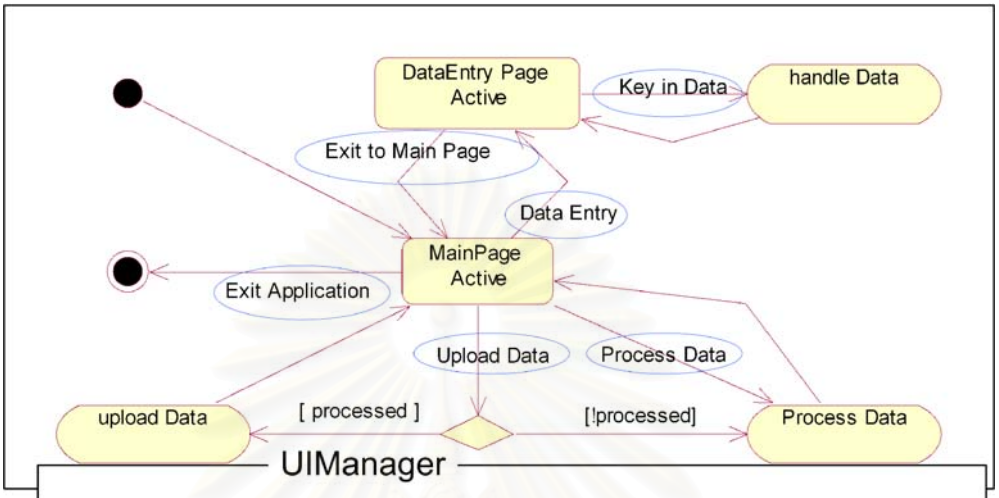
รูปที่ 4.33 แสดงตัวอย่างการสร้างโค้ดส่วนตรรกะหลักจากแผนภาพซีเควนซ์ของกิจกรรม changePage (รูปที่ 4.23) ซึ่งเป็นการเปลี่ยนหน้าจอแสดงผล โดยเริ่มจากการขอสิทธิการแสดงผล (1) ซึ่งเป็นอินสแตนซ์ของคลาส Display จึงใช้ชื่อตัวแปรว่า dummyDisplay ตามแนวทางการตั้งชื่อในหัวข้อที่ 4.5.1 จากนั้นจึงมอบสิทธิดังกล่าวให้กับหน้าจอที่ต้องการแสดงผล (2) ซึ่งเมธอด changePage() จะเรียกใช้งานเมธอด getDisplay() (3) โดยในการอิมพลีเมนต์จะเรียกเมธอด getDisplay() ของคลาส CentralManager อีกทีหนึ่ง (4) ลักษณะของการเรียกใช้งานเป็นทอดๆ เช่นนี้เพื่อเป็นการแยกระหว่างความสามารถที่คลาสควรจะมี กับการอิมพลีเมนต์ความสามารถนั้น ยกตัวอย่างเช่น คลาส UIManager มีหน้าที่ควบคุมตรรกะที่เกี่ยวกับการแสดงผลทั้งหมดจึงควรมีเมธอด getDisplay() ส่วนการอิมพลีเมนต์นั้นแพลตฟอร์มเจทูเอ็มอีบังคับให้ต้องขอสิทธิของการแสดงผลจากคลาส MIDlet เท่านั้น ดังเช่นแสดงในขั้นตอนที่ (4) โดย theLaucher คือชื่อตัวแปรอินสแตนซ์ของคลาส OneByteClientApp ซึ่งสืบทอดมาจากคลาส MIDlet (ดูรูปที่ 4.25)



รูปที่ 4.34 การเชื่อมโยงระหว่างแบบจำลองพีเอสเอ็มเชิงพฤติกรรมกับโค้ดที่สร้างได้ของกิจกรรม uploadData ของตัวรวบรวมข้อมูลระยะไกล

รูปที่ 4.34 แสดงตัวอย่างการสร้างโค้ดส่วนตรรกะหลักจากแผนภาพซีเควนซ์ของกิจกรรม uploadData (รูปที่ 4.21) ซึ่งเป็นการส่งข้อมูลจากเครื่องไคลเอนต์ไปยังเครื่องเซิร์ฟเวอร์ โดยเริ่มจากการประมวลผลข้อมูลเพื่อเตรียมอัปโหลด (1) จากนั้นจึงส่งข้อมูลที่ประมวลผลแล้วไปยังเครื่องเซิร์ฟเวอร์ (2) ซึ่งอิมพลีเมนต์ด้วยเมธอด upload() ของคลาส CommunicationManager (3) โดยมีการเรียกใช้เอพีไอที่เกี่ยวข้องกับการสร้างการเชื่อมต่อระหว่างเครื่องไคลเอนต์และเครื่องเซิร์ฟเวอร์ ซึ่งได้กล่าวแนะนำเอพีไอแต่ละตัวไปแล้วในหัวข้อที่ 2.1.6 ยกตัวอย่างการใช้งานเอพีไอ OutputStream.write() ซึ่งต้องการพารามิเตอร์ในการใช้งาน 3 ค่า ซึ่งจากแนวทางการตั้งชื่อตัวแปรที่ได้กล่าวไปในหัวข้อ 4.5.1 ทำให้ได้ชื่อของตัวแปรพารามิเตอร์เป็น arg1, arg2 และ arg3

ตามลำดับ พร้อมทั้งสร้างโค้ดส่วนของการตั้งค่าตัวแปรพารามิเตอร์ดังแสดงในกรอบสี่เหลี่ยมประ ในรูปที่ 4.34



UIManager

```
public void handleEvent(int globalEvent)
{
        switch (globalEvent)
        {
        // --------- handle MainPage Event ---------
        case GlobalEvent.MainPage_QuitApp:
                this.doMainPage_QuitApp();
                break;
        case GlobalEvent.MainPage_AddNewData:
                this.doMainPage_AddNewData();
                break;
        case GlobalEvent.MainPage_ProcessData:
                this.doMainPage_ProcessData();
                break;
        case GlobalEvent.MainPage_UploadData:
                this.doMainPage_UploadData();
                break;

        // --------- handle DataEntryPage Event ---------
        case GlobalEvent.DataEntryPage_SaveData:
                this.doDataEntryPage_SaveData();
                break;
        case GlobalEvent.DataEntryPage_QuitToMain:
                this.doDataEntryPage_QuitToMain();
                break;

        default:
                Notifier.debug("unpredicted event occurs");
        }
}
```


รูปที่ 4.35 ตัวอย่างโค้ดส่วนตรรกะหลักของคลาส UIManager ของตัวรวบรวมข้อมูลระยะไกล

รูปที่ 4.35 แสดงตัวอย่างการสร้างโค้ดส่วนตรรกะหลักจากแผนภาพแอคทิวิตี (รูปที่ 4.24) โดยในงานวิจัยชิ้นนี้เลือกใช้วิธีการจำลองการทำงานของสเตทแมชชีน โดยใช้เมธอดชื่อ handleEvent() ทำหน้าที่ในการตอบสนองต่อเหตุการณ์ทั้งหมดในระบบ

4.5.4 การสร้างโค้ดส่วนรองรับตรรกะหลัก

โค้ดส่วนรองรับตรรกะหลักนั้นถูกสร้างขึ้นจากหลักการที่ได้กล่าวไปแล้วในหัวข้อ 3.4.4 โดยได้แสดงตัวอย่างของการสร้างโค้ดส่วนรองรับตรรกะหลักดังรูปที่ 4.36 และรูปที่ 4.37

```
import javax.microedition.io.Connector;
import javax.microedition.io.HttpConnection;
import java.io.OutputStream;

import ISEL.OneByteClient.Notifier;

public class CommunicationManager
{
    private HttpConnection theHttpConnection;
    private OutputStream theOutputStream;

    public static void upload(ProcessedData p){

        try{
            theHttpConnection = (HttpConnection) Connector.open(url);
        }
        catch(Exception e){
            Notifier.debug(e.toString());
        }

        if (theHttpConnection != null)
        {
            this.configHttpConnection();

            theOutputStream = theHttpConnection.openOutputStream();

            byte[]      arg1 =   ;  // b
            int         arg2 =   ;  // off
            int         arg3 =   ;  // len
            theOutputStream.write(arg1,arg2,arg3);

            theOutputstream.flush();
        }
    }

    private void configHttpConnection(){
        theHttpConnection.setRequestMethod(HttpConnection.POST);
        theHttpConnection.setRequestProperty("User-Agent",
            "Profile/MIDP-1.0 Configuration/CLDC-1.0");
        theHttpConnection.setRequestProperty("Content-Language", "en-US");
        theHttpConnection.setRequestProperty("Content-Type","xml");
    }

}
```

รูปที่ 4.36 ตัวอย่างโค้ดส่วนรองรับตรรกะหลักของคลาส CommunicationManager ของตัวรวบรวมข้อมูลระยะไกล

จากรูปที่ 4.36 เป็นตัวอย่างการสร้างโค้ดส่วนรองรับตรรกะหลักของคลาส CommunicationManager โดยโค้ดส่วนที่มีการแรเงานั้นเป็นโค้ดส่วนตรรกะหลักซึ่งได้ในขั้นตอนที่ผ่านมา (ดังรูปที่ 4.34) ซึ่งจากโค้ดส่วนดังกล่าวจะสามารถสร้างโค้ดส่วนรองรับตรรกะหลักได้เช่น การนำเข้าไลบรารี (1) การประกาศชื่อและชนิดของตัวแปร (2), (8) การสร้างโค้ดส่วนจัดการเอ็กซ์เซปชัน (3) การเตรียมค่าอาร์กิวเมนต์ (6) และโค้ดที่เป็นไปตามรูปแบบการใช้งานเอพีไอ (4), (5) โดยสำหรับตัวอย่างนี้มีสมมติฐานว่าในคู่มือของการใช้งานคลาส HttpConnection ระบุว่าหลังจากการสร้างอินสแตนซ์ของคลาส HttpConnection ก่อนใช้งานต้องทำการคอนฟิกคอนเนคชันก่อนเสมอ ดังนั้นเครื่องมือสร้างโค้ดซึ่งสามารถรับรู้วิธีการใช้งานเอพีไอดังกล่าว จึงสามารถสร้างเมธอด configHttpConnection() ได้

```
package ISEL.OneByteClient;

import javax.microedition.rms.RecordStore;
import javax.microedition.rms.RecordStoreException;
import javax.microedition.rms.RecordStoreNotOpenException;

import ISEL.OneByteClient.DataEntity;
import ISEL.OneByteClient.Notifier;
import ISEL.OneByteClient.SystemConst;

public class StorageManager
{
	private RecordStore theRecordStore;
	private String recordStoreName = SystemConst.DATABASE_NAME;
	private boolean createIfNecessary = true;

	public StorageManager(){

		try{
			String arg1 = recordStoreName;		// recordStoreName
			boolean arg2 = createIfNecessary;	// createIfNecessary
			theRecordStore =
				RecordStore.openRecordStore(arg1,arg2);
		}
		catch(RecordStoreException e){
			Notifier.showException(e);
		}
	}

	public void addData(DataEntity d){

		int offsetByte = 0;
		try{
			byte[]		arg1 = DataEntity.serialize(d);	// data
			int		arg2 = offsetByte;		// offset
			int		arg3 = d.getStreamLength();	// numBytes
			theRecordStore.addRecord(arg1,arg2,arg3);
		}
		catch (RecordStoreException e){
			Notifier.showException(e);
		}
		catch (NullPointerException e){
			Notifier.showException(e);
		}
	}
}
```

รูปที่ 4.37 ตัวอย่างโค้ดส่วนรองรับตรรกะหลักของคลาส StorageManager ของตัวรวบรวมข้อมูลระยะไกล

จากรูปที่ 4.37 เป็นตัวอย่างการสร้างโค้ดส่วนรองรับตรรกะหลักของคลาส StorageManager ของฝั่งไคลเอนต์ โค้ดส่วนที่แรเงาคือโค้ดส่วนตรรกะหลัก ซึ่งสามารถนำไปสร้างโค้ดส่วนรองรับตรรกะหลักได้เช่น การประกาศชื่อและชนิดของตัวแปร (1) การนำเข้าไลบรารี (2) การเตรียมค่าตัวแปรอาร์กิวเมนต์ (3) การสร้างโค้ดส่วนจัดการเอ็กซ์เซปชัน (4) เป็นต้น

ในทางปฏิบัตินั้น โค้ดที่สร้างได้จากแบบจำลองโดยอัตโนมัติอาจไม่ใช่โค้ดที่ดีที่สุด ดังนั้นในการพัฒนาเครื่องมือสร้างโค้ด อาจต้องทำให้เครื่องมือมีความสามารถในการทำโค้ดให้เหมาะสมที่สุด (Optimization) เพื่อให้โค้ดที่สร้างได้มีประสิทธิภาพในการทำงานที่ดีขึ้น ยกตัวอย่างการทำโค้ดให้เหมาะสมในรูปที่ 4.38 ซึ่งตามตรรกะโดยแท้จริงแล้ว การใช้เมธอด getDisplay() เพื่อรับสิทธิในการแสดงผลบนหน้าจอ จะต้องไปเรียกใช้ theworker.getDisplay() ทุกครั้ง (ดังเช่นในรูปที่ 4.33) แต่เนื่องจากสิทธิดังกล่าวมีเพียงอินสแตนซ์เดียวและไม่เปลี่ยนแปลง ดังนั้นจึงได้ตั้งตัวแปร appDisplay มาจดจำอินสแตนซ์ดังกล่าวแทนทำให้ไม่ต้องเรียกใช้ theworker.getDisplay() ทุกครั้ง ซึ่งทำให้ประสิทธิภาพในการทำงานขั้นตอนดังกล่าวนี้ดีขึ้น ซึ่งถ้าเครื่องมือสร้างโค้ดมีความฉลาดเพียงพอและสามารถทำโค้ดให้เหมาะสมในทุกกรณี ก็จะทำให้ประสิทธิภาพโดยรวมของระบบดีขึ้น

```
public UIManager(CentralManager cm){

     this.theWorker = cm;
     appDisplay = this.theWorker.getDisplay();
     theDataEntryPage = new DataEntryPage(this);
     theMainPage = new MainPage(this);
}

public void changePage(Screen s){
     try
     {
          appDisplay.setCurrent(s);
     }
     catch(Exception e){
          Notifier.showException(e);
     }
}
```

รูปที่ 4.38 ตัวอย่างการปรับปรุงโค้ดให้เหมาะสมที่สุด

4.6 ระบบตัวอย่างที่ได้

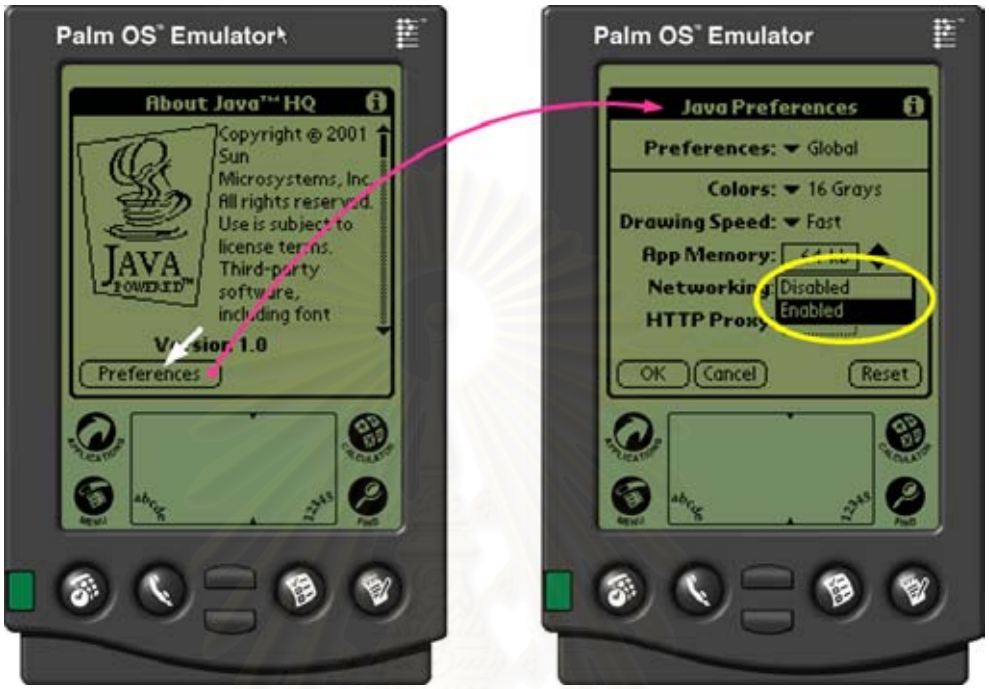
4.6.1 ตัวรวบรวมข้อมูลระยะไกล

รูปที่ 4.39 หน้าจอของเครื่องปาล์มหลังจากติดตั้งโปรแกรมลอตเตอรีออนไลน์แล้ว

เนื่องจากโปรแกรมลอตเตอรีออนไลน์ถูกพัฒนาด้วยแพลตฟอร์มภาษาจาวา ซึ่งในการทำงานของระบบจำเป็นต้องมีจาวาเวอร์ชวลแมชชีนอยู่บนเครื่องเสมอ ดังนั้นก่อนการติดตั้งโปรแกรมลอตเตอรีออนไลน์จึงจำเป็นต้องติดตั้งโปรแกรมจาวาเวอร์ชวลแมชชีนเสียก่อน ซึ่งโปรแกรมดังกล่าวสามารถดาวน์โหลดได้ฟรี [24] โดยเมื่อติดตั้งจาวาเวอร์ชวลแมชชีนสำหรับปาล์มแล้ว จะปรากฏไอคอนของโปรแกรมซึ่งมีชื่อว่า Java™HQ ดังแสดงในรูปที่ 4.39 เป็นการแสดงว่าติดตั้งสำเร็จแล้ว จากนั้นจึงค่อยทำการติดตั้งโปรแกรมลอตเตอรีออนไลน์ โดยแอพพลิเคชันสำหรับเครื่องปาล์มจะมีนามสกุลเป็น .prc ซึ่งหลังจากติดตั้งสำเร็จแล้วจะปรากฏไอคอนของโปรแกรมชื่อว่า OneByteClient ดังแสดงในรูปที่ 4.39

ขั้นตอนต่อไปคือการคอนฟิกโปรแกรมจาวาเวอร์ชวลแมชชีน ให้สามารถเชื่อมโยงกับเครือข่ายได้โดยเมื่อคลิกที่ไอคอนของจาวาเวอร์ชวลแมชชีนจะปรากฏหน้าจอดังรูปที่ 4.40 (ซ้าย)

จากนั้นให้คลิกที่ Preference จะปรากฏหน้าจอดังรูปที่ 4.40 (ขวา) แล้วทำการตั้งค่าของ “Networking” ให้เป็น “Enabled”



รูปที่ 4.40 การคอนฟิกจาวาเวอร์ชวลแมชชีนให้สามารถเชื่อมโยงกับเครือข่ายได้

รูปที่ 4.41 หน้าจอหลักของโปรแกรมลอตเตอรีออนไลน์

รูปที 4.41 แสดงหน้าจอแรกในการใช้งานโปรแกรมลอตเตอรีออนไลน์ โดยมีเมนูทางเลือก 3 ทางได้แก่ การป้อนข้อมูล (1) การประมวลผลข้อมูลก่อนอัปโหลด (2) และการอัปโหลดข้อมูล (3) ซึ่งเมื่อคลิกที่เมนูการป้อนข้อมูลจะปรากฏหน้าจอดังรูปที 4.42

รูปที 4.42 แสดงหน้าจอการป้อนข้อมูลซึ่งมีฟิลด์ของข้อมูลที่ต้องป้อนทั้งหมด 5 ฟิลด์ได้แก่ ชื่อของผู้ซื้อ (1) ชนิดของลอตเตอรี (2) เลขที่ซื้อ (3) จำนวนเงิน (4) บันทึกหมายเหตุ (5) โดยเมื่อป้อนข้อมูลเสร็จแล้วให้คลิกปุ่ม Save (7) ซึ่งหน้าจอจะถูกล้างและพร้อมสำหรับรับข้อมูลชุดใหม่ในทันทีโดยที่ข้อมูลเก่านั้นได้ถูกบันทึกไว้แล้ว ในกรณีที่การป้อนข้อมูลผิดพลาดต้องการป้อนใหม่ก็สามารถล้างหน้าจอเพื่อเตรียมป้อนข้อมูลใหม่โดยกดปุ่ม Clear (8) และสุดท้ายเมื่อป้อนข้อมูลจนหมดแล้วสามารถกดปุ่ม Main (6) เพื่อออกไปสู่หน้าจอหลัก

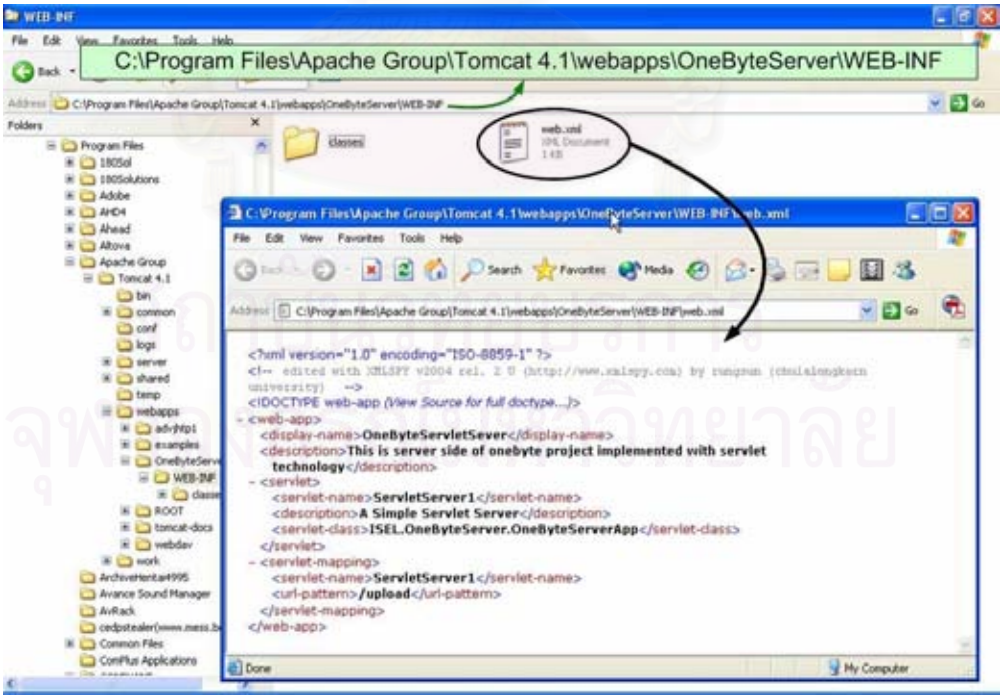
รูปที่ 4.42 หน้าจอการป้อนข้อมูลของโปรแกรมลอตเตอรีออนไลน์

4.6.2 ตัวเก็บข้อมูลส่วนกลาง

เนื่องจากโปรแกรมฝั่งเซิร์ฟเวอร์นั้นอิมพลีเมนต์ด้วยเจทูอีอี/เซิร์ฟเล็ต ที่ได้อธิบายหลักการทำงานไปแล้วในหัวข้อที 2.1.7 ซึ่งสรุปได้ว่าเซิร์ฟเล็ตนั้นต้องทำงานในสภาพแวดล้อมของเซิร์ฟเล็ต

คอนเทนเนอร์ซึ่งในงานวิจัยนี้เลือกใช้โปรแกรมอาปาเช่ทอมแคท ซึ่งเป็นเซิร์ฟเล็ตคอนเทนเนอร์ตัวหนึ่ง โดยเมื่อติดตั้งทอมแคทแล้วจะปรากฏชอร์ทคัทที่เมนูสตาร์ทดังรูปที่ 4.43

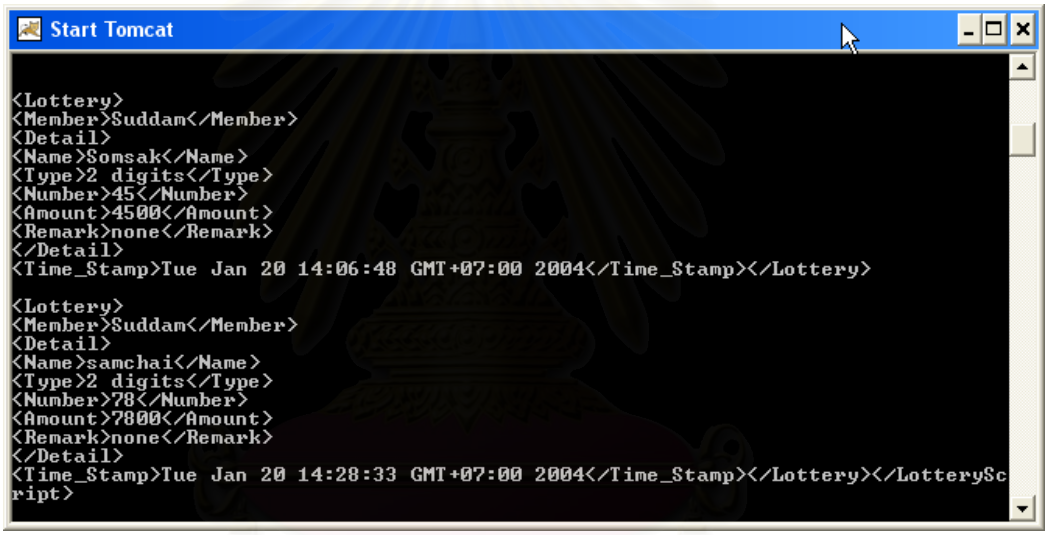
รูปที่ 4.43 ชอร์ทคัทของโปรแกรมอาปาเช่ทอมแคทที่ปรากฏหลังจากติดตั้งโปรแกรมแล้ว



รูปที่ 4.44 แสดงการติดตั้งโปรแกรมลอตเตอรีออนไลน์ฝั่งเซิร์ฟเวอร์บนโปรแกรมทอมแคท

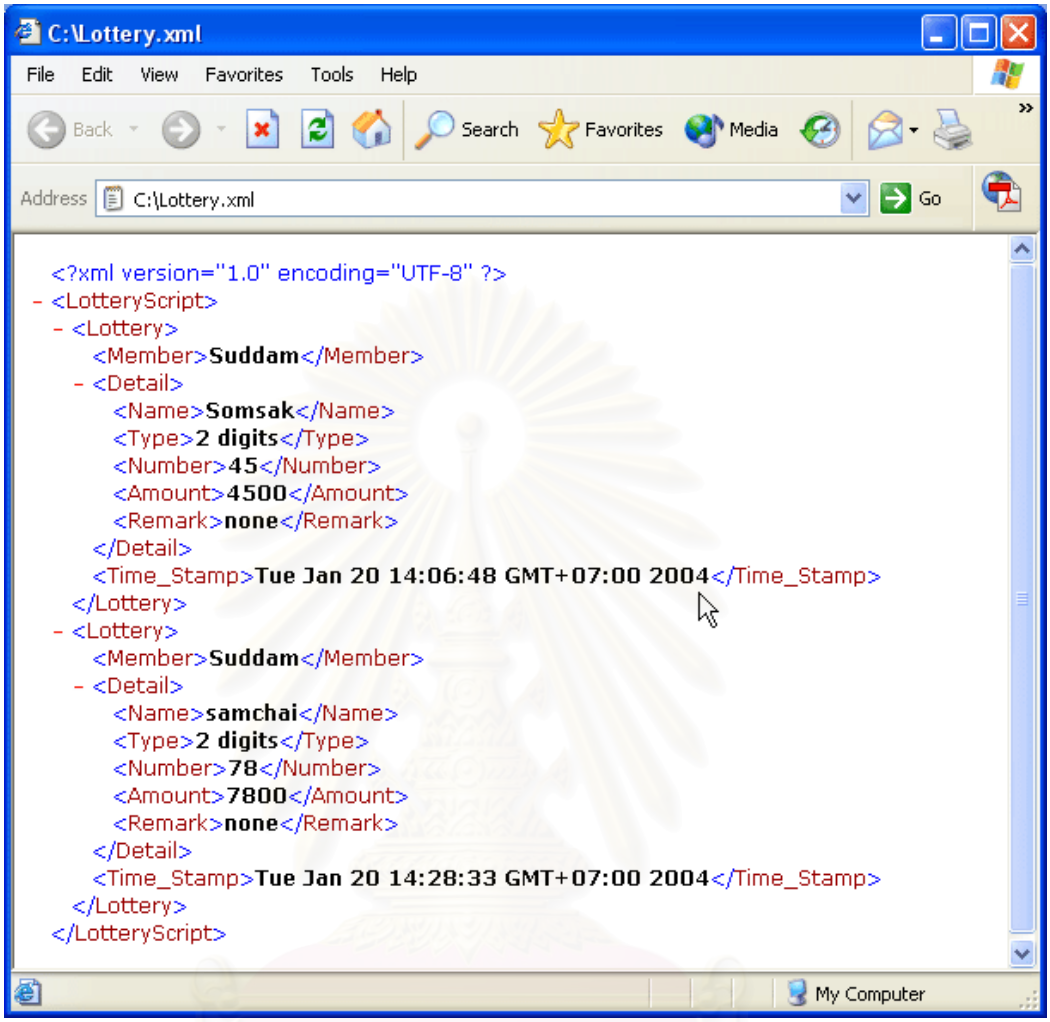
สำหรับการติดตั้งโปรแกรมลอตเตอรีออนไลน์ฝั่งเซิร์ฟเวอร์นั้น ทำโดยการสร้างโฟลเดอร์ของโปรแกรมลงในโฟลเดอร์ webapps ของทอมแคท จากนั้นสร้างโฟลเดอร์ชื่อ WEB-INF ซึ่งภายในบรรจุไฟล์ web.xml ซึ่งเป็นคอนฟิกกูเรชันไฟล์สำหรับแมปยูอาร์แอลกับตำแหน่งจริงของคลาสแอพพลิเคชันเซิร์ฟเล็ต ซึ่งมีเนื้อหาดังแสดงในรูปที่ 4.44 ส่วนโฟลเดอร์ classes เป็นโฟลเดอร์ที่บรรจุไฟล์คลาสต่างๆ ของแอพพลิเคชัน

เมื่อสั่งให้โปรแกรมทอมแคททำงาน จะปรากฏหน้าจอคอนโซลของโปรแกรมทอมแคทแสดงว่าโปรแกรมพร้อมรับคำร้องขอจากเครื่องไคลเอนต์แล้ว ซึ่งหากมีคำร้องขอมาที่เครื่องเซิร์ฟเวอร์ เซิร์ฟเวอร์จะบันทึกข้อมูลที่ได้รับมาพร้อมทั้งพิมพ์ข้อมูลออกทางคอนโซลดังแสดงในรูปที่ 4.45



รูปที่ 4.45 หน้าต่างคอนโซลของโปรแกรมทอมแคทที่พิมพ์ข้อมูลที่รับได้ออกมา

สำหรับผลลัพธ์ทั้งหมดของการประมวลผลข้อมูลของฝั่งเซิร์ฟเวอร์นั้น จะถูกเก็บอยู่ในไฟล์อักขระที่ชื่อ Lottery.xml ซึ่งอยู่ในโฟลเดอร์ C:\ เนื่องจากชื่อไฟล์ดังกล่าวเป็นค่าคงที่ของระบบซึ่งกำหนดในคลาส SystemConst ดังนั้นในกรณีนี้จึงไม่สามารถเปลี่ยนแปลงไฟล์ปลายทางของผลลัพธ์การประมวลผลได้ ดังแสดงตัวอย่างไฟล์ผลลัพธ์ดังกล่าวในรูปที่ 4.46



รูปที่ 4.46 ตัวอย่างของไฟล์ผลลัพธ์การประมวลผลของโปรแกรมฝั่งเซิร์ฟเวอร์

บทที่ 5
ข้อคิดเห็นเกี่ยวกับวิธีการพัฒนาระบบโดยใช้เอ็มดีเอ

5.1 ความพร้อมของการใช้งานเทคโนโลยีเอ็มดีเอ

5.1.1 ความพร้อมของมาตรฐานในปัจจุบัน

เอ็มดีเอได้กำหนดกรอบการทำงานของการพัฒนาระบบสารสนเทศ โดยอาศัยเทคโนโลยีของการสร้างโมเดล ซึ่งสามารถแบ่งได้เป็น 2 ส่วน

1. การสร้างโมเดล ซึ่งเอ็มดีเอได้ให้แนวทางโดยการแบ่งโมเดลออกเป็นสองส่วนได้แก่ พีไอเอ็ม และพีเอสเอ็ม ซึ่งมาตรฐานที่รองรับการสร้างโมเดลในส่วนนี้ได้แก่
   1.1. ยูเอ็มแอล ซึ่งจากข้อกำหนดของภาษานั้น เป็นภาษาที่ใช้สร้างแบบจำลองสำหรับงานทั่วไป (General Purpose) แต่ในทางปฏิบัติมักพบว่าภาษายูเอ็มแอลจะเหมาะสมต่อการใช้ออกแบบระบบซอฟต์แวร์เชิงวัตถุมากว่าการนำไปใช้งานในวัตถุประสงค์อื่นๆ
   1.2. ยูเอ็มแอลโปรไฟล์สำหรับการสร้างแบบจำลองของแต่ละโดเมน (UML Profile for Domain Modeling) หมายถึง การขยายภาษายูเอ็มแอลให้สามารถสื่อความหมายที่เฉพาะเจาะจงกับสายงาน (แต่ไม่ได้ยึดติดกับเทคโนโลยีของแพลตฟอร์ม) โดยมีการกำหนดเมตาโมเดลขึ้นมาเฉพาะสำหรับงานนั้นๆ โดยอาจมีการกำหนดโนเทชันพิเศษ (Concrete Syntax) ทำหน้าที่เป็นภาษาสำหรับการเขียนโมเดลนั้นๆ หรือไม่ก็ได้ ตัวอย่างของยูเอ็มแอลโปรไฟล์ดังกล่าวที่มีในปัจจุบันได้แก่
      1.2.1. ยูเอ็มแอลโปรไฟล์สำหรับอีด็อก [10] สำหรับการออกแบบระบบงานที่มีลักษณะเป็นระบบประมวลผลแบบกระจาย โดยจะเน้นที่การแสดงส่วนประกอบ และการเชื่อมต่อของแต่ละหน่วยทำงาน นอกจากนั้นยังมีการกำหนดซีซีเอโนเทชัน ซึ่งเป็นภาษาพิเศษเพื่อใช้เขียนแสดงความหมายของซีซีเอโปรไฟล์ ซึ่งเป็นโปรไฟล์ย่อยในอีด็อก
      1.2.2. ยูเอ็มแอลโปรไฟล์สำหรับอีเอไอ [11] สำหรับการออกแบบระบบงานที่เป็นการเชื่อมโยงระบบงานต่างๆ ขององค์กร
      1.2.3. ยูเอ็มแอลโปรไฟล์สำหรับข้อกำหนดเชิงประสิทธิภาพและเวลา [25] สำหรับการออกแบบระบบประมวลผลแบบทันที (Real-time System)

1.2.4. ซีดับเบิลยูเอ็ม [26] สำหรับการออกแบบโมเดลของคลังข้อมูล (Data Warehouse)

1.3. ยูเอ็มแอลโปรไฟล์สำหรับการสร้างแบบจำลองที่ขึ้นกับเทคโนโลยีของแพลตฟอร์ม หมายถึง การขยายภาษายูเอ็มแอลให้สามารถสื่อความหมายที่เฉพาะเจาะจงกับคอมโพเนนท์ที่มีอยู่จริงในแพลตฟอร์มนั้นๆ ซึ่งทำให้แบบจำลองที่ได้มีความใกล้เคียงกับการอิมพลีเมนต์จริงมากที่สุด ยกตัวอย่างยูเอ็มแอลโปรไฟล์ประเภทนี้เช่น

1.3.1. ยูเอ็มแอลโปรไฟล์สำหรับคอร์บา [8] ซึ่งเป็นโปรไฟล์ที่ใช้แสดงคอนเซปต์และคอมโพเนนท์ต่างๆ ที่มีอยู่ในแพลตฟอร์มคอร์บา

1.3.2. ยูเอ็มแอลโปรไฟล์สำหรับอีเจบี ซึ่งเป็นโปรไฟล์ที่ใช้แสดงคอนเซปต์ และคอมโพเนนท์ต่างๆ ที่มีอยู่ในแพลตฟอร์มอีเจบี

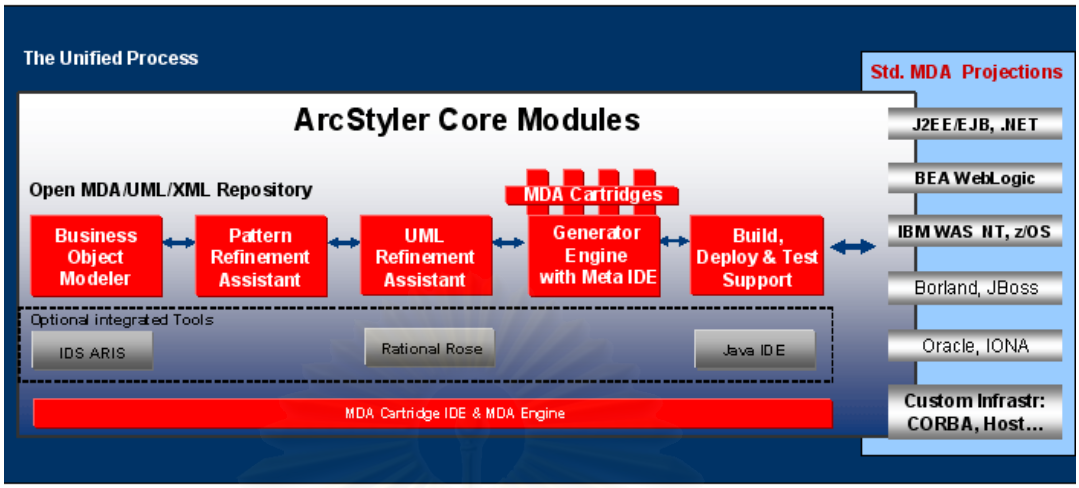
2. การจัดการโมเดล

2.1. เอ็มโอเอฟ ใช้สำหรับการอธิบายความหมาย และกำหนดวากยสัมพันธ์เชิงนามธรรมของโครงสร้างโมเดลต่างๆ ดังที่ได้อธิบายไปแล้วในหัวข้อ 2.1.4

2.2. เอ็กซ์เอ็มไอ ใช้สำหรับการแสดงแบบจำลองใดๆ ในรูปอักขระตามรูปแบบของภาษาเอ็กซ์เอ็มแอล ซึ่งทำให้การอ่านโมเดลโดยเครื่องมือต่างๆ สามารถทำได้ง่ายเนื่องจากสามารถประยุกต์ใช้เครื่องมือประมวลผลภาษาเอ็กซ์เอ็มแอลมาใช้ได้ทันที

5.1.2 ความพร้อมของเครื่องมือในปัจจุบัน

สำหรับความพร้อมด้านเครื่องมือต่างๆ ในปัจจุบันนั้น ถึงแม้จะมีจำนวนเครื่องมือให้เลือกค่อนข้างมาก โดยต่างอ้างว่ารองรับการพัฒนาด้วยเอ็มดีเอ ยกตัวอย่างเช่น เรชันนอลโรส เอ็กซ์ดีอี (Rational Rose XDE) อาร์คสไตเลอร์ (ArcStyler) ออบเจ็กต์เทียริง (Objecteering) โพไซดอน (Poseidon) โคดาเจ็น อาร์คิเท็ค (Codagen Architect) ออพติมอลเจ (OptimalJ) อิคลิปส์จีเอ็มที (Eclipse's GMT) และแมจิกดรอว์ (MagicDraw) เป็นต้น อย่างไรก็ตามจากการศึกษาเอกสารของผลิตภัณฑ์บางส่วนนั้น ผู้วิจัยเห็นว่าเครื่องมือทั้งหมดนั้นยังมิได้เป็นเครื่องมือพัฒนาระบบด้วยเอ็มดีเออย่างสมบูรณ์ แต่เป็นเพียงเครื่องมือสร้างส่วนของโค้ดจากแผนภาพยูเอ็มแอลเท่านั้น โดยทำได้เพียงการสร้างโค้ดส่วนโครงสร้างจากแผนภาพคลาสเท่านั้น ยังไม่พบเครื่องมือใดจากรายการข้างต้นที่สามารถสร้างโค้ดจากแผนภาพที่แสดงพฤติกรรมของระบบได้



รูปที่ 5.1 ส่วนประกอบของโปรแกรมอาร์คสไตเลอร์

ในงานวิจัยนี้ได้เลือกศึกษาเครื่องมือชื่ออาร์คสไตเลอร์ซึ่งทางโอเอ็มจีกล่าวว่าเป็นเครื่องมือที่รองรับการพัฒนาระบบด้วยเอ็มดีเอที่ดีที่สุดตัวหนึ่งในปัจจุบัน โดยเครื่องมือดังกล่าวมีส่วนประกอบดังรูปที่ 5.1 ซึ่งมีส่วนประกอบหลักๆ ได้แก่ บิสิเนสออบเจ็กต์โมเดลเลอร์ (Business Object Modeler) ซึ่งเป็นเครื่องมือสร้างแบบจำลองพีไอเอ็มเชิงธุรกิจของระบบ จากนั้นจึงนำมาเพิ่มรายละเอียดของแบบจำลองด้วยเครื่องมือ แพทเทิร์นรีไฟน์เมนต์แอสซิสแทนท์ (Pattern Refinement Assistant) แล้วนำไปตรวจสอบความถูกต้องและความสอดคล้องของแบบจำลองด้วยเครื่องมือ ยูเอ็มแอลรีไฟน์เมนต์แอสซิสแทนท์ (UML Refinement Assistant) แล้วจึงนำแบบจำลองไปสร้างโค้ดด้วยเจเนอเรเตอร์เอ็นจิน (Generator Engine) และนำไปใช้งาน (Deploy) ด้วยเครื่องมือสนับสนุนการสร้าง นำไปใช้งาน และการทดสอบ (Build Deploy and Test Support) นอกจากนั้นยังมีการกำหนดเอ็มดีเอคอมโพเนนท์เรียกว่าเอ็มดีเอคาร์ทริดจ์ (MDA Cartridges) ซึ่งทางผู้ผลิตกล่าวว่าเป็นเอ็มดีเอคอมโพเนนท์ที่มีความสมบูรณ์ในตัวเองกล่าวคือสามารถถูกนำไปประมวลผลได้ทันทีในลักษณะที่เป็นมอดูลพลักแอนด์เพลย์ (Plug and Play Module)

อย่างไรก็ตามจากการทดสอบโปรแกรมอาร์คสไตเลอร์ในเวอร์ชันทดลอง (Trial Version) ซึ่งมีข้อจำกัดในการใช้งานที่ค่อนข้างมาก เช่นมีส่วนประกอบไม่ครบตามที่แสดงไว้ในรูปที่ 5.1 ยกตัวอย่างเช่น ไม่มีเครื่องมือบิสิเนสออบเจ็กต์โมเดลเลอร์ ทำให้ไม่ทราบถึงวิธีการจัดการของเครื่องมือเกี่ยวกับการสร้างแบบจำลองพีไอเอ็มของระบบ นอกจากนั้นยังไม่พบว่าเครื่องมือดังกล่าวจะสามารถรองรับการแปลงระหว่างแบบจำลองได้อย่างไร และไม่มีเอกสารใดที่บอกถึง

วิธีการจัดการเกี่ยวกับเมตาโมเดล โดยส่วนของเครื่องมือที่ได้มาทดสอบนั้นเป็นเพียงส่วนของการสร้างโค้ดส่วนโครงสร้างของแพลตฟอร์มปลายทาง (เช่น อีเจบี) จากแผนภาพคลาสเท่านั้น ถึงแม้ว่าเอกสารคู่มือของเครื่องมือจะระบุว่าเครื่องมือรองรับการสร้างโค้ด จากแผนภาพสเตทชาร์ตก็ตาม แต่อาจเนื่องจากข้อจำกัดของเครื่องมือซึ่งเป็นเวอร์ชันทดสอบทำให้ไม่สามารถทดสอบความสามารถดังกล่าวของเครื่องมือได้ นอกจากนั้นยังพบว่าการใช้งานเอ็มดีเอคาร์ทริดจ์นั้นค่อนข้างยุ่งยากมาก กล่าวคือต้องมีการกำหนดค่าคอนฟิกกูเรชันต่างๆ เป็นจำนวนมาก รวมทั้งการตั้งค่าสภาพแวดล้อมต่างๆ ให้กับเอ็มดีเอคอมโพเนนท์ดังกล่าวก่อนใช้งาน ดังนั้นผู้วิจัยมีความเห็นว่าเครื่องมือดังกล่าวยังไม่มีความเหมาะสม คล่องตัว และสมบูรณ์เพียงพอในการรองรับการพัฒนาระบบตามแนวคิดเอ็มดีเอ

5.2 อุปสรรคของการใช้งานเทคโนโลยีเอ็มดีเอ

5.2.1 มาตรฐานที่มีอยู่ยังไม่สมบูรณ์เพียงพอในการใช้งานจริง

ถึงแม้ว่าในปัจจุบันจะมีการกำหนดมาตรฐาน ที่เกี่ยวกับการพัฒนาแบบจำลองตามแนวคิดเอ็มดีเอแล้วส่วนหนึ่งตามที่กล่าวไปในหัวข้อที่ 5.1.1 อย่างไรก็ตามในการใช้งานจริงนั้น มาตรฐานทีมีอยู่อาจไม่เพียงพอเนื่องจาก ในการพัฒนาระบบจริง นักออกแบบมักต้องการเมตาโมเดลที่ใช้สื่อความหมายที่เฉพาะเจาะจงกับงานของตน ยกตัวอย่างเช่น ในงานวิจัยชิ้นนี้มีการกำหนด <<UserAction>> (ดูรูปที่ 4.10) ซึ่งหมายถึงเป็นขั้นตอนหนึ่งในกระบวนการของระบบแต่เป็นหน้าทีของผู้ใช้งานระบบต้องเป็นผู้ทำ เช่นการป้อนข้อมูลเข้าเครื่องปาล์มเป็นต้น ซึ่งการสื่อความหมายที่เฉพาะเจาะจงดังกล่าวนี้ ยูเอ็มแอล และยูเอ็มแอลโปรไฟล์ที่มีอยู่นั้นไม่สามารถใช้ได้ เป็นต้น

5.2.2 ยังไม่มีภาพทีชัดเจนของกระบวนพัฒนาระบบทั้งหมดตั้งแต่ต้นจนจบสมบูรณ์

ถึงแม้ว่าโอเอ็มจีจะให้แนวทางในการพัฒนาระบบมาก็ตาม แต่แนวทางดังกล่าวเป็นเพียงแนวทางคร่าวๆ ซึ่งยังขาดรายละเอียดในทางปฏิบัติอีกมากทำให้ผู้นำไปใช้งานมักเกิดความสับสนในการนำแนวคิดเอ็มดีเอไปใช้งานจริง ยกตัวอย่างเช่น แบบจำลองพีไอเอ็มต้องประกอบด้วยแผนภาพอะไรบ้าง จำนวนกี่แผนภาพ ต้องมีรายละเอียดเพียงใด หรือถ้าจะไม่กำหนดรายละเอียดใดๆ เลยในระดับพีไอเอ็มแล้วผลักให้เป็นภาระในระดับพีเอสเอ็มสามารถทำได้หรือไม่ ขีดแบ่งทีชัดเจนของพีไอเอ็มและพีเอสเอ็มคืออะไร ยกตัวอย่างเช่นการกำหนดรูปแบบการแลกเปลี่ยนข้อมูลว่าต้องเป็นรูปแบบของภาษาเอ็กซ์เอ็มแอล ในมุมหนึ่งก็อาจมองได้ว่าอยู่ในระดับพีเอสเอ็มเพราะ

เป็นการระบุว่าข้อมูลต้องมีการอิมพลีเมนต์เป็นเอ็กซ์เอ็มแอล แต่อีกมุมหนึ่งก็อาจมองว่าอยู่ในระดับพีไอเอ็มได้ เพราะไม่ได้กำหนดว่าต้องใช้คอมโพเนนท์แบบใดในการอิมพลีเมนต์การประมวลผลเอกสาร นอกจากนั้นส่วนที่สำคัญที่สุดของกระบวนการพัฒนาระบบด้วยเอ็มดีเอนั้นกลับเป็นส่วนที่มีความชัดเจนน้อยที่สุดได้แก่ การแปลงระหว่างแบบจำลองคนละชนิด หรือคนละระดับ และการแปลงแบบจำลองเป็นโค้ด

5.2.3 ความน่าเชื่อถือของโค้ดที่ได้จากการสร้างโดยอัตโนมัติตามแนวคิดเอ็มดีเอ

ถึงแม้ว่าจะสามารถพัฒนากระบวนการพัฒนาระบบตามแนวคิดเอ็มดีได้จนสมบูรณ์แล้วก็จะยังคงมีปัญหาเรื่องความน่าเชื่อถือของโค้ดที่สร้างขึ้นได้ เนื่องจากโค้ดถูกสร้างจากกระบวนการอัตโนมัติด้วยกฎการแปลงที่ค่อนข้างซับซ้อน และจากแบบจำลองที่ซับซ้อนของระบบขนาดใหญ่ทำให้การตรวจสอบความถูกต้องของโค้ดที่สร้างได้เป็นเรื่องที่ยากและควรจะต้องคำนึงถึงด้วย โดยประเด็นดังกล่าวเกี่ยวข้องกับอีก 2 ประเด็นย่อยได้แก่ ความถูกต้องและสมบูรณ์ของแบบจำลองซึ่งในปัจจุบันยังไม่มีมาตรฐานใดที่สามารถตรวจสอบความถูกต้อง ความสมบูรณ์ รวมถึงความสอดคล้องระหว่างมุมมองต่างๆ ของแบบจำลองได้ อีกประเด็นคือความถูกต้องของกฎการแปลงซึ่งรวมถึงแนวคิดสำคัญของเอ็มดีเอที่ว่ากฎการแปลงสามารถนำกลับมาใช้ใหม่ได้เสมอนั้นเป็นจริงหรือไม่ ซึ่งถ้าแนวคิดดังกล่าวไม่สามารถเป็นจริงได้อาจทำให้การนำเอ็มดีเอมาใช้จริงนั้น ไม่ประสบความสำเร็จก็เป็นได้ เนื่องจากต้องสร้างกฎการแปลงขึ้นใหม่ทุกๆ ครั้งในการพัฒนาระบบ

5.2.4 ขาดเครื่องมือที่มีความสมบูรณ์ที่สามารถครอบคลุมการพัฒนาทั้งกระบวนการ

ความสำเร็จของการใช้งานเอ็มดีเอนั้น อยู่ที่ความสามารถของเครื่องมือในการแปลงแบบจำลองและโค้ดอย่างอัตโนมัติได้มากที่สุด ทำให้ประสิทธิภาพของกระบวนการพัฒนาระบบสูงขึ้น ในขณะที่ค่าใช้จ่ายนั้นลดลง ดังนั้นความสามารถของเครื่องมือจึงมีผลต่อความสำเร็จของแนวคิดเอ็มดีเอเป็นอย่างมาก อย่างไรก็ตามในปัจจุบันนั้นเครื่องมือทั้งหมดยังไม่มีความสามารถในระดับที่เอ็มดีเอต้องการ ทำให้การนำแนวคิดเอ็มดีเอมาใช้ในการการออกแบบและพัฒนาระบบจริงนั้น ยังเป็นเรื่องที่ยากและอาจไม่เพิ่มประสิทธิภาพแต่เพิ่มค่าใช้จ่ายในการพัฒนาระบบอีกด้วย

5.3 แนวทางการวิจัยในอนาคต

5.3.1 ศึกษาและเปรียบเทียบความสำเร็จของเอ็มดีดี (MDD: Model Driven Development) จากแนวทางอื่นๆ

โดยแท้จริงแล้วนอกจากเอ็มดีเอของโอเอ็มจีแล้ว แนวคิดในการพัฒนาระบบโดยอาศัยการพัฒนาแบบจำลองนั้นมีมานานแล้วโดยเฉพาะในงานวิศวกรรมสาขาอื่นๆ เช่นโยธา เครื่องกล หรืออิเล็กทรอนิกส์ ล้วนพัฒนาระบบโดยมีพื้นฐานจากการพัฒนาแบบจำลองทั้งสิ้น ยกตัวอย่างเช่นในการออกแบบระบบทางอิเล็กทรอนิกส์นั้น เริ่มจากการกำหนดความต้องการของระบบ แล้วจึงกำหนดส่วนประกอบของระบบว่าควรประกอบด้วยวงจรใดบ้าง เพื่อทำหน้าที่ใด แล้วจึงสร้างแบบจำลองโดยการวาดสกีมาติก (Schematic) ของวงจร ซึ่งสกีมาติกดังกล่าวนั้นท้ายสุดแล้วจะถูกแปลงสู่ระดับอิมพลีเมนต์ ซึ่งอาจเป็นการอิมพลีเมนต์ด้วยวงจรรวม (IC: Integrated Circuit) และเชื่อมต่อกันด้วยลายวงจรบนแผ่นพีซีบี (PCB: Printed Circuit Board) หรือการสังเคราะห์วงจรขึ้นเองโดยอาศัยเทคโนโลยีของเอฟพีจีเอ (FPGA: Field Programmable Gate Array) เป็นต้น ซึ่งกระบวนการดังกล่าวนี้เป็นตัวอย่างหนึ่งของการพัฒนาระบบแบบเอ็มดีดีอย่างเต็มรูปแบบ ดังนั้นจึงอาจสามารถเทียบเคียงกระบวนการดังกล่าวมาใช้ในการพัฒนาระบบซอฟต์แวร์ด้วยเอ็มดีเอได้

5.3.2 พัฒนายูเอ็มแอลโปรไฟล์โดยอ้างอิงกับภาษาโมเดลอื่นๆ

ข้อจำกัดอีกประการหนึ่งของเอ็มดีเอในการสร้างแบบจำลองนั้น คือแบบจำลองทุกๆ แบบจำลองตามแนวคิดของเอ็มดีเอต้องสอดคล้องกับสถาปัตยกรรมเอ็มโอเอฟ ดังที่ได้กล่าวไว้ในหัวข้อที่ 3.2.2.2 กล่าวคือสิ่งที่แบบจำลองจะแสดงออกต้องมีการกำหนดความหมายและวากยสัมพันธ์เชิงนามธรรมไว้ในระดับเมตาโมเดล กล่าวคือทุกโมเดลในกรอบของเอ็มดีเอจำเป็นต้องมีเมตาโมเดลรองรับ ดังนั้นภาษาโมเดลอื่นๆ ซึ่งไม่สอดคล้องเงื่อนไขดังกล่าวนี้จึงไม่สามารถใช้งานกับเอ็มดีเอได้ ยกตัวอย่างเช่นบีเพล (BPEL: Business Process Execution Language) หรืออีบีเอ็กซ์เอ็มแอล (ebXML: e-Business XML) ซึ่งเป็นภาษาที่ได้รับความนิยมในการออกแบบระบบกระแสงาน แต่อาจมีความสามารถในการแสดงความหมายในงานเฉพาะเจาะจงบางอย่างได้ดีกว่าภาษายูเอ็มแอล ซึ่งทำให้การออกแบบระบบทำได้ง่ายขึ้น ดังนั้นการทำให้เอ็มดีเอรองรับภาษาดังกล่าว (เช่น อาจกำหนดเป็นยูเอ็มแอลโปรไฟล์สำหรับ บีเพล เป็น

ต้น) น่าจะทำให้การสร้างแบบจำลองของระบบกระแสงาน ทำได้ง่ายขึ้นและแบบจำลองมีความสมบูรณ์มากขึ้น

5.3.3 การเพิ่มรายละเอียดเชิงข้อกำหนดโดยการประยุกต์ใช้ภาษาโอซีแอล

ในการกำหนดรายละเอียดเชิงข้อจำกัด (Constraint) ให้กับแบบจำลองที่สร้างขึ้นด้วยภาษายูเอ็มแอลนั้น โอเอ็มจีได้กำหนดมาตรฐานภาษาโอซีแอล [27](OCL: Object Constraint Language) โดยมีลักษณะของการกำหนดเงื่อนไขก่อนทำงาน (Pre-Condition) เงื่อนไขหลังทำงาน (Post-Condition) และข้อจำกัดระหว่างการทำงาน (Constraint) ซึ่งทำให้แบบจำลองที่ได้มีการสื่อความหมายที่รัดกุมยิ่งขึ้น ซึ่งเมื่อแบบจำลองมีความสมบูรณ์มากขึ้น ดังนั้นโค้ดที่สร้างได้จากแบบจำลองดังกล่าวก็น่าจะมีความสมบูรณ์มากขึ้นเช่นกัน

5.3.4 การเพิ่มรายละเอียดเกี่ยวกับการประมวลผลโดยใช้ภาษาแอคชัน

ในข้อกำหนดของภาษายูเอ็มแอลเวอร์ชัน 1.5 ได้มีการเพิ่มข้อกำหนดของภาษาแอคชัน (Action Language) ซึ่งมีกลไกสำหรับอธิบายพฤติกรรมระดับของการประมวลผลระบบเชิงวัตถุ ยกตัวอย่างเช่น มีการกำหนดกลไกของภาษาที่ใช้อธิบายการสร้าง และทำลายวัตถุ การกำหนดเงื่อนไขและเลือกทางเลือกในการประมวลผล การเรียกใช้งานโอเปอร์เรชัน การส่งอีเวนต์ระหว่างวัตถุ ซึ่งโดยรวมแล้วทำให้แบบจำลองมีรายละเอียดในระดับเทียบเท่ากับภาษาโปรแกรมแต่ไม่ยึดติดกับภาษาใดภาษาหนึ่ง ซึ่งตรงตามจุดประสงค์ของการสร้างแบบจำลองพีไอเอ็ม

5.3.5 การประยุกต์ใช้หลักการของดีไซน์แพทเทิร์นในการสร้างแบบจำลอง

เนื่องจากระบบงานทั่วๆ ไปนั้นมักมีความต้องการพื้นฐานบางอย่างที่เหมือนกัน ดังนั้นในการออกแบบมักพบว่าบ่อยครั้งระบบงานต่างกันกับมีการออกแบบที่คล้ายกัน ซึ่งหลักการดังกล่าวนั้นคือหลักการของดีไซน์แพทเทิร์น [28] นั่นเอง ซึ่งการนำหลักการของดีไซน์แพทเทิร์นมาใช้ในเอ็มดีเอนั้นน่าจะเป็นประโยชน์มาก เนื่องจากช่วยในการออกแบบพีไอเอ็มและพีเอสเอ็มได้อย่างรวดเร็ว และเมื่อมีการนำกลับมาใช้ใหม่ของแพทเทิร์นต่างๆ การแปลงแบบจำลองดังกล่าวไปเป็นโค้ดก็น่าจะมีแพทเทิร์นเช่นเดียวกัน ทำให้การนำกฎการแปลงดังกล่าวกลับมาใช้ใหม่มีความเป็นไปได้สูงขึ้น

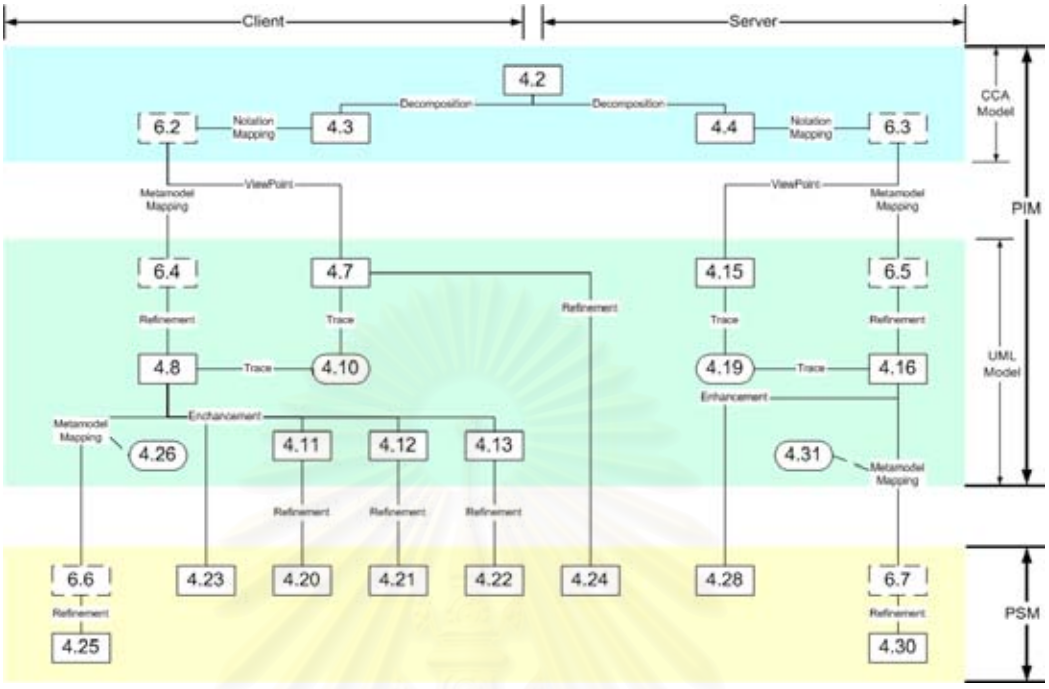
บทที่ 6
สรุปผลการวิจัย และข้อเสนอแนะ

จากการศึกษาแนวคิดการพัฒนาระบบด้วยเอ็มดีเอพบว่า เป็นแนวคิดที่ช่วยแก้ปัญหาเรื่องความสามารถในการทำงานร่วมกันระหว่างแพลตฟอร์มที่ต่างกันของระบบงานต่างๆ นอกจากนั้นยังช่วยให้ระบบมีความยืดหยุ่นสูงต่อการเปลี่ยนแปลงของเทคโนโลยีในอนาคตอีกด้วย อีกทั้งการพัฒนาระบบโดยมีพื้นฐานจากการพัฒนาแบบจำลอง ทำให้ระบบที่ได้น่าจะมีคุณภาพดีขึ้นและมีข้อผิดพลาดน้อยลง ใช้ทรัพยากรในการพัฒนาระบบน้อยลง ซึ่งประโยชน์ต่างๆ เหล่านี้ล้วนช่วยให้การพัฒนาอุตสาหกรรมซอฟต์แวร์ก้าวหน้าไปอย่างรวดเร็ว แต่อย่างไรก็ตามในปัจจุบันการนำแนวคิดเอ็มดีเอไปใช้งานจริงยังมีปัญหาอยู่ เนื่องจากการขาดความชัดเจนเชิงกระบวนการ และข้อกำหนดในหลายประการ ดังนั้นในงานวิทยานิพนธ์นี้จึงมีจุดประสงค์เพื่อศึกษาการพัฒนาระบบด้วยแนวคิดเอ็มดีเอ โดยพยายามศึกษาขั้นตอนการพัฒนาระบบทั้งกระบวนการตั้งแต่ต้นจนจบ และเพื่อเป็นการทดสอบแนวคิดที่ได้ศึกษามา จึงมีการทดลองพัฒนาระบบงานจริงด้วยแนวคิดดังกล่าวอีกด้วย โดยระบบตัวอย่างที่พัฒนาขึ้นเป็นระบบการส่งข้อมูลลอตเตอรีออนไลน์จากเครื่องไคลเอนต์ซึ่งเป็นเครื่องปาล์มไปยังเครื่องเซิร์ฟเวอร์ซึ่งเป็นเครื่องพีซี โดยการพัฒนาระบบจะเริ่มตั้งแต่การสร้างแบบจำลองพีไอเอ็ม จนได้ออกมาเป็นโค้ดที่เสร็จสมบูรณ์ ซึ่งสามารถสรุปผลที่ได้จากการวิจัย อภิปรายผลการวิจัย รวมถึงข้อเสนอแนะต่างๆ ดังนี้

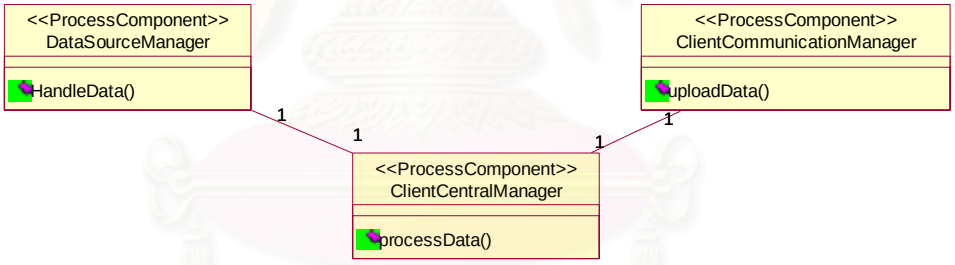
6.1 สรุปผลการวิจัย

6.1.1 การพัฒนาระบบตัวอย่างตามแนวคิดเอ็มดีเอ

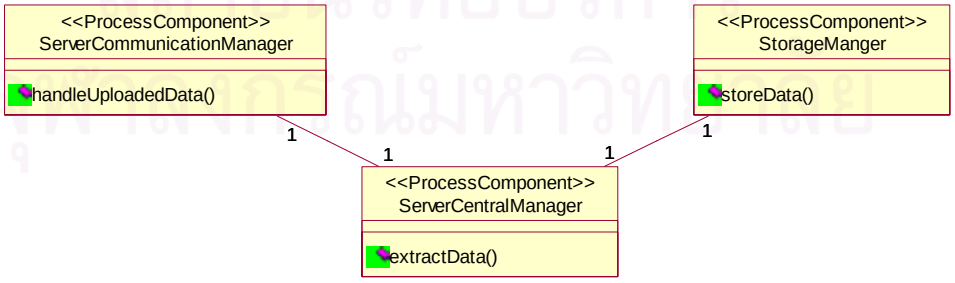
รูปที่ 6.1 เป็นการแสดงความสัมพันธ์ของแผนภาพต่างๆ ในบทที่ 4 โดยตัวเลขในกรอบสี่เหลี่ยมแสดงหมายเลขรูปของแผนภาพ ส่วนเส้นเชื่อมต่างๆ นั้นแสดงความสัมพันธ์ระหว่างแผนภาพ โดยมีการระบุชื่อของชนิดความสัมพันธ์ไว้ ส่วนรูปสี่เหลี่ยมที่มีกรอบเป็นเส้นประนั้นแสดงถึงแผนภาพที่ได้ละไว้ในบทที่ 4 เพื่อความกระชับของการนำเสนอ แต่ควรจะมีหากต้องการให้แบบจำลองของระบบมีความสมบูรณ์มากขึ้น (ในที่นี้ได้นำเสนอเพิ่มเติมไว้ในรูปที่ 6.2-6.5)



รูปที่ 6.1 ความสัมพันธ์ของแผนภาพต่างๆ ในแบบจำลองของระบบตัวอย่าง



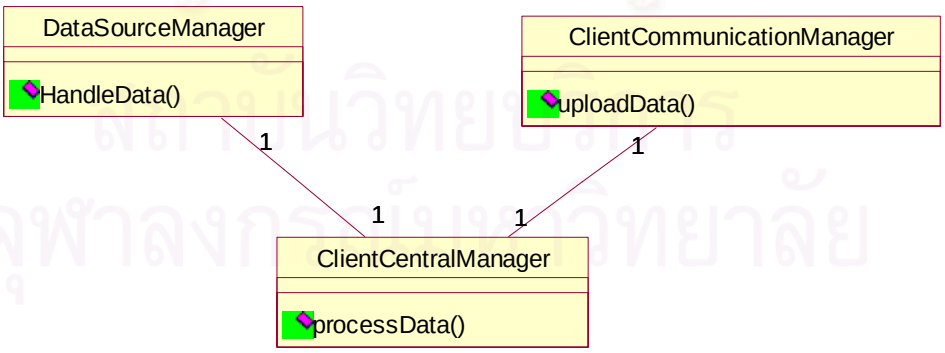
รูปที่ 6.2 แผนภาพแสดงส่วนประกอบของตัวรวบรวมข้อมูลระยะไกลแสดงด้วยยูเอ็มแอลโปรไฟล์



รูปที่ 6.3 แผนภาพแสดงส่วนประกอบของตัวเก็บข้อมูลส่วนกลางแสดงด้วยยูเอ็มแอลโปรไฟล์

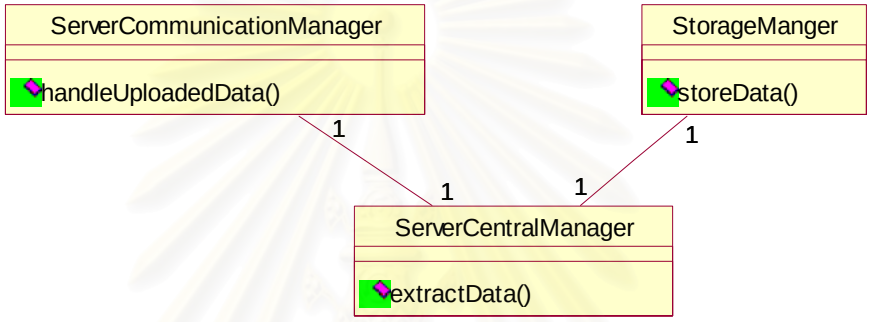
รูปที่ 4.2 เป็นแผนภาพซึ่งแสดงการมองระบบจากมุมมองของซีซีเอโมเดล โดยมองทั้งระบบเป็นโปรเซสคอมมิวนิตี (ProcessCommunity) ซึ่งประกอบด้วย 2 โปรเซสคอมโพเนนท์ย่อย โดยได้อธิบายส่วนประกอบภายในแต่ละโปรเซสคอมโพเนนท์ในรูปที่ 4.3 (ฝั่งไคลเอนต์) และรูปที่ 4.4 (ฝั่งเซิร์ฟเวอร์) ซึ่งแผนภาพดังกล่าวสามารถแสดงด้วยโนเทชันของภาษายูเอ็มแอลได้ดังแสดงในรูปที่ 6.2 (ฝั่งไคลเอนต์) และรูปที่ 6.3 (ฝั่งเซิร์ฟเวอร์) ซึ่งในที่นี้ใช้ชื่อความสัมพันธ์ของทั้งสองแผนภาพว่า Notation Mapping หมายความว่าทั้งสองแผนภาพซึ่งแสดงด้วยโนเทชันที่แตกต่างกันนั้น แท้จริงแล้วมีความสามารถในการสื่อความหมายที่เหมือนกันทุกประการนั่นเอง

หลังจากได้แบบจำลองของระบบในมุมมองของซีซีเอโมเดลแล้ว จะพบว่าภาษาซีซีเอมีความสามารถจำกัดในการอธิบายความหมายของแบบจำลองในระดับที่ลึกลงไป ดังนั้นจึงต้องมีการเปลี่ยนมุมมองที่ใช้มองระบบ (ซึ่งหมายถึงเปลี่ยนภาษาโมเดล และเมตาโมเดลที่ใช้อธิบายระบบด้วย) โดยในที่นี้เลือกใช้ยูเอ็มแอลโมเดลในการอธิบายระบบในระดับถัดไป และเนื่องจากมีการเปลี่ยนชุดของเมตาโมเดลที่ใช้ในการอธิบายระบบ จึงต้องมีขั้นตอนของการแมปปิงระหว่างคอนเซปต์ระหว่างเมตาโมเดลที่แตกต่างกัน กล่าวคือต้องมีการกำหนดว่าคอนเซปต์หนึ่งในเมตาโมเดลแรกนั้น มีความสอดคล้องกับคอนเซปต์ใดในเมตาโมเดลที่สองนั่นเอง ซึ่งขั้นตอนดังกล่าวนั้นในที่นี้ใช้ชื่อความสัมพันธ์ว่า Metamodel Mapping และเนื่องจากในยูเอ็มแอลโมเดลนั้นแบ่งมุมมองของระบบออกเป็น 2 มุมมองได้แก่ มุมมองเชิงโครงสร้าง (Structural View) และมุมมองเชิงพฤติกรรม (Behavioral View) และเนื่องจากข้อมูลของมุมมองเชิงพฤติกรรมนั้นไม่เคยปรากฏมาก่อนในระบบของแบบจำลองที่ใช้ซีซีเอโมเดล ดังนั้นจึงจำเป็นต้องมีการกำหนดรายละเอียดของแบบจำลองเชิงพฤติกรรมดังกล่าวเพิ่มเติมขึ้น ซึ่งในที่นี้ใช้ชื่อความสัมพันธ์ว่า ViewPoint

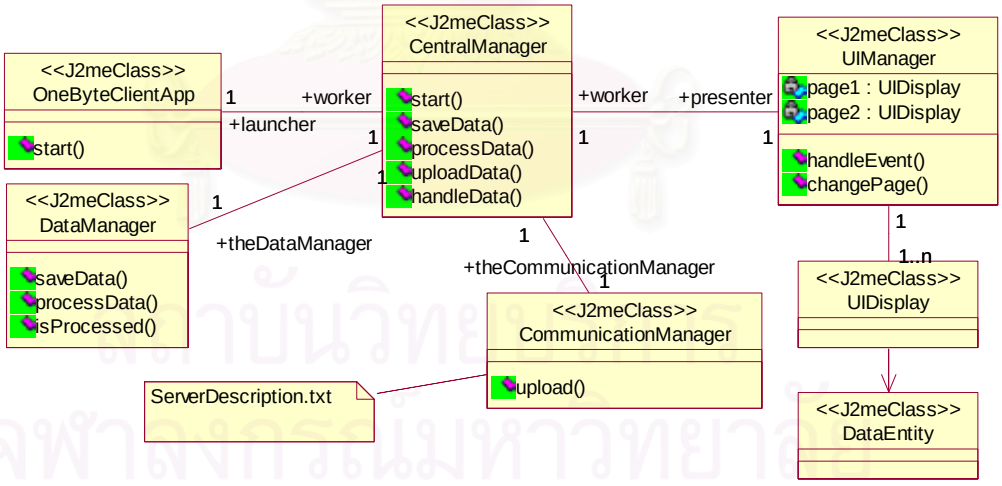


รูปที่ 6.4 แผนภาพแสดงส่วนประกอบของตัวรวบรวมข้อมูลระยะไกลแสดงด้วยภาษายูเอ็มแอล

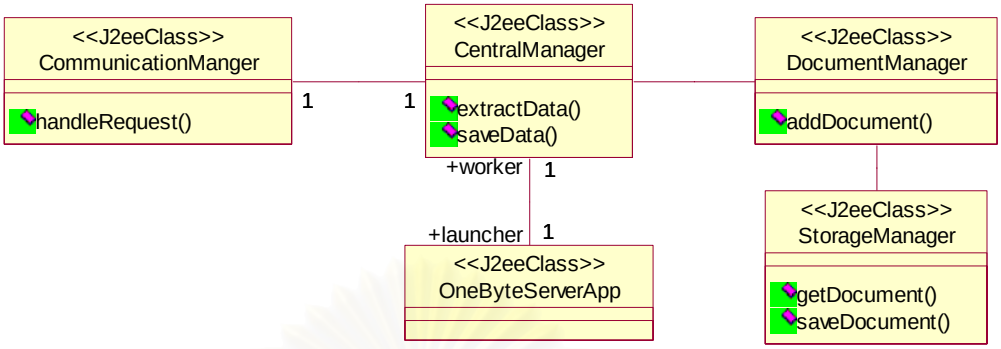
หลังจากมีการเพิ่มรายละเอียด (Refinement) แบบจำลองเชิงโครงสร้างในรูปที่ 6.4 แล้ว จะได้เป็นแบบจำลองเชิงโครงสร้างที่สมบูรณ์ขึ้นในรูปที่ 4.8 ซึ่งมีความเชื่อมโยงกับแบบจำลองเชิงพฤติกรรมที่ได้กำหนดขึ้น โดยได้แสดงความเชื่อมโยงดังกล่าวด้วยแผนภาพในรูปที่ 4.10 จากนั้นมีการเพิ่มรายละเอียดของแบบจำลองขึ้นอีก แต่วิธีการเพิ่มรายละเอียดดังกล่าวจะแตกต่างจาก Refinement ในข้างต้นเนื่องจากเป็นการอธิบายแผนภาพหนึ่งด้วยคอนเซปต์ของอีกแผนภาพหนึ่ง (อธิบายโอเปอร์เรชันในแผนภาพคลาสด้วยแผนภาพซีเควนซ์) ซึ่งในที่นี้ใช้ชื่อความสัมพันธ์ว่า Enhancement



รูปที่ 6.5 แผนภาพแสดงส่วนประกอบของตัวเก็บข้อมูลส่วนกลางแสดงด้วยภาษายูเอ็มแอล



รูปที่ 6.6 แผนภาพแสดงส่วนประกอบของตัวรวบรวมข้อมูลระยะไกลแสดงด้วยยูเอ็มแอลโปรไฟล์สำหรับเจทูเอ็มอี



รูปที่ 6.7 แผนภาพแสดงส่วนประกอบของตัวเก็บข้อมูลส่วนกลางแสดงด้วยยูเอ็มแอลโปรไฟล์สำหรับเจทูอีอี

ขั้นตอนของการแมปปิงจากแบบจำลองพีไอเอ็มไปเป็นแบบจำลองพีเอสเอ็มนั้น เป็นการแมปปิงระหว่างคอนเซปต์ในยูเอ็มแอลเมตาโมเดล ไปยังคอนเซปต์ในยูเอ็มแอลโปรไฟล์สำหรับเจทูเอ็มอีเมตาโมเดล (ทางฝั่งไคลเอนต์) และไปยังคอนเซปต์ในยูเอ็มแอลโปรไฟล์สำหรับเจทูอีอีเมตาโมเดล (ทางฝั่งเซิร์ฟเวอร์) ทำให้ได้แผนภาพใหม่ในรูปที่ 6.6 และรูปที่ 6.7 ตามลำดับ ซึ่งเมื่อมีการเพิ่มรายละเอียดเข้าไปจะกลายเป็นรูปที่ 4.25 และ 4.30 ตามลำดับ ส่วนแผนภาพเชิงพฤติกรรมทั้งหลายก็มีการเพิ่มรายละเอียดเช่นกันดังแสดงในรูปที่ 4.20 – 4.24 แต่เนื่องจากคอนเซปต์ในแผนภาพเชิงพฤติกรรมทั้งสองระดับเป็นคอนเซปต์เดียวกัน ดังนั้นจึงไม่จำเป็นต้องมีขั้นตอนของการแมปปิงเมตาโมเดล

จากตัวอย่างการพัฒนาระบบตัวอย่างด้วยแนวคิดเอ็มดีเอที่ได้กล่าวข้างต้น ทำให้สามารถเข้าใจภาพรวมของกระบวนการพัฒนาระบบด้วยแนวคิดเอ็มดีมากขึ้น อย่างไรก็ตามการนำแนวคิดดังกล่าวไปใช้จริงนั้นยังคงมีประเด็นอีกมากที่ยังขาดความกระจ่าง และงานวิจัยนี้ไม่ได้ครอบคลุมถึง ยกตัวอย่างเช่น การสร้างเครื่องมือช่วยในกระบวนพัฒนาระบบให้เป็นไปอย่างอัตโนมัติ การกำหนดหลักเกณฑ์มาตรฐานในการทำแมปปิงทั้งในระดับโมเดลและเมตาโมเดล การกำหนดหลักเกณฑ์การตรวจสอบความสมบูรณ์ของแบบจำลองพีเอสเอ็มจนถึงระดับที่สามารถนำมาสร้างโค้ดได้อย่างสมบูรณ์ วิธีการตรวจสอบขอบเขตของผลลัพธ์ในระดับล่างเนื่องจากการเปลี่ยนแปลงในระดับบน (Scope of Change Propagation) รวมถึงการทำให้หลักการของวิศวกรรมไปกลับ (Roundtrip Engineering) สามารถเกิดขึ้นได้อย่างสมบูรณ์ และเป็นไปอย่างอัตโนมัติมากที่สุด ซึ่งจะทำให้แบบจำลองของระบบมีความยั่งยืน และมีคุณค่าสูงสุด

6.1.2 การพัฒนาระบบตามแนวคิดเอ็มดีเอ

จากการทดลองพัฒนาระบบตัวอย่างพบว่า การพัฒนาระบบตามแนวคิดเอ็มดีเอคือกระบวนการพัฒนาที่อาศัยเทคโนโลยีของการจัดการแบบจำลอง โดยการสร้างระบบใดๆ จำเป็นต้องมีการสร้างแบบจำลองของระบบนั้นก่อนเสมอ โดยสามารถแบ่งขั้นตอนการสร้างแบบจำลองเป็น 2 ขั้นตอนได้แก่ แบบจำลองพีไอเอ็มซึ่งเป็นแบบจำลองที่อธิบายเฉพาะหน้าที่และพฤติกรรมของระบบเท่านั้น โดยไม่มีรายละเอียดของการอิมพลีเมนต์อยู่เลย และอีกขั้นตอนคือการสร้างแบบจำลองพีเอสเอ็มซึ่งเป็นแบบจำลองที่มีรายละเอียดเชิงเทคโนโลยี และรายละเอียดเชิงวิศวกรรมของการอิมพลีเมนต์ระบบนั้นๆ อย่างครบถ้วน

เอ็มดีเอแตกต่างจากกระบวนการพัฒนาระบบแบบดั้งเดิมเช่น วอร์เตอร์ฟอลโมเดล (Waterfall Model) หรือสไปรอลโมเดล (Spiral Model) เนื่องจากเอ็มดีเอเป็นมากกว่ากระบวนการพัฒนาระบบกล่าวคือ นอกจากเอ็มดีเอจะกำหนดลำดับขั้นของการพัฒนาแล้ว ยังได้กำหนดเทคโนโลยี (ได้แก่แมปปิง) ที่ช่วยให้การพัฒนาระบบเป็นไปอย่างอัตโนมัติมากที่สุด นอกจากนั้นยังคำนึงถึงการเชื่อมโยงการทำงานร่วมกันระหว่างระบบ และคำนึงถึงการปรับเปลี่ยนระบบเนื่องจากการเปลี่ยนแปลงความต้องการ และการเปลี่ยนแปลงของเทคโนโลยีอีกด้วย นอกจากนี้เอ็มดีเอยังแตกต่างจากเทคโนโลยีมิดเดิลแวร์ทั้งหลาย เนื่องจากเอ็มดีอยู่ในระดับที่สูงกว่าเทคโนโลยีมิดเดิลแวร์ กล่าวคือเอ็มดีเอไม่ได้กำหนดแพลตฟอร์มขึ้นมาใหม่แต่กำหนดแนวทางการเชื่อมโยงการทำงานระหว่างแพลตฟอร์มโดยกำหนดการเชื่อมโยงในระดับแบบจำลองและอาศัยเทคโนโลยีการแปลงทำให้เกิดผลการเชื่อมโยงดังกล่าวในระดับอิมพลีเมนต์

ประเด็นสำคัญในการพัฒนาแบบจำลองของระบบ โดยเฉพาะอย่างยิ่งในระดับของแบบจำลองพีไอเอ็มนั้นคือ ทำอย่างไรให้แบบจำลองของระบบสามารถแสดงรายละเอียดทั้งในเชิงโครงสร้าง และเชิงพฤติกรรมอย่างถูกต้องและครบถ้วน ในขณะเดียวกันนั้นแบบจำลองดังกล่าวยังคงมีความยืดหยุ่นมากเพียงพอในการเลือกใช้เทคโนโลยีใดๆ ก็ได้ในการอิมพลีเมนต์ ซึ่งจากการทดลองในงานวิจัยนี้ พบว่าการพัฒนาแบบจำลองพีไอเอ็มนั้นเป็นงานที่มีความซับซ้อนมาก ส่วนหนึ่งเกิดจากข้อดีของเอ็มดีเอที่ยอมให้ใช้ชุดของแผนภาพหลายแผนภาพ และมีระดับนามธรรมได้หลายระดับ จึงเกิดข้อเสียที่ตามมาได้แก่ การรักษาความสอดคล้องระหว่างแผนภาพที่ใช้ร่วมกันในการอธิบายระบบ นอกจากนั้นยังพบว่าแบบจำลองพีไอเอ็มนั้นยังอาจสามารถครอบคลุมได้ในหลายมุมมองทั้งในแง่แบบจำลองเชิงธุรกิจ (Business Model) และแง่แบบจำลองเชิงระบบคอมพิวเตอร์ (Computer System Model) ยกตัวอย่างเช่น

1. แบบจำลองไม่ขึ้นกับการคำนวณ (CIM: Computation Independent Model) ได้แก่ การกำหนดความสัมพันธ์ระหว่างคอนเซปต์ต่างๆ ในระบบ (ในลักษณะของการสร้างออนโทโลจีของคอนเซปต์ต่างๆ ในโดเมนระบบนั้น) โดยไม่สนใจตรรกะของการนำข้อมูลไปประมวลผล เช่น การกำหนดข้อตกลงร่วมของคำศัพท์และคำจำกัดความของคำศัพท์ภายในองค์กร หรือดาตาดิกชันนารีขององค์กร เพื่อให้ทุกคนในองค์กรสื่อความหมายตรงกัน
2. แบบจำลองไม่ขึ้นกับเทคโนโลยี (TIM: Technology Independent Model) ซึ่งในที่นี้คำว่าเทคโนโลยีจะมีความหมายแตกต่างจากคำว่าแพลตฟอร์มตามคำจำกัดความของเอ็มดีเอ กล่าวคือแพลตฟอร์มของเอ็มดีเอนั้นมองที่การเลือกรูปแบบการอิมพลีเมนต์ซึ่งมีอยู่แล้วนำมาประยุกต์ใช้ให้เหมาะสมกับธุรกิจ ส่วนเทคโนโลยีนั้นเป็นการมองถึงการเปลี่ยนแปลงวิธีการอิมพลีเมนต์เนื่องจากมีนวตกรรมใหม่เกิดขึ้น ในขณะที่หลักการทางธุรกิจยังเหมือนเดิม ยกตัวอย่างเช่นการซื้อลอตเตอรีกับตัวแทนขาย ซึ่งส่งข้อมูลการซื้อไปยังส่วนกลางผ่านจุดทำการซึ่งตั้งอยู่ในที่ทำการไปรษณีย์ และการซื้อลอตเตอรีกับตัวแทนขายซึ่งส่งข้อมูลไปยังส่วนกลาง ผ่านการเชื่อมต่ออินเทอร์เน็ตโดยใช้โปรแกรมเฉพาะบนเครื่องปาล์มนั้น น่าจะมีแบบจำลองเชิงธุรกิจไม่ขึ้นกับเทคโนโลยีที่เหมือนกัน ทั้งนี้เนื่องจากระบบทั้งสองมีลักษณะความต้องการทางธุรกิจที่เหมือนกัน แต่ต่างกันเพียงในกรณีหลังมีการประยุกต์ใช้งานเทคโนโลยีของคอมพิวเตอร์มือถือเป็นต้น
3. แบบจำลองไม่ขึ้นกับวงการธุรกิจ (BIM: Business Domain Independent Model) ยกตัวอย่างเช่น ในหัวข้อที่ 4.3.2 ได้มีการพูดถึง ระบบเครื่องตรวจวัดอุณหภูมิ และระบบการจัดทำผลสำรวจประชามติ ซึ่งมีแบบจำลองที่เหมือนกับระบบตัวอย่าง ดังนั้นแบบจำลองดังกล่าวจึงเป็นแบบจำลองที่ไม่ขึ้นกับวงการธุรกิจเนื่องจากระบบทั้งสามนั้นอยู่ในวงการธุรกิจที่แตกต่างกัน แต่มีรูปแบบของความต้องการทางธุรกิจ (Business Pattern) ที่คล้ายกัน ดังนั้นจึงมีแบบจำลองของระบบที่เหมือนกัน

ความซับซ้อนอีกประการได้แก่การที่แต่ละมุมมองนั้นเป็นการพิจารณาระบบด้วยมุมมองที่เป็นอิสระจากกัน ซึ่งแต่ละมุมมองต่างก็มีปรัชญาในการพิจารณาระบบที่แตกต่างกัน และต่างก็ได้ชื่อว่าเป็นแบบจำลองที่ไม่ขึ้นกับแพลตฟอร์มด้วยกันทั้งสิ้น ดังนั้นการจัดระเบียบและการรวบรวมแบบจำลองจากมุมมองต่างๆ ให้กลายเป็นแบบจำลองสมบูรณ์แบบของระบบจึงเป็นปัญหาสำคัญที่ต้องพิจารณา นอกจากนี้การแบ่งแบบจำลองพีไอเอ็มเป็นมุมมองต่างๆ นั้นทำให้เข้าใจว่าการใช้

งานแบบจำลองในรูปแบบมาตรฐาน (Standard Model Pattern) นั้นสามารถเกิดขึ้นได้ในหลายทาง ยกตัวอย่างเช่น ระบบงานของบริษัทเงินทุนหลักทรัพย์จะมีแบบจำลองไม่ขึ้นกับการคำนวณส่วนหนึ่งเหมือนระบบงานของธนาคาร เนื่องจากเป็นธุรกิจที่เกี่ยวข้องกับระบบการเงินเหมือนกัน ตัวอย่างอื่นๆ ได้แก่ ระบบการประมูลออนไลน์นั้นน่าจะสามารถยืมแบบจำลองของระบบการประมูลสินค้าปกติมาใช้ได้ เนื่องจากระบบทั้งสองมีความต้องการพื้นฐานทางธุรกิจที่เหมือนกัน เพียงแต่ใช้เทคโนโลยีที่แตกต่างกันในการอิมพลีเมนต์ระบบ ดังนั้นในอนาคตเมื่อเทคโนโลยีเอ็มดีเอมีความพร้อมมากเพียงพอนั้น การสร้างแบบจำลองพีไอเอ็มจะเป็นการนำกลับมาใช้ใหม่ของแบบจำลองของมาตรฐานของระบบอื่นๆ ที่มีลักษณะความต้องการทางธุรกิจที่คล้ายกัน

ปัจจัยกำหนดความสำเร็จของเอ็มดีเอคือ ความสามารถของเครื่องมือช่วยพัฒนาระบบและมาตรฐานที่เกี่ยวข้องกับการพัฒนาระบบเช่น ยูเอ็มแอลโปรไฟล์สำหรับการออกแบบงานเฉพาะอย่าง และมาตรฐานกฎการแปลงระหว่างโปรไฟล์นั้นๆ ไปยังแพลตฟอร์มเป้าหมาย เนื่องจากเป้าหมายของเอ็มดีเอคือ การทำให้กระบวนงานการพัฒนาระบบส่วนใหญ่เป็นงานที่สามารถทำได้โดยเครื่องมืออย่างอัตโนมัติ

6.2 ข้อเสนอแนะ

1. ในการสร้างแบบจำลองของระบบนั้นควรพยายามใช้โปรไฟล์มาตรฐานที่มีอยู่ให้มากที่สุดเท่าที่เป็นไปได้เนื่องจากมีเมตาโมเดลรองรับอยู่แล้ว และแบบจำลองที่ได้จะได้สามารถใช้ประโยชน์จากกฎการแปลงต่างๆ ในอนาคต
2. การเลือกภาษาที่ใช้แสดงแบบจำลองเป็นสิ่งที่สำคัญมากในการสร้างแบบจำลอง เนื่องจากภาษาโมเดลเป็นตัวกำหนดกรอบความคิด และคอนเซปต์พื้นฐานต่างๆ ดังนั้นภาษาโมเดลที่ดีที่สุด คือภาษาที่สามารถแสดงข้อเท็จจริงของระบบได้มากที่สุดและเข้าใจง่ายที่สุด
3. ในการสร้างเครื่องมือพัฒนาระบบนั้นควรเริ่มจากการกำหนดเมตาโมเดลที่เครื่องมือรองรับ ซึ่งหมายถึงขอบเขตของคอนเซปต์ในแบบจำลองที่เครื่องมือนั้นสามารถเข้าใจได้ นอกจากนั้นยังมีเรื่องของการสร้างเครื่องมือบรรณาธิกรณ์แบบจำลองเชิงกราฟฟิก (Graphical Model Editor) ซึ่งท้ายที่สุดควรสามารถบันทึกแบบจำลองที่ได้ในรูปของมาตรฐานเอ็กซ์เอ็มไอได้ เนื่องจากเครื่องมือต้องใช้ตัวแจงเอ็กซ์เอ็มไอ (XMI Parser) ในการตีความหมายของแบบจำลอง ขั้นตอนต่อไปคือการตรวจสอบความถูกต้องและการแจ้งข้อผิดพลาดเชิงวากยสัมพันธ์ (Syntax Error) ของแบบจำลอง หมายถึงเครื่องมือต้องสามารถเข้าใจสิ่งที่แบบจำลองนั้นต้องการจะสื่อได้อย่างถูกต้อง ส่วน

การทำแมปปิงนั้นต้องมีสมมติฐานว่าแบบจำลองที่จะทำแมปปิงได้นั้นต้องเป็นแบบจำลองที่มีความถูกต้อง (ไม่มีส่วนใดซึ่งเครื่องมือไม่สามารถตีความได้จากแบบจำลอง) ส่วนการกำหนดกฎการแปลงนั้นน่าจะกำหนดในลักษณะของชุดของกฎย่อยๆ จำนวนมาก โดยมีเครื่องมือการตรวจสอบความขัดแย้งกันเองระหว่างกฎ ซึ่งจะทำให้ผู้สร้างกฎสามารถทำงานได้ง่ายขึ้น

รายการอ้างอิง

1. OMG. Developing in OMG's Model-Driven Architecture. (Online). Available from: ftp://ftp.omg.org/pub/docs/omg/01-12-01.pdf: OMG, [November 2001].
2. Ober, I. Difficulties in Defining Precise Semantics for UML. Proceedings of 14th European Conference on Object-Oriented Programming. France, June, 2000.
3. OMG. MDA - A Technical Perspective. OMG's white paper (July 9, 2001):
4. OMG. OMG Unified Modeling Language Specification Version 1.5. March 2003.
5. Booch, G., Rumbaugh, J., and Jacobson, I. The Unified Modeling Language User Guide. 1999.
6. Mowbray, T.J. and Ruh, W.A. Inside CORBA. 1997.
7. วีรศักดิ์ ซึงถาวร. Enterprise JavaBeans. Bangkok: Se-education, 2003.
8. OMG. UML Profile for CORBA Specification Version 1.0. April 2002.
9. OMG. Meta Object Facility Specification Version 1.4. April, 2002.
10. OMG. UML Profile for Enterprise Distributed Object Computing Specification. February 2002.
11. OMG. UML Profile and Interchange Modelsfor Enterprise Application Integration (EAI) Specification. February 2002.
12. Microsystems, S. J2ME Documentation. (Online). Available from: http://java.sun.com/j2me/docs/index.html: Sun Microsystems, 2004.
13. Foundation, A.S. The Apache Tomcat Project. (Online). Available from: http://jakarta.apache.org/tomcat/: 2004.
14. Gerber, A., et al. Transformation: The Missing Link of MDA. Proceedings of International Conference on Graph Transformation (ICGT). Spain, 2002.
15. Ambler, S.W. Agile model driven development is good enough. Software, IEEE 20 (2003): 71-73.
16. Wang, H. and Zhang, D. MDA-based development of e-learning system. Proceedings of Computer Software and Applications Conference, 2003. COMPSAC 2003. Proceedings. 27th Annual International. 2003.

17. Thongmak, M. and Muenchaisri, P. Design of Rules for Transforming UML Sequence Diagrams into Java Code. Proceedings of Ninth Asia-Pacific Software Engineering Conference (APSEC'02). Gold Coast, Australia, December, 2002.
18. Oldevik, J., et al. Framework for model transformation and code generation. Proceedings of Enterprise Distributed Object Computing Conference, 2002. EDOC '02. Proceedings. Sixth International. 2002.
19. Peltier, M., Bezivin, J., and Guillaume, G. MTRANS: A general framework, based on XSLT for model transformations. Proceedings of Workshop on Transformation in UML (WTUML'01). Italy, April, 2001.
20. Milicev, D. Automatic Model Transformations Using Extended UML Object Diagrams in Modeling Environments. IEEE Transactions on Software Engineering, p. 413-431. April, 2002.
21. Harrison, W., Barton, C., and Raghavachari, M. Mapping UML Designs to Java. Proceedings of Object-Oriented Programming Systems, Lanuages, and Applications. 2000.
22. Regep, D. and Kordon, F. Using MetaScribe to prototype an UML to C++/Ada95 code generator. Proceedings of 11th IEEE International Workshop on Rapid System Prototyping. Paris, France, June, 2000.
23. Park, D.H. and Kim, S.D. XML Rule Based Source Code Generator for UML CASE Tool. Proceedings of 8th Asia-Pacific Software Engineering Conference (APSEC 2001). Macau, China, December, 2001.
24. Microsystems, S. MIDP for PalmOS 1.0. (Online). Available from: http://developers.sun.com/techtopics/mobility/midp/articles/palm/: Sun Microsystems, 2003.
25. OMG. UML Profile for Schedulability, Performance, and Time Specification. March 2002.
26. OMG. Common Warehouse Metamodel Specification. October, 2001.
27. OMG. Object Constraint Language Specification. September, 1997.
28. Gamma, E., et al. Design Patterns. 1995.

ประวัติผู้เขียนวิทยานิพนธ์

นายรังสรรค์ เกียรติ์ภานนท์ เกิดเมื่อวันที่ 12 กันยายน พ.ศ. 2522 ที่จังหวัดกรุงเทพมหานคร สำเร็จการศึกษาระดับปริญญาตรีวิศวกรรมศาสตรบัณฑิต คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย สาขาวิชาวิศวกรรมไฟฟ้าอิเล็กทรอนิกส์ ในปีการศึกษา 2543 และเข้าศึกษาต่อในหลักสูตรวิทยาศาสตรมหาบัณฑิต ที่ภาควิชาวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัยในปีการศึกษา 2544