การแปลงโปรแกรมที่พัฒนาขึ้นโดยภาษาจาวาให้เข้าสู่แผนภาพยูเอ็มแอลโดยใช้มาตรฐานเอ็กซ์เอ็มไอ

An Approach for Transforming Java Code to UML Diagrams with XMI Standard

โดย

บุญประเสริฐ สุรักษ์รัตนสกุล

Boonprasert Surakratanasakul

โครงการวิจัยนี้เป็นโครงงานวิจัยโดยใช้เงินรายได้

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ประจำปีงบประมาณ 2547

การแปลงโปรแกรมที่พัฒนาขึ้นโดยภาษาจาวาให้เข้าสู่แผนภาพยูเอ็มแอลโดยใช้มาตรฐานเอ็กซ์เอ็มไอ

An Approach for Transforming Java Code to UML Diagrams with XMI Standard

โดย

บุญประเสริฐ สุรักษ์รัตนสกุล

Boonprasert Surakratanasakul

โครงการวิจัยนี้เป็นโครงงานวิจัยโดยใช้เงินรายได้

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ประจำปีงบประมาณ 2547

บทคัดย่อ

งานวิจัยนี้ เสนอวิธีการการแปลงโปรแกรมที่พัฒนาขึ้นโดยภาษาจาวาให้เข้าสู่แผนภาพยูเอ็มแอล ตามหลักการเชิงวัตถุโดยใช้มาตรฐานเอ็กซ์เอ็มไอ ซึ่งเป็นมาตรฐานสำหรับการแลกเปลี่ยนข้อมูลเป็นสื่อกลางในกระบวนการแปลงซึ่งกำหนดโดยองค์กรโอเอ็มจี ซึ่งเป็นหน่วยงานกำหนดมาตรฐานของภาษายูเอ็มแอลและมาตรฐานเอ็กซ์เอ็มไอ การแปลงนี้จะเป็นไปในลักษณะของกระบวนการวิศวกรรมแบบย้อนกลับ โดยงานวิจัยจัดทำขึ้นโดยกำหนดใช้มาตรฐานภาษาโปรแกรมจาวา รุ่นที่ 1.2 และภาษายูเอ็มแอล รุ่นที่ 1.4 เป็นมาตรฐาน สำหรับงานวิจัยนี้

กิตติกรรมประกาศ

งานวิจัยนี้ สำเร็จด้วยความเรียบร้อยสมบูรณ์ ด้วยงบวิจัยที่ได้รับการสนับสนุนจากงบเงินรายได้ของคณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ประจำปีงบประมาณ 2547 ซึ่งได้เล็งเห็นถึงความสำคัญของการพัฒนาและสร้างสรรค์งานวิจัยของบุคลากรภายในคณะฯ และที่จะกล่าวถึงมิได้ คือ ส่วนงานนโยบายและแผนงานของคณะเทคโนโลยีสารสนเทศ คุณบุญช่วย ชาติทอง และ คุณอภิญญา ปิ่นเงิน ที่ช่วยเหลือประสานงานให้งานวิจัยนี้สำเร็จได้ด้วยดี ซึ่งงานวิจัยนี้จะไม่เกิดขึ้นถ้าขาดท่านเหล่านี้ และยังมีอีกมากมายที่ไม่สามารถกล่าวถึงหมดได้ ณ ที่นี้ ขอขอบพระคุณครับ

บุญประเสริฐ สุรักษ์รัตนสกุล

สารบาญ

สารบาญตาราง

สารบาญภาพประกอบ

ภาพที่ หน้า

บทที่ 1

บทนำ

1.1 ความสำคัญและ ที่มาของปัญหา

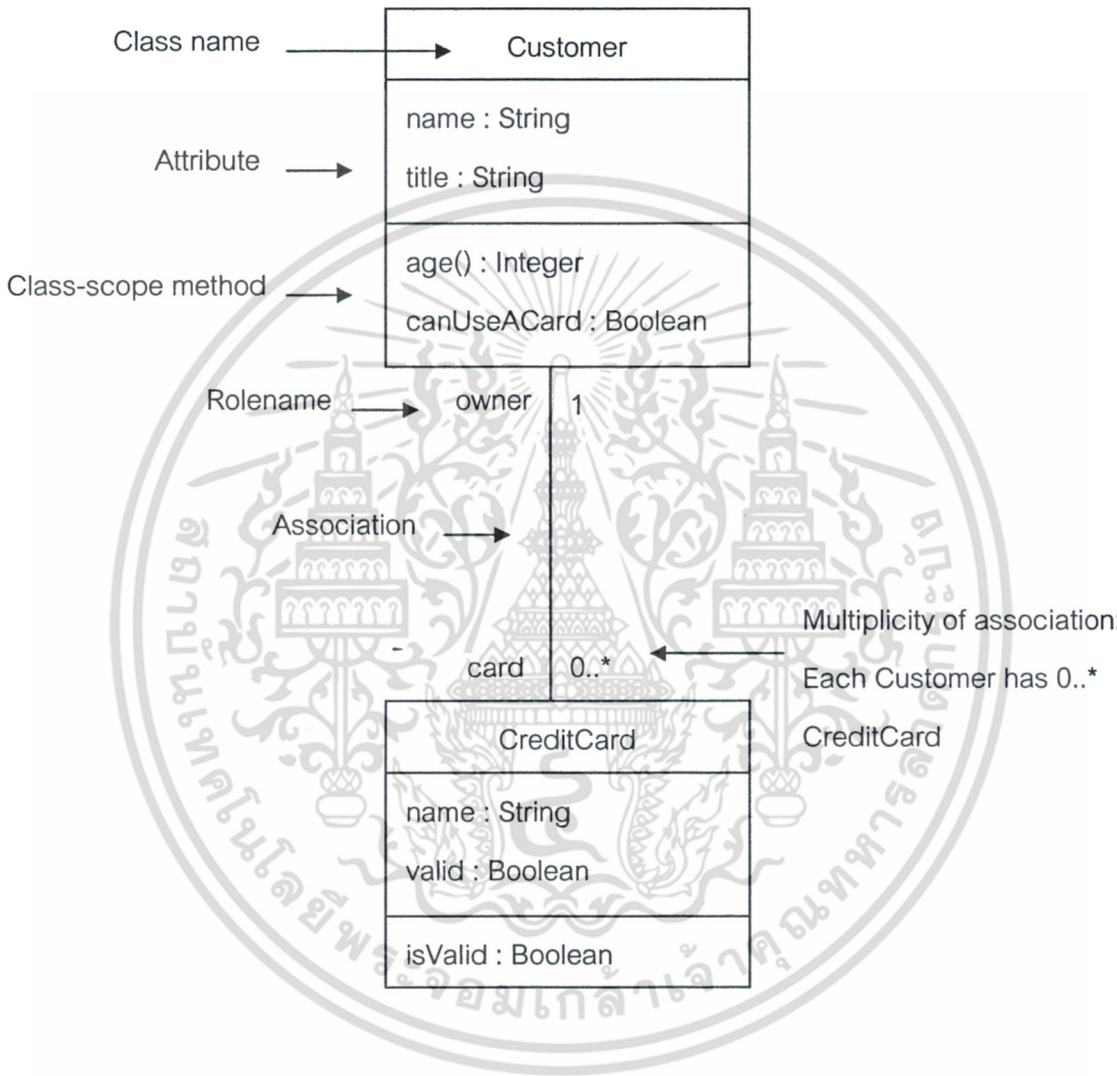
พื้นฐานของภาษาทุกสรรพภาษา ต้องประกอบด้วยสิ่งสองสิ่งเป็นพื้นฐาน คือ วากยสัมพันธ์ (Syntax) และความหมาย (Semantic) หากภาษาใดขาดสิ่งใดสิ่งหนึ่งหรือทั้งสองสิ่ง ก็ไม่สามารถนับได้ว่าสิ่งนั้นคือภาษา ภาษายูเอ็มแอล (UML – Unified Modeling Language) ในกระบวนการออกแบบและพัฒนาระบบซอฟต์แวร์ก็เช่นเดียวกัน ผู้ที่ใช้งานแผนภาพยูเอ็มแอล (UML Diagrams) จำเป็นต้องศึกษาและทำความเข้าใจวากยสัมพันธ์และความหมายของแผนภาพยูเอ็มแอล เพื่อให้สามารถใช้งานแผนภาพยูเอ็มแอลได้อย่างถูกต้องทั้งรูปแบบและความหมาย โดยเฉพาะอย่างยิ่งการแปลงข้อมูลของภาษายูเอ็มแอล

ก. วากยสัมพันธ์ของแผนภาพยูเอ็มแอล (Abstract Syntax of UML)

วากยสัมพันธ์ในแผนภาพยูเอ็มแอล เป็นการกล่าวถึงโครงสร้างหรือสัญลักษณ์ต่างๆที่ใช้แทนวัตถุที่ใช้ในกระบวนการพัฒนาซอฟต์แวร์โดยแผนภาพยูเอ็มแอล ภาพที่ 1.1 แสดงวากยสัมพันธ์ของแผนภาพคลาส ซึ่งเป็นแผนภาพหนึ่งในแผนภาพยูเอ็มแอล วากยสัมพันธ์กล่าวถึงว่าสัญลักษณ์ใดแทนสิ่งใดและมีโครงสร้างอย่างไร จากภาพที่ 1.1 คลาสถูกแทนด้วยสัญลักษณ์รูปสี่เหลี่ยม โดยสัญลักษณ์รูปสี่เหลี่ยมแบ่งส่วนภายในออกเป็นอย่างน้อย 3 ส่วน คือ ส่วนบนสุดเป็นส่วนของชื่อคลาส ส่วนกลางเป็นส่วนของแอตทริบิวส์ที่กำหนดภายในคลาส และส่วนล่างสุดเป็นส่วนของเมธอดของคลาส เมื่อผู้ใช้มีการกระทำกับแผนภาพคลาสไม่ว่าเป็นการสร้างหรือการแก้ไขสัญลักษณ์เหล่านี้เป็นสิ่งที่ผู้ใช้ต้องคำนึงถึงด้วย หรือถ้าผู้ใช้ไปเจอสัญลักษณ์นี้ที่ใด ก็สามารถกล่าวได้ว่าสัญลักษณ์ที่มีลักษณะรูปแบบนี้ คือ สัญลักษณ์ของคลาส

ภาพที่ 1.1

วากยสัมพันธ์ของแผนภาพยูเอ็มแอล



ข.ความหมายแบบอพลวัต (Static Semantics)

จากภาพที่ 1.1 นอกจากสัญลักษณ์ที่เป็นลักษณะเฉพาะดังสัญลักษณ์ที่แทนคลาสแล้ว สัญลักษณ์คลาสยังสามารถมีความสัมพันธ์ร่วมกันด้วย ซึ่งความสัมพันธ์ที่เกิดขึ้นก็ถูกแทนด้วยสัญลักษณ์หนึ่งๆเช่นกัน จากภาพที่ 1.1 ความสัมพันธ์ดังกล่าว ถูกแทนด้วยสัญลักษณ์เส้นตรง โดยเส้นตรงที่แทนความสัมพันธ์ อาจมีองค์ประกอบอื่นร่วมด้วย เช่นในภาพ ความสัมพันธ์ประกอบด้วยบทบาทระหว่างสองคลาสและจำนวนที่สัมพันธ์กัน เมื่อมีความสัมพันธ์กันในบทบาท

ดังกล่าว ตำแหน่งต่างๆขององค์ประกอบที่เป็นบทบาทและจำนวนความสัมพันธ์นับว่าเป็นสิ่งสำสำคัญ หรือกล่าวโดยรวมว่ารูปแบบที่ถูกต้องนั้น เป็นสิ่งที่ผู้ใช้งานแผนภาพยูเอ็มแอลต้องคำนึงถึงเมื่อมีการสร้างหรือแก้ไขแผนภาพ ซึ่งสิ่งที่ให้ความหมายของสิ่งเหล่านี้ คือ ความหมายแบบอพลวัต นั่นเอง

ค. ความหมายแบบพลวัต (Dynamic Semantics)

เมื่อสัญลักษณ์ต่างๆ แทนวากยสัมพันธ์ที่ถูกต้องและมีรูปแบบที่ถูกต้องในแผนภาพยูเอ็มแอลตามความหมายอพลวัตแล้ว สิ่งเหล่านี้ทำให้เกิดความหมายแบบพลวัตของแผนภาพยูเอ็มแอลขึ้นมา เมื่อผู้ใช้พร้อมที่จะนำแผนภาพนั้นไปใช้งาน ซึ่งการใช้งานที่ถูกต้องของแผนภาพจะต้องอยู่บนพื้นฐานของความหมายแบบพลวัตนี้

นอกจากภาษายูเอ็มแอลที่กล่าวมาข้างต้น ภาษาจาวา (Java) ก็เป็นภาษาโปรแกรมคอมพิวเตอร์หนึ่ง ที่นับได้ว่าเป็นภาษาที่ประกอบไปด้วยวากยสัมพันธ์และความหมายของภาษา และเนื่องจากทั้งสองภาษา ต่างก็มีวากยสัมพันธ์และความหมายและมีความเกี่ยวข้องกันในโดเมนของการพัฒนาซอฟท์แวร์ ทำให้เกิดแนวคิดที่จะแปลงทั้งสองภาษานี้เข้าด้วยกัน เพื่อให้การพัฒนาซอฟท์แวร์เป็นไปได้ง่ายขึ้น ซึ่งก่อนหน้านี้ การแปลงระหว่างทั้งสองภาษา มักเป็นไปแนวทางที่เป็นแบบจากภาษายูเอ็มแอลไปเป็นภาษาจาวา ตามหลักทั่วไปของการพัฒนาทางวิศวกรรมซอฟท์แวร์หรือที่เรียบกว่า กระบวนการพัฒนาแบบเดินหน้า (Forward engineering) แต่ว่า ณ ปัจจุบัน การพัฒนาอีกรูปแบบ คือ การนำเข้าภาษาโปรแกรมจาวาแล้วย้อนกลับไปสู่การเป็นแผนภาพยูเอ็มแอลก็เป็นวิธีหนึ่งที่มีความสำคัญ ซึ่งเรียกว่า วิศวกรรมแบบย้อนกลับ(Backward engineering) ซึ่งทำให้การพัฒนาซอฟท์แวร์มีความยืดหยุ่นและลดการใช้ทรัพยากร เนื่องจากมีการนำกลับมาใช้ใหม่ในส่วนของโปรแกรมที่ได้เขียนขึ้นมาแล้ว ทำให้การแปลงแผนภาพทั้งสองเป็นสิ่งที่มีความสำคัญและเป็นที่มาของงานวิจัยชิ้นนี้

1.2 <u>ขอบเขตของงานวิจัย</u>

จากปัญหาที่กล่าวมาข้างต้น เป็นที่มาของขอบเขตงานวิจัยนี้ โดยงานวิจัยนี้จะนำเสนอ

ก. แนวคิดทางสถาปัตยกรรมของการแปลงโปรแกรมที่พัฒนาขึ้นโดยแปลงจากภาษาจาวาให้เข้าสู่แผนภาพยูเอ็มแอลโดยใช้มาตรฐานเอ็กซ์เอ็มไอ

ข. แนวทางในการดำเนินการสร้างเครื่องมือเพื่อใช้ในการแปลงโปรแกรมที่พัฒนาขึ้นโดยแปลงจากภาษาจาวาให้เข้าสู่แผนภาพยูเอ็มแอลโดยใช้มาตรฐานเอ็กซ์เอ็มไอ

1.3 ประโยชน์ที่ได้รับ

ประโยชน์ที่คาดว่าจะได้รับจากงานวิจัย

ก. เครื่องมือที่ใช้ในการแปลงโปรแกรมที่พัฒนาขึ้นโดยภาษาจาวาให้เข้าสู่แผนภาพยูเอ็มแอลโดยใช้มาตรฐานเอ็กซ์เอ็มไอ

1.4 โครงร่างของงานวิจัย

เนื้อหาของงานวิจัยนี้ แบ่งเป็นส่วนต่างๆดังนี้ บทที่ 2 เป็นส่วนของการแลกเปลี่ยนข้อมูลซึ่งประกอบด้วยความเป็นมาของภาษายูเอ็มแอล วากยสัมพันธ์อธิบายโครงสร้างของแผนภาพในภาษายูเอ็มแอล และแนวทางในการออกแบบคุณสมบัติทางสถาปัตยกรรม บทที่ 3 จะกล่าวถึงการแลกเปลี่ยนข้อมูล ซึ่งประกอบด้วย การแปลงข้อมูลของภาษาจาวาเข้าสู่มาตรฐานเอ็กซ์เอ็มไอและการแปลงข้อมูลจากเอ็กซ์เอ็มไอให้เป็นแผนภาพยูเอ็มแอล บทสุดท้ายบทที่ 4 เป็นสรุปผลที่ได้จากการวิจัยและข้อเสนอแนะงานวิจัยที่จะดำเนินการต่อไปในอนาคต

บทที่ 2

ความรู้เบื้องต้นที่เกี่ยวข้องกับงานวิจัย

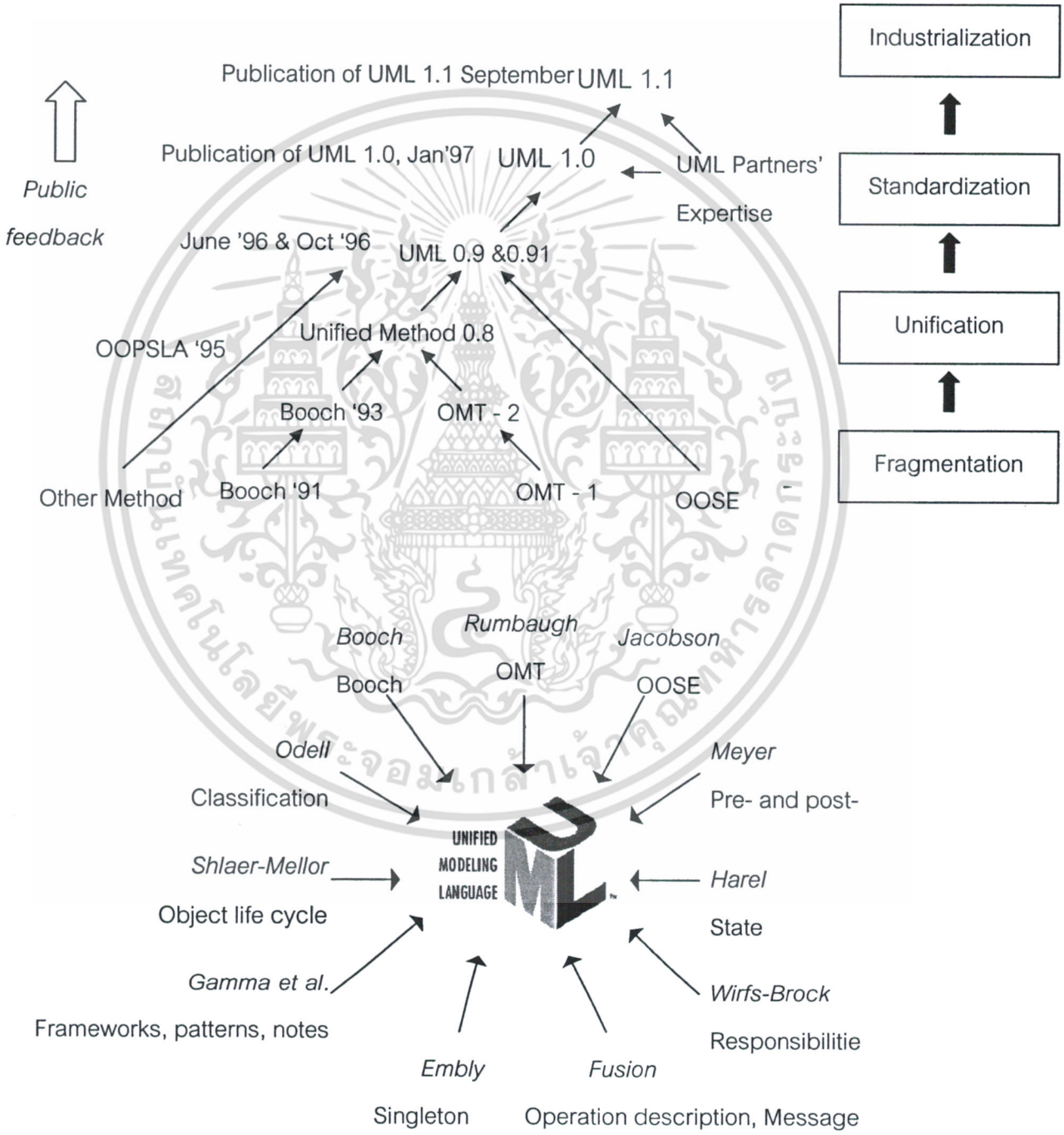
ในบทนี้ จะนำเสนอความเป็นมาของภาษายูเอ็มแอล เพื่อกล่าวถึงลักษณะโครงสร้างของภาษา วากยสัมพันธ์อธิบายโครงสร้างของแผนภาพในภาษายูเอ็มแอล และแนวทางในการออกแบบคุณสมบัติทางสถาปัตยกรรม ส่วนถัดไปจะเป็นการกล่าวถึงความรู้ในเรื่องของภาษายูเอ็มแอลเป็นสิ่งสำคัญในงานวิจัยชิ้นนี้ เนื่องจากการแปลงนั้นจะใช้หลักการการแปลงในระดับภาษาที่ใช้บรรยายภาษายูเอ็มแอล (Metadata Language) ซึ่งผู้ที่ทำการแปลงต้องมีความเข้าใจในเรื่องของโครงสร้างภาษายูเอ็มแอลเพื่อให้การแปลงข้อมูลนั้น มีความถูกต้องและเป็นไปตามหลักมาตรฐาน

2.1 ความเป็นมาของภาษายูเอ็มแอล

จากการที่แนวคิดเรื่องเชิงวัตถุสามารถแก้ปัญหาในการออกแบบและพัฒนาระบบซอฟต์แวร์ที่มีขนาดใหญ่และมีความซับซ้อนได้อย่างมีประสิทธิภาพ ทำให้แนวคิดเรื่องเชิงวัตถุได้รับความนิยมจากผู้ออกแบบและพัฒนาระบบซอฟต์แวร์ในปัจจุบัน ทำให้มีผู้คิดค้นวิธีการพัฒนาซอฟต์แวร์เชิงวัตถุขึ้นหลายวิธี ผู้ใช้เกิดความสับสนว่าควรใช้วิธีการของผู้ใด ซึ่งถ้าเลือกวิธีการวิเคราะห์ออกแบบของผู้ใดก็จำเป็นต้องใช้วิธีการนำเสนอผลลัพธ์ของการวิเคราะห์ออกแบบตามที่ผู้สร้างกำหนดด้วย ซึ่งแต่ละวิธีการต่างก็มีการกำหนดสัญลักษณ์รูปภาพ (Notation) แตกต่างกันไป รวมทั้งจุดเด่นจุดด้อยของแต่ละวิธีการก็แตกต่างกันไปเช่นกัน ดังนั้นจึงเป็นการยากที่จะค้นหาวิธีการใดดีที่สุด ผู้ออกแบบและพัฒนาระบบซอฟต์แวร์ที่มีประสบการณ์มักจะใช้หลายวิธีการร่วมกัน อันเป็นที่มาของการรวมกันของวิธีการต่างๆ ในช่วงแรกเป็นการร่วมงานกันระหว่างวิธีการชื่อ Booch ของ Grady Booch และ OMT ของ James Rumbaugh เป็นการพัฒนากระบวนการพัฒนาซอฟต์แวร์เชิงวัตถุที่มีเป้าหมายให้เป็นหนึ่งเดียวกัน (Unified Method) โดยนำเอาวิธีการของ Booch และ OMT (Object Modeling Technique) มารวมกันและปรับปรุงใหม่ โดยมีความตั้งใจให้เป็นภาษาที่คล้ายกับ PL/1 ในสมัยก่อน คือ รวบรวมเอาภาษาทั้งหลายมาเป็นภาษาเดียว ต่อมา Ivar Jacobson ผู้พัฒนากระบวนการ OOSE หรือ Objectory Process ได้เข้าร่วมโครงการดังกล่าวภายหลัง ซึ่งในครั้งนั้นเป็นการสร้างภาษาโมเดลขึ้นใหม่ เรียกว่า

ภาษายูเอ็มแอล (UML - Unified Modeling Language)

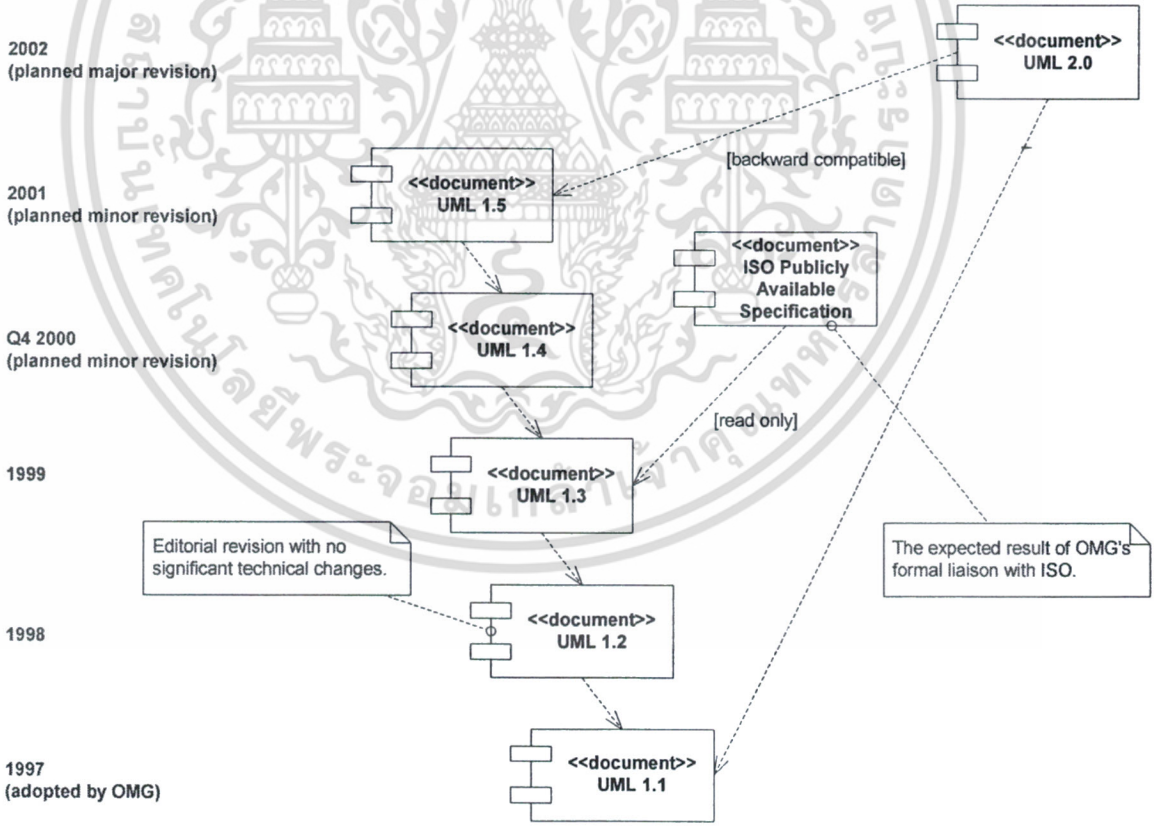
ภาพที่ 2.1
แสดงที่มาของภาษายูเอ็มแอล



เนื่องจากวิธีการพัฒนาซอฟต์แวร์เชิงวัตถุของทั้งสามมีชื่อเสียงอยู่แล้วในช่วงเวลานั้น ดังนั้นภาษายูเอ็มแอล ที่ถูกพัฒนาขึ้นมาใหม่จึงกลายเป็นที่นิยมใช้กันอย่างแพร่หลาย ในปี ค.ศ. 1997 ภาษายูเอ็มแอล รุ่นที่ 1.1 ได้ถูกเสนอให้กับหน่วยงานโอเอ็มจี (OMG - Object Management Group) กำหนดให้เป็นภาษาโมเดลมาตรฐาน (Standard Modeling Language) จากนั้นภาษายูเอ็มแอลถูกพัฒนาต่อโดยโอเอ็มจี ซึ่งในปัจจุบันภาษายูเอ็มแอลรุ่นที่เผยแพร่ออกสู่สาธารณชนคือ ภาษายูเอ็มแอล รุ่นที่ 1.4

ภาพที่ 2.2

แสดงพัฒนาการมาตรฐานของภาษายูเอ็มแอล ที่กำหนดมาตรฐานโดยโอเอ็มจี



จากการที่ภาษายูเอ็มแอลเป็นภาษาที่มีรูปภาพมาตรฐาน (Standard Visual Modeling Language) หรืออาจกล่าวอีกนัยได้ว่าภาษายูเอ็มแอลเป็นภาษาสากลที่ใช้ในการออกแบบและพัฒนาระบบซอฟต์แวร์เชิงวัตถุ ดังนั้น เอกสารการวิเคราะห์และออกแบบที่ถูกสร้างด้วยภาษายูเอ็มแอลจึงสามารถแลกเปลี่ยนและทำความเข้าใจตรงกันได้ระหว่างผู้ร่วมงานภายในกลุ่มผู้พัฒนาระบบ

นอกจากนี้ ภาษายูเอ็มแอลยังมีคุณสมบัติที่สามารถนำเสนอและสนับสนุนหลักการเชิงวัตถุได้อย่างครบถ้วนชัดเจน และไม่ผูกติดกับภาษาโปรแกรมภาษาใดภาษาหนึ่ง กล่าวคือ โมเดลที่ถูกสร้างขึ้นจากภาษายูเอ็มแอล สามารถถูกแปลงไปเป็นระบบจริงที่ถูกสร้างขึ้นด้วยภาษาโปรแกรมเชิงวัตถุใดก็ได้ อีกทั้งภาษายูเอ็มแอลเป็นภาษาที่ง่ายต่อการทำความเข้าใจ ผู้ที่ทำการศึกษาหรือนำไปใช้งานไม่จำเป็นต้องมีความรู้อื่นใดนอกจากแนวคิดเชิงวัตถุและจากการที่ภาษายูเอ็มแอลเป็นภาษาที่มีมาตรฐาน ผู้ออกแบบและพัฒนาจึงจำเป็นต้องศึกษาไวยากรณ์หรือโครงสร้างของภาษายูเอ็มแอล ก่อนนำไปใช้งาน ภาษายูเอ็มแอลประกอบด้วยองค์ประกอบพื้นฐาน 3 ส่วน คือ

ส่วนที่ 1 หน่วยโครงสร้างพื้นฐาน (Basic Building Block)

ส่วนที่ 2 กฎการใช้งาน (Rules)

ส่วนที่ 3 กลไกการทำงาน (Mechanism)

โดยองค์ประกอบที่มีความสำคัญและผู้ใช้จำเป็นต้องทำความเข้าใจให้ได้คือ ส่วนของหน่วยโครงสร้างพื้นฐาน ซึ่งส่วนนี้ประกอบด้วยองค์ประกอบย่อย 3 ส่วน คือ

<u>สัญลักษณ์ทั่วไป</u> (Things) คือ สัญลักษณ์พื้นฐานที่ถูกใช้งานในการสร้างแผนภาพต่างๆ ของภาษายูเอ็มแอลถือว่าเป็นรูปแบบที่เล็กที่สุดของโมเดล สัญลักษณ์ทั่วไปแบ่งได้ออกเป็น 4 หมวด คือ

หมวดโครงสร้าง (Structural Things)

หมวดพฤติกรรม (Behavioral Things)

หมวดการจัดกลุ่ม (Grouping Things)

หมวดคำอธิบาย (Annotation Things)

ความสัมพันธ์ (Relationships) เป็นสิ่งที่ใช้แสดงความสัมพันธ์ระหว่างสัญลักษณ์ทั่วไป มี 4 ชนิด คือ

ความสัมพันธ์แบบขึ้นต่อกัน (Dependency Relationship): สิ่งสองสิ่งที่มีความสัมพันธ์กันเช่นนี้ คุณสมบัติของสิ่งหนึ่งขึ้นอยู่กับคุณสมบัติของอีกสิ่งหนึ่ง

ความสัมพันธ์แบบเกี่ยวข้องกัน (Association Relationship): สิ่งสองสิ่งที่มีความสัมพันธ์เชื่อมโยงกัน

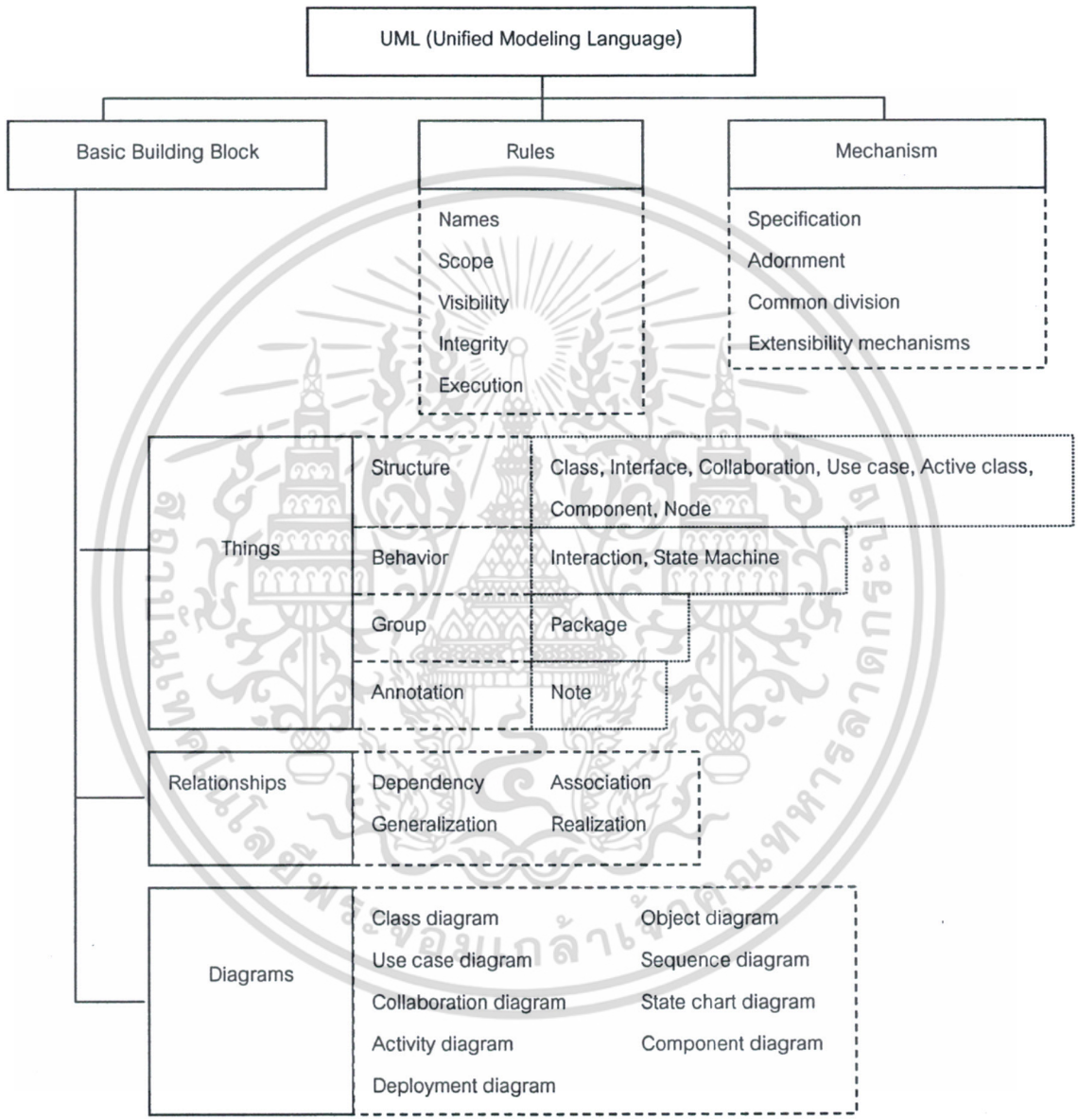
ความสัมพันธ์แบบทั่วไป (Generalization Relationship): สิ่งสองสิ่งที่มีความสัมพันธ์กันเช่นนี้ คือ คุณสมบัติของสิ่งหนึ่งเป็นคุณสมบัติพื้นฐานของอีกสิ่งหนึ่ง ซึ่งอาจจะมีคุณสมบัติมากกว่าคุณสมบัติพื้นฐานนั้น

ความสัมพันธ์แบบต้นแบบ (Realization Relationship): สิ่งหนึ่งถูกสร้างให้มีคุณสมบัติของอีกสิ่งหนึ่ง

แผนภาพ (Diagrams) คือ แผนภาพที่เกิดจากแนวคิดที่ว่าสัญลักษณ์ทั่วไปใดๆก็ตาม ถ้ามีคุณสมบัติบางประการที่สามารถจัดให้อยู่ในกลุ่มเดียวกันได้ ก็จะใช้แผนภาพมาจัดกลุ่มให้แก่สัญลักษณ์เหล่านั้น

ภาพที่ 2.3

แผนภาพแสดงองค์ประกอบของภาษายูเอ็มแอล



นอกจากองค์ประกอบของภาษายูเอ็มแอลที่ผู้ออกแบบและพัฒนาต้องทำความเข้าใจแล้ว การเลือกใช้งานแผนภาพ ก็เป็นอีกประการหนึ่งที่มีความสำคัญ สาเหตุที่ภาษายูเอ็มแอลต้องมีแผนภาพที่หลากหลาย เนื่องจากในระบบที่มีขนาดใหญ่และมีความซับซ้อนมาก ยังไม่มีวิธีการแก้ไขปัญหาที่ใช้แผนภาพเพียงหนึ่งแผนภาพแล้วสามารถให้มุมมองได้ทั้งหมด ทั้งมุมมองเชิงโครงสร้างของระบบ (Static View) และมุมมองเชิงพฤติกรรมของระบบ (Behavior View) ดังแสดงในตารางผนวก ก.1 จำเป็นต้องมีการเลือกแผนภาพหนึ่งมาใช้ในมุมมองหนึ่งและเลือกอีกแผนภาพหนึ่งมาใช้ในอีกมุมมองหนึ่ง เพื่อให้ได้แผนภาพที่สามารถเสนอมุมมองได้อย่างครบถ้วน จำเป็นต้องใช้หลายแผนภาพโดยต้องเข้าใจจุดประสงค์ของแต่ละแผนภาพและเลือกใช้แผนภาพให้ตรงกับมุมมองที่ต้องการ

ตารางที่ 2.1
แสดงแผนภาพในภาษายูเอ็มแอล
โดยจำแนกตามมุมมองเชิงโครงสร้างและมุมมองเชิงพฤติกรรมของระบบ

แผนภาพแสดงโครงสร้าง (Static)	แผนภาพแสดงพฤติกรรม (Dynamic)
แผนภาพคลาส (Class Diagram)	แผนภาพยูสเคส (Use Case Diagram)
แผนภาพออบเจ็ค (Object Diagram)	แผนภาพซีเควนซ์ (Sequence Diagram)
แผนภาพคอมโพเนนต์ (Component Diagram)	แผนภาพคอลแลบอเรชั่น (Collaboration Diagram)
แผนภาพดีพลอยเมนต์ (Deployment Diagram)	แผนภาพสเตตทรานซิชั่น (State Transition Diagram)
	แผนภาพแอ็กทิวิตี้ (Activity Diagram)

ตารางที่ 2.2

แสดงมุมมองและแผนภาพที่สอดคล้องกับมุมมอง

มุมมอง	แผนภาพที่สอดคล้อง
มุมมองยูสเคส พฤติกรรม (Behavior)	แผนภาพยูสเคส แผนภาพแอ็กทิวิตี้ (สำหรับโมเดลแสดงพฤติกรรม)
มุมมองออกแบบ คำศัพท์ (Vocabulary) การทำงาน (Functionality)	แผนภาพคลาส (สำหรับโมเดลแสดงโครงสร้าง) แผนภาพอินเตอร์แอ็กชั่น (สำหรับโมเดลแสดงพฤติกรรม) แผนภาพสเตตทรานซิชั่น (สำหรับโมเดลแสดงโครงสร้าง)
มุมมองการดำเนินการ ประสิทธิภาพ (Performance) Scalability อัตราการให้ผลผลิตต่อหน่วย (Throughput)	แผนภาพคลาส (สำหรับโมเดลแสดงโครงสร้าง) แผนภาพอินเตอร์แอ็กชั่น (สำหรับโมเดลแสดงพฤติกรรม)
อิมพลีเมนท์เทชั่น System assembly Configuration management	แผนภาพคอมโพเนนต์
มุมมองดีพลอยเมนต์ สถาปัตยกรรมระบบ (System topology) การกระจาย (Distribution) การส่งมอบ (Delivery) การติดตั้ง (Installation)	แผนภาพดีพลอยเมนต์

ปัจจุบันแผนภาพในภาษายูเอ็มแอล สามารถแบ่งออกได้เป็น 9 แผนภาพ โดยแต่ละแผนภาพเปรียบเสมือนมุมมองในด้านต่างๆของระบบที่ผู้ออกแบบและวิเคราะห์กำลังจะพัฒนา โดยในแต่ละแผนภาพที่แสดงมุมมองต่างๆ ประกอบด้วยสัญลักษณ์ที่ใช้สื่อความหมายเฉพาะภายในแผนภาพนั้น

นอกจากความรู้พื้นฐานที่เกี่ยวข้องกับองค์ประกอบและแผนภาพของภาษายูเอ็มแอลแล้ว สิ่งที่ทำให้การใช้งานภาษายูเอ็มแอลเป็นไปอย่างประสบผลสำเร็จอีกอย่างก็คือ ผู้ออกแบบและพัฒนาจำเป็นต้องเรียนรู้กระบวนการในการเลือกใช้แผนภาพตามความเหมาะสม ในขั้นตอนกระบวนการพัฒนาซอฟต์แวร์เชิงวัตถุ ซึ่งในส่วนนี้ อธิบายถึงการใช้งานแผนภาพของภาษายูเอ็มแอล เพื่อสร้างโมเดลระบบอย่างเป็นขั้นตอนสอดคล้องกับกระบวนการพัฒนาซอฟต์แวร์โปรเซสเทรดดิชั่น (Software Process Tradition) ทั้งนี้ในการพัฒนาระบบโดยทั่วไป ผู้พัฒนาอาจไม่จำเป็นต้องสร้างแผนภาพให้ครบทุกแผนภาพเสมอไป ผู้พัฒนาสามารถปรับเปลี่ยนการใช้งานแผนภาพได้ตามความเหมาะสม และเมื่อถึงจุดๆหนึ่งที่ผู้ออกแบบและพัฒนามีประสบการณ์มากพอ ก็อาจสามารถคิดค้นกระบวนวิธีใช้งานที่ตนถนัดขึ้นใช้เองได้

<u>เฟสการวิเคราะห์ความต้องการของระบบหรือผู้ใช้ (Requirement Analysis)</u> วัตถุประสงค์ของเฟสแรกนี้ คือ การทำความเข้าใจกับขอบเขตของปัญหา (Problem Domain) การกำหนดขอบเขตของการพัฒนาและฟังก์ชันการทำงานต่างๆของระบบที่พัฒนา

สิ่งที่ผู้พัฒนาต้องทำในขั้นตอนนี้ คือ การนำเอาความต้องการแบบ Functionality หรือ Functional Requirements ซึ่งบ่งบอกถึงความสามารถหรือฟังก์ชันที่ผู้ใช้สามารถเรียกใช้จากระบบมาแปลงเป็นโมเดลความต้องการของผู้ใช้งาน ซึ่งแผนภาพยูสเคส ถูกใช้สำหรับสร้างโมเดลนี้ ที่ขาดไม่ได้ในแผนภาพยูสเคส คือ คำบรรยายยูสเคส ซึ่งเป็นรายละเอียดของแต่ละฟังก์ชันว่าเริ่มต้นอย่างไร มีการดำเนินเหตุการณ์เกิดขึ้นอย่างไรและสิ้นสุดลงอย่างไร เรียกรวมๆว่า ลำดับการเกิดเหตุการณ์ (Flow of Event) รวมถึงเหตุการณ์ยกเว้น (Exception Flow of Event) ที่อาจเกิดขึ้นระหว่างการปฏิบัติฟังก์ชันดังกล่าวของระบบ ก็ต้องถูกบันทึกด้วยเช่นกัน

เฟสการวิเคราะห์ระบบ (Domain Analysis) ในเฟสที่สองนี้เป็นการบรรยายถึงโครงสร้างและพฤติกรรมของระบบที่กำลังพัฒนา ซึ่งกิจกรรมสำคัญต่างๆที่ผู้พัฒนาต้องกระทำในเฟสนี้ ได้แก่

a) การสร้างแผนภาพคลาส เพื่อศึกษาโครงสร้าง (Structure) หรือส่วนที่เป็นโครงสร้างของระบบ โดยการ

I. หาคลาสทั้งหมดที่มีในระบบโดยใช้เทคนิค Heuristic Mapping คือ ให้คำนามที่พบในคำบรรยายยูสเคสที่ได้จากเฟสแรกมาแปลงเป็นคลาส คำกิริยาจะถูกแปลงเป็นโอเปอเรชั่น (Operation) และคำวิเศษณ์ถูกแปลงเป็นแอตทริบิวส์ (Attribute) ทั้งนี้ไม่ได้หมายความว่าทุกคำนามถูกแปลงเป็นคลาสเสมอไป คำนามบางคำเขียนต่างกันแต่หมายถึงสิ่งเดียวกัน หรือคำนามบางคำอาจไม่จำเป็นที่ต้องถูกสร้างขึ้นในระบบ ทั้งนี้การศึกษาระบบที่มีอยู่ การฝึกฝน รวมทั้งประสบการณ์ถือเป็นสิ่งสำคัญที่ช่วยให้ผู้พัฒนาค้นพบคลาสเป้าหมายและรายละเอียดของคลาสได้อย่างถูกต้อง

II. ค้นหาความสัมพันธ์ (Relationships) ระหว่างคลาส อันได้แก่ การขึ้นต่อกัน (Dependency), การถ่ายทอดคุณสมบัติ (Inheritance), ความสัมพันธ์แบบเชื่อมโยง (Association) ทั้งแบบสัมพันธ์แบบองค์ประกอบร่วม (Composition) และสัมพันธ์แบบมีส่วนร่วม (Aggregation) ด้วยเช่นกัน

b) การสร้างแผนภาพซีเควนซ์ เพื่อศึกษาพฤติกรรม (Behavior) หรือส่วนที่มีความเป็นพลวัตของระบบ โดยการสร้างแผนภาพซีเควนซ์ สร้างขึ้นสำหรับแต่ละยูสเคสในแผนภาพยูสเคสจากเฟสแรก นอกจากนี้การสร้างแผนภาพซีเควนซ์ยังช่วยให้พบโอเปอเรชั่นเพิ่มเติมขึ้นอีกด้วย

วิธีการพัฒนาซอฟต์แวร์เชิงวัตถุแบบเทรดดิชั่น เป็นไปในลักษณะของการวนซ้ำและขยายต่อเติมการพัฒนาขึ้นเรื่อยๆจนสมบูรณ์ ซึ่งลักษณะดังกล่าวเริ่มปรากฏในเฟสนี้ กล่าวคือ แทนที่ต้องทำการวิเคราะห์ทุกยูสเคสจากแผนภาพยูสเคสในครั้งเดียว ซึ่งหากเป็นระบบที่ซับซ้อนและมีขนาดใหญ่ก็อาจประกอบไปด้วยยูสเคสหลายร้อยยูสเคส ดังนั้น ผู้พัฒนาสามารถเลือกเพียงหนึ่งยูสเคสมาทำการวิเคราะห์หาคลาสและความสัมพันธ์รวมถึงลักษณะไดนามิคของระบบในเฟสนี้ ครั้นเมื่อวงจรการพัฒนากลับมาที่เฟสนี้อีกครั้ง ก็เป็นการวิเคราะห์ออกแบบยูสเคสถัดๆไปนั่นเอง กล่าวโดยสรุป ในเฟสของการวิเคราะห์เป็นการโมเดลปัญหา ซึ่งผลที่ได้ในเฟสนี้คือ เอกสารที่บันทึกโมเดลการวิเคราะห์ระบบ (Analysis Model Document) อันประกอบไปด้วย แผนภาพคลาส, แผนภาพซีเควนซ์ นอกจากนี้อาจมีการใช้สัญลักษณ์แพ็กเกจหรือแผนภาพอื่นๆ เช่น แผนภาพ

คอลแลบอเรชั่น, แผนภาพสเตตทรานซิชั่น, แผนภาพแอ็กทิวิตี้ มาช่วยในการสร้างโมเดลด้วยเช่นกัน

<u>เฟสการออกแบบ (Design)</u> ในเฟสการออกแบบระบบผู้พัฒนาจะทำการกำหนดรายละเอียดเชิงเทคนิคของระบบให้พร้อม เพื่อนำไปอิมพลิเมนท์จริงในเฟสถัดไป หากพูดในเชิงหลักการ จะเรียกได้ว่าเฟสนี้เป็นเฟสการค้นหาวิธีการแก้ปัญหาภายหลังจากการทำความเข้าใจปัญหาในเฟสก่อนหน้านี้ สิ่งที่ต้องเตรียมสำหรับใช้ในเฟสนี้ คือ โมเดลการวิเคราะห์จากเฟสที่สอง ซึ่งอาจรวมถึงความต้องการแบบ Nonfunctional ก็จะถูกนำมาพิจารณาในการออกแบบเฟสนี้ด้วยเช่นกัน กิจกรรมในการเฟสการออกแบบระบบมีดังต่อไปนี้

a) การเพิ่มเติมคลาสหรือแพ็กเกจลงไปภายในโมเดลการวิเคราะห์ระบบที่ได้จากเฟสที่สอง ตัวอย่างเช่น การเพิ่มแพ็กเกจที่เกี่ยวข้องกับฐานข้อมูล การติดต่อสื่อสาร เป็นต้น โดยแพ็กเกจหรือคลาสเหล่านี้จะทำงานร่วมกันกับเพ็กเกจเดิมที่มีอยู่ในโมเดลการวิเคราะห์ระบบ นอกจากนี้ ยังรวมถึงการแก้ไขปรับปรุงเพิ่มเติมแอตทริบิวส์หรือโอเปอเรชั่นของคลาสต่างๆในโมเดลการวิเคราะห์ระบบด้วยเช่นกัน

b) จัดหาไลเบอรี่คลาส (Library Class) หรือคอมโพเนนต์จากที่อื่นเพื่อนำมาใช้อีกครั้งเพื่อช่วยลดเวลาในการพัฒนาระบบ (Reuse of Class Libraries)

c) กำหนดรายละเอียดส่วนติดต่อกับผู้ใช้ของระบบ (User Interface Design)

d) หาวิธีจัดการกับข้อผิดพลาดที่อาจจะเกิดขึ้นในการใช้งานระบบ (Exception Handling)

e) ออกแบบสถาปัตยกรรมระบบ (System Architecture Design) เพื่อทำการพิจารณาที่อยู่หรือตำแหน่งการติดตั้งของคอมโพเนนต์หรือเพ็กเกจต่างๆ พิจารณาว่าคลาสใดควรอยู่ใน Component หรือ ไฟล์ใดควรติดตั้งไว้ที่ระบบคอมพิวเตอร์ส่วนใด นั่นคือ การสร้างแผนภาพคอมโพเนนต์และแผนภาพดีพลอยเมนต์นั่นเอง สำหรับรูปแบบของสถาปัตยกรรมที่ใช้กันโดยทั่วไป คือ สถาปัตยกรรมแบบกระจาย (Distributed System) ซึ่งได้แก่ 2-เทียร์ (2-Tier), 3-เทียร์ (3- Tier) และมัลติเทียร์ (Multi-Tier System)

f) ออกแบบส่วนที่เป็น Nonfunctional Requirement เช่น กลุ่มของผู้ใช้งานมีความต้องการใช้งานระบบใหม่ร่วมกับฐานข้อมูลหรือระบบเดิมที่มีอยู่ (Legacy System Integration)

g) นอกจากนี้ การออกแบบยังรวมถึงการตัดสินใจเลือกใช้เทคโนโลยีต่างๆที่มีอยู่ให้

เหมาะสมกับความต้องการและงบประมาณของผู้ใช้งานด้วยเช่นกัน

กล่าวโดยสรุป แผนภาพยูเอ็มแอลที่ถูกสร้างในเฟสนี้ได้แก่ แผนภาพคอมโพเนนต์และแผนภาพดีพลอยเมนต์ ในส่วนของแผนภาพคลาสและแผนภาพซีเควนซ์จะถูกเพิ่มรายละเอียดเชิงเทคนิค เรียกผลลัพธ์รวมของเฟสนี้ว่า โมเดลการออกแบบระบบ (Design Model) ซึ่งประกอบไปด้วยแผนภาพต่างๆข้างต้น รวมถึงข้อกำหนดด้านอื่นๆ ซึ่งระบุถึงรายละเอียดของความต้องการแบบ Nonfunctional เทคนิควิธีการแก้ปัญหา และเอกสารการออกแบบ (UML Design Document) นี้จะถูกส่งต่อไปให้โปรแกรมเมอร์เพื่อนำไปพัฒนาในเฟสถัดไป

<u>เฟสการสร้างโปรแกรมระบบ (Construction)</u> วัตถุประสงค์หลักของเฟสนี้ คือ การแปลงผลที่ได้จากเฟสการออกแบบไปเป็นโค๊ต กล่าวคือ นักวิเคราะห์ออกแบบระบบจะต้องทำการสร้างโมเดลระบบที่สมบูรณ์อันเป็นข้อมูลสำคัญสำหรับโปรแกรมเมอร์

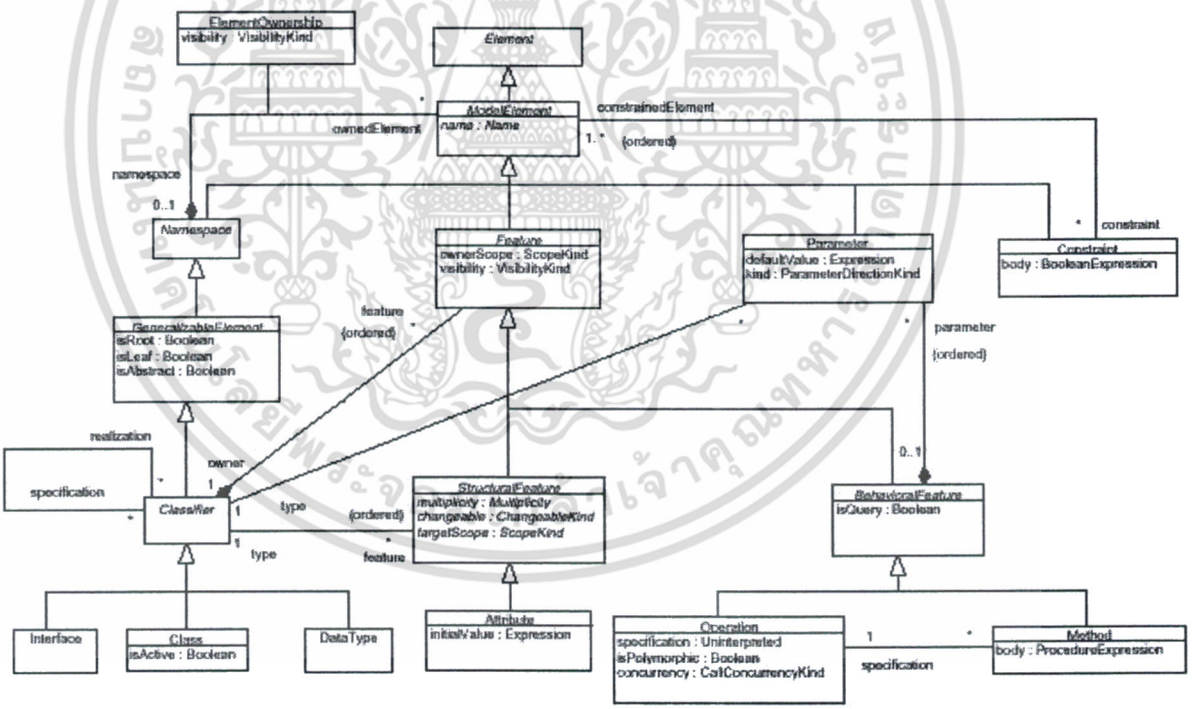
<u>เฟสการทดสอบระบบ (Testing)</u> ในเฟสนี้ ผู้พัฒนาจะต้องทำการค้นหาข้อผิดพลาด (Error) ภายในระบบที่กำลังพัฒนา ซึ่ง โดยปกติการทดสอบระบบจะอ้างอิงผลการวิเคราะห์ความต้องการผู้ใช้งานระบบในเฟสแรกเป็นหลักว่าเป็นไปตามความต้องการของผู้ใช้งานจริงอย่างครบถ้วนหรือไม่ทั้งด้านความต้องการแบบ Functional และ Nonfunctional

2.2 วากยสัมพันธ์อธิบายโครงสร้างของแผนภาพในภาษายูเอ็มแอล

วากยสัมพันธ์ (Syntax) เป็นกฎเกณฑ์ของภาษา ในการแปลความภาษาหนึ่งๆ ผู้ที่ทำหน้าที่การแปลงภาษาต้องมีความเข้าใจในเรื่องของวายกสัมพันธ์ วากยสัมพันธ์ของภาษายูเอ็มแอลเป็นสิ่งหนึ่งที่มีความสำคัญในงานวิจัย เนื่องจากภาษายูเอ็มแอลมีแนวคิดของภาษามาจากแนวคิดเชิงวัตถุเช่นเดียวกับภาษาจาวา

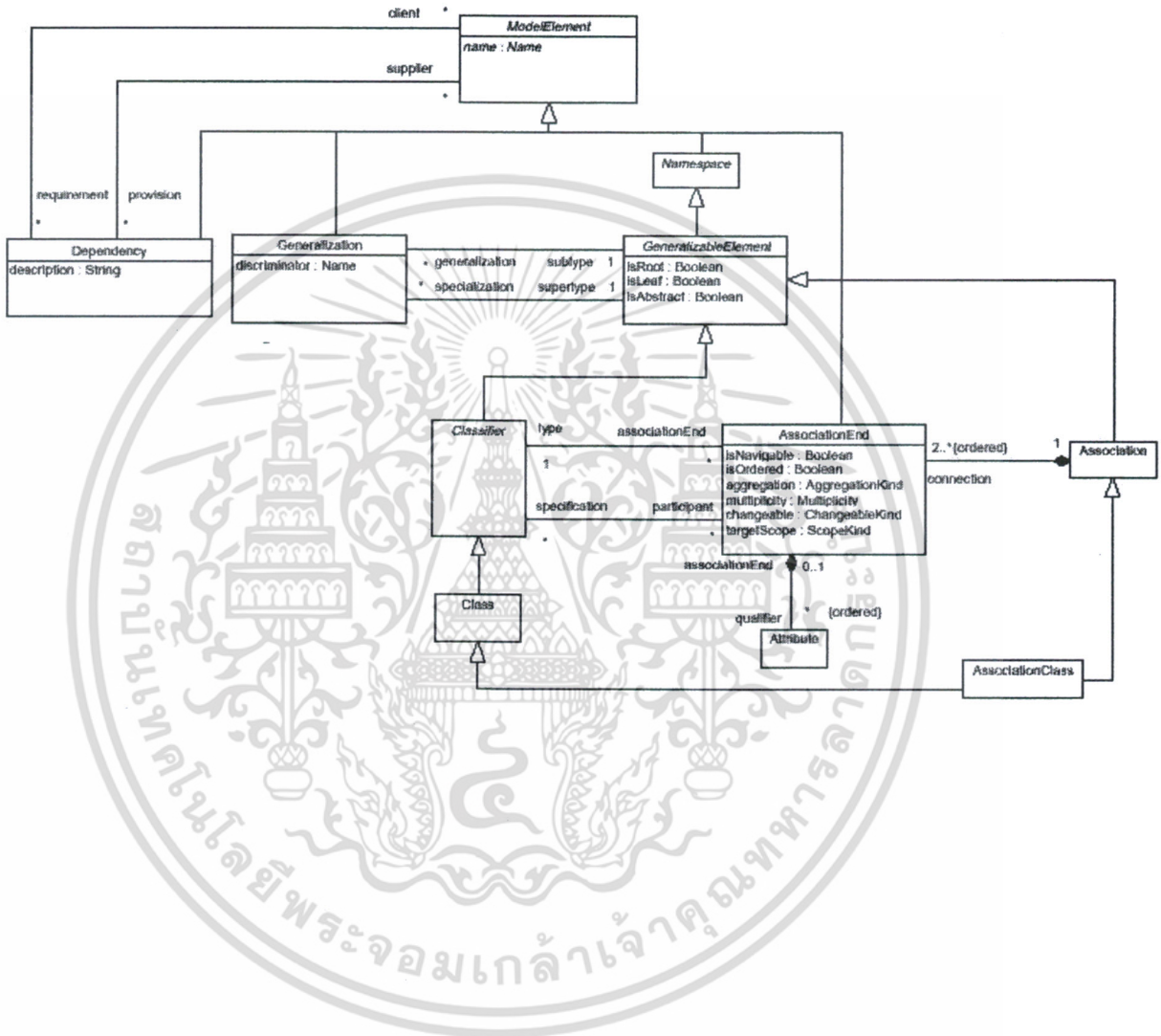
ภาพที่ 2.4

ส่วนหลัก - แกน (Core Package-Backbone)



ภาพที่ 2.5

ส่วนหลัก – ความสัมพันธ์ (Core Package - Relationships)

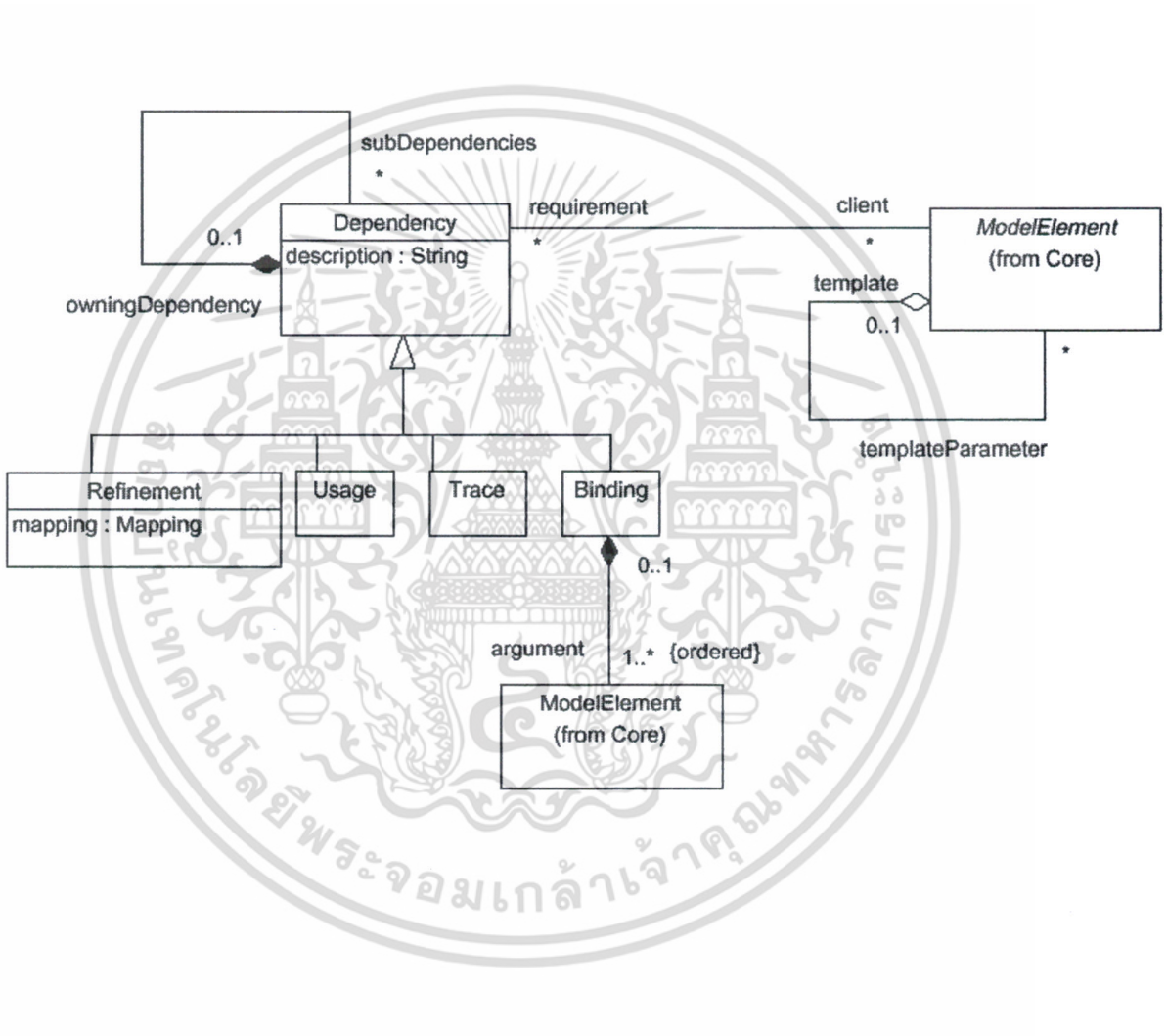


วากยสัมพันธ์ของแพ็กเกจที่เป็นองค์ประกอบพื้นฐานในส่วนเสริม

ภาพที่ 2.6

ส่วนเสริม – การขึ้นต่อกันและเท็มเพลต

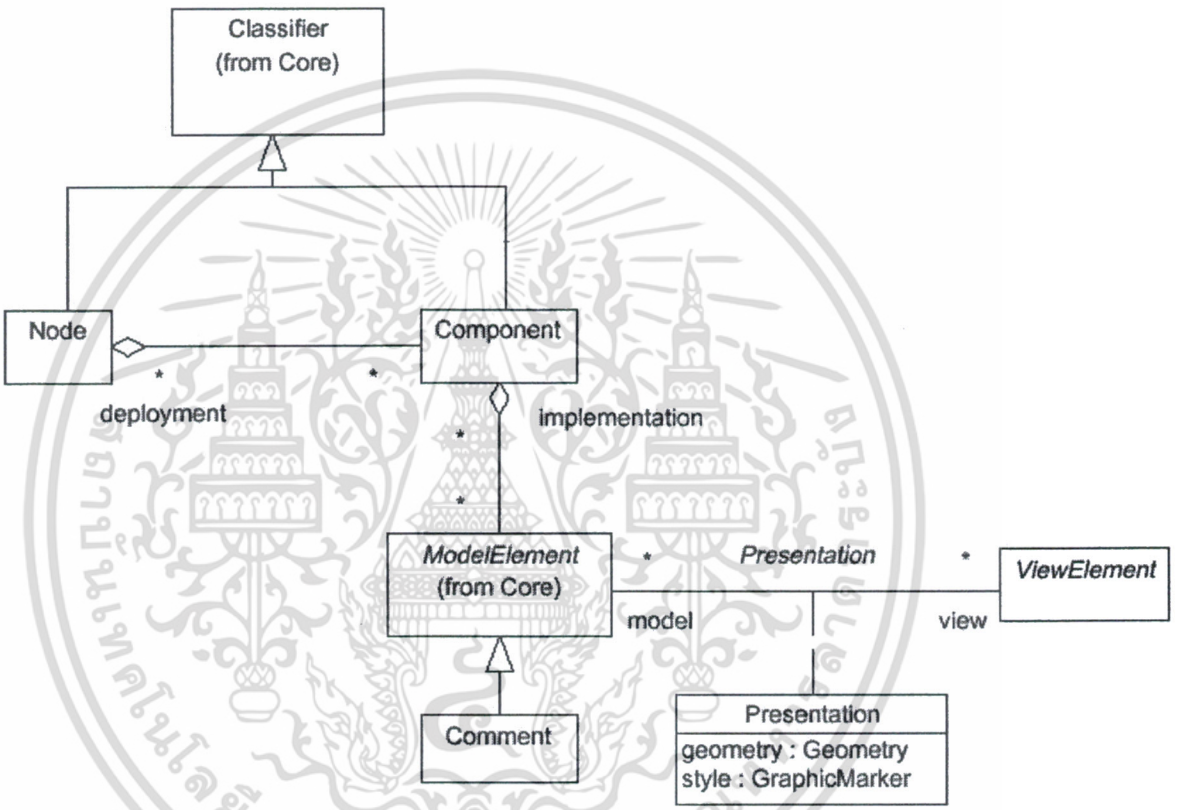
(Auxiliary Elements – Dependencies and Templates)



ภาพที่ 2.7

ส่วนเสริม – โครงสร้างทางกายภาพและมุมมองของอิลิเมนท์

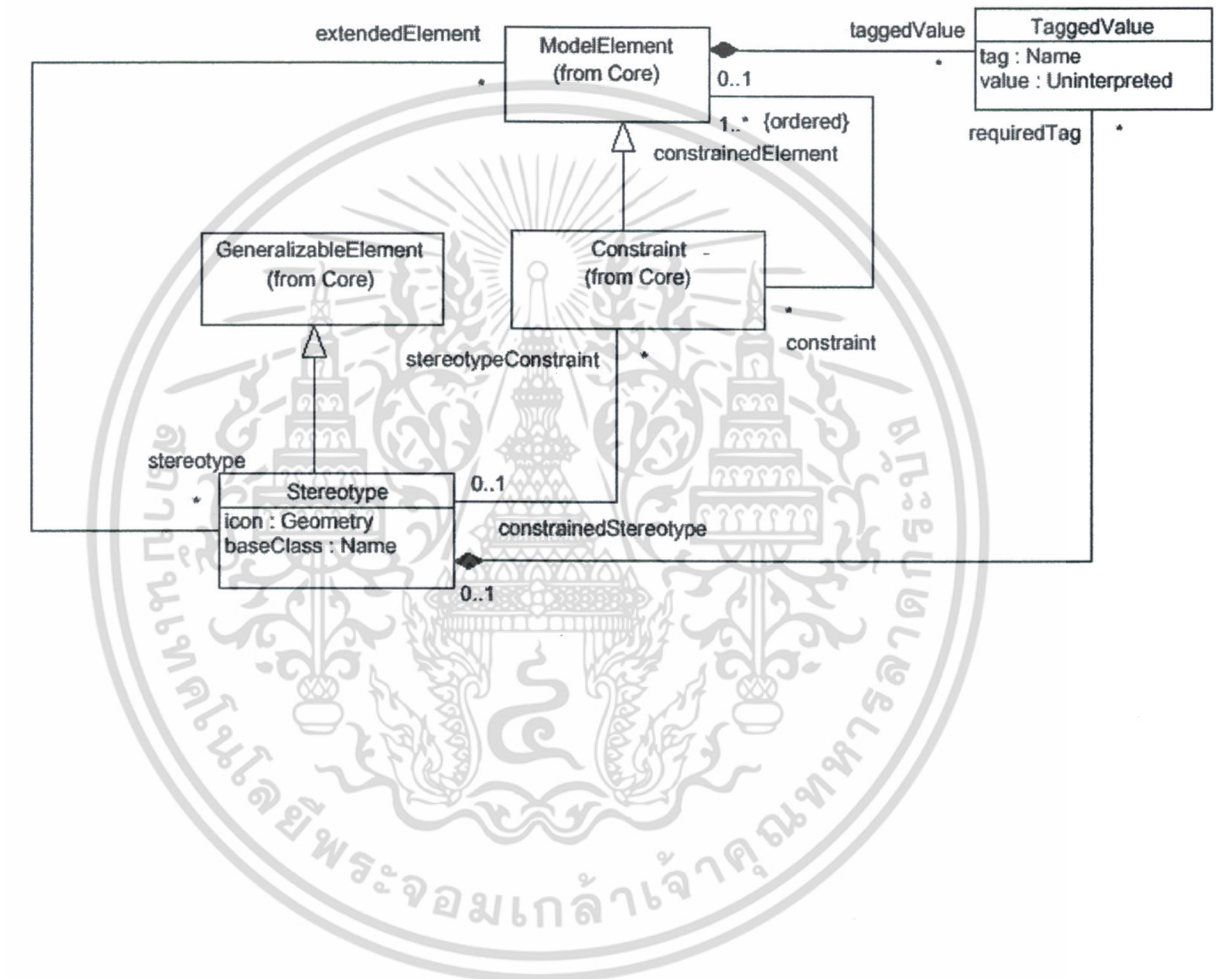
(Auxiliary Elements – Physical Structures and View Elements)



วากยสัมพันธ์ของแพ็กเกจที่เป็นองค์ประกอบพื้นฐานในส่วนกลไกสำหรับขยาย

ภาพที่ 2.8

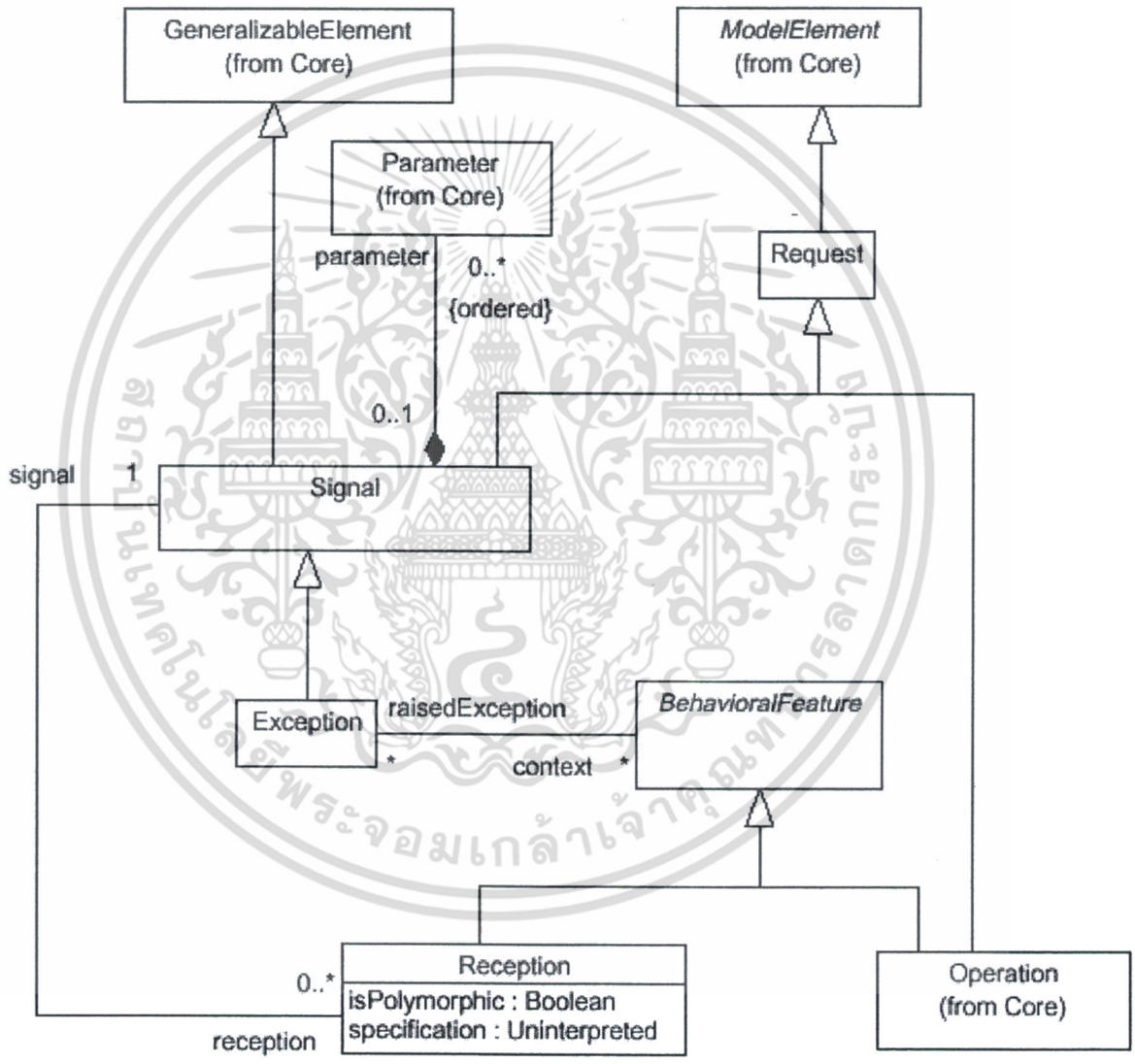
ส่วนกลไกสำหรับขยาย (Extension Mechanisms)



วากยสัมพันธ์แพ็กเกจที่เก็บองค์ประกอบที่เกี่ยวข้องกับพฤติกรรมในส่วนพฤติกรรมสามัญ

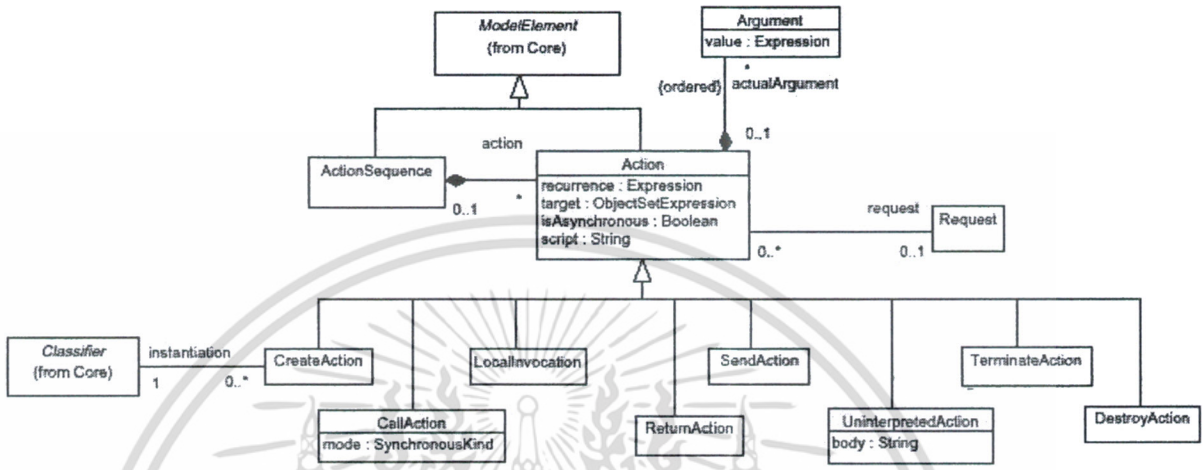
ภาพที่ 2.9

พฤติกรรมสามัญ – การร้องขอ (Common Behavior – Requests)



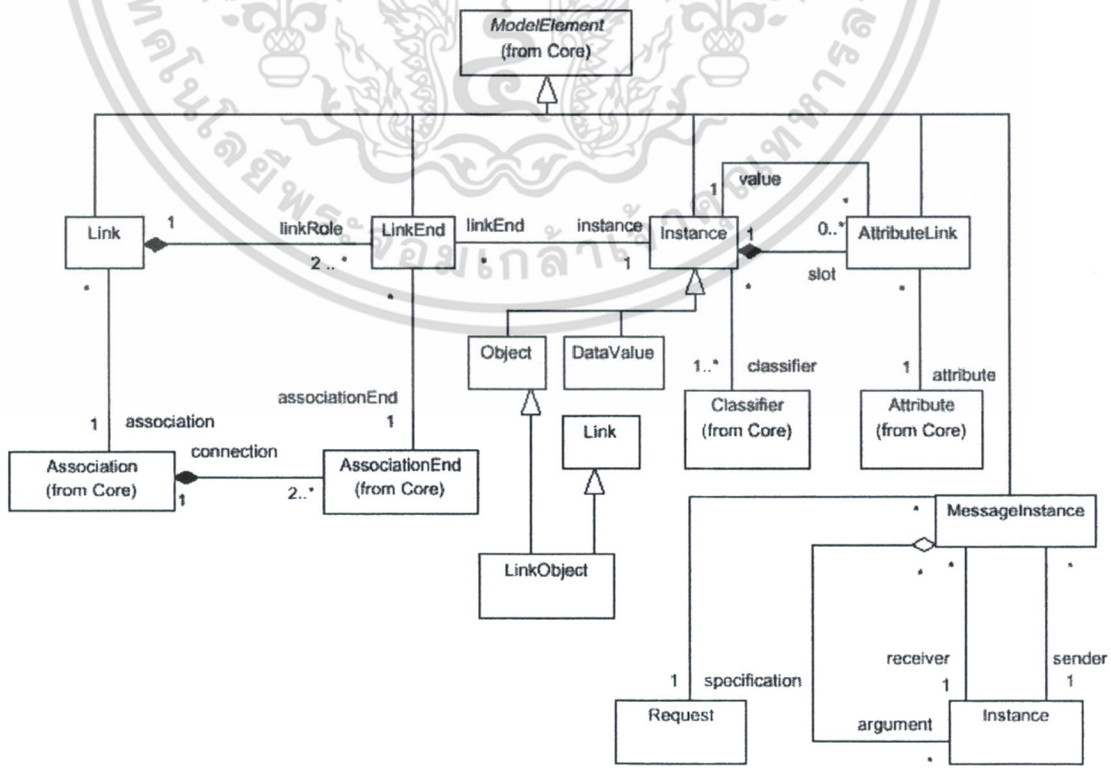
ภาพที่ 2.10

พฤติกรรมสามัญ – การกระทำ (Common Behavior – Actions)



ภาพที่ 2.11

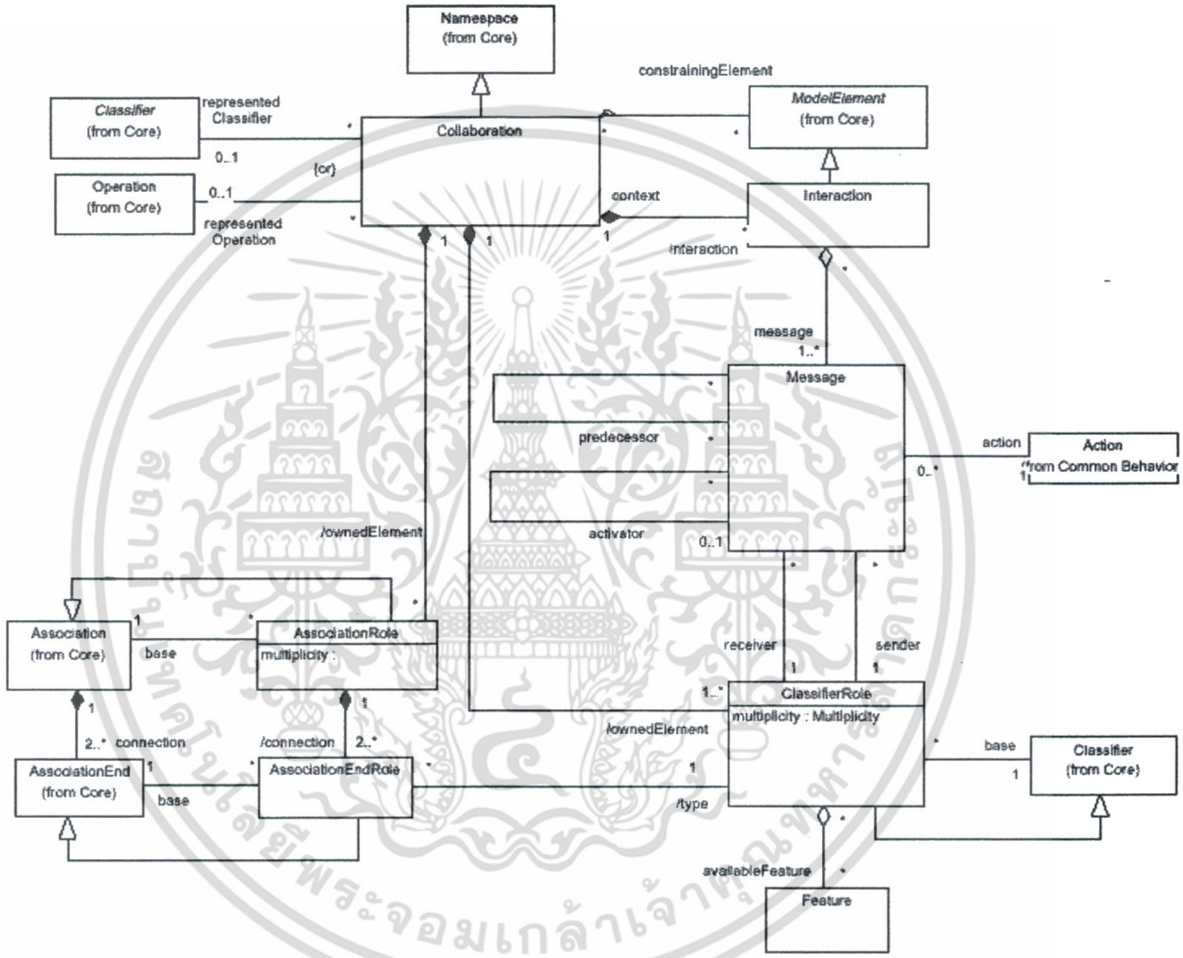
พฤติกรรมสามัญ – อินสแตนส์และลิงค์ (Common Behavior – Instances and Links)



วากยสัมพันธ์แพ็กเกจที่เก็บองค์ประกอบที่เกี่ยวข้องกับพฤติกรรมในส่วนคอลแลบอเรชั่น

ภาพที่ 2.12

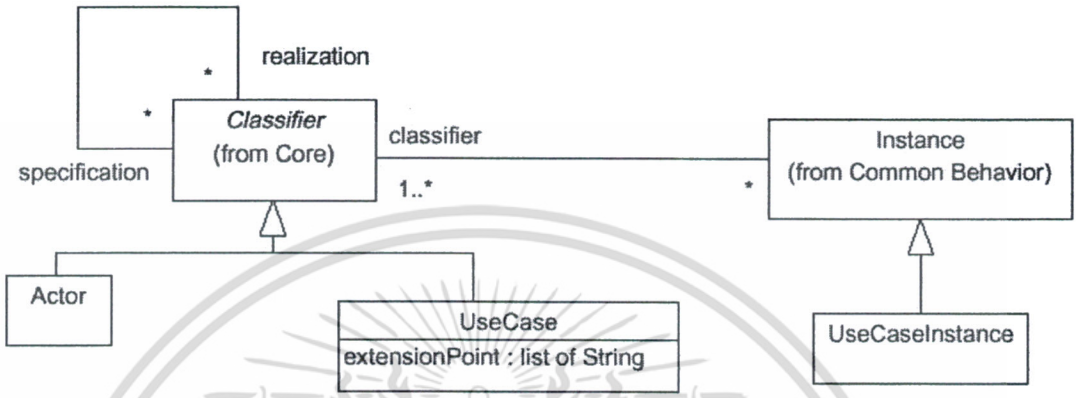
คอลแลบอเรชั่น (Collaborations)



วากยสัมพันธ์แพ็กเกจที่เก็บองค์ประกอบที่เกี่ยวข้องกับพฤติกรรมในส่วนยูสเคส

ภาพที่ 2.13

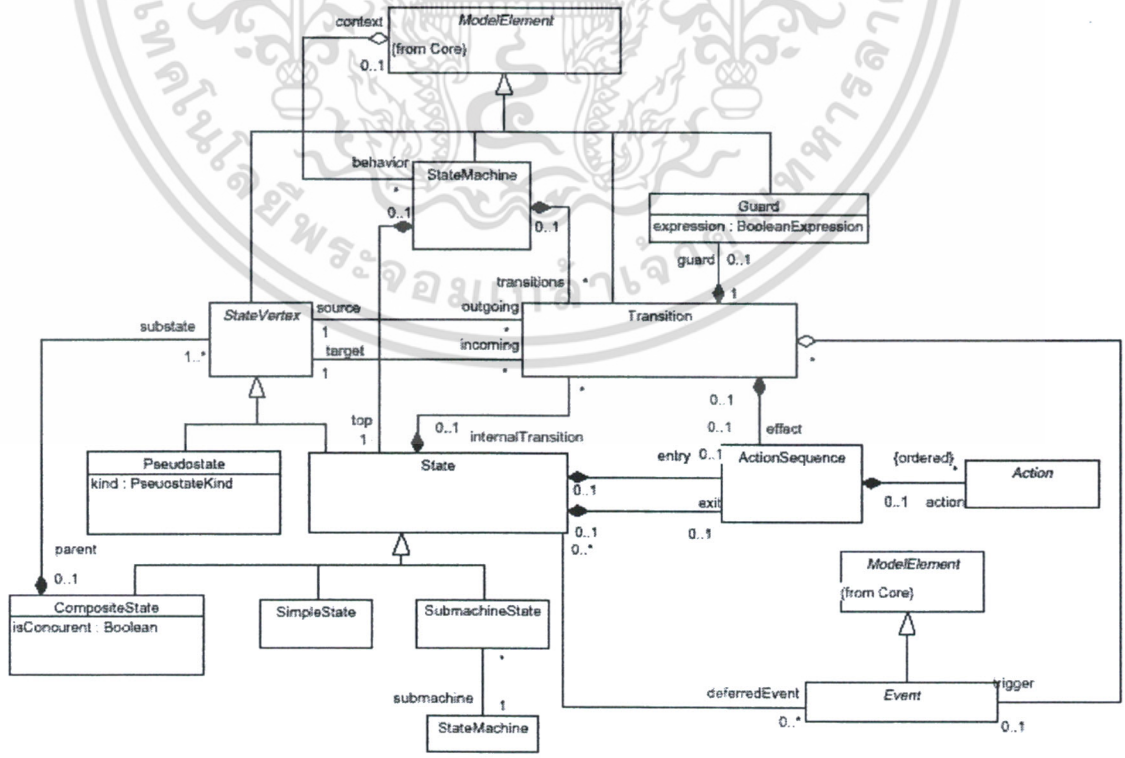
ยูสเคส (Use Cases)



วากยสัมพันธ์แพ็กเกจที่เก็บองค์ประกอบที่เกี่ยวข้องกับพฤติกรรมในส่วนสเตตแมชชีน

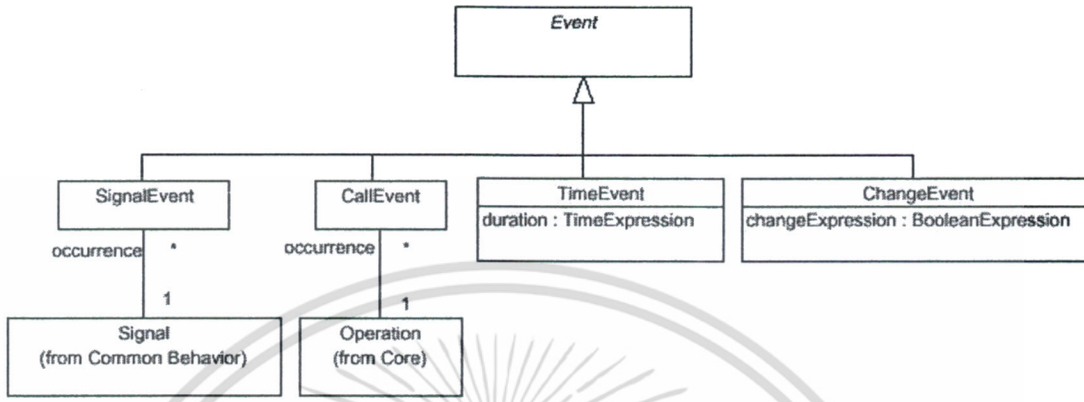
ภาพที่ 2.14

สเตตแมชชีน - หลัก (State Machines – Main)



ภาพที่ 2.15

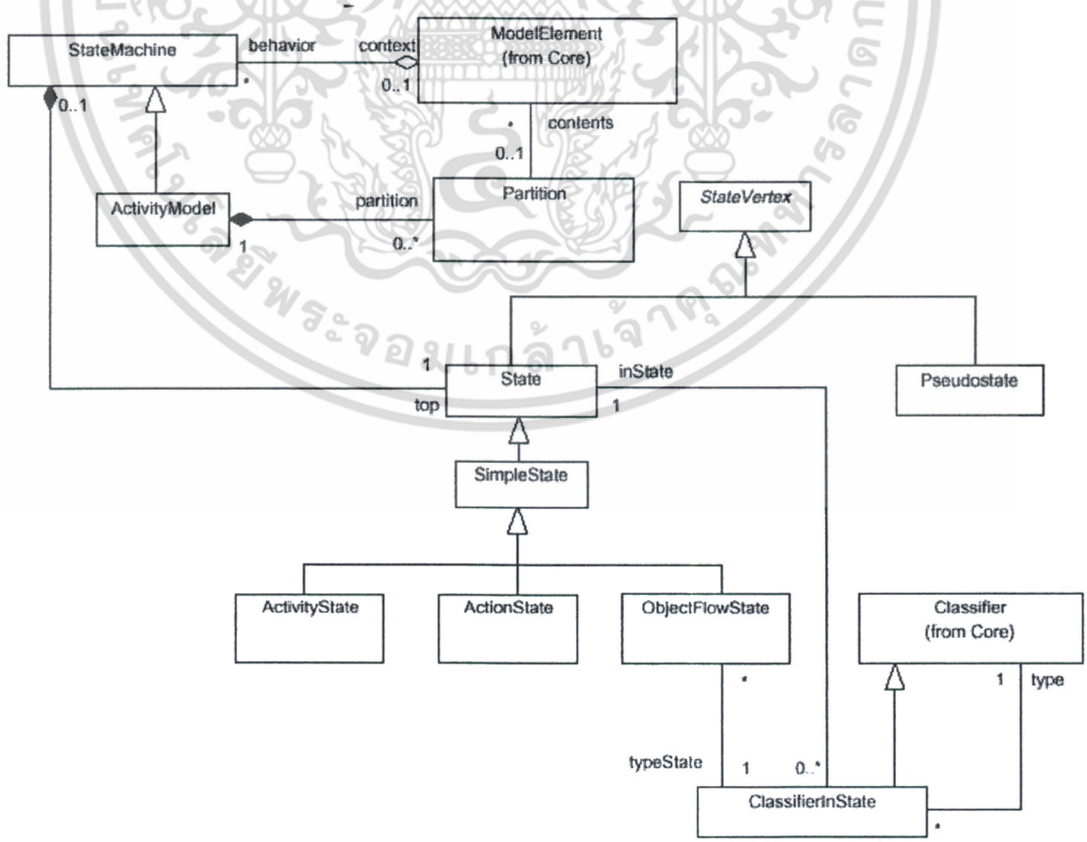
สเตตแมชชีน – เหตุการณ์ (State Machines – Events)



วากยสัมพันธ์แพ็กเกจที่เก็บองค์ประกอบที่เกี่ยวข้องกับพฤติกรรมในส่วนแอตติวิตี้

ภาพที่ 2.16

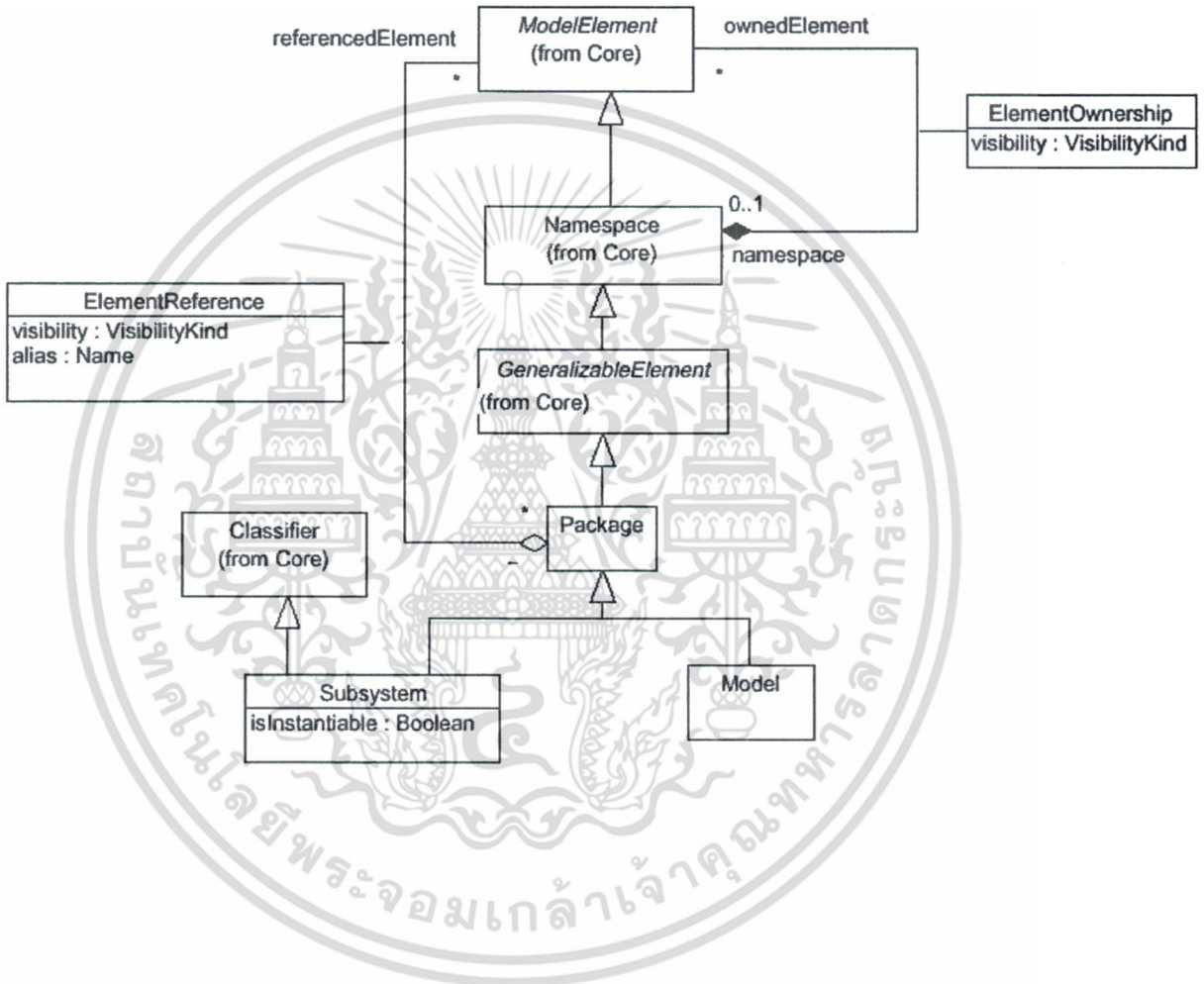
แอกติวิตี้โมเดล (Activity Models)



วากยสัมพันธ์แพ็กเกจที่เก็บองค์ประกอบที่ใช้สำหรับจัดกลุ่ม

ภาพที่ 2.17

การจัดกลุ่ม (Model Management)



2.3 แนวทางในการออกแบบคุณสมบัติทางสถาปัตยกรรม

จากงานศึกษาความรู้ที่เกี่ยวข้องกับงานวิจัยเบื้องต้น จะเห็นได้ว่าภาษายูเอ็มแอลเป็นภาษาที่มีโครงสร้างที่ชัดเจนแน่นอน และสามารถบรรยายหลักการได้ด้วยแผนภาพที่เสนอมาข้างต้น ซึ่งนอกจากจะมีหลักเกณฑ์ของภาษาหรือไวยากรณ์ของภาษาที่ชัดเจนแล้ว ในแต่ละส่วนก็ประกอบด้วยความหมายทั้งสิ้น โดยสิ่งที่แสดงทั้งโครงสร้างและความหมายของภาษานี้ จะเรียกว่าภาษาอธิบายสารสนเทศของยูเอ็มแอล (UML Matadata Language) โดยข้อมูลเหล่านี้จะถูกนำมาใช้เป็นตัวกำหนดมาตรฐานของภาษายูเอ็มแอลในแต่ละรุ่นที่รับรองโดยองค์กรโอเอ็มจี

แต่เนื่องจากภาษายูเอ็มแอลเป็นภาษาที่เป็นกราฟิก ทำให้การใช้งานเป็นไปด้วยความยุ่งยาก เนื่องจากการถ่ายเทข้อมูลมีข้อจำกัดและภาษากราฟิกไม่เหมาะสำหรับการทำงานด้วยโปรแกรมคอมพิวเตอร์ ดังนั้นโอเอ็มจีจึงได้พัฒนาสิ่งที่แทนภาษายูเอ็มแอลในรูปลักษณะอื่นแทน โดยสื่อที่นำมาใช้งานนั้น คือ ภาษาเอ็กซ์เอ็มแอล สาเหตุที่เลือกเอาภาษาเอ็กซ์เอ็มแอลมาเป็นภาษาที่แทนข้อมูลภาษายูเอ็มแอลนั้น เนื่องจากภาษาเอ็กซ์เอ็มแอลเป็นภาษาที่มีความยืดหยุ่น ทำให้สามารถรองรับความหลากหลายของภาษายูเอ็มแอลได้ อีกทั้งภาษาเอ็กเอ็มแอลยังเป็นภาษาที่มาตรฐานตาม W3C ทำให้การแลกเปลี่ยนข้อมูลเป็นไปได้ง่ายและถูกต้อง

ด้วยเหตุนี้ วิธีการที่จะพัฒนาขึ้นในงานวิจัยนี้ จึงใช้ภาษาเอ็กซ์เอ็มแอลเป็นสื่อกลาง โดยที่ภาษาเอ็กซ์แอลนั้นต้องมีความเป็นมาตรฐาน เพื่อให้การแลกเปลี่ยนข้อมูลมีความถูกต้องและเป็นที่ยอมรับ สามารถพัฒนาใช้งานร่วมกับเครื่องมือทางวิศวกรรมซอฟท์แวร์อื่นๆได้

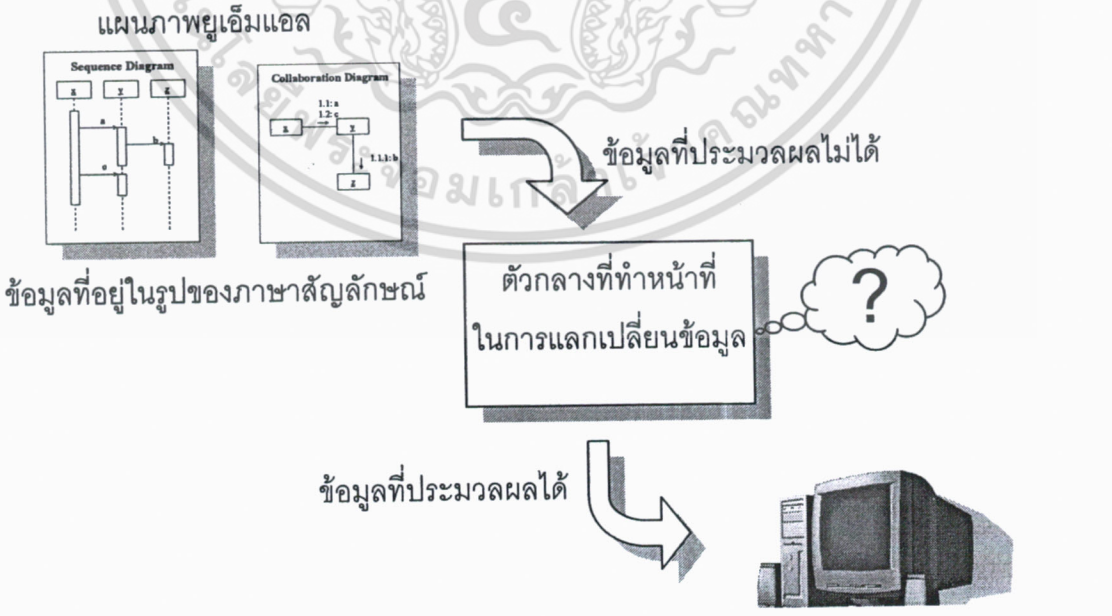
บทที่ 3

การแลกเปลี่ยนข้อมูล

3.1 การแลกเปลี่ยนข้อมูลภาษายูเอ็มแอลเข้าสู่มาตรฐานเอ็กซ์เอ็มไอ

โดยปกติแล้วภาษายูเอ็มแอลเป็นภาษาสัญลักษณ์ คือ อยู่ในลักษณะของข้อมูลรูปภาพ ซึ่งก็คือ แผนภาพที่ใช้แสดงมุมมองต่างๆของระบบในกระบวนการพัฒนาระบบซอฟต์แวร์ และจากการที่ภาษายูเอ็มแอลมีลักษณะเป็นข้อมูลแผนภาพนี้เอง การแปลงข้อมูลจากภาษาจาวาให้ไปเป็นภาษายูเอ็มแอลจึงไม่สามารถกระทำได้โดยตรง ในงานวิจัยจึงจำเป็นต้องมีวิธีการแลกเปลี่ยนข้อมูลจากภาษายูเอ็มแอล ที่อยู่ในรูปภาพสัญลักษณ์ เป็นข้อมูลที่สามารถทำการประมวลผลได้โดยโปรแกรมทางคอมพิวเตอร์ เพื่อให้โปรแกรมทางคอมพิวเตอร์ทำการแปลงข้อมูลที่เป็นโปรแกรมภาษาจาวาให้เข้าสู่แผนภาพในภาษายูเอ็มแอล

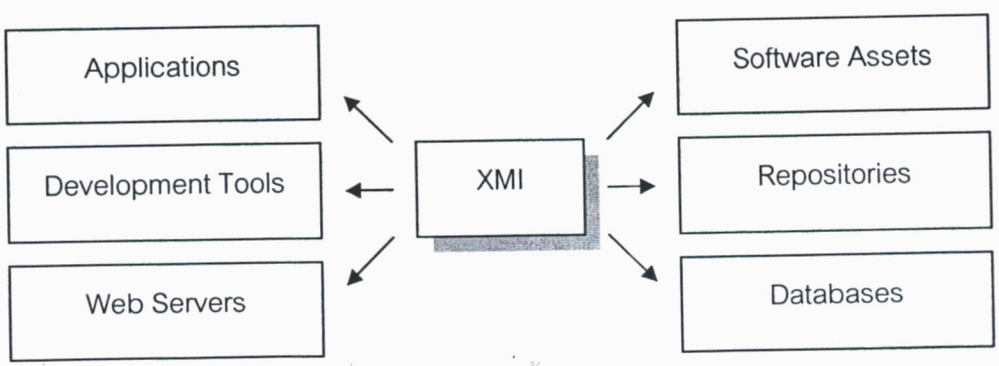
ภาพที่ 3.1
แนวคิดการแลกเปลี่ยนข้อมูลเพื่อให้สามารถประมวลผลได้โดยคอมพิวเตอร์



จากแนวคิดการแลกเปลี่ยนข้อมูลข้างต้น งานวิจัยนี้ได้ใช้วิธีการแลกเปลี่ยนข้อมูลจากแผนภาพยูเอ็มแอล ให้อยู่ในรูปของเอกสารเอ็กซ์เอ็มแอล เหตุผลของการเลือกใช้เอกสารเอ็กซ์เอ็มแอลแทนข้อมูลจากแผนภาพยูเอ็มแอล เนื่องจากว่าเอกสารเอ็กซ์เอ็มแอลสามารถใช้งานได้กับเครื่องมือประเภทเคสทูล (CASE - Computer Aided Software Engineering) ที่มีอยู่หลากหลายได้ในปัจจุบัน อีกทั้งเอกสารเอ็กซ์เอ็มแอล ยังเป็นเอกสารที่มีความเป็นมาตรฐาน ซึ่งมาตรฐานดังกล่าวกำหนดโดย W3C (World Wide Web Consortium) ซึ่งถือว่าเป็นมาตรฐานเปิด นอกจากนี้เอกสารเอ็กซ์เอ็มแอล ยังมีโครงสร้างที่มีประสิทธิภาพที่ง่ายต่อการทำความเข้าใจ เนื่องจากเอกสารเอ็กซ์เอ็มแอลเขียนอยู่ในรูปของภาษามนุษย์ ผู้ใช้งานสามารถอ่านและตีความได้ด้วยตนเอง และสาเหตุประการสำคัญอีกประการหนึ่งที่เลือกใช้เอกสารเอ็กซ์เอ็มแอล คือ ทางโอเอ็มจี (OMG - Object Management Group) ซึ่งเป็นผู้กำหนดมาตรฐานของภาษายูเอ็มแอล มีการกำหนดมาตรฐานการแลกเปลี่ยนข้อมูล (Standard Exchange Format) ของแผนภาพของภาษายูเอ็มแอลไปเป็นเอกสารเอ็กซ์เอ็มแอล โดยกลไกการแลกเปลี่ยนข้อมูลที่ชื่อว่า เอ็กซ์เอ็มไอ (XMI - XML Metadata Interchange) เนื่องจากประโยชน์ของการแลกเปลี่ยนข้อมูลในรูปแบบเว็ปเบสเอ็กซ์เอ็มแอล (web-based XML) เป็นผลดีต่อการพัฒนาภาษายูเอ็มแอลในอนาคต อีกทั้งแนวทางของโมเดลในปัจจุบันมีการใช้หลายแผนภาพในการให้มุมมองต่างๆ (Heterogeneous) แต่แผนภาพต้องมีความสัมพันธ์ (Homogenous) หรือมีความสอดคล้องกัน (ICSE2000 Workshop on Standard Exchange Format, Limerick Ireland) ซึ่งเหมาะกับคุณลักษณะของเอกสารเอ็กซ์เอ็มแอล ทำให้ในปัจจุบันได้มีหลายหน่วยงานวิจัยที่เกี่ยวข้องกับการพัฒนาระบบด้วยภาษายูเอ็มแอล ได้พัฒนาเครื่องมือการแลกเปลี่ยนข้อมูลนี้ขึ้น

ภาพที่ 3.2

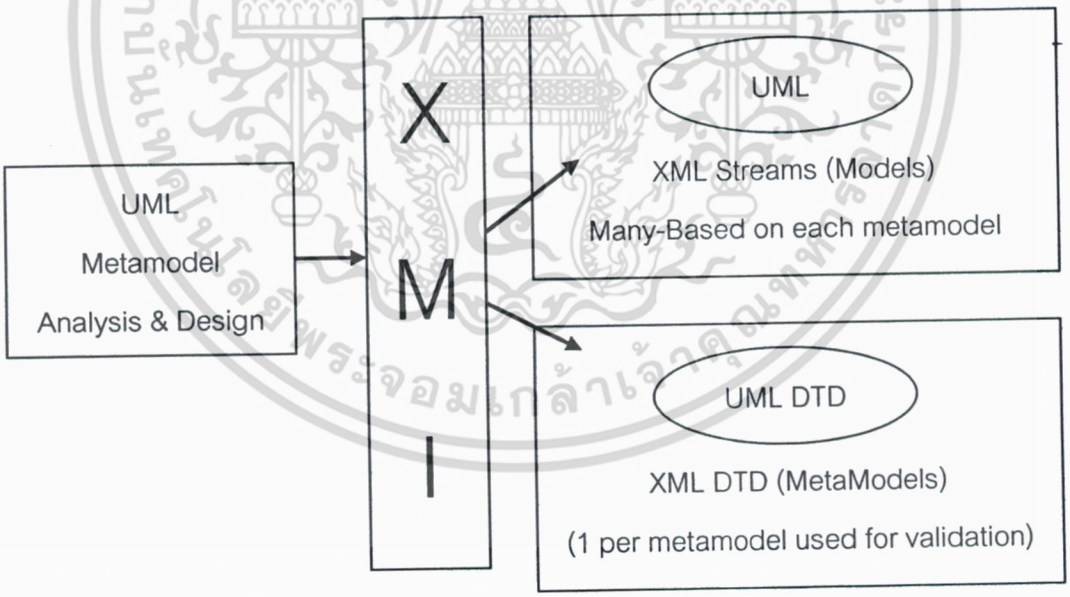
แสดงประโยชน์ของการแลกเปลี่ยนข้อมูลโดยใช้เอ็กซ์เอ็มไอ



สำหรับมาตรฐานเอ็กซ์เอ็มไอรุ่นที่ใช้ในงานวิจัยนี้คือ เอ็กซ์เอ็มไอ เวอร์ชั่น 1.1 (OMG standard 2nd February 2000) ซึ่งเป็นเวอร์ชั่นที่ใช้งานกันอยู่ในปัจจุบัน เอกสารเอ็กซ์เอ็มแอลที่ได้จากการแลกเปลี่ยนโดยมาตรฐานเอ็กซ์เอ็มไอ จะประกอบด้วย XML DTD (Document Type Declaration) ที่ชื่อ UML.DTD เป็นไฟล์กำหนดไวยากรณ์ในการอ่านเอกสารเอ็กซ์เอ็มแอลที่ได้จากการแปลงข้อมูลจากแผนภาพยูเอ็มแอล ซึ่ง UML.DTD นั้นเป็นส่วนประกอบหนึ่งของเอกสารเอ็กซ์เอ็มแอลโดย DTD จะมีลักษณะเหมือนฐานข้อมูล คือ มีการกำหนดชื่อ และเก็บคำจำกัดความของอิลิเมนท์ (Element) ต่างๆที่ใช้ในเอกสารเอ็กซ์เอ็มแอล รวมทั้งแสดงคำสั่งและกำหนดแอตทริบิวส์ (Attribute) ของอิลิเมนท์นั้นด้วย ดังนั้นในงานวิจัยนี้จึงจำเป็นจะต้องมี UML.DTD เพื่อใช้ในการกำหนดโครงสร้างของเอกสารเอ็กซ์เอ็มแอล ประกอบในการใช้งานด้วยทุกครั้ง

ภาพที่ 3.3

การแลกเปลี่ยนข้อมูลของภาษายูเอ็มแอล โดยใช้มาตรฐานการแปลงเอ็กซ์เอ็มไอ



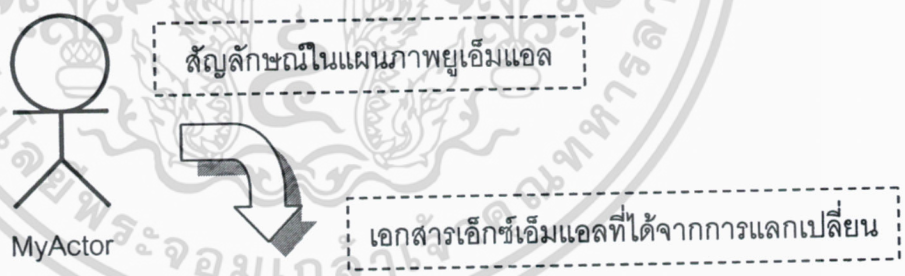
สำหรับเครื่องมือที่ใช้ในการแลกเปลี่ยนข้อมูลในงานวิจัยนี้ คือ โปรแกรม Unisys Rose XML Tool ซึ่งเป็นโปรแกรมเสริม (Plug-in) ในโปรแกรม Rational Rose โดยผู้ใช้สามารถดาวน์โหลดโปรแกรมดังกล่าวได้จากโฮมเพจของ Rational Software Corporation แล้วทำการ

แอดอิน (Add-in) ลงในโปรแกรม Rational Rose ได้ นอกจากการแลกเปลี่ยนข้อมูลจากแผนภาพยูเอ็มแอลเป็นเอกสารเอ็กซ์เอ็มแอลแล้ว เครื่องมือนี้ยังสามารถทำการแลกเปลี่ยนข้อมูลกลับจากเอกสารเอ็กซ์เอ็มแอลเป็นแผนภาพยูเอ็มแอล (Reverse Engineering) ได้อีกด้วย

ลักษณะของเอกสารเอ็กซ์เอ็มแอลที่ได้จากการแลกเปลี่ยนข้อมูลจากแผนภาพยูเอ็มแอลจะมีการกำหนดแท็ก (Tag) ที่สื่อความหมายที่มีความสัมพันธ์กับข้อมูลที่ถูกแลกเปลี่ยน ในตัวอย่างต่อจากนี้ คือ ตัวอย่างการแลกเปลี่ยนข้อมูลที่ใช้สัญลักษณ์รูปคนแทนการเรียกผู้ใช้ระบบหรือที่รู้จักกันในภาษายูเอ็มแอลว่า แอคเตอร์ (Actor) โดยสัญลักษณ์นี้มีชื่อว่า "MyActor" ซึ่งเอกสารเอ็กซ์เอ็มแอลที่ได้นั้น ประกอบด้วยแท็กต่างๆที่ให้ความหมายและข้อมูลที่เกี่ยวกับสัญลักษณ์นั้น เช่น ชื่อของสัญลักษณ์ถูกแทนด้วยแท็กชื่อ <Foundation.Core.ModelElement.name> และความสามารถในการเข้าถึงข้อมูลถูกแทนด้วยแท็กชื่อ <Foundation.Core.ModelElement.visibility xmi.value='public'/> เป็นต้น ซึ่งความหมายของแท็กต่างเหล่านี้ ผู้ใช้สามารถศึกษาได้จาก UML.DTD ได้ ดังที่กล่าวมาแล้วในข้างต้น

ภาพที่ 3.4

แสดงตัวอย่างเอ็กซ์เอ็มไอเทียบกับสัญลักษณ์ในแผนภาพยูเอ็มแอล



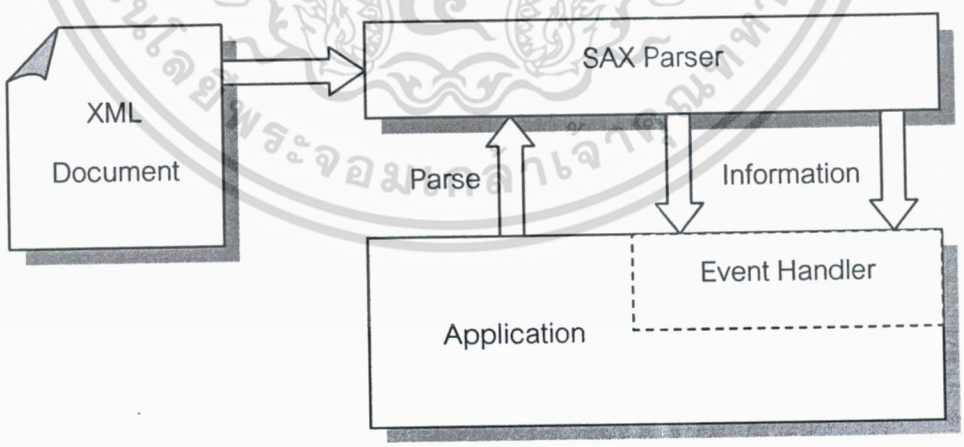

```
<Behavioral_Elements.Use_Cases.Actor xmi.idref = 'S.10003>
    <Foundation.Core.ModelElement.name>MyActor</Foundation.Core.ModelElement.
name>
    <Foundation.Core.ModelElement.visibility xmi.value='public'/>
    <Foundation.Core.GerneralizableElement.name.isRoot xmi.value='true'/>
    <Foundation.Core.GerneralizableElement.name.isLeaf xmi.value='true'/>
        ...
</Behavioral_Elements.Use_Cases.Actor xmi.idref = 'S.10003>
```

จากการแลกเปลี่ยนข้อมูลของภาษายูเอ็มแอลโดยใช้มาตรฐานเอ็กซ์เอ็มไอ ผลลัพธ์ที่ได้อยู่ในรูปของเอกสารเอ็กซ์เอ็มแอล ซึ่งสามารถทำการประมวลผลได้ด้วยโปรแกรมทางคอมพิวเตอร์ แต่การประมวลผลข้อมูลในเอกสารเอ็กซ์เอ็มแอลได้นั้น ผู้ใช้จำเป็นต้องรู้โครงสร้างในเอกสารเอ็กซ์เอ็มแอลเสียก่อน เพื่อให้สามารถเข้าถึงข้อมูลและดึงข้อมูลนำมาใช้จากเอกสารเอ็กซ์เอ็มแอลได้ วิธีการเข้าถึงข้อมูลในเอกสารเอ็กซ์เอ็มแอล ใช้ตัววิเคราะห์โครงสร้างของเอกสาร (Parser) ซึ่งปัจจุบันมีวิธีการที่เป็นที่นิยมใช้อยู่ 2 วิธี คือ SAX (Simple API for XML) และDOM (Document Object Model)

SAX (Simple API for XML) เป็นวิธีการเข้าถึงข้อมูลโดยใช้เหตุการณ์เป็นตัวกำหนด ส่วนใหญ่วิธีนี้มักใช้ในงานที่เกี่ยวข้องกับการรับ-ส่งข้อมูลเอกสารเอ็กซ์เอ็มแอลผ่านระบบเครือเนื่องจากวิธีการนี้ให้ความรวดเร็วและสิ้นเปลืองหน่วยความจำน้อย แต่ในทางโปรแกรมมิ่งมันนิยมใช้ เนื่องจากวิธีนี้เป็นวิธีที่ยากต่อการใช้งาน เนื่องจากการเข้าถึงข้อมูลเป็นไปตามลำดับเหตุการณ์ซึ่งไม่สอดคล้องกับวิธีการทางโปรแกรมในการเข้าถึงข้อมูล

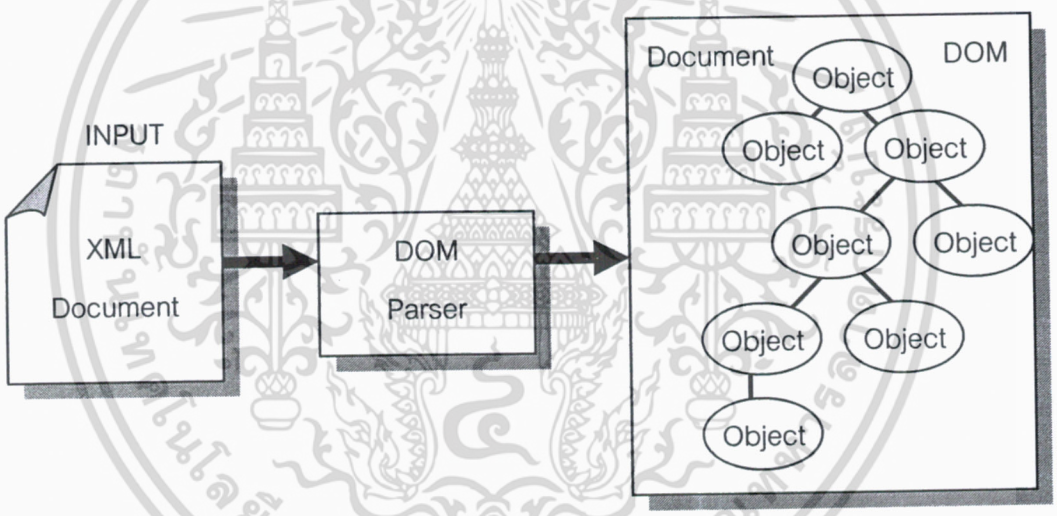
ภาพที่ 3.5

แสดงการพัฒนาแอพพลิเคชั่นโดยใช้ SAX Parser



DOM (Document Object Model) วิธีการเข้าถึงแบบ DOM ใช้วิธีการจัดโครงสร้างข้อมูลของเอกสารเอ็กซ์เอ็มแอล ในรูปแบบโครงสร้างลำดับชั้นเหมือนโครงสร้างต้นไม้ ซึ่งในโครงสร้างต้นไม้ ประกอบด้วยอิลิเมนท์โหนด (Element Node) โดยภายในแต่ละอิลิเมนท์โหนด ประกอบด้วยข้อมูลที่เกี่ยวข้องอิลิเมนท์นั้น ในมาตรฐานเอกซ์เอ็มไอ อิลิเมนท์โหนดถูกกำหนดด้วยแท็กโดยชื่อของแท็ก จะให้ความหมายที่อ้างอิงถึงองค์ประกอบในภาษายูเอ็มแอล ตามที่ถูกกำหนดไว้ในมาตรฐานเอ็กซ์เอ็มไอ

ภาพที่ 3.6
แสดงวิธีการเข้าถึงข้อมูลโดยวิธี DOM Parsing

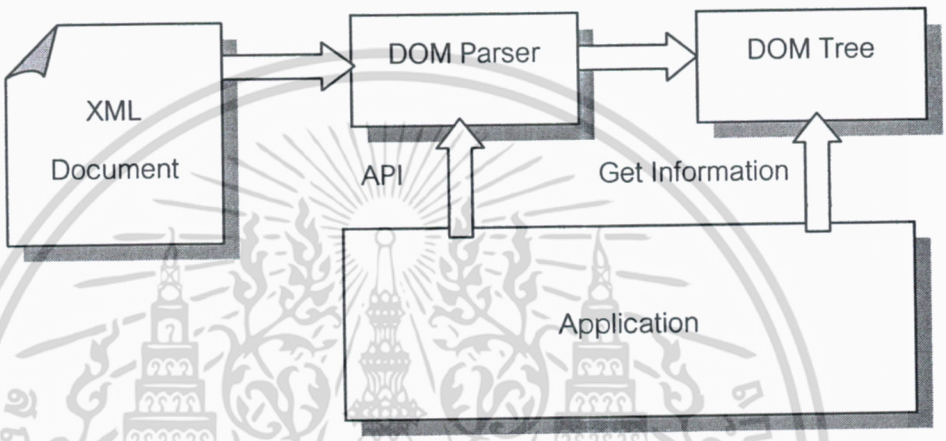


ในด้านการพัฒนาโปรแกรม ผู้พัฒนาโปรแกรมส่วนใหญ่จะใช้ การเข้าถึงเอกสารแบบ DOM มากกว่าการเข้าถึงแบบ SAX เพราะจากข้างต้นเห็นได้ว่าวิธีการแบบ DOM ทำให้เห็นว่าเอกสารเอ็กซ์เอ็มแอล เป็นเอกสารที่มีโครงสร้างเชิงวัตถุ โดยมี DOM ทำหน้าที่จัดการโครงสร้างข้อมูลต่างๆภายในเอกสารเอ็กซ์เอ็มแอล ทำให้ผู้ใช้งานสามารถเข้าถึงโครงสร้างข้อมูลได้แบบสุ่ม (Random Access) ซึ่งแตกต่างจากวิธีการเข้าถึงข้อมูลแบบ SAX ที่เป็นลำดับขั้นตอนของเหตุการณ์ นอกจากนี้ DOM ยังสนับสนุน API (Application Program Interface) ของโปรแกรมภาษาต่างๆที่ช่วยในการเชื่อมโยงระหว่าง DOM กับภาษาโปรแกรมภายนอกไว้ด้วย ในงานวิจัยนี้

จึงใช้การเข้าถึงข้อมูลในเอกสารเอ็กซ์เอ็มแอลแบบ DOM เป็นวิธีการในการเข้าถึงข้อมูลเอกสารเอ็กซ์เอ็มแอล ด้วยเหตุผลที่กล่าวมาแล้วข้างต้น

ภาพที่ 3.7

แสดงการพัฒนาแอพพลิเคชั่นโดยใช้ DOM Parser

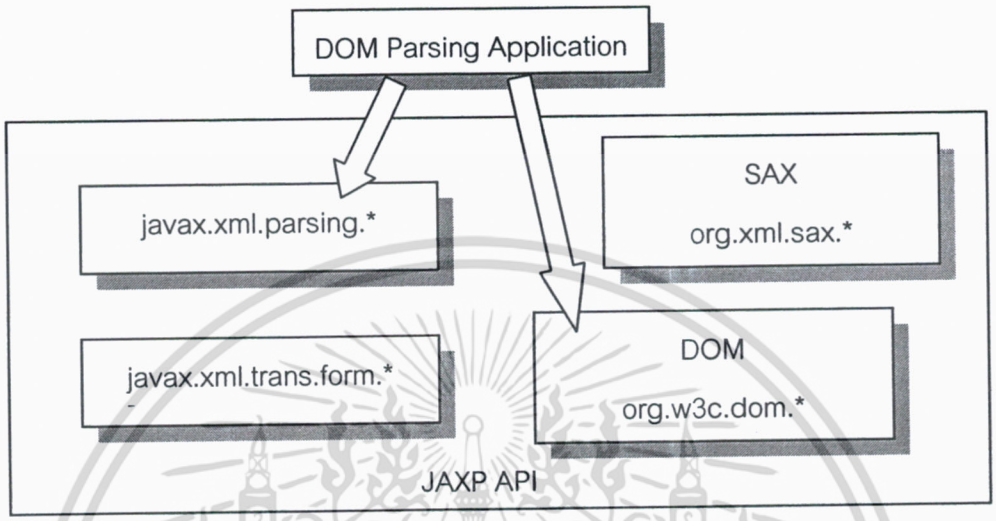


3.2 <u>การเข้าถึงข้อมูลในเอกสารเอ็กซ์เอ็มแอล</u>

วิธีการเข้าถึงข้อมูลแบบ DOM (Document Object Model) สนับสนุน API (Application Program Interface) ของโปรแกรมภาษาต่างๆ ทำให้สามารถเข้าถึงข้อมูลและดึงข้อมูลจากแผนภาพต้นไม้ที่แทนข้อมูลของแผนภาพยูเอ็มแอล มาทำการตรวจสอบได้ ในงานวิจัยนี้ได้ใช้โปรแกรมภาษาจาวา (Java™ 2 Platform, Standard Edition (J2SE) v 1.4.0) เป็นภาษาโปรแกรมในการพัฒนาเครื่องมือในการดึงข้อมูลจากเอกสารเอ็กซ์เอ็มแอล สาเหตุที่เลือกใช้โปรแกรมภาษาจาวา J2SE v.1.4.0 เนื่องจากโปรแกรมภาษาจาวา เป็นโปรแกรมภาษาที่สนับสนุนและรองรับเทคโนโลยีเอ็กซ์เอ็มแอล โดยมี JAXP (Java API for XML Processing 1.1) เป็น API ในการติดต่อและจัดการกับข้อมูลในเอกสารเอ็กซ์เอ็มแอล ผู้พัฒนาโปรแกรมสามารถเรียกใช้แพ็กเกจ JavaX ที่มากับตัวโปรแกรมภาษาในการดึงข้อมูลจากเอกสารXMLโดยใช้คำสั่งที่โปรแกรมกำหนดให้ได้เลย อีกทั้งภาษาจาวาสามารถทำการแลกเปลี่ยนข้อมูลระหว่างตัวโปรแกรมภาษาจาวากับโปรแกรมภาษาอื่นได้ง่าย ซึ่งเป็นผลดีต่อการพัฒนาโปรแกรมในส่วนถัดไป

ภาพที่ 3.8

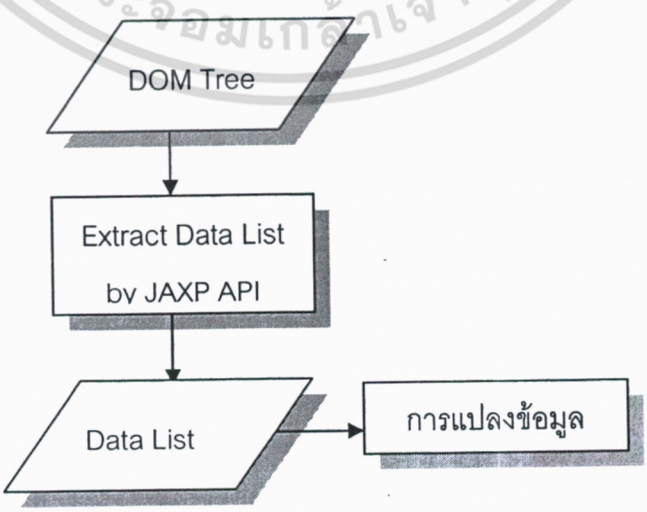
JAXP 1.1 คอมโพเนนท์



จากส่วนที่กล่าวมาข้างต้น ผลลัพธ์ที่ได้จากการพัฒนาโปรแกรมโดยโปรแกรมภาษาจาวา คือ ชุดของข้อมูล (Data Lists) ที่พร้อมรับการแปลงข้อมูล โดยการแปลงข้อมูลนั้นจะใช้วิธีการอ่านแท็กซ์ในเอกสาร XMI.DTD เพื่อทำการวิเคราะห์ข้อมูลอยู่ในรูปแบบของภาษายูเอ็มแอล

ภาพที่ 3.9

กระบวนการทำงานของส่วนในการเข้าถึงข้อมูล



ในการวิเคราะห์ข้อมูล ผู้วิจัยได้ทำการศึกษาองค์ประกอบพื้นฐานของภาษานิยามกฎเชิงวัตถุ เนื่องจากมาตรฐานของภาษาเชิงวัตถุจะเป็นสิ่งกำกับการจัดรูปแบบของการแลกเปลี่ยนข้อมูลโดยภาษานิยามกฎเชิงวัตถุนั้น ประกอบด้วยวัตถุ (Object) และคุณสมบัติของวัตถุ (Object Properties) ซึ่งชนิดของวัตถุสามารถแบ่งออกได้เป็นสองประเภทใหญ่ๆ คือ

ก. ชนิดของวัตถุที่ได้กำหนดไว้แล้ว (Pre-defined types) สามารถแบ่งออกได้เป็น

I. ชนิดพื้นฐาน (Basic Types) ได้แก่ จำนวนเต็ม (Integer), จำนวนจริง (Real), ชุดอักขระ (String) และ บูลีน (Boolean)

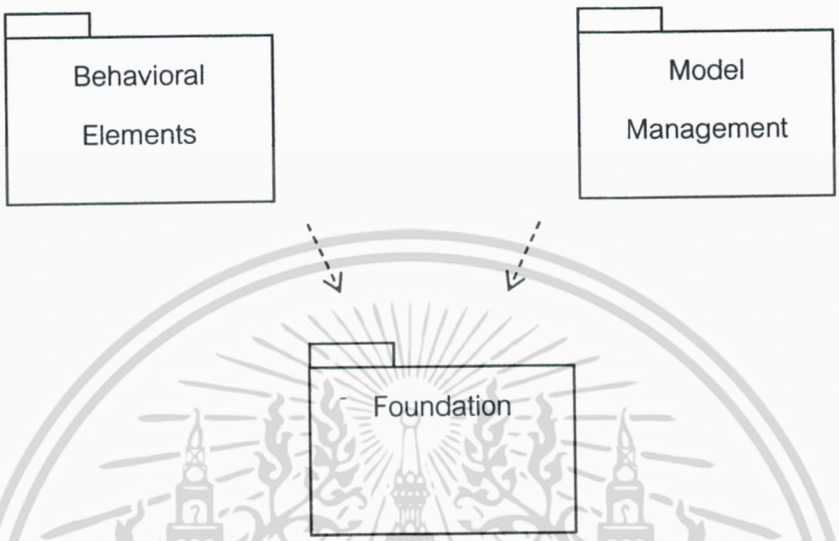
II.ชนิดชุดข้อมูล (Collection Types) ได้แก่ คอลเลคชั่น (Collection), เซต (Set), แบ็ค (Bag) และซีเควนซ์ (Sequence)

ข.ชนิดของวัตถุที่ผู้ใช้เป็นผู้กำหนดขึ้นเอง (User-defined types)

สิ่งเหล่านี้เป็นความรู้เบื้องต้นเกี่ยวกับภาษานิยามกฎเชิงวัตถุ ซึ่งผู้แปลงข้อมูลต้องตรวจสอบรูปแบบที่ถูกต้องของแผนภาพยูเอ็มแอล ที่อยู่ในรูปของภาษานิยามกฎเชิงวัตถุเป็นอย่างดีเสียก่อน นอกจากนี้สิ่งที่ต้องคำนึงถึงในการสร้างกฎการแปลงข้อมูล คือ กฎที่สร้างขึ้นต้องมีความชัดเจนและไม่ยากในการทำความเข้าใจ โดยทั่วไปแล้วกฎสำหรับการตรวจสอบรูปแบบที่ถูกต้องของแผนภาพยูเอ็มแอล ถูกจำแนกออกเป็นประเภทตามแต่ละองค์ประกอบในภาษายูเอ็มแอล ซึ่งการจำแนกกฎการตรวจสอบ ช่วยให้ผู้สร้างกฎสามารถทำความเข้าใจกฎการตรวจสอบได้ดียิ่งขึ้น การจำแนกทำตามโมเดลโครงสร้างของภาษายูเอ็มแอล (UML Metamodel) ในระดับบนสุด (Top-Level Package) จะประกอบด้วยแพ็กเกจ 3 แพ็กเกจหลัก คือ แพ็กเกจที่เป็นองค์ประกอบพื้นฐาน (Foundation Package), แพ็กเกจที่เก็บองค์ประกอบที่เกี่ยวข้องกับพฤติกรรม (Behavioral Elements Package) และ แพ็กเกจที่เก็บองค์ประกอบที่ใช้สำหรับจัดกลุ่ม (Model Management Package) ดังภาพที่ 3.10

ภาพที่ 3.10

ระดับบนสุดของโมเดลโครงสร้างภาษายูเอ็มแอล



โดยในแต่ละแพ็กเกจประกอบด้วยส่วนต่างๆที่จำแนกองค์ประกอบ เช่น ในแพ็กเกจที่เป็นองค์ประกอบพื้นฐานประกอบส่วนต่างๆดังนี้

ส่วนหลัก (Core)

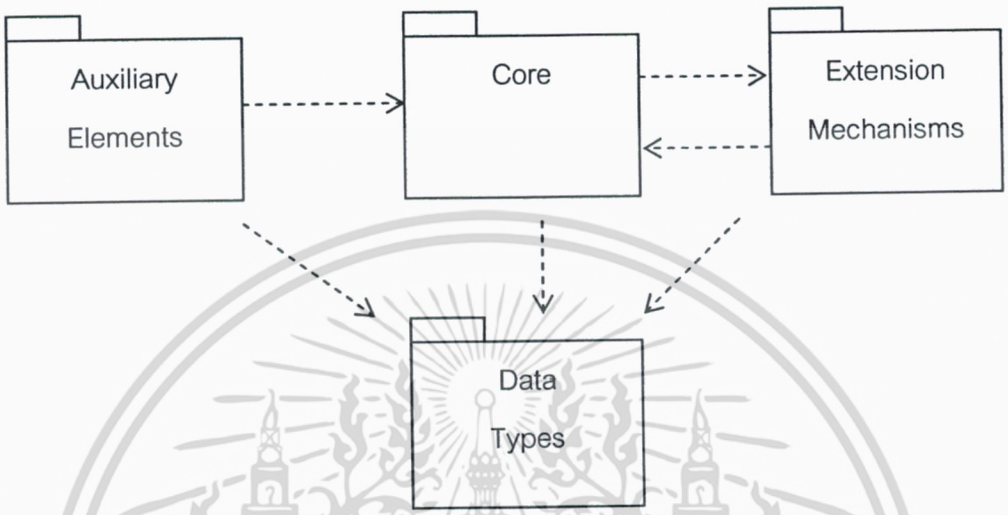
ส่วนเสริม (Auxiliary Elements)

ส่วนกลไกสำหรับขยาย (Extension Mechanisms)

ส่วนชนิดข้อมูล (Data Types)

ภาพที่ 3.11

แพ็กเกจที่เป็นองค์ประกอบพื้นฐาน (Foundation Packages)



ในส่วนหลัก เป็นส่วนที่แสดงรายละเอียดพื้นฐานของส่วนต่างๆ ซึ่งกฎสำหรับการตรวจสอบรูปแบบที่ถูกต้องของภาษายูเอ็มแอลถูกจำแนกตามรายละเอียดย่อยต่างๆ เช่น ความสัมพันธ์, คุณสมบัติ, คลาส และข้อจำกัด เป็นต้น

ในส่วนเสริม ถูกจำแนกตามรายละเอียดย่อยต่างๆ เช่น การเชื่อมโยง, คอมเมนท์, คอมโพเนนท์, การขึ้นต่อกัน, โหนด, และยูสเคจ เป็นต้น

ในส่วนกลไกสำหรับขยาย ถูกจำแนกตามรายละเอียดย่อยต่างๆ เช่น ข้อจำกัด, สเตอริโอไทป์, และค่าของแท็ก เป็นต้น

ในแพ็กเกจที่เก็บองค์ประกอบที่เกี่ยวข้องกับพฤติกรรม จะประกอบด้วย

คอลแลบอเรชั่น (Collaborations)

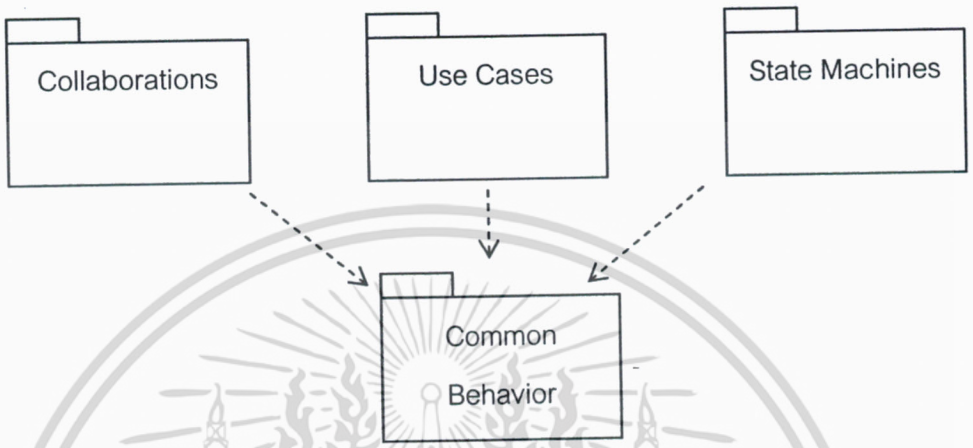
ยูสเคส (Use Cases)

สเตตแมชชีน (State Machines)

พฤติกรรมสามัญ (Common Behavior)

ภาพที่ 3.12

แพ็กเกจที่เก็บองค์ประกอบที่เกี่ยวข้องกับพฤติกรรม (Behavioral Elements Package)



ในส่วนของแพ็กเกจที่เก็บองค์ประกอบที่เกี่ยวข้องกับพฤติกรรมก็เช่นเดียวกัน ประกอบด้วยรายละเอียดที่เป็นส่วนที่จำแนกกฎของรูปแบบที่ถูกต้องของภาษายูเอ็มแอล

ในส่วนของพฤติกรรมสามัญถูกจำแนกตามรายละเอียดย่อยต่างๆ เช่น แอตทริบิวส์ , แอคชั่น, ลิงค์, ออบเจ็กต์ และซิกแนล เป็นต้น

ในส่วนของคอลแลบอเรชั่นถูกจำแนกตามรายละเอียดย่อยต่างๆ เช่น คอลแลบอเรชั่น, แอสโซซิเอชั่นโรล, คลาสซิไฟล์โรล, อินเตอร์แอ็คชั่น และ เมสเสจ เป็นต้น

ในส่วนของยูสเคสถูกจำแนกตามรายละเอียดย่อยต่างๆ เช่น แอคเตอร์, ยูสเคส และ ยูสเคสอินสแตนส์

ในส่วนของสเตตแมชชีนถูกจำแนกตามรายละเอียดย่อยต่างๆ เช่น คอมโพไซต์สเตต, สเตตแมชชีน และทรานซิชั่น เป็นต้น

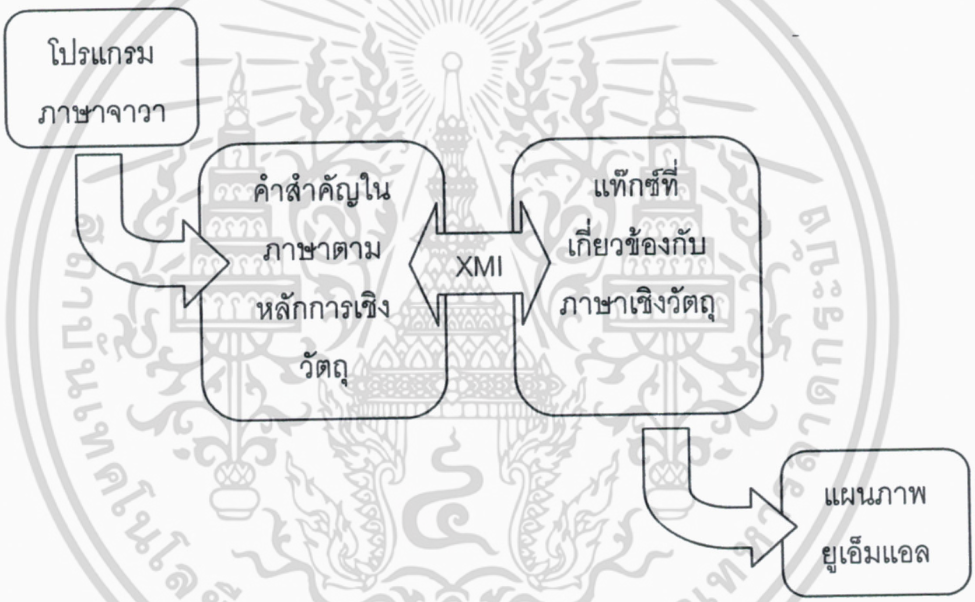
ในส่วนของแพ็กเกจที่เก็บองค์ประกอบที่ใช้สำหรับจัดกลุ่มจะประกอบด้วย แพ็กเกจ และ ระบบย่อย

บทที่ 4

สรุปผลการศึกษาวิจัยและข้อเสนอแนะ

4.1 สรุปผลการศึกษาวิจัย

ภาพที่ 4.1
แสดงขั้นตอนการแปลงภาษาจาวาเข้าสู่ภาษายูเอ็มแอล



จากงานวิจัยเห็นได้ว่า เครื่องมือที่ได้ทำการพัฒนาขึ้นจากแนวคิดของงานวิจัย สามารถทำการแปลงภาษาจาวาให้เข้าสู่แผนภาพยูเอ็มแอลได้ โดยใช้มาตรฐานเอ็กซ์เอ็มไอ

ซึ่งการแปลงนี้จะใช้เอกสารเอ็กซ์เอ็มแอลเป็นสื่อกลางสำหรับการแปลงข้อมูล โดยเอกสารเอ็กซ์เอ็มแอลดังกล่าว จะใช้ xmi.dtd เป็นตัวกลางในการแปลเอกสาร เนื่องจากข้อมูลเอ็กซ์เอ็มแอลจะถูกเก็บด้วยไวยากรณ์ตามมาตรฐานของเอ็กซ์เอ็มไอ ซึ่งเป็นมาตรฐานกลางในการแลกเปลี่ยนข้อมูล ทำให้งานวิจัยนี้สามารถตอบสนองจุดประสงค์ที่ได้กล่าวไว้ในขอบเขตของงานวิจัย คือ วิธีการที่สามารถแปลงภาษาโปรแกรมจาวาให้เข้าสู่แผนภาพยูเอ็มแอลโดยใช้มาตรฐานเอ็กซ์เอ็มไอได้

4.2 ข้อเสนอแนะ

จากงานวิจัยนี้ เอกสารเอ็กซ์เอ็มแอลที่อิงตามมาตรฐานเอ็กซ์เอ็มไอ จะใช้พื้นที่ในการเก็บค่อนข้างมาก เนื่องจากแท็กซ์ของเอ็กซ์เอ็มไอที่มีนั้น ต้องสร้างทุกแท็กซ์ที่นิยามเอกสารยูเอ็มแอลมีอยู่ ซึ่งบางแท็กซ์นั้นอาจไม่มีข้อมูลที่ใช้งานหรือเกี่ยวข้องเลย ทำให้เปลืองเนื้อที่เอกสาร อีกทั้งบางแท็กซ์ยังไม่ได้มีความจำเป็นเกี่ยวข้องกับการแปลงผลที่ได้ถูกนำไปใช้ เป็นเพียงแค่การอ้างอิงเท่านั้น แต่ก็ต้องทำการเก็บด้วย ข้อที่เสนอแนะที่ควรทำ คือ การลดรูปของเอกสารเอ็กซ์เอ็มแอลเพื่อให้การแปลงทำได้รวดเร็วขึ้น แต่การลดรูปนั้น ต้องไม่ให้เกิดการสูญเสียข้อมูลหรือสารสนเทศของเอกสารทั้งภาษาจาวาและแผนภาพยูเอ็มแอลไป

บรรณานุกรม

Electronic Resources

OMG Unified Modeling Language Specification, version 1.4, September 2001.

OMG XML Metadata Interchange (XMI) Specification, version 1.2, January 2002.

UML Semantics, version 1.1, September 1997.

Object Constraint Language Specification, version 1.1, September 1997.

JAXP Java API for XML Processing, version 1.1, Sun Microsystem, Inc., 6 February 2001.

Thanwadee, S., and Anthony, F. "Automated Consistency Checking for Multiperspective Software Specification", (1999).

Alexander, F. Egyed., et al. "UML/Analyzer Guide", 2000: 1-20.

Michael, Moors., Rational Software Corporation. "Rose Model Checker", 2000.

Glare, Gryee.; Anthony, Finkelstein.; and Christian, Nentwich. "Lightweight Checking for UML Based Software Development".

Ali, Hamie.; John, Howse.; and Stuart, Kent. "Interpreting the Object Constraint Language".

Xlinkit, "Xlinkit Consistency Checking Tool", <http://www.xlinkit.com>.

World Wide Web Consortium (W3C), <http://www.w3c.org>.

Unisys Rose XML Tool, <http://www.unisys.com>.

J2SE (Java™ 2 Platform Standard Edition, version 1.4.0) <http://java/sun.com>.

Amzi! Prolog + Logic Server, <http://www.amzi.com>.