

ระบบงานตั้งหนี้โครงการบัญชีผูกหนี้ของการเคหะแห่งชาติ

Project Debt Initial System of The National of Authority Thailand

โดย

นางสาวรวีวรรณ เหล่าตั้งจิตตรง

รหัส 41067026



H001616

อาจารย์ที่ปรึกษา

ผศ. บรรจง ปิยะธำรง

วัน เดือน ปี	22 S.A. 2549
เลขทะเบียน	01616
เลขเรียกหนังสือ	
"ห้องสมุดคณะเทคโนโลยีสารสนเทศ สอศ."	

รายงานนี้เป็นส่วนหนึ่งของวิชาโครงการพัฒนาระบบงาน

หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ

ภาคเรียนที่ 2 ปีการศึกษา 2542

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ชื่อหัวข้อ	ระบบงานดังนี้โครงการบัญชีลูกหนี้ของการเคหะแห่งชาติ
นักศึกษา	นางสาวรวีวรรณ เหล่าตั้งจิตตรง
อาจารย์ที่ปรึกษา	ผศ. บรรจง ปิยะธำรง
ระดับการศึกษา	วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
แขนงวิชา	วิทยาการสารสนเทศ
ปีการศึกษา	2542

บทคัดย่อ

ปัจจุบันมีการนำเทคโนโลยีสารสนเทศมาประยุกต์ใช้งานด้านต่าง ๆ มากมาย ไม่ว่าจะเป็นด้านธุรกิจ การศึกษา หรือทางด้านวิทยาศาสตร์ โครงการนี้เป็นส่วนหนึ่งของโครงการพัฒนาระบบงาน มีวัตถุประสงค์เพื่อพัฒนาซอฟต์แวร์ต้นแบบระบบดังนี้โครงการบัญชีลูกหนี้ของการเคหะแห่งชาติ และเพื่อเป็นข้อมูลสำหรับผู้บริหารในการตัดสินใจว่าระบบงานใดที่มีระดับความสำคัญสูงควรที่จะปรับปรุง หรือเปลี่ยนแปลงให้มีประสิทธิภาพมากขึ้น โครงการนี้มีการวิเคราะห์ และออกแบบระบบเชิงวัตถุ (Object-Oriented Analysis and Design) เนื่องจากมีเทคโนโลยีและเครื่องมือสนับสนุนมาก แต่ระบบสารสนเทศยังคงเป็นระบบฐานข้อมูลเชิงสัมพันธ์เนื่องจากมีความน่าเชื่อถือในการทำงานมากกว่า จึงต้องมีการปรับการวิเคราะห์ และออกแบบให้ใช้กับระบบฐานข้อมูลเชิงสัมพันธ์ด้วย ในการวิเคราะห์ และออกแบบระบบนี้ใช้ไคอะแกรมของยูเอ็มแอล (UML: Unified Modeling Language) โดยใช้ แอคติวิตีไคอะแกรมแสดงลำดับขั้นตอนกิจกรรมของระบบ ยูสเคสไคอะแกรม (Use-Case Diagram) แสดงผู้ติดต่อกับระบบและหน้าที่ของระบบ อินเทอร์แรกชันไคอะแกรม (Interaction Diagram) แสดงขั้นตอนการทำงานร่วมกันของออบเจกต์ในแต่ละยูสเคส ซึ่งจะช่วยในการวิเคราะห์และออกแบบคลาส คลาสไคอะแกรม (Class Diagram) แสดงโครงสร้างของคลาสในระบบเพื่อการออกแบบโครงสร้างฐานข้อมูล การออกแบบโมดูล และการออกแบบยูสเซอร์อินเตอร์เฟซ คอมโพเนนท์ไคอะแกรม (Component Diagram) แสดงองค์ประกอบทางกายภาพของโค้ดที่ประกอบขึ้นเป็นแอปพลิเคชันซอฟต์แวร์ และดีพลอยเมนต์ไคอะแกรม (Deployment Diagram) แสดงสถาปัตยกรรมทางกายภาพของฮาร์ดแวร์ และซอฟต์แวร์ในระบบ

Title	Project Debt Initial System of The National of Authority Thailand
Student	Miss Raweewan Lowtungjitttrong
Advisor	Asst. Prof. Bunjong Piyatamrourng
Level of Study	Master of Science in Information Technology
Major	Information Science
Academic Year	1999

ABSTRACT

At the present, the information technology is used to apply in Business Education and Science. This project is a part of the System Development Project. The purpose is to develop a prototype software of the Project Debt Initial System of The National of Authority Thailand for being used by manager in decision making with system is important and should by improved or modified to gain more efficiency. This project presents the Object-Oriented Analysis and Design because more technologies and tools to support. But this information system is the relational system because it's work. The need to adjust Object-Oriented Analysis and Design for using the relational system. This analysis and design system uses UML (Unified Modeling Language) diagram. An activity diagram shows a sequential flow of activity of system. A use-case diagram shows a number of external actors and functions of system. An interaction diagram shows collaboration between a number of objects for each use case for class analysis and design. The class diagram shows the static structure of class in the system for data structure, module and user interfaces design. A component diagram shows physical structure of code components of application software. And a deployment diagram shows the physical architecture of hardware and software in the system.

กิติกรรมประกาศ

โครงการพัฒนาระบบดังนี้โครงการบัญชีลูกหนี้ของการเคหะแห่งชาติ ได้รับการสนับสนุนจากหลายฝ่ายเป็นอย่างดี ที่ให้คำปรึกษาแนะนำ เพื่อให้งานได้สำเร็จลุล่วงตามเป้าหมายที่วางไว้ ผู้จัดทำจึงใคร่ขอขอบพระคุณบุคคลดังนี้

ขอขอบพระคุณบิดามารดา ที่ให้ความรัก และให้โอกาสทางการศึกษา

ขอขอบพระคุณ ผศ. บรรจง ปิยะธำรง ผู้เป็นอาจารย์ที่ปรึกษาโครงการ ให้คำชี้แนะต่าง ๆ และเอื้อเฟื้อกระดาษ และเครื่องพิมพ์

ขอขอบพระคุณผู้เป็นเจ้าของระบบจากกองบัญชีลูกหนี้ของการเคหะแห่งชาติทุกท่าน ที่ให้ความร่วมมือ ให้คำแนะนำเกี่ยวกับระบบงานต่าง ๆ เป็นอย่างดี

ขอขอบคุณเพื่อน ๆ ร่วมอาจารย์ที่ปรึกษา และร่วมระบบงานบัญชีลูกหนี้ของการเคหะแห่งชาติทุกท่าน ที่ให้ความร่วมมือ ให้คำปรึกษาเกี่ยวกับข้อมูล และระบบงานที่เกี่ยวข้องกัน รวมไปถึงการแลกเปลี่ยนความคิดเห็นเกี่ยวกับเทคโนโลยีที่ใช้ ทั้งการวิเคราะห์ ออกแบบ และการอิมพลีเมนต์

ขอขอบใจ นางสาวเบญจลักษณ์ สอนจังหวัด ผู้เป็นหลานสาวของผู้จัดทำ ที่สละเวลาส่วนหนึ่งของการพักผ่อนหลังเลิกงาน และวันหยุดที่แสนจะมีค่า ช่วยตัดแต่งภาพประกอบรายงาน และภาวนาให้โครงการชิ้นนี้ของผู้จัดทำสำเร็จลุล่วงทันกำหนด

ขอขอบใจเพื่อน ๆ ร่วมรุ่น IS 5 ที่ให้สติ และกำลังใจในการทำโครงการชิ้นนี้

รวีวรรณ เหล่าตั้งจิตตรง

สารบัญ

หน้า

บทคัดย่อภาษาไทย	I
บทคัดย่อภาษาอังกฤษ	II
กิตติกรรมประกาศ.....	III
สารบัญ	IV
สารบัญตาราง	VII
สารบัญภาพ	IX
บทที่	
1. บทนำ.....	1
1.1. ความเป็นมาของปัญหา.....	1
1.2. วัตถุประสงค์.....	2
1.3. เป้าหมาย.....	2
1.4. ขอบเขตของระบบ.....	2
1.5. ขั้นตอนและวิธีดำเนินการพัฒนาระบบ.....	3
1.6. รายละเอียดของแต่ละบท.....	4
2. ทฤษฎีที่เกี่ยวข้อง	5
2.1. วงจรการพัฒนาซอฟต์แวร์เชิงวัตถุ.....	5
2.2. การพัฒนาซอฟต์แวร์โดยการแบ่งชั้น	6
2.3. ยูเอ็มแอล.....	7
2.4. การวิเคราะห์เชิงวัตถุ.....	17
2.5. การออกแบบเชิงวัตถุ	17
3. การศึกษาความเป็นไปได้และการวิเคราะห์ระบบ.....	18
3.1. ภาพรวมระบบ (แอกติวิตี้ไดอะแกรม).....	18
3.2. ยูสเคสไดอะแกรม.....	19
3.3. คลาสไดอะแกรมจากการวิเคราะห์	22
3.4. อินเตอร์เฟซชั้นไดอะแกรม	24

บทที่

4. การออกแบบระบบ	44
4.1. คลาสในบิสซิเนสเลเยอร์	44
4.2. คลาสในแอคเซสเลเยอร์	56
4.3. คลาสในวิวเลเยอร์	57
5. การออกแบบฐานข้อมูลเชิงสัมพันธ์ของระบบ	62
5.1. การสร้างคลาสไดอะแกรมสำหรับแปลง	62
5.2. คำสั่งนิยามฐานข้อมูล	67
5.3. แผนภาพโครงสร้างฐานข้อมูลเชิงสัมพันธ์	70
5.4. อธิบายโครงสร้างฐานข้อมูลรวม	71
6. การออกแบบยูสเซอร์อินเตอร์เฟซ	74
6.1. การออกแบบเมนู	74
6.2. การออกแบบฟอร์มยูสเซอร์อินเตอร์เฟซ	77
6.3. การออกแบบรายงาน	86
7. การสร้างระบบ	90
7.1. การสร้างสตอโปรซีเจอร์	91
7.2. คอมโพเนนท์ไดอะแกรม	180
7.3. ดีพลอยเมนต์ไดอะแกรม	180
8. สรุปโครงการพัฒนาซอฟต์แวร์	182
8.1. ผลการพัฒนาซอฟต์แวร์	182
8.2. ปัญหาและอุปสรรคในการดำเนินงาน	183
8.3. ประโยชน์ที่ได้รับ	184
8.4. ข้อเสนอแนะ	184

สารบัญ (ต่อ)

หน้า

บรรณานุกรมสารบัญ..... 186

ประวัติผู้เขียน 187



สารบัญตาราง

หน้า

ตารางที่

3.1	สรุปความสัมพันธ์องค์ประกอบภายในยูสเคสไดอะแกรม.....	20
3.2	สรุปความสัมพันธ์ระหว่างคลาสในระบบ.....	23
4.1	แอทริบิวต์ของคลาส Location.....	46
4.2	การดำเนินการของคลาส Location.....	46
4.3	แอทริบิวต์สำหรับคลาส Project.....	47
4.4	การดำเนินการของคลาส Project.....	47
4.5	แอทริบิวต์สำหรับคลาส Model.....	48
4.6	การดำเนินการของคลาส Model.....	48
4.7	แอทริบิวต์สำหรับคลาส Building Type.....	49
4.8	การดำเนินการของคลาส Building Type.....	49
4.9	แอทริบิวต์สำหรับคลาส ComponentItem.....	49
4.10	การดำเนินการของคลาส ComponentItem.....	50
4.11	แอทริบิวต์สำหรับคลาส Component.....	50
4.12	การดำเนินการของคลาส Component.....	50
4.13	แอทริบิวต์สำหรับคลาส Quotation.....	51
4.14	การดำเนินการของคลาส Quotation.....	51
4.15	แอทริบิวต์สำหรับคลาส DownPayment.....	52
4.16	การดำเนินการของคลาส DownPayment.....	53
4.17	แอทริบิวต์สำหรับคลาส NewProjectHouseNo.....	53
4.18	การดำเนินการของคลาส NewProjectHouseNo.....	54
4.19	แอทริบิวต์สำหรับคลาส Unit.....	55
4.20	การดำเนินการของคลาส Unit.....	55
4.21	คำอธิบายคลาสในวิวเลเยอร์.....	59
5.1	ฐานข้อมูล Location หรือสำนักงานเคหะชุมชน.....	71

สารบัญตาราง (ต่อ)

หน้า

ตารางที่

5.2	ฐานข้อมูล Project หรือ โครงการ.....	71
5.3	ฐานข้อมูล Model หรือ โมเดล.....	71
5.4	ฐานข้อมูล Building Type หรือประเภทสิ่งปลูกสร้าง.....	71
5.5	ฐานข้อมูล ComponentItem หรือองค์ประกอบในโมเดล.....	72
5.6	ฐานข้อมูล Component หรือองค์ประกอบ.....	72
5.7	ฐานข้อมูล Quotation หรือการขายโครงการ.....	72
5.8	ฐานข้อมูล DownPayment หรือการผ่อนชำระเงินล่วงหน้า.....	72
5.9	ฐานข้อมูล NewProjectHouseNo หรือบ้านเลขที่โครงการ.....	73
5.10	ฐานข้อมูล Unit หรือหน่วยอาศัย.....	73
6.1	คำอธิบายเมนูย่อยของเมนูจัดการข้อมูล.....	75
6.2	คำอธิบายเมนูย่อยของเมนูรายงาน.....	76
7.1	คำอธิบายโมดูลในแฟ้มเก็บ Location.....	92
7.2	คำอธิบายโมดูลในแฟ้มเก็บ Project _Actions.....	94
7.3	คำอธิบายโมดูลในแฟ้มเก็บ Model _Actions.....	98
7.4	คำอธิบายโมดูลในแฟ้มเก็บ Building Type _Actions.....	99
7.5	คำอธิบายโมดูลในแฟ้มเก็บ ComponentItem _Actions.....	103
7.6	คำอธิบายโมดูลในแฟ้มเก็บ Component _Actions.....	104
7.7	คำอธิบายโมดูลในแฟ้มเก็บ Quotation _Actions.....	108
7.8	คำอธิบายโมดูลในแฟ้มเก็บ DownPayment _Actions.....	109
7.9	คำอธิบายโมดูลในแฟ้มเก็บ NewProjectHouseNo _Actions.....	112
7.10	คำอธิบายโมดูลในแฟ้มเก็บ Unit _Actions.....	113

สารบัญภาพ

	หน้า
รูปที่	
2.1 ตัวอย่าง Use-case diagram.....	8
2.2 ตัวอย่างคลาส Title.....	9
2.3 ตัวอย่างความสัมพันธ์แบบ association.....	9
2.4 ตัวอย่างความสัมพันธ์แบบ association ที่สัมพันธ์กันด้วยแอทริบิวต์.....	10
2.5 ตัวอย่างความสัมพันธ์แบบ aggregation ที่ประกอบด้วยนักกีฬา.....	10
2.6 ตัวอย่างความสัมพันธ์แบบ composition.....	10
2.7 ตัวอย่างความสัมพันธ์แบบ generalization ยานพาหนะแบ่งได้หลายชนิด.....	11
2.8 ตัวอย่างคลาสไลโคแกรม.....	11
2.9 ตัวอย่างตัวอย่างออบเจกต์ไลโคแกรม.....	11
2.10 ตัวอย่างชีเคว้นไลโคแกรม.....	12
2.11 ตัวอย่างคอลเลกชันบอแรชันไลโคแกรม.....	12
2.12 ตัวอย่างสเตทชาร์ทไลโคแกรม.....	13
2.13 ตัวอย่างแอคทิวิตีไลโคแกรม.....	14
2.14 ตัวอย่าง คอม โพนেন্টไลโคแกรม.....	15
2.15 ตัวอย่าง คีพลอยเมนต์ไลโคแกรม.....	15
2.16 ตัวอย่างแพ็คเกจซึ่งแสดงองค์ประกอบภายใน.....	16
2.17 ตัวอย่างแพ็คเกจที่มีความสัมพันธ์แบบขึ้นแก่กัน.....	16
3.1 แอคทิวิตีไลโคแกรมของภาพรวมระบบ.....	18
3.2 ยูสเคสไลโคแกรมระบบ.....	19
3.3 คลาสไลโคแกรมที่ได้จากการวิเคราะห์.....	22
3.4 ชีเคว้นไลโคแกรม Add project's quotation.....	28
3.5 ชีเคว้นไลโคแกรม Add down payment.....	29
3.6 ชีเคว้นไลโคแกรม Add new project's house numbers.....	30
3.7 ชีเคว้นไลโคแกรม Initial debtor code.....	31

สารบัญภาพ (ต่อ)

หน้า

รูปที่

3.8	ชีเคว้นไคอะแกรม Add new model.....	32
3.9	ชีเคว้นไคอะแกรม Add new component.....	33
3.10	ชีเคว้นไคอะแกรม Define component item for model.....	34
3.11	ชีเคว้นไคอะแกรม Add new building type.....	34
3.12	ชีเคว้นไคอะแกรม Add new project.....	35
3.13	ชีเคว้นไคอะแกรม Add Location information.....	36
3.14	ชีเคว้นไคอะแกรม Create project's quotations report.....	37
3.15	ชีเคว้นไคอะแกรม Create building's remainder record report.....	37
3.16	ชีเคว้นไคอะแกรม Create debtor code list report.....	38
3.17	คอลเลบบอเลชั่นไคอะแกรม Add project's quotation.....	38
3.18	คอลเลบบอเลชั่นไคอะแกรม Add down payment.....	39
3.19	คอลเลบบอเลชั่นไคอะแกรม Add new project's house numbers.....	39
3.20	คอลเลบบอเลชั่นไคอะแกรม Initial debtor code.....	40
3.21	คอลเลบบอเลชั่นไคอะแกรม Add new model.....	40
3.22	คอลเลบบอเลชั่นไคอะแกรม Add new component.....	41
3.23	คอลเลบบอเลชั่นไคอะแกรม Define component item for model.....	41
3.24	คอลเลบบอเลชั่นไคอะแกรม Add new building type.....	42
3.25	คอลเลบบอเลชั่นไคอะแกรม Add new project.....	42
3.26	คอลเลบบอเลชั่นไคอะแกรม Add Location information.....	42
3.27	คอลเลบบอเลชั่นไคอะแกรม Create project's quotations report.....	43
3.28	คอลเลบบอเลชั่นไคอะแกรม Create building's remainder record report.....	43
3.29	คอลเลบบอเลชั่นไคอะแกรม Create debtor code list report.....	43
4.1	แพ็กเกจสำหรับเลขอร์ในระบบตั้งหนี้โครงการบัญชีลูกหนี้ของการเคหะแห่งชาติ	44
4.2	คลาสไคอะแกรมสำหรับบิสซิเนสเลเยอร์ของระบบ.....	45

สารบัญภาพ (ต่อ)

รูปที่	หน้า
4.3 คลาสไดอะแกรมแสดงความสัมพันธ์ระหว่างคลาสในบิสซิเนสเลเยอร์.....	57
4.4 คลาสไดอะแกรมแสดงความสัมพันธ์ระหว่างคลาสในวิวเลเยอร์.....	58
4.5 คลาสไดอะแกรมแสดงความสัมพันธ์ระหว่างคลาสในวิวเลเยอร์กับบิสซิเนสเลเยอร์.....	61
5.1 คลาสรีเลย์ชั้นโครงการ กับสำนักงานเคหะชุมชน.....	63
5.2 คลาสรีเลย์ชั้นโมเดล กับชนิดสิ่งปลูกสร้าง.....	63
5.3 คลาสรีเลย์ชั้นหน่วยอาศัย กับข้อมูลขายโครงการ.....	64
5.4 คลาสรีเลย์ชั้นโมเดล รายการองค์ประกอบในโมเดล และองค์ประกอบ.....	64
5.5 คลาสรีเลย์ชั้นข้อมูลขายโครงการ โครงการ และโมเดล.....	65
5.6 คลาสรีเลย์ชั้นข้อมูลขายโครงการ กับบ้านเลขที่สำหรับหน่วยอาศัยในโครงการ.....	65
5.7 คลาสรีเลย์ชั้นข้อมูลขายโครงการ กับการผ่อนชำระล่วงหน้า.....	66
5.8 คำตั้งนิยามฐานข้อมูลที่ได้ในไฟล์ schema.SQL.....	67
5.9 แผนภาพโครงสร้างฐานข้อมูลเชิงสัมพันธ์.....	70
6.1 เมนูหลัก ซึ่งอยู่บนฟอร์มหลัก.....	74
6.2 เมนูย่อยสำหรับการจัดการข้อมูลในฐานข้อมูล.....	75
6.3 เมนูย่อยสำหรับการออกรายงาน.....	75
6.4 เมนูย่อยสำหรับการจัดการข้อมูลในฐานข้อมูล.....	76
6.5 เมนูย่อยสำหรับการจัดการหน้าต่างที่กำลังเปิดอยู่.....	76
6.6 ตัวอย่างแถบแสดงสถานะ.....	77
6.7 รายการค่าข้อมูล.....	78
6.8 ไดอะล็อกแสดงข้อความเตือน.....	78
6.9 ไดอะล็อกการเข้าสู่ระบบ.....	79
6.10 ฟอร์มจัดการข้อมูลขายโครงการ.....	79
6.11 ฟอร์มจัดการข้อมูลเตรียมบ้านเลขที่.....	80

สารบัญภาพ (ต่อ)

หน้า

รูปที่

6.12	ฟอร์มจัดการข้อมูลโครงการ.....	81
6.13	ฟอร์มจัดการข้อมูลโมเดล.....	81
6.14	ฟอร์มจัดการข้อมูลสำนักงานเคหะชุมชน.....	82
6.15	ฟอร์มจัดการข้อมูลประเภทสิ่งปลูกสร้าง.....	82
6.16	ฟอร์มจัดการข้อมูลองค์ประกอบ.....	83
6.17	ฟอร์มจัดการข้อมูลองค์ประกอบในโมเดล.....	83
6.18	ฟอร์มจัดการข้อมูลการผ่อนชำระล่วงหน้า.....	84
6.19	ฟอร์มออกรายงานข้อมูลขายโครงการ.....	85
6.20	ฟอร์มออกรายงานทะเบียนการตั้งหนี้ หรือรหัสลูกหนี้โครงการ.....	85
6.21	ฟอร์มออกรายงานทะเบียนอาคารคงเหลือที่ไม่มีการตั้งหนี้.....	86
6.22	ตัวอย่างรายงานข้อมูลขายโครงการ.....	87
6.23	ตัวอย่างรายงานทะเบียนการตั้งหนี้ หรือรหัสลูกหนี้โครงการ.....	88
6.24	ตัวอย่างรายงานทะเบียนอาคารคงเหลือที่ไม่มีการตั้งหนี้.....	89
7.1	โค้ดต้นฉบับส่วน Specification ของแฟ้มเก็บ Location_Actions.....	92
7.2	โค้ดต้นฉบับส่วน Body ของแฟ้มเก็บ Location_Actions.....	94
7.3	โค้ดต้นฉบับส่วน Specification ของแฟ้มเก็บ Project_Actions.....	98
7.4	โค้ดต้นฉบับส่วน Body ของแฟ้มเก็บ Project_Actions.....	99
7.5	โค้ดต้นฉบับส่วน Specification ของแฟ้มเก็บ Model_Actions.....	103
7.6	โค้ดต้นฉบับส่วน Body ของแฟ้มเก็บ Model_Actions.....	104
7.7	โค้ดต้นฉบับส่วน Specification ของแฟ้มเก็บ BuildingType_Actions.....	108
7.8	โค้ดต้นฉบับส่วน Body ของแฟ้มเก็บ BuildingType_Actions.....	109
7.9	โค้ดต้นฉบับส่วน Specification ของแฟ้มเก็บ ComponentItem_Actions.....	112
7.10	โค้ดต้นฉบับส่วน Body ของแฟ้มเก็บ ComponentItem_Actions.....	113
7.11	โค้ดต้นฉบับส่วน Specification ของแฟ้มเก็บ Component_Actions.....	116

สารบัญภาพ (ต่อ)

รูปที่		หน้า
7.12	โค้ดต้นฉบับส่วน Body ของแพ็คเกจ Component_Actions.....	117
7.13	โค้ดต้นฉบับส่วน Specification ของแพ็คเกจ Quotation_Actions.....	120
7.14	โค้ดต้นฉบับส่วน Body ของแพ็คเกจ Quotation_Actions.....	122
7.15	โค้ดต้นฉบับส่วน Specification ของแพ็คเกจ DownPayment_Actions.....	127
7.16	โค้ดต้นฉบับส่วน Body ของแพ็คเกจ DownPayment_Actions.....	128
7.17	โค้ดต้นฉบับส่วน Specification ของแพ็คเกจ NewProjectHouseNo_Actions.....	133
7.18	โค้ดต้นฉบับส่วน Body ของแพ็คเกจ NewProjectHouseNo_Actions.....	135
7.19	โค้ดต้นฉบับส่วน Specification ของแพ็คเกจ Unit_Actions.....	142
7.20	โค้ดต้นฉบับส่วน Body ของแพ็คเกจ Unit_Actions.....	144
7.21	แอกติวิตี้ไคอะแกรมของโมดูล Location_Actions.isNewLocation.....	150
7.22	แอกติวิตี้ไคอะแกรมของโมดูล Location_Actions.getLocationName.....	150
7.23	แอกติวิตี้ไคอะแกรมของโมดูล Location_Actions.retrieveLocation.....	151
7.24	แอกติวิตี้ไคอะแกรมของโมดูล Location_Actions.insertLocation.....	151
7.25	แอกติวิตี้ไคอะแกรมของโมดูล Location_Actions.updateLocation.....	151
7.26	แอกติวิตี้ไคอะแกรมของโมดูล Location_Actions.deleteLocation.....	152
7.27	แอกติวิตี้ไคอะแกรมของโมดูล Project_Actions.isNewProject.....	152
7.28	แอกติวิตี้ไคอะแกรมของโมดูล Project_Actions.getProjectName.....	152
7.29	แอกติวิตี้ไคอะแกรมของโมดูล Project_Actions.getlocationID.....	153
7.30	แอกติวิตี้ไคอะแกรมของโมดูล Project_Actions.retrieveProject.....	153
7.31	แอกติวิตี้ไคอะแกรมของโมดูล Project_Actions.insertProject.....	153
7.32	แอกติวิตี้ไคอะแกรมของโมดูล Project_Actions.updateProject.....	154
7.33	แอกติวิตี้ไคอะแกรมของโมดูล Project_Actions.deleteProject.....	154
7.34	แอกติวิตี้ไคอะแกรมของโมดูล Model_Actions.isNewModel.....	154
7.35	แอกติวิตี้ไคอะแกรมของโมดูล Model_Actions.getModelName.....	155

สารบัญภาพ (ต่อ)

	หน้า
รูปที่	
7.36 แอคตีวิตี้ไคอะแกรมของโมดูล Model_Actions.getBuildingTypeID.....	155
7.37 แอคตีวิตี้ไคอะแกรมของโมดูล Model_Actions.retrieveModel.....	156
7.38 แอคตีวิตี้ไคอะแกรมของโมดูล Model_Actions.insertModel.....	156
7.39 แอคตีวิตี้ไคอะแกรมของโมดูล Model_Actions.updateModel.....	156
7.40 แอคตีวิตี้ไคอะแกรมของโมดูล Model_Actions.deleteModel.....	157
7.41 แอคตีวิตี้ไคอะแกรมของโมดูล BuildingType_Actions.isNewBuildingType.....	157
7.42 แอคตีวิตี้ไคอะแกรมของโมดูล BuildingType_Actions.getBuildingTypeName...	158
7.43 แอคตีวิตี้ไคอะแกรมของโมดูล BuildingType_Actions.retrieveBuildingType.....	158
7.44 แอคตีวิตี้ไคอะแกรมของโมดูล BuildingType_Actions.insertBuildingType.....	158
7.45 แอคตีวิตี้ไคอะแกรมของโมดูล BuildingType_Actions.updateBuildingType.....	159
7.46 แอคตีวิตี้ไคอะแกรมของโมดูล BuildingType_Actions.deleteBuildingType.....	159
7.47 แอคตีวิตี้ไคอะแกรมของโมดูล ComponentItem_Actions.isNewComponentItem.	160
7.48 แอคตีวิตี้ไคอะแกรมของโมดูล ComponentItem_Actions.amountOfComponentItems.....	160
7.49 แอคตีวิตี้ไคอะแกรมของโมดูล ComponentItem_Actions.retrieveComponentItem.....	161
7.50 แอคตีวิตี้ไคอะแกรมของโมดูล ComponentItem_Actions.insertComponentItem..	161
7.51 แอคตีวิตี้ไคอะแกรมของโมดูล ComponentItem_Actions.updateComponentItem.	161
7.52 แอคตีวิตี้ไคอะแกรมของโมดูล ComponentItem_Actions.deleteComponentItem..	162
7.53 แอคตีวิตี้ไคอะแกรมของโมดูล Component_Actions.isNewComponent.....	162
7.54 แอคตีวิตี้ไคอะแกรมของโมดูล Component_Actions.getComponentName.....	163
7.55 แอคตีวิตี้ไคอะแกรมของโมดูล Component_Actions.retrieveComponent.....	163
7.56 แอคตีวิตี้ไคอะแกรมของโมดูล Component_Actions.insertComponent.....	164
7.57 แอคตีวิตี้ไคอะแกรมของโมดูล Component_Actions.updateComponent.....	164

สารบัญภาพ (ต่อ)

หน้า

รูปที่

7.58	แอกติวิตี้ไคอะแกรมของโมดูล Component_Actions.deleteComponent.....	164
7.59	แอกติวิตี้ไคอะแกรมของโมดูล Quotation_Actions.isNew QuotationItems.....	165
7.60	แอกติวิตี้ไคอะแกรมของโมดูล Quotation_Actions.hasQuotationItems.....	165
7.61	แอกติวิตี้ไคอะแกรมของโมดูล Quotation_Actions.amountOfQuotationItems.....	166
7.62	แอกติวิตี้ไคอะแกรมของโมดูล Quotation_Actions.retrieve Quotation.....	166
7.63	แอกติวิตี้ไคอะแกรมของโมดูล Quotation_Actions.insertQuotationItem.....	167
7.64	แอกติวิตี้ไคอะแกรมของโมดูล Quotation_Actions.updateQuotationItem.....	167
7.65	แอกติวิตี้ไคอะแกรมของโมดูล Quotation_Actions.deleteQuotationItem.....	167
7.66	แอกติวิตี้ไคอะแกรมของโมดูล DownPayment_Actions.isNewDownPayment.....	168
7.67	แอกติวิตี้ไคอะแกรมของโมดูล DownPayment_Actions.amountOfDownPayment.....	168
7.68	แอกติวิตี้ไคอะแกรมของโมดูล DownPayment_Actions.retrieveDownPayment.....	169
7.69	แอกติวิตี้ไคอะแกรมของโมดูล DownPayment_Actions.insertDownPayment.....	169
7.70	แอกติวิตี้ไคอะแกรมของโมดูล DownPayment_Actions.updateDownPayment.....	169
7.71	แอกติวิตี้ไคอะแกรมของโมดูล DownPayment_Actions.deleteDownPayment.....	170
7.72	แอกติวิตี้ไคอะแกรมของโมดูล NewProjectHouseNo_Actions.isNewProjectHouseNo.....	170
7.73	แอกติวิตี้ไคอะแกรมของโมดูล NewProjectHouseNo_Actions.isOverlapwith.....	171
7.74	แอกติวิตี้ไคอะแกรมของโมดูล NewProjectHouseNo_Actions.amountOfNewProjectHouseNo.....	172
7.75	แอกติวิตี้ไคอะแกรมของโมดูล NewProjectHouseNo_Actions.retrieveHouseNo.....	172
7.76	แอกติวิตี้ไคอะแกรมของโมดูล NewProjectHouseNo_Actions.retrieveProject.....	172
7.77	แอกติวิตี้ไคอะแกรมของโมดูล NewProjectHouseNo_Actions.setIsGenerated.....	173
7.78	แอกติวิตี้ไคอะแกรมของโมดูล NewProjectHouseNo_Actions.insertProjectHouseNo.....	173

สารบัญภาพ (ต่อ)

หน้า

รูปที่

7.79	แอคตีวิตี้ไคอะแกรมของโมดูล NewProjectHouseNo_Actions.updateProjectHouseNo.....	173
7.80	แอคตีวิตี้ไคอะแกรมของโมดูล NewProjectHouseNo_Actions.deleteProjectHouseNo.....	174
7.81	แอคตีวิตี้ไคอะแกรมของโมดูล Unit_Actions.isNewUnit.....	174
7.82	แอคตีวิตี้ไคอะแกรมของโมดูล Unit_Actions.sumAmount.....	175
7.83	แอคตีวิตี้ไคอะแกรมของโมดูล Unit_Actions.amountOfProjectUnit.....	175
7.84	แอคตีวิตี้ไคอะแกรมของโมดูล Unit_Actions.formatedDebtorCode.....	176
7.85	แอคตีวิตี้ไคอะแกรมของโมดูล Unit_Actions.retrieveProjectUnit.....	176
7.86	แอคตีวิตี้ไคอะแกรมของโมดูล Unit_Actions.initialDebtorCode.....	176
7.87	แอคตีวิตี้ไคอะแกรมของโมดูล Unit_Actions.generateProjectHouseNo.....	177
7.88	แอคตีวิตี้ไคอะแกรมของโมดูล Unit_Actions.insertUnit.....	177
7.89	แอคตีวิตี้ไคอะแกรมของโมดูล Unit_Actions.deleteUnit.....	178
7.90	แอคตีวิตี้ไคอะแกรมของโมดูล Unit_Actions.deleteProjectHouseNo.....	178
7.91	คอมโพเนนท์ไคอะแกรมแสดงองค์ประกอบซอฟต์แวร์ระบบตั้งหนี้โครงการ บัญชีลูกหนี้.....	179
7.92	ดีพลอยเมนต์ไคอะแกรมระบบตั้งหนี้โครงการของบัญชีลูกหนี้การเคหะแห่งชาติ.	180

บทที่ 1

บทนำ

1.1 ความเป็นมาของปัญหา

กองบัญชีลูกหนี้มีหน้าที่ในการจัดทำบัญชีเกี่ยวกับลูกหนี้โครงการของการเคหะแห่งชาติทั้งหมด ระบบตั้งหนี้เป็นส่วนหนึ่งในกองบัญชีลูกหนี้ซึ่งมีหน้าที่ในการจัดการข้อมูลเบื้องต้นที่เกี่ยวกับการขายโครงการที่อยู่อาศัยเพื่อประโยชน์ในการทำบัญชีลูกหนี้ต่อไป โดยเมื่อมีการเปิดโครงการใหม่ ฝ่ายการตลาดจะส่งรายละเอียดต่างๆเกี่ยวกับโครงการที่เปิดใหม่ และข้อมูลเกี่ยวกับการขายโครงการมายังฝ่ายงานในระบบตั้งหนี้ของกองบัญชีลูกหนี้ ซึ่งเดิมใช้ระบบคอมพิวเตอร์มาช่วยในการประมวลผลข้อมูลอยู่แล้ว แต่ยังมีปัญหา ดังนี้

1. มีรูปแบบการจัดเก็บข้อมูล และเทคโนโลยีใช้เฉพาะภายใน (stand alone) ไม่ได้เชื่อมโยงกับระบบอื่น ๆ ทั้งที่ระบบอื่นยังต้องมีการใช้ข้อมูลอย่างเดียวกันที่ระบบตั้งหนี้มี ทำให้เกิดการจัดเก็บข้อมูลซ้ำซ้อน และยากแก่การจัดการ
2. มีการกำหนดรหัสแทนข้อมูลขึ้นใช้เองซึ่งไม่ค่อยตรงกับข้อมูลที่ได้รับมาจากฝ่ายการตลาด เช่น ประเภทอาคารของฝ่ายการตลาด บอกเป็นชนิดสิ่งก่อสร้างรวมกับชื่อแบบ ซึ่งชนิดสิ่งก่อสร้างตรงกับประเภทอาคารของกองบัญชีลูกหนี้ ส่วนชื่อแบบหมายถึงองค์ประกอบในแต่ละหน่วยอาศัย เช่น แบบ D1 คือ มี 2 ห้องนอน 1 ห้องเอนกประสงค์ เป็นต้น ปัญหาที่เกิดขึ้นคือ เมื่อรับข้อมูลมาต้องตีความ และแทนความหมายให้เข้ากับรหัสที่ทางกองบัญชีใช้อยู่แล้ว
3. การกำหนดรหัสข้อมูลอยู่ในลักษณะที่แฝงความหมายลงไปด้วย (meaning full digit) เช่น รหัสลักษณะอาคาร ประกอบด้วย 4 หลัก ให้ 2 หลักแรกแทนประเภทอาคาร และที่เหลือแทนจำนวนชั้น ปัญหาที่ตามมาคือ ลักษณะอาคารบางอย่างมีการระบุถึงองค์ประกอบอื่นด้วย เช่น มี 2 ห้องนอน 1 ห้องเอนกประสงค์ ด้วย มีบางกรณีที่ว่าถ้าอาคารตั้งแต่ 10 ชั้นขึ้นไปจะไม่ระบุจำนวนห้องในหน่วยอาศัย และมีลักษณะอาคารที่ใช้แทนที่ดินโล่ง ซึ่งไม่มีองค์ประกอบดังที่กล่าวมาแล้ว

ดังนั้นจึงเกิดแนวคิดที่จะพัฒนาระบบงานให้มีระบบสารสนเทศที่มีความเป็นหนึ่งเดียวกันของการเคหะแห่งชาติ และระบบตั้งหนี้โครงการก็เป็นหนึ่งในนั้นที่จะพัฒนาให้มีการจัดเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การข้อมูลข่าวสารได้อย่างถูกต้อง เพิ่มประสิทธิภาพการทำงาน และสะดวกต่อการสืบค้น โดยระบบที่พัฒนาจะทำงานผ่านระบบคอมพิวเตอร์ที่มีการประมวลผลแบบศูนย์กลาง โดยมีการเก็บข้อมูลทั้งหมดไว้ที่ศูนย์คอมพิวเตอร์การเคหะแห่งชาติ และมีการติดตั้งเครือข่ายไปยัง สช. (สำนักงานเคหะชุมชน) และหน่วยงานภายในส่วนกลางของการเคหะแห่งชาติที่เกี่ยวข้อง

1.2 วัตถุประสงค์

1. ปรับปรุงขั้นตอนการทำงานในการตั้งหนี้โครงการ ให้มีความสะดวก รวดเร็ว และมีประสิทธิภาพอย่างต่อเนื่อง
2. ลดขั้นตอน และความยุ่งยากในการกำหนดรหัสลูกหนี้โครงการ
3. บริหารข้อมูลในการตั้งหนี้โครงการให้มีความถูกต้องทันสมัยเพื่อประโยชน์ในการชำระหนี้ของลูกหนี้โครงการต่อไป

1.3 เป้าหมาย

1. เจ้าหน้าที่ได้รับความสะดวก รวดเร็วในการตั้งหนี้โครงการใหม่
2. การตั้งหนี้โครงการมีความถูกต้อง ทันสมัย และเป็นประโยชน์ในการทำระบบบัญชีลูกหนี้ต่อไป

1.4 ขอบเขตของระบบ

ระบบตั้งหนี้โครงการบัญชีลูกหนี้ของการเคหะแห่งชาติ เป็นส่วนหนึ่งของระบบบัญชีลูกหนี้ของการเคหะแห่งชาติ เมื่อฝ่ายการตลาดมีการจัดตั้งโครงการใหม่เพื่อเสนอขาย และเปิดให้ลูกค้าจองบ้านหรือที่อยู่อาศัย รายละเอียดข้อมูลขายโครงการใหม่ ราคาขาย เงื่อนไขการผ่อนชำระ และข้อมูลบ้านเลขที่ของแต่ละหน่วยอาศัย จะถูกนำส่งมายังกองบัญชีลูกหนี้เพื่อจัดการตั้งหนี้โครงการ

1.5 ขั้นตอน และวิธีดำเนินการพัฒนาระบบ

เพื่อให้การพัฒนาระบบงานบรรลุถึงวัตถุประสงค์ที่ได้กำหนดไว้ ผู้พัฒนาจะใช้เวลาพัฒนาซอฟต์แวร์จะใช้แนวทางใหม่ คือ วงจรการพัฒนาซอฟต์แวร์เชิงวัตถุ (object-oriented software development life cycle) ซึ่งประกอบด้วย 3 กระบวนการพื้นฐาน ได้แก่ การวิเคราะห์เชิงวัตถุ (object-oriented analysis) การออกแบบเชิงวัตถุ (object-oriented design) และการอิมพลีเมนต์ (implementation) ด้วยวิธีในการโมเดลที่เป็นแบบเดียวกัน เรียกว่า UA (Unified Approach) โดยใช้ภาษาการโมเดลที่เรียกว่า UML (Unified Modeling Language)

ขั้นตอนในการพัฒนาเป็นดังนี้

1. ศึกษาระบบงานปัจจุบัน และศึกษาความเป็นไปได้ของการพัฒนาระบบ กำหนดขอบเขตของการพัฒนาระบบ กำหนดขอบเขตของการพัฒนาระบบ
2. รวบรวมเอกสารที่เกี่ยวข้องทั้งหมด ทำความเข้าใจระบบเดิม วิเคราะห์ระบบ ซึ่งในขั้นตอนวิเคราะห์ระบบต้องติดต่อกับเจ้าหน้าที่ที่เกี่ยวข้อง เพื่อให้ได้ข้อมูล และวิเคราะห์ได้ตรงความต้องการมากที่สุด
3. การวิเคราะห์เชิงวัตถุ (object-oriented analysis) จากการสัมภาษณ์ และเอกสารในข้อ 2 อธิบายโดยใช้ไดอะแกรมของยูเอ็มแอล สร้างยูสเคสไดอะแกรม อินเทอร์เฟซไดอะแกรม และคลาสไดอะแกรม
4. การออกแบบเชิงวัตถุ (object-oriented design) สร้างคลาสไดอะแกรมให้มีความสมบูรณ์ที่สุด โดยออกแบบคลาสที่วิเคราะห์ได้ในข้อที่ 3 โดยให้มีแอททริบิวต์ และการดำเนินการที่ครบถ้วนต่อหน้าที่การทำงานในระบบซึ่งได้เป็นคลาสในบิสซิเนสเลเยอร์เพื่อนำไปออกแบบโครงสร้างฐานข้อมูลเชิงสัมพันธ์ ในขั้นตอนนี้จะเน้นการสร้างแอคเซสเลเยอร์ เพื่อการออกแบบโมดูล และวิวเลเยอร์เพื่อการออกแบบยูสเซอร์อินเตอร์เฟซ
5. ออกแบบโครงสร้างฐานข้อมูล โดยใช้การแปลงคลาสในบิสซิเนสเลเยอร์ แปลงเป็นตาราง ขั้นตอนนี้จะใช้เครื่องมือ Rational rose ในการสร้าง schema ได้เป็นคำสังนิยามฐานข้อมูล ซึ่งขั้นตอนนี้เป็นการพัฒนาโดยอาศัยองค์ประกอบร่วม (Component-based development) คือใช้องค์ประกอบร่วมที่เป็น schema ของออราเคิล 8
6. การสร้างต้นแบบ (Prototyping) ขั้นตอนนี้ทำควบคู่กับการวิเคราะห์ และออกแบบ โดยใช้ Developer ของออราเคิล 8 ออกแบบเมนู ฟอรั่มยูสเซอร์อินเตอร์เฟซ และแบบรายงาน ซึ่งช่วยในการปรับปรุงไดอะแกรมต่าง ๆ ให้ตรงกับความต้องการ การสร้างต้นแบบนี้จะทำงานได้เป็นซอฟต์แวร์ที่ใช้งานได้จริง
7. จัดทำเอกสารประกอบการพัฒนาระบบ และคู่มือผู้ใช้ ขั้นตอนนี้ทำควบคู่กับพัฒนาระบบ

1.6 รายละเอียดของแต่ละบท

- บทที่ 1 เป็นบทนำกล่าวถึงความเป็นมาของปัญหา วัตถุประสงค์ เป้าหมาย ขอบเขตของระบบ ขั้นตอนแล้ววิธีการวิเคราะห์ และออกแบบระบบดังนี้ของบัญชีลูกหนี้การค้าแห่งชาติ และรายละเอียดของแต่ละบทที่นำเสนอในรายงานฉบับนี้
- บทที่ 2 อธิบายทฤษฎีที่เกี่ยวข้องที่นำมาใช้ประกอบในการวิเคราะห์และออกแบบระบบเชิงวัตถุ กล่าวถึงขั้นตอนวิธี และการนำไดอะแกรมของยูเอ็มแอลมาใช้ในการวิเคราะห์และออกแบบระบบ
- บทที่ 3 ศึกษาความเป็นไปได้ และการวิเคราะห์ระบบด้วยวิธีเชิงวัตถุ แสดงภาพรวมระบบด้วยแอคตีวิตีไดอะแกรม วิเคราะห์ยูสเคสไดอะแกรมเพื่อหาคลาสของระบบ และสร้างอินเทอแรคชันไดอะแกรมที่จะบอกถึงเหตุการณ์ที่เกิดขึ้นในระบบ ซึ่งจะช่วยในการวิเคราะห์ และออกแบบคลาส
- บทที่ 4 การออกแบบระบบ ด้วยการออกแบบคลาสที่วิเคราะห์ได้ในบทที่ 3 ต่อไป อาศัยการพัฒนาด้วยการแบ่งชั้น เป็น 3 เลเยอร์ คือ บิสซิเนสเลเยอร์ แอพเพลเลเยอร์ และวิวเลเยอร์ ซึ่งคลาสในบิสซิเนสเลเยอร์จะนำไปสู่การออกแบบโครงสร้างฐานข้อมูล คลาสในแอพเพลเลเยอร์จะนำไปสู่การออกแบบโมดูล และคลาสในวิวเลเยอร์จะนำไปสู่การออกแบบยูสเซอร์อินเตอร์เฟส
- บทที่ 5 อธิบายวิธีในการออกแบบฐานข้อมูลเชิงสัมพันธ์ โดยการแปลงคลาสในบิสซิเนสเลเยอร์มาเป็นตาราง ซึ่งอาศัยเครื่องมือของ rational rose ได้คำสั่งในการนิยามฐานข้อมูล แล้วอธิบายในรูปของแผนภาพโครงสร้างฐานข้อมูลเชิงสัมพันธ์ ที่เรียกว่า ER Diagram และอธิบายสรุปรายละเอียดโครงสร้างฐานข้อมูลด้วย
- บทที่ 6 กล่าวถึงการออกแบบยูสเซอร์อินเตอร์เฟส ได้แก่ เมนู แบบฟอร์ม และรายงาน
- บทที่ 7 การสร้างระบบ โดยนำสิ่งที่ได้จากวิเคราะห์ และออกแบบมาเขียนซอฟต์แวร์ แสดงการสร้างสตอโปรซีเจอร์ แสดงองค์ประกอบซอฟต์แวร์ด้วยคอมโพเนนท์ไดอะแกรม และแสดงสถาปัตยกรรมระบบด้วยดีพลอยเมนต์ไดอะแกรม
- บทที่ 8 สรุปโครงการพัฒนาซอฟต์แวร์ ปัญหาและอุปสรรคในการดำเนินงาน ประโยชน์ที่ได้รับ และข้อเสนอแนะต่าง ๆ

บทที่ 2

ทฤษฎีที่เกี่ยวข้อง

การกระบวนการพัฒนาซอฟต์แวร์ ประกอบด้วย การวิเคราะห์ (analysis) การออกแบบ (design) การเขียนโปรแกรม (programming) หรือการลงมือปฏิบัติ (implementation) การทดสอบ (testing) และการปรับปรุง (refinement) ซึ่งเป็นการเปลี่ยนความต้องการไปเป็นซอฟต์แวร์ที่ตอบสนองความต้องการดังกล่าว แต่บางคนยังไม่ได้ให้ความสำคัญกับกระบวนการพัฒนาซอฟต์แวร์ โดยมักเข้าไปเขียนโปรแกรมเลย เปรียบได้กับการสร้างบ้านโดยไม่มีโครงของบ้าน ซึ่งสำเร็จได้ยาก แต่ด้วยวิธีเชิงวัตถุ (object-oriented approach) จะให้ความสำคัญกับกระบวนการดังกล่าว โดยใช้เวลาส่วนใหญ่กับการหาความต้องการ พัฒนาแบบจำลองความต้องการ (requirement model) และแบบจำลองการวิเคราะห์ (analysis model) ซึ่งทำให้สามารถเขียนโปรแกรมได้รวดเร็ว อย่างไรก็ตามก็ตามควรที่จะมีการทำต้นแบบ (prototype) เพื่อหาว่าระบบที่ออกแบบมานี้คุ้มหรือไม่ ควรที่จะทำต่อไปหรือไม่ และยังทำให้ทราบว่าตรงกับความต้องการของผู้ใช้หรือไม่ เครื่องมือที่ใช้เหมาะสมเพียงใด เป็นไปตามฟังก์ชันที่ระบุหรือไม่ นอกจากนี้ยังทำให้ทราบความต้องการของผู้ใช้ด้วย

2.1 วงจรการพัฒนาซอฟต์แวร์เชิงวัตถุ (object-oriented system development life cycle)

วงจรการพัฒนาซอฟต์แวร์เชิงวัตถุประกอบด้วย 3 กระบวนการพื้นฐาน ได้แก่ การวิเคราะห์เชิงวัตถุ (object-oriented analysis) การออกแบบเชิงวัตถุ (object-oriented design) และการเขียนโปรแกรมเชิงวัตถุ (object-oriented implementation) Booch, Rumbaugh และ Jacobson ร่วมกันสร้างวิธีในการโมเดลที่เป็นแบบเดียวกัน เรียกว่า Unified Approach (UA) โดยใช้ภาษาการโมเดลที่เรียกว่า UML (Unified Modeling Language) โดยกระบวนการในการพัฒนาประกอบด้วยกิจกรรมดังนี้

1. การวิเคราะห์เชิงวัตถุ (object-oriented analysis)
2. การออกแบบเชิงวัตถุ (object-oriented design)
3. การสร้างต้นแบบ (Prototyping)
4. การพัฒนาโดยอาศัยองค์ประกอบร่วม (Component-based development)

การทดสอบแบบเพิ่ม (Incremental testing) ออบเจกต์จะขึ้นกับยูสเคสโมเดล (use-case model) หรือเรียกวิธีนี้ว่า use-case model approach

2.2 การพัฒนาซอฟต์แวร์โดยการแบ่งชั้น (Layered Approach to Software Development)

ปัจจุบันการพัฒนาระบบโดยใช้เคสทูล (CASE tool) หรือการพัฒนาโดยสภาพแวดล้อมที่เป็นไคลเอน/เซิร์ฟเวอร์ มักมีสถาปัตยกรรมแบบสองชั้น คือ ชั้นของการติดต่อ (interface layer) และการเข้าถึงข้อมูล (access layer) แต่สถาปัตยกรรมที่ดีกว่ามีชั้นของการทำงาน (business layer) เพิ่มมาเป็นการเพิ่มรายละเอียดการทำงานระหว่าง 2 ชั้นแรก กลายเป็นวิธีสามชั้น (three-layer approach)

1. บิสซิเนสเลเยอร์ (Business Layer)

ชั้นของการทำงานประกอบด้วยออบเจกต์ที่เป็นตัวแทนของระบบ เช่น คำสั่งซื้อ ลูกค้า รายการซื้อ เป็นต้น หน้าที่ของชั้นนี้เป็นการโมเดลออบเจกต์ และการทำงานระหว่างออบเจกต์ในระบบ ซึ่งไม่จำเป็นต้องมีรายละเอียดเกี่ยวกับวิธีในการแสดง และวิธีในการเข้าถึงข้อมูล

2. วิวเลเยอร์ (View Layer) หรือยูสเซอร์อินเตอร์เฟซเลเยอร์ (User Interface Layer)

ชั้นของการติดต่อประกอบด้วยออบเจกต์ที่ผู้ใช้ใช้ในการควบคุม และจัดการติดต่อ และการแสดงผลการทำงานของออบเจกต์

3. แอ็กเซสเลเยอร์ (Access Layer)

ชั้นการเข้าถึงข้อมูลประกอบด้วยออบเจกต์ที่รู้วิธีการสื่อสารกับแหล่งที่เก็บข้อมูลทางกายภาพ ซึ่งชั้นนี้มีหน้าที่หลักคือ การร้องขอในการส่งข้อมูลจากชั้นการทำงานด้วยโปรโตคอลหรือข้อตกลงที่เหมาะสม เพื่อเข้าถึงข้อมูล และการส่งผลการเข้าถึงข้อมูลจากแหล่งข้อมูลดังกล่าว

2.3 UML (Unified Modeling Language)

ในการวิเคราะห์ และออกแบบระบบมักจะใช้เทคนิคในการโมเดลโดยใช้รูปภาพเป็นสื่อ หรือภาษารูปภาพ (graphical language) มีสัญลักษณ์ต่าง ๆ ที่จะแทนความหมายได้ดีกว่าการอธิบาย ด้วยข้อความ ซึ่งในเคสทูล (CASE tool) จะใช้ร่วมกับ methodology ต่าง ๆ

UML เป็นภาษารูปภาพที่ใช้ในการเขียนข้อกำหนด การพัฒนา และการทำเอกสารสำหรับ ระบบซอฟต์แวร์ ซึ่งประกอบด้วยกฎ และความหมาย (rule and semantic) โมเดลที่ประกอบด้วยกฎ และความหมายสามารถอธิบายด้วยภาษาอังกฤษ ในรูปของ OCL (object constraint language) ทำให้สามารถแปลงไปเป็นภาษาการ โปรแกรมเชิงวัตถุได้

ข้อได้เปรียบของ UML

1. มีสัญลักษณ์ที่พร้อมในการสร้างโมเดล และสื่อความหมายได้ดี
2. สามารถขยายการใช้งานด้วยหลักการ
3. ภาษาการ โปรแกรมควบคุมไปกับกระบวนการพัฒนา
4. ภาษาในการโมเดลมีรูปแบบที่เข้าใจตรงกัน
5. สนับสนุนเครื่องมือที่เกี่ยวกับเชิงวัตถุในตลาด
6. เป็นการรวบรวมวิธีที่ดีที่สุดมาใช้

เนื่องจากทุกระบบประกอบด้วยคุณลักษณะ และพฤติกรรม ดังนั้นโมเดลของทุกระบบจะ ประกอบด้วย 2 โมเดล คือสแตติกโมเดล (static model) เพื่อโมเดลโครงสร้าง และ ไดนามิกโมเดล (dynamic model) เพื่อโมเดลกระบวนการทำงาน กิจกรรม หรือพฤติกรรมของระบบ ในขั้นตอน การวิเคราะห์จะสร้างสแตติกโมเดล ใช้ในการอธิบายโครงสร้างของออบเจกต์ และความสัมพันธ์ ระหว่างออบเจกต์ในระบบ ส่วน ไดนามิกโมเดล จะมีประโยชน์ในขั้นตอนการออกแบบ และการ โปรแกรม

ยูเอ็มแอลมีไดอะแกรมรูปภาพให้ใช้ 9 ไดอะแกรม ดังต่อไปนี้

1. ยูสเคสไดอะแกรม (Use-case diagram)

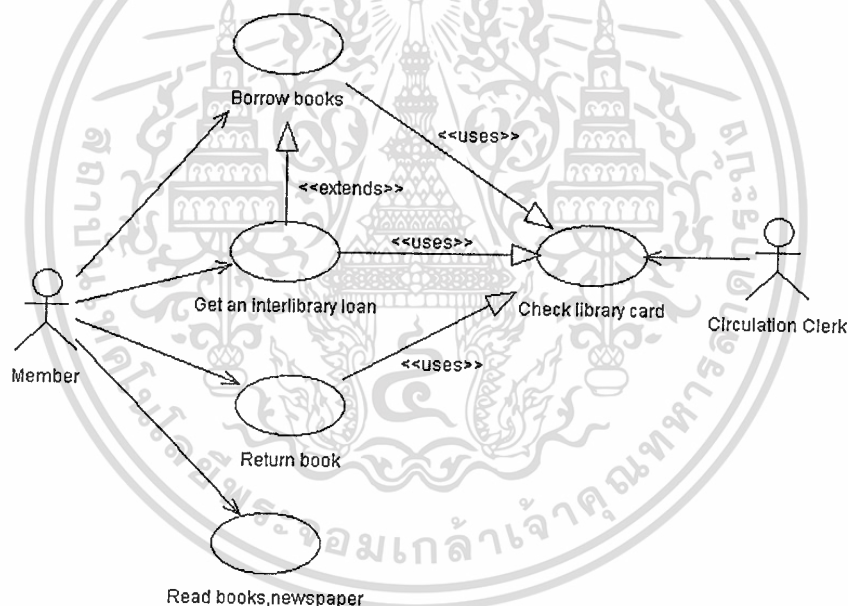
แสดงเหตุการณ์ ผู้ทำให้เกิดเหตุการณ์ และการตอบสนองของระบบของระบบ โดยทำ การแบ่งปัญหาขอบเขตหน้าที่ของระบบทั้งหมด ให้เป็นส่วนเล็กลงเรียกว่า ยูสเคส (use case)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แทนด้วย สัญลักษณ์วงรี แอคเตอร์ (actor) แทนสิ่งที่ติดต่อกับระบบเพื่อแลกเปลี่ยนสารสนเทศ อาจจะเป็นบุคคล หรือระบบภายนอก

ความสัมพันธ์ภายในไดอะแกรม มี 2 ชนิด คือ

- 1) Communication เป็นความสัมพันธ์ระหว่างแอคเตอร์กับยูสเคส แทนด้วยสัญลักษณ์เส้นลูกศรหัวเปิดทางด้านที่ยูสเคส ตัวอย่างดังรูปที่ 2.1
- 2) Association เป็นความสัมพันธ์ระหว่างยูสเคส แทนด้วยสัญลักษณ์เส้นลูกศรหัวปิดกลวงมี 2 แบบ คือ uses ใช้เมื่อยูสเคสหนึ่งต้องการใช้อีกยูสเคสหนึ่ง และแบบ extends ใช้เมื่อมียูสเคสที่มีกิจกรรมเหมือนกับอีกยูสเคสหนึ่ง แต่มีกิจกรรมบางอย่างที่เฉพาะกว่า ตัวอย่างดังรูปที่ ดังรูปที่ 2.1



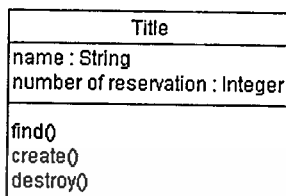
รูปที่ 2.1 ตัวอย่าง Use-case diagram

2. คลาสไดอะแกรม (Class diagram)

Class diagram เป็นการโมเดลเชิงวัตถุ แสดงโครงสร้างที่คงที่ (static model) ของโมเดล เป็นการแทนสิ่งต่าง ๆ ในระบบด้วยด้วยคลาส (class) และความสัมพันธ์ (association) ระหว่างคลาสที่จะประสานการทำงานในระบบ

1) คลาส

คลาสประกอบด้วย 3 ส่วนคือ ชื่อคลาส (class name) คุณสมบัติ (attributes) และการดำเนินการ (operations) ตัวอย่างคลาส เป็นดังรูปที่ 2.2



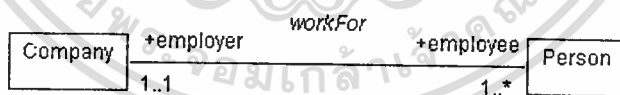
รูปที่ 2.2 ตัวอย่างคลาส Title

2) จำนวนความสัมพันธ์ระหว่างคลาส (Multiplicity)

จำนวนความสัมพันธ์ระหว่างคลาสดำหนดเป็นช่วงในรูปแบบ จำนวนต่ำสุด..จำนวนสูงสุด ตัวอย่างเช่น 0..1 0..* 1..* เป็นต้น

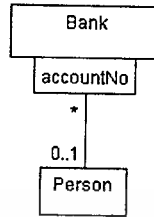
3) ความสัมพันธ์ระหว่างคลาส

- Association เป็นความสัมพันธ์ระหว่างคลาสอาจจะเป็นแบบสองทาง (binary association) หรือมากกว่า (n-ary association) ตัวอย่างความสัมพันธ์แบบสองทางดังรูปที่ 2.3 บทบาท (role) ของแต่ละคลาสถูกแสดงติดกับคลาส ชื่อความสัมพันธ์ จะบอกความสัมพันธ์จากซ้ายไปขวา



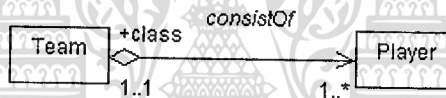
รูปที่ 2.3 ตัวอย่างความสัมพันธ์แบบ association

กรณีทีคลาสหนึ่งมีความสัมพันธ์กับอีกคลาสหนึ่งด้วยแอทริบิวต์ จะเรียกแอทริบิวต์ดังกล่าวว่า qualifier ตัวอย่างเช่นรูปที่ 2.4 บอกคนจะเป็นลูกค้าธนาคารโดยมีหมายเลขบัญชี



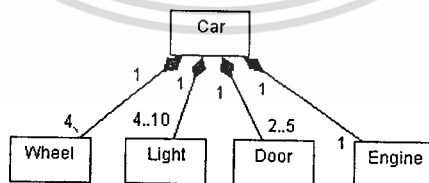
รูปที่ 2.4 ตัวอย่างความสัมพันธ์แบบ association ที่สัมพันธ์กันด้วยแอทริบิวต์

- Aggregation เป็นความสัมพันธ์แบบ association แบบหนึ่ง โดยที่คลาสหนึ่งเป็นส่วนหนึ่งของอีกคลาสหนึ่ง (whole-part) ใช้สัญลักษณ์ลูกศรที่มีหัวเป็นรูปสี่เหลี่ยมขนมเปียกปูนไปร่งทางด้านที่เป็นทั้งหมด (whole) ตัวอย่างดังรูปที่ 2.5



รูปที่ 2.5 ตัวอย่างความสัมพันธ์แบบ aggregation ทีมประกอบด้วยนักกีฬา

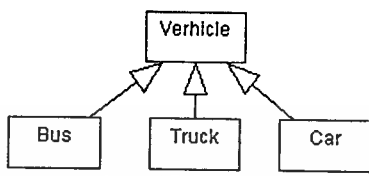
- Composition aggregation เป็นความสัมพันธ์หากว่าไม่มีคลาสทางด้านทั้งหมด (whole) จะไม่สามารถมีคลาสทางด้านบางส่วน (part) ได้ ใช้สัญลักษณ์ลูกศรเป็นรูปสี่เหลี่ยมขนมเปียกปูนทึบ ตัวอย่างดังรูปที่ 2.6



รูปที่ 2.6 ตัวอย่างความสัมพันธ์แบบ composition ล้อ ไฟ ประตู่ และเครื่องจักร เป็นส่วนหนึ่งของรถ

- Generalization เป็นความสัมพันธ์ระหว่างคลาสทั่วไปกับคลาสที่เฉพาะกว่า หรือเรียกว่าความสัมพันธ์แบบ is-a ซึ่งคลาสทั่วไป เรียกว่า ซุปเปอร์คลาส (superclass) และคลาส

ที่เฉพาะกว่า เรียกว่า ซับคลาส (subclass) จะได้รับการถ่ายทอดคุณสมบัติทั้งแอทริบิวต์ และการดำเนินการจากซูเปอร์คลาส ใช้สัญลักษณ์ลูกศรหัวสามเหลี่ยมโปร่งจากซับ คลาสไปยังซูเปอร์คลาส ตัวอย่างดังรูปที่ 2.7



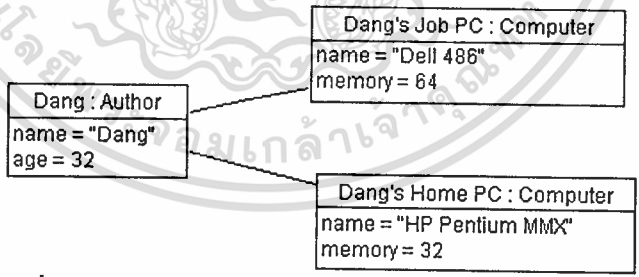
รูปที่ 2.7 ตัวอย่างความสัมพันธ์แบบ generalization ขานพาหะแบ่งได้หลายชนิด

3. ออบเจกต์ไดอะแกรม (Object diagram)

ออบเจกต์ไดอะแกรม เป็นไดอะแกรมที่แสดงความเป็นตัวตน (instance) ของคลาสไดอะแกรม ณ เวลานั้น ตัวอย่าง คลาสไดอะแกรม ดังรูปที่ 2.8 และ ออบเจกต์ไดอะแกรมดังรูปที่ 2.9



รูปที่ 2.8 ตัวอย่างคลาสไดอะแกรม

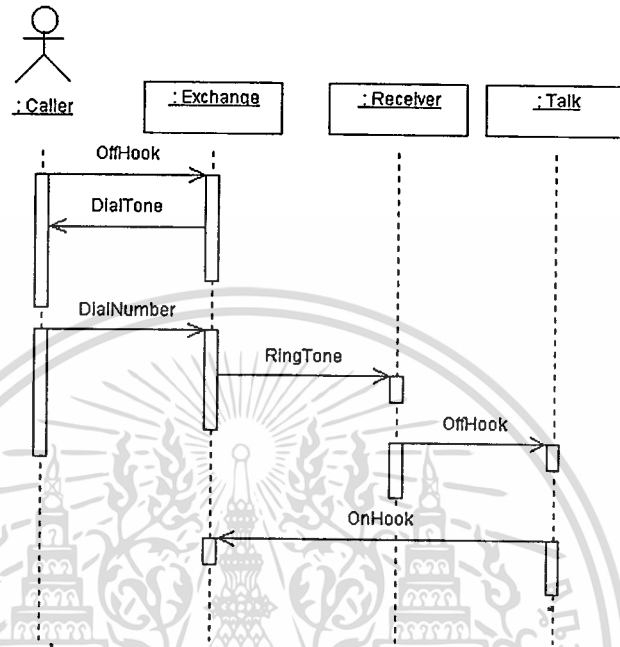


รูปที่ 2.9 ตัวอย่างออบเจกต์ไดอะแกรม

4. ซีควีนซ์ไดอะแกรม (Sequence diagram)

ซีควีนซ์ไดอะแกรมแสดงการติดต่อในการทำงานระหว่างออบเจกต์ในระบบตามลำดับของเวลา (time sequence) ไดอะแกรมนี้จะมี 2 มิติ คือ ในแนวตั้งแสดงเวลา และแนวนอนแสดงออบเจกต์ที่แตกต่างกัน เส้นตามแนวตั้งเรียกว่า เส้นชีวิต (life line) บ่งบอกออบเจกต์ที่

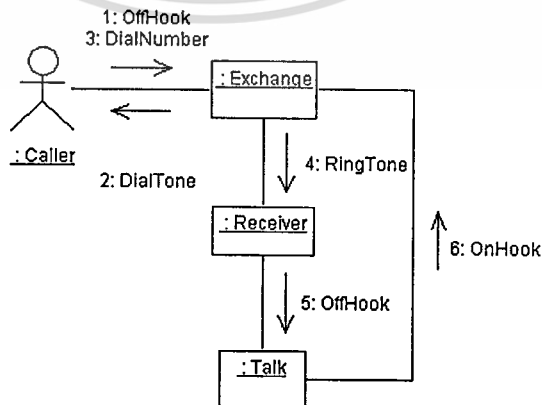
ปรากฏขึ้นในการทำกิจกรรมกับออบเจกต์อื่น ระหว่างออบเจกต์ในแนวตั้งจะมีเส้นในแนวนอนแสดงการส่งผ่านข้อความติดต่อกัน ตัวอย่างดังรูปที่ 2.10



รูปที่ 2.10 ตัวอย่างซีควีนโคอะแกรม

5. คอลแลบอเรชันโคอะแกรม (Collaboration diagram)

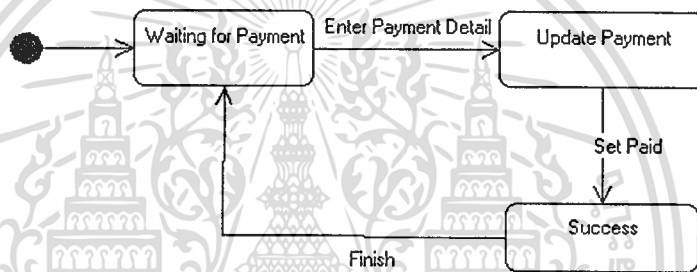
คอลแลบอเรชันโคอะแกรม แสดงการทำงานประสานกันระหว่างออบเจกต์ ซึ่งสอดคล้องกับซีควีนโคอะแกรม ต่างกันที่หมายเลขข้อความที่ส่งผ่านจะบอกลำดับการทำงานของออบเจกต์ และรูปแบบในการนำเสนอกระชับกว่า ตัวอย่างดังรูปที่ 2.11



รูปที่ 2.11 ตัวอย่างคอลแลบอเรชันโคอะแกรม

6. สเตทชาร์ทไดอะแกรม (Statechart diagram)

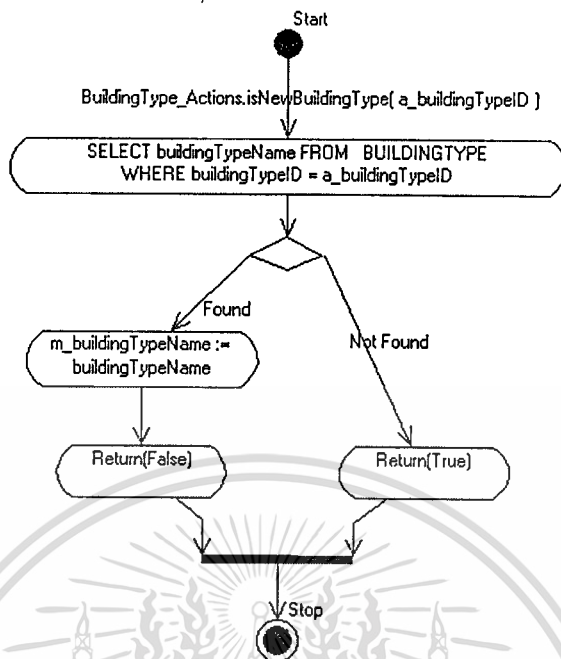
สเตทชาร์ทไดอะแกรมใช้อธิบายวงจรชีวิตของออบเจกต์ ระบบงานย่อย และระบบงาน ในแต่ละคลาสสามารถกำหนดสถานะ (state) ให้ชัดเจน รวมถึงเหตุการณ์ที่ทำให้เกิดการเปลี่ยนแปลงตามแต่ละสถานะ ซึ่งไม่จำเป็นต้องเขียนสเตทชาร์ทไดอะแกรมของทุกคลาส จะเขียนก็ต่อเมื่อมีสถานะที่ต่างกัน และมีผลกระทบหรือเปลี่ยนแปลงพฤติกรรมของคลาส สัญลักษณ์วงกลมทึบเป็นจุดเริ่มต้น หรือเริ่มสร้างออบเจกต์ ส่วนจุดสิ้นสุด แทนด้วยสัญลักษณ์วงกลมทึบที่มีวงกลมล้อมรอบและลูกศรระหว่างสถานะแสดงการเปลี่ยนแปลงระหว่างสถานะ หรือเหตุการณ์ที่เกิด ตัวอย่างดังรูปที่ 2.12



รูปที่ 2.12 ตัวอย่างสเตทชาร์ทไดอะแกรม

7. แอกติวิตีไดอะแกรม (Activity diagram)

แอกติวิตีไดอะแกรมแสดงการไหลของการกระทำ (action) กิจกรรม (activity) การสร้างการดำเนินการ และผลลัพธ์ของการกระทำนั้น คล้ายกับสเตทไดอะแกรม แต่จุดประสงค์ต่างกันตรงที่ แอกติวิตีไดอะแกรมเน้นที่การกระทำ และผลลัพธ์ ก็คือการเปลี่ยนแปลงสถานะของออบเจกต์ในสเตทชาร์ทไดอะแกรม สัญลักษณ์ที่ใช้เหมือนกับสเตทชาร์ทไดอะแกรมแต่มีสัญลักษณ์ในการตัดสินใจ (decision) เพิ่มมา เป็นรูปสี่เหลี่ยมขนมเปียกปูน ตัวอย่างดังรูปที่ 2.13



รูปที่ 2.13 ตัวอย่างแอคตีวิตี้ไดอะแกรม

8. คอมโพเนนท์ไดอะแกรม (Component diagram)

คอมโพเนนท์ไดอะแกรมอธิบายองค์ประกอบ (component) ทางกายภาพของการออกแบบอย่างเช่น โค้ดต้นฉบับ (source code) โปรแกรมที่ใช้งาน (executable program) และการติดต่อกับผู้ใช้ (user interface) เป็นต้น สัญลักษณ์ที่ใช้แทนคอมโพเนนท์ ตัวอย่างดังรูปที่ 2.14

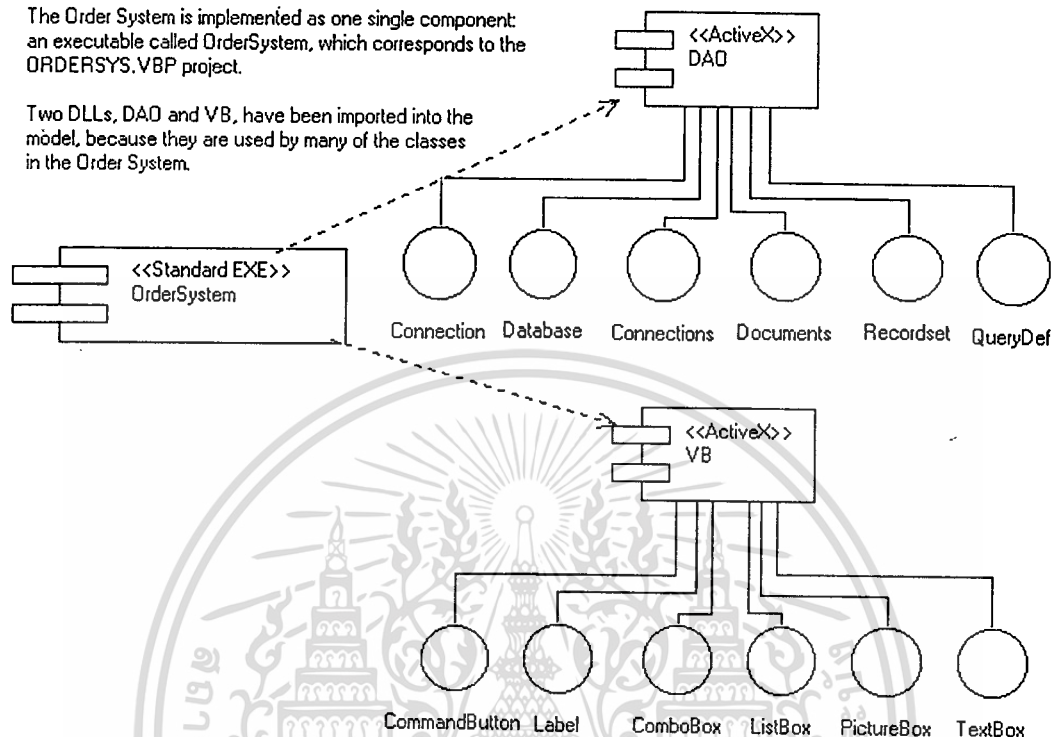
9. ดีพลอยเมนต์ไดอะแกรม (Deployment diagram)

เป็นการทำนายสถาปัตยกรรมของระบบ ขณะที่โปรแกรมกำลังทำงาน (run-time) รวมถึง โปรโตคอล (protocol) ที่เชื่อมต่อระหว่างโหนด (node) ซึ่งได้แก่โปรเซสเซอร์ อุปกรณ์ต่างๆ และซอฟต์แวร์ต่างๆ ไว้ด้วยกัน สัญลักษณ์ที่ใช้แทนโหนดคือกล่องสี่เหลี่ยม ส่วนสัญลักษณ์เส้นทึบที่ลากเชื่อมกันระหว่างโหนด แทนการเชื่อมโยง โดยมีป้ายกำกับบอกถึงโปรโตคอลในการเชื่อมโยงด้วย ตัวอย่างดังรูปที่ 2.15

Order System Components

The Order System is implemented as one single component: an executable called OrderSystem, which corresponds to the ORDERSYS.VBP project.

Two DLLs, DAO and VB, have been imported into the model, because they are used by many of the classes in the Order System.



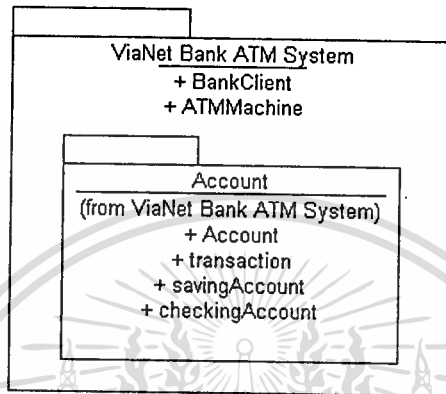
รูปที่ 2.14 ตัวอย่าง คอมโพเนนต์ที่โคอะแกรม



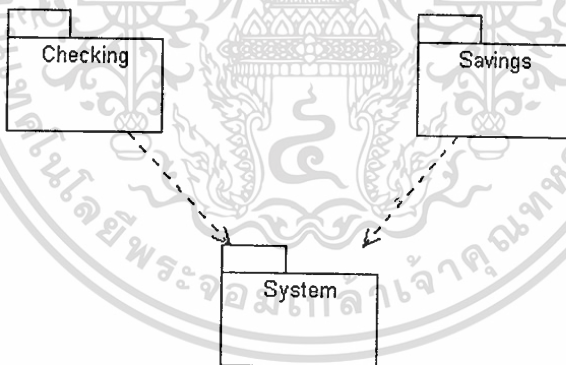
รูปที่ 2.15 ตัวอย่าง คีพลอยเมนต์ที่โคอะแกรม

ในการจัดการโมเดลสามารถรวบรวมกลุ่มโมเดลเข้าไว้ในแพ็คเกจ (package) โดยที่ในแพ็คเกจสามารถมีแพ็คเกจย่อยได้ หรือมีทั้งโมเดล และแพ็คเกจย่อยภายใน สัญลักษณ์ที่ใช้แทนแพ็คเกจเป็นรูปโพลีเดร่ ตัวอย่างรูปที่ 2.16 แสดงแพ็คเกจของคลาสโคอะแกรม Vianet Bank ATM System ประกอบด้วยคลาส BankClient คลาส ATMMachine และแพ็คเกจย่อย Account ซึ่งประกอบด้วยคลาส Account คลาส transaction คลาส savingAccount และ คลาส checkingAccount ส่วนตัวอย่างรูปที่ 2.17 แสดงตัวอย่างแพ็คเกจของยูสเคสโคอะแกรมที่มีความสัมพันธ์แบบขึ้นแก่กัน โดยมีเส้นประที่มีหัวลูกศรเปิดฉาก

จากแพ็คเกจหนึ่งไปยังอีกแพ็คเกจหนึ่งซึ่งหัวลูกศรจะอยู่ทางด้านแพ็คเกจที่มีอิทธิพล จากตัวอย่างนี้แพ็คเกจ savings และแพ็คเกจ checking ขึ้นกับแพ็คเกจ system



รูปที่ 2.16 ตัวอย่างแพ็คเกจซึ่งแสดงองค์ประกอบภายใน



รูปที่ 2.17 ตัวอย่างแพ็คเกจที่มีความสัมพันธ์แบบขึ้นแก่กัน

2.4 การวิเคราะห์เชิงวัตถุ (Object-oriented analysis)

การวิเคราะห์เชิงวัตถุตามแนวคิด UA จะใช้แอกเตอร์ และยูสเคสในการอธิบายระบบในมุมมองของผู้ใช้ ซึ่งแอกเตอร์เป็นปัจจัยภายนอกที่ติดต่อกับระบบ ส่วนยูสเคสเป็นเหตุการณ์ที่อธิบายว่าแอกเตอร์จะใช้ระบบอย่างไร

กระบวนการในการวิเคราะห์เชิงวัตถุประกอบด้วยขั้นตอนดังนี้

1. มองหาแอกเตอร์ซึ่งเป็นผู้ใช้หรือผู้ติดต่อกับระบบ
2. สร้างโมเดลที่แสดงถึงกิจกรรมภาพรวมของระบบด้วย activity diagram
3. กำหนดยูสเคส โดยพิจารณาว่ากิจกรรม หรือเหตุการณ์ใดที่แอกเตอร์ทำกับระบบ
4. สร้าง interaction diagram ได้แก่ sequence diagram และ collaboration diagram
5. สร้าง class diagram
6. ทบทวน และปรับปรุงโดยการย้อนไปขั้นตอนก่อนหน้า

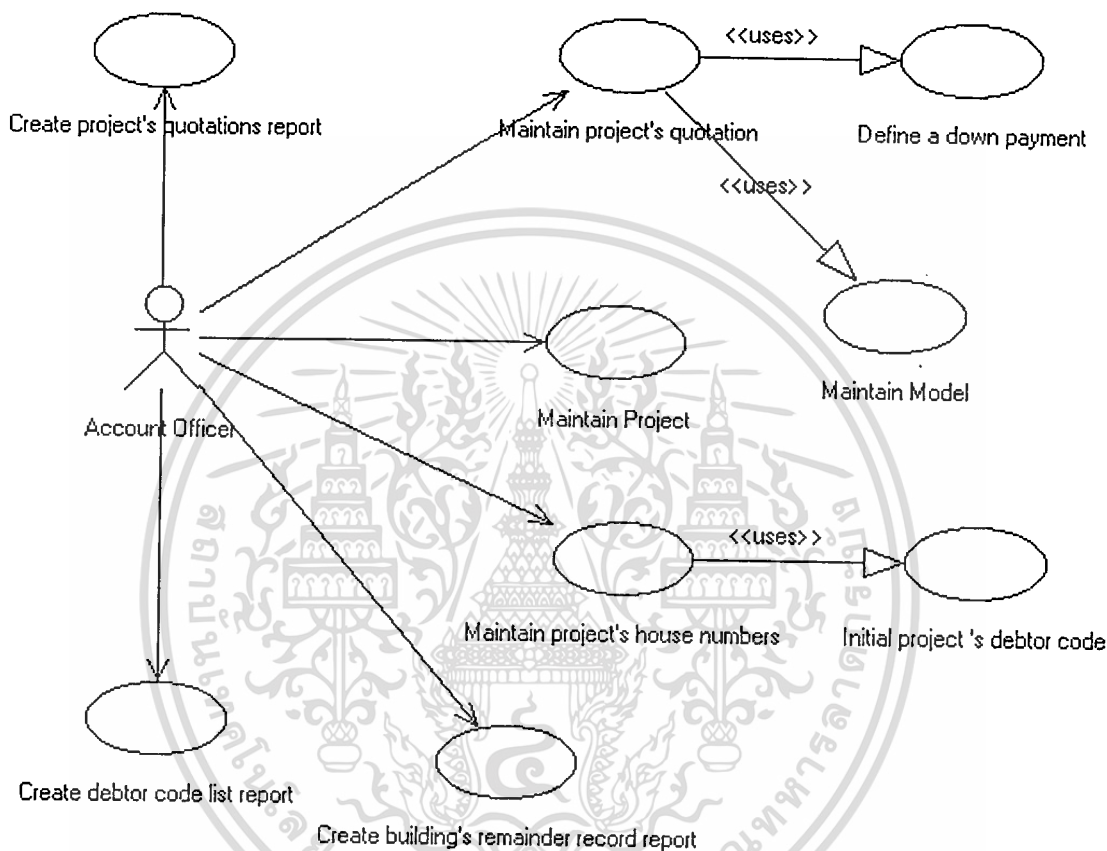
2.5 การออกแบบเชิงวัตถุ (object-oriented design)

ในระหว่างการออกแบบจะมีปรับคลาสที่ได้มาจากตอนวิเคราะห์ระบบ เพื่อให้เหมาะกับการโปรแกรม โดยอาจมีการสร้างคลาส แอททริบิวต์ หรือ การดำเนินการเพิ่มเติมเข้ามาเพื่อการโปรแกรม และการสร้างการติดต่อกับผู้ใช้ ที่เรียกว่า UI (user interface)

กระบวนการในการออกแบบเชิงวัตถุประกอบด้วยขั้นตอนดังนี้

1. พยายามทำให้คลาสใดอะแกรมมีความสมบูรณ์ที่สุด โดยเพิ่มรายละเอียดของ แอททริบิวต์ การดำเนินการ และความสัมพันธ์ระหว่างคลาส ซึ่งคลาสเหล่านี้จะอยู่ในบิสซิเนสเลเยอร์
2. สร้างแอกเซสเลเยอร์โดยสร้างคลาสในการเข้าถึงที่มีรูปแบบคล้ายกับคลาสในบิสซิเนสเลเยอร์ แล้วสร้างความสัมพันธ์ระหว่างคลาสที่สร้างขึ้นด้วย
3. ออกแบบคลาสสำหรับวิวเลเยอร์ ได้แก่ คลาสสำหรับ UI แล้ว
4. สร้างต้นแบบ หรือโปรโตไทป์สำหรับคลาสในวิวเลเยอร์ ให้ตรงกับความต้องการ และผู้ใช้พอใจ โดยมีการทดสอบการใช้งาน
5. ทบทวน และปรับปรุง ทั้งการวิเคราะห์ และการออกแบบ

3.2 ยูสเคสไดอะแกรมระบบ



รูปที่ 3.2 ยูสเคสไดอะแกรมระบบ

1. แอคเตอร์ (Actors)

Account Officer หรือ เจ้าหน้าที่การบัญชีเป็นผู้มีหน้าที่ในการจัดการข้อมูลเบื้องต้นเกี่ยวกับการขาย และเป็นผู้ตั้งหนี้โครงการของการเคหะแห่งชาติ

2. ยูสเคส (Usecases)

ความต้องการของระบบวิเคราะห์ได้จากความสัมพันธ์ระหว่างแอคเตอร์ กับยูสเคส และระหว่างยูสเคสกับยูสเคสที่ขึ้นแก่กัน ในการตั้งหนี้โครงการใหม่ คือการกำหนดรหัสลูกหนี้ (Initial project's debtor code) ซึ่งจำเป็นต้องใช้ข้อมูลจากการจัดการข้อมูลขายโครงการ

(Maintain project's quotation) และการจัดการบ้านเลขที่ให้กับหน่วยอาศัย (Maintain project's house numbers) ตารางที่ 3.1 แสดงคำอธิบายสรุปความสัมพันธ์ระหว่างแอคเตอร์ กับยูสเคส และระหว่างยูสเคสกับยูสเคสในระบบ

ตารางที่ 3.1 สรุปความสัมพันธ์องค์ประกอบภายในยูสเคสไดอะแกรม (ส่วนที่ 1/2)

ยูสเคส	ภาพรวม	ความสัมพันธ์กับยูสเคสอื่น
1. Maintain project's quotation	เมื่อกองการขายฝ่ายการตลาดส่งข้อมูลขายโครงการใหม่ เสนอไขการผ่อนชำระทั้งที่เป็นการผ่อนชำระเงินล่วงหน้า และอัตราดอกเบี้ยการผ่อนชำระค่าเช่าซื้อต่อปีมาให้กับกองบัญชีลูกหนี้ และเจ้าหน้าที่การบัญชีจะเป็นผู้จัดการกับข้อมูลดังกล่าวไว้ใช้อ้างอิง และใช้ในการตั้งหนี้โครงการใหม่	1. ใช้ยูสเคส Define a down payment ในการกำหนดการผ่อนชำระล่วงหน้า 2. ใช้ยูสเคส Maintain Model เมื่อมีการเพิ่ม ลบ หรือแก้ไขข้อมูลโมเดลของสิ่งปลูกสร้าง 3. ใช้ยูสเคส Maintain Project เมื่อมีการเพิ่ม ลบ หรือแก้ไขข้อมูลโครงการ
2. Define a down payment	กำหนดการผ่อนชำระล่วงหน้าสำหรับโมเดลสิ่งปลูกสร้างของแต่ละโครงการ	1. ถูกใช้โดยยูสเคส Maintain project's quotation
3. Maintain project's house numbers	เมื่อกองการขายฝ่ายการตลาดส่งข้อมูลบ้านเลขที่ของแต่ละหน่วยอาศัยในโครงการมาให้กับกองบัญชีลูกหนี้ และเจ้าหน้าที่การบัญชีจะเป็นผู้จัดการกับข้อมูลดังกล่าวไว้กำหนดเลขที่บ้านสำหรับหน่วยอาศัย ตามโมเดลของโครงการนั้น ๆ	1. ใช้ยูสเคส Initial project 's debtor code ในการกำหนดบ้านเลขที่ให้กับแต่ละหน่วยอาศัย และตั้งหนี้โครงการ 2. ใช้ยูสเคส Maintain Model เมื่อมีการเพิ่ม ลบ หรือแก้ไขข้อมูลโมเดลของสิ่งปลูกสร้าง 3. ใช้ยูสเคส Maintain Project เมื่อมีการเพิ่ม ลบ หรือแก้ไขข้อมูลโครงการ
4. Initial project 's debtor code	ตั้งหนี้โครงการโดยการกำหนดรหัสลูกหนี้สำหรับแต่ละหน่วยอาศัยตามตามรหัสโครงการ โมเดล และบ้านเลขที่	-

ตารางที่ 3.1 สรุปความสัมพันธ์องค์ประกอบภายในยูสเคสไดอะแกรม (ส่วนที่ 2/2)

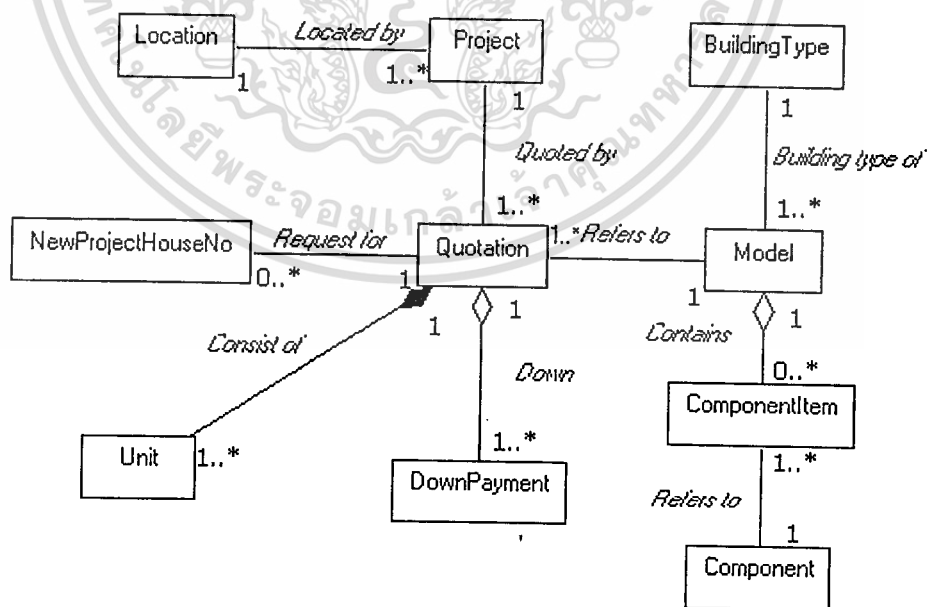
ยูสเคส	ภาพรวม	ความสัมพันธ์กับยูสเคสอื่น
5. Maintain Model	เพิ่ม ลบ หรือแก้ไขข้อมูลโมเดลสิ่งปลูกสร้าง	1. ถูกใช้โดยยูสเคส Maintain project quotation เมื่อมีการเพิ่ม ลบ หรือแก้ไขข้อมูลโมเดลของสิ่งปลูกสร้างขณะที่จัดการข้อมูลการขาย
6. Maintain Project	เพิ่ม ลบ หรือแก้ไขข้อมูลโครงการ	-
7. Create project's quotations report	จัดทำรายงานการขายโครงการ	-
8. Create building's remainder record report	จัดทำรายงานสรุปทะเบียนอาคารคงเหลือที่ยังไม่มีการตั้งหนี้โครงการ	-
9. Create debtor code list report	จัดทำรายงานรายการรหัสลูกหนี้โครงการ	-

3.3 คลาสไดอะแกรมจากการวิเคราะห์ระบบ

จากยูสเคสไดอะแกรมข้างต้น (รูปที่ 3.2) สามารถวิเคราะห์หาคلاسในระบบได้ดังนี้

Project	โครงการ
Location	ที่ตั้งสำนักงานเคหะชุมชน
Unit	หน่วยอาศัย
BuildingType	ประเภทสิ่งปลูกสร้าง
Model	โมเดล
Quotation	ข้อมูลขายโครงการ
NewProjectHouseNo	บ้านเลขที่โครงการ
Component	องค์ประกอบในโมเดล
ComponentItem	รายการองค์ประกอบในโมเดล
DownPayment	การผ่อนชำระเงินล่วงหน้า

และนำมาสร้างเป็นคลาสไดอะแกรมดังรูปที่ 3.3 และคำอธิบายสรุปดังตารางที่ 3.2



รูปที่ 3.3 คลาสไดอะแกรมที่ได้จากการวิเคราะห์

ตารางที่ 3.2 สรุปความสัมพันธ์ระหว่างคลาสในระบบ

คลาสที่มีความสัมพันธ์กัน	ความสัมพันธ์	คำอธิบาย
1. Location กับ Project	Located by	ในเขตของสำนักงานเคหะชุมชนหนึ่ง มีหลายโครงการที่อยู่อาศัย
2. BuildingType กับ Model	Building type of	โมเดล แต่ละโมเดลจะอยู่ในประเภทสิ่งปลูกสร้างหนึ่งเท่านั้น เช่น โมเดล F1 เป็นสิ่งปลูกสร้างประเภท บ้านเดี่ยว
3. Model กับ ComponentItem	Contains	แต่ละโมเดลจะมีรายการองค์ประกอบ เช่น ห้องนอน ห้องเอนกประสงค์ ซึ่งบางโมเดลอาจจะไม่มีองค์ประกอบเลยก็ได้
4. ComponentItem กับ Component	Refer to	ในแต่ละรายการองค์ประกอบของโมเดลมีการอ้างถึงชื่อองค์ประกอบ
5. Project กับ Quotation	Quote by	โครงการหนึ่งจะมีการเสนอขายหลายรายการ ซึ่งรายการหนึ่งสำหรับแต่ละโมเดล
6. Quotation กับ Model	Refers to	ข้อมูลการขายแต่ละรายการจะอ้างถึงโมเดล
7. Quotation กับ DownPayment	Down	ในข้อมูลการขายแต่ละรายการจะมีการผ่อนชำระล่วงหน้าหลายงวดด้วยกัน หรืออาจจะไม่มีเลยก็ได้
8. NewProjectHouseNo กับ Quotation	Request for	การกำหนดบ้านเลขที่ให้กับข้อมูลขายแต่ละ โมเดล นั้นจะมีหลายบ้านเลขที่
9. Quotation กับ Unit	Consist of	ในข้อมูลการขายหนึ่งรายการหรือ แต่ละ โมเดล จะต้องประกอบด้วยหน่วยอาศัยหลายหน่วย

คลาสที่ได้จากการวิเคราะห์จะประกอบด้วยแอทริบิวต์ และการดำเนินการ ซึ่งไม่ได้แสดงไว้ในบทนี้ แต่จะมีการออกแบบต่อไปในบทที่ 4 โดยอาศัยยูสเคสไดอะแกรม ซีควีนซ์ไดอะแกรม และคอลแลบอเลชันไดอะแกรม ในการปรับปรุงแอทริบิวต์ กำหนดชนิดของข้อมูล และรูปแบบการใช้การดำเนินการให้สอดคล้องกับความต้องการ

3.4 อินเทอร์เฟซหน้าไออะแกรม

พิจารณายูสเคสไออะแกรมข้างต้น แต่ละยูสเคสแสดงถึงเหตุการณ์ หรือหน้าที่การทำงานของระบบ ซึ่งสามารถนำมาอธิบายโดยใช้อินเทอร์เฟซหน้าไออะแกรม อาจจะเป็นชีเควินไออะแกรม หรือ คอลแลปบอเรชันไออะแกรมโดยพิจารณาไปแต่ละยูสเคสดังต่อไปนี้

1. ยูสเคส Maintain project's quotation

- ชีเควินไออะแกรม Add project's quotation ดังรูปที่ 3.4
- คอลแลปบอเรชันไออะแกรม Add project's quotation ดังรูปที่ 3.17

แสดงขั้นตอนในการเพิ่มรายการข้อมูลขายโครงการ เริ่มต้นด้วยเจ้าหน้าที่การบัญชีต้องเลือกโครงการที่ฟอร์มจัดการข้อมูลการขายโครงการ หากมีรายการขายในโครงการอยู่แล้วก็จะถูกแสดงขึ้นมา กดปุ่มเพิ่มรายการ ใส่ข้อมูลใหม่ และยืนยันการบันทึกโดยกดปุ่มบันทึกการขาย

2. ยูสเคส Define a down payment

- ชีเควินไออะแกรม Add down payment ดังรูปที่ 3.5
- คอลแลปบอเรชันไออะแกรม Add down payment ดังรูปที่ 3.18

แสดงขั้นตอนในการเพิ่มข้อมูลการผ่อนชำระล่วงหน้า เริ่มต้นด้วยเจ้าหน้าที่การบัญชีต้องเลือกโครงการที่ฟอร์มจัดการข้อมูลการขาย รายการข้อมูลการขายที่มีการบันทึกไปแล้วจะถูกแสดงขึ้นมา ผู้ใช้กดปุ่มกำหนดการผ่อนชำระ จะแสดงฟอร์มจัดการข้อมูลการผ่อนชำระ หากมีรายการข้อมูลการผ่อนชำระอยู่แล้วก็จะแสดงขึ้น กดปุ่มเพิ่มรายการ ใส่ข้อมูลใหม่ และยืนยันการบันทึกโดยกดปุ่มบันทึกการขาย

3. ยูสเคส Maintain project's house numbers

- ชีเควินไออะแกรม Add new project's house numbers ดังรูปที่ 3.6
- คอลแลปบอเรชันไออะแกรม Add new project's house numbers ดังรูปที่ 3.19

แสดงขั้นตอนในการเพิ่มข้อมูลบ้านเลขที่โครงการ เริ่มต้นด้วยเจ้าหน้าที่การบัญชีต้องเลือกโครงการที่ฟอร์ม หากมีรายการข้อมูลอยู่แล้วก็จะแสดงขึ้นขึ้น กดปุ่มเพิ่มรายการ ใส่ข้อมูลใหม่ และยืนยันการบันทึกโดยกดปุ่มบันทึกการขาย

4. ยูสเคส Initial project 's debtor code

- ซีเควินไออะแกรม Initial debtor code ดังรูปที่ 3.7
- คอลเลบบอเลชั่นไออะแกรม Initial debtor code ดังรูปที่ 3.20

แสดงขั้นตอนในการตั้งหนี้โครงการ การตั้งหนี้โครงการจะทำงานผ่านฟอร์มจัดการข้อมูลบ้านเลขที่โครงการ เจ้าหน้าที่การบัญชีต้องเลือกโครงการ หากมีรายการข้อมูลบ้านเลขที่อยู่แล้วก็จะแสดงขึ้นมา เจ้าหน้าที่ยืนยันการตั้งหนี้โดยกดปุ่มตั้งหนี้โครงการ จะมีการสร้างข้อมูลหน่วยอาศัยก่อน แล้วจึงค่อยตั้งหนี้ หรือเริ่มต้นสร้างรหัสลูกหนี้

5. ยูสเคส Maintain Model

- ซีเควินไออะแกรม Add new model ดังรูปที่ 3.8
- คอลเลบบอเลชั่นไออะแกรม Add new model ดังรูปที่ 3.21

แสดงขั้นตอนในการเพิ่มรายการข้อมูลโมเดล เริ่มต้นด้วยเจ้าหน้าที่การบัญชีกดปุ่มเพิ่มรายการ ให้รหัสโมเดล ใส่รายละเอียดของโมเดล กำหนดประเภทสิ่งปลูกสร้างให้กับโมเดลนี้ และสุดท้ายยืนยันการบันทึกโดยกดปุ่มบันทึกรายการ

- ซีเควินไออะแกรม Add new component ดังรูปที่ 3.9
- คอลเลบบอเลชั่นไออะแกรม Add new component ดังรูปที่ 3.22

แสดงขั้นตอนในการเพิ่มรายการข้อมูล โมเดล เริ่มต้นด้วยเจ้าหน้าที่การบัญชีให้รหัสองค์ประกอบ ถ้าเป็นข้อมูลใหม่ก็จะสามารถกำหนดรายละเอียดขององค์ประกอบ และสุดท้ายยืนยันการบันทึกรายการ

- ซีเควินไออะแกรม Define component item for model ดังรูปที่ 3.10
- คอลเลบบอเลชั่นไออะแกรม Define component item for model ดังรูปที่ 3.23

แสดงขั้นตอนในการกำหนดองค์ประกอบในโมเดล เริ่มต้นด้วยเจ้าหน้าที่การบัญชีกดปุ่มกำหนดองค์ประกอบในโมเดลของรายการโมเดลที่สนใจ ฟอร์มจัดการองค์ประกอบในโมเดลจะถูกเปิด พร้อมแสดงรายการที่มีการกำหนดไปแล้ว ผู้ใช้กดปุ่มเพิ่มรายการ แล้วกำหนดถึงองค์ประกอบที่จะมีในโมเดลดังกล่าว และสุดท้ายยืนยันการบันทึกโดยกดปุ่มบันทึกรายการ

- ซีเควินไออะแกรม Add new building type ดังรูปที่ 3.11
- คอลเลบบอเลชั่นไออะแกรม Add new building type ดังรูปที่ 3.24

แสดงขั้นตอนในการเพิ่มรหัสประเภทสิ่งปลูกสร้างใหม่ เริ่มต้นด้วยเจ้าหน้าที่การบัญชี กดปุ่มเพิ่มรายการ แล้วให้รหัสประเภทสิ่งปลูกสร้าง ใส่ชื่อประเภท และสุดท้ายยืนยันการบันทึกโดยกดปุ่มบันทึกรายการ

6. ยูสเคส Maintain Project

- ซีเควินไดอะแกรม Add new project ดังรูปที่ 3.12
- คอลแลบบอเรชันไดอะแกรม Add new project ดังรูปที่ 3.25

แสดงขั้นตอนในการเพิ่มข้อมูลโครงการใหม่ เริ่มต้นด้วยเจ้าหน้าที่การบัญชีกดปุ่มเพิ่มรายการ แล้วกำหนดรหัสโครงการ ถ้าเป็นข้อมูลใหม่ก็จะสามารถใส่รายละเอียดเกี่ยวกับโครงการนี้ และสุดท้ายยืนยันการบันทึกโดยกดปุ่มบันทึก

- ซีเควินไดอะแกรม Add Location information ดังรูปที่ 3.13
- คอลแลบบอเรชันไดอะแกรม Add Location information ดังรูปที่ 3.26

แสดงขั้นตอนในการเพิ่มข้อมูลสำนักงานเกษะชุมชนใหม่ เริ่มต้นด้วยเจ้าหน้าที่การบัญชีกดปุ่มเพิ่มรายการ แล้วกำหนดรหัสสำนักงานเกษะชุมชน ถ้าเป็นข้อมูลใหม่ก็จะสามารถใส่รายละเอียดเกี่ยวกับสำนักงานนี้ และสุดท้ายยืนยันการบันทึกโดยการกดปุ่มบันทึกรายการ

7. ยูสเคส Create project's quotations report

- ซีเควินไดอะแกรม Create project's quotations report ดังรูปที่ 3.14
- คอลแลบบอเรชันไดอะแกรม Create project's quotations report ดังรูปที่ 3.27

แสดงขั้นตอนในการออกรายงานข้อมูลการขายโครงการ เริ่มต้นด้วยเจ้าหน้าที่การบัญชีเลือกโครงการที่ต้องการ แล้วจะมีการเตรียมข้อมูลแสดงผลรายงานมาที่ฟอร์มก่อน เจ้าหน้าที่ยืนยันการออกรายงานโดยกดปุ่มรายงานเพื่อที่จะพิมพ์ คำสั่งนี้จะทำการรันรายงาน

8. ยูสเคส Create building's remainder record report

- ซีเควินไดอะแกรม Create building's remainder record report ดังรูปที่ 3.15
- คอลแลบบอเรชันไดอะแกรม Create building's remainder record report ดังรูปที่ 3.28

แสดงขั้นตอนในการออกรายงานทะเบียนอาคารคงเหลือที่ยังไม่มีการตั้งหนี้ เริ่มต้นด้วยเจ้าหน้าที่การบัญชีเลือกโครงการที่ต้องการ แล้วจะมีการเตรียมข้อมูลแสดงผลรายงานมา

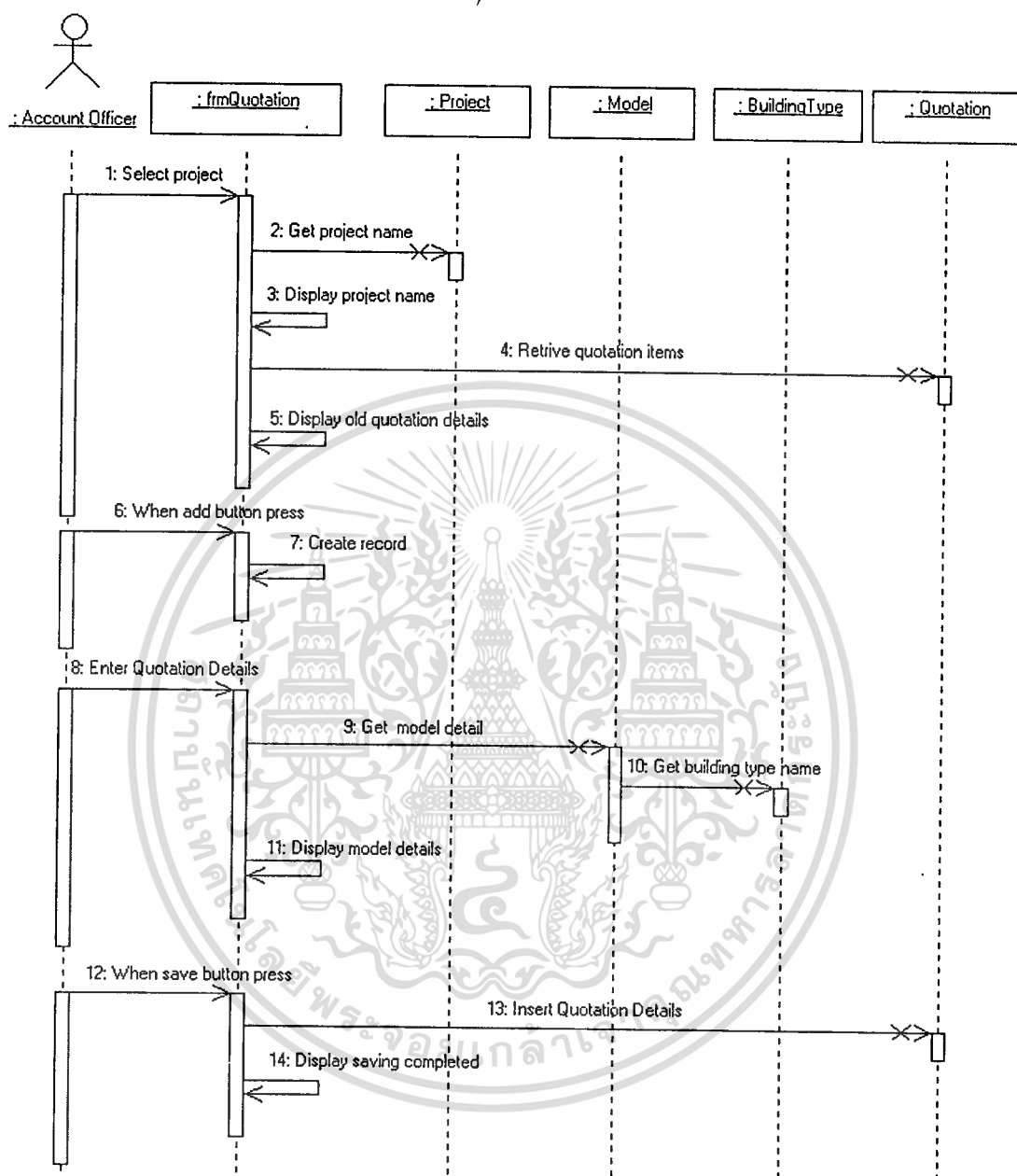
ที่ฟอร์มก่อน เจ้าหน้าที่ยื่นรับการออกรายงานโดยกดปุ่มรายงานเพื่อที่จะพิมพ์ คำสั่งนี้จะทำการรันรายงาน

9. ยูสเคส Create debtor code list report

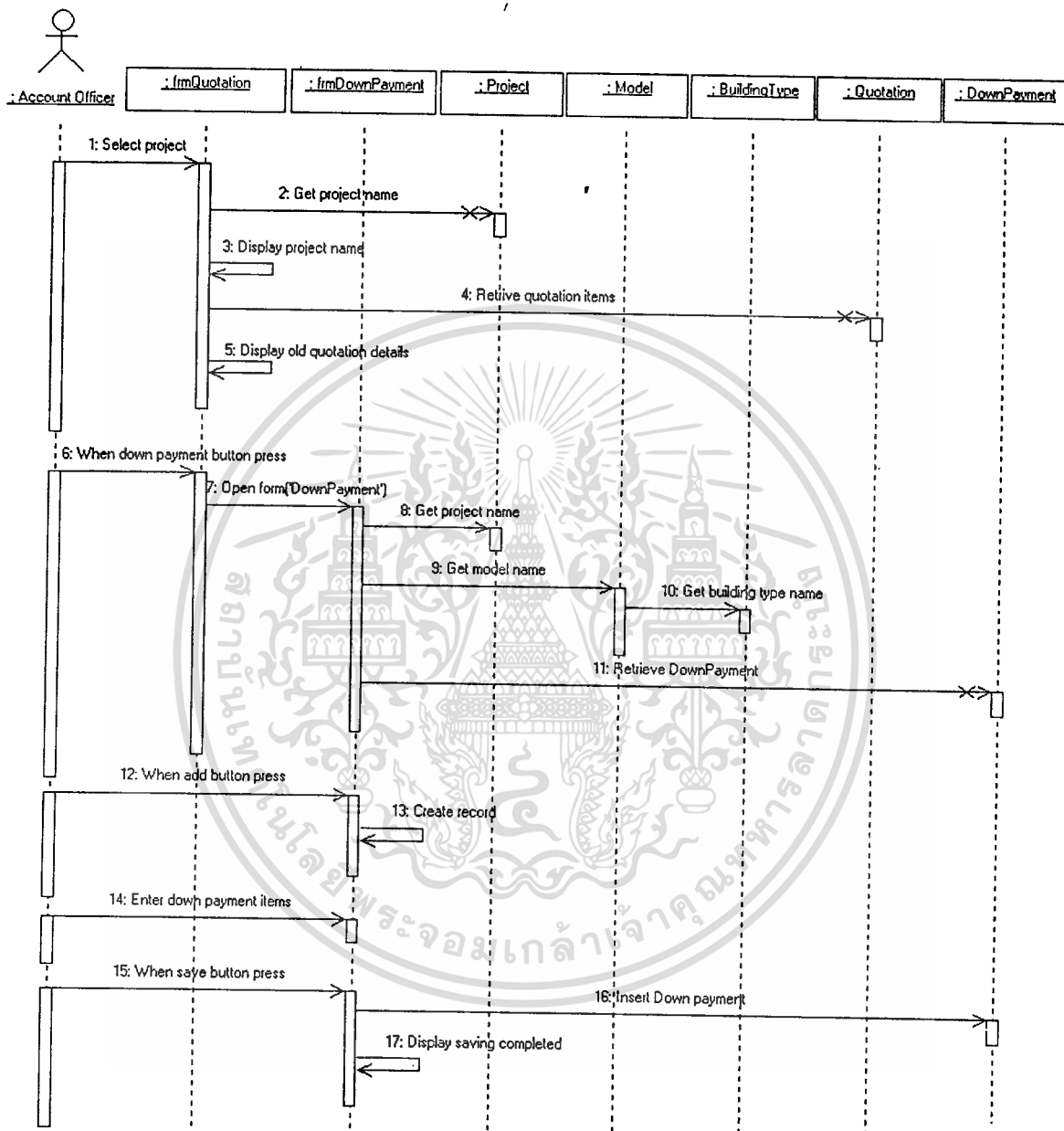
- ซีเควินไดอะแกรม Create debtor code list report ดังรูปที่ 3.16
- คอลเลบบอเลชันไดอะแกรม Create debtor code list report ดังรูปที่ 3.29

แสดงขั้นตอนในการออกรายงานทะเบียนรหัสลูกหนี้ เริ่มต้นด้วยเจ้าหน้าที่การบัญชีเลือกโครงการที่ต้องการ แล้วจะมีการเตรียมข้อมูลแสดงผลรายงานมาที่ฟอร์มก่อน เจ้าหน้าที่ยื่นรับการออกรายงานโดยกดปุ่มรายงานเพื่อที่จะพิมพ์ คำสั่งนี้จะทำการรันรายงาน

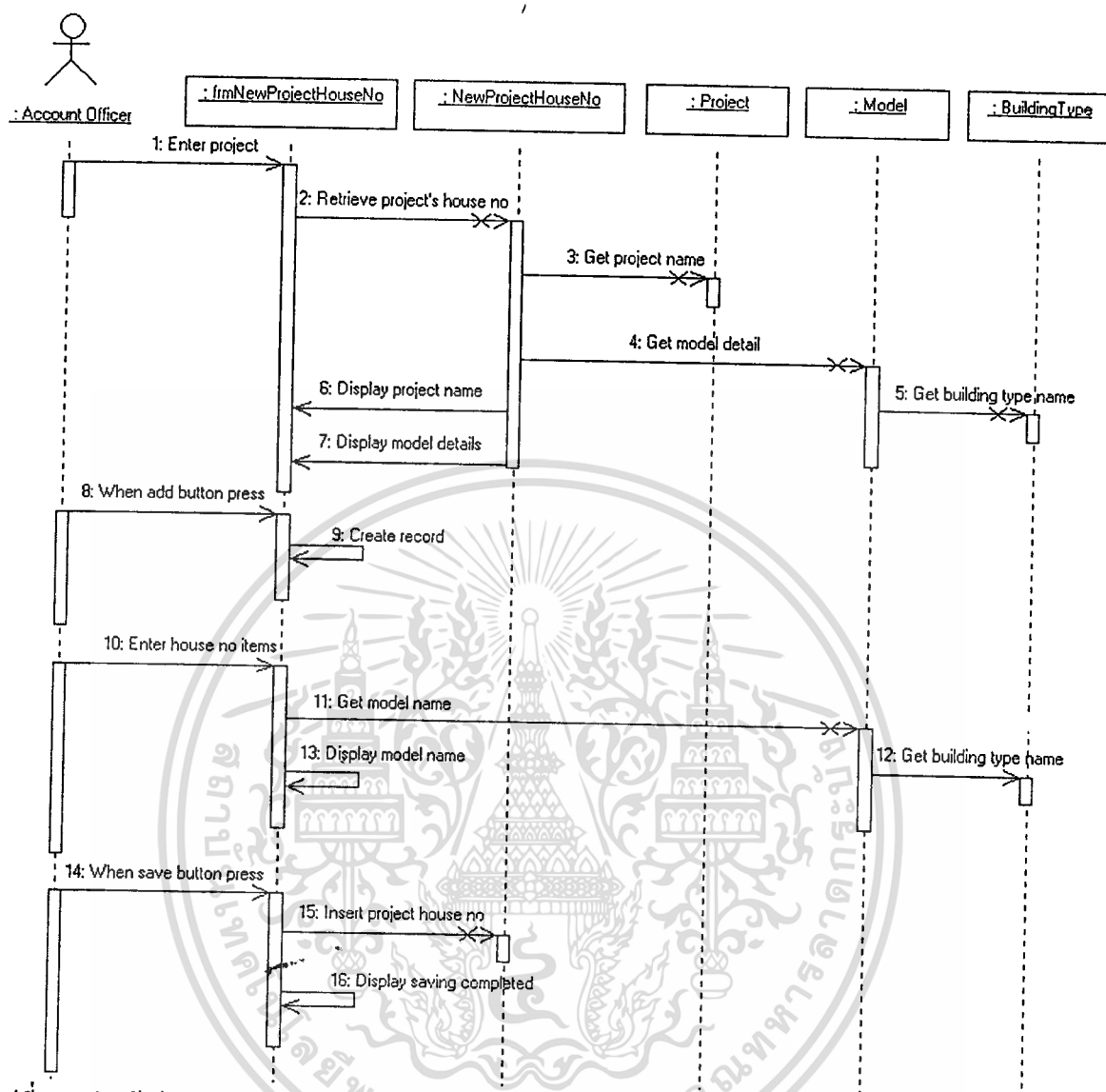
อินเทอร์เฟซไดอะแกรมเหล่านี้ช่วยในการวิเคราะห์คลาส ทำให้ทราบว่าแต่ละคลาสที่เข้ามา ร่วมการทำงานในเหตุการณ์ต่างที่กล่าวมาแล้ว ควรจะมีข้อมูลหรือแอทริบิวต์อะไร และนำไปใช้อย่างไร กล่าวคือ จะต้องมีการดำเนินการอะไรบ้าง ซึ่งยูสเคสที่วิเคราะห์ได้ จะทำให้ทราบว่าจะมีอินเทอร์เฟซ ไดอะแกรมอะไรบ้าง แต่ว่าอินเทอร์เฟซที่นำเสนอไปนี้เป็นส่วนหนึ่งของแต่ละยูสเคสเท่านั้น แต่สามารถคาดการณ์การดำเนินการของคลาสต่อไปได้ เช่นสำหรับคลาส Project พิจารณายูสเคส Mantain project อินเทอร์เฟซที่นำเสนอมีเพียง ซีเควินไดอะแกรม และคอลเลบบอเลชันไดอะแกรม Add new project เท่านั้น การดำเนินการที่ได้ คือ insertProject (จากคลาสไดแกรม บทที่ 4) ดังนั้นก็สามารถคาดเดาได้ว่า หากต้องการลบ หรือแก้ไขข้อมูลโครงการ การดำเนินการก็น่าจะเป็น deleteProject และ updateProject ตามลำดับ



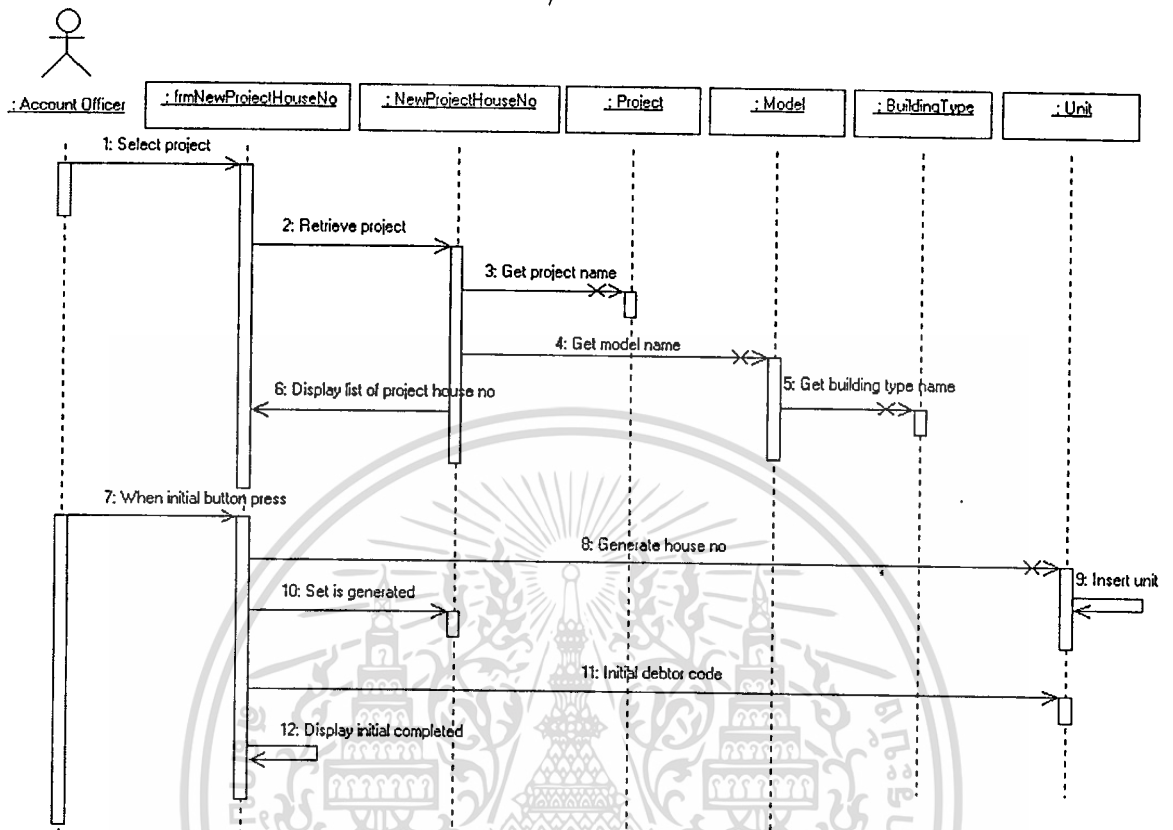
รูปที่ 3.4 ซีเควิน ไลอะแกรม Add project's quotation



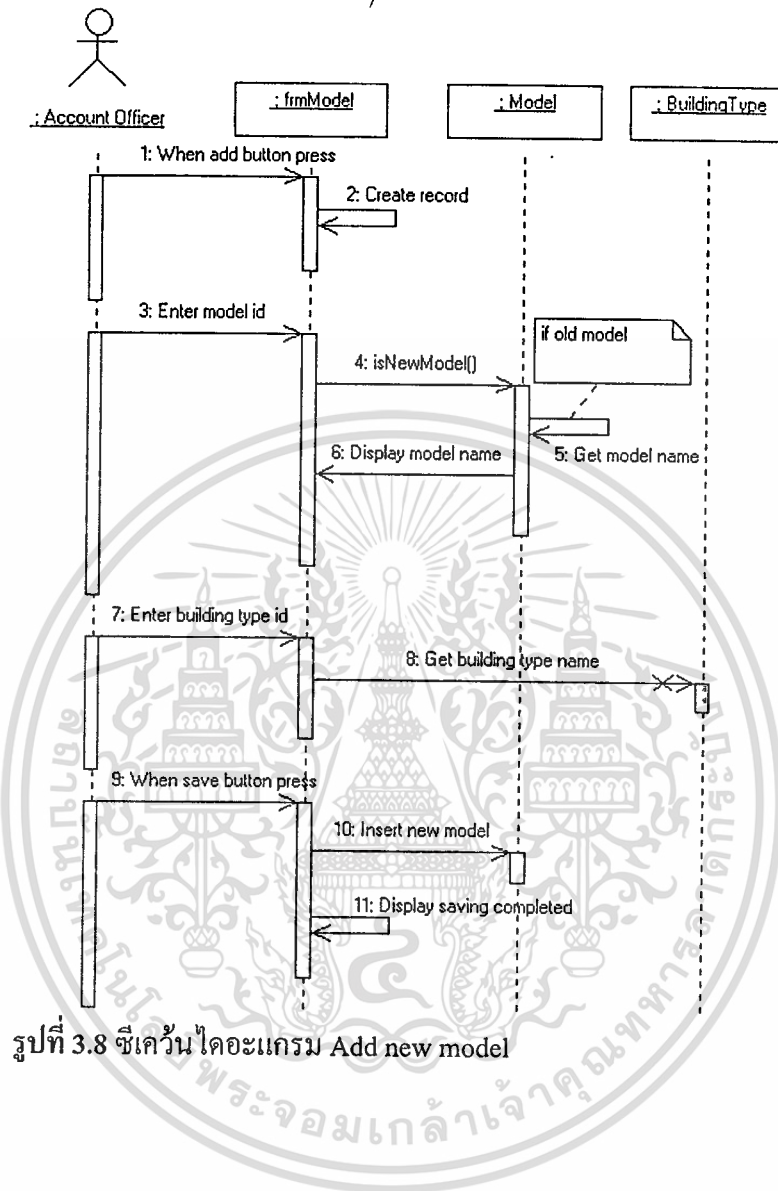
รูปที่ 3.5 ซีเควินไดอะแกรม Add down payment



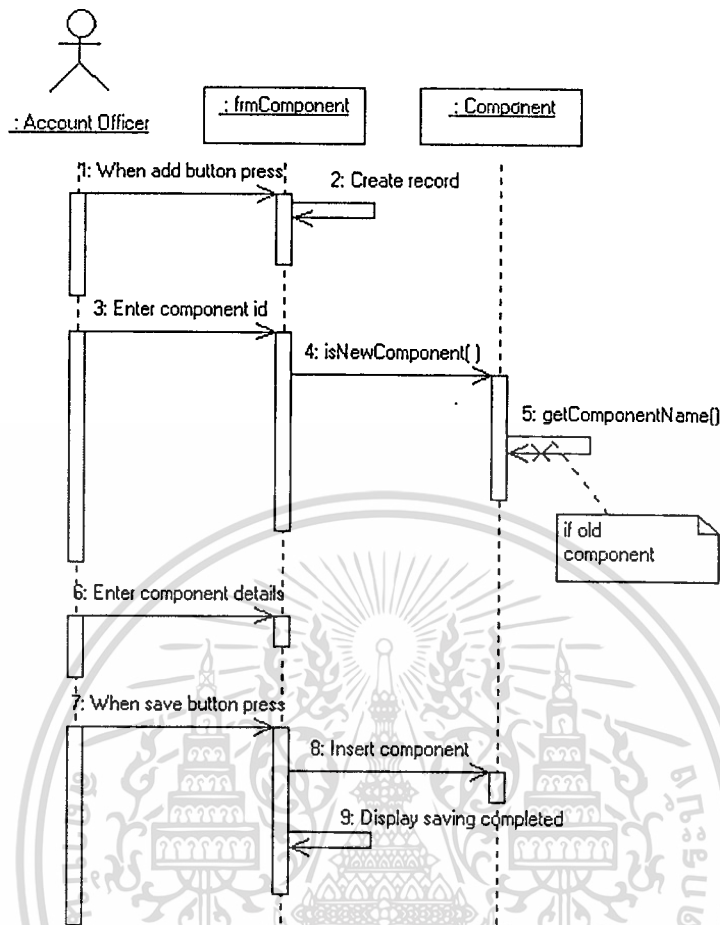
รูปที่ 3.6 ซีเควินโค้ดแแกรม Add new project's house numbers



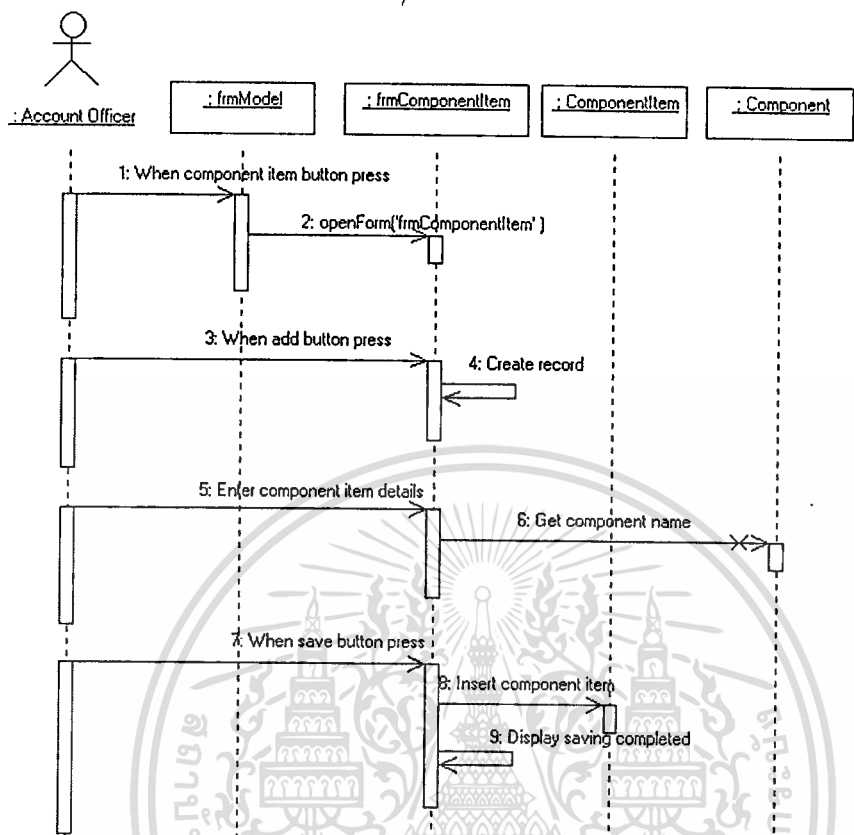
รูปที่ 3.7 ซีควีนโคดแแกรม Initial debtor code



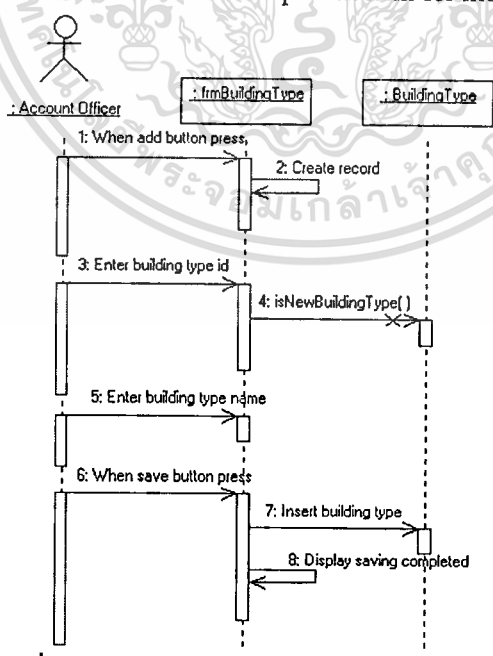
รูปที่ 3.8 ซีเค้นไต่อะแกรม Add new model



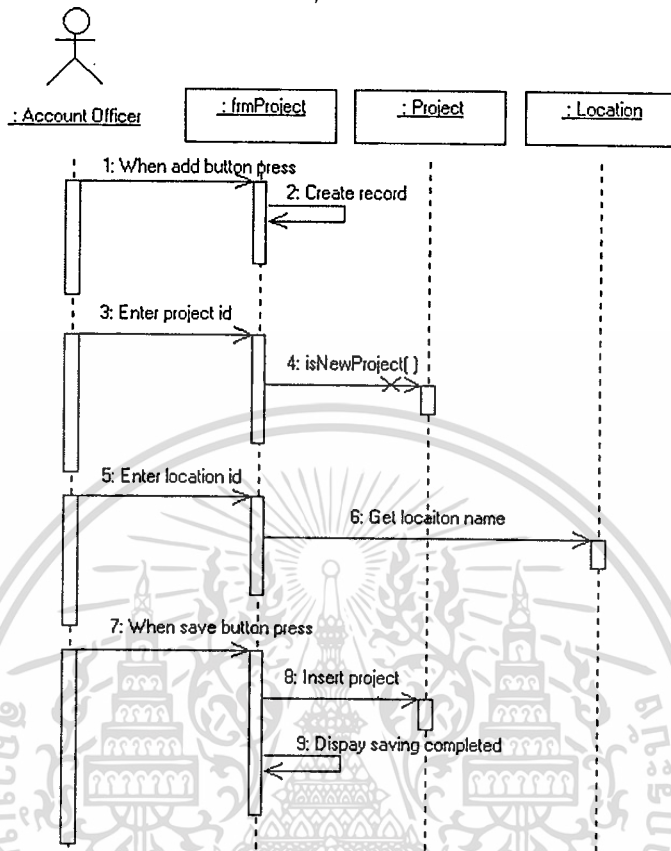
รูปที่ 3.9 ซีควีนซ์ไดอะแกรม Add new component



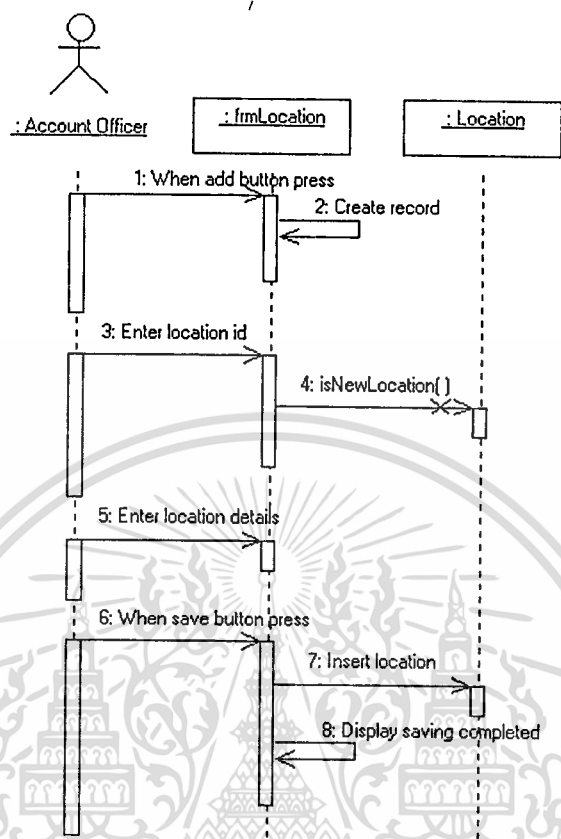
รูปที่ 3.10 ซีควีนโคแอม Define component item for model



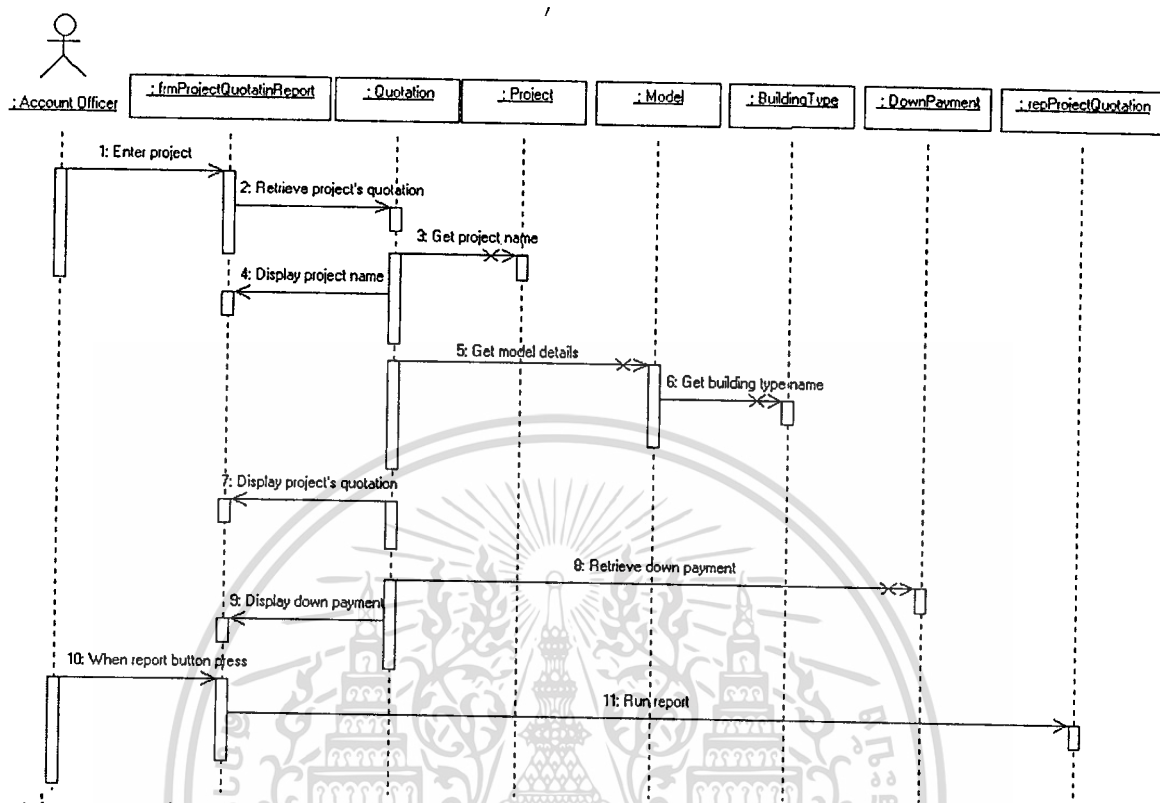
รูปที่ 3.11 ซีควีนโคแอม Add new building type



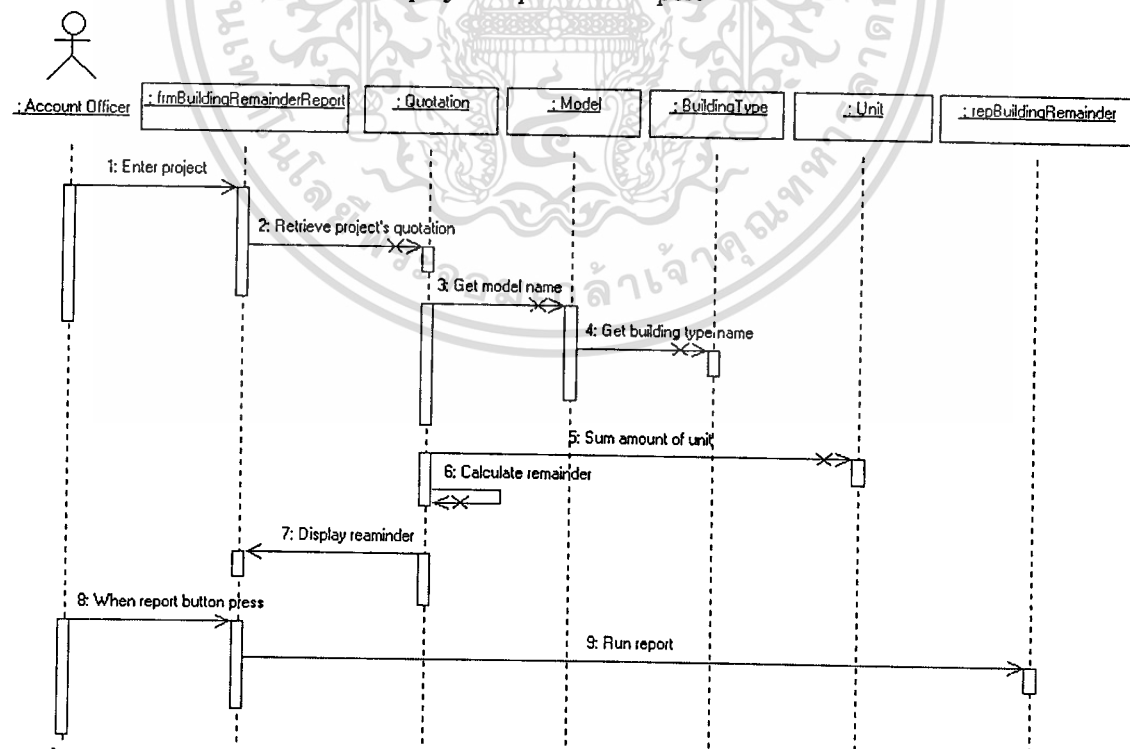
รูปที่ 3.12 ซีเคว้น โค้ดโปรแกรม Add new project



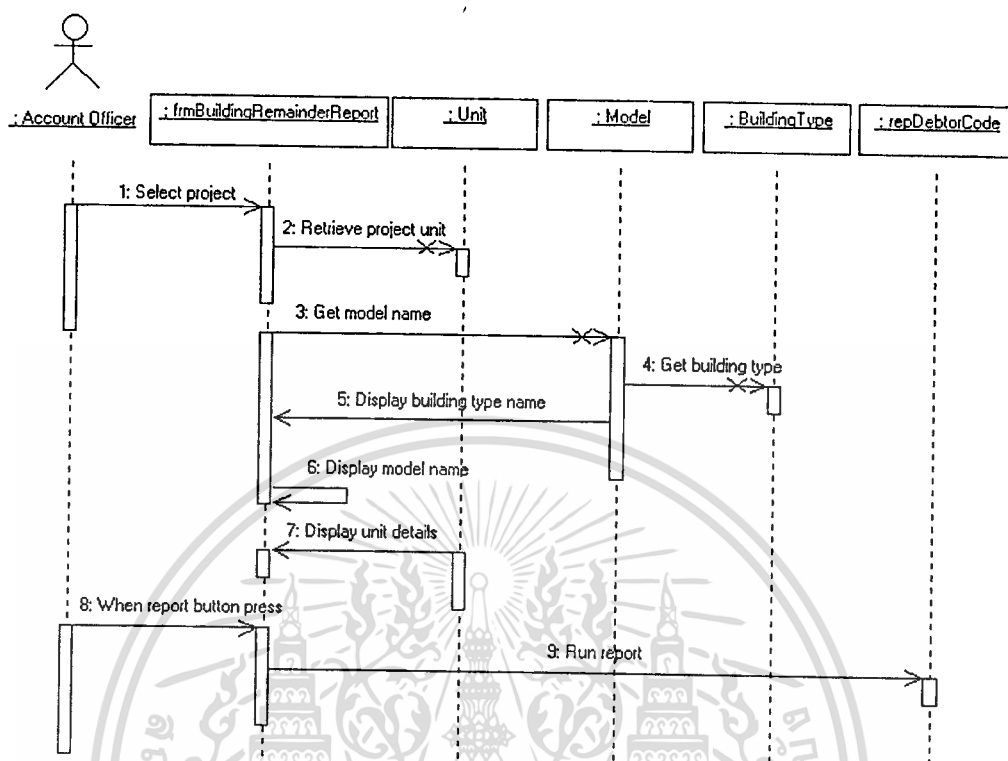
รูปที่ 3.13 ซีควีนไคอะแกรม Add Location information



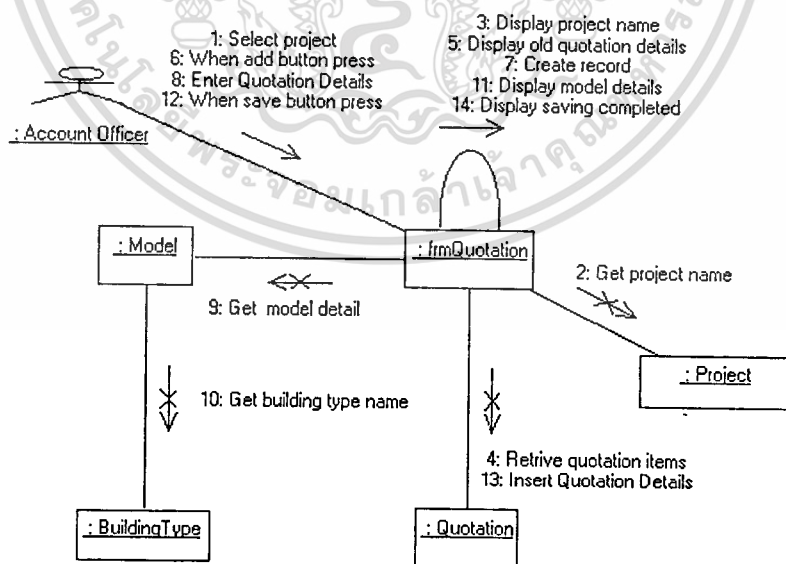
รูปที่ 3.14 ซีเควินไดอะแกรม Create project's quotations report



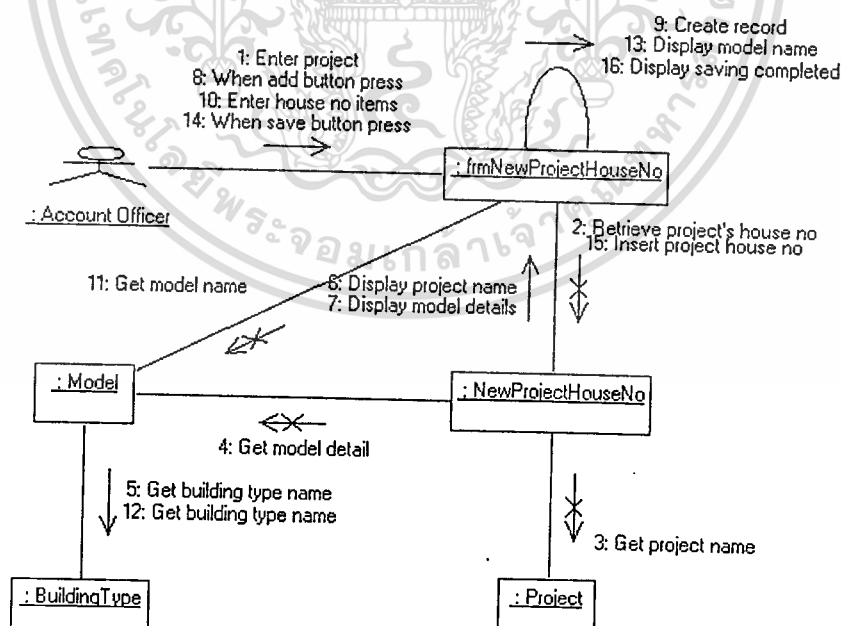
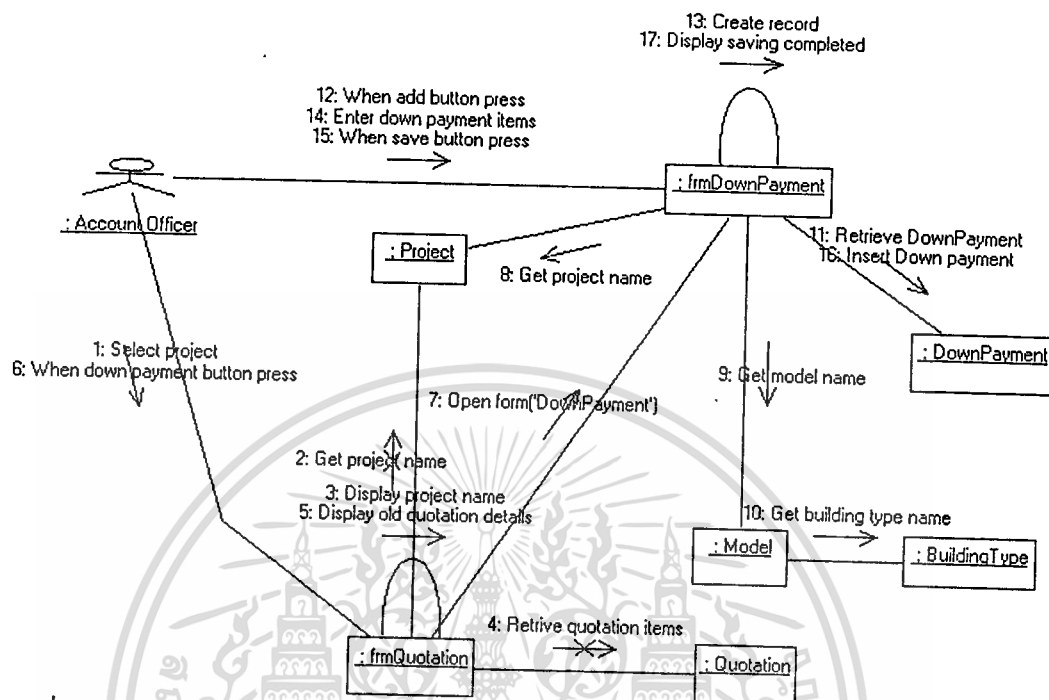
รูปที่ 3.15 ซีเควินไดอะแกรม Create building's remainder record report

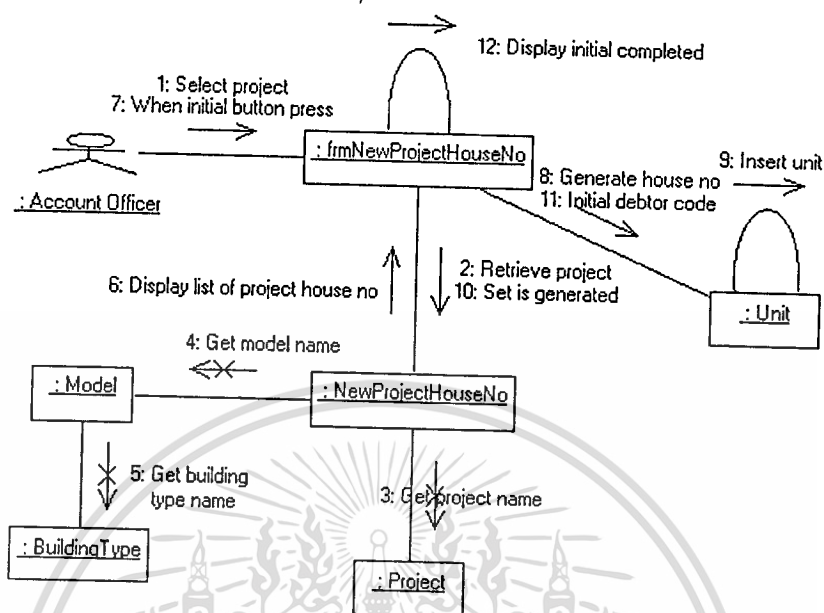


รูปที่ 3.16 ซีควีนไคอะแกรม Create debtor code list report

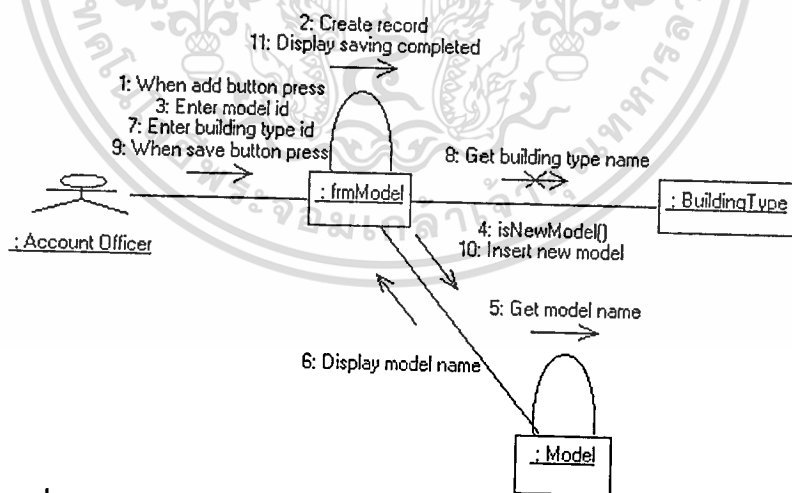


รูปที่ 3.17 คอลแลบบอเรชันไคอะแกรม Add project's quotation

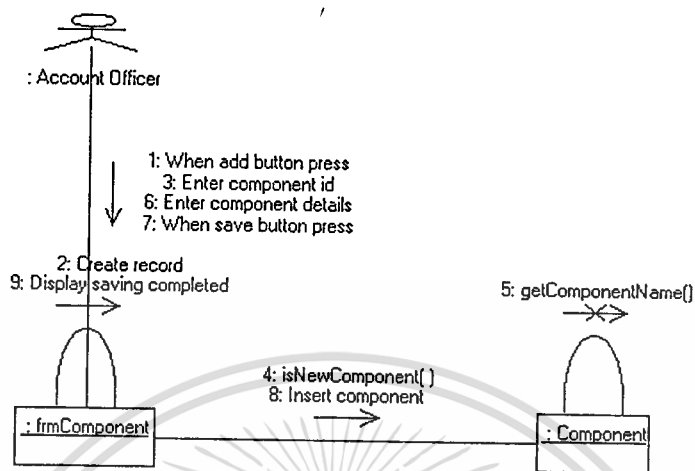




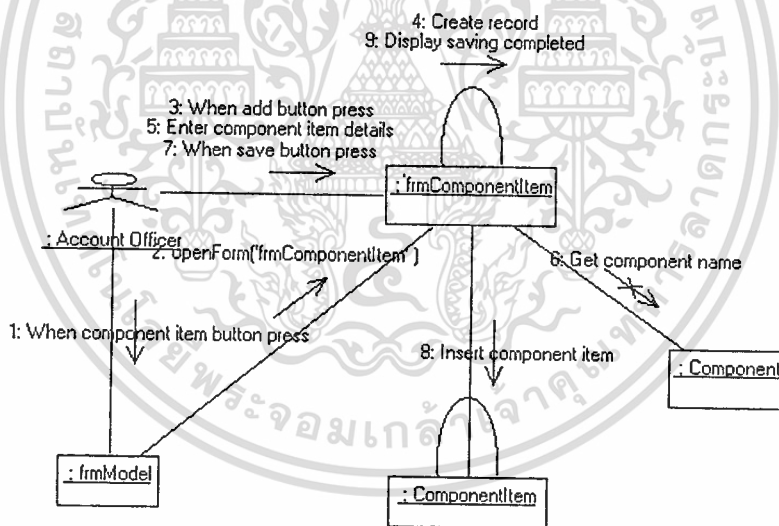
รูปที่ 3.20 คอลเลบบอเลชั่นไคอะแกรม Initial debtor code



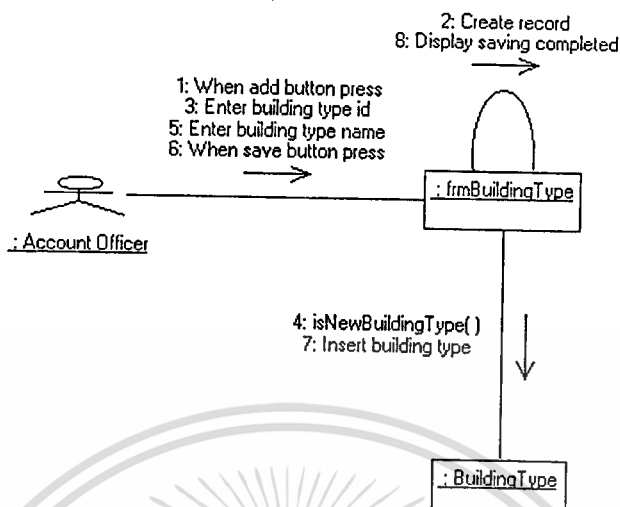
รูปที่ 3.21 คอลเลบบอเลชั่นไคอะแกรม Add new model



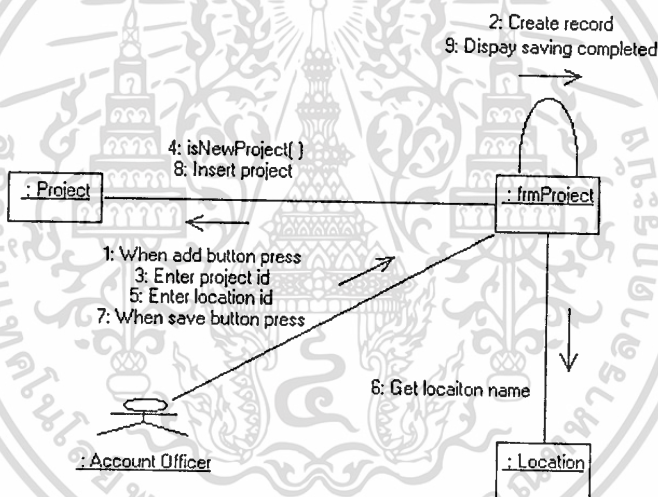
รูปที่ 3.22 คอลเลบบอเลชั่นไดอะแกรม Add new component



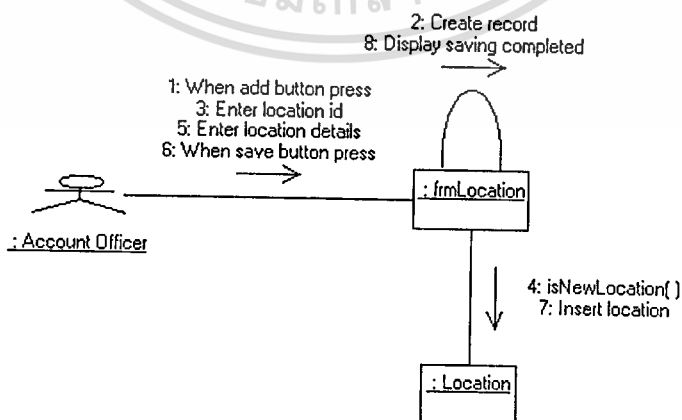
รูปที่ 3.23 คอลเลบบอเลชั่นไดอะแกรม Define component item for model



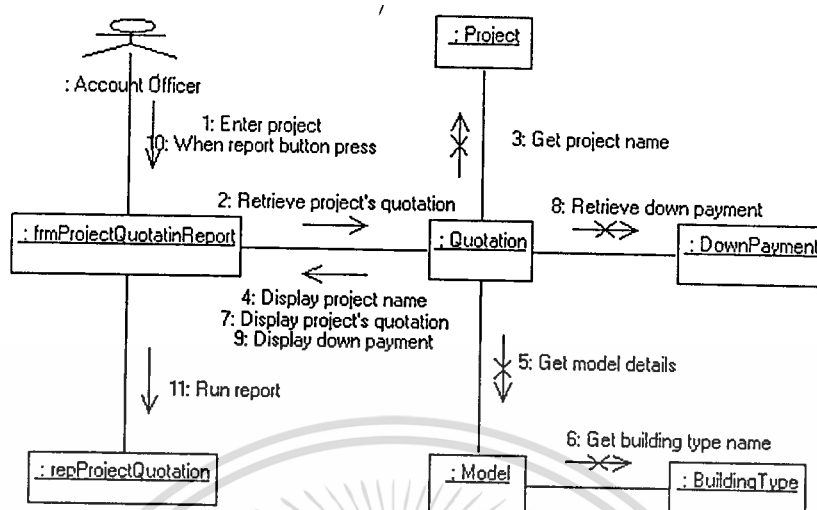
รูปที่ 3.24 คอลเลบบอเลชั่นไคอะแกรม Add new building type



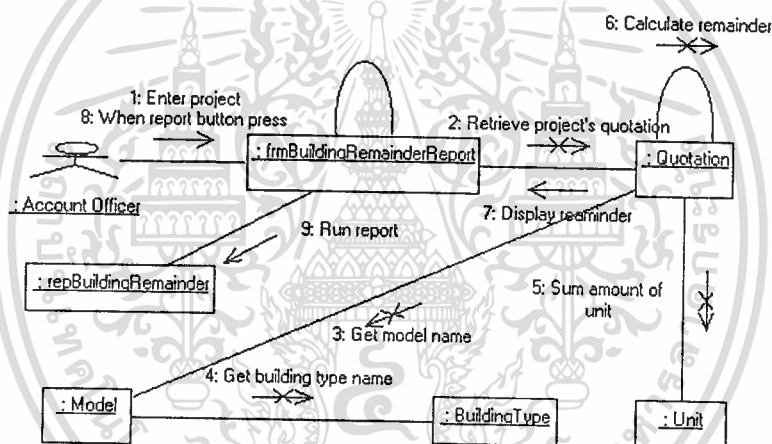
รูปที่ 3.25 คอลเลบบอเลชั่นไคอะแกรม Add new project



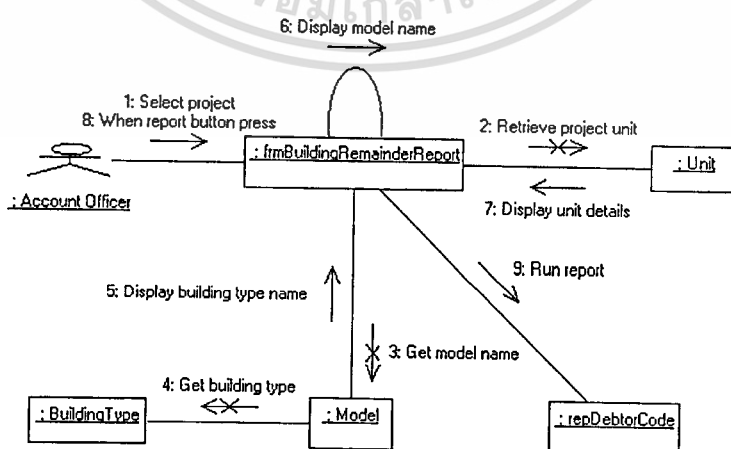
รูปที่ 3.26 คอลเลบบอเลชั่นไคอะแกรม Add Location information



รูปที่ 3.27 คอลแลบบอเลชั่น โค้ดแแกรม Create project's quotations report



รูปที่ 3.28 คอลแลบบอเลชั่น โค้ดแแกรม Create building's remainder record report

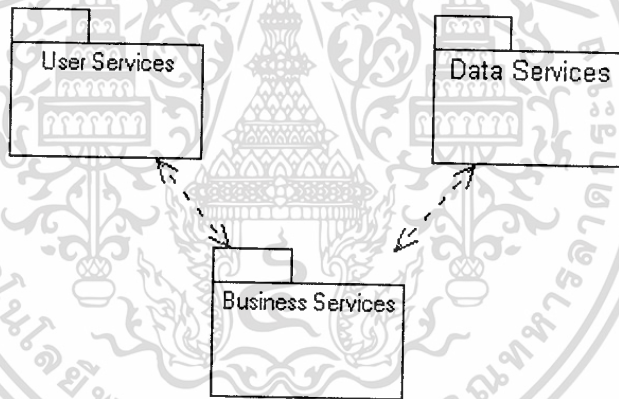


รูปที่ 3.29 คอลแลบบอเลชั่น โค้ดแแกรม Create debtor code list report

บทที่ 4

การออกแบบระบบ

การออกแบบระบบตั้งนี้โครงการบัญชีลูกหนี้ของการเคหะแห่งชาติ ได้แก่การปรับปรุง คลาส กำหนดแอทริบิวต์และการดำเนินการของคลาสที่วิเคราะห์ได้จากบทที่ 3 และปรับปรุงคลาส ไลอะแกรม (รูปที่ 3.3 บทที่ 3) โดยอาศัยยูสเคสในยูสเคสไลอะแกรม ซีเควินไลอะแกรม คอลเลบ บอเลชั่นไลอะแกรม และสเตทชาร์ทไลอะแกรม ซึ่งคลาสที่ได้คือคลาสที่อยู่ในบิสซิเนสเลเยอร์ ต่อ ไปจึงออกแบบคลาสสำหรับแอคเซสเลเยอร์ และคลาสสำหรับวิวเลเยอร์ โดยแยกเก็บต่างแพ็คเกจ ได้แก่ แพ็คเกจ Business services แพ็คเกจ User services และแพ็คเกจ Data services ตามลำดับดัง รูปที่ 4.1

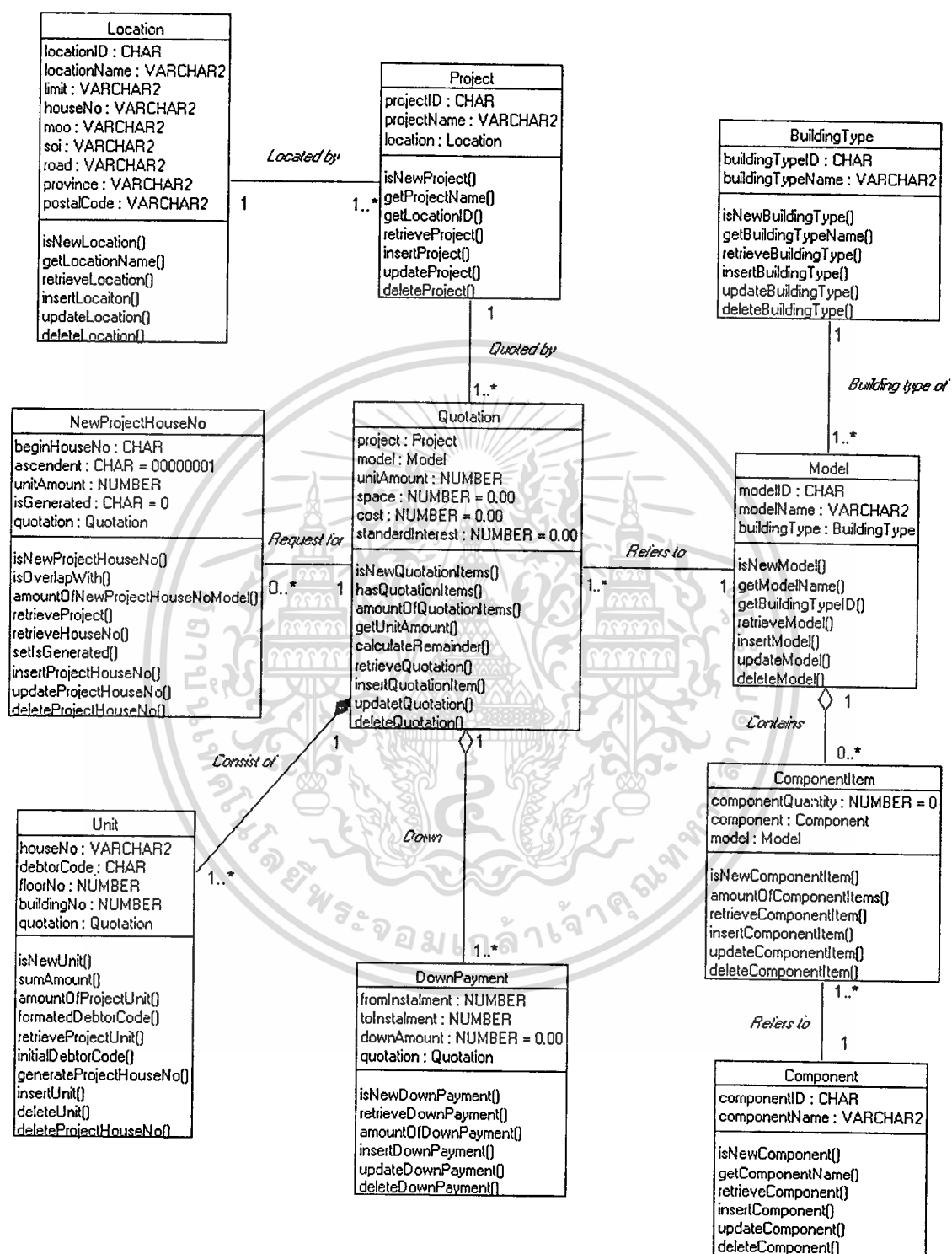


รูปที่ 4.1 แพ็คเกจสำหรับเลเยอร์ในระบบตั้งนี้โครงการบัญชีลูกหนี้ของการเคหะแห่งชาติ

คลาสไลอะแกรมของแต่ละเลเยอร์จะได้นำเสนอต่อไปนี้

4.1 คลาสในบิสซิเนสเลเยอร์

คลาสในบิสซิเนสเลเยอร์ได้จากการวิเคราะห์จากบทที่ 3 ซึ่งคลาสไลอะแกรมที่ปรับ ประแล้วเป็นดังรูปที่ 4.2 ซึ่งแสดงแอทริบิวต์และการดำเนินการของแต่ละคลาส และดังตารางที่ 4.1-4.20 ธิบายแอทริบิวต์ และการดำเนินการของคลาส



รูปที่ 4.2 คลาสไดอะแกรมสำหรับบิสซิเนสเลเยอร์ของระบบ

ตารางที่ 4.1 แอทริบิวต์ของคลาส Location

แอทริบิวต์	ชนิดข้อมูล	คำอธิบาย
locationID	CHAR(4)	รหัสสำนักงานเคหะชุมชน
locatinName	VARCHAR2(50)	ชื่อสำนักงานเคหะชุมชน
limit	VARCHAR2(2)	เขต
houseNo	VARCHAR2(9)	บ้านเลขที่
moo	VARCHAR2(2)	หมู่ที่
soi	VARCHAR2(3)	ซอย
road	VARCHAR2(30)	ถนน
province	VARCHAR2(30)	จังหวัด
postalCode	VARCHAR2(8)	รหัสไปรษณีย์

ตารางที่ 4.2 การดำเนินการของคลาส Location (ส่วนที่ 1/2)

การดำเนินการ	คำอธิบาย
isNewLocation parameter : a_locationID return : BOOLEAN	ตรวจสอบว่ารหัสสำนักงานเคหะชุมชนที่ระบุมีอยู่หรือไม่ หากมีจะให้ค่ากลับเป็น TRUE ถ้าไม่มีจะให้ค่ากลับเป็น FALSE
getLocationName parameter : a_locationID return : locationName	หาชื่อสำนักงานเคหะชุมชน
retrievetLocation parameter : a_locationID return : Location	เข้าถึงข้อมูลสำนักงานเคหะชุมชน
insertLocation parameter : a_locationID,a_locationName, a_limit,a_houseNo, a_moo,a_soi, a_road,a_province,a_postalCode	เพิ่มรายการข้อมูลสำนักงานเคหะชุมชน
updateLocation parameter : a_locationID,a_locatinName, a_limit,a_houseNo, a_moo,a_soi, a_road,a_province,a_postalCode	ปรับปรุงรายการข้อมูลสำนักงานเคหะชุมชนที่มีรหัสสำนักงานเคหะชุมชนตามที่ระบุ

ตารางที่ 4.2 การดำเนินการของคลาส Location (ส่วนที่ 2/2)

การดำเนินการ	คำอธิบาย
deleteLocation parameter : a_locationID,a_locatinName, a_limit,a_houseNo, a_moo,a_soi, a_road,a_province,a_postalCode	ลบข้อมูลสำนักงานเคหะชุมชนที่มีรหัสสำนักงานเคหะชุมชนตามที่ระบุ

ตารางที่ 4.3 แอทริบิวต์สำหรับคลาส Project

แอทริบิวต์	ชนิดข้อมูล	คำอธิบาย
projectID	CHAR(4)	รหัสโครงการ
projectName	VARCHAR2(100)	ชื่อโครงการ
location	Location	ใช้เชื่อมโยงกับคลาส Project

ตารางที่ 4.4 การดำเนินการของคลาส Project (ส่วนที่ 1/2)

การดำเนินการ	คำอธิบาย
isNewProject parameter : a_projectID return : BOOLEAN	ตรวจสอบว่ารหัสโครงการที่ระบุมีอยู่หรือไม่ หากมีจะให้ค่ากลับเป็น TRUE ถ้าไม่มีจะให้ค่ากลับเป็น FALSE
getProjectName parameter : a_projectID return : projectName	หาชื่อสำนักงานเคหะชุมชน
getLocationID parameter : a_projectID return : locationID	หารหัสสำนักงานเคหะชุมชนที่โครงการนั้นสังกัดอยู่
retrieveProject return : Project	เข้าถึงข้อมูลโครงการทั้งหมด
insertProject parameter : a_projectID, a_projectName, a_locationID	เพิ่มรายการโครงการ
updateProject parameter : a_projectID, a_projectName, a_locationID	ปรับปรุงการอ้างอิงข้อมูลสำนักงานเคหะชุมชนที่โครงการนี้อยู่

ตารางที่ 4.4 การดำเนินการของคลาส Project (ส่วนที่ 2/2)

การดำเนินการ	คำอธิบาย
deleteProject parameter : a_projectID	ลบรายการ โครงการ

ตารางที่ 4.5 แอทริบิวต์สำหรับคลาส Model

แอทริบิวต์	ชนิดข้อมูล	คำอธิบาย
modelID	CHAR(2)	รหัสโมเดล
modelName	VARCHAR2(20)	ชื่อโมเดล
buildingType	BuildingType	ใช้เชื่อมโยงกับคลาส BuildingType

ตารางที่ 4.6 การดำเนินการของคลาส Model

การดำเนินการ	คำอธิบาย
isNewModel parameter : a_modelID return : BOOLEAN	ตรวจสอบว่ารหัสโมเดลที่ระบุมีอยู่หรือไม่ หากมีจะให้ค่ากลับเป็น TRUE ถ้าไม่มีจะให้ค่ากลับเป็น FALSE
getModelName parameter : a_modelID return : modelName	หาชื่อโมเดลตามรหัสโมเดล
getBuildingTypeID parameter : a_modelID return : buildingTypeID	หารหัสประเภทสิ่งปลูกสร้างของโมเดลตามรหัสโมเดล
retrieveModel return : Model	เข้าถึงข้อมูล โมเดลทั้งหมด
insertModel parameter : a_modelID, a_modelName, a_buildingTypeID	เพิ่มข้อมูล โมเดล
updateModel Parameter : a_modelID, a_modelName, a_buildingTypeID	แก้ไขข้อมูลชื่อโมเดล และประเภทสิ่งปลูกสร้าง
deleteModel parameter : a_modelID	ลบข้อมูล โมเดล

ตารางที่ 4.7 แอทริบิวต์สำหรับคลาส Building Type

แอทริบิวต์	ชนิดข้อมูล	คำอธิบาย
buildingTypeID	CHAR(2)	รหัสประเภทสิ่งปลูกสร้าง
buildingTypeName	VARCHAR2(50)	ชื่อประเภทสิ่งปลูกสร้าง

ตารางที่ 4.8 การดำเนินการของคลาส Building Type

การดำเนินการ	คำอธิบาย
isNewBuildingType parameter : a_buildingTypeID return : BOOLEAN	ตรวจสอบว่ารหัสประเภทสิ่งปลูกสร้างที่ระบุมีอยู่หรือไม่ หากมีจะให้ค่ากลับเป็น TRUE ถ้าไม่มีจะให้ค่ากลับเป็น FALSE
getBuildingTypeName parameter : a_locationID return : buildingTypeName	หาชื่อประเภทสิ่งปลูกสร้างตามรหัสที่ระบุมา
retrieveBuildingType return : BuildingType	เข้าถึงข้อมูลประเภทสิ่งปลูกสร้างทั้งหมด
insertBuildingType parameter : a_buildingTypeID,a_buildingTypeName	เพิ่มรายการประเภทสิ่งปลูกสร้าง
updateBuildingType parameter : a_buildingTypeID,a_buildingTypeName	ปรับปรุงชื่อประเภทสิ่งปลูกสร้างที่มีรหัสประเภทสิ่งปลูกสร้างที่ระบุ
deleteBuildingType parameter : a_buildingTypeID	ลบรายการประเภทสิ่งปลูกสร้างที่มีรหัสประเภทสิ่งปลูกสร้างที่ระบุ

ตารางที่ 4.9 แอทริบิวต์สำหรับคลาส ComponentItem

แอทริบิวต์	ชนิดข้อมูล	คำอธิบาย
componentAmount	NUMBER(2)	จำนวนองค์ประกอบในโมเดล
component	Component	ใช้เชื่อมโยงกับคลาส Component
model	Model	ใช้เชื่อมโยงกับคลาส Model

ตารางที่ 4.10 การดำเนินการของคลาส ComponentItem

การดำเนินการ	คำอธิบาย
isNewComponentItem parameter : a_modelID,a_ComponentItem return : BOOLEAN	ตรวจสอบว่ารายการรหัสองค์ประกอบของโมเดลที่ระบุมีอยู่หรือไม่ หากมีจะให้ค่ากลับเป็น TRUE ถ้าไม่มีจะให้ค่ากลับเป็น FALSE
amountOfComponentItems parameter : a_modelID return : NUMBER	คำนวณจำนวนรายการองค์ประกอบซึ่งมีรหัสโมเดลตามที่ระบุ
retrieveComponentItem parameter : a_modelID return : ComponentItem	เข้าถึงข้อมูลองค์ประกอบในโมเดลที่ระบุ
InsertComponentItem parameter : a_modelID,a_ComponentItem, a_componentAmount	เพิ่มรายการองค์ประกอบใน โมเดล
updateComponentItem parameter : a_modelID,a_ComponentItem, a_componentAmount	ปรับปรุงจำนวนองค์ประกอบใน โมเดลที่มีรหัสองค์ประกอบ และโมเดลตามที่ระบุ
deleteComponentItem parameter : a_modelID,a_ComponentItem	ลบรายการรายการองค์ประกอบใน โมเดลที่มีรหัสองค์ประกอบ และโมเดลตามที่ระบุ

ตารางที่ 4.11 แอทริบิวต์สำหรับคลาส Component

แอทริบิวต์	ชนิดข้อมูล	คำอธิบาย
componentID	CHAR(2)	รหัสองค์ประกอบ
componentName	VARCHAR2(50)	ชื่อองค์ประกอบ

ตารางที่ 4.12 การดำเนินการของคลาส Component (ส่วนที่ 1/2)

การดำเนินการ	คำอธิบาย
isNewComponent parameter : a_ComponentID return : BOOLEAN	ตรวจสอบว่ารหัสองค์ประกอบที่ระบุมีอยู่หรือไม่ หากมีจะให้ค่ากลับเป็น TRUE ถ้าไม่มีจะให้ค่ากลับเป็น FALSE
getComponentName parameter : a_coponentID return : componentName	หาชื่อองค์ประกอบตามรหัสองค์ประกอบที่ระบุ

ตารางที่ 4.12 การดำเนินการของคลาส Component (ส่วนที่ 2/2)

การดำเนินการ	คำอธิบาย
retrieveComponent return : Component	เข้าถึงข้อมูลองค์ประกอบทั้งหมด
InsertComponent parameter : a_ComponentID, a_componentName	เพิ่มรายการองค์ประกอบ
updateComponent parameter : a_ComponentID, a_componentName	ปรับปรุงชื่อองค์ประกอบที่มีรหัสองค์ประกอบตามที่ระบุ
deleteComponent parameter : a_ComponentID	ลบรายการรายการองค์ประกอบที่มีรหัสองค์ประกอบตามที่ระบุ

ตารางที่ 4.13 แอทริบิวต์สำหรับคลาส Quotation

แอทริบิวต์	ชนิดข้อมูล	คำอธิบาย
project	Project	ใช้เชื่อมโยงกับคลาส Project
model	Model	ใช้เชื่อมโยงกับคลาส Model
unitAmount = 0	NUMBER(3)	จำนวนหน่วยอาสัย
space = 0.00	NUMBER(5,2)	เนื้อที่ต่อหน่วย (ตร.ว)
cost = 0.00	NUMBER(11,2)	ราคาขายเงินสดต่อหน่วย (บาท)
standardInterest	NUMBER(4,2)	อัตราดอกเบี้ยการผ่อนชำระค่าเช่าซื้อต่อปี (%)

ตารางที่ 4.14 การดำเนินการของคลาส Quotation (ส่วนที่ 1/2)

การดำเนินการ	คำอธิบาย
isNewQuotationItems parameter : a_projectID,a_modelID return : BOOLEAN	ตรวจสอบว่ามีรายการขายที่เป็นโมเดลและของโครงการที่ระบุหรือไม่ ถ้ามีจะให้ค่ากลับเป็น TRUE ถ้าไม่มีจะให้ค่ากลับเป็น FALSE
hasQuotationItems parameter : a_projectID return : BOOLEAN	ตรวจสอบว่ามีรายการขายของโครงการที่ระบุหรือไม่ ถ้ามีจะให้ค่ากลับเป็น TRUE ถ้าไม่มีจะให้ค่ากลับเป็น FALSE
amountOfQuotationItems parameter : a_projectID return : NUMBER	คำนวณหาจำนวนรายการขายของโครงการที่ระบุ

ตารางที่ 4.14 การดำเนินการของคลาส Quotation (ส่วนที่ 2/2)

การดำเนินการ	คำอธิบาย
getUnitAmount parameter : a_projectID,a_modelID return : NUMBER	หาจำนวนหน่วยอาศัยของ โมเดล และ โครงการที่ระบุ
calculateRemainder parameter : a_projectID,a_modelID return : NUMBER	คำนวณจำนวนหน่วยอาศัยของ โมเดล และ โครงการที่ระบุ ซึ่งยังไม่มี การตั้งหนี้
retrieveQuotation parameter : a_projectID return : Quotation	เข้าถึงข้อมูลขายของโครงการที่ระบุ
insertQuotationItem parameter : a_projectID, a_modelID,an_UnitAmount, a_space, a_cost, a_standardInterest	เพิ่มรายการขาย
updateQuotationItem parameter : a_projectID, a_modelID, an_UnitAmount, a_space, a_cost, a_standardInterest	ปรับปรุงข้อมูล จำนวนหน่วยอาศัย เนื้อที่ต่อ หน่วย ราคาขายเงินสด และอัตราดอกเบี้ยการ ผ่อนชำระค่าเช่าซื้อต่อปี ของโมเดลในโครงการที่ระบุ
deleteQuotationItem parameter : a_projectID, a_modelID	ลบรายการขายที่มีโมเดล และ โครงการตามที่ ระบุ

ตารางที่ 4.15 แอทริบิวต์สำหรับคลาส DownPayment

แอทริบิวต์	ชนิดข้อมูล	คำอธิบาย
fromInstalment	NUMBER(2)	จากงวดที่
toInstalment	NUMBER(2)	ถึงงวดที่
downAmount = 0.00	NUMBER(8,2)	ค่าผ่อนชำระเงินล่วงหน้าต่องวด (บาท)
quotation	Quotation	ใช้เชื่อมโยงกับคลาส Quotation

ตารางที่ 4.16 การดำเนินการของคลาส DownPayment

การดำเนินการ	คำอธิบาย
isNewDownPayment parameter : a_projectID, a_modelID, a_fromInstalment return : BOOLEAN	ตรวจสอบว่ามีรายการค่าผ่อนชำระล่วงหน้าจากงวดที่ถึงงวดที่ของโมเดล และโครงการที่ระบุหรือไม่ ถ้ามีจะให้ค่ากลับเป็น TRUE ถ้าไม่มีจะให้ค่ากลับเป็น FALSE
amountOfDownPayment parameter : a_projectID, a_modelID return : NUMBER	คำนวณหาจำนวนรายการการผ่อนชำระล่วงหน้าของโมเดลและโครงการที่ระบุ
retrieveDownPayment parameter : a_projectID, a_modelID return : DownPayment	เข้าถึงรายการผ่อนชำระล่วงหน้าของ โมเดล และโครงการที่ระบุ
insertDownPayment parameter : a_projectID, a_modelID, a_fromInstalment, a_toInstalment, a_downAmount	เพิ่มรายการค่าผ่อนชำระล่วงหน้าจากงวดที่ถึงงวดที่ของโมเดล และ โครงการที่ระบุ
updateDownPayment parameter : a_projectID, a_modelID, a_fromInstalment, a_toInstalment, a_downAmount	ปรับปรุงข้อมูล ถึงงวดที่ และค่าผ่อนชำระล่วงหน้า ของจากงวดที่ของโมเดลในโครงการที่ระบุ
deleteDownPayment parameter : a_projectID, a_modelID, a_fromInstalment	ลบรายการค่าผ่อนชำระล่วงหน้า ของจากงวดที่ของโมเดลในโครงการที่ระบุ

ตารางที่ 4.17 แอทริบิวต์สำหรับคลาส NewProjectHouseNo

แอทริบิวต์	ชนิดข้อมูล	คำอธิบาย
beginHouseNo	CHAR(8)	บ้านเลขที่เริ่มต้น
Ascendent = 00000001	CHAR(8)	ค่าเพิ่ม
isGenerated = 0	NUMBER(1)	ใช้แล้ว
quotation	Quotation	ใช้เชื่อมโยงกับคลาส Quotation

ตารางที่ 4.18 การดำเนินการของคลาส NewProjectHouseNo

การดำเนินการ	คำอธิบาย
isNewProjectHouseNo parameter : a_projectID,a_modelID,beginHouseNo return : BOOLEAN	ตรวจสอบว่ามีรายการบ้านเลขที่ ที่มีบ้านเลขที่เริ่มต้น โมเดล และ โครงการตามที่ระบุหรือไม่ ถ้ามีจะให้ค่ากลับเป็น TRUE ถ้าไม่มีจะให้ค่ากลับเป็น FALSE
isOverlapWith parameter : a_beginHouseNo,an_ascendent, an_unitAmount,a_beginHouseNo_2, an_ascendent_2,an_unitAmount_2	ตรวจสอบว่าช่วงบ้านเลขที่ที่กำหนดซ้ำซ้อนกันกับที่เคยกำหนดไว้หรือไม่ หากซ้ำซ้อนกันจะให้ค่ากลับเป็นจริง ถ้าไม่จะให้ค่ากลับเป็นเท็จ
amountOfNewProjectHouseNo parameter : a_projectID,a_modelID return : NUMBER	คำนวณหาจำนวนรายการกำหนดบ้านเลขที่ของโครงการ และ โมเดลที่ระบุ
retrieveHouseNo parameter : a_projectID,a_modelID, beginHouseNo return : NewProjectHouseNo	เข้าถึงข้อมูลบ้านเลขที่ของโครงการ โมเดลที่ระบุ ที่มีบ้านเลขที่เริ่มต้นไม่เป็นไปตามที่ระบุ ซึ่งจะใช้ในการตรวจสอบการซ้ำซ้อนของบ้านเลขที่
retrieveProject parameter : a_projectID return : NewProjectHouseNo	เข้าถึงข้อมูลบ้านเลขที่ของโครงการที่ระบุ
setIsGenerated parameter : a_projectID,a_modelID, a_beginHouseNo,isGenerated	กำหนดค่าให้กับแอตทริบิวต์ isGenerated โดยที่ 1 แทน การนำไปใช้แล้ว ส่วน 0 แทนยังไม่มี การนำรายการเลขที่บ้าน ไปใช้
insertProjectHouseNo parameter : a_projectID, a_modelID, a_beginHouseNo, unitAmount, an_assendent	เพิ่มรายการบ้านเลขที่
updateProjectHouseNo parameter : a_projectID, a_modelID, a_beginHouseNo, unitAmount, an_assendent	ปรับปรุงข้อมูล ค่าเพิ่ม และจำนวนหน่วยอาศัย
deleteProjectHouseNo parameter : a_projectID, a_modelID, a_beginHouseNo	ลบรายการบ้านเลขที่

ตารางที่ 4.19 แอทริบิวต์สำหรับคลาส Unit

แอทริบิวต์	ชนิดข้อมูล	คำอธิบาย
HouseNo	VARCHAR2(9)	บ้านเลขที่
DebtorCode	CHAR(14)	รหัสลูกหนี้
BuildingNo	NUMBER(2)	หมายเลขอาคาร
Quotation	Quotation	ใช้เชื่อมโยงกับคลาส Quotation

ตารางที่ 4.20 การดำเนินการของคลาส Unit (ส่วนที่ 1/1)

การดำเนินการ	คำอธิบาย
isNewUnit parameter : a_projectID,a_modelID, a_HouseNo return : BOOLEAN	ตรวจสอบว่ามีหน่วยอาศัยที่มีบ้านเลขที่ โมเดล ของโครงการตามที่ระบุหรือไม่ ถ้ามีจะให้ค่า กลับเป็น TRUE ถ้าไม่มีจะให้ค่ากลับเป็น FALSE
sumAmount parameter : a_projectID,a_modelID return : NUMBER	คำนวณจำนวนหน่วยอาศัยที่มีโมเดล และ โครง การตามที่ระบุ
amountOfProjectUnit parameter : a_projectID return : NUMBER	คำนวณหาจำนวนหน่วยอาศัยของโครงการที่ ระบุ
fromatedDebtorCode parameter : a_debtorCode return : VARCHAR2	ให้รหัสลูกหนี้ที่อยู่ในรูปแบบ โครงการ- โมเดล-บ้านเลขที่ คือ xxxx-xx-xxxx/xxxx
retrieveProjectUnit parameter : a_projectID return : Unit	เข้าถึงข้อมูลบ้านเลขที่ของทั้งโครงการที่ระบุ
initialDebtorCode parameter : a_projectID	ตั้งรหัสโครงการ โดยกำหนดรหัสลูกหนี้ให้กับ แต่ละหน่วยอาศัยทั้งหมดในโครงการที่ระบุ
generateProjectHouseNo parameter : a_projectID,a_modelID,an_unitAmount, a_beginHouseNo,an_ascendent return : -	สร้างรายการหน่วยอาศัยของโครงการ ที่เป็น โมเดลตามที่ระบุ ทั้งหมดจำนวนหน่วยอาศัยที่ ระบุ โดยเริ่มจากบ้านเลขที่ที่ระบุ ที่มีค่าเพิ่ม ตามที่ระบุ
insertUnit parameter : a_projectID,a_modelID,a_HouseNo	เพิ่มรายการในหน่วยอาศัยให้มีบ้านเลขที่ โมเดลของโครงการตามที่ระบุ

ตารางที่ 4.20 การดำเนินการของคลาส Unit (ส่วนที่ 1/2)

การดำเนินการ	คำอธิบาย
deleteUnit parameter : a_projectID, a_modelID, HouseNo	ลบหน่วยอาศัยของโครงการ ที่เป็น โมเดล และ บ้านเลขที่ตามที่ระบุ
deleteProjectHouseNo parameter : a_projectID, a_modelID, a_beginHouseNo	ลบหน่วยอาศัยที่อยู่ในช่วงบ้านเลขที่เริ่มต้น จำนวนหน่วย ค่าเพิ่ม ที่เป็น โมเดลของโครงการ ที่ระบุ

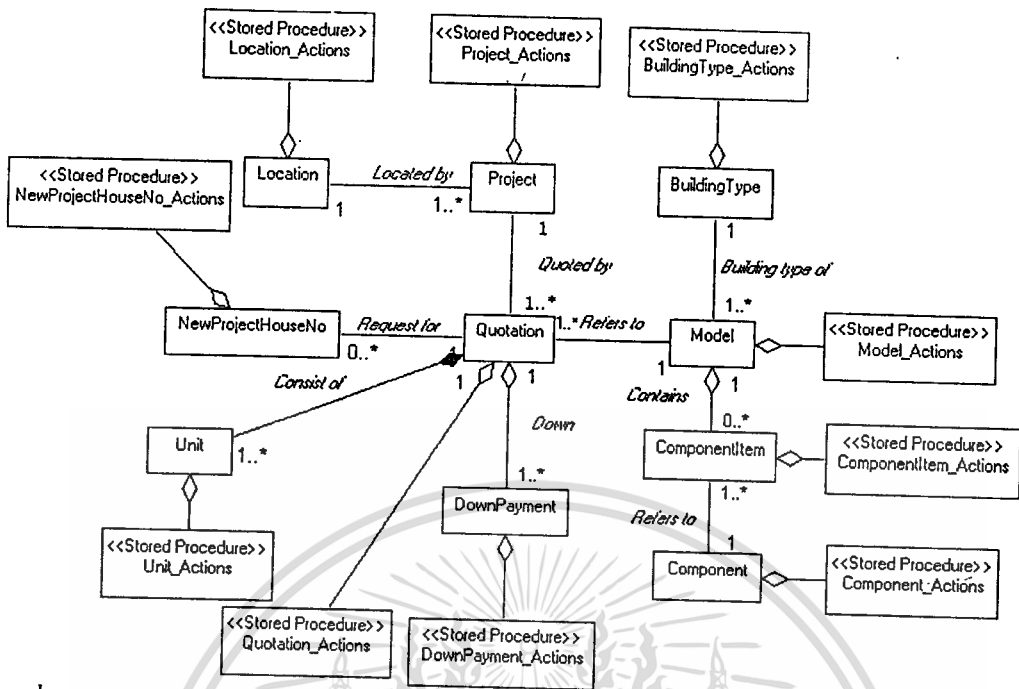
4.2 คลาสในแอคเซสเลเยอร์

คลาสในแอคเซสเลเยอร์เป็นคลาสที่เป็นการดำเนินการของคลาสในบิสซิเนสเลเยอร์ ซึ่งเก็บไว้ที่ฐานข้อมูลที่เรียกว่า สตอโปรซีเจอร์ (Stored Procedure) ในการทำงานคลาสใน บิสซิเนสเลเยอร์จะร้องขอกับระบบจัดการฐานข้อมูล

รูปที่ 4.3 คลาสไดอะแกรมแสดงความสัมพันธ์ระหว่างคลาสในบิสซิเนสเลเยอร์ กับ คลาสในแอคเซสเลเยอร์ การออกแบบคลาสในแอคเซสเลเยอร์ได้จากการดำเนินการของคลาส ในบิสซิเนสเลเยอร์ โดยตั้งชื่อคลาสคู่กับคลาสในบิสซิเนสเลเยอร์ ต่อด้วย “_Actions” บ่งบอก ถึงการดำเนินการ เช่น คลาส Project ก็จะใช้คลาส Project_Actions ในการเข้าถึงข้อมูลใน ฐานข้อมูล แต่ละคลาสมีความเป็นอิสระต่อกันมาก แต่ละคลาสจะประกอบด้วยดำเนินการ ซึ่งเป็นโมดูลที่ทำงานเฉพาะกับคลาสในบิสซิเนสเลเยอร์ ซึ่งในการอิมพลีเมนต์คลาสในแอค เซสเลเยอร์เหล่านี้เทียบได้กับ สตอโปรซีเจอร์ที่เป็นโมดูลที่เขียนขึ้นด้วยภาษา PL/SQL ซึ่งผ่าน การแปลภาษาไว้ล่วงหน้าแล้ว และจัดเก็บอยู่ในดาตาเบสเซิร์ฟเวอร์ จะถูกสั่งให้ทำงานในเข้าถึง ข้อมูลในฐานข้อมูลในดาตาเบสเซิร์ฟเวอร์

โดยที่แต่ละคลาสในแอคเซสเลเยอร์จะกลายเป็นแต่ละแพ็คเกจในสตอโปรซีเจอร์ และ แต่ละการดำเนินการในคลาสจะกลายเป็นหนึ่งโมดูลในแพ็คเกจ ซึ่งสามารถอธิบายอัลกอริทึม ของแต่ละการดำเนินการหรือโมดูลด้วยแอคตีวิตีไดอะแกรม

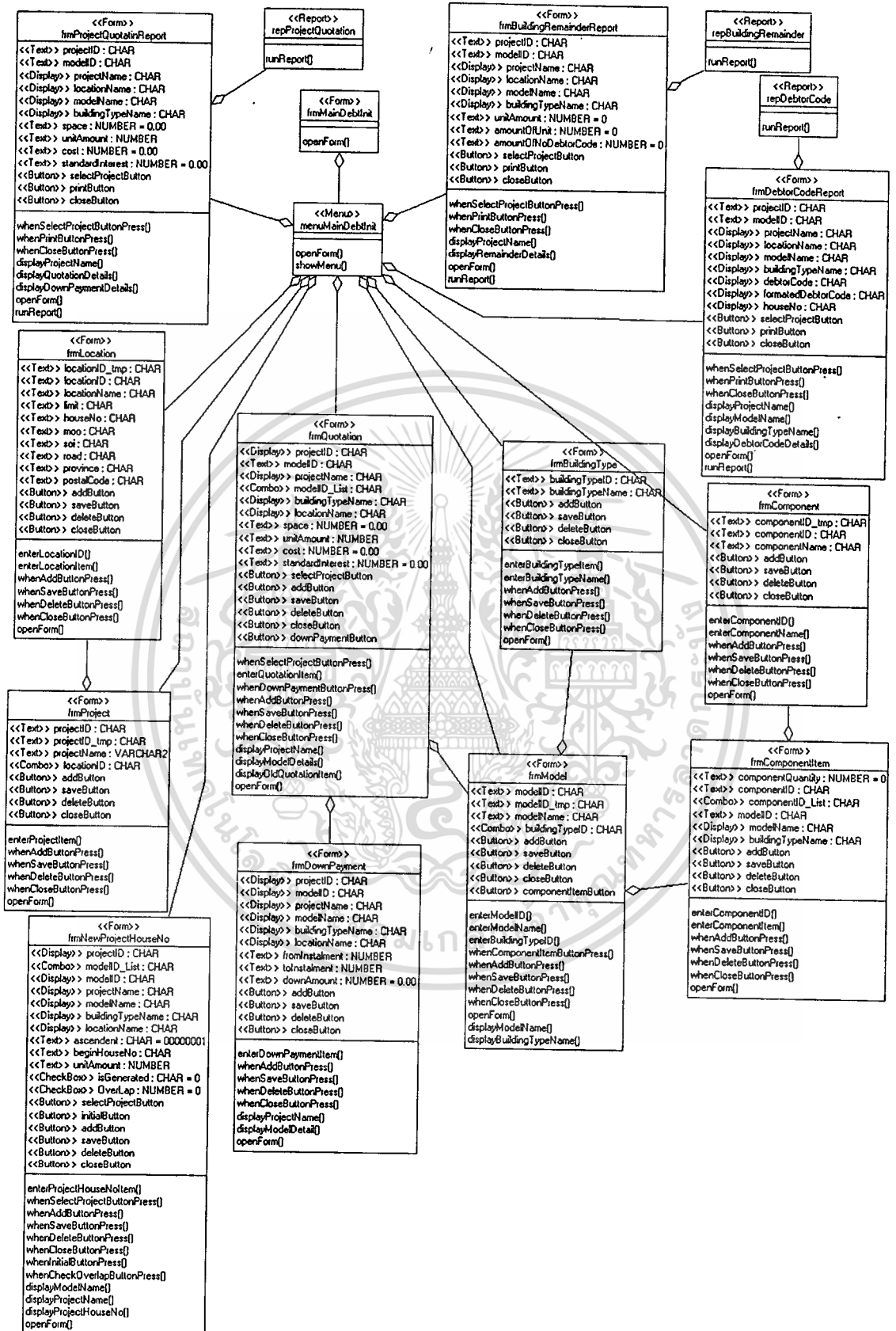
ดังนั้นคำอธิบายรายละเอียดของคลาสในแอคเซสเลเยอร์เหมือนกับของการดำเนินการ ในหัวข้อ 4.1 หมายเลขตารางเลขคู่ 4.2,4.4,4.6,...,4.20 ส่วนรายละเอียดการออกแบบอัลกอริทึม และการสร้างได้รวบรวมไว้ในบทที่ 7



รูปที่ 4.3 คลาสไดอะแกรมแสดงความสัมพันธ์ระหว่างคลาสในบิตชีเนสเลเยอร์ กับคลาสในแอคเซสเลเยอร์

4.3 คลาสในวิวเลเยอร์

คลาสในวิวเลเยอร์เลเยอร์เป็นคลาสที่เป็นยูสอินเตอร์เฟซ ประกอบด้วยคลาสยูสเซอร์อินเตอร์เฟซต่าง ๆ ได้แก่ เมนู และฟอร์ม ซึ่งจะใช้ประกอบในการออกแบบยูสเซอร์อินเตอร์เฟซต่อไป รูปที่ 4.4 เป็นคลาสไดอะแกรมแสดงความสัมพันธ์ระหว่างคลาสในวิวเลเยอร์



รูปที่ 4.4 คลาสไดอะแกรมแสดงความสัมพันธ์ระหว่างคลาสในวิวเลเยอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การออกแบบคลาสในวิวเลเยอร์ได้มาจากการสร้างชีแคว้นไคอะแกรม และคอลแลบบอเลชั่นไคอะแกรม ควบคู่กับการทำต้นแบบโดยใช้ form builder และ report builder ของ Oracle Developer 6 ได้คลาสหลักคือ คลาสของเมนูหลัก menuMainDebInit ผูกติดกับคลาส frmMainDebtInit คลาสส่วนใหญ่ถูกเรียกใช้ผ่านคลาสเมนูหลัก จะมีบางคลาสที่ขึ้นกับอีกคลาสหนึ่ง และบางคลาสก็สามารถถูกเรียกใช้จากหลายคลาสได้ มีข้อสังเกตว่า สำหรับคลาสที่เป็นเมนูจะมีสเตอริโอไทป์ (Stereotype) เป็น <<Menu>> คลาสที่เป็นฟอร์มหรือหน้าจอจะมีสเตอริโอไทป์เป็น <<Form>> ส่วนคลาสที่เป็นรายงานจะมีสเตอริโอไทป์เป็น <<Report>> คำอธิบายความสัมพันธ์ระหว่างคลาสในวิวเลเยอร์เป็นดังตารางที่ 4.21 ส่วนความสัมพันธ์กับคลาสในบิสซิเนสเลเยอร์ดังรูปที่ 4.5

ตารางที่ 4.21 คำอธิบายคลาสในวิวเลเยอร์ (ส่วนที่ 1/3)

ชื่อคลาส	คำอธิบาย
1. menuMainDebtInit	เป็นคลาสเมนูหลักที่จะโยงไปเรียกใช้คลาสฟอร์ม และคลาสรายงานต่าง ๆ ได้แก่ คลาสฟอร์ม frmProject คลาสฟอร์ม frmLocation คลาสฟอร์ม frmModel คลาสฟอร์ม frmComponent คลาสฟอร์ม frmBuildingType คลาสฟอร์ม frmQuotation คลาสฟอร์ม frmNewProjectHouseNo คลาสฟอร์ม frmProjectQuotationReport คลาสรายงาน repProjectQuotation คลาสรายงาน repDebtorCode คลาสรายงาน repBuildingRemainder แต่คลาสเมนูหลักนี้จะผูกติดกับคลาสฟอร์มหลัก
2. frmMainDebInit	เป็นคลาสฟอร์มหลักสำหรับคลาสเมนูหลัก
3. frmProject	เป็นคลาสฟอร์มสำหรับการจัดการข้อมูลโครงการ และขณะจัดการข้อมูลผู้ใช้สามารถจัดการข้อมูลสำนักงานเคหะชุมชน ผ่านคลาสฟอร์ม frmLocation
4. frmLocation	เป็นคลาสฟอร์มสำหรับการจัดการข้อมูลสำนักงานเคหะชุมชน

ตารางที่ 4.21 คำอธิบายคลาสในวิวเลเยอร์ (ส่วนที่ 2/3)

ชื่อคลาส	คำอธิบาย
5. frmModel	เป็นคลาสฟอร์มสำหรับการจัดการข้อมูลโมเดล ซึ่งสามารถกำหนดรายละเอียดองค์ประกอบในโมเดล โดยเรียกใช้คลาสฟอร์ม frmComponentItem และสามารถจัดการข้อมูลประเภทสิ่งปลูกสร้างโดยเรียกใช้คลาสฟอร์ม frmBuildingType
6. frmComponent	เป็นคลาสฟอร์มสำหรับการจัดการข้อมูลองค์ประกอบ
7. frmComponentItem	เป็นคลาสฟอร์มสำหรับการจัดการข้อมูลองค์ประกอบในโมเดล ซึ่งจะถูกรวบรวมใช้โดยคลาสฟอร์ม frmModel
8. frmBuildingType	เป็นคลาสฟอร์มสำหรับการจัดการข้อมูลประเภทสิ่งปลูกสร้าง
9. frmQuotation	เป็นคลาสฟอร์มสำหรับการจัดการข้อมูลการขายโครงการ ในขณะมีการจัดการข้อมูลการขายนี้สามารถเข้าไปจัดการข้อมูลโมเดล ผ่านคลาสฟอร์ม frmModel
10. frmDownPayment	เป็นคลาสฟอร์มสำหรับการจัดการข้อมูลการผ่อนชำระล่วงหน้า
11. frmNewProjectHouseNo	เป็นคลาสฟอร์มสำหรับการจัดการข้อมูลบ้านเลขที่โครงการ เพื่อการสร้างข้อมูลหน่วยอาศัย และการตั้งหนี้โครงการ
12. frmProjectQuotationReport	เป็นคลาสฟอร์มสำหรับการออกรายงานการขายโครงการ โดยให้ผู้ใช้ต้องเลือกโครงการที่ต้องการ ก่อนที่จะสร้างรายงานได้เป็นคลาสรายงาน repProjectQuotation
13. frmDebtorCodeReport	เป็นคลาสฟอร์มสำหรับการออกรายงานรหัสลูกหนี้โครงการ โดยให้ผู้ใช้ต้องเลือกโครงการที่ต้องการ ก่อนที่จะสร้างรายงานได้เป็นคลาสรายงาน repDebtorCode
14. frmBuildingRemainderReport	เป็นคลาสฟอร์มสำหรับการออกรายงานทะเบียนอาคารคงเหลือที่ยังไม่มีการตั้งหนี้ โดยให้ผู้ใช้ต้องเลือกโครงการที่ต้องการ ก่อนที่จะสร้างรายงานได้เป็นคลาสรายงาน repBuildingRemainder
15. repProjectQuotation	เป็นคลาสรายงานการขายโครงการที่ผู้ใช้กำหนดจากคลาสฟอร์ม frmBuildingRemainderReport มีรายละเอียดเกี่ยวกับการขายโครงการ การผ่อนชำระล่วงหน้า และอัตราดอกเบี้ยการชำระค่าเช่าซื้อต่อปี
16. repDebtorCode	เป็นคลาสรายงานรหัสลูกหนี้โครงการที่ผู้ใช้กำหนดจากคลาสฟอร์ม frmDebtorCodeReport มีรายละเอียดเกี่ยวกับรหัสลูกหนี้ที่ได้จากการตั้งหนี้โครงการ แก่แต่ละหน่วยอาศัย

บทที่ 5

การออกแบบฐานข้อมูลเชิงสัมพันธ์

การออกแบบระบบฐานข้อมูลเชิงสัมพันธ์ หรือรีเลชันนอลของระบบดังนี้โครงการได้มาจากการแปลงคลาสในบิสซิเนสคลาสเป็นโครงสร้าง (Schema) ของฐานข้อมูลหรือรีเลชัน (Relation) แอทริบิวต์ในคลาสจะกลายเป็นคอตัมน์ของรีเลชัน การดำเนินการของคลาสจะเปรียบได้กับสตอโปรซีเจอร์ (Stored Procedure) และความสัมพันธ์ระหว่างคลาสจะกลายเป็นรีเลชันชิป (Relationship) ระหว่างรีเลชัน หรือการอ้างอิงด้วยคีย์นอก (Foreign Key)

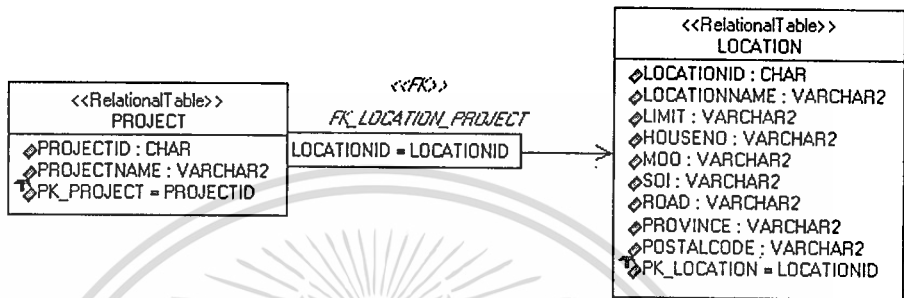
ในทางปฏิบัติการแปลงดังกล่าวสามารถใช้ความสามารถของ Rational Rose สร้างคลาสขึ้นใหม่อีกชุดหนึ่งในที่นี้จะรวบรวมไว้ในแพ็คเกจชื่อ Relational Table โดยการลอกแบบมาจากคลาสในบิสซิเนสคลาสที่รวบรวมไว้ในแพ็คเกจ Object Type นำมาสร้างเป็นอีกคลาสไดอะแกรมหนึ่งแล้วใช้เครื่องมือของ Rational Rose สร้างคำสั่งในการนิยามฐานข้อมูล (DDL: Data Definition Language) ได้เป็นไฟล์ .sql และสามารถนำไปใช้นิยามฐานข้อมูลสำหรับระบบจัดการฐานข้อมูลที่ต้องการ เช่น ออราเคิล 8 เป็นต้น

5.1 การสร้างคลาสไดอะแกรมสำหรับแปลง

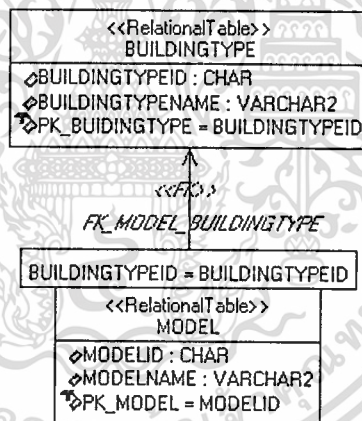
การสร้างคลาสไดอะแกรมเพื่อการแปลงที่นำไปสู่การนิยามฐานข้อมูล ควรจะสร้างคลาสใหม่ขึ้นมาโดยการลอกเลียนมาจากคลาสในบิสซิเนสคลาส และปรับให้เหมาะสมกับระบบฐานข้อมูลเชิงสัมพันธ์ ซึ่งอาจจะแปลงจากหนึ่งคลาสเป็นหนึ่งรีเลชัน หรือจากหลายคลาสเป็นหนึ่งรีเลชัน หรือจากคลาสที่มีคุณสมบัติแบบซูเปอร์คลาส/ซับคลาส ไปเป็นหนึ่งรีเลชัน หรือแอกรีเลชันก็ได้ตามความเหมาะสม แต่ด้วยความสามารถของ Rational Rose มีเครื่องมือช่วยในการสนับสนุนการนิยามฐานข้อมูลของออราเคิล 8 ตามขั้นตอนต่อไปนี้

1. เพิ่มคอมโพเนนท์ ที่มีสเตอร์ไอโไทป์เป็น schema สำหรับออราเคิล 8 ชื่อของคอมโพเนนท์จะหมายถึงชื่อ schema ในระบบฐานข้อมูล
2. กำหนดสเตอร์ไอโไทป์ <<Object Type>> ให้กับทุกคลาสในบิสซิเนสคลาสที่จะใช้เป็นต้นแบบ

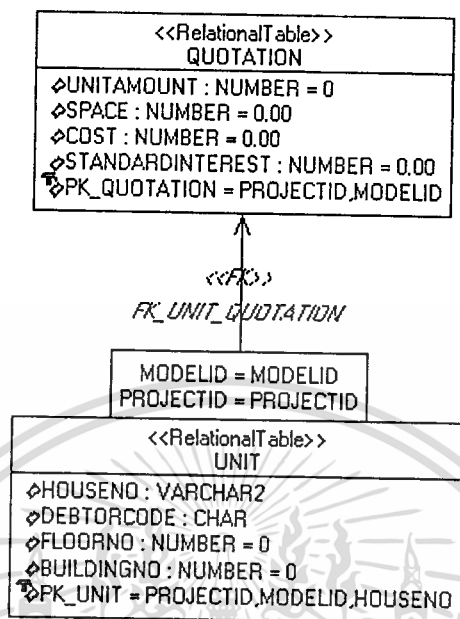
3. ใช้ Create Data Type Wizard สร้างคลาสใหม่ที่มีสเตอริโอไทป์เป็น <<Relational Table>> ทำตามลำดับที่เครื่องมือนี้แนะนำจะได้คลาสใหม่ดังรูปที่ 5.1- 5.7 ซึ่งเป็นคลาสสำหรับฐานข้อมูลระบบดังนี้โครงการบัญชีลูกหนี้การเคหะแห่งชาติ



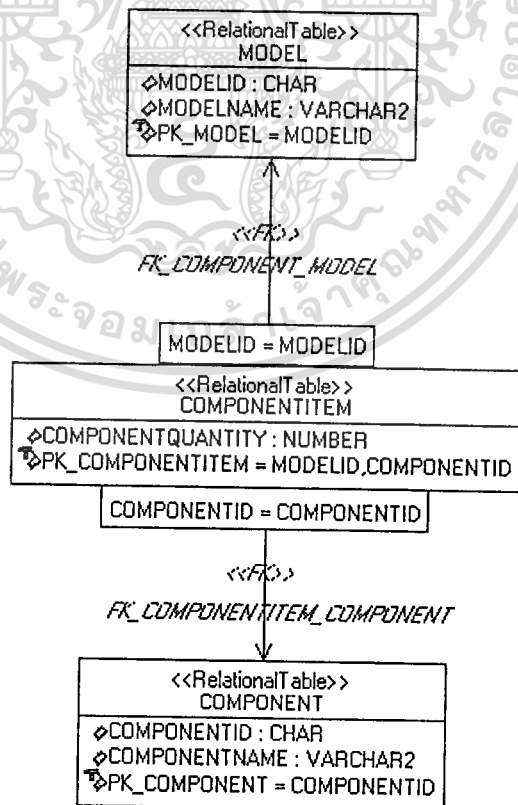
รูปที่ 5.1 คลาสรีเลชันโครงการ กับสำนักงานเคหะชุมชน



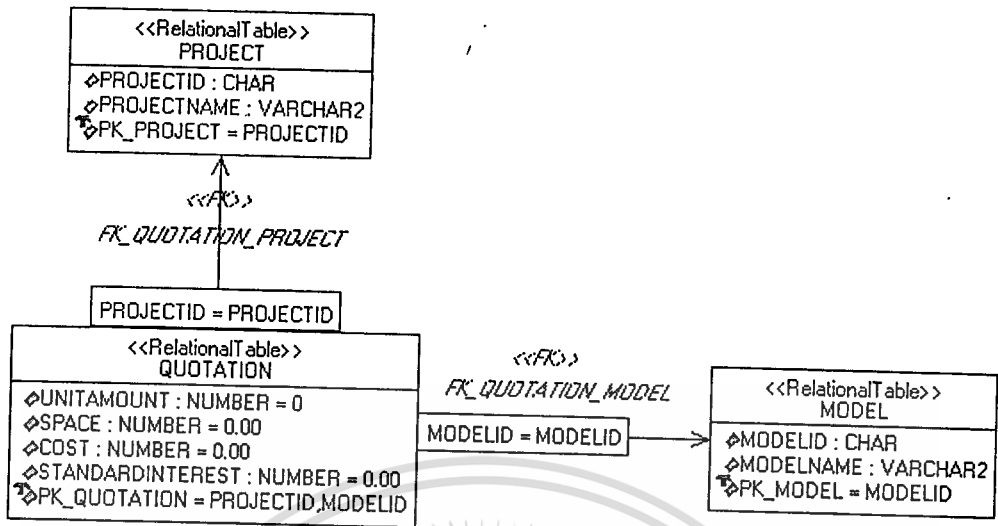
รูปที่ 5.2 คลาสรีเลชัน โมเดล กับชนิดสิ่งปลูกสร้าง



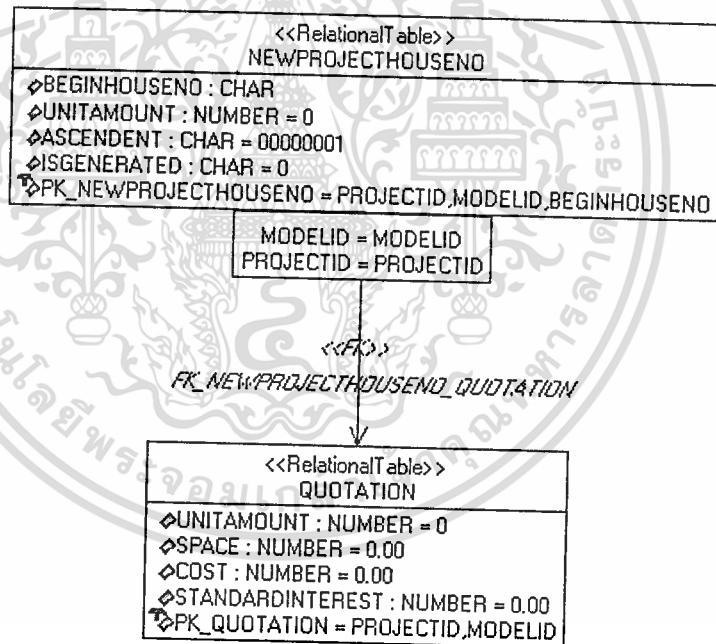
รูปที่ 5.3 คลาสรีเลชันหน่วยอาศัย กับข้อมูลขายโครงการ



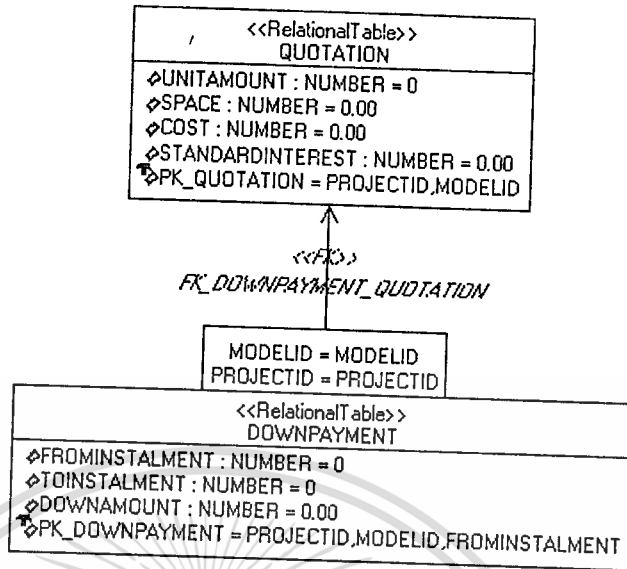
รูปที่ 5.4 คลาสรีเลชัน โมเดล รายการองค์ประกอบในโมเดล และองค์ประกอบ



รูปที่ 5.5 คลาสรีเลขณ์ข้อมูลขายโครงการ โครงการ และโมเดล



รูปที่ 5.6 คลาสรีเลขณ์ข้อมูลขายโครงการ กับบ้านเลขที่สำหรับหน่วยอาศัยในโครงการ



รูปที่ 5.7 คลาสรีเลย์ข้อมูลขายโครงการ กับการผ่อนชำระล่วงหน้า

ข้อสังเกต

1. แอทริบิวต์ที่จะเป็นคอลลัมน์ในรีเลย์ ซึ่งสามารถลอกเลียนมาจากคลาสในบิสซิเนสคลาส หรือจะสร้างขึ้นใหม่ตามความเหมาะสม และต้องความสามารถในการใช้แอทริบิวต์ (visibility) เป็น public
2. กำหนดแอทริบิวต์ที่จะเป็นคีย์หลัก (Primary Key) มีสเตอร์ไอโทปเป็น <<PK>> และกำหนดค่าเริ่มต้นให้เป็นชื่อแอทริบิวต์ที่ต้องการ และกำหนดความสามารถในการใช้แอทริบิวต์นี้ให้เป็น implementation หรือสัญลักษณ์รูปค้อน
3. กำหนดแอทริบิวต์ที่จะเป็นคีย์นอก (Foreign Key) (ถ้ามี) ที่ความสัมพันธ์ระหว่างคลาสนั้นชื่อความสัมพันธ์เป็นชื่อคีย์นอก มีสเตอร์ไอโทปเป็น <<FK>> และควอริไฟเออร์เป็นแอทริบิวต์ที่จะเป็นคีย์นอก โดยกำหนดค่าเริ่มต้นเป็นแอทริบิวต์ที่อ้างถึงคีย์หลักของอีกรีเลย์
4. ความสัมพันธ์ระหว่างคลาสนั้นเป็นแบบทิศทางเดียวหัวลูกศรเปิดอยู่ทางด้านคลาสนี้คีย์นอกอ้างอิง
5. กำหนดการนิยามข้อมูล ตามรูปแบบของออราเคิล 8

5.2 คำสั่งนิยามฐานข้อมูล

การใช้เครื่องมือใน Rational Rose ในการนิยามฐานข้อมูลของระบบจะต้องเซตให้คลาสใหม่ที่สร้างขึ้นอ้างอิงถึงคอมโพเนนต์ DebtInit แล้วจึงใช้คำสั่ง Schema Generation สำหรับออรากิเล 8 จะได้ไฟล์ .sql ซึ่งภายในจะบรรจุคำสั่งในการนิยามฐานข้อมูลระบบสำหรับ schema DebtInit นั้นเอง และในที่นี้ได้บันทึกเก็บในชื่อไฟล์ schema.sql ดังรูปที่ 5.8

```
CREATE TABLE DebtInit.COMPONENT (
    COMPONENTID CHAR(2) NOT NULL,
    COMPONENTNAME VARCHAR2(50),
    CONSTRAINT PK_COMPONENT PRIMARY KEY (COMPONENTID));

CREATE TABLE DebtInit.BUILDINGTYPE (
    BUILDINGTYPEID CHAR(2) NOT NULL,
    BUILDINGTYPENAME VARCHAR2(50),
    CONSTRAINT PK_BUIDINGTYPE PRIMARY KEY (BUILDINGTYPEID));

CREATE TABLE DebtInit.LOCATION (
    LOCATIONID CHAR(4) NOT NULL,
    LOCATIONNAME VARCHAR2(100),
    LIMIT VARCHAR2(2),
    HOUSENO VARCHAR2(9),
    MOO VARCHAR2(2),
    SOI VARCHAR2(2),
    ROAD VARCHAR2(20),
    PROVINCE VARCHAR2(20),
    POSTALCODE VARCHAR2(5),
    CONSTRAINT PK_LOCATION PRIMARY KEY (LOCATIONID));

CREATE TABLE DebtInit.PROJECT (
    LOCATIONID CHAR(4),
    PROJECTID CHAR(4) NOT NULL,
    PROJECTNAME VARCHAR2(100),
    CONSTRAINT PK_PROJECT PRIMARY KEY (PROJECTID),
    CONSTRAINT FK_LOCATION_PROJECT FOREIGN KEY(LOCATIONID) REFERENCES
DebtInit.LOCATION(LOCATIONID));

CREATE TABLE DebtInit.MODEL (
    BUILDINGTYPEID CHAR(2),
    MODELID CHAR(2) NOT NULL,
```

รูปที่ 5.8 คำสั่งนิยามฐานข้อมูลที่ได้ในไฟล์ schema.sql (ส่วนที่ 1/3)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

MODELNAME VARCHAR2(20),
CONSTRAINT PK_MODEL PRIMARY KEY (MODELID),
CONSTRAINT FK_MODEL_BUILDINGTYPE FOREIGN KEY(BUILDINGTYPEID)
REFERENCES DebtInit.BUILDINGTYPE(BUILDINGTYPEID));

CREATE TABLE DebtInit.COMPONENTITEM (
MODELID CHAR(2) NOT NULL,
COMPONENTID CHAR(2) NOT NULL,
COMPONENTQUANTITY NUMBER(2),
CONSTRAINT PK_COMPONENTITEM PRIMARY KEY (MODELID,COMPONENTID),
CONSTRAINT FK_COMPONENTITEM_COMPONENT FOREIGN KEY(COMPONENTID)
REFERENCES DebtInit.COMPONENT(COMPONENTID),
CONSTRAINT FK_COMPONENT_MODEL FOREIGN KEY(MODELID) REFERENCES
DebtInit.MODEL(MODELID));

CREATE TABLE DebtInit.QUOTATION (
PROJECTID CHAR(4) NOT NULL,
MODELID CHAR(2) NOT NULL,
UNITAMOUNT NUMBER(2) DEFAULT 0,
SPACE NUMBER(5,2) DEFAULT 0.00,
COST NUMBER(11,2) DEFAULT 0.00,
STANDARDINTEREST NUMBER(4,2) DEFAULT 0.00,
CONSTRAINT PK_QUOTATION PRIMARY KEY (PROJECTID,MODELID),
CONSTRAINT FK_QUOTATION_PROJECT FOREIGN KEY(PROJECTID) REFERENCES
DebtInit.PROJECT(PROJECTID),
CONSTRAINT FK_QUOTATION_MODEL FOREIGN KEY(MODELID) REFERENCES
DebtInit.MODEL(MODELID));

CREATE TABLE DebtInit.NEWPROJECTHOUSENO (
PROJECTID CHAR(4) NOT NULL,
MODELID CHAR(2) NOT NULL,
BEGINHOUSENO CHAR(8) NOT NULL,
UNITAMOUNT NUMBER(3) DEFAULT 0,
ASCENDENT CHAR(8) DEFAULT 00000001,
ISGENERATED CHAR(1) DEFAULT 0,
CONSTRAINT PK_NEWPROJECTHOUSENO PRIMARY KEY
(PROJECTID,MODELID,BEGINHOUSENO),
CONSTRAINT FK_NEWPROJECTHOUSENO_QUOTATION FOREIGN KEY
(MODELID,PROJECTID) REFERENCES DebtInit.QUOTATION(MODELID,PROJECTID));

CREATE TABLE DebtInit.DOWNPAYMENT (
PROJECTID CHAR(4) NOT NULL,

```

รูปที่ 5.8 คำสั่งนิยามฐานข้อมูลที่ได้ในไฟล์ schema.sql (ส่วนที่ 2/3)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

MODELID CHAR(2) NOT NULL,
FROMINSTALMENT NUMBER(2) NOT NULL DEFAULT 0,
TOINSTALMENT NUMBER(2) DEFAULT 0,
DOWNAMOUNT NUMBER(8,2) DEFAULT 0.00,
CONSTRAINT PK_DOWNPAYMENT PRIMARY KEY
(PROJECTID,MODELID,FROMINSTALMENT),
CONSTRAINT FK_DOWNPAYMENT_QUOTATION FOREIGN
KEY(MODELID,PROJECTID) REFERENCES DebtInit.QUOTATION(MODELID,PROJECTID));

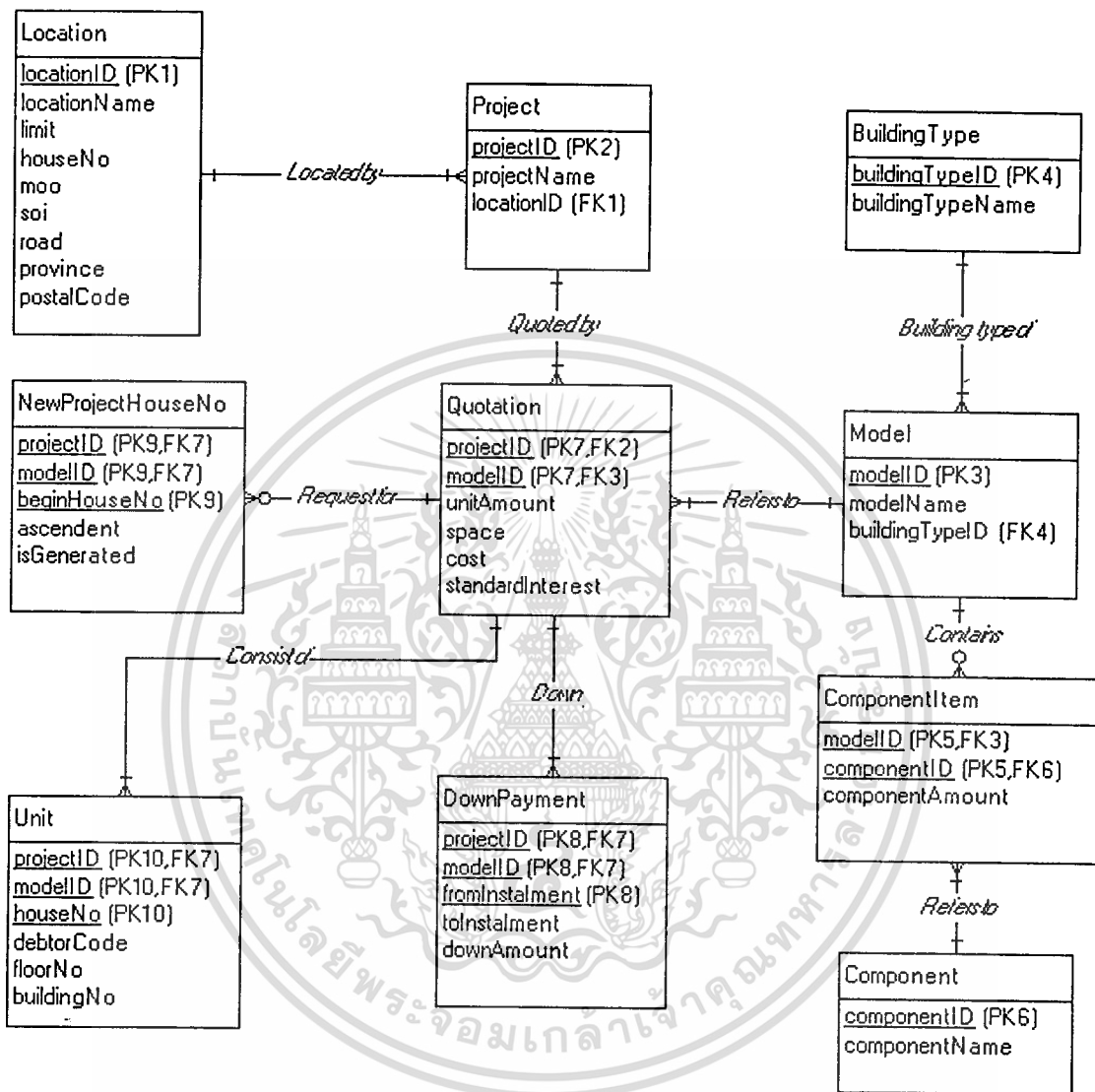
CREATE TABLE DebtInit.UNIT (
PROJECTID CHAR(4) NOT NULL,
MODELID CHAR(2) NOT NULL,
HOUSENO VARCHAR2(9) NOT NULL,
DEBTORCODE CHAR(14),
FLOORNO NUMBER(2) DEFAULT 0,
BUILDINGNO NUMBER(2) DEFAULT 0,
CONSTRAINT PK_UNIT PRIMARY KEY (PROJECTID,MODELID,HOUSENO),
CONSTRAINT FK_UNIT_QUOTATION FOREIGN KEY(MODELID,PROJECTID)
REFERENCES DebtInit.QUOTATION(MODELID,PROJECTID));

```

รูปที่ 5.8 คำสั่งนิยามฐานข้อมูลที่ได้ในไฟล์ schema.sql (ส่วนที่ 3/3)

5.3 แผนภาพโครงสร้างฐานข้อมูลเชิงสัมพันธ์

แผนภาพโครงสร้างฐานข้อมูลเชิงสัมพันธ์สรุปได้ดังรูปที่ 5.8



รูปที่ 5.9 แผนภาพโครงสร้างฐานข้อมูลเชิงสัมพันธ์

5.4 อธิบายโครงสร้างฐานข้อมูลรวม

ตารางที่ 5.1 ฐานข้อมูล Location หรือสำนักงานเคหะชุมชน

แอทริบิวต์	ชนิดข้อมูล	คำอธิบาย
<u>LocationID</u> (PK1)	CHAR(4) NOT NULL	รหัสสำนักงานเคหะชุมชน
locatinName	VARCHAR2(50)	ชื่อสำนักงานเคหะชุมชน
limit	VARCHAR2(2)	เขต
houseNo	VARCHAR2(9)	บ้านเลขที่
moo	VARCHAR2(2)	หมู่ที่
soi	VARCHAR2(3)	ซอย
road	VARCHAR2(30)	ถนน
province	VARCHAR2(30)	จังหวัด
postalCode	VARCHAR2(8)	รหัสไปรษณีย์

ตารางที่ 5.2 ฐานข้อมูล Project หรือ โครงการ

แอทริบิวต์	ชนิดข้อมูล	คำอธิบาย
<u>ProjectID</u> (PK2)	CHAR(4) NOT NULL	รหัสโครงการ
ProjectName	VARCHAR2(100)	ชื่อ โครงการ
LocationID (FK1)	CHAR(4) NOT NULL	รหัสสำนักงานเคหะชุมชน

ตารางที่ 5.3 ฐานข้อมูล Model หรือ โมเดล

แอทริบิวต์	ชนิดข้อมูล	คำอธิบาย
<u>ModelID</u> (PK3)	CHAR(2) NOT NULL	รหัสโมเดล
modelName	VARCHAR2(20)	ชื่อ โมเดล
BuildingTypeID (FK4)	CHAR(2) NOT NULL	รหัสประเภทสิ่งปลูกสร้าง

ตารางที่ 5.4 ฐานข้อมูล Building Type หรือประเภทสิ่งปลูกสร้าง

แอทริบิวต์	ชนิดข้อมูล	คำอธิบาย
<u>buildingTypeID</u> (PK4)	CHAR(2) NOT NULL	รหัสประเภทสิ่งปลูกสร้าง
buildingTypeName	VARCHAR2(50)	ชื่อประเภทสิ่งปลูกสร้าง

ตารางที่ 5.5 ฐานข้อมูล ComponentItem หรือองค์ประกอบในโมเดล

แอทริบิวต์	ชนิดข้อมูล	คำอธิบาย
<u>ModelID</u> (PK5)	CHAR(2) NOT NULL	รหัสโมเดล
<u>ComponentID</u> (PK5,FK6)	CHAR(2) NOT NULL	รหัสองค์ประกอบ
componentAmount	NUMBER(2)	จำนวนองค์ประกอบในโมเดล

ตารางที่ 5.6 ฐานข้อมูล Component หรือองค์ประกอบ

แอทริบิวต์	ชนิดข้อมูล	คำอธิบาย
<u>ComponentID</u> (PK6)	CHAR(2) NOT NULL	รหัสองค์ประกอบ
componentName	VARCHAR2(50)	ชื่อองค์ประกอบ

ตารางที่ 5.7 ฐานข้อมูล Quotation หรือการขายโครงการ

แอทริบิวต์	ชนิดข้อมูล	คำอธิบาย
<u>ProjectID</u> (PK7,FK2)	CHAR(4) NOT NULL	รหัสโครงการ
<u>ModelID</u> (PK7,FK3)	CHAR(2) NOT NULL	รหัสโมเดล
UnitAmount	NUMBER(3) DEFAULT 0	จำนวนหน่วยอาศัย
Space	NUMBER(5,2) DEFAULT 0.00	เนื้อที่ต่อหน่วย (ตร.ว)
Cost	NUMBER(11,2) DEFAULT 0.00	ราคาขายเงินสดต่อหน่วย (บาท)
StandardInterest	NUMBER(4,2)	อัตราดอกเบี้ยการผ่อนชำระค่าเช่า ซื้อต่อปี (%)

ตารางที่ 5.8 ฐานข้อมูล DownPayment หรือการผ่อนชำระเงินล่วงหน้า

แอทริบิวต์	ชนิดข้อมูล	คำอธิบาย
<u>ProjectID</u> (PK8,FK7)	CHAR(4) NOT NULL	รหัสโครงการ
<u>ModelID</u> (PK8,FK7)	CHAR(2) NOT NULL	รหัสโมเดล
<u>FromInstalment</u> (PK8)	NUMBER(2) NOT NULL	จากงวดที่
ToInstalment	NUMBER(2)	ถึงงวดที่
downAmount	NUMBER(8,2) DEFAULT 0.00	ค่าผ่อนชำระเงินล่วงหน้าต่องวด (บาท)

ตารางที่ 5.9 ฐานข้อมูล NewProjectHouseNo หรือบ้านเลขที่โครงการ

แอทริบิวต์	ชนิดข้อมูล	คำอธิบาย
<u>ProjectID</u> (PK9,FK7)	CHAR(4) NOT NULL	รหัสโครงการ
<u>ModelID</u> (PK9,FK7)	CHAR(2) NOT NULL	รหัสโมเดล
<u>BeginHouseNo</u> (PK9)	CHAR(8) NOT NULL	บ้านเลขที่เริ่มต้น
Ascendent	CHAR(8) DEFAULT 00000001	ค่าเพิ่ม
isGenerated	NUMBER(1) DEFAULT 0	ใช้แล้ว

ตารางที่ 5.10 ฐานข้อมูล Unit หรือหน่วยอาศัย

แอทริบิวต์	ชนิดข้อมูล	คำอธิบาย
<u>ProjectID</u> (PK10,FK7)	CHAR(4) NOT NULL	รหัสโครงการ
<u>ModelID</u> (PK10,FK7)	CHAR(2) NOT NULL	รหัสโมเดล
<u>HouseNo</u> (PK10)	VARCHAR2(9) NOT NULL	บ้านเลขที่
DebtorCode	CHAR(14)	รหัสลูกหนี้
BuildingNo	NUMBER(2)	หมายเลขอาคาร

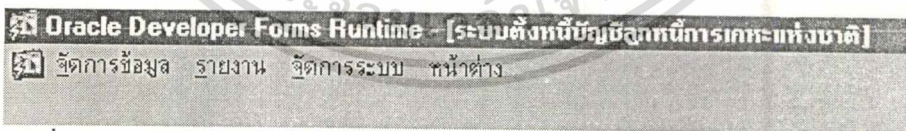
บทที่ 6

การออกแบบยูสเซอร์อินเตอร์เฟซ

การออกแบบยูสเซอร์อินเตอร์เฟซระบบตั้งนี้โครงการบัญชีลูกหนี้ของการเคหะแห่งชาติ อาศัยคลาสในวิเลเยอร์จากบทที่ 4 และการสร้างต้นแบบด้วย developer ของออราเคิล ได้แก่ ฟอรั่ม หรือหน้าจอ เมนู และรายงานเป็นต้น ในทางปฏิบัติยูสเซอร์อินเตอร์เฟซเหล่านี้จะขึ้นอยู่กับฐานข้อมูลของระบบ ซึ่งเปรียบได้กับคลาสในบิสซิเนสเลเยอร์ ส่วนการทำงานเกี่ยวกับฐานข้อมูลผ่านยูสเซอร์อินเตอร์เฟซ มีการเรียกใช้ผ่านสตอโปรซีเจอร์ที่เก็บอยู่ที่ดาตาเบสเซิร์ฟเวอร์ กล่าวคือคลาสในแอคเซสเลเยอร์นั่นเอง ในทางปฏิบัติแล้วการทำงานผ่านฟอรั่มของออราเคิล จะมีการเรียกใช้คำสั่งจาก Standard Extension ใน Built-in Package ของ Developer ของออราเคิล เช่น คำสั่งในการยอมรับการเปลี่ยนแปลงข้อมูลผ่านฟอรั่ม หรือ `commit_form` หรือจะเป็นคำสั่งในการเข้าถึงข้อมูลในฐานข้อมูลมาแสดงที่หน้าจอ หรือ `execute_query` เป็นต้น ถือได้ว่า Standard Extension เป็นคลาสหนึ่งในแอคเซสเลเยอร์

6.1 การออกแบบเมนู

การออกแบบยูสเซอร์อินเตอร์เฟซส่วนสำคัญเริ่มต้น คือ เมนูหลัก ซึ่งขึ้นอยู่กับฟอรั่มหลักดังรูปที่ 6.1

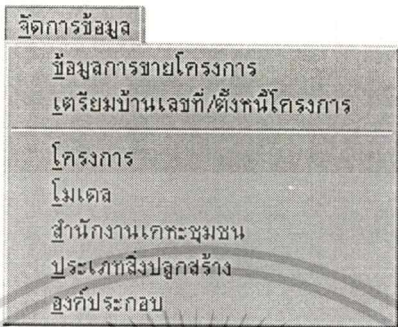


รูปที่ 6.1 เมนูหลัก ซึ่งอยู่บนฟอรั่มหลัก

เมนูหลักแบ่งเป็นหมวดได้แก่ เมนูจัดการข้อมูล เมนูรายงาน และจัดการระบบ ส่วนเมนูหน้าต่างนั้นเป็นเมนูที่ Developer ของออราเคิลให้มาอัตโนมัติเมื่อมีการรันฟอรั่ม

1. เมนูจัดการข้อมูล

เมนูจัดการข้อมูล ประกอบด้วยเมนูย่อยที่ทำหน้าที่เปิดฟอร์มที่เกี่ยวข้องกับการจัดการข้อมูลในฐานข้อมูลของระบบ ดังรูปที่ 6.2 และคำอธิบายเมนูย่อยดังตารางที่ 6.1



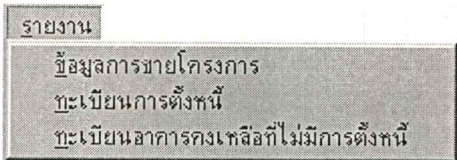
รูปที่ 6.2 เมนูย่อยสำหรับจัดการข้อมูลในฐานข้อมูล

ตารางที่ 6.1 คำอธิบายเมนูย่อยของเมนูจัดการข้อมูล

ชื่อเมนูย่อย	คำอธิบาย
1. ข้อมูลขายโครงการ	เพื่อเปิดฟอร์มจัดการข้อมูลขายโครงการ รูปที่ 6.10
2. เตรียมบ้านเลขที่	เพื่อเปิดฟอร์มจัดการข้อมูลในการเตรียมบ้านเลขที่ รูปที่ 6.11
3. โครงการ	เพื่อเปิดฟอร์มจัดการข้อมูลโครงการ รูปที่ 6.12
4. โมเดล	เพื่อเปิดฟอร์มจัดการข้อมูลโมเดล รูปที่ 6.13
5. สำนักงานเคหะชุมชน	เพื่อเปิดฟอร์มจัดการข้อมูลสำนักงานเคหะชุมชน รูปที่ 6.14
6. ประเภทสิ่งปลูกสร้าง	เพื่อเปิดฟอร์มจัดการข้อมูลประเภทสิ่งปลูกสร้าง รูปที่ 6.15
7. องค์ประกอบ	เพื่อเปิดฟอร์มจัดการข้อมูลองค์ประกอบ รูปที่ 6.16

2. เมนูรายงาน

เมนูรายงานประกอบด้วยเมนูเมนูย่อยที่ทำหน้าที่เปิดฟอร์มที่เกี่ยวข้องกับการจัดการออกรายงาน ดังรูปที่ 6.3 และคำอธิบายเมนูย่อยดังตารางที่ 6.2



รูปที่ 6.3 เมนูย่อยสำหรับการออกรายงาน

ตารางที่ 6.2 คำอธิบายเมนูย่อยของเมนูรายงาน

ชื่อเมนูย่อย	คำอธิบาย
1. ข้อมูลขายโครงการ	เพื่อเปิดฟอร์มออกรายงานข้อมูลขายโครงการ รูปที่ 6.19
2. ทะเบียนการตั้งหนี้	เพื่อเปิดฟอร์มออกรายงานการตั้งหนี้ หรือรหัสลูกหนี้โครงการจัดการ รูปที่ 6.20
3. ทะเบียนอาคารคงเหลือที่ไม่มี การตั้งหนี้	เพื่อเปิดฟอร์มออกรายงานทะเบียนอาคารคงเหลือที่ยังไม่มีการตั้งหนี้ รูปที่ 6.21

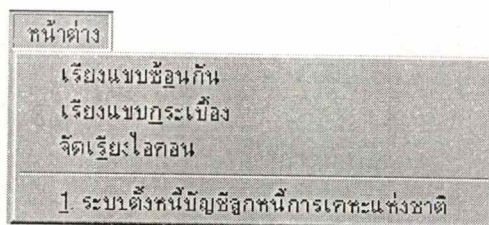
3. เมนูจัดการระบบ

ประกอบด้วยเมนูย่อยที่ทำหน้าที่เปิดฟอร์มที่เกี่ยวข้องกับการจัดการระบบ ในที่นี้มีเมนูย่อยเดียวคือ เมนูออกจากระบบ ดังรูปที่ 6.4 เป็นการสิ้นสุดการทำงาน หรือออกจากโปรแกรมระบบตั้งหนี้การเคหะแห่งชาติ



4. เมนูหน้าต่าง

เมนูหน้าต่าง เป็นเมนูที่ developer ของอราเคลให้ใช้แสดงอัตโนมัติมีการใช้เมนูขณะรันโปรแกรม ซึ่งประกอบด้วยเมนูย่อยที่ใช้จัดการหน้าต่างขณะรันโปรแกรม ดังรูปที่ 6.5 เมนูย่อย 3 เมนูบนใช้การจัดการหน้าต่าง ส่วนเมนูย่อยล่างสุดคือหน้าต่างปัจจุบันที่กำลังเปิดอยู่ จากตัวอย่างนี้ก็คือ ของฟอร์มหลักนั่นเอง



6.2 การออกแบบฟอร์มยูสเซอร์อินเตอร์เฟซ

การออกแบบฟอร์มยูสเซอร์อินเตอร์เฟซ มาจากคลาสฟอร์มในวิวเลเยอร์ แต่ละฟอร์มอาจถูกเรียกใช้ผ่านเมนู หรือจากฟอร์มที่เกี่ยวข้องกันก็ได้

1. แถบแสดงสถานะ (Status Bar)

ส่วนประกอบสำคัญที่ทุกฟอร์มมีคือ แถบแสดงสถานะการทำงาน ตัวอย่างดังรูปที่ 6.6 ประกอบด้วย

- สถานะการทำงานเกี่ยวกับฐานข้อมูลที่ฟอร์มอ้างอิง ได้แก่
 EDIT หมายถึง อยู่ในสถานะแก้ไขข้อมูล ซึ่งสามารถแก้ไข หรือลบข้อมูลได้
 ADD หมายถึง อยู่ในสถานะที่มีการเพิ่มข้อมูล
 DISP หมายถึง อยู่ในสถานะการแสดงผลเท่านั้น ไม่สามารถแก้ไข ลบหรือเพิ่ม
- แผ่นแสดงข้อความเพื่อแจ้งข่าวสารเกี่ยวกับการทำงาน และผลการทำงาน
- ชื่อที่ใช้เข้าสู่ระบบ
- วันที่ปัจจุบัน

DISP	พบทั้งหมด 3 รายการ	noe	05/03/43
------	--------------------	-----	----------

รูปที่ 6.6 ตัวอย่างแถบแสดงสถานะ

2. ปุ่มคำสั่ง (Command Button)

ปุ่มคำสั่งในฟอร์มมีวัตถุประสงค์ต่างกันไป ดังนี้

- ปุ่มคำสั่ง “เลือกโครงการ” ใช้แสดงรายการชื่อโครงการ ซึ่งผู้ใช้สามารถค้นหา และเลือกการรายการชื่อโครงการที่ต้องการ ดังรูปที่ 6
- ปุ่มคำสั่ง “เพิ่มรายการ” ใช้เมื่อต้องการเพิ่มรายการข้อมูล จะพบในฟอร์มจัดการข้อมูล
- ปุ่มคำสั่ง “ลบรายการ” ใช้เมื่อต้องการลบรายการข้อมูลที่ต้องการ จะพบในฟอร์มจัดการข้อมูล
- ปุ่มคำสั่ง “บันทึกรายการ” ใช้เมื่อยอมรับการเพิ่ม หรือแก้ไขรายการข้อมูล จะพบในฟอร์มจัดการข้อมูล
- ปุ่มคำสั่ง “ปิด” ใช้เมื่อต้องการปิดฟอร์มปัจจุบันที่กำลังทำงานอยู่
- ปุ่มคำสั่ง “..” ใช้เมื่อต้องการเปิดฟอร์มที่เกี่ยวข้อง เช่นเมื่อต้องการจัดการข้อมูลเกี่ยวกับการผ่อนชำระล่วงหน้า จากฟอร์มจัดการข้อมูลการขายโครงการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้คัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- ปุ่มคำสั่ง “ตั้งหนี้โครงการ” ใช้เมื่อต้องการตั้งให้โปรแกรมทำการตั้งหนี้โครงการ และกำหนดรหัสลูกหนี้ให้แต่ละหน่วยอาศัย ปุ่มนี้มีเฉพาะในฟอร์มจัดการข้อมูลบ้านเลขที่โครงการ
- ปุ่มคำสั่ง “รายงาน” ใช้เมื่อต้องการตั้งให้โปรแกรมสร้างรายงานที่ต้องการ ปุ่มนี้พบเฉพาะในฟอร์มออกรายงานเท่านั้น

3. รายการค่าข้อมูล (List Of Value)

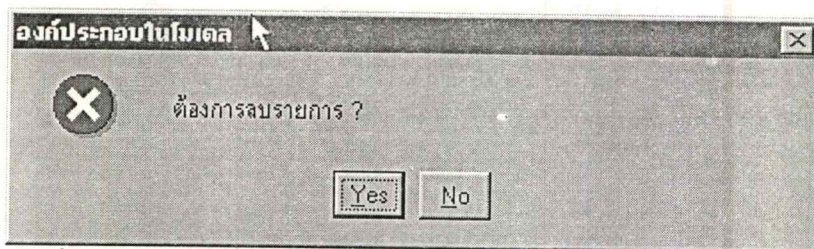
รายการค่าข้อมูลใช้แสดงรายชื่อโครงการ ที่สามารถค้นหา และเลือกรายการที่ต้องการได้ ดังรูปที่ 6.7



รูปที่ 6.7 รายการค่าข้อมูล

4. ใดะลอกแสดงข้อความเตือน (Alert)

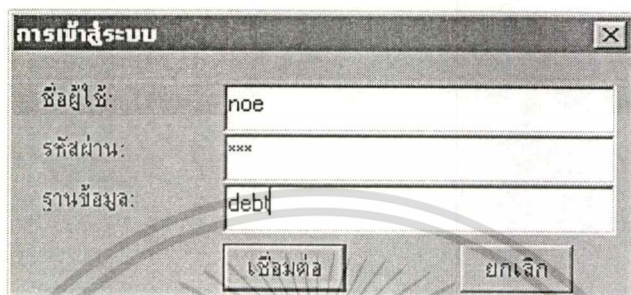
ใดะลอกแสดงข้อความเตือน แสดงขึ้นเพื่อเตือน หรือแนะนำขั้นตอนต่อไปให้ผู้ใช้งาน เช่น รูปที่ 6.8 แสดงข้อความให้ผู้ใช้งานตัดสินใจอีกครั้งว่าจะลบข้อมูลข้อมูลจริงหรือไม่



รูปที่ 6.8 ใดะลอกแสดงข้อความเตือน

5. ไดอะล็อกการเข้าสู่ระบบ (Log on Dialog)

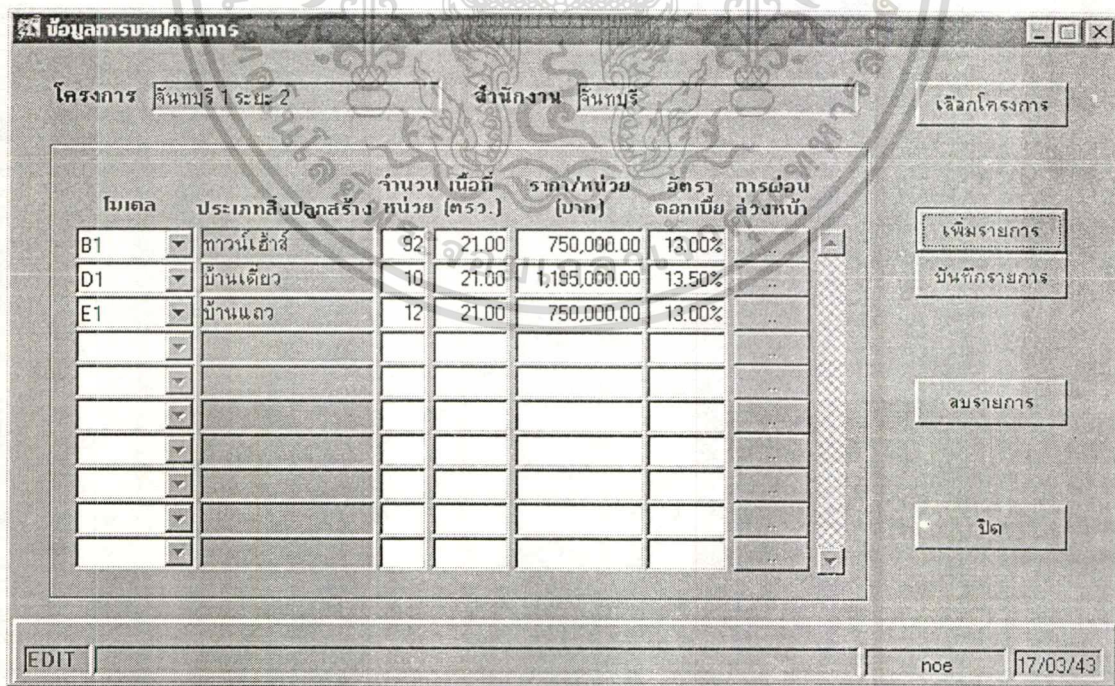
ไดอะล็อกการเข้าสู่ระบบ แสดงทุกครั้งเมื่อผู้ใช้งาน โปรแกรมเริ่มต้นเข้าสู่ระบบ ผู้ใช้ ต้องกรอกชื่อผู้ใช้ (login name) รหัสผ่าน (password) และฐานข้อมูล (Database Connection) ดังรูปที่ 6.9



รูปที่ 6.9 ไดอะล็อกการเข้าสู่ระบบ

6. ฟอर्मจัดการข้อมูลขายโครงการ

ฟอर्मจัดการข้อมูลขายโครงการ ผู้ใช้ต้องคลิกปุ่มเลือกโครงการ จะแสดงรายชื่อโครงการที่กล่าวมาแล้วในรูปที่ 6.7 หลังจากนั้นจะแสดงข้อมูลขายโครงการ ดังรูปที่ 6.10



โมเดล	ประเภทสิ่งปลูกสร้าง	จำนวน	เนื้อที่	ราคา/หน่วย	อัตราดอกเบี้ย	การผ่อน
		หน่วย	(ตรว.)	(บาท)	ต่อเดือน	ล่วงหน้า
B1	ทาวน์เฮ้าส์	92	21.00	750,000.00	13.00%	..
D1	บ้านเดี่ยว	10	21.00	1,195,000.00	13.50%	..
E1	บ้านแถว	12	21.00	750,000.00	13.00%	..
						..
						..
						..
						..
						..

รูปที่ 6.10 ฟอर्मจัดการข้อมูลขายโครงการ

8. ฟอรั่มจัดการข้อมูลโครงการ

ฟอรั่มจัดการข้อมูลโครงการ ดังรูปที่ 6.12

รหัส	ชื่อโครงการ	สำนักงานเกษตรชุมชน
0005	คลองจัน 1 ระยะ 1	
0003	คลองปะปา 1 ระยะ 1	บางปะกง
0004	จันทบุรี 1 ระยะ 2	จันทบุรี
0001	ร่มเกล้าวิสัย	บางปะกง
0002	วิมานฟ้ารุ่ง	แหลมสัก

เพิ่มรายการ
บันทึกรายการ
ลบรายการ
ปิด

EDIT noe 17/03/43

รูปที่ 6.12 ฟอรั่มจัดการข้อมูลโครงการ

9. ฟอรั่มจัดการข้อมูลโมเดล

ฟอรั่มจัดการข้อมูลโมเดล ดังรูปที่ 6.13 ซึ่งสามารถถูกเรียกโดยฟอรั่มจัดการข้อมูลขายโครงการ

รหัส	ชื่อโมเดล	ประเภทสิ่งปลูกสร้าง
01	A1	ที่ดินโล่ง
05	A2	ที่ดินโล่ง
02	B1	ทาวน์เฮ้าส์
03	D1	บ้านเดี่ยว
04	E1	บ้านแถว

เพิ่มรายการ
บันทึกรายการ
ลบรายการ
ปิด

EDIT ลบเรียบร้อยแล้ว noe 17/03/43

รูปที่ 6.13 ฟอรั่มจัดการข้อมูลโมเดล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

10. โปรแกรมจัดการข้อมูลสำนักงานเคหะชุมชน

โปรแกรมจัดการข้อมูลสำนักงานเคหะชุมชน ดังรูปที่ 6.14 สามารถถูกเรียกใช้โดยโปรแกรมจัดการข้อมูลโครงการ

รหัส	ชื่อสำนักงานเคหะชุมชน	เขต	บ้านเลขที่	หมู่ที่	ถนน	จังหวัด	รหัสไปรษณีย์
0003	คลองจั่น	03	12/11		สุขาภิบาล 1	กรุงเทพฯ	10240
0004	จันทบุรี	04	279/176		สามสมัย	จันทบุรี	42100
0001	บางปะกง	01	5/21	1	สองฤดู	สมุทรปราการ	56000
0002	แหลมสิง	02	26	2	ประชาราษฎร์	ชุมพร	86000

รูปที่ 6.14 โปรแกรมจัดการข้อมูลสำนักงานเคหะชุมชน

11. โปรแกรมจัดการข้อมูลประเภทสิ่งปลูกสร้าง

โปรแกรมจัดการข้อมูลประเภทสิ่งปลูกสร้าง ดังรูปที่ 6.15 สามารถถูกเรียกใช้โดยโปรแกรมโมเดล

รหัส	ประเภทสิ่งปลูกสร้าง
04	ทาวนเฮ้าส์
00	ที่ดินโล่ง
03	บ้านเดี่ยว
01	บ้านแถว
06	บ้านแฝด
05	อาคารพาณิชย์

รูปที่ 6.15 โปรแกรมจัดการข้อมูลประเภทสิ่งปลูกสร้าง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้คัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

12. โปรแกรมจัดการข้อมูลองค์ประกอบ

โปรแกรมจัดการข้อมูลองค์ประกอบ ดังรูปที่ 6.16 สามารถถูกเรียกใช้โดยโปรแกรมจัดการข้อมูลองค์ประกอบในโมเดล

รหัส	ชื่อองค์ประกอบ
01	ชิ้น
03	กล่องนอน
05	กล่องนั่งเล่น
04	กล่องสวม
02	กล่องอเนกประสงค์

รูปที่ 6.16 โปรแกรมจัดการข้อมูลองค์ประกอบ

13. โปรแกรมจัดการข้อมูลองค์ประกอบในโมเดล

โปรแกรมจัดการข้อมูลองค์ประกอบในโมเดล ถูกเรียกใช้จากฟอร์มโมเดลเพื่อกำหนดรายการองค์ประกอบที่บ่งบอกว่าเป็นโมเดลดังกล่าว ดังรูปที่ 6.17

องค์ประกอบ	จำนวน
ชิ้น	1
กล่องอเนกประสงค์	2
กล่องนอน	3

รูปที่ 6.17 โปรแกรมจัดการข้อมูลองค์ประกอบในโมเดล

14. โปรแกรมจัดการข้อมูลการผ่อนชำระล่วงหน้า

โปรแกรมจัดการข้อมูลการผ่อนชำระล่วงหน้า ดังรูปที่ 6.18 ต้องเรียกใช้ผ่านฟอร์มจัดการข้อมูลขายโครงการ เพื่อกำหนดค่าผ่อนชำระล่วงหน้าต้องงวด

รูปที่ 6.18 โปรแกรมจัดการข้อมูลการผ่อนชำระล่วงหน้า

15. โปรแกรมออกรายงานข้อมูลขายโครงการ

โปรแกรมออกรายงานข้อมูลขายโครงการ ดังรูปที่ 6.19 ต้องเลือกโครงการที่ต้องการรายงานข้อมูลขาย

16. โปรแกรมออกรายงานทะเบียนการตั้งหนี้ หรือรหัสลูกหนี้โครงการ

โปรแกรมออกรายงานทะเบียนการตั้งหนี้ หรือรหัสลูกหนี้โครงการ ดังรูปที่ 6.20 ต้องเลือกโครงการที่ต้องการรายงานข้อมูลขาย

17. โปรแกรมออกรายงานทะเบียนอาคารคงเหลือที่ไม่มีการตั้งหนี้

โปรแกรมออกรายงานทะเบียนอาคารคงเหลือที่ไม่มีการตั้งหนี้ ดังรูปที่ 6.21 ต้องเลือกโครงการที่ต้องการรายงานข้อมูลขาย

รายงานการขายโครงการเคหะชุมชน

โครงการ จินตपुरิ 1 ระยะ 2 สำนักงาน จินตपुरิ

โมเดล	ประเภทสิ่งปลูกสร้าง	จำนวน หน่วย	เนื้อที่ (ตรว.)	ราคา/หน่วย (บาท)	อัตรา ดอกเบี้ย
B1	ทาวน์เฮ้าส์	92	21.00	750,000.00	13.00%
D1	บ้านเดี่ยว	10	21.00	1,195,000.00	13.50%
E1	บ้านแถว	12	21.00	750,000.00	13.00%

เลือกโครงการ
รายงาน
ปิด

DISP พบทั้งหมด 3 รายการ noe 05/03/43

รูปที่ 6.19 ฟอรม์ออกรายงานข้อมูลขายโครงการ

รายงานลูกหนี้โครงการ

โครงการ จินตपुरิ 1 ระยะ 2 สำนักงาน จินตपुरิ

โมเดล	ประเภทสิ่งปลูกสร้าง	รหัสลูกหนี้	บ้านเลขที่
B1	ทาวน์เฮ้าส์	0004-02-0100/0001	0100/0001
B1	ทาวน์เฮ้าส์	0004-02-0100/0002	0100/0002
B1	ทาวน์เฮ้าส์	0004-02-0100/0003	0100/0003
B1	ทาวน์เฮ้าส์	0004-02-0100/0004	0100/0004
B1	ทาวน์เฮ้าส์	0004-02-0100/0005	0100/0005
B1	ทาวน์เฮ้าส์	0004-02-0100/0006	0100/0006
B1	ทาวน์เฮ้าส์	0004-02-0100/0007	0100/0007
B1	ทาวน์เฮ้าส์	0004-02-0100/0008	0100/0008
B1	ทาวน์เฮ้าส์	0004-02-0100/0009	0100/0009
B1	ทาวน์เฮ้าส์	0004-02-0100/0010	0100/0010

เลือกโครงการ
รายงาน
ปิด

DISP พบทั้งหมด 114 รายการ noe 25/03/43

รูปที่ 6.20 ฟอรม์ออกรายงานทะเบียนการตั้งหนี้ หรือรหัสลูกหนี้โครงการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้คัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 6.21 รายงานทะเบียนอาคารคงเหลือที่ยังไม่มีการตั้งหนี้

โครงการ สำนักงาน

โมเดล	ชนิดสิ่งปลูกสร้าง	จำนวน หน่วย	ตั้งหนี้ ไปแล้ว	ยังไม่ได้ ตั้งหนี้
B1	ทาวน์เฮ้าส์	20	10	10
D1	บ้านเดี่ยว	10	5	5

เลือกโครงการ
รายงาน
ปิด

DISP พบทั้งหมด 2 รายการ noe 04/03/43

รูปที่ 6.21 ฟอรม์ออกรายงานทะเบียนอาคารคงเหลือที่ยังไม่มีการตั้งหนี้

6.3 การออกแบบรายงาน

การออกแบบรายงานมาจากคลาสรายงานในวิวเลเยอร์ แต่ละรายงานถูกเรียกใช้จากฟอรม์ออกรายงานที่กล่าวมาแล้วในหัวข้อ 6.3

1. แบบรายงานข้อมูลขายโครงการ

จากฟอรม์ออกรายงานข้อมูลขายโครงการ เมื่อผู้ใช้คลิกปุ่มรายงาน จะแสดงรายงานข้อมูลขายโครงการดังตัวอย่างรายงานรูปที่ 6.22

รายงานการขายโครงการเคหะชุมชน

พิมพ์วันที่ 04/03/00 วันที่ 1 / 1

โครงการ จันทบุรี 1 ระยะ 2 สำนักงานเคหะชุมชน จันทบุรี

อันดับ ที่	โมดล	ประเภท สิ่งปลูกสร้าง	จำนวน หน่วย	พื้นที่ (ตร.ว.ก.หน่วย)	ราคาขายเงินสด (บาท/หน่วย)	อัตราดอกเบี้ย ผ่อนชำระ (%/ปี)	งวดที่ ถึงงวดที่	ผ่อนชำระเงิน ล่วงหน้า (บาท/เดือน)
1	B1	ทาวน์เฮาส์	92	21	750,000.00	13	1 12	5,000.00
							13 24	6,000.00
2	D1	บ้านเดี่ยว	10	21	1,195,000.00	13.5	1 12	6,000.00
							13 24	7,500.00
3	E1	บ้านแถว	12	21	750,000.00	13	1 12	5,100.00
							13 24	6,100.00

รูปที่ 6.22 ตัวอย่างรายงานข้อมูลขายโครงการแบบรายงานทะเบียนการตั้งหนี้ หรือรหัสลูกหนี้โครงการ

จากฟอร์มออกรายงานทะเบียนการตั้งหนี้ หรือรหัสลูกหนี้โครงการ เมื่อผู้ใช้คลิกสกรีนรายงาน จะแสดงรายงานทะเบียนการตั้งหนี้ หรือรหัสลูกหนี้โครงการ ดังตัวอย่างรายงานรูปที่ 6.23

รายงานรหัสลูกหนี้โครงการ

พิมพ์วันที่ 04/03/00 หน้าที่ 5 / 5

โครงการ จันทบุรี 1 ระยะ 2

สำนักงานเกษตรชุมชน จันทบุรี

โมเดล E1

ประเภทสิ่งปลูกสร้าง บ้านแถว

อันดับที่	รหัสลูกหนี้	บ้านเลขที่
8	0004-04-0300/0008	0300/0008
9	0004-04-0300/0009	0300/0009
10	0004-04-0300/0010	0300/0010
11	0004-04-0300/0011	0300/0011
12	0004-04-0300/0012	0300/0012

รวม : 12 รายการ

รวมทั้งหมด : 114 รายการ

รูปที่ 6.23 ตัวอย่างรายงานทะเบียนการตั้งหนี้ หรือรหัสลูกหนี้โครงการ

2. แบบรายงานทะเบียนอาคารคงเหลือที่ไม่มีการตั้งหนี้

จากฟอร์มรายงานทะเบียนอาคารคงเหลือที่ไม่มีการตั้งหนี้ เมื่อผู้ใช้คลิกปุ่มรายงาน จะแสดงรายงานทะเบียนอาคารคงเหลือที่ไม่มีการตั้งหนี้ ดังตัวอย่างรายงานรูปที่ 6.24

รายงานทะเบียนอาคารคงเหลือที่ยังไม่มีการตั้งหนี้

พิมพ์วันที่ 04/03/00 หน้าที่ 1 / 1

โครงการ รมเกล้าวิลล่า

สำนักงานเคหะชุมชน บางปะกง

โมเดล	ประเภทสิ่งปลูกสร้าง	จำนวนหน่วย	ตั้งหนี้ไปแล้ว	ยังไม่ได้ตั้งหนี้
B1	ทาวน์เฮ้าส์	20	10	<u>10</u>
		ลำดับที่	บ้านเลขที่เริ่มต้น	ค่าเพิ่ม
		1	00040001	00000001
				จำนวนหน่วยอาศัย
				รวม
D1	บ้านเดี่ยว	10	5	<u>5</u>
		ลำดับที่	บ้านเลขที่เริ่มต้น	ค่าเพิ่ม
		2	00050001	00000001
				จำนวนหน่วยอาศัย
				รวม

รูปที่ 6.24 ตัวอย่างรายงานทะเบียนอาคารคงเหลือที่ไม่มีการตั้งหนี้

บทที่ 7

การสร้างระบบ

การสร้างระบบ หรือการอิมพลีเมนต์ (Implementation) คือการลงมือปฏิบัติ หรือเขียนโปรแกรม โดยปฏิบัติต่อเนื่อง และควบคู่กับการวิเคราะห์ และออกแบบระบบ ซึ่งระบบงานนี้เป็นสถาปัตยกรรมแบบไคลเอนต์/เซิร์ฟเวอร์ การพัฒนาทางฝั่งไคลเอนต์ ได้แก่ แอปพลิเคชันที่เป็นยูสเซอร์อินเตอร์เฟซ หรือวิวเลเยอร์ และบิสซิเนสเลเยอร์ ส่วนทางฝั่งเซิร์ฟเวอร์ คือดาตาเบสเซิร์ฟเวอร์ จะมีแอสเซสเลเยอร์ เป็นโมดูลที่เรียกว่า สโตโปรซีเจอร์

เครื่องมือที่ใช้ในการพัฒนาระบบ ที่สำคัญคือ

1. Rational Rose เป็น visual modeling ถือเป็น case tool สำหรับการวิเคราะห์ และออกแบบเชิงวัตถุที่สนับสนุนการใช้ยูเอ็มแอล และสนับสนุนการนิยามโครงสร้างฐานข้อมูลของออราเคิล
2. Developer 6 ของบริษัทออราเคิล ได้แก่ Form Builder และ Report Builder ใช้ในการออกแบบยูสเซอร์อินเตอร์เฟซ สร้างต้นแบบ จนได้เป็นแอปพลิเคชันซอฟต์แวร์
3. Oracle Server 8.0 เป็นดาตาเบสเซิร์ฟเวอร์ ซึ่งสามารถทำงานบนระบบปฏิบัติการ Window NT

การสร้างระบบต่อเนื่องจากบทที่ 5 ที่เป็นการนำคลาสในบิสซิเนสเลเยอร์มาแปลงเป็นตาราง หรือโครงสร้างฐานข้อมูล และบทที่ 6 เป็นการนำคลาสในวิวเลเยอร์มาออกแบบยูสเซอร์อินเตอร์เฟซ ซึ่งยูสเซอร์อินเตอร์เฟซที่เป็นฟอร์ม และรายงานจำเป็นต้องอาศัยฐานข้อมูลของระบบเป็นพื้นฐานในการออกแบบ และทำงาน ดังนั้นจึงต้องนำไฟล์สคริปต์ schema.sql ที่ได้ในบทที่ 5 มารันเพื่อบริการโครงสร้างฐานข้อมูลไว้ที่ดาตาเบสเซิร์ฟเวอร์ ของระบบเสียก่อน และรันสคริปต์สร้างสโตโปรซีเจอร์ซึ่งได้เตรียมไว้แล้วคือ StoredProcedure.sql ในหัวข้อ 7.1 ต่อจากนี้จึงจะใช้ Developer 6 ในการพัฒนาแอปพลิเคชันซอฟต์แวร์ต่อไป

7.1 การสร้างสตอโปรซีเจอร์

สตอโปรซีเจอร์ (Stored Procedure) สำหรับระบบดังนี้ของบัญชีถูกหน้การเคหะแห่งชาติดีที่จะกล่าวต่อไปเป็นโมดูลที่เขียนขึ้นด้วยภาษา PL/SQL ซึ่งผ่านการแปลภาษาไว้ล่วงหน้าแล้ว และจัดเก็บอยู่ในดาตาเบสเซิร์ฟเวอร์ จะถูกสั่งให้ทำงานในเข้าถึงข้อมูลในฐานข้อมูลในดาตาเบสเซิร์ฟเวอร์ ซึ่งเทียบได้กับการดำเนินการในสถาปัตยกรรมเชิงวัตถุ หรือในที่นี้คือการดำเนินการของคลาสในแอคเซสเลเยอร์ที่ได้ออกแบบไว้

สตอโปรซีเจอร์ในดาตาเบสเซิร์ฟเวอร์ของออราเคิลสามารถเขียนในรูปของแพ็คเกจ ซึ่งประกอบด้วย 2 ส่วนหลัก คือ ส่วนของการกำหนดรูปแบบ และส่วนของเนื้อหา ที่เรียกว่า Package Specification และ Package Body ตามลำดับ ดังรูปที่ 7.1 ซึ่งเป็นรายละเอียดการสร้างสตอโปรซีเจอร์ที่เป็นแพ็คเกจในไฟล์สคริปต์ StoredProcedure.sql โดยจะแยกอธิบายหน้าที่การทำงานของแต่ละแพ็คเกจ ส่วนอัลกอริทึมแต่ละโมดูลในแพ็คเกจได้มีการออกแบบไว้ในบทที่ 4 เป็นการดำเนินการของคลาสในแอคเซสเลเยอร์ โดยที่หนึ่งคลาสคือหนึ่งแพ็คเกจนั่นเอง แต่ได้รวบรวมนำเสนอด้วยแอคตีวิตีไดอะแกรมไว้ในบทนี้ ดังต่อไปนี้

1. แพ็คเกจ Location_Actions

หน้าที่จัดการข้อมูลสำนักงานเคหะชุมชน หรือตาราง Location ซึ่งประกอบด้วยโปรซีเจอร์ (Procedure) และฟังก์ชัน (Function) ดังตารางที่ 7.1 ส่วนโค้ดต้นฉบับดังรูปที่ 7.1 และ 7.2

ตารางที่ 7.1 คำอธิบาย โมดูลในแพ็คเกจ Location_Actions (ส่วนที่ 1/2)

โมดูล	คำอธิบาย
isNewLocation parameter : a_locationID return : BOOLEAN	ตรวจสอบว่ารหัสสำนักงานเคหะชุมชนที่ระบุมีอยู่หรือไม่ หากมีจะให้ค่ากลับเป็น TRUE ถ้าไม่มีจะให้ค่ากลับเป็น FALSE การทำงานดังรูปที่ 7.21
getLocationName parameter : a_locationID return : locationName	หาชื่อสำนักงานเคหะชุมชน ที่มีรหัสสำนักงานตามที่ระบุ หากไม่พบจะให้ค่ากลับเป็น “ไม่พบชื่อ” การทำงานดังรูปที่ 7.22
retrieveLocation parameter : result_set	เข้าถึงข้อมูลสำนักงานเคหะชุมชน โดยมีการส่งพารามิเตอร์แบบ pass by value คือ result_set เป็นชุดข้อมูลสำนักงานเคหะชุมชนทั้งหมด ซึ่งประกาศไว้เป็นชนิด Location_cur (ในส่วน package specification) การทำงานดังรูปที่ 7.23

ตารางที่ 7.1 คำอธิบายโมดูลในแพ็คเกจ Location (ส่วนที่ 2/2)

โมดูล	คำอธิบาย
insertLocation parameter : a_locationID,a_locatinName, a_limit,a_houseNo, a_moo,a_soi, a_road,a_province,a_postalCode	เพิ่มรายการข้อมูลสำนักงานเคหะชุมชน การทำงานดังรูปที่ 7.24
* updateLocation parameter : a_locationID,a_locatinName, a_limit,a_houseNo, a_moo,a_soi, a_road,a_province,a_postalCode	ปรับปรุงรายการข้อมูลสำนักงานเคหะชุมชนที่มีรหัสสำนักงานเคหะชุมชนตามที่ระบุ การทำงานดังรูปที่ 7.25
deleteLocation parameter : a_locationID,a_locatinName, a_limit,a_houseNo, a_moo,a_soi, a_road,a_province,a_postalCode	ลบข้อมูลสำนักงานเคหะชุมชนที่มีรหัสสำนักงานเคหะชุมชนตามที่ระบุ การทำงานดังรูปที่ 7.26

```
CREATE OR REPLACE PACKAGE Location_Actions
IS
    TYPE Location_rec is RECORD (locationID LOCATION.locationID%TYPE,
        locationName LOCATION.locationName%TYPE,
        limit LOCATION.limit%TYPE,
        useno LOCATION.houseno%TYPE,
        moo LOCATION.moo%TYPE,
        soi LOCATION.soi%TYPE,
        road LOCATION.road%TYPE,
        province LOCATION.province%TYPE,
        postalCode LOCATION.postalCode%TYPE);

    TYPE Location_cur is REF CURSOR RETURN Location_rec;

    FUNCTION isNewLocation (a_locationID LOCATION.locationID%TYPE) RETURN BOOLEAN;

    FUNCTION getLocationName(a_locationID LOCATION.locationID%TYPE) RETURN
LOCATION.locationName%TYPE;
```

รูปที่ 7.1 โค้ดต้นฉบับส่วน Specification ของแพ็คเกจ Location_Actions (ส่วนที่ 1/2)

```

PROCEDURE retrieveLocation(result_set IN OUT Location_cur);

PROCEDURE insertLocation(
    a_locationID LOCATION.locationID%TYPE,
    a_locationName LOCATION.locationName%TYPE,
    a_limit LOCATION.limit%TYPE,
    an_houseno LOCATION.houseno%TYPE,
    a_moo LOCATION.moo%TYPE,
    a_soi LOCATION.soi%TYPE,
    a_road LOCATION.road%TYPE,
    a_province LOCATION.province%TYPE,
    a_postalCode LOCATION.postalCode%TYPE);

PROCEDURE updateLocation(
    a_locationID LOCATION.locationID%TYPE,
    a_locationName LOCATION.locationName%TYPE,
    a_limit LOCATION.limit%TYPE,
    an_houseno LOCATION.houseno%TYPE,
    a_moo LOCATION.moo%TYPE,
    a_soi LOCATION.soi%TYPE,
    a_road LOCATION.road%TYPE,
    a_province LOCATION.province%TYPE,
    a_postalCode LOCATION.postalCode%TYPE);

PROCEDURE deleteLocation(a_locationID LOCATION.locationID%TYPE);

END Location_Actions;

/

```

รูปที่ 7.1 โค้ดต้นฉบับส่วน Specification ของแพ็คเกจ Location_Actions (ส่วนที่ 2/2)

```

CREATE OR REPLACE PACKAGE BODY Location_Actions IS
--
FUNCTION isNewLocation (a_locationID LOCATION.locationID%TYPE)
RETURN BOOLEAN
IS
    a_locationName LOCATION.locationName%TYPE;
BEGIN
    SELECT locationName
    INTO a_locationName
    FROM LOCATION
    WHERE
        a_locationID = locationID;
    -- Old record
    RETURN(FALSE);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN(TRUE);
END isNewLocation;
--
FUNCTION getLocationName(a_locationID LOCATION.locationID%TYPE)
RETURN LOCATION.locationName%TYPE
IS
    m_locationName LOCATION.locationName%TYPE;
BEGIN
    SELECT      locationName
    INTO        m_locationName
    FROM        LOCATION
    WHERE       locationID = a_locationID;
    RETURN(m_locationName);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN('ไม่พบชื่อ');
END getLocationName;

```

รูปที่ 7.2 โค้ดต้นฉบับส่วน Body ของแพ็คเกจ Location_Actions (ส่วนที่ 1/4)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
PROCEDURE retrieveLocation(result_set IN OUT Location_cur)
```

```
IS
```

```
BEGIN
```

```
    OPEN result_set FOR
```

```
        SELECT locationID,
```

```
               locationName,
```

```
               limit,
```

```
               houseno,
```

```
               moo,
```

```
               soi,
```

```
               road,
```

```
               province,
```

```
               postalCode
```

```
        FROM LOCATION
```

```
        ORDER BY locationName;
```

```
END retrieveLocation;
```

```
--
```

```
PROCEDURE insertLocation(
```

```
    a_locationID LOCATION.locationID%TYPE,
```

```
    a_locationName LOCATION.locationName%TYPE,
```

```
    a_limit LOCATION.limit%TYPE,
```

```
    an_houseno LOCATION.houseno%TYPE,
```

```
    a_moo LOCATION.moo%TYPE,
```

```
    a_soi LOCATION.soi%TYPE,
```

```
    a_road LOCATION.road%TYPE,
```

```
    a_province LOCATION.province%TYPE,
```

```
    a_postalCode LOCATION.postalCode%TYPE)
```

```
IS
```

```
BEGIN
```

```
    INSERT INTO LOCATION (
```

```
        locationID,
```

```
        locationName,
```

```
        limit,
```

รูปที่ 7.2 โค้ดต้นฉบับส่วน Body ของแพ็คเกจ Location_Actions (ส่วนที่ 2/4)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        houseno,
        moo,
        soi,
        road,
        province,
        postalCode)
VALUES (
        a_locationID,
        a_locationName,
        a_limit,
        an_houseno,
        a_moo,
        a_soi,
        a_road,
        a_province,
        a_postalCode);
COMMIT;
END insertLocation;
--
PROCEDURE updateLocation(
        a_locationID LOCATION.locationID%TYPE,
        a_locationName LOCATION.locationName%TYPE,
        a_limit LOCATION.limit%TYPE,
        an_houseno LOCATION.houseno%TYPE,
        a_moo LOCATION.moo%TYPE,
        a_soi LOCATION.soi%TYPE,
        a_road LOCATION.road%TYPE,
        a_province LOCATION.province%TYPE,
        a_postalCode LOCATION.postalCode%TYPE)
IS
BEGIN
        UPDATE LOCATION
                SET      locationName=a_locationName,

```

รูปที่ 7.2 โค้ดต้นฉบับส่วน Body ของแพ็คเกจ Location_Actions (ส่วนที่ 3/4)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        limit=a_limit ,
        houseno=an_houseno ,
        moo=a_moo ,
        soi=a_soi ,
        road=a_road ,
        province=a_province ,
        postalCode=a_postalCode

WHERE locationID = a_locationID;

COMMIT;

END updateLocation;

--

PROCEDURE deleteLocation(a_locationID LOCATION.locationID%TYPE)
IS
BEGIN
    DELETE LOCATION
    WHERE locationID=a_locationID;
    COMMIT;
END deleteLocation;

--

END Location_Actions;

/
```

รูปที่ 7.2 โค้ดต้นฉบับส่วน Body ของแพ็กเกจ Location_Actions (ส่วนที่ 4/4)

2. แพ็กเกจ Project_Actions

หน้าที่จัดการข้อมูลโครงการ หรือตาราง Project ซึ่งประกอบด้วยโปรซีเจอร์ และ ฟังก์ชัน ดังตารางที่ 7.2 ส่วนโค้ดต้นฉบับดังรูปที่ 7.3 และ 7.4

ตารางที่ 7.2 คำอธิบายโมดูลในแพ็กเกจ Project_Actions (ส่วนที่ 1/2)

โมดูล	คำอธิบาย
isNewProject parameter : a_projectID return : BOOLEAN	ตรวจสอบว่ารหัสโครงการที่ระบุมีอยู่หรือไม่ หากมีจะให้ค่ากลับเป็น TRUE ถ้าไม่มีจะให้ค่ากลับเป็น FALSE การทำงานดังรูปที่ 7.27

ตารางที่ 7.2 คำอธิบายโมดูลในแพ็กเกจ Project_Actions (ส่วนที่ 2/2)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โมดูล	คำอธิบาย
getProjectName parameter : a_projectID return : projectName	หาชื่อสำนักงานเคหะชุมชนที่ตรงกับรหัสโครงการที่ระบุ หากหาไม่พบจะให้ค่ากลับเป็น “ไม่พบชื่อ” การทำงานดังรูปที่ 7.28
getLocationID parameter : a_projectID return : locationID	หารหัสสำนักงานเคหะชุมชนที่โครงการนั้นสังกัดอยู่ การทำงานดังรูปที่ 7.29
retrieveProject parameter : result_set	เข้าถึงข้อมูลโครงการทั้งหมด โดยมีการส่งพารามิเตอร์แบบ pass by value คือ result_set เป็นชุดข้อมูลสำนักงานเคหะชุมชนทั้งหมด ซึ่งประกาศไว้เป็นชนิด Project_cur (ในส่วนของ package specification) การทำงานดังรูปที่ 7.30
insertProject parameter : a_projectID, a_projectName, a_locationID	เพิ่มรายการ โครงการ การทำงานดังรูปที่ 7.31
updateProject parameter : a_projectID, a_projectName, a_locationID	ปรับปรุงการอ้างอิงข้อมูลสำนักงานเคหะชุมชนที่โครงการนี้อยู่ การทำงานดังรูปที่ 7.32
DeleteProject parameter : a_projectID	ลบรายการ โครงการ การทำงานดังรูปที่ 7.33

```

CREATE OR REPLACE PACKAGE Project_Actions IS
TYPE project_rec is RECORD (
    projectID PROJECT.projectID%TYPE,
    projectName PROJECT.projectName%TYPE,
    locationID PROJECT.locationID%TYPE);
TYPE project_cur is REF CURSOR RETURN project_rec;

FUNCTION isNewProject (a_projectID PROJECT.projectID%TYPE) RETURN BOOLEAN;

FUNCTION getProjectName(a_projectID PROJECT.projectID%TYPE) RETURN
PROJECT.projectName%TYPE;

```

รูปที่ 7.3 โค้ดต้นฉบับส่วน Specification ของแพ็คเกจ Project_Actions (ส่วนที่ 1/2)

```

FUNCTION getLocationID(a_projectID PROJECT.projectID%TYPE) RETURN
PROJECT.locationID%TYPE;

PROCEDURE retrieveProject(result_set IN OUT project_cur);

PROCEDURE insertProject(
    a_projectID PROJECT.projectID%TYPE,
    a_projectName PROJECT.projectName%TYPE,
    a_locationID PROJECT.locationID%TYPE);

PROCEDURE updateProject(
    a_projectID PROJECT.projectID%TYPE,
    a_projectName PROJECT.projectName%TYPE,
    a_locationID PROJECT.locationID%TYPE);

PROCEDURE deleteProject(a_projectID PROJECT.projectID%TYPE);

END Project_Actions;
/

```

รูปที่ 7.3 โค้ดต้นฉบับส่วน Specification ของแพ็คเกจ Project_Actions (ส่วนที่ 2/2)

```

CREATE OR REPLACE PACKAGE BODY Project_Actions IS
--
FUNCTION isNewProject (a_projectID PROJECT.projectID%TYPE)
RETURN BOOLEAN
IS
    m_projectName PROJECT.projectName%TYPE;
BEGIN
    SELECT projectName
    INTO m_projectName
    FROM PROJECT
    WHERE projectID = a_projectID;

    --Old record

```

รูปที่ 7.4 โค้ดต้นฉบับส่วน Body ของแพ็คเกจ Project_Actions (ส่วนที่ 1/4) นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        RETURN(FALSE);

    EXCEPTION

        WHEN NO_DATA_FOUND THEN

            RETURN(TRUE);

END isNewProject;

--

FUNCTION getProjectName(a_projectID PROJECT.projectID%TYPE)
RETURN PROJECT.projectName%TYPE
IS
    m_projectName PROJECT.projectName%TYPE;
BEGIN
    SELECT      projectName
    INTO        m_projectName
    FROM        PROJECT
    WHERE       projectID = a_projectID;

    RETURN(m_projectName);

    EXCEPTION

        WHEN NO_DATA_FOUND THEN

            RETURN('ไม่พบชื่อ');

END getProjectName;

--

FUNCTION getLocationID(a_projectID PROJECT.projectID%TYPE)
RETURN PROJECT.locationID%TYPE
IS
    m_locationID PROJECT.locationID%TYPE;
BEGIN
    SELECT locationID
    INTO m_locationID
    FROM PROJECT
    WHERE

```

รูปที่ 7.4 โค้ดต้นฉบับส่วน Body ของแพ็คเกจ Project_Actions (ส่วนที่ 2/4)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        projectID = a_projectID;
        RETURN(m_locationID);
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RETURN('ไม่พบรหัส');
END getLocationID;
--
PROCEDURE retrieveProject(result_set IN OUT project_cur)
IS
BEGIN
    OPEN result_set FOR
        SELECT projectID, projectName, locationID
        FROM PROJECT
        ORDER BY projectName;
END retrieveProject;
--
PROCEDURE insertProject(
    a_projectID PROJECT.projectID%TYPE,
    a_projectName PROJECT.projectName%TYPE,
    a_locationID PROJECT.locationID%TYPE)
IS
BEGIN
    INSERT INTO Project (projectID, projectName, locationID)
    VALUES (a_projectID, a_projectName, a_locationID);
    COMMIT;
END insertProject;
--
PROCEDURE updateProject(
    a_projectID PROJECT.projectID%TYPE,
    a_projectName PROJECT.projectName%TYPE,
    a_locationID PROJECT.locationID%TYPE)
IS
BEGIN

```

รูปที่ 7.4 โค้ดต้นฉบับส่วน Body ของแพ็คเกจ Project_Actions (ส่วนที่ 3/4)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
UPDATE Project
    SET projectName=a_projectName
    ,locationID =a_locationID
    WHERE projectID = a_projectID;
COMMIT;
END updateProject;
--
PROCEDURE deleteProject(a_projectID PROJECT.projectID%TYPE)
IS
BEGIN
    DELETE Project
    WHERE projectID=a_projectID;
    COMMIT;
END deleteProject;
--
END Project_Actions;
/
```

รูปที่ 7.4 โค้ดต้นฉบับส่วน Body ของแพ็กเกจ Project_Actions (ส่วนที่ 4/4)

3. แพ็กเกจ Model_Actions

หน้าที่จัดการข้อมูล โมเดล หรือตาราง Model ซึ่งประกอบด้วยโปรซีเจอร์ และฟังก์ชัน ดังตารางที่ 7.3 ส่วนโค้ดต้นฉบับดังรูปที่ 7.5 และ 7.6

ตารางที่ 7.3 คำอธิบายโมดูลในแพ็กเกจ Model_Actions (ส่วนที่ 1/2)

โมดูล	คำอธิบาย
isNewModel parameter : a_modelID return : BOOLEAN	ตรวจสอบว่ารหัสโมเดลที่ระบุมีอยู่หรือไม่ หากมีจะให้ค่ากลับเป็น TRUE ถ้าไม่มีจะให้ค่ากลับเป็น FALSE การทำงานดังรูปที่ 7.34
getModelName parameter : a_modelID return : modelName	หาชื่อ โมเดลตามรหัสโมเดล การทำงานดังรูปที่ 7.35

ตารางที่ 7.3 คำอธิบายโมดูลในแพ็คเกจ Model_Actions (ส่วนที่ 2/2)

โมดูล	คำอธิบาย
getBuildingTypeID parameter : a_modelID return : buildingTypeID	หารหัสประเภทสิ่งปลูกสร้างของโมเดลตามรหัสโมเดล การทำงานดังรูปที่ 7.36
retrieveModel Parameter : result_set	เข้าถึงข้อมูลโมเดล โดยมีการส่งพารามิเตอร์แบบ pass by value คือ result_set เป็นชุดข้อมูลโมเดลทั้งหมด ซึ่งประกาศไว้เป็นชนิด Model_cur (ในส่วนของ package specification) การทำงานดังรูปที่ 7.37
InsertModel parameter : a_modelID, a_modelName, a_buildingTypeID	เพิ่มข้อมูล โมเดล การทำงานดังรูปที่ 7.38
updateModel Parameter : a_modelID, a_modelName, a_buildingTypeID	แก้ไขข้อมูลชื่อ โมเดล และประเภทสิ่งปลูกสร้าง การทำงานดังรูปที่ 7.39
deleteModel parameter : a_modelID	ลบข้อมูล โมเดล การทำงานดังรูปที่ 7.40

```
CREATE OR REPLACE PACKAGE Model_Actions IS
TYPE model_rec is RECORD (modelID MODEL.modelID%TYPE,
    modelName MODEL.modelName%TYPE,
    buildingTypeID MODEL.buildingTypeID%TYPE);
TYPE model_cur is REF CURSOR RETURN model_rec;

FUNCTION isNewModel (a_modelID MODEL.modelID%TYPE) RETURN BOOLEAN;

FUNCTION getModelName(a_ModelID MODEL.modelID%TYPE) RETURN MODEL.modelName%TYPE
;

FUNCTION getBuildingTypeID(a_modelID MODEL.modelID%TYPE) RETURN
MODEL.buildingTypeID%TYPE;

PROCEDURE retrieveModel(result_set IN OUT model_cur);
```

รูปที่ 7.5 โค้ดต้นฉบับส่วน Specification ของแพ็คเกจ Model_Actions (ส่วนที่ 1/2)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

PROCEDURE insertModel(
    a_modelID MODEL.modelID%TYPE,
    a_modelName MODEL.modelName%TYPE,
    a_buildingTypeID MODEL.buildingTypeID%TYPE);

PROCEDURE updateModel(
    a_modelID MODEL.modelID%TYPE,
    a_modelName MODEL.modelName%TYPE,
    a_buildingTypeID MODEL.buildingTypeID%TYPE);

PROCEDURE deleteModel(a_modelID MODEL.modelID%TYPE);

END Model_Actions;
/

```

รูปที่ 7.5 โค้ดต้นฉบับส่วน Specification ของแพ็คเกจ Model_Actions (ส่วนที่ 2/2)

```

CREATE OR REPLACE PACKAGE BODY Model_Actions IS
--
FUNCTION isNewModel (a_modelID MODEL.modelID%TYPE)
RETURN BOOLEAN
IS
    a_modelName MODEL.modelName%TYPE;
BEGIN
    SELECT modelName
    INTO a_modelName
    FROM model
    WHERE a_modelID = modelID;

    -- Old record
    RETURN(FALSE);

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN(TRUE);

END isNewModel;

```

รูปที่ 7.6 โค้ดต้นฉบับส่วน Body ของแพ็คเกจ Model_Actions (ส่วนที่ 1/4)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

FUNCTION getModelName(a_modelID MODEL.modelID%TYPE)
RETURN MODEL.modelName%TYPE
IS
    m_modelName MODEL.modelName%TYPE;
BEGIN
    SELECT modelName
    INTO m_modelName
    FROM MODEL
    WHERE
        modelID = a_modelID;
        RETURN(m_modelName);
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RETURN('ไม่พบชื่อ');
END getModelName;
--
FUNCTION getBuildingTypeID(a_modelID MODEL.modelID%TYPE)
RETURN MODEL.buildingTypeID%TYPE
IS
    m_buildingTypeID MODEL.buildingTypeID%TYPE;
BEGIN
    SELECT buildingTypeID
    INTO m_buildingTypeID
    FROM MODEL
    WHERE
        modelID = a_modelID;
        RETURN(m_buildingTypeID);
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RETURN('ไม่พบรหัส');
END getBuildingTypeID;

```

รูปที่ 7.6 โค้ดต้นฉบับส่วน Body ของแพ็คเกจ Model_Actions (ส่วนที่ 2/4)

```

PROCEDURE retrieveModel(result_set IN OUT model_cur)
IS
BEGIN
    OPEN result_set FOR
    SELECT modelID, modelName, buildingTypeID
    FROM MODEL
    ORDER BY modelName;
END retrieveModel;

--

PROCEDURE insertModel(
    a_modelID MODEL.modelID%TYPE,
    a_modelName MODEL.modelName%TYPE,
    a_buildingTypeID MODEL.buildingTypeID%TYPE)
IS
BEGIN
    INSERT INTO Model(modelID,modelName,buildingTypeID)
    VALUES (a_modelID,a_modelName,a_buildingTypeID);
    COMMIT;
END insertModel;

--

PROCEDURE updateModel(
    a_modelID MODEL.modelID%TYPE,
    a_modelName MODEL.modelName%TYPE,
    a_buildingTypeID MODEL.buildingTypeID%TYPE)
IS
BEGIN
    UPDATE Model
    SET modelName=a_modelName
        ,buildingTypeID=a_buildingTypeID
    WHERE modelID = a_modelID;
    COMMIT;
END updateModel;

```

รูปที่ 7.6 โค้ดต้นฉบับส่วน Body ของแพ็คเกจ Model_Actions (ส่วนที่ 3/4)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

```

PROCEDURE deleteModel(a_modelID MODEL.modelID%TYPE)
IS
BEGIN
    DELETE Model
    WHERE modelID=a_modelID;
    COMMIT;
END deleteModel;
--
END Model_Actions;
/

```

รูปที่ 7.6 โค้ดต้นฉบับส่วน Body ของแพ็คเกจ Model_Actions (ส่วนที่ 4/4)

4. แพ็คเกจ BuildingType_Actions

หน้าที่จัดการข้อมูลประเภทสิ่งปลูกสร้าง หรือตาราง BuildingType ซึ่งประกอบด้วย
โปรซีเจอร์ และฟังก์ชัน ดังตารางที่ 7.4 ส่วนโค้ดต้นฉบับดังรูปที่ 7.7 และ 7.8

ตารางที่ 7.4 คำอธิบายโมดูลในแพ็คเกจ Building Type_Actions (ส่วนที่ 1/2)

โมดูล	คำอธิบาย
isNewBuildingType parameter : a_buildingTypeID return : BOOLEAN	ตรวจสอบว่ารหัสประเภทสิ่งปลูกสร้างที่ระบุมีอยู่หรือไม่ หากมีจะให้ค่ากลับเป็น TRUE ถ้าไม่มีจะให้ค่ากลับเป็น FALSE การทำงานดังรูปที่ 7.41
getBuildingTypeName parameter : a_locationID return : buildingTypeName	หาชื่อประเภทสิ่งปลูกสร้างตามรหัสที่ระบุมา การทำงานดังรูปที่ 7.42
retrieveBuildingType parameter : result_set	เข้าถึงข้อมูลประเภทสิ่งปลูกสร้าง โดยมีการส่งพารามิเตอร์แบบ pass by value คือ result_set เป็นชุดข้อมูลประเภทสิ่งปลูกสร้าง ซึ่งประกาศไว้เป็นชนิด Location_cur (ในส่วน package specification) การทำงานดังรูปที่ 7.43
insertBuildingType parameter : a_buildingTypeID,a_buildingTypeName	เพิ่มรายการประเภทสิ่งปลูกสร้าง การทำงานดังรูปที่ 7.44
updateBuildingType parameter : a_buildingTypeID,a_buildingTypeName	ปรับปรุงชื่อประเภทสิ่งปลูกสร้างที่มีรหัสประเภทสิ่งปลูกสร้างที่ระบุ การทำงานดังรูปที่ 7.45

ตารางที่ 7.4 คำอธิบายโมดูลในแพ็คเกจ Building Type_Actions (ส่วนที่ 2/2)

โมดูล	คำอธิบาย
deleteBuildingType parameter : a_buildingTypeID	ลบรายการประเภทสิ่งปลูกสร้างที่มีรหัสประเภทสิ่งปลูกสร้างที่ระบุ การทำงานดังรูปที่ 7.46

```

CREATE OR REPLACE PACKAGE BuildingType_Actions IS
TYPE BuildingType_rec is RECORD (
    buildingTypeID BUILDINGTYPE.BuildingTypeID%TYPE,
    buildingTypeName BUILDINGTYPE.buildingTypeName%TYPE);
TYPE BuildingType_cur is REF CURSOR RETURN BuildingType_rec;

FUNCTION isNewBuildingType (a_buildingTypeID BUILDINGTYPE.BuildingTypeID%TYPE) RETURN
BOOLEAN;

FUNCTION getBuildingTypeName(a_BuildingTypeID
BUILDINGTYPE.BuildingTypeID%TYPE) RETURN
BUILDINGTYPE.BuildingTypeName%TYPE;

PROCEDURE retrieveBuildingType(result_set IN OUT BuildingType_cur);

PROCEDURE insertBuildingType(a_buildingTypeID BUILDINGTYPE.BuildingTypeID%TYPE,
    a_buildingTypeName BUILDINGTYPE.BuildingTypeName%TYPE);

PROCEDURE updateBuildingType(a_buildingTypeID BUILDINGTYPE.BuildingTypeID%TYPE,
    a_buildingTypeName BUILDINGTYPE.BuildingTypeName%TYPE);

PROCEDURE deleteBuildingType(a_buildingTypeID BUILDINGTYPE.BuildingTypeID%TYPE);

END BuildingType_Actions;
/

```

รูปที่ 7.7 โค้ดต้นฉบับส่วน Specification ของแพ็คเกจ BuildingType_Actions

```

CREATE OR REPLACE PACKAGE BODY BuildingType_Actions IS
--
FUNCTION isNewBuildingType (a_buildingTypeID BUILDINGTYPE.BuildingTypeID%TYPE)
RETURN BOOLEAN
IS
    m_BuildingTypeName BUILDINGTYPE.BuildingTypeName%TYPE;
BEGIN
    SELECT BuildingTypeName
    INTO m_BuildingTypeName
    FROM BuildingType
    WHERE
        buildingTypeID = a_buildingTypeID;
    -- Old record
    RETURN(FALSE);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN(TRUE);
END isNewBuildingType;
--
FUNCTION getBuildingTypeName(a_BuildingTypeID BUILDINGTYPE.BuildingTypeID%TYPE)
RETURN BUILDINGTYPE.BuildingTypeName%TYPE
IS
    m_BuildingTypeName BUILDINGTYPE.BuildingTypeName%TYPE;
BEGIN
    SELECT BuildingTypeName
    INTO m_BuildingTypeName
    FROM BUILDINGTYPE
    WHERE BuildingTypeID = a_BuildingTypeID;
    RETURN(m_BuildingTypeName);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN('ไม่พบชื่อ');
END getBuildingTypeName;

```

รูปที่ 7.8 โค้ดต้นฉบับส่วน Body ของแพ็คเกจ BuildingType_Actions (ส่วนที่ 1/3)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

PROCEDURE retrieveBuildingType(result_set IN OUT BuildingType_cur)
IS
BEGIN
    OPEN result_set FOR
    SELECT buildingTypeID, buildingTypeName
    FROM BUILDINGTYPE
    ORDER BY buildingTypeName;
END;

--

PROCEDURE insertBuildingType(
    a_buildingTypeID BUILDINGTYPE.BuildingTypeID%TYPE,
    a_buildingTypeName BUILDINGTYPE.BuildingTypeName%TYPE)
IS
BEGIN
    INSERT INTO BuildingType (buildingTypeID,buildingTypeName)
    VALUES (a_buildingTypeID,a_buildingTypeName);
    COMMIT;
END insertBuildingType;

--

PROCEDURE updateBuildingType(
    a_buildingTypeID BUILDINGTYPE.BuildingTypeID%TYPE,
    a_buildingTypeName BUILDINGTYPE.BuildingTypeName%TYPE)
IS
BEGIN
    UPDATE BuildingType
    SET BuildingTypeName=a_BuildingTypeName
    WHERE BuildingTypeID = a_BuildingTypeID;
    COMMIT;
END updateBuildingType;

--

PROCEDURE deleteBuildingType(a_buildingTypeID BUILDINGTYPE.BuildingTypeID%TYPE)
IS
BEGIN

```

รูปที่ 7.8 โค้ดต้นฉบับส่วน Body ของแพ็คเกจ BuildingType_Actions (ส่วนที่ 2/3)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

DELETE BuildingType
WHERE buildingTypeID=a_buildingTypeID;

COMMIT;

END deleteBuildingType;

--

END BuildingType_Actions;

/

```

รูปที่ 7.8 โค้ดต้นฉบับส่วน Body ของแพ็คเกจ BuildingType_Actions (ส่วนที่ 3/3)

5. แพ็คเกจ ComponentItem_Actions

หน้าที่จัดการข้อมูลองค์ประกอบในโมเดล หรือตาราง ComponentItem ซึ่งประกอบด้วยโปรซีเจอร์ และฟังก์ชัน ดังตารางที่ 7.5 ส่วนโค้ดต้นฉบับดังรูปที่ 7.9 และ 7.10

ตารางที่ 7.5 คำอธิบายโมดูลในแพ็คเกจ ComponentItem_Actions (ส่วนที่ 1/2)

โมดูล	คำอธิบาย
isNewComponentItem parameter : a_modelID,a_ComponentItem return : BOOLEAN	ตรวจสอบว่ารายการรหัสองค์ประกอบของโมเดลที่ระบุมีอยู่หรือไม่ หากมีจะให้ค่ากลับเป็น TRUE ถ้าไม่มีจะให้ค่ากลับเป็น FALSE การทำงานดังรูปที่ 7.47
amountOfComponentItems parameter : a_modelID return : NUMBER	คำนวณจำนวนรายการองค์ประกอบซึ่งมีรหัสโมเดลตามที่ระบุ การทำงานดังรูปที่ 7.48
retrieveComponentItem parameter : result_set,a_modelID	เข้าถึงข้อมูลองค์ประกอบในโมเดลที่ระบุ โดยมีการส่งพารามิเตอร์แบบ pass by value คือ result_set เป็นชุดข้อมูลองค์ประกอบในโมเดลทั้งหมด ซึ่งประกาศไว้เป็นชนิด ComponentItem_cur (ในส่วน package specification) การทำงานดังรูปที่ 7.49
InsertComponentItem parameter : a_modelID,a_ComponentItem, a_componentAmount	เพิ่มรายการองค์ประกอบในโมเดล การทำงานดังรูปที่ 7.50
updateComponentItem parameter : a_modelID,a_ComponentItem, a_componentAmount	ปรับปรุงจำนวนองค์ประกอบในโมเดลที่มีรหัสองค์ประกอบ และโมเดลตามที่ระบุ การทำงานดังรูปที่ 7.51

ตารางที่ 7.5 คำอธิบายโมดูลในแพ็คเกจ ComponentItem_Actions (ส่วนที่ 2/2)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โมดูล	คำอธิบาย
deleteComponentItem parameter : a_modelID,a_ComponentItem	ลบรายการรายการองค์ประกอบใน โมเดลที่มีรหัส องค์ประกอบ และ โมเดลตามที่ระบุ การทำงานดังรูปที่ 7.52

```
CREATE OR REPLACE PACKAGE ComponentItem_Actions IS
TYPE ComponentItem_rec is RECORD (modelID MODEL.modelID%TYPE,
    componentID COMPONENT.componentID%TYPE,
    componentQuantity COMPONENTITEM.componentQuantity%TYPE);
TYPE ComponentItem_cur is REF CURSOR RETURN ComponentItem_rec;
FUNCTION isNewComponentItem (a_modelID MODEL.modelID%TYPE,
    a_componentID COMPONENT.componentID%TYPE) RETURN BOOLEAN;

FUNCTION amountOfComponentItems(a_modelID MODEL.modelID%TYPE)RETURN NUMBER;

PROCEDURE retrieveComponentItem(result_set IN OUT ComponentItem_cur,a_modelID
MODEL.modelID%TYPE);

PROCEDURE insertComponentItem(
    a_modelID MODEL.modelID%TYPE,
    a_componentID COMPONENT.componentID%TYPE,
    a_componentQuantity COMPONENTITEM.componentQuantity%TYPE);

PROCEDURE updateComponentItem(
    a_modelID MODEL.modelID%TYPE,
    a_componentID COMPONENT.componentID%TYPE,
    a_componentQuantity COMPONENTITEM.componentQuantity%TYPE);

PROCEDURE deleteComponentItem(
    a_modelID MODEL.modelID%TYPE,
    a_componentID COMPONENT.componentID%TYPE);

END ComponentItem_Actions;
/
```

รูปที่ 7.9 โค้ดต้นฉบับส่วน Specification ของแพ็คเกจ ComponentItem_Actions

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

CREATE OR REPLACE PACKAGE BODY ComponentItem_Actions IS
--
FUNCTION isNewComponentItem (a_modelID MODEL.modelID%TYPE,
                             a_componentID COMPONENT.componentID%TYPE)
RETURN BOOLEAN
IS
    m_componentQuantity COMPONENTITEM.componentQuantity%TYPE;
BEGIN
    SELECT componentQuantity
    INTO m_componentQuantity
    FROM COMPONENTITEM
    WHERE modelID = a_modelID AND
           componentID = a_componentID;
    RETURN(FALSE);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN(TRUE);
END isNewComponentItem;
--
FUNCTION amountOfComponentItems(a_modelID MODEL.modelID%TYPE)
RETURN NUMBER
IS
    m_TotalComponentItem NUMBER := 0;
BEGIN
    SELECT Count(*)
    INTO m_TotalComponentItem
    FROM COMPONENTITEM
    WHERE modelID = a_ModelID;
    RETURN(m_TotalComponentItem);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN(m_TotalComponentItem);
END amountOfComponentItems;

```

รูปที่ 7.10 โค้ดต้นฉบับส่วน Body ของแพ็คเกจ ComponentItem_Actions (ส่วนที่ 1/3)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

PROCEDURE retrieveComponentItem(result_set IN OUT ComponentItem_cur,a_modelID
MODEL.modelID%TYPE)
IS
BEGIN
    OPEN result_set FOR
    SELECT modelID,componentID, componentQuantity
    FROM COMPONENTITEM
    WHERE modelID = a_modelID
    ORDER BY componentID;
END retrieveComponentItem;
--
PROCEDURE insertComponentItem(
    a_modelID MODEL.modelID%TYPE,
    a_componentID COMPONENT.componentID%TYPE,
    a_componentQuantity
COMPONENTITEM.componentQuantity%TYPE)
IS
BEGIN
    INSERT INTO componentitem (modelID,componentID,componentQuantity)
    VALUES (a_modelID,a_componentID,a_componentQuantity);
    COMMIT;
END insertComponentItem;
--
PROCEDURE updateComponentItem(
    a_modelID MODEL.modelID%TYPE,
    a_componentID COMPONENT.componentID%TYPE,
    a_componentQuantity
COMPONENTITEM.componentQuantity%TYPE)
IS
BEGIN
    UPDATE Componentitem
    SET componentQuantity= a_componentQuantity
    WHERE modelID = a_modelID AND

```

รูปที่ 7.10 โค้ดต้นฉบับส่วน Body ของแพ็คเกจ ComponentItem_Actions (ส่วนที่ 2/3)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        componentID = a_componentID;

    COMMIT;

END updateComponentItem;

--

PROCEDURE deleteComponentItem(

        a_modelID MODEL.modelID%TYPE,

        a_componentID COMPONENT.componentID%TYPE)

IS

BEGIN

    DELETE componentitem

    WHERE modelID = a_modelID AND

        componentID = a_componentID;

    COMMIT;

END deleteComponentItem;

--

END ComponentItem_Actions;

/
```

รูปที่ 7.10 โค้ดต้นฉบับส่วน Body ของแพ็คเกจ ComponentItem_Actions (ส่วนที่ 3/3)

6. แพ็คเกจ Component_Actions

หน้าที่จัดการข้อมูลองค์ประกอบ หรือตาราง Component ซึ่งประกอบด้วยโปรซีเจอร์ และฟังก์ชัน ดังตารางที่ 7.6 ส่วนโค้ดต้นฉบับดังรูปที่ 7.11 และ 7.12

ตารางที่ 7.6 คำอธิบายโมดูลในแพ็คเกจ Component_Actions (ส่วนที่ 1/2)

โมดูล	คำอธิบาย
isNewComponent parameter : a_ComponentID return : BOOLEAN	ตรวจสอบว่ารหัสองค์ประกอบที่ระบุมีอยู่หรือไม่ หากมีจะให้ค่ากลับเป็น TRUE ถ้าไม่มีจะให้ค่ากลับเป็น FALSE การทำงานดังรูปที่ 7.53
getComponentName parameter : a_coponentID return : componentName	หาชื่อองค์ประกอบตามรหัสองค์ประกอบที่ระบุ การทำงานดังรูปที่ 7.54

ตารางที่ 7.6 คำอธิบายโมดูลในแพ็คเกจ Component_Actions (ส่วนที่ 2/2)

โมดูล	คำอธิบาย
retrieveComponent parameter : result_set	เข้าถึงข้อมูลองค์ประกอบทั้งหมด โดยมีการส่งพารามิเตอร์แบบ pass by value คือ result_set เป็นชุดข้อมูลองค์ประกอบทั้งหมด ซึ่งประกาศไว้เป็นชนิด Component_cur (ในส่วน package specification) การทำงานดังรูปที่ 7.55
InsertComponent parameter : a_ComponentID, a_componentName	เพิ่มรายการองค์ประกอบ การทำงานดังรูปที่ 7.56
updateComponent parameter : a_ComponentID, a_componentName	ปรับปรุงชื่อองค์ประกอบที่มีรหัสสององค์ประกอบ ตามที่ระบุ การทำงานดังรูปที่ 7.57
deleteComponent parameter : a_ComponentID	ลบรายการองค์ประกอบที่มีรหัสสององค์ประกอบตามที่ระบุ การทำงานดังรูปที่ 7.58

```
CREATE OR REPLACE PACKAGE Component_Actions IS
TYPE component_rec is RECORD (componentID COMPONENT.componentID%TYPE,
    componentName COMPONENT.componentName%TYPE);
TYPE component_cur is REF CURSOR RETURN component_rec;

FUNCTION isNewComponent (a_componentID COMPONENT.componentID%TYPE) RETURN
BOOLEAN;

FUNCTION GetComponentName(a_componentID COMPONENT.componentID%TYPE) RETURN
COMPONENT.componentName%TYPE;

PROCEDURE retrieveComponent(result_set IN OUT component_cur);

PROCEDURE insertComponent(
    a_componentID COMPONENT.componentID%TYPE,
    a_componentName COMPONENT.componentName%TYPE);

PROCEDURE updateComponent(a_componentID COMPONENT.componentID%TYPE,
    a_componentName COMPONENT.componentName%TYPE);
```

รูปที่ 7.11 โค้ดต้นฉบับส่วน Specification ของแพ็คเกจ Component_Actions (ส่วนที่ 1/2)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
PROCEDURE deleteComponent(a_componentID COMPONENT.componentID%TYPE);
```

```
END Component_Actions;
```

```
/
```

รูปที่ 7.11 โค้ดต้นฉบับส่วน Specification ของแพ็คเกจ Component_Actions (ส่วนที่ 2/2)

```
CREATE OR REPLACE PACKAGE BODY Component_Actions IS
```

```
--
```

```
FUNCTION isNewComponent (a_componentID COMPONENT.componentID%TYPE) RETURN BOOLEAN  
IS
```

```
  m_componentName COMPONENT.componentName%TYPE;
```

```
BEGIN
```

```
  SELECT componentName
```

```
  INTO m_componentName
```

```
  FROM COMPONENT
```

```
  WHERE
```

```
    componentID = a_componentID;
```

```
  -- Old record
```

```
    RETURN(FALSE);
```

```
EXCEPTION
```

```
  WHEN NO_DATA_FOUND THEN
```

```
    RETURN(TRUE);
```

```
END isNewComponent;
```

```
--
```

```
FUNCTION GetComponentName(a_componentID COMPONENT.componentID%TYPE)
```

```
RETURN COMPONENT.componentName%TYPE
```

```
IS
```

```
  m_componentName COMPONENT.componentName%TYPE;
```

```
BEGIN
```

```
  SELECT componentName
```

```
  INTO m_componentName
```

```
  FROM COMPONENT
```

รูปที่ 7.12 โค้ดต้นฉบับส่วน Body ของแพ็คเกจ Component_Actions (ส่วนที่ 1/3)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

WHERE

        componentID = a_componentID;

        RETURN(m_componentName);

    EXCEPTION

        WHEN NO_DATA_FOUND THEN

            RETURN('ไม่พบชื่อ');

END GetComponentName;

--

PROCEDURE retrieveComponent(result_set IN OUT component_cur)
IS
BEGIN
    OPEN result_set FOR
    SELECT componentID, componentName
    FROM COMPONENT
    ORDER BY componentName;
END retrieveComponent;

--

PROCEDURE insertComponent(
    a_componentID COMPONENT.componentID%TYPE,
    a_componentName COMPONENT.componentName%TYPE)
IS
BEGIN
    INSERT INTO component (componentID, componentName)
    VALUES (a_componentID, a_componentName);

    COMMIT;
END insertComponent;

--

PROCEDURE updateComponent(
    a_componentID COMPONENT.componentID%TYPE,
    a_componentName COMPONENT.componentName%TYPE)
IS
BEGIN
    UPDATE Component

```

รูปที่ 7.12 โค้ดต้นฉบับส่วน Body ของแพ็คเกจ Component_Actions (ส่วนที่ 2/3)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
SET componentName= a_componentName
WHERE componentID = a_componentID;
COMMIT;
END updateComponent;
--
PROCEDURE deleteComponent(a_componentID COMPONENT.componentID%TYPE)
IS
BEGIN
    DELETE component
    WHERE componentID=a_componentID;
    COMMIT;
END deleteComponent;
END Component_Actions;
/
```

รูปที่ 7.12 โค้ดต้นฉบับส่วน Body ของแพ็กเกจ Component_Actions (ส่วนที่ 3/3)

7. แพ็กเกจ Quotation_Actions

หน้าที่จัดการข้อมูลขายโครงการ หรือตาราง Quotation ซึ่งประกอบด้วยโปรซีเจอร์ และฟังก์ชัน ดังตารางที่ 7.7 ส่วนโค้ดต้นฉบับดังรูปที่ 7.13 และ 7.14

ตารางที่ 7.7 คำอธิบายโมดูลในแพ็กเกจ Quotation_Actions (ส่วนที่ 1/2)

โมดูล	คำอธิบาย
isNewQuotationItems parameter : a_projectID,a_modelID return : BOOLEAN	ตรวจสอบว่ามีรายการขายที่เป็นโมเดลและของโครงการที่ระบุหรือไม่ ถ้ามีจะให้ค่ากลับเป็น TRUE ถ้าไม่มีจะให้ค่ากลับเป็น FALSE การทำงานดังรูปที่ 7.59
hasQuotationItems parameter : a_projectID return : BOOLEAN	ตรวจสอบว่ามีรายการขายของโครงการที่ระบุหรือไม่ ถ้ามีจะให้ค่ากลับเป็น TRUE ถ้าไม่มีจะให้ค่ากลับเป็น FALSE การทำงานดังรูปที่ 7.60
amountOfQuotationItems parameter : a_projectID return : NUMBER	คำนวณหาจำนวนรายการขายของโครงการที่ระบุ การทำงานดังรูปที่ 7.61

ตารางที่ 7.7 คำอธิบายโมดูลในแฟ้มเกจ Quotation_Actions (ส่วนที่ 2/2)

โมดูล	คำอธิบาย
retrieveQuotation parameter : result_set,a_projectID	เข้าถึงข้อมูลขายของโครงการที่ระบุ โดยมีการส่งพารามิเตอร์แบบ pass by value คือ result_set เป็นชุดข้อมูลรายการขายของโครงการที่ระบุทั้งหมด ซึ่งประกาศไว้เป็นชนิด Quotation_cur (ใน ส่วน package specification) การทำงานดังรูปที่ 7.62
insertQuotationItem parameter : a_projectID, a_modelID,an_UnitAmount, a_space, a_cost, a_standardInterest	เพิ่มรายการขาย การทำงานดังรูปที่ 7.63
updateQuotationItem parameter : a_projectID, a_modelID, an_UnitAmount, a_space, a_cost, a_standardInterest	ปรับปรุงข้อมูล จำนวนหน่วยอาสัย เนื้อที่ต่อหน่วยราคาขายเงินสด และอัตราดอกเบี้ยการผ่อนชำระค่าเช่าซื้อต่อปี ของโมเดลในโครงการที่ระบุ การทำงานดังรูปที่ 7.64
deleteQuotationItem parameter : a_projectID, a_modelID	ลบรายการขายที่มีโมเดล และโครงการตามที่ระบุ การทำงานดังรูปที่ 7.65

```
CREATE OR REPLACE PACKAGE Quotation_Actions IS
TYPE quotation_rec is RECORD (
projectID PROJECT.projectID%TYPE,
modelID MODEL.modelID%TYPE,
unitAmount QUOTATION.unitAmount%TYPE,
space QUOTATION.space%TYPE,
cost QUOTATION.cost%TYPE,
standardInterest QUOTATION.standardInterest%TYPE);
TYPE quotation_cur is REF CURSOR RETURN quotation_rec;
FUNCTION isNewQuotationItems (
a_projectID QUOTATION.projectID%TYPE,
a_ModelID QUOTATION.modelID%TYPE) RETURN BOOLEAN;
FUNCTION hasQuotationItems (a_projectID QUOTATION.projectID%TYPE) RETURN BOOLEAN;
```

รูปที่ 7.13 โค้ดต้นฉบับส่วน Specification ของแฟ้มเกจ Quotation_Actions (ส่วนที่ 1/2)

```

FUNCTION amountOfQuotationItems (a_projectID QUOTATION.projectID%TYPE) RETURN NUMBER;

PROCEDURE retrieveQuotation(result_set IN OUT quotation_cur,a_projectID
QUOTATION.projectID%TYPE);

PROCEDURE insertQuotationItem(a_projectID PROJECT.projectID%TYPE,
                               a_modelID MODEL.modelID%TYPE,
                               an_unitAmount QUOTATION.unitAmount%TYPE,
                               a_space QUOTATION.space%TYPE,
                               a_cost QUOTATION.cost%TYPE,
                               a_standardInterest QUOTATION.standardInterest%TYPE);

PROCEDURE updateQuotationItem(a_projectID PROJECT.projectID%TYPE,
                               a_modelID MODEL.modelID%TYPE,
                               an_unitAmount QUOTATION.unitAmount%TYPE,
                               a_space QUOTATION.space%TYPE,
                               a_cost QUOTATION.cost%TYPE,
                               a_standardInterest QUOTATION.standardInterest%TYPE);

PROCEDURE deleteQuotationItem(a_projectID PROJECT.projectID%TYPE,
                               a_modelID MODEL.modelID%TYPE);

END Quotation_Actions;

/

```

รูปที่ 7.13 โค้ดต้นฉบับส่วน Specification ของแพ็คเกจ Quotation_Actions (ส่วนที่ 2/2)

```

CREATE OR REPLACE PACKAGE BODY Quotation_Actions IS
    m_amount NUMBER := 0;
    --
    FUNCTION isNewQuotationItems (
        a_projectID QUOTATION.projectID%TYPE,
        a_ModelID QUOTATION.modelID%TYPE)
    RETURN BOOLEAN
    IS
        m_modelID QUOTATION.modelID%TYPE;
    BEGIN
        SELECT modelID
        INTO m_modelID
        FROM QUOTATION
        WHERE
            a_projectID = projectID AND
            a_modelID = modelID;
        -- have some records
        RETURN(FALSE);

    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RETURN(TRUE);
    END isNewQuotationItems;
    --
    FUNCTION hasQuotationItems (a_projectID QUOTATION.projectID%TYPE) RETURN BOOLEAN IS
    BEGIN
        SELECT count(*)
        INTO m_amount
        FROM QUOTATION
        WHERE
            a_projectID = projectID;
        -- have some records
        RETURN(TRUE);
    
```

รูปที่ 7.14 โค้ดต้นฉบับส่วน Body ของแพ็คเกจ Quotation_Actions (ส่วนที่ 1/4)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN(FALSE);
END hasQuotationItems;
--
FUNCTION amountOfQuotationItems (a_projectID QUOTATION.projectID%TYPE)
^
RETURN NUMBER
IS
BEGIN
    SELECT count(*)
    INTO m_amount
    FROM QUOTATION
    WHERE
        a_projectID = projectID;
    -- have some records
    RETURN(m_amount);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN(m_amount);
END amountOfQuotationItems;
--
PROCEDURE retrieveQuotation(result_set IN OUT quotation_cur,a_projectID
QUOTATION.projectID%TYPE)
IS
BEGIN
    OPEN result_set FOR
    SELECT projectID,modelID,unitAmount,space,cost,standardInterest
    FROM QUOTATION
    WHERE projectID = a_projectID
    ORDER BY modelID;
END retrieveQuotation;
--

```

รูปที่ 7.14 โค้ดต้นฉบับส่วน Body ของแพ็คเกจ Quotation_Actions (ส่วนที่ 2/4)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

PROCEDURE insertQuotationItem(a_projectID PROJECT.projectID%TYPE,
                             a_modelID MODEL.modelID%TYPE,

                             an_unitAmount QUOTATION.unitAmount%TYPE,

                             a_space QUOTATION.space%TYPE,
                             a_cost QUOTATION.cost%TYPE,
                             a_standardInterest QUOTATION.standardInterest%TYPE)
IS
BEGIN
    INSERT INTO QUOTATION(
        projectID,
        modelID,
        unitAmount,
        space,
        cost,
        standardInterest)
    VALUES (
        a_projectID,
        a_modelID,
        an_unitAmount,
        a_space,
        a_cost,
        a_standardInterest);

    COMMIT;
END insertQuotationItem;
--
PROCEDURE updateQuotationItem(a_projectID PROJECT.projectID%TYPE,
                              a_modelID MODEL.modelID%TYPE,
                              an_unitAmount QUOTATION.unitAmount%TYPE,
                              a_space QUOTATION.space%TYPE,
                              a_cost QUOTATION.cost%TYPE,
                              a_standardInterest QUOTATION.standardInterest%TYPE)

```

รูปที่ 7.14 โค้ดต้นฉบับส่วน Body ของแพ็คเกจ Quotation_Actions (ส่วนที่ 3/4)

```

IS
BEGIN
    UPDATE QUOTATION
    SET      unitAmount = an_unitAmount,
            space = a_space,
            cost = a_cost,
            standardInterest = a_standardInterest
    WHERE
        projectID = a_projectID AND
        modelID = a_modelID;
    COMMIT;
END updateQuotationItem;
--
PROCEDURE deleteQuotationItem(a_projectID PROJECT.projectID%TYPE,
                             a_modelID MODEL.modelID%TYPE)
IS
BEGIN
    DELETE QUOTATION
    WHERE projectID = a_projectID AND
          modelID = a_modelID;
    COMMIT;
END deleteQuotationItem;
--
END Quotation_Actions;
/

```

รูปที่ 7.14 โค้ดต้นฉบับส่วน Body ของแพ็คเกจ Quotation_Actions (ส่วนที่ 4/4)

8. แพ็กเกจ DownPayment_Actions

หน้าที่จัดการข้อมูลการผ่อนชำระล่วงหน้า หรือตาราง DownPayment ซึ่งประกอบด้วยโปรซีเจอร์ และฟังก์ชัน ดังตารางที่ 7.8 ส่วนโค้ดต้นฉบับดังรูปที่ 7.15 และ 7.16

ตารางที่ 7.8 คำอธิบายโมดูลในแพ็กเกจ DownPayment_Actions

โมดูล	คำอธิบาย
isNewDownPayment parameter : a_projectID, a_modelID, a_fromInstalment return : BOOLEAN	ตรวจสอบว่ามีรายการค่าผ่อนชำระล่วงหน้าจากงวดที่ถึงงวดของโมเดล และโครงการที่ระบุหรือไม่ ถ้ามีจะให้ค่ากลับเป็น TRUE ถ้าไม่มีจะให้ค่ากลับเป็น FALSE การทำงานดังรูปที่ 7.66
amountOfDownPayment parameter : a_projectID, a_modelID return : NUMBER	คำนวณหาจำนวนรายการการผ่อนชำระล่วงหน้าของโมเดลและโครงการที่ระบุ การทำงานดังรูปที่ 7.67
retrieveDownPayment parameter : result_set, a_projectID, a_modelID	เข้าถึงรายการผ่อนชำระล่วงหน้าของ โมเดล และโครงการที่ระบุ โดยมีการส่งพารามิเตอร์แบบ pass by value คือ result_set เป็นชุดข้อมูลการผ่อนชำระ ซึ่งประกาศไว้เป็นชนิด DownPayment_cur (ในส่วนของ package specification) การทำงานดังรูปที่ 7.68
insertDownPayment parameter : a_projectID, a_modelID, a_fromInstalment, a_toInstalment, a_downAmount	เพิ่มรายการค่าผ่อนชำระล่วงหน้าจากงวดที่ถึงงวดของโมเดล และโครงการที่ระบุ การทำงานดังรูปที่ 7.69
updateDownPayment parameter : a_projectID, a_modelID, a_fromInstalment, a_toInstalment, a_downAmount	ปรับปรุงข้อมูล ถึงงวดที่ และค่าผ่อนชำระล่วงหน้า ของจากงวดที่ของโมเดลในโครงการที่ระบุ การทำงานดังรูปที่ 7.70
deleteDownPayment parameter : a_projectID, a_modelID, a_fromInstalment	ลบรายการค่าผ่อนชำระล่วงหน้า ของจากงวดที่ของโมเดลในโครงการที่ระบุ การทำงานดังรูปที่ 7.71

```

CREATE OR REPLACE PACKAGE DownPayment_Actions IS

TYPE DownPayment_rec is RECORD (projectID PROJECT.projectID%TYPE,

                                modelID MODEL.modelID%TYPE,

                                fromInstalment DOWNPAYMENT.fromInstalment%TYPE,

                                toInstalment DOWNPAYMENT.toInstalment%TYPE,

                                downAmount DOWNPAYMENT.downAmount%TYPE);

TYPE DownPayment_cur is REF CURSOR RETURN DownPayment_rec;

FUNCTION isNewDownPayment (a_projectID PROJECT.projectID%TYPE,
a_ModelID MODEL.modelID%TYPE,
a_fromInstalment DOWNPAYMENT.fromInstalment%TYPE) RETURN BOOLEAN;

FUNCTION amountOfDownPayment(a_projectID PROJECT.projectID%TYPE,
a_modelID MODEL.modelID%TYPE) RETURN NUMBER;

PROCEDURE retrieveDownPayment(
result_set IN OUT downpayment_cur,
a_projectID PROJECT.projectID%TYPE,
a_ModelID MODEL.modelID%TYPE);

PROCEDURE insertDownPayment(a_projectID PROJECT.projectID%TYPE,
a_modelID MODEL.modelID%TYPE,
a_fromInstalment DOWNPAYMENT.fromInstalment%TYPE,
a_toInstalment DOWNPAYMENT.toInstalment%TYPE,
a_downAmount DOWNPAYMENT.downAmount%TYPE);

PROCEDURE updateDownPayment(a_projectID PROJECT.projectID%TYPE,
a_modelID MODEL.modelID%TYPE,
a_fromInstalment DOWNPAYMENT.fromInstalment%TYPE,
a_toInstalment DOWNPAYMENT.toInstalment%TYPE,
a_downAmount DOWNPAYMENT.downAmount%TYPE);

```

รูปที่ 7.15 โค้ดต้นฉบับส่วน Specification ของแพ็คเกจ DownPayment_Actions (ส่วนที่ 1/2)

```

PROCEDURE deleteDownPayment(a_projectID PROJECT.projectID%TYPE,
                             a_modelID MODEL.modelID%TYPE,
                             a_fromInstalment DOWNPAYMENT.fromInstalment%TYPE);

END DownPayment_Actions;
/

```

รูปที่ 7.15 โค้ดต้นฉบับส่วน Specification ของแพ็คเกจ DownPayment_Actions (ส่วนที่ 2/2)

```

CREATE OR REPLACE PACKAGE BODY DownPayment_Actions IS
--
FUNCTION isNewDownPayment (a_projectID PROJECT.projectID%TYPE,
                           a_ModelID MODEL.modelID%TYPE,
                           a_fromInstalment DOWNPAYMENT.fromInstalment%TYPE)
RETURN BOOLEAN
IS
    m_modelID MODEL.modelID%TYPE;
BEGIN
    SELECT modelID
    INTO m_modelID
    FROM DOWNPAYMENT
    WHERE projectID = a_projectID AND
           modelID = a_modelID AND
           fromInstalment = a_fromInstalment;

    -- have some records
    RETURN(FALSE);

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN(TRUE);
END isNewDownPayment;
--

```

รูปที่ 7.16 โค้ดต้นฉบับส่วน Body ของแพ็คเกจ DownPayment_Actions (ส่วนที่ 1/4)

```

FUNCTION amountOfDownPayment(a_projectID PROJECT.projectID%TYPE,
                             a_modelID MODEL.modelID%TYPE) RETURN NUMBER
IS
    m_amount NUMBER:=0;
BEGIN
    SELECT count(*)
    INTO m_amount
    FROM DOWNPAYMENT
    WHERE
        projectID      = a_projectID
        AND modelID     = a_modelID;
    -- have some records
    RETURN(m_amount);

    EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN(m_amount);
END amountOfDownPayment;
--
PROCEDURE retrieveDownPayment(
    result_set IN OUT downpayment_cur,
    a_projectID PROJECT.projectID%TYPE,
    a_ModelID MODEL.modelID%TYPE)
IS
BEGIN
    OPEN result_set FOR
    SELECT projectID,modelID,fromInstalment,toInstalment,downAmount
    FROM DOWNPAYMENT
    WHERE projectID = a_projectID AND
        modelID = a_modelID
    ORDER BY projectID,modelId,fromInstalment;
END retrieveDownPayment;
--

```

รูปที่ 7.16 โค้ดต้นฉบับส่วน Body ของแพ็คเกจ DownPayment_Actions (ส่วนที่ 2/4)

```

PROCEDURE insertDownPayment(a_projectID PROJECT.projectID%TYPE,
                           a_modelID MODEL.modelID%TYPE,
                           a_fromInstalment DOWNPAYMENT.fromInstalment%TYPE;
                           a_toInstalment DOWNPAYMENT.toInstalment%TYPE,
                           a_downAmount DOWNPAYMENT.downAmount%TYPE)
IS
BEGIN
    INSERT INTO DOWNPAYMENT(projectID,
                           modelID,
                           fromInstalment,
                           toInstalment,
                           downAmount)
    VALUES (
        a_projectID,
        a_modelID,
        a_fromInstalment,
        a_toInstalment,
        a_downAmount);

    COMMIT;
END insertDownPayment;
--
PROCEDURE updateDownPayment(a_projectID PROJECT.projectID%TYPE,
                           a_modelID MODEL.modelID%TYPE,
                           a_fromInstalment DOWNPAYMENT.fromInstalment%TYPE,
                           a_toInstalment DOWNPAYMENT.toInstalment%TYPE,
                           a_downAmount DOWNPAYMENT.downAmount%TYPE)
IS
BEGIN
    UPDATE DOWNPAYMENT
    SET     toInstalment = a_toInstalment,
           downAmount = a_downAmount
    WHERE
        projectID = a_projectID AND

```

รูปที่ 7.16 โค้ดต้นฉบับส่วน Body ของแพ็คเกจ DownPayment_Actions (ส่วนที่ 3/4)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        modelID = a_modelID AND
        fromInstalment = a_fromInstalment;

    COMMIT;
END updateDownPayment;
--

PROCEDURE deleteDownPayment(a_projectID PROJECT.projectID%TYPE,
                             a_modelID MODEL.modelID%TYPE,
                             a_fromInstalment DOWNPAYMENT.fromInstalment%TYPE)
IS
BEGIN
    DELETE DOWNPAYMENT
    WHERE projectID = a_projectID AND
           modelID = a_modelID AND
           fromInstalment = a_fromInstalment;

    COMMIT;
END deleteDownPayment;
--
END DownPayment_Actions;
/

```

รูปที่ 7.16 โค้ดต้นฉบับส่วน Body ของแพ็กเกจ DownPayment_Actions (ส่วนที่ 4/4)

9. แพ็กเกจ NewProjectHouseNo_Actions

หน้าที่จัดการข้อมูลเตรียมบ้านเลขที่ หรือตาราง NewProjectHouseNo ซึ่งประกอบด้วย โปรซีเจอร์ และฟังก์ชัน ดังตารางที่ 7.9 ส่วนโค้ดต้นฉบับดังรูปที่ 7.17 และ 7.18

ตารางที่ 7.9 คำอธิบายโมดูลในแพ็กเกจ NewProjectHouseNo_Actions (ส่วนที่ 1/3)

โมดูล	คำอธิบาย
isNewProjectHouseNo parameter : a_projectID,a_modelID,beginHouseNo return : BOOLEAN	ตรวจสอบว่ามีรายการบ้านเลขที่ ที่มีบ้านเลขที่เริ่มต้น โมเดล และโครงการตามที่ระบุหรือไม่ ถ้ามีจะให้ค่ากลับเป็น TRUE ถ้าไม่มีจะให้ค่ากลับเป็น FALSE การทำงานดังรูปที่ 7.72

ตารางที่ 7.9 คำอธิบายโมดูลในแพ็คเกจ NewProjectHouseNo_Actions (ส่วนที่ 2/3)

โมดูล	คำอธิบาย
isOverlapWith parameter : a_beginHouseNo,an_ascendent, an_unitAmount,a_beginHouseNo_2, an_ascendent_2,an_unitAmount_2	ตรวจสอบว่าช่วงบ้านเลขที่ที่กำหนดซ้ำซ้อนกันกับที่ เคยกำหนดไว้หรือไม่ หากซ้ำซ้อนกันจะให้ค่ากลับ เป็นจริง ถ้าไม่จะให้ค่ากลับเป็นเท็จ การทำงานดังรูปที่ 7.73
amountOfNewProjectHouseNo parameter : a_projectID,a_modelID return : NUMBER	ค้นหาจำนวนรายการกำหนดบ้านเลขที่ของ โครงการ และโมเดลที่ระบุ การทำงานดังรูปที่ 7.74
retrieveHouseNo parameter : result_set,a_projectID,a_modelID, beginHouseNo	เข้าถึงข้อมูลบ้านเลขที่ของโครงการ โมเดลที่ระบุ ที่ มีบ้านเลขที่เริ่มต้นไม่เป็นไปตามที่ระบุ ซึ่งจะใช้ใน การตรวจสอบการซ้ำซ้อนของบ้านเลขที่ โดยมีการ ส่งพารามิเตอร์แบบ pass by value คือ result_set เป็นชุดข้อมูลเตรียมบ้านเลขที่ ซึ่งประกาศไว้เป็น ชนิด NewProjectHouseNo_cur (ในส่วน package specification) การทำงานดังรูปที่ 7.75
retrieveProject parameter : result_set,a_projectID	เข้าถึงข้อมูลบ้านเลขที่ของโครงการที่ระบุ โดยมีการ ส่งพารามิเตอร์แบบ pass by value คือ result_set เป็นชุดข้อมูลเตรียมบ้านเลขที่ ซึ่งประกาศไว้เป็น ชนิด NewProjectHouseNo_cur (ในส่วน package specification) การทำงานดังรูปที่ 7.76
setIsGenerated parameter : a_projectID,a_modelID, a_beginHouseNo,isGenerated	กำหนดค่าให้กับแอททริบิวต์ isGenerated โดยที่ 1 แทน การนำไปใช้แล้ว ส่วน 0 แทนยังไม่มี การนำ รายการเลขที่บ้านไปใช้ การทำงานดังรูปที่ 7.77
insertProjectHouseNo parameter : a_projectID, a_modelID, a_beginHouseNo, unitAmount, an_assendent	เพิ่มรายการบ้านเลขที่ การทำงานดังรูปที่ 7.78
updateProjectHouseNo parameter : a_projectID, a_modelID, a_beginHouseNo, unitAmount, an_assendent	ปรับปรุงข้อมูล ค่าเพิ่ม และจำนวนหน่วยอาศัย การทำงานดังรูปที่ 7.79

ตารางที่ 7.9 คำอธิบายโมดูลในแพ็คเกจ NewProjectHouseNo_Actions (ส่วนที่ 3/3)

โมดูล	คำอธิบาย
deleteProjectHouseNo parameter : a_projectID, a_modelID, a_beginHouseNo	ลบรายการบ้านเลขที่ การทำงานดังรูปที่ 7.80

```

CREATE OR REPLACE PACKAGE NewProjectHouseNo_Actions IS
TYPE NewProjectHouseNo_rec is RECORD (projectID PROJECT.projectID%TYPE,
                                         modelID MODEL.modelID%TYPE,
                                         beginHouseNo
NEWPROJECTHOUSENO.beginHouseNo%TYPE,
                                         unitAmount NEWPROJECTHOUSENO.unitAmount%TYPE,
                                         ascendent NEWPROJECTHOUSENO.ascendent%TYPE,
                                         isGenerated NEWPROJECTHOUSENO.isGenerated%TYPE);
TYPE NewProjectHouseNo_cur is REF CURSOR RETURN NewProjectHouseNo_rec;
TYPE HouseNo_cur is REF CURSOR RETURN NewProjectHouseNo_rec;

FUNCTION isNewProjectHouseNo (a_projectID PROJECT.projectID%TYPE,
                              a_modelID MODEL.modelID%TYPE,
                              a_beginHouseNo NEWPROJECTHOUSENO.beginHouseNo%TYPE)
RETURN BOOLEAN;

FUNCTION isOverlapWith(a_beginHouseNo NEWPROJECTHOUSENO.beginHouseNo%TYPE,
                      an_ascendent    NEWPROJECTHOUSENO.ascendent%TYPE,
                      an_unitAmount   NEWPROJECTHOUSENO.unitAmount%TYPE,
                      a_beginHouseNo_2 NEWPROJECTHOUSENO.beginHouseNo%TYPE,
                      an_ascendent_2  NEWPROJECTHOUSENO.ascendent%TYPE,
                      an_unitAmount_2 NEWPROJECTHOUSENO.unitAmount%TYPE) RETURN BOOLEAN
;

```

รูปที่ 7.17 โค้ดต้นฉบับส่วน Specification ของแพ็คเกจ NewProjectHouseNo_Actions (ส่วนที่ 1/2)

```

FUNCTION amountOfNewProjectHouseNoModel(a_projectID PROJECT.projectID%TYPE,
                                         a_modelID MODEL.modelID%TYPE) RETURN NUMBER;

PROCEDURE retrieveProject(result_set IN OUT newprojecthouse_no_cur,
                          a_projectID NEWPROJECTHOUSENO.projectID%TYPE);

PROCEDURE retrieveHouseNo(result_set2 IN OUT HouseNo_cur,
                           a_projectID NEWPROJECTHOUSENO.projectID%TYPE,
                           a_modelID MODEL.modelID%TYPE,
                           a_beginHouseNo NEWPROJECTHOUSENO.beginHouseNo%TYPE);

PROCEDURE setIsGenerated (a_projectID PROJECT.projectID%TYPE,
                           a_modelID MODEL.modelID%TYPE,
                           a_beginHouseNo NEWPROJECTHOUSENO.beginHouseNo%TYPE,
                           an_isGenerated NEWPROJECTHOUSENO.isGenerated%TYPE);

PROCEDURE insertProjectHouseNo(a_projectID PROJECT.projectID%TYPE,
                                a_modelID MODEL.modelID%TYPE,
                                a_beginHouseNo NEWPROJECTHOUSENO.beginHouseNo%TYPE,
                                an_unitAmount NEWPROJECTHOUSENO.unitAmount%TYPE,
                                an_ascendent NEWPROJECTHOUSENO.ascendent%TYPE);

PROCEDURE updateProjectHouseNo(a_projectID PROJECT.projectID%TYPE,
                                a_modelID MODEL.modelID%TYPE,
                                a_beginHouseNo NEWPROJECTHOUSENO.beginHouseNo%TYPE,
                                an_unitAmount NEWPROJECTHOUSENO.unitAmount%TYPE,
                                an_ascendent NEWPROJECTHOUSENO.ascendent%TYPE);

PROCEDURE deleteProjectHouseNo(a_projectID PROJECT.projectID%TYPE,
                                a_modelID MODEL.modelID%TYPE,
                                a_beginHouseNo NEWPROJECTHOUSENO.beginHouseNo%TYPE);

END NewProjectHouseNo_Actions;

/

```

รูปที่ 7.17 โค้ดต้นฉบับส่วน Specification ของแพ็คเกจ NewProjectHouseNo_Actions (ส่วนที่ 2/2)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

CREATE OR REPLACE PACKAGE BODY NewProjectHouseNo_Actions IS
--
FUNCTION isNewProjectHouseNo (a_projectID PROJECT.projectID%TYPE,
                                a_modelID MODEL.modelID%TYPE,
                                a_beginHouseNo
NEWPROJECTHOUSENO.beginHouseNo%TYPE)
RETURN BOOLEAN
IS
    m_modelID MODEL.modelID%TYPE;
BEGIN
    SELECT modelID
    INTO m_modelID
    FROM NEWPROJECTHOUSENO
    WHERE projectID = a_projectID AND
          modelID = a_modelID AND
          beginHouseNo = a_beginHouseNo;
    -- have some records
    RETURN(FALSE);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN(TRUE);
END isNewProjectHouseNo;
--
FUNCTION isOverlapWith(a_beginHouseNo NEWPROJECTHOUSENO.beginHouseNo%TYPE,
                        an_ascendent NEWPROJECTHOUSENO.ascendent%TYPE,
                        an_unitAmount NEWPROJECTHOUSENO.unitAmount%TYPE,
                        a_beginHouseNo_2 NEWPROJECTHOUSENO.beginHouseNo%TYPE,
                        an_ascendent_2 NEWPROJECTHOUSENO.ascendent%TYPE,
                        an_unitAmount_2 NEWPROJECTHOUSENO.unitAmount%TYPE) RETURN
BOOLEAN
IS
    max_i NUMBER := an_unitAmount;
    max_j NUMBER := an_unitAmount_2;

```

รูปที่ 7.18 โค้ดต้นฉบับส่วน Body ของแพ็คเกจ NewProjectHouseNo_Actions (ส่วนที่ 1/7)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

m_ascendent1 NUMBER := To_Number(SubStr(an_ascendent,1,4));
m_ascendent2 NUMBER := To_Number(SubStr(an_ascendent,5,4));
m_currentHouseNo1 NUMBER := To_Number(SubStr(a_beginHouseNo,1,4));
m_currentHouseNo2 NUMBER := To_Number(SubStr(a_beginHouseNo,5,4));
m_houseNo UNIT.houseNo%TYPE := "";
m_part1 VARCHAR2(4):="";
m_part2 VARCHAR2(4):="";
--for overlab with
m_ascendent1_2 NUMBER := To_Number(SubStr(an_ascendent_2,1,4));
m_ascendent2_2 NUMBER := To_Number(SubStr(an_ascendent_2,5,4));
m_currentHouseNo1_2 NUMBER := To_Number(SubStr(a_beginHouseNo_2,1,4));
m_currentHouseNo2_2 NUMBER := To_Number(SubStr(a_beginHouseNo_2,5,4));
m_houseNo_2 UNIT.houseNo%TYPE := "";
m_part1_2 VARCHAR2(4):="";
m_part2_2 VARCHAR2(4):="";
--set overlab variable for return value
overlab BOOLEAN := FALSE;

BEGIN
FOR i IN 1..max_i LOOP
    IF i > 1 THEN
        m_currentHouseNo1 := m_currentHouseNo1 + m_ascendent1;
        m_currentHouseNo2 := m_currentHouseNo2 + m_ascendent2;
    END IF;

    m_part1 := SubStr(To_Char(m_currentHouseNo1 + 10000),2,4);
    m_part2 := SubStr(To_Char(m_currentHouseNo2 + 10000),2,4);

    m_houseNo := m_part1 || '/' || m_part2;

    --initial compare house no
    m_ascendent1_2 := To_Number(SubStr(an_ascendent_2,1,4));
    m_ascendent2_2 := To_Number(SubStr(an_ascendent_2,5,4));

```

รูปที่ 7.18 โค้ดต้นฉบับส่วน Body ของแพ็คเกจ NewProjectHouseNo_Actions (ส่วนที่ 2/7)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

m_currentHouseNo1_2 := To_Number(SubStr(a_beginHouseNo_2,1,4));
m_currentHouseNo2_2 := To_Number(SubStr(a_beginHouseNo_2,5,4));
m_houseNo_2 := "";
m_part1_2 := "";
m_part2_2 := "";

overlab := FALSE;
--loop for compare
FOR j IN 1..max_j LOOP
    IF j > 1 THEN
        m_currentHouseNo1_2 := m_currentHouseNo1_2 + m_ascendent1_2;
        m_currentHouseNo2_2 := m_currentHouseNo2_2 + m_ascendent2_2;
    END IF;

    m_part1_2 := SubStr(To_Char(m_currentHouseNo1_2 + 10000),2,4);
    m_part2_2 := SubStr(To_Char(m_currentHouseNo2_2 + 10000),2,4);

    m_houseNo_2 := m_part1_2 || '/' || m_part2_2;
    IF m_houseNo = m_houseNo_2 THEN
        overlab := TRUE;
        RETURN(overlab);
    ELSE
        overlab := FALSE;
    END IF;
END LOOP; -- loop j
END LOOP; -- loop i
RETURN(overlab);
END isOverlapWith;
--
FUNCTION amountOfNewProjectHouseNoModel(a_projectID PROJECT.projectID%TYPE,
                                         a_modelID MODEL.modelID%TYPE)
RETURN NUMBER
IS

```

รูปที่ 7.18 โค้ดต้นฉบับส่วน Body ของฟังก์ชัน NewProjectHouseNo_Actions (ส่วนที่ 3/7)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        m_amount NUMBER:=0;
BEGIN
    SELECT count(*)
    INTO m_amount
    FROM NEWPROJECTHOUSENO
    WHERE
        a_projectID      = projectID AND
        a_modelID = modelID;
    -- have some records
    RETURN(m_amount);

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN(m_amount);
END amountOfNewProjectHouseNoModel;
--
PROCEDURE retrieveProject(
    result_set IN OUT newprojecthouseno_cur,
    a_projectID NEWPROJECTHOUSENO.projectID%TYPE)
IS
BEGIN
    OPEN result_set FOR
    SELECT projectID,modelID,beginHouseNo,unitAmount,ascendent,isGenerated
    FROM NewProjectHouseNo
    WHERE projectID = a_projectID
    ORDER BY modelID;
END retrieveProject;
--
PROCEDURE retrieveHouseNo(result_set2 IN OUT HouseNo_cur,
    a_projectID NEWPROJECTHOUSENO.projectID%TYPE,
    a_modelID MODEL.modelID%TYPE,
    a_beginHouseNo NEWPROJECTHOUSENO.beginHouseNo%TYPE)
IS

```

รูปที่ 7.18 โค้ดต้นฉบับส่วน Body ของแพ็คเกจ NewProjectHouseNo_Actions (ส่วนที่ 4/7)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

BEGIN
    OPEN result_set2 FOR
    SELECT projectID,modelID,beginHouseNo,unitAmount,ascendent,isGenerated
    FROM NewProjectHouseNo
    WHERE projectID = a_projectID AND
           modelID = a_modelID AND
           beginHouseNo != a_beginHouseNo
    ORDER BY beginHouseNo;
END retrieveHouseNo;

--
PROCEDURE setIsGenerated (a_projectID PROJECT.projectID%TYPE,
                          a_modelID MODEL.modelID%TYPE
                          a_beginHouseNo NEWPROJECTHOUSENO.beginHouseNo%TYPE,
                          an_isGenerated NEWPROJECTHOUSENO.isGenerated%TYPE)
IS
BEGIN
    UPDATE NEWPROJECTHOUSENO
    SET isGenerated = an_isGenerated
    WHERE projectID = a_projectID AND
           modelID = a_modelID AND
           beginHouseNo = a_beginHouseNo;

    COMMIT;
END setIsGenerated;

--
PROCEDURE insertProjectHouseNo(a_projectID PROJECT.projectID%TYPE,
                               a_modelID MODEL.modelID%TYPE,
                               a_beginHouseNo NEWPROJECTHOUSENO.beginHouseNo%TYPE,
                               an_unitAmount NEWPROJECTHOUSENO.unitAmount%TYPE,
                               an_ascendent NEWPROJECTHOUSENO.ascendent%TYPE)
IS
BEGIN
    INSERT INTO NEWPROJECTHOUSENO(
                                   projectID,

```

รูปที่ 7.18 โค้ดต้นฉบับส่วน Body ของแพ็คเกจ NewProjectHouseNo_Actions (ส่วนที่ 5/7)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

                                modellID,
                                /
                                beginHouseNo,
                                unitAmount,
                                ascendent)
VALUES (
                                a_projectID,
                                a_modelID,
                                a_beginHouseNo,
                                an_unitAmount,
                                an_ascendent);

COMMIT;
END insertProjectHouseNo;
--
PROCEDURE updateProjectHouseNo(a_projectID PROJECT.projectID%TYPE,
                                a_modelID MODEL.modelID%TYPE,
                                a_beginHouseNo NEWPROJECTHOUSENO.beginHouseNo%TYPE,
                                an_unitAmount NEWPROJECTHOUSENO.unitAmount%TYPE,
                                an_ascendent NEWPROJECTHOUSENO.ascendent%TYPE)
IS
BEGIN
    UPDATE NEWPROJECTHOUSENO
    SET ascendent = an_ascendent,
        unitAmount = an_unitAmount
    WHERE projectID = a_projectID AND
        modelID = a_modelID AND
        beginHouseNo = a_beginHouseNo;

    COMMIT;
END updateProjectHouseNo;
--
PROCEDURE deleteProjectHouseNo(a_projectID PROJECT.projectID%TYPE,
                                a_modelID MODEL.modelID%TYPE,
                                a_beginHouseNo
NEWPROJECTHOUSENO.beginHouseNo%TYPE)
IS

```

รูปที่ 7.18 โค้ดต้นฉบับส่วน Body ของแพ็คเกจ NewProjectHouseNo_Actions (ส่วนที่ 6/7)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
BEGIN
    DELETE NEWPROJECTHOUSENO
    WHERE projectID = a_projectID AND
           modelID = a_modelID AND
           beginHouseNo = a_beginHouseNo;
    COMMIT;
END deleteProjectHouseNo;
--
END NewProjectHouseNo_Actions;
/
```

รูปที่ 7.18 โค้ดต้นฉบับส่วน Body ของแพ็คเกจ NewProjectHouseNo_Actions (ส่วนที่ 7/7)

10. แพ็คเกจ Unit_Actions

หน้าที่จัดการข้อมูลหน่วยอาศัย หรือตาราง Unit ซึ่งประกอบด้วยโปรซีเจอร์ และ ฟังก์ชัน ดังตารางที่ 7.10 ส่วนโค้ดต้นฉบับดังรูปที่ 7.19 และ 7.20

ตารางที่ 7.10 คำอธิบายโมดูลในแพ็คเกจ Unit_Actions (ส่วนที่ 1/2)

โมดูล	คำอธิบาย
isNewUnit parameter : a_projectID,a_modelID, a_HouseNo return : BOOLEAN	ตรวจสอบว่ามีหน่วยอาศัยที่มีบ้านเลขที่ โมเดลของโครงการตามที่ระบุหรือไม่ ถ้ามีจะให้ค่ากลับเป็น TRUE ถ้าไม่มีจะให้ค่ากลับเป็น FALSE การทำงานดังรูปที่ 7.81
sumAmount parameter : a_projectID,a_modelID return : NUMBER	คำนวณจำนวนหน่วยอาศัยที่มีโมเดล และโครงการตามที่ระบุ การทำงานดังรูปที่ 7.82
amountOfProjectUnit parameter : a_projectID return : NUMBER	คำนวณหาจำนวนหน่วยอาศัยของโครงการที่ระบุ การทำงานดังรูปที่ 7.83
fromatedDebtorCode parameter : a_debtorCode return : VARCHAR2	ให้รหัสลูกหนี้ที่อยู่ในรูปแบบโครงการ-โมเดล-บ้านเลขที่ คือ xxxx-xx-xxxx/xxxx การทำงานดังรูปที่ 7.84

ตารางที่ 7.10 คำอธิบายโมดูลในแพ็คเกจ Unit_Actions (ส่วนที่ 2/2)

โมดูล	คำอธิบาย
retrieveProjectUnit parameter : result_set,a_projectID	เข้าถึงข้อมูลบ้านเลขที่ของทั้งโครงการที่ระบุ โดยมี การส่งพารามิเตอร์แบบ pass by value คือ result_set เป็นชุดข้อมูลหน่วยอาศัย ซึ่งประกาศไว้เป็นชนิด Unit_cur (ใน ส่วน package specification) การทำงานดังรูปที่ 7.85
initialDebtorCode parameter : a_projectID	ตั้งหนี้โครงการ โดยกำหนดรหัสลูกหนี้ให้กับแต่ละ หน่วยอาศัยทั้งหมดในโครงการที่ระบุ การทำงานดังรูปที่ 7.86
generateProjectHouseNo parameter : a_projectID,a_modelID,an_unitAmount, a_beginHouseNo,an_ascendent return : -	สร้างรายการหน่วยอาศัยของโครงการ ที่เป็น โมเดล ตามที่ระบุ ทั้งหมดจำนวนหน่วยอาศัยที่ระบุ โดยเริ่ม จากบ้านเลขที่ที่ระบุ ที่มีค่าเพิ่มตามที่ระบุ การทำงานดังรูปที่ 7.87
insertUnit parameter : a_projectID,a_modelID,a_HouseNo	เพิ่มรายการ ในหน่วยอาศัยให้มีบ้านเลขที่ โมเดลของ โครงการตามที่ระบุ การทำงานดังรูปที่ 7.88
deleteUnit parameter : a_projectID, a_modelID, HouseNo	ลบหน่วยอาศัยของโครงการ ที่เป็น โมเดล และบ้าน เลขที่ตามที่ระบุ การทำงานดังรูปที่ 7.89
deleteProjectHouseNo parameter : a_projectID, a_modelID, a_beginHouseNo	ลบหน่วยอาศัยที่อยู่ในช่วงบ้านเลขที่เริ่มต้น จำนวน หน่วย ค่าเพิ่ม ที่เป็น โมเดลของโครงการที่ระบุ การทำงานดังรูปที่ 7.90

CREATE OR REPLACE PACKAGE Unit_Actions IS

TYPE Unit_rec is RECORD (projectID PROJECT.projectID%TYPE,
modelID MODEL.modelID%TYPE,
houseNo UNIT.houseNo%TYPE,
debtorCode UNIT.debtorCode%TYPE,
buildingNo UNIT.buildingNo%TYPE);

TYPE Unit_cur is REF CURSOR RETURN Unit_rec;

FUNCTION isNewUnit(a_projectID PROJECT.projectID%TYPE,
a_modelID MODEL.modelID%TYPE,
a_houseNo UNIT.houseNo%TYPE) RETURN BOOLEAN;

รูปที่ 7.19 โค้ดต้นฉบับส่วน Specification ของแพ็คเกจ Unit_Actions (ส่วนที่ 1/3)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

FUNCTION sumAmount(
    a_projectID PROJECT.projectID%TYPE,
    a_modelID MODEL.modelID%TYPE)
    RETURN NUMBER;

FUNCTION amountOfProjectUnit(a_projectID PROJECT.projectID%TYPE) RETURN NUMBER;

FUNCTION formattedDebtorCode(a_debtorCode UNIT.debtorCode%TYPE) RETURN VARCHAR2

PROCEDURE retrieveProjectUnit(result_set IN OUT Unit_cur,a_projectID PROJECT.projectID%TYPE);

PROCEDURE initialDebtorCode(a_projectID PROJECT.projectID%TYPE);

PROCEDURE generateProjectHouseNo(a_projectID PROJECT.projectID%TYPE,
    a_modelID MODEL.modelID%TYPE,
    an_unitAmount
NEWPROJECTHOUSENO.unitAmount%TYPE,
    a_beginHouseNo
NEWPROJECTHOUSENO.beginHouseNo%TYPE,
    an_ascendent NEWPROJECTHOUSENO.ascendent%TYPE);

PROCEDURE insertUnit(
    a_projectID PROJECT.projectID%TYPE,
    a_modelID MODEL.modelID%TYPE,
    a_houseNo UNIT.houseNo%TYPE);

PROCEDURE deleteUnit(
    a_projectID PROJECT.projectID%TYPE,
    a_modelID MODEL.modelID%TYPE,
    a_houseNo UNIT.houseNo%TYPE);

PROCEDURE deleteProjectHouseNo(a_projectID PROJECT.projectID%TYPE,
    a_modelID MODEL.modelID%TYPE,

```

รูปที่ 7.19 โค้ดต้นฉบับส่วน Specification ของแพ็คเกจ Unit_Actions (ส่วนที่ 2/3)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

                                a_beginHouseNo
NEWPROJECTHOUSENO.beginHouseNo%TYPE,
                                an_unitAmount NEWPROJECTHOUSENO.unitAmount%TYPE,
                                an_ascendent NEWPROJECTHOUSENO.ascendent%TYPE);
END Unit_Actions;
/

```

รูปที่ 7.19 โค้ดต้นฉบับส่วน Specification ของแพ็คเกจ Unit_Actions (ส่วนที่ 3/3)

```

CREATE OR REPLACE PACKAGE BODY Unit_Actions IS
--
FUNCTION isNewUnit(a_projectID PROJECT.projectID%TYPE,
                  a_modelID MODEL.modelID%TYPE,
                  a_houseNo UNIT.houseNo%TYPE)
RETURN BOOLEAN
IS
    m_projectID PROJECT.projectID%TYPE;
BEGIN
    SELECT projectID
    INTO m_projectID
    FROM UNIT
    WHERE projectID = a_projectID
        AND modelID = a_modelID
        AND houseNo = a_houseNo;
    RETURN(FALSE);
EXCEPTION
    WHEN No_Data_Found THEN
        RETURN(TRUE);
END isNewUnit;
--
FUNCTION sumAmount(
                  a_projectID PROJECT.projectID%TYPE,
                  a_modelID MODEL.modelID%TYPE)
RETURN NUMBER

```

รูปที่ 7.20 โค้ดต้นฉบับส่วน Body ของแพ็คเกจ Unit_Actions (ส่วนที่ 1/6)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

IS
    m_sumAmount NUMBER := 0;
BEGIN
    SELECT count(*)
    INTO m_sumAmount
    FROM UNIT
    WHERE projectID = a_projectID
        AND modelID = a_modelID;
        RETURN(m_sumAmount);
EXCEPTION
    WHEN No_Data_Found THEN
        RETURN(0);
END;
--
FUNCTION amountOfProjectUnit(a_projectID PROJECT.projectID%TYPE) RETURN NUMBER
IS
    m_amount NUMBER:=0;
BEGIN
    SELECT count(*)
    INTO m_amount
    FROM UNIT
    WHERE
        projectID = a_projectID;
        -- have some records
        RETURN(m_amount);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN(0);
END amountOfProjectUnit;
--

```

รูปที่ 7.20 โค้ดต้นฉบับส่วน Body ของแพ็คเกจ Unit_Actions (ส่วนที่ 2/6)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

FUNCTION formattedDebtorCode(a_debtorCode UNIT.debtorCode%TYPE) RETURN VARCHAR2
IS
    m_formatted VARCHAR2(17);
BEGIN
    m_formatted := SubStr(a_debtorCode,1,4)|| '-' || SubStr(a_debtorCode,5,2)|| '-' || SubStr
(a_debtorCode,7,4)|| '/' || SubStr(a_debtorCode,11,4);
    RETURN(m_formatted);
END formattedDebtorCode;
--
PROCEDURE retrieveProjectUnit(result_set IN OUT Unit_cur,a_projectID PROJECT.projectID%TYPE)
IS
BEGIN
    OPEN result_set FOR
    SELECT projectID,modelID,houseNo,debtorCode,buildingNo
    FROM UNIT
    WHERE projectID = a_projectID
    ORDER BY projectID,modelID,houseNo;
END retrieveProjectUnit;
--
PROCEDURE initialDebtorCode(a_projectID PROJECT.projectID%TYPE)
IS
BEGIN
    UPDATE UNIT
    SET debtorCode = projectID || modelID || SubStr(houseNo,1,4)|| SubStr(houseNo,6,4)
    WHERE projectID = a_projectID
        AND debtorCode IS NULL;
    COMMIT;
END initialDebtorCode;
--
PROCEDURE generateProjectHouseNo(a_projectID PROJECT.projectID%TYPE,
                                a_modelID MODEL.modelID%TYPE,
                                an_unitAmount NEWPROJECTHOUSENO.unitAmount%TYPE,

```

รูปที่ 7.20 โค้ดต้นฉบับส่วน Body ของแฟกเกจ Unit_Actions (ส่วนที่ 3/6)

เอกสารนี้เป็นเอกสารสงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

IS

```

a_beginHouseNo NEWPROJECTHOUSENO.beginHouseNo%TYPE,
an_ascendent NEWPROJECTHOUSENO.ascendent%TYPE)

```

```

i      NUMBER:=0;
max_i NUMBER := an_unitAmount;
m_ascendent1 NUMBER := To_Number(SubStr(an_ascendent,1,4));
m_ascendent2 NUMBER := To_Number(SubStr(an_ascendent,5,4));
m_currentHouseNo1 NUMBER := To_Number(SubStr(a_beginHouseNo,1,4));
m_currentHouseNo2 NUMBER := To_Number(SubStr(a_beginHouseNo,5,4));
m_houseNo UNIT.houseNo%TYPE := "";
m_part1 VARCHAR2(4):="";
m_part2 VARCHAR2(4):="";

```

BEGIN

FOR i IN 1..max_i LOOP

IF i > 1 THEN

```

m_currentHouseNo1 := m_currentHouseNo1 + m_ascendent1;

```

```

m_currentHouseNo2 := m_currentHouseNo2 + m_ascendent2;

```

END IF;

```

m_part1 := SubStr(To_Char(m_currentHouseNo1 + 10000),2,4);

```

```

m_part2 := SubStr(To_Char(m_currentHouseNo2 + 10000),2,4);

```

```

m_houseNo := m_part1 || '/' || m_part2;

```

IF Unit_Actions.isNewUnit(a_projectId,a_modelID,m_houseNo) THEN

```

Unit_Actions.insertUnit(a_projectId,a_modelID,m_houseNo

```

END IF;

END LOOP;

END generateProjectHouseNo;

รูปที่ 7.20 โค้ดต้นฉบับส่วน Body ของแพ็คเกจ Unit_Actions (ส่วนที่ 4/6)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

PROCEDURE insertUnit(a_projectID PROJECT.projectID%TYPE,
                    a_modelID MODEL.modelID%TYPE,
                    a_houseNo UNIT.houseNo%TYPE)
IS
BEGIN
    INSERT INTO UNIT (projectID,modelID,houseNo)
        VALUES(a_projectID,a_modelID,a_houseNo);
    COMMIT;
END insertUnit;

--
PROCEDURE deleteUnit(a_projectID PROJECT.projectID%TYPE,
                    a_modelID MODEL.modelID%TYPE,
                    a_houseNo UNIT.houseNo%TYPE)
IS
BEGIN
    DELETE UNIT
    WHERE projectID = a_projectID
        AND modelID = a_modelID
        AND houseNo = a_houseNo;
    COMMIT;
END deleteUnit;

--
PROCEDURE deleteProjectHouseNo(a_projectID PROJECT.projectID%TYPE,
                                a_modelID MODEL.modelID%TYPE,
                                a_beginHouseNo NEWPROJECTHOUSENO.beginHouseNo%TYPE,
                                an_unitAmount NEWPROJECTHOUSENO.unitAmount%TYPE,
                                an_ascendent NEWPROJECTHOUSENO.ascendent%TYPE)
IS
    i          NUMBER:=0;
    max_i NUMBER := an_unitAmount;
    m_ascendent1 NUMBER := To_Number(SubStr(an_ascendent,1,4));
    m_ascendent2 NUMBER := To_Number(SubStr(an_ascendent,5,4));
    m_currentHouseNo1 NUMBER := To_Number(SubStr(a_beginHouseNo,1,4));

```

รูปที่ 7.20 โค้ดต้นฉบับส่วน Body ของแพ็คเกจ Unit_Actions (ส่วนที่ 5/6)

เอกสารนี้เป็นเอกสารทบทวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

m_currentHouseNo2 NUMBER := To_Number(SubStr(a_beginHouseNo,5,4));
m_houseNo UNIT.houseNo%TYPE := "";
m_part1 VARCHAR2(4):="";
m_part2 VARCHAR2(4):="";
BEGIN
  FOR i IN 1..max_i LOOP
    IF i > 1 THEN
      m_currentHouseNo1 := m_currentHouseNo1 + m_ascendent1;
      m_currentHouseNo2 := m_currentHouseNo2 + m_ascendent2;
    END IF;

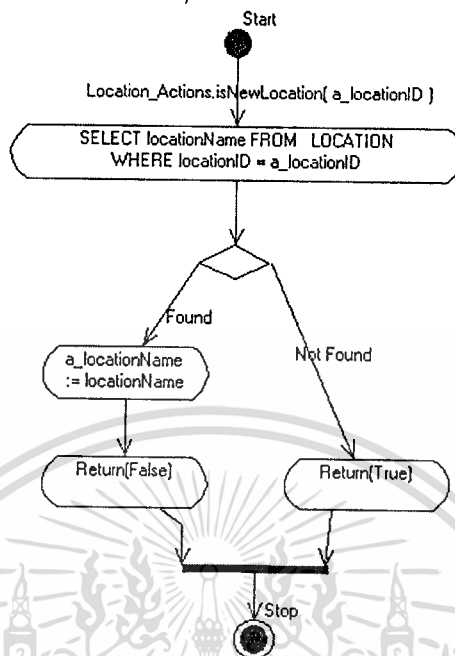
    m_part1 := SubStr(To_Char(m_currentHouseNo1 + 10000),2,4);
    m_part2 := SubStr(To_Char(m_currentHouseNo2 + 10000),2,4);

    m_houseNo := m_part1 || '/' || m_part2;

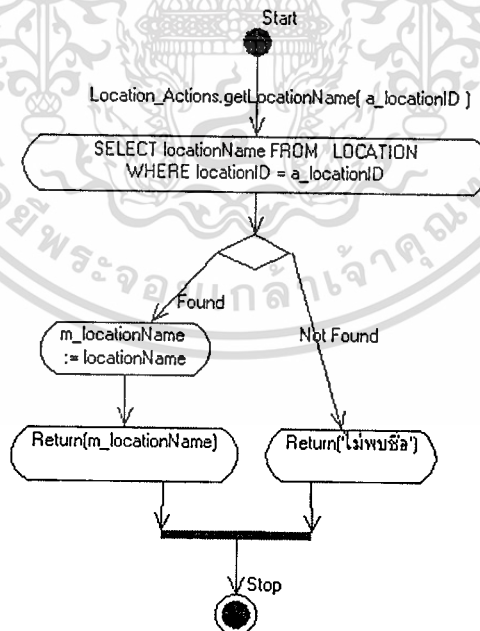
    IF Unit_Actions.isNewUnit(a_projectId,a_modelID,m_houseNo)= FALSE THEN
      Unit_Actions.deleteUnit(a_projectId,a_modelID,m_houseNo);
    END IF;
  END LOOP;
END deleteProjectHouseNo;
--
END Unit_Actions;
/

```

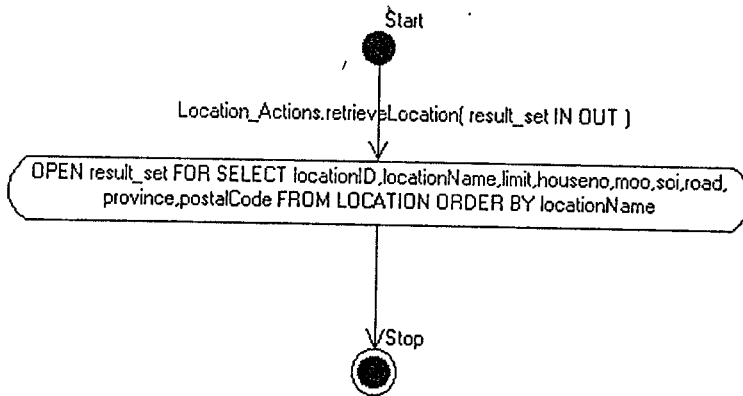
รูปที่ 7.20 โค้ดต้นฉบับส่วน Body ของแพ็คเกจ Unit_Actions (ส่วนที่ 6/6)



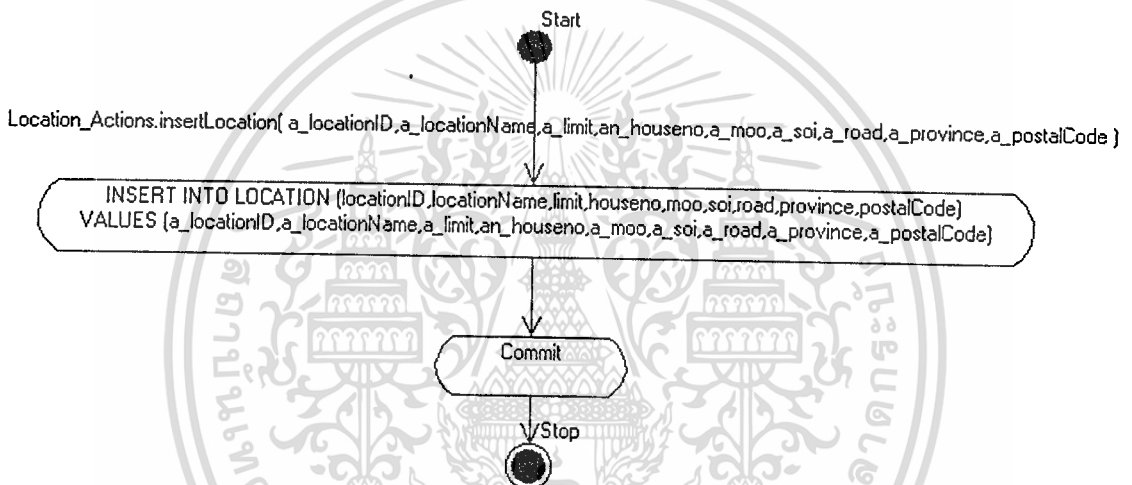
รูปที่ 7.21 แอคตีวิตี้ไดอะแกรมของโมดูล `Location_Actions.isNewLocation`



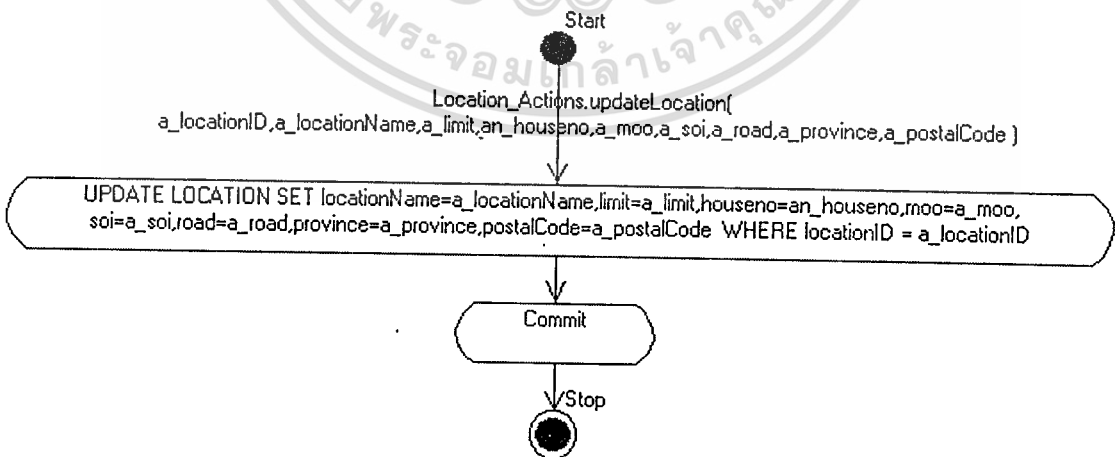
รูปที่ 7.22 แอคตีวิตี้ไดอะแกรมของโมดูล `Location_Actions.getLocationName`



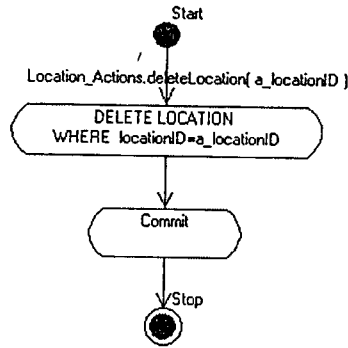
รูปที่ 7.23 แอคตีวิตี้ไดอะแกรมของโมดูล `Location_Actions.retrieveLocation`



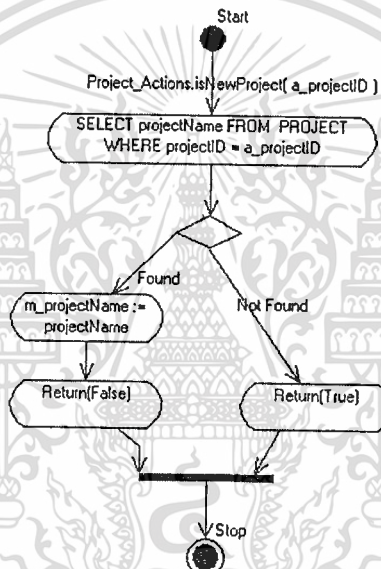
รูปที่ 7.24 แอคตีวิตี้ไดอะแกรมของโมดูล `Location_Actions.insertLocation`



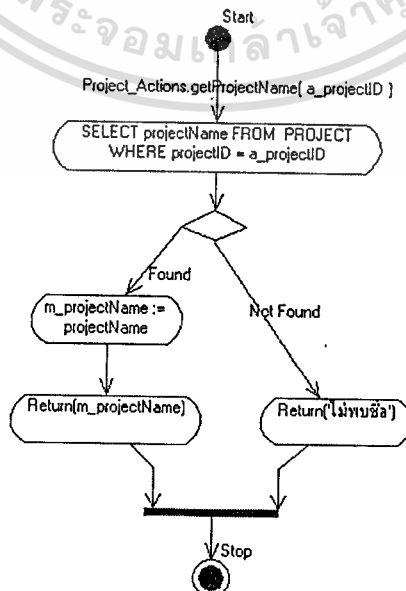
รูปที่ 7.25 แอคตีวิตี้ไดอะแกรมของโมดูล `Location_Actions.updateLocation`



รูปที่ 7.26 แอคตวิทีไคอะแกรมของโมดูล Location_Actions.deleteLocation

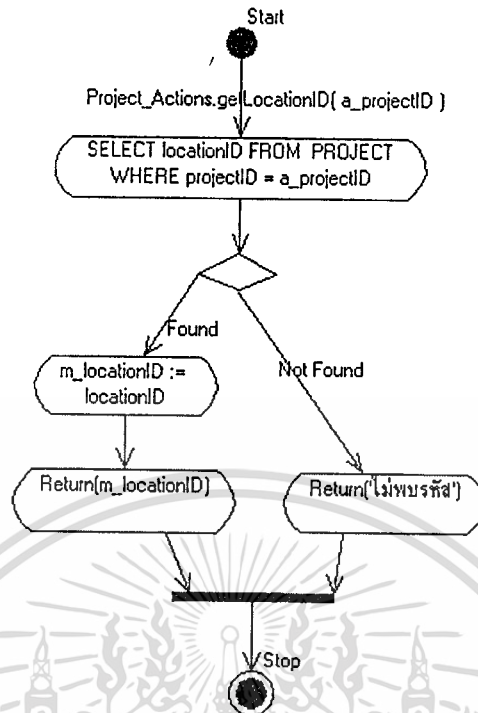


รูปที่ 7.27 แอคตวิทีไคอะแกรมของโมดูล Project_Actions.isNewProject



รูปที่ 7.28 แอคตวิทีไคอะแกรมของโมดูล Project_Actions.getProjectName

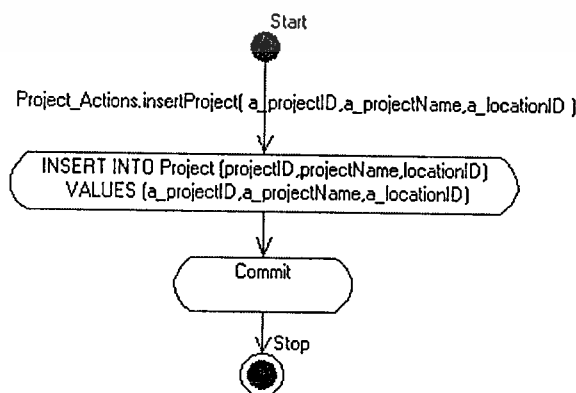
เอกสารนี้เป็นเอกสารของบริษัทเอกชนที่มีลิขสิทธิ์และใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 7.29 แอคตีวิตี้ไดอะแกรมของโมดูล `Project_Actions.getLocationID`

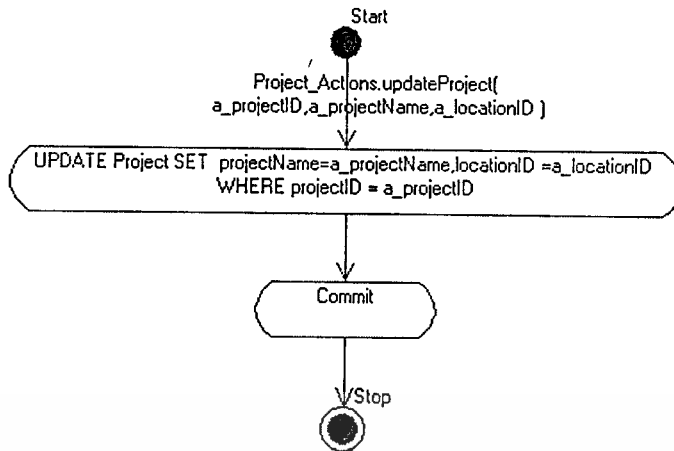


รูปที่ 7.30 แอคตีวิตี้ไดอะแกรมของ โมดูล `Project_Actions.retrieveProject`

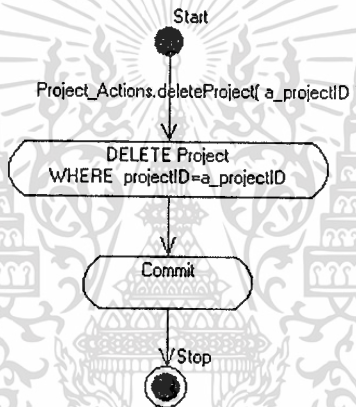


รูปที่ 7.31 แอคตีวิตี้ไดอะแกรมของ โมดูล `Project_Actions.insertProject`

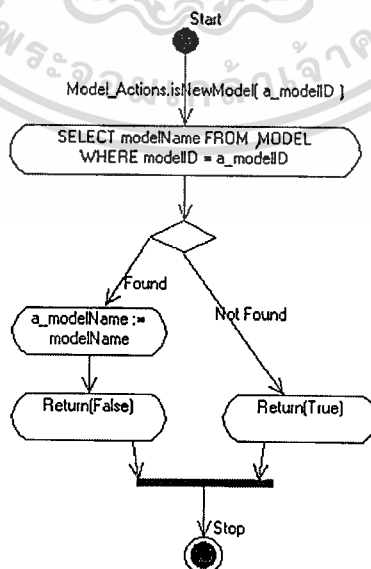
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 7.32 แอคทิวิตี้ไดอะแกรมของโมดูล `Project_Actions.updateProject`

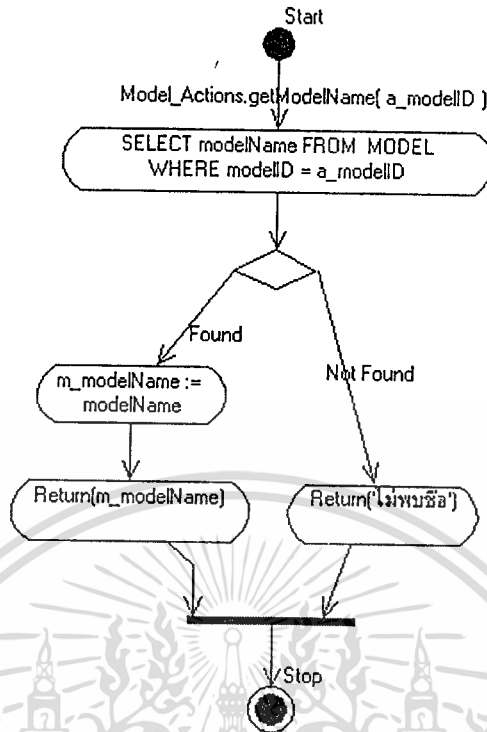


รูปที่ 7.33 แอคทิวิตี้ไดอะแกรมของโมดูล `Project_Actions.deleteProject`

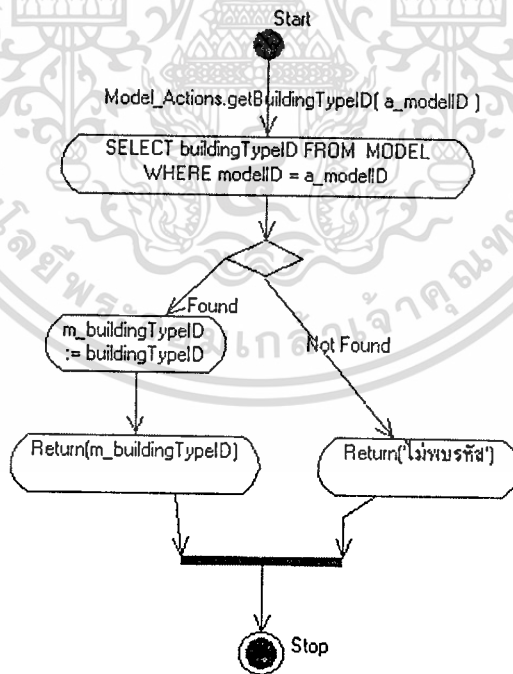


รูปที่ 7.34 แอคทิวิตี้ไดอะแกรมของโมดูล `Model_Actions.isNewModel`

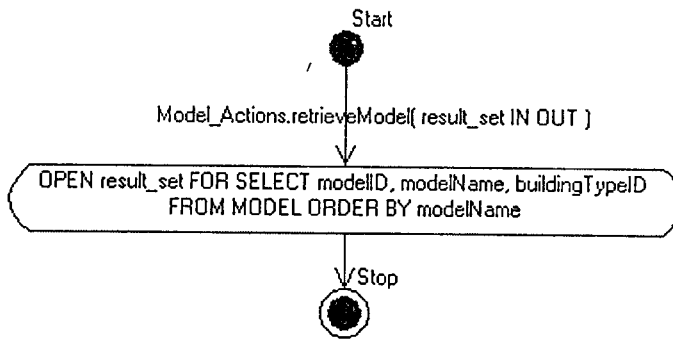
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



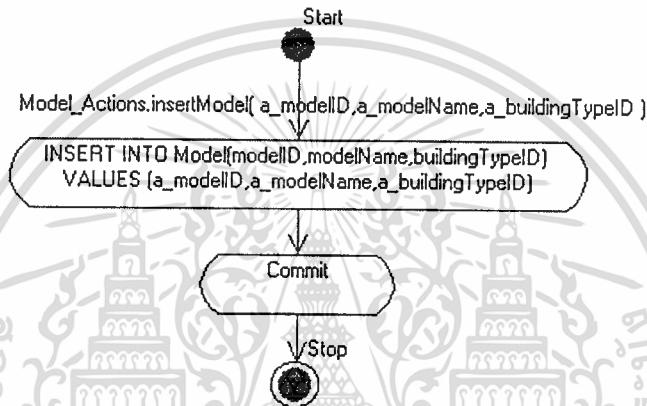
รูปที่ 7.35 แอคตีวิตี้ไดอะแกรมของโมดูล Model_Actions.getModelName



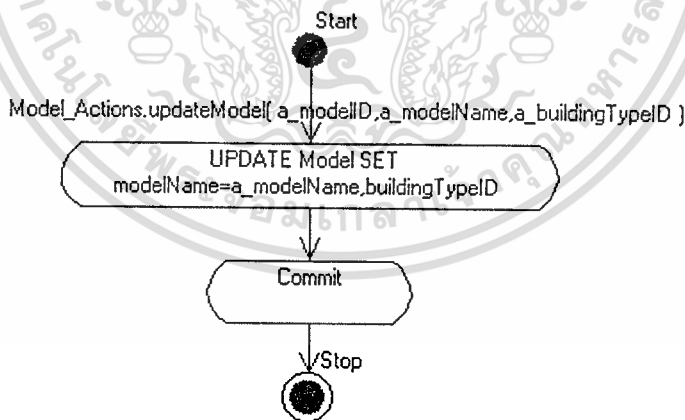
รูปที่ 7.36 แอคตีวิตี้ไดอะแกรมของโมดูล Model_Actions.getBuildingTypeID



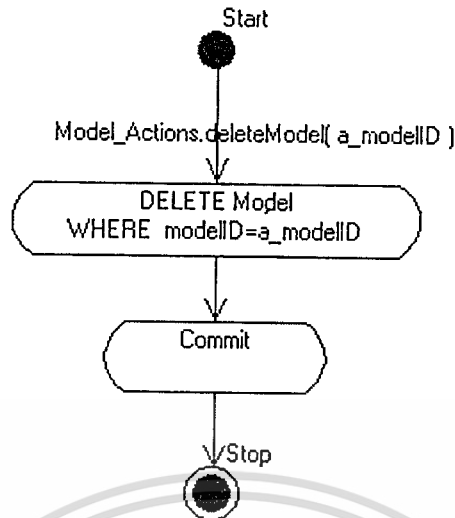
รูปที่ 7.37 แอคตีวิตี้ไดอะแกรมของโมดูล `Model_Actions.retrieveModel`



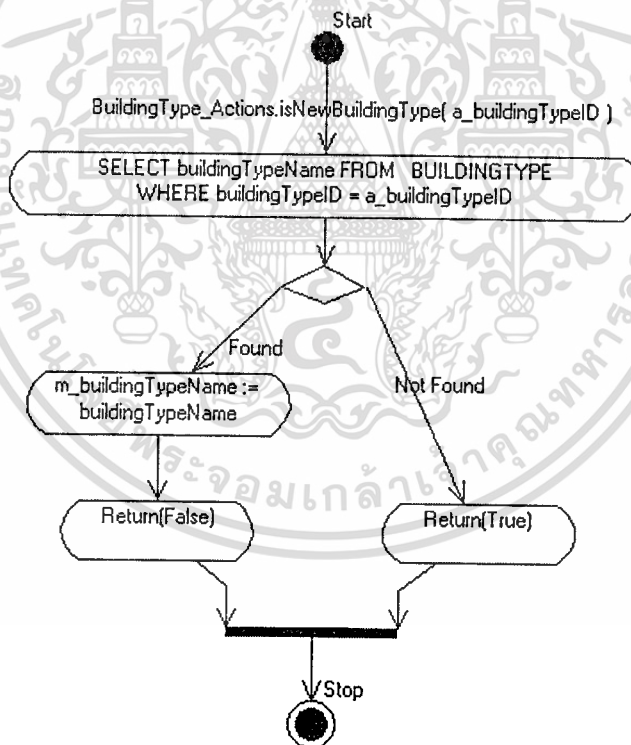
รูปที่ 7.38 แอคตีวิตี้ไดอะแกรมของโมดูล `Model_Actions.insertModel`



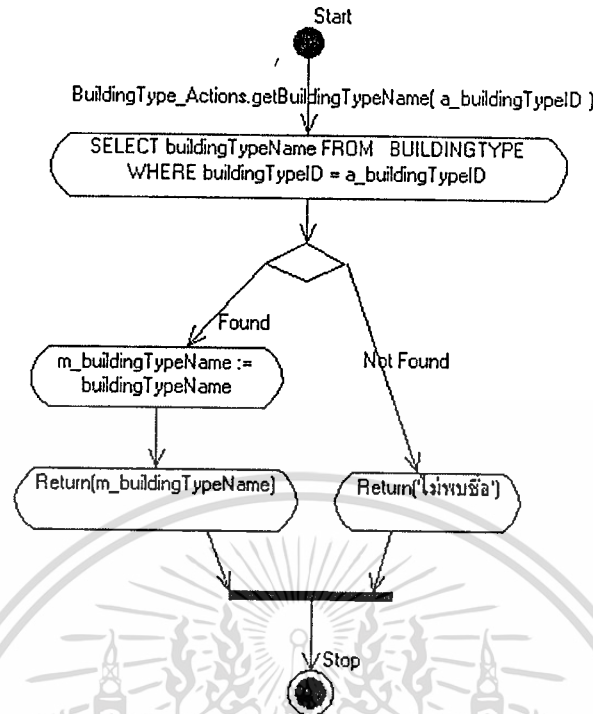
รูปที่ 7.39 แอคตีวิตี้ไดอะแกรมของโมดูล `Model_Actions.updateModel`



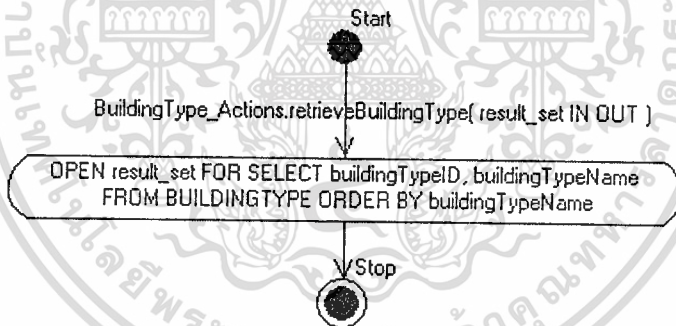
รูปที่ 7.40 แอคตีวิตี้ไดอะแกรมของโมดูล `Model_Actions.deleteModel`



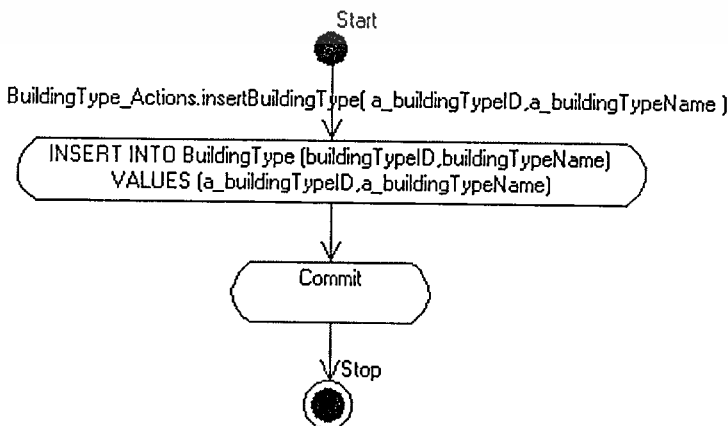
รูปที่ 7.41 แอคตีวิตี้ไดอะแกรมของโมดูล `BuildingType_Actions.isNewBuildingType`



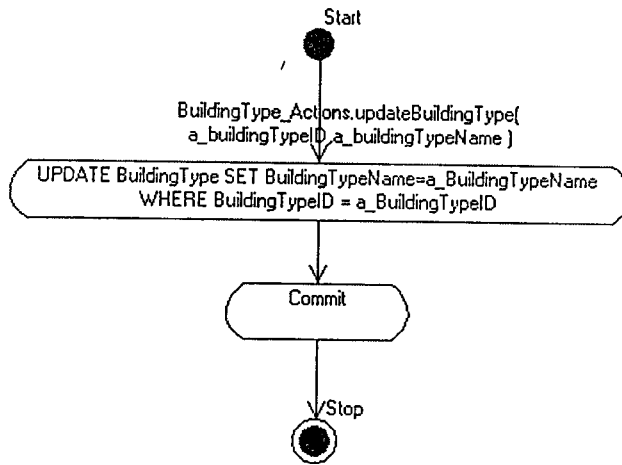
รูปที่ 7.42 แอคตวิต้ไอะแกรมของ โมดูล BuildingType_Actions.getBuildingTypeName



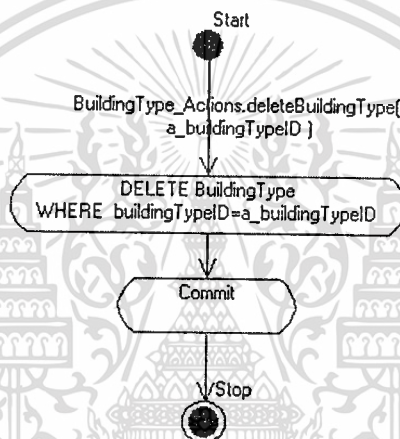
รูปที่ 7.43 แอคตวิต้ไอะแกรมของ โมดูล BuildingType_Actions.rctrieveBuildingType



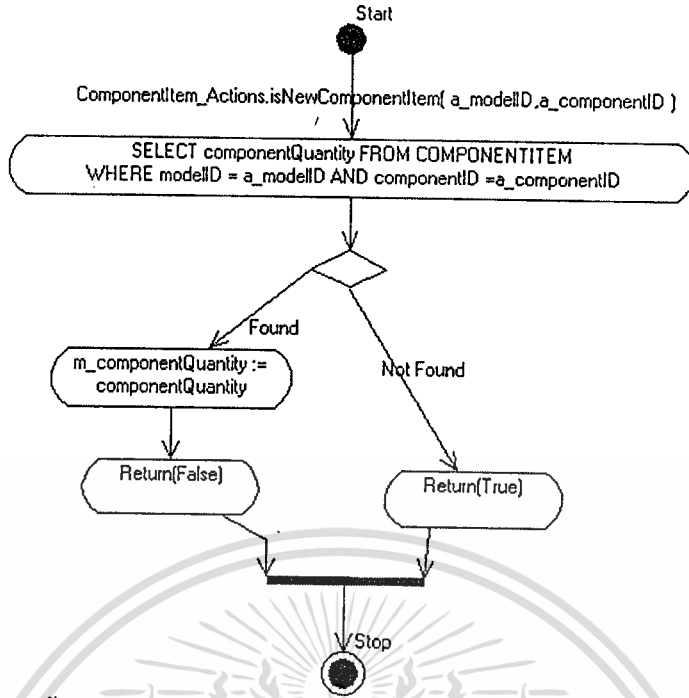
รูปที่ 7.44 แอคตวิต้ไอะแกรมของ โมดูล BuildingType_Actions.insertBuildingType



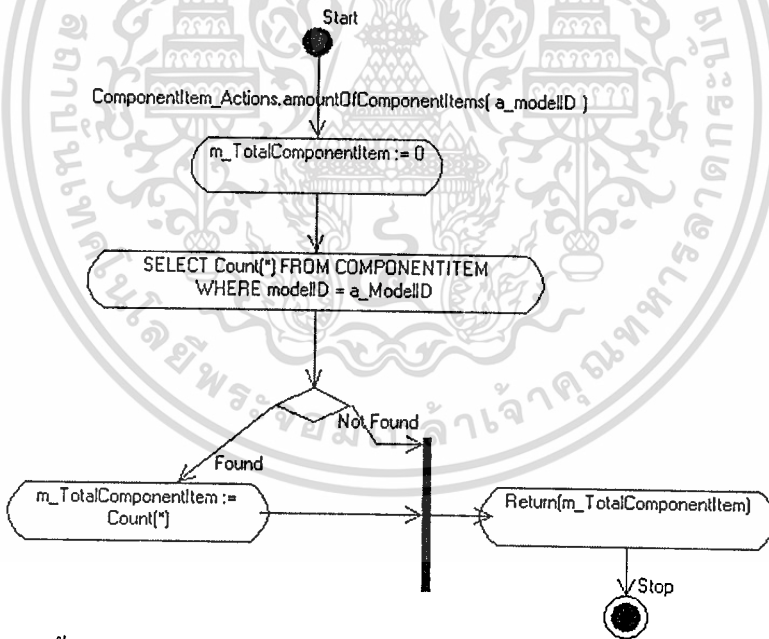
รูปที่ 7.45 แอคตีวิตี้ไดอะแกรมของโมดูล BuildingType_Actions.updateBuildingType



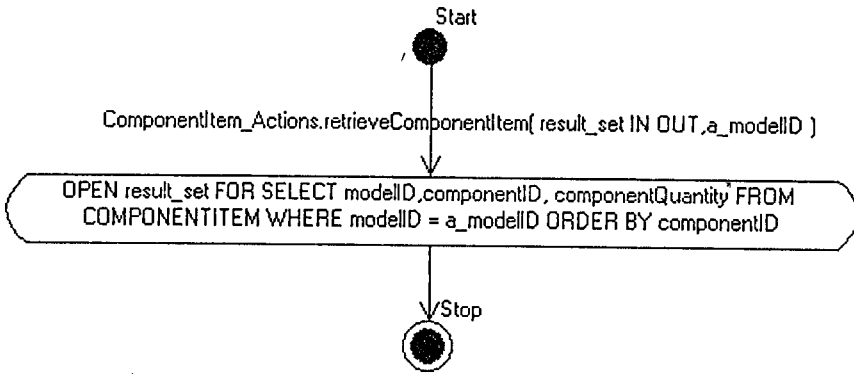
รูปที่ 7.46 แอคตีวิตี้ไดอะแกรมของโมดูล BuildingType_Actions.deleteBuildingType



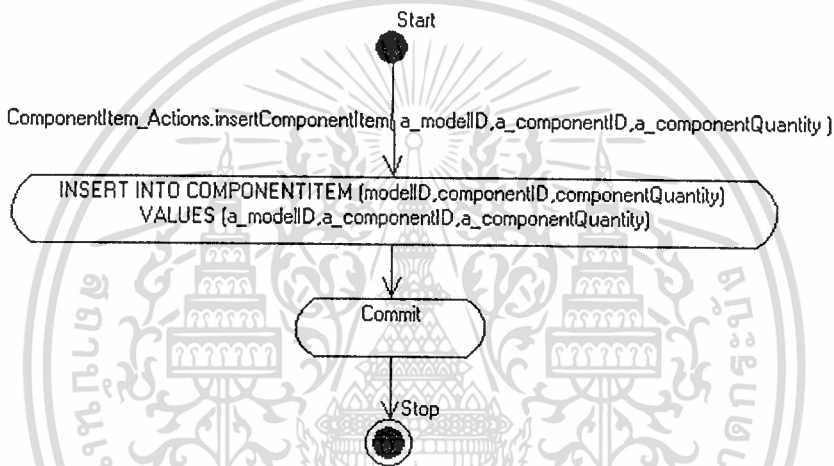
รูปที่ 7.47 แอคติวิตีไดอะแกรมของโมดูล `ComponentItem_Actions.isNewComponentItem`



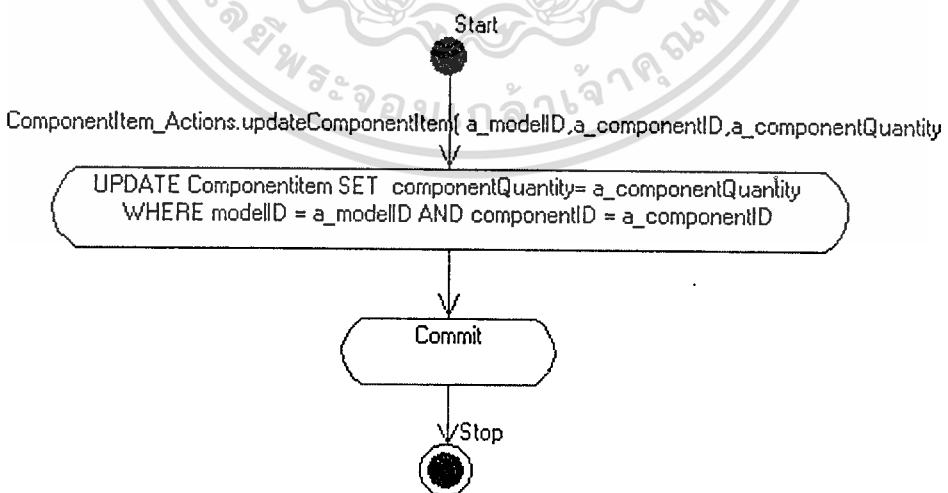
รูปที่ 7.48 แอคติวิตีไดอะแกรมของโมดูล `ComponentItem_Actions. AmountOfComponentItems`



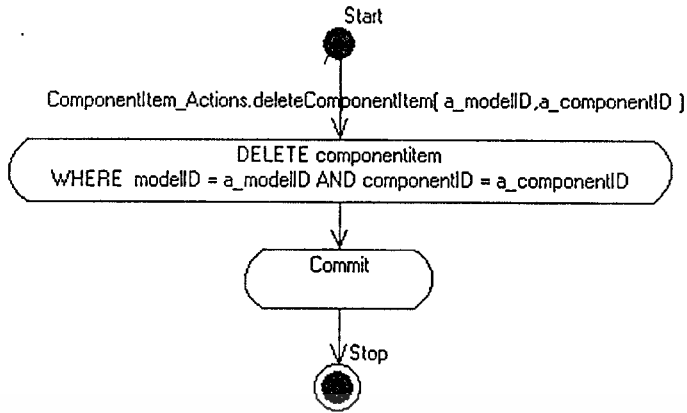
รูปที่ 7.49 แอคทีวิตี้ไคอะแกรมของโมดูล `ComponentItem_Actions.retrieveComponentItem`



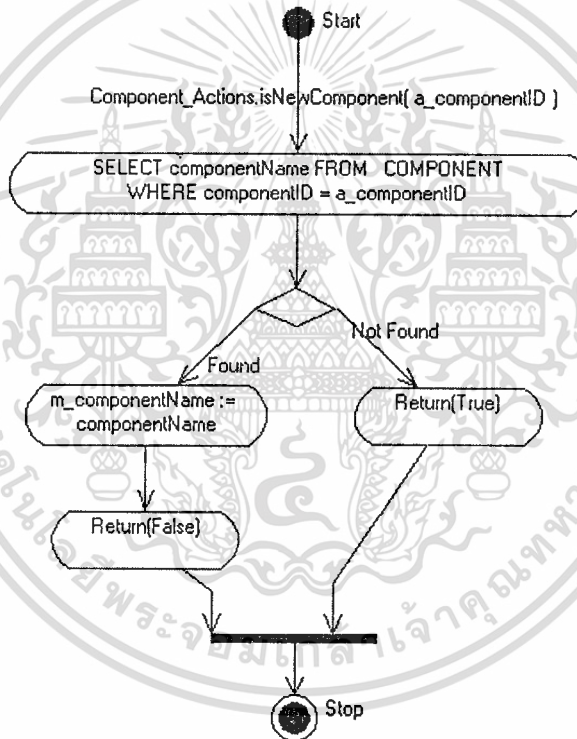
รูปที่ 7.50 แอคทีวิตี้ไคอะแกรมของโมดูล `ComponentItem_Actions.insertComponentItem`



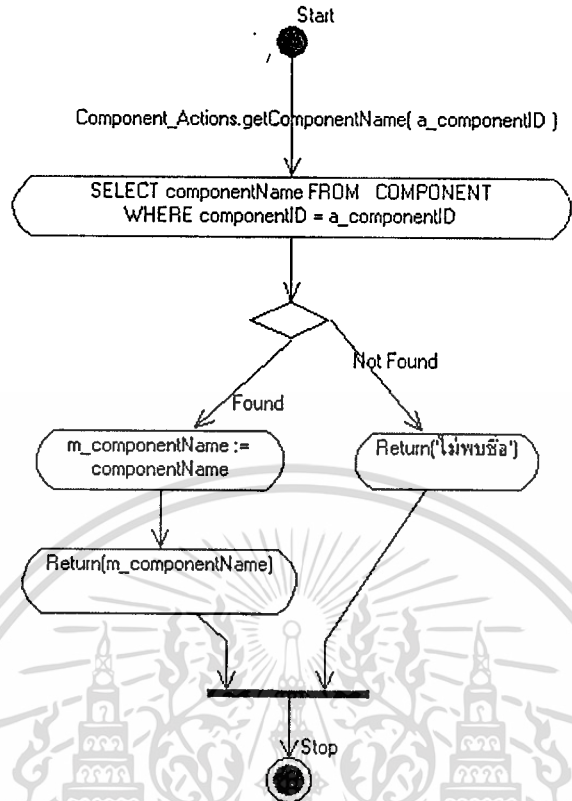
รูปที่ 7.51 แอคทีวิตี้ไคอะแกรมของโมดูล `ComponentItem_Actions.updateComponentItem`



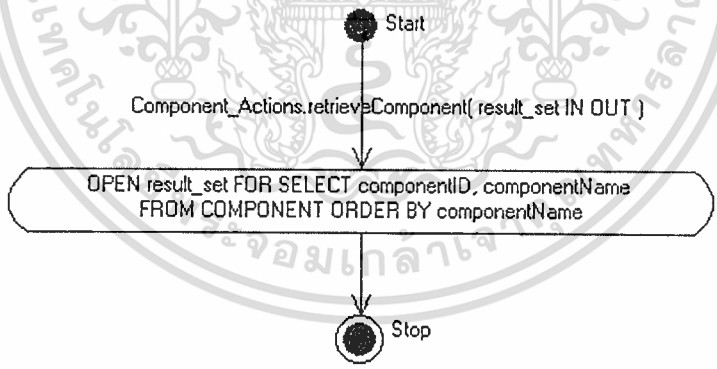
รูปที่ 7.52 แอคทีวิตีไดอะแกรมของโมดูล `ComponentItem_Actions.deleteComponentItem`



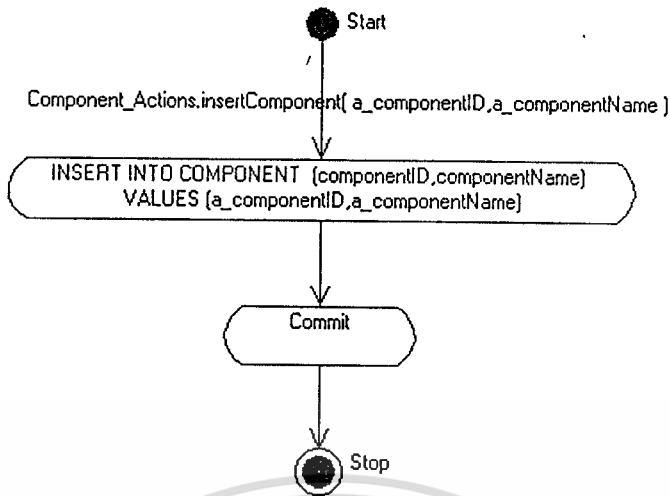
รูปที่ 7.53 แอคทีวิตีไดอะแกรมของโมดูล `Component_Actions.isNewComponent`



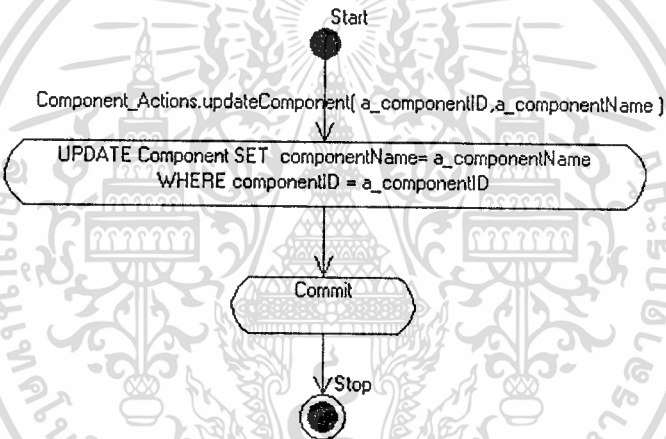
รูปที่ 7.54 แอคตีวิตี้ไดอะแกรมของโมดูล Component_Actions.getComponentName



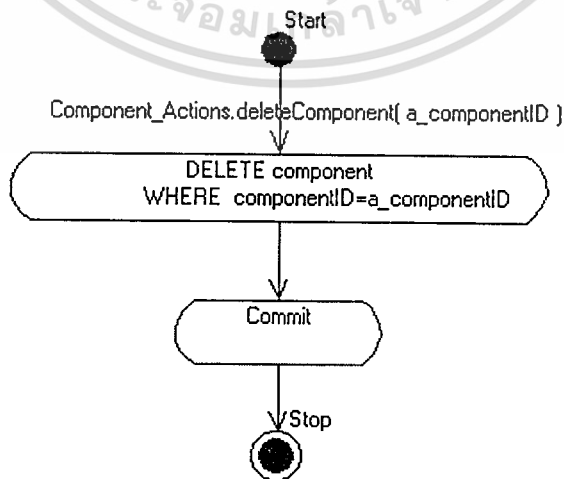
รูปที่ 7.55 แอคตีวิตี้ไดอะแกรมของโมดูล Component_Actions.retrieveComponent



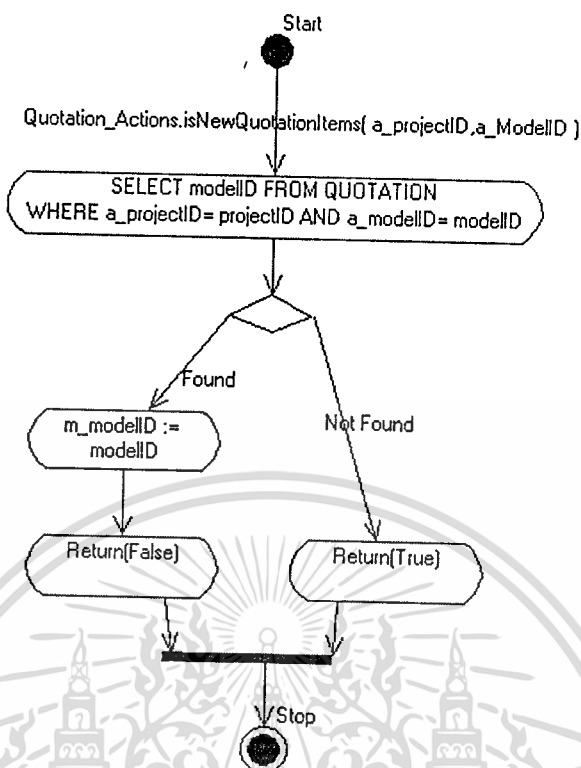
รูปที่ 7.56 แอคตีวิตี้ไดอะแกรมของโมดูล `Component_Actions.insertComponent`



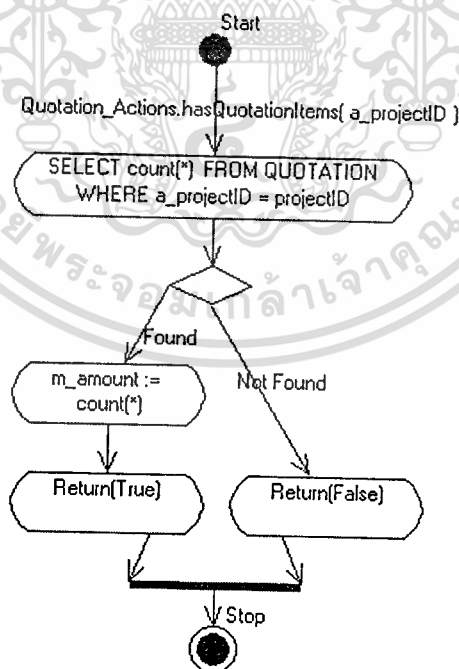
รูปที่ 7.57 แอคตีวิตี้ไดอะแกรมของโมดูล `Component_Actions.updateComponent`



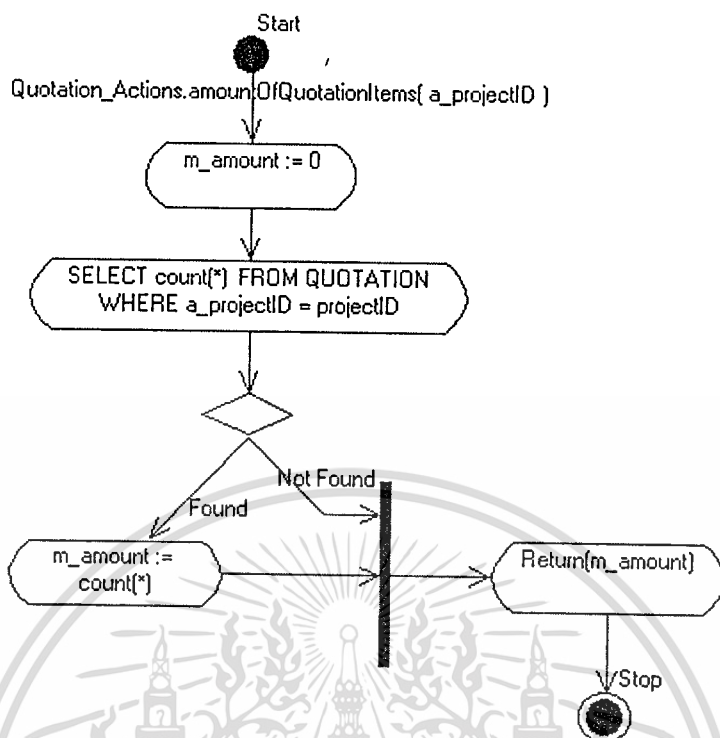
รูปที่ 7.58 แอคตีวิตี้ไดอะแกรมของโมดูล `Component_Actions.deleteComponent`



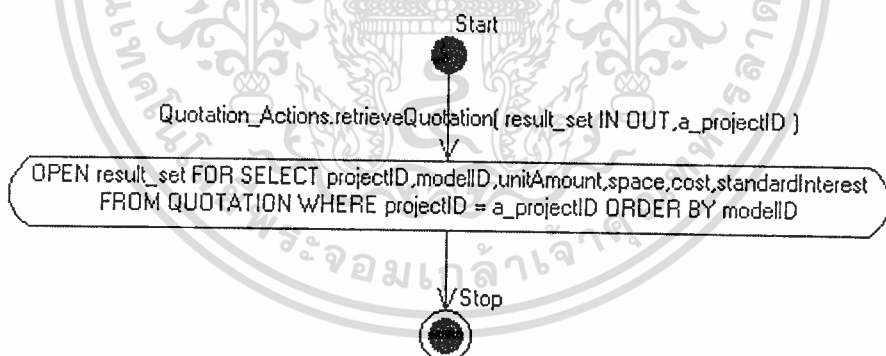
รูปที่ 7.59 แอคทีวิตี้ไดอะแกรมของโมดูล `Quotation_Actions.isNew QuotationItems`



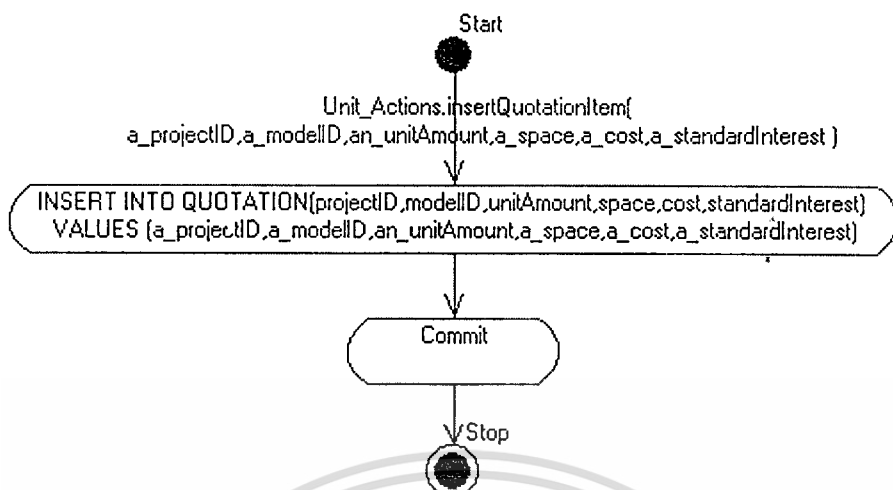
รูปที่ 7.60 แอคทีวิตี้ไดอะแกรมของโมดูล `Quotation_Actions.hasQuotationItems`



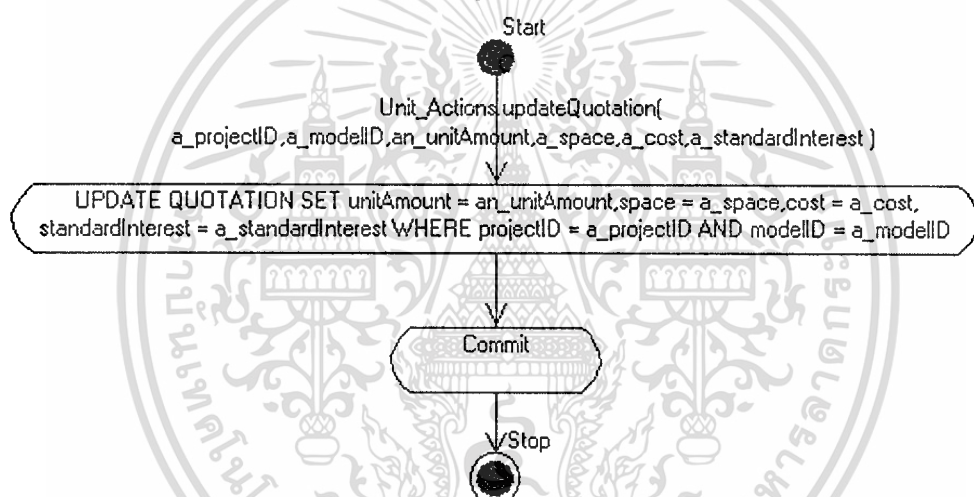
รูปที่ 7.61 แอคทีวิตี้ไดอะแกรมของโมดูล `Quotation_Actions.amountOfQuotationItems`



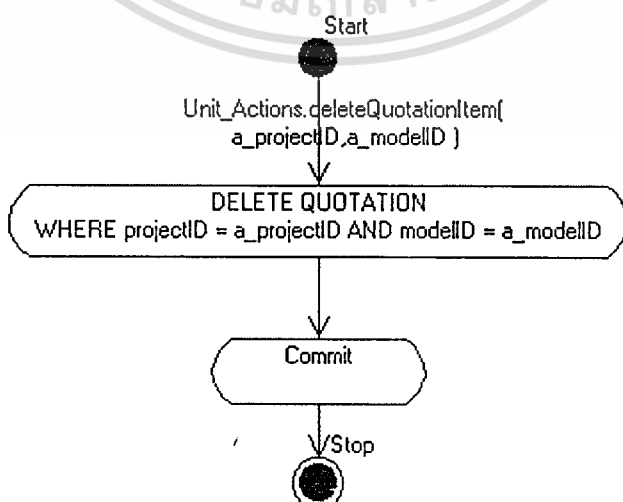
รูปที่ 7.62 แอคทีวิตี้ไดอะแกรมของโมดูล `Quotation_Actions.retrieve Quotation`



รูปที่ 7.63 แอคตวิดีโคอะแกรมของโมดูล Quotation_Actions.insertQuotationItem

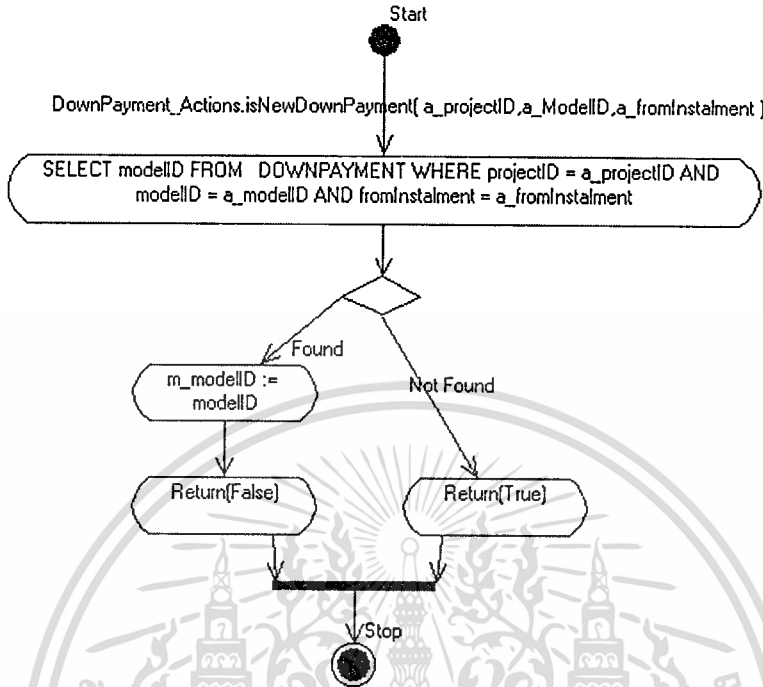


รูปที่ 7.64 แอคตวิดีโคอะแกรมของโมดูล Quotation_Actions.updateQuotationItem

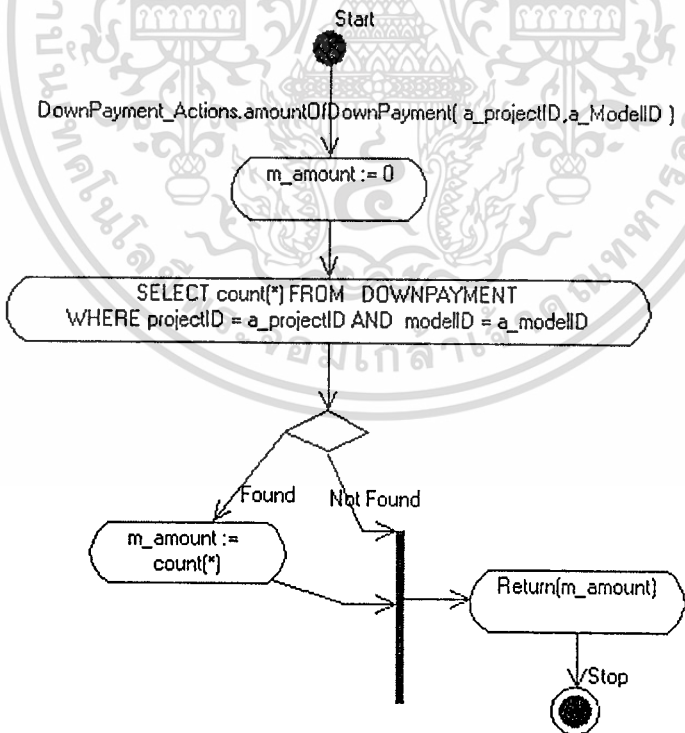


รูปที่ 7.65 แอคตวิดีโคอะแกรมของโมดูล Quotation_Actions.deleteQuotationItem

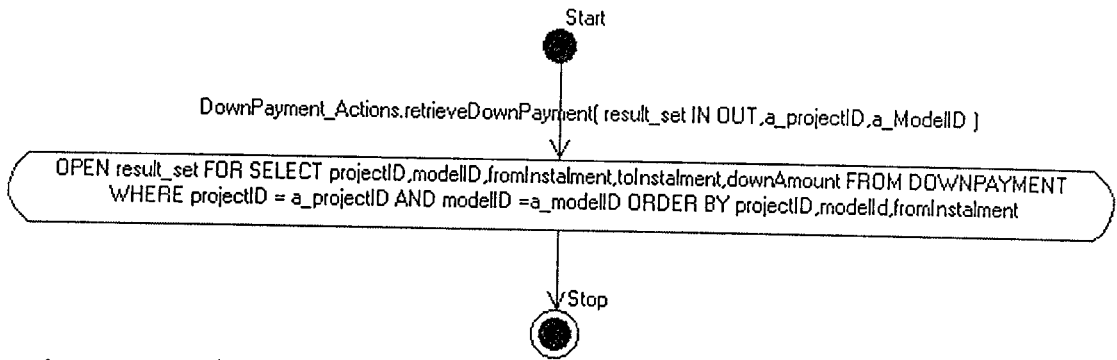
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



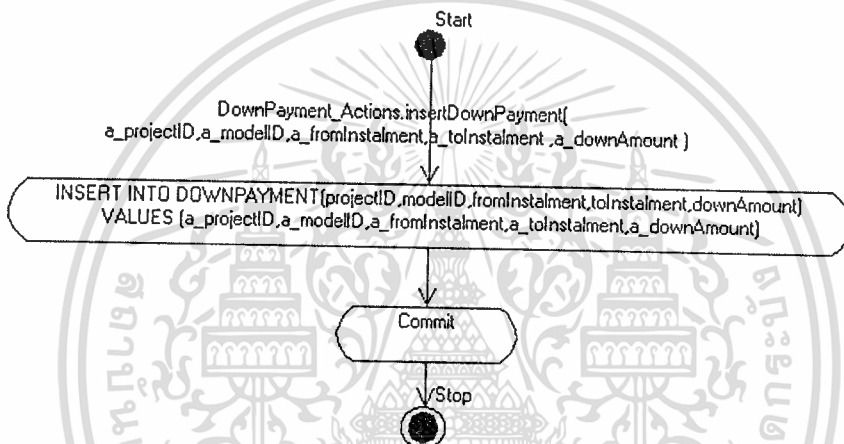
รูปที่ 7.66 แอคตีวิตี้ไดอะแกรมของโมดูล DownPayment_Actions.isNewDownPayment



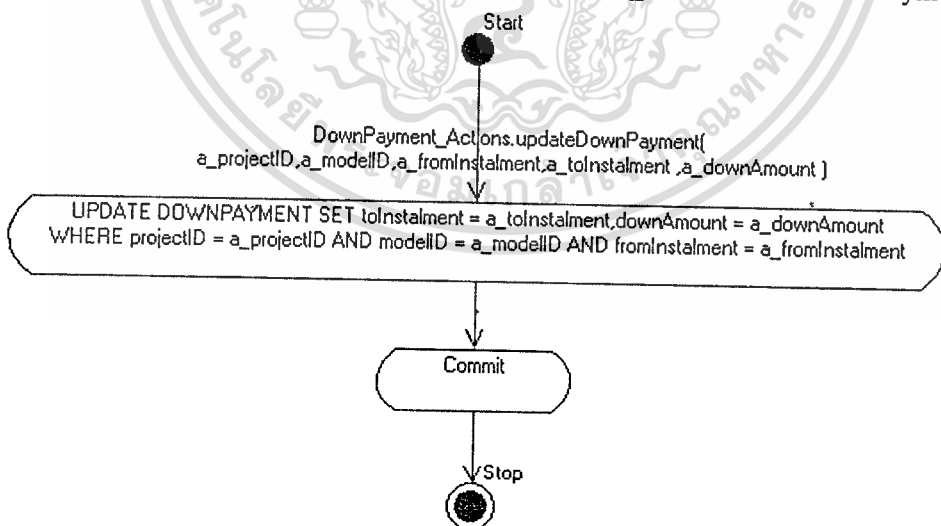
รูปที่ 7.67 แอคตีวิตี้ไดอะแกรมของโมดูล DownPayment_Actions.amountOfDownPayment



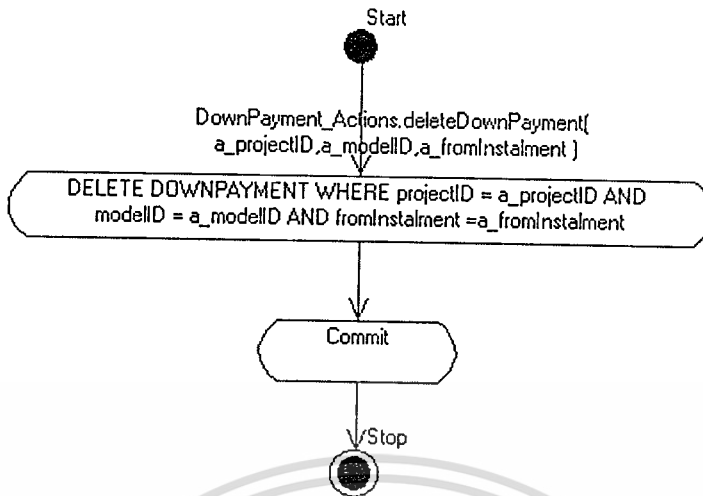
รูปที่ 7.68 แอคตีวิตี้ไดอะแกรมของโมดูล `DownPayment_Actions.retrieveDownPayment`



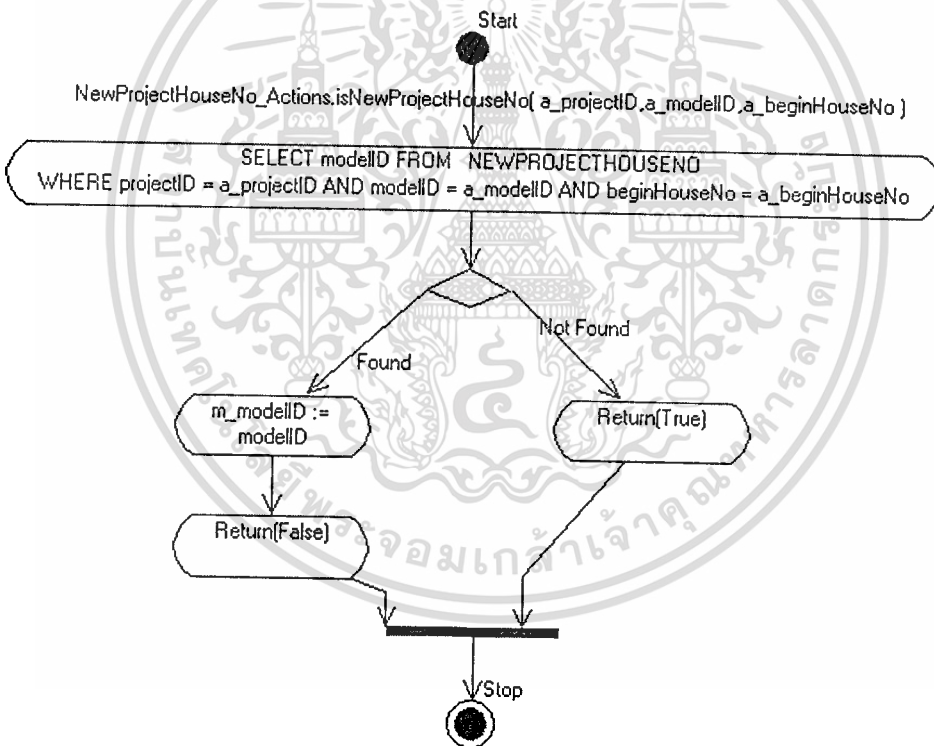
รูปที่ 7.69 แอคตีวิตี้ไดอะแกรมของโมดูล `DownPayment_Actions.insertDownPayment`



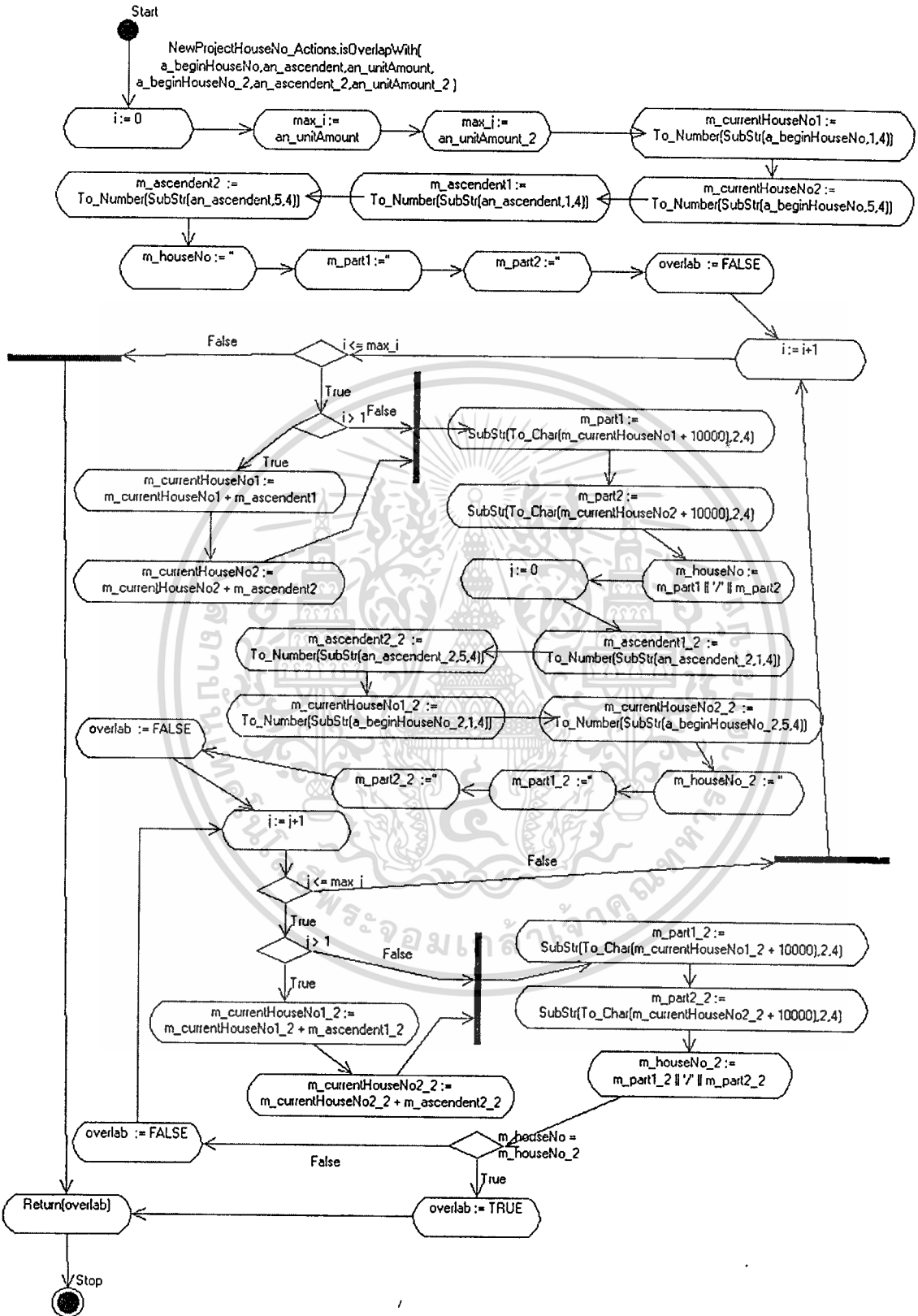
รูปที่ 7.70 แอคตีวิตี้ไดอะแกรมของโมดูล `DownPayment_Actions.updateDownPayment`



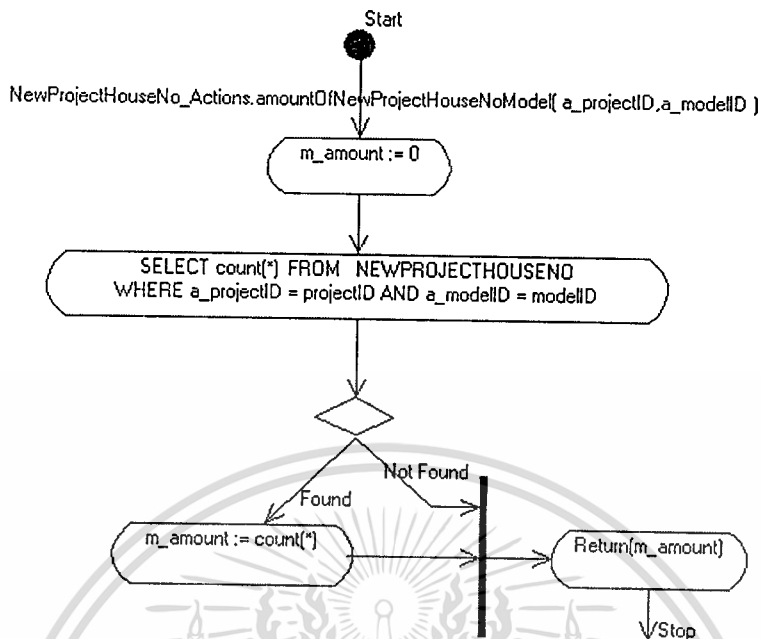
รูปที่ 7.71 แอคตีวิตี้ไดอะแกรมของโมดูล DownPayment_Actions.deleteDownPayment



รูปที่ 7.72 แอคตีวิตี้ไดอะแกรมของโมดูล NewProjectHouseNo_Actions.isNewProjectHouseNo

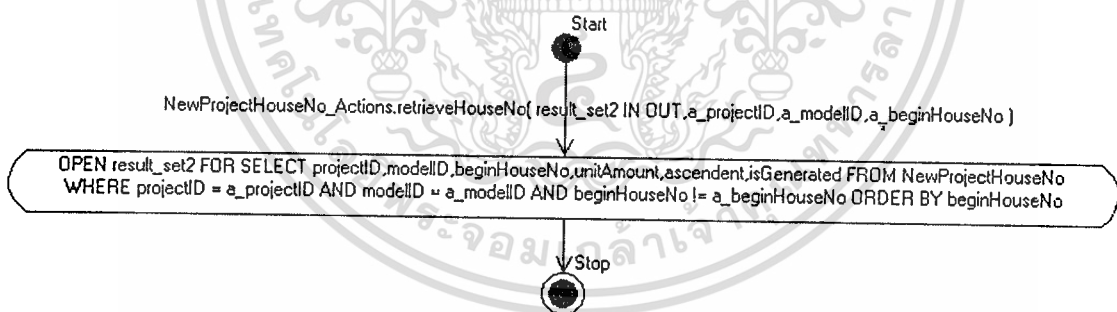


รูปที่ 7.73 แอคตีวิตี้ไดอะแกรมของโมดูล NewProjectHouseNo_Actions.isOverlapwith

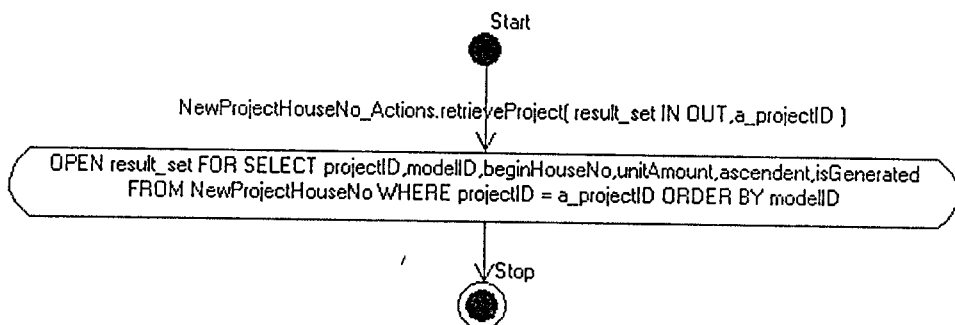


รูปที่ 7.74 แอคตีวิตี้ไดอะแกรมของโมดูล

NewProjectHouseNo_Actions.amountOfNewProjectHouseNo

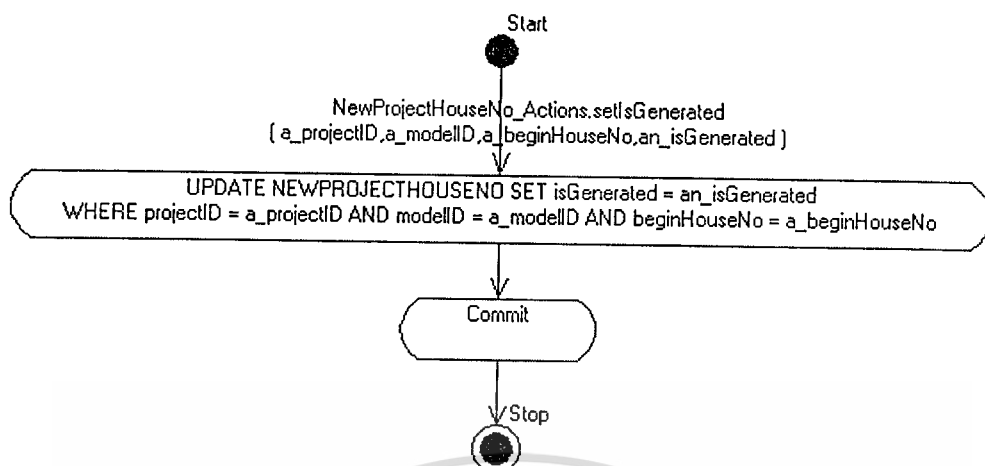


รูปที่ 7.75 แอคตีวิตี้ไดอะแกรมของโมดูล NewProjectHouseNo_Actions.retrieveHouseNo

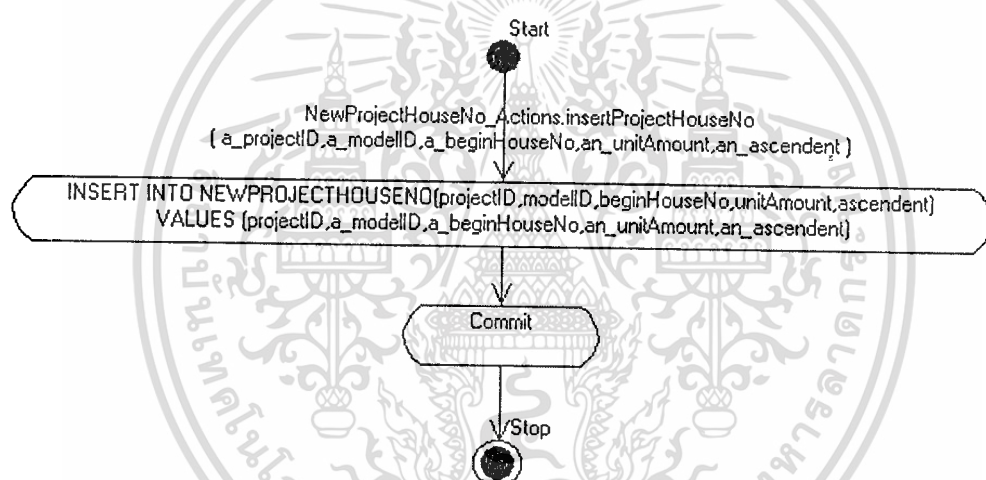


รูปที่ 7.76 แอคตีวิตี้ไดอะแกรมของโมดูล NewProjectHouseNo_Actions.retrieveProject

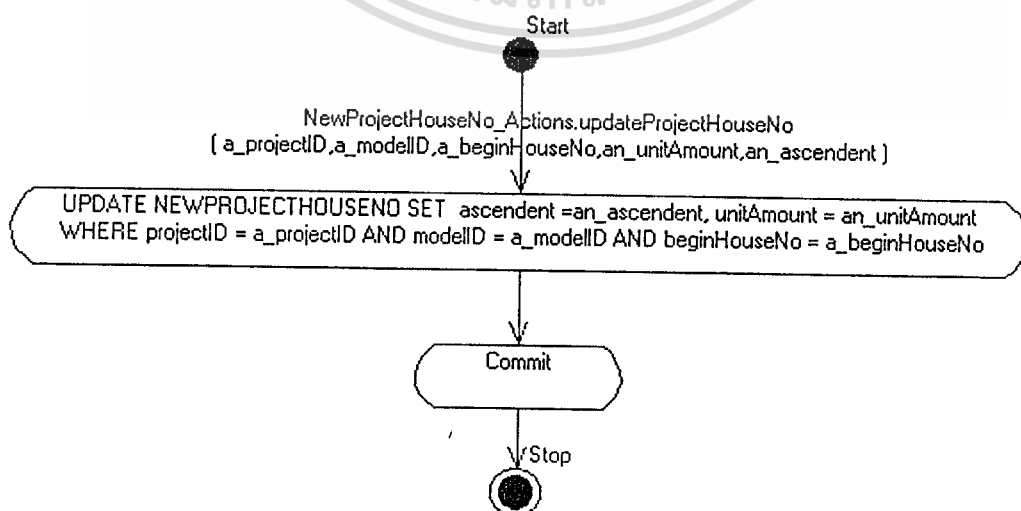
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 7.77 แอคติวิตี้ไคอะแกรมของโมดูล `NewProjectHouseNo_Actions.setIsGenerated`



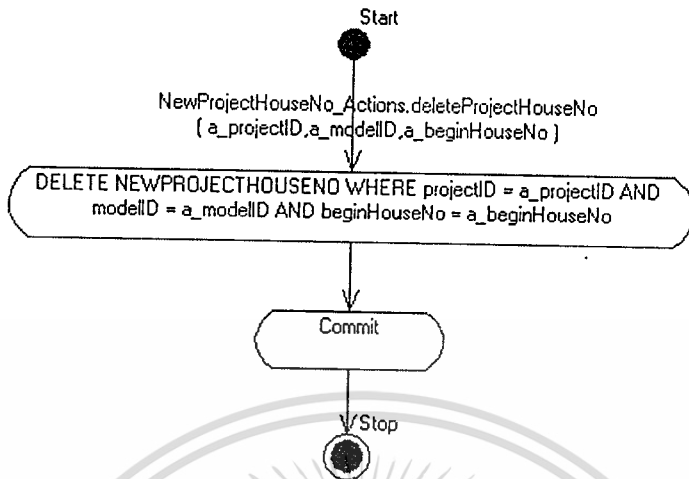
รูปที่ 7.78 แอคติวิตี้ไคอะแกรมของโมดูล `NewProjectHouseNo_Actions.insertProjectHouseNo`



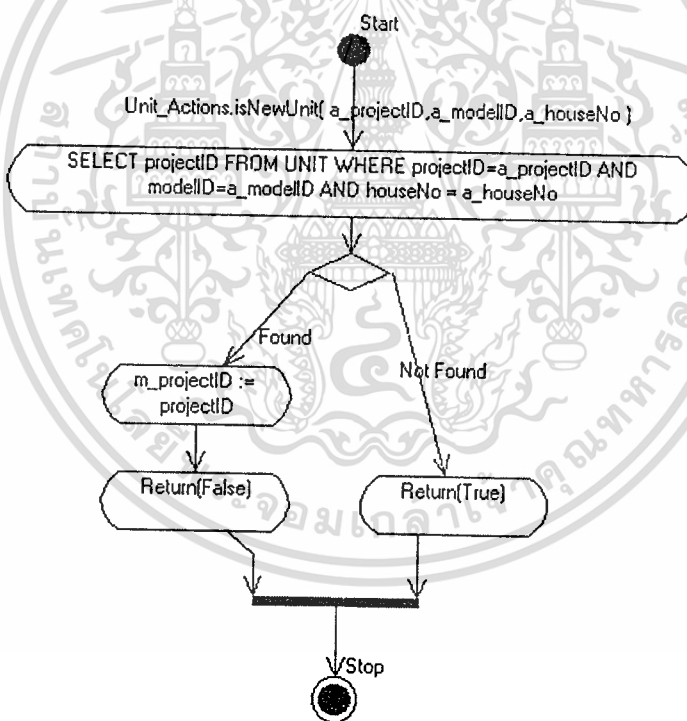
รูปที่ 7.79 แอคติวิตี้ไคอะแกรมของโมดูล `NewProjectHouseNo_Actions.updateProjectHouseNo`

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ในการค้า

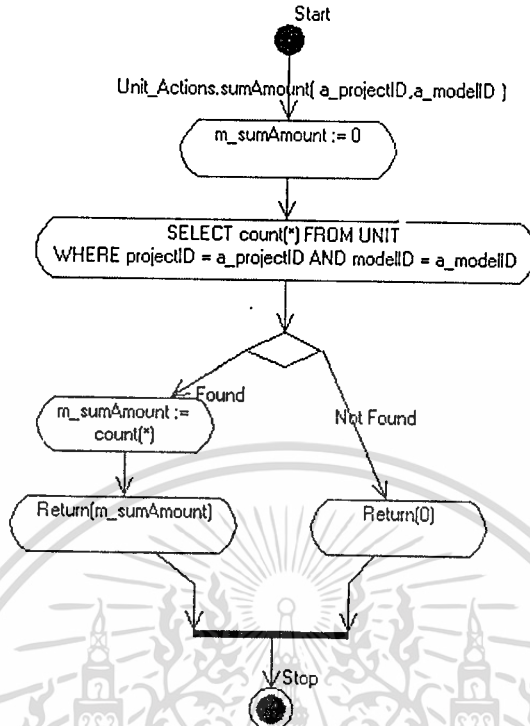
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



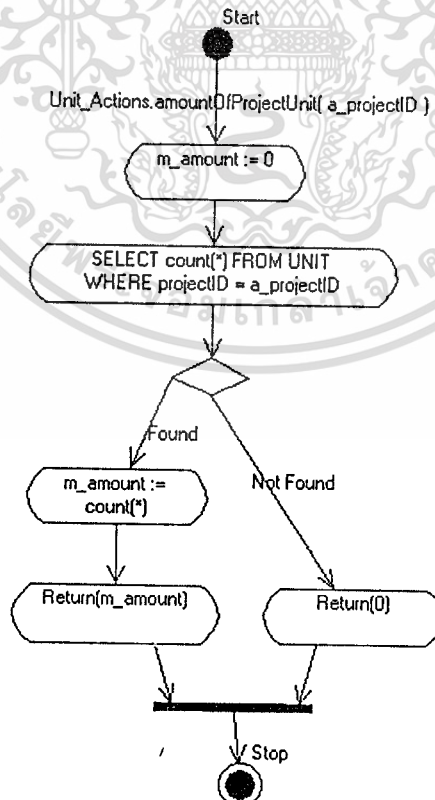
รูปที่ 7.80 แอคติวิตี้ไคอะแกรมของโมดูล NewProjectHouseNo_Actions.deleteProjectHouseNo



รูปที่ 7.81 แอคติวิตี้ไคอะแกรมของโมดูล Unit_Actions.isNewUnit

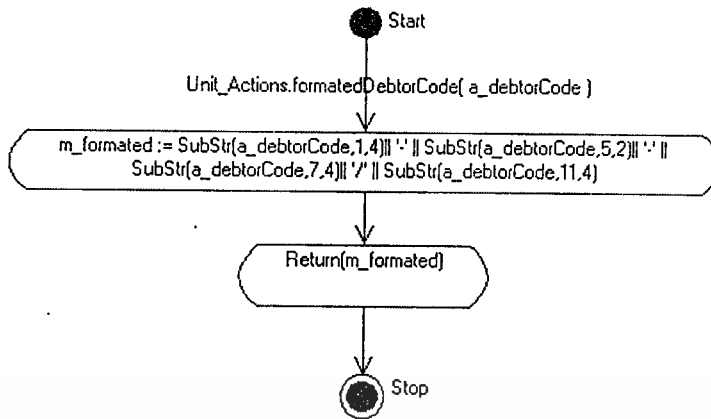


รูปที่ 7.82 แอคตวิธี่โคอะแกรมของ โมดูล Unit_Actions.sumAmount

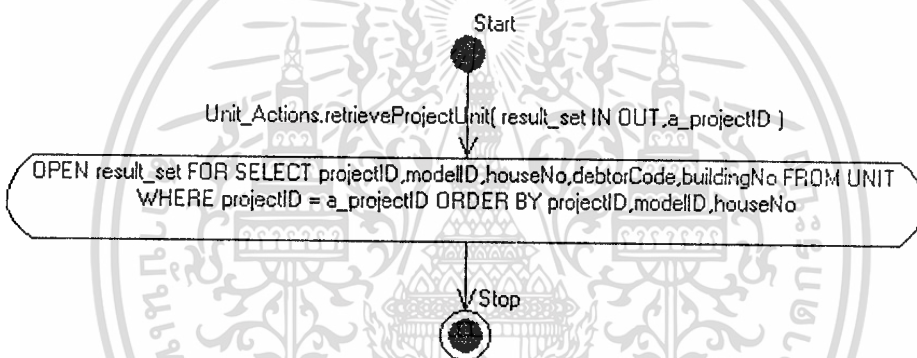


รูปที่ 7.83 แอคตวิธี่โคอะแกรมของ โมดูล Unit_Actions.amountOfProjectUnit

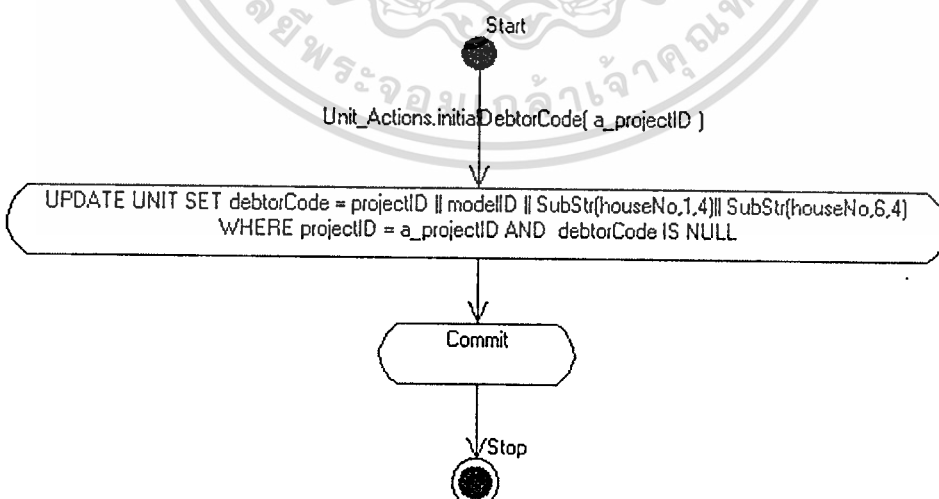
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



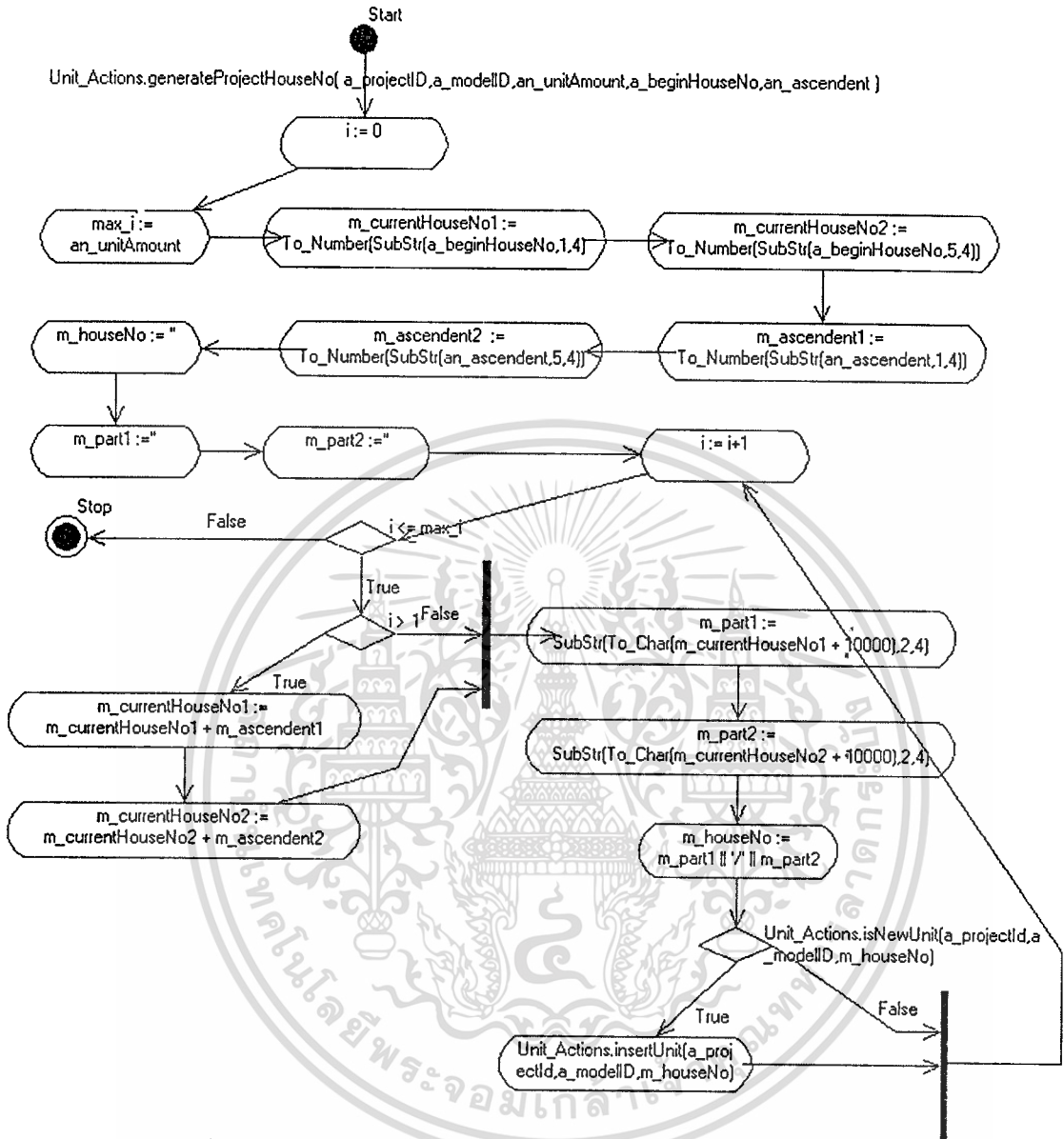
รูปที่ 7.84 แอคตีวิตี้ไดอะแกรมของโมดูล `Unit_Actions.formatedDebtorCode`



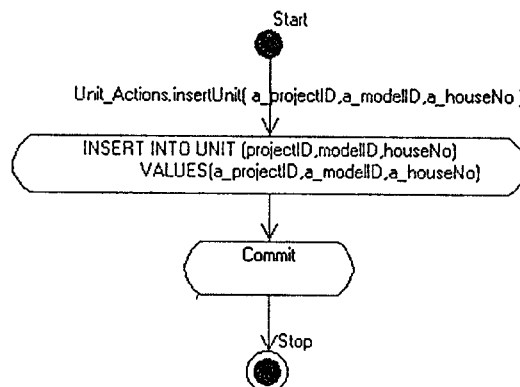
รูปที่ 7.85 แอคตีวิตี้ไดอะแกรมของโมดูล `Unit_Actions.retrieveProjectUnit`



รูปที่ 7.86 แอคตีวิตี้ไดอะแกรมของโมดูล `Unit_Actions.initialDebtorCode`

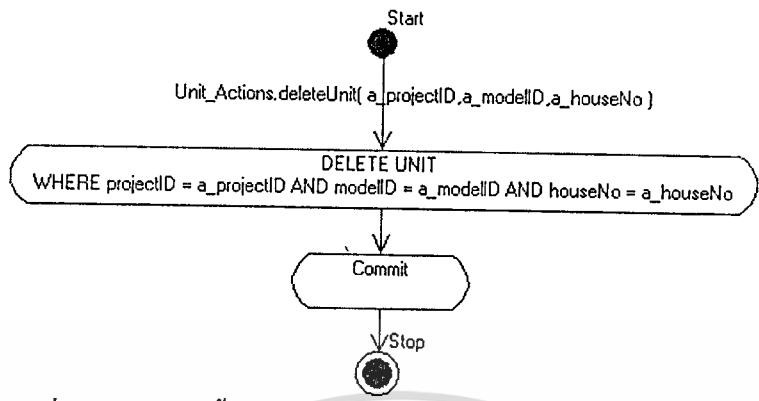


รูปที่ 7.87 แอคตีวิตี้ไดอะแกรมของโมดูล Unit_Actions.generateProjectHouseNo

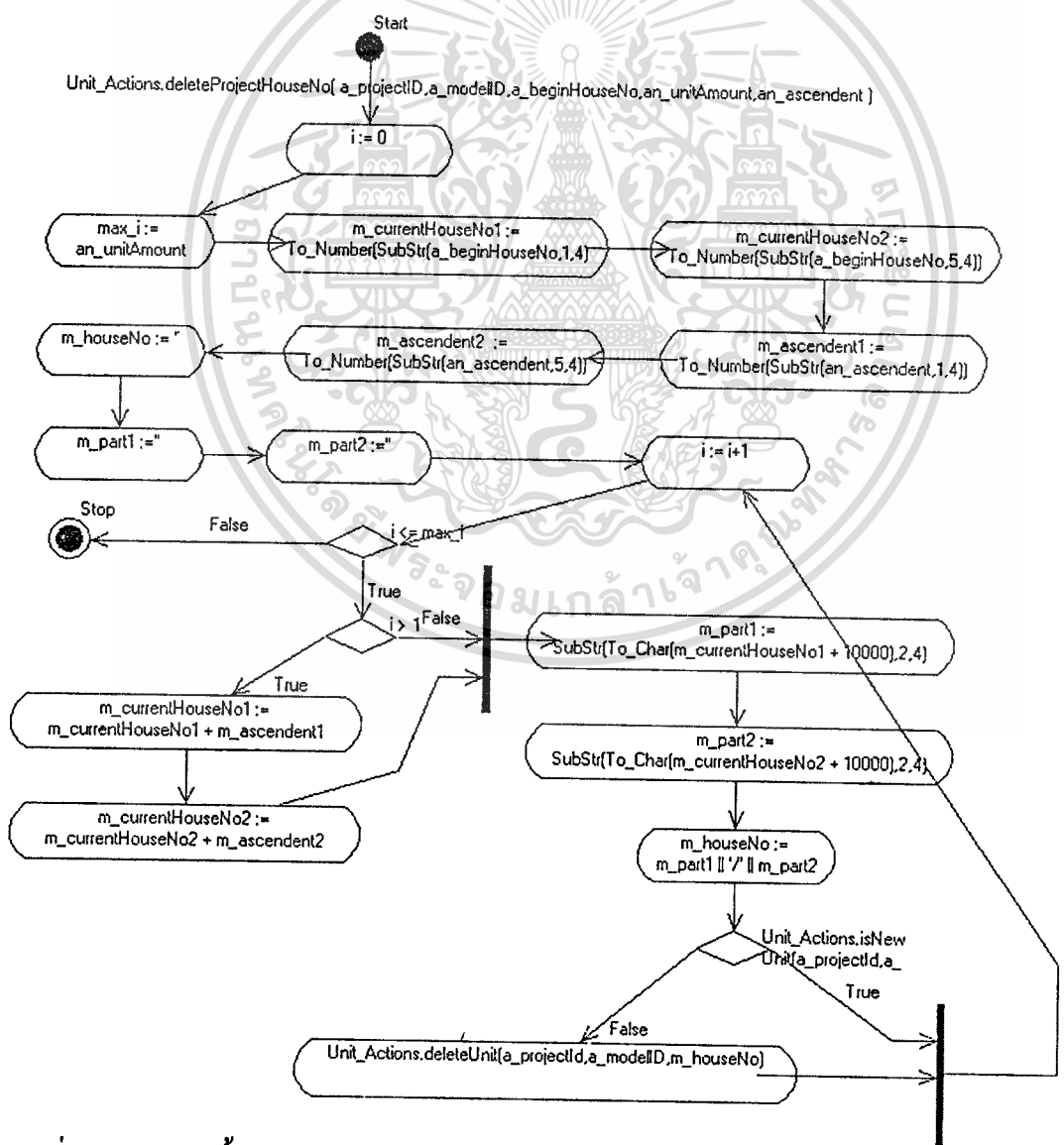


รูปที่ 7.88 แอคตีวิตี้ไดอะแกรมของโมดูล Unit_Actions.insertUnit

เอกสารนี้เป็นเอกสารลิขสิทธิ์ของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี ใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

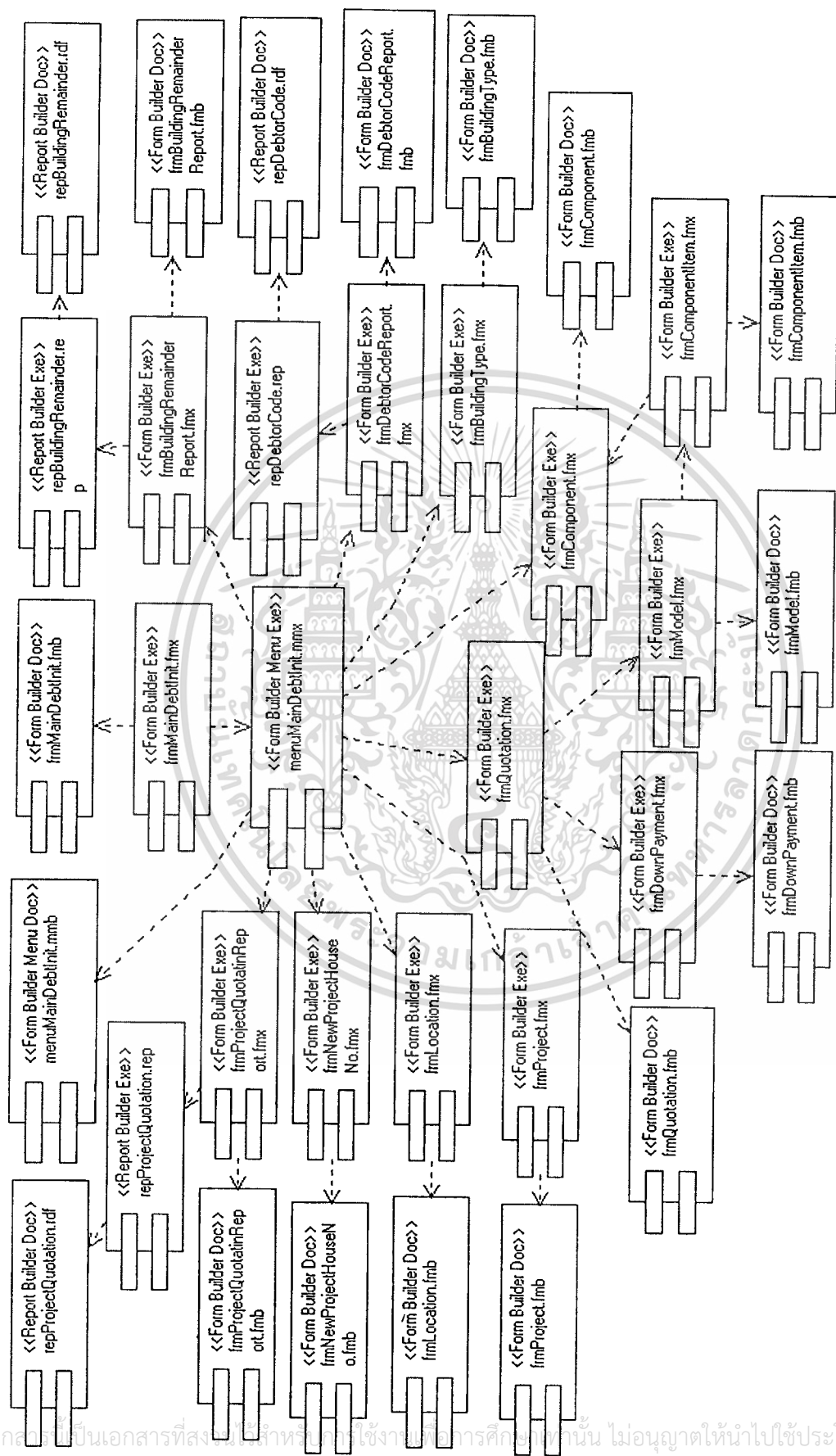


รูปที่ 7.89 แอคทีวิตีไดอะแกรมของโมดูล Unit_Actions.deleteUnit



รูปที่ 7.90 แอคทีวิตีไดอะแกรมของโมดูล Unit_Actions.deleteProjectHouseNo

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ไว้สำหรับใช้ภายในเท่านั้น
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 7.91 คอมโพเนนต์โปรแกรมแสดงองค์ประกอบซอฟต์แวร์ระบบบัญชีนี้ โครงสร้างบัญชีลูกหนี้ของการเคหะแห่งชาติ

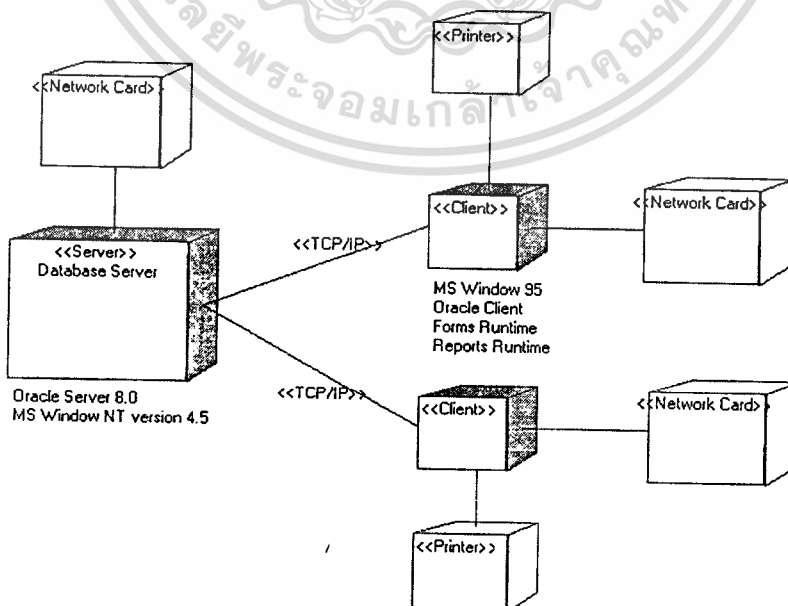
7.2 คอมโพเนนท์ใดอะแกรม

คอมโพเนนท์ใดอะแกรมของระบบตั้งหนี้โครงการบัญชีลูกหนี้การเคหะแห่งชาติ แสดงองค์ประกอบของระบบซอฟต์แวร์ ดังรูปที่ 7.1 ประกอบด้วยไฟล์ที่สร้างโดย Oracle Developer คือ ไฟล์โค้ดต้นฉบับของฟอร์ม (Form Builder Doc) ไฟล์รันไทม์ของฟอร์ม (Form Builder Exe) ไฟล์โค้ดต้นฉบับของเมนูฟอร์ม (Form Builder Menu Doc) ไฟล์รันไทม์ของเมนูฟอร์ม (Form Builder Menu Exe) ไฟล์โค้ดต้นฉบับของรีพอร์ต (Report Builder Document) และ ไฟล์รันไทม์ของรีพอร์ต (Report Builder Exec) ซึ่งจะมีนามสกุลเป็น .fmb .fmf .mmf .mmx .rdf และ .rep ตามลำดับ โดยที่ไฟล์เมนูฟอร์มจะทำงานอยู่บนไฟล์ฟอร์มเสมอ ไฟล์รันไทม์เกิดจากการแปล (Generate) จากไฟล์โค้ดต้นฉบับ ขณะรันไทม์ไฟล์รันไทม์ต่าง ๆ จะทำงานได้ด้วยไฟล์สำหรับรันโดยเฉพาะ คือสำหรับไฟล์รันไทม์ของฟอร์มจะใช้ไฟล์ ifrun60.exe ส่วนไฟล์รันไทม์ของรีพอร์ตจะใช้ Rwsxu60.exe ซึ่งมากับแอปพลิเคชันของออราเคิล แต่ในที่นี้ไม่ได้แสดงไว้ในใดอะแกรม

ไฟล์ดังกล่าวนี้จะสอดคล้องกับคลาสในวิวเลเยอร์ที่ออกแบบในบทที่ 4 และมีรูปแบบดัง ฟอร์ม เมนู และรายงาน ในบทที่ 6

7.3 ดีพลอยเมนใดอะแกรม

ดีพลอยเมนใดอะแกรมของระบบตั้งหนี้โครงการของบัญชีลูกหนี้การเคหะแห่งชาติ แสดงการออกแบบสถาปัตยกรรมทางกายภาพของระบบ ที่เป็นไคลเอน/เซิร์ฟเวอร์ ดังรูปที่ 7.2



รูปที่ 7.92 ดีพลอยเมนใดอะแกรมระบบตั้งหนี้โครงการของบัญชีลูกหนี้การเคหะแห่งชาติ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แยกพิจารณาฮาร์ดแวร์ และซอฟต์แวร์ของฝั่งเซิร์ฟเวอร์ และฝั่งไคลเอน อย่างน้อยควรเป็นดังนี้

1. ฝั่งเซิร์ฟเวอร์

- เครื่องคอมพิวเตอร์ที่เป็นดาตาเบสเซิร์ฟเวอร์ 1 เครื่อง
CPU ประสิทธิภาพ Intel Pentium 166 MHz ขึ้นไป
RAM ไม่ต่ำกว่า 96 MB
HD ความจุไม่ต่ำกว่า 1 GB
- ซอฟต์แวร์ระบบปฏิบัติการ Window NT 4.5
- ซอฟต์แวร์ระบบจัดการฐานข้อมูลเชิงสัมพันธ์ Oracle Server 8.0
- อุปกรณ์เชื่อมต่อเครือข่าย (Network Card)
- สายเชื่อมต่อ

2. ฝั่งไคลเอนท์

- เครื่องคอมพิวเตอร์ที่เป็นไคลเอนท์
CPU ประสิทธิภาพ Intel Pentium 100 MHz ขึ้นไป
RAM ไม่ต่ำกว่า 64 MB
HD ความจุ 500 MB ขึ้นไป
- เครื่องพิมพ์รวม
เครื่องพิมพ์แบบเลเซอร์
- ซอฟต์แวร์ระบบจัดการฐานข้อมูล Oracle Server 8.0
- อุปกรณ์เชื่อมต่อเครือข่าย (Network Card)
- สายเชื่อมต่อ
- ซอฟต์แวร์ระบบปฏิบัติการ Window 95 หรือ Window NT Workstation
- ซอฟต์แวร์ไคลเอนท์ของออราเคิล (Oracle Client)
- ไฟล์รันไทม์ของฟอร์ม (.fmx ,.mmx)
- ไฟล์รันไทม์ของรีพอร์ต (.rep)

บทที่ 8

สรุปโครงการพัฒนาซอฟต์แวร์

8.1 ผลการพัฒนาซอฟต์แวร์

1. จากการศึกษาความต้องการของผู้ใช้ระบบงานเพื่อนำมาเป็นส่วนที่ใช้สำหรับการพัฒนา
งานจริง โดยการสัมภาษณ์จากเจ้าหน้าที่ของบัญชีกองบัญชีลูกหนี้ และรวบรวมเอกสาร
รายงาน ต่าง ๆ ของระบบเดิม วิเคราะห์ปัญหาในระบบปัจจุบันพบว่าเหตุผลส่วนใหญ่มาจากการ
กำหนดรหัสข้อมูลที่ยังมีความหมายลงไป และได้ออกแบบการกำหนดรหัสให้เป็นอิสระ
มากขึ้น
2. ใช้ไดอะแกรมของยูเอ็มแอล ในการวิเคราะห์และออกแบบระบบเชิงวัตถุ ดังนี้
 - แอคติวิตี้ไดอะแกรม อธิบายภาพรวมของระบบ และขั้นตอนการ
ทำงานของโมดูลในสตอโปรซีเจอร์
 - ยูสเคสไดอะแกรม อธิบายภาพหน้าที่การทำงานของระบบ และผู้ติด
ต่อระบบคือเจ้าหน้าที่บัญชี ทำให้เห็นการแบ่ง
ส่วนการทำงาน แต่ละยูสเคสจะประกอบด้วยเหตุ
การณ์ และกิจกรรมของระบบ ใช้หาคาสของ
ระบบ
 - อินเทอแรกชันไดอะแกรม อธิบายเหตุการณ์การทำงานในแต่ละยูสเคส
และนำไปใช้วิเคราะห์คลาส ทำให้ทราบแอทริ
บิวต์ และการดำเนินการที่จะทำให้ระบบทำงาน
ได้ตามความต้องการ
 - คลาสไดอะแกรม วิเคราะห์ และออกแบบคลาสของระบบ ได้แก่
คลาสในบิสซิเนสเลเยอร์ ใช้ออกแบบโครงสร้าง
ฐานข้อมูล คลาสในแอคเซสเลเยอร์ใช้ออกแบบ
โมดูล และคลาสในวิวเลเยอร์ ใช้ออกแบบยูส
เซอร์อินเตอร์เฟส

- คอมพิวเตอร์ไคอะแกรม อธิบายองค์ประกอบระบบซอฟต์แวร์ที่พัฒนา ที่ประกอบด้วยไฟล์โค้ดต้นฉบับ และไฟล์รันไทม์
- ดีพลอยเมนไคอะแกรม นำเสนอการสถาปัตยกรรมของระบบแบบไคลเอน/เซิร์ฟเวอร์

3. ออกแบบโครงสร้างฐานข้อมูลโดยการแปลงจากคลาสในบิสซิเนสเลเยอร์เป็นคำสั่งภาษานิยามข้อมูล
4. ออกแบบยูสเซอร์อินเตอร์เฟซโดยอาศัยคลาสในวิวเลเยอร์ ได้แก่ การออกแบบเมนู ฟอรั่ม ยูสเซอร์อินเตอร์เฟซ และแบบรายงานต่าง ๆ
5. ออกแบบโมดูลโดยอาศัยการดำเนินการของคลาสในแอคเซสเลเยอร์ ซึ่งจะกลายเป็นสโตโปรซีเจอร์ ซึ่งอยู่ในรูปแบบของแพ็คเกจของอราเคิล 8 ที่เขียนด้วยภาษา PL/SQL
6. พัฒนาด้านแบบซอฟต์แวร์ระบบตั้งนี้โครงการของบัญชีลูกหนี้การเคหะแห่งชาติ ด้วย Oracle Developer 6 ใช้กับระบบจัดการฐานข้อมูลของอราเคิล 8 เป็นสถาปัตยกรรมแบบไคลเอน/เซิร์ฟเวอร์

8.2 ปัญหาและอุปสรรคในการดำเนินงาน

ปัญหาและอุปสรรคในระหว่างการทำงานพัฒนาซอฟต์แวร์ระบบ ได้แก่

1. เนื่องจากต้องมีการสอบถามถึงรายละเอียดต่าง ๆ แต่ผู้ที่ให้ข้อมูล และผู้ให้สัมภาษณ์ไม่ใช่เจ้าหน้าที่บัญชีที่เป็นเจ้าของระบบงานนี้ แต่เป็นเจ้าหน้าที่ที่ทำงานอยู่ในกองบัญชีลูกหนี้ และพอทราบระบบงาน เอกสารข้อมูลที่ได้รับอยู่ในหลายรูปแบบ ต้องนำมาปะติดปะต่อกัน และเกี่ยวกับการกำหนดรหัสข้อมูลรหัสลูกหนี้ยังต้องอยู่ในรูปแบบที่ฟังความหมายลงไป เพราะเป็นข้อกำหนดของหน่วยงานเอง ซึ่งจะเก็บถึง 14 หลัก
2. การวิเคราะห์ และออกแบบระบบที่ใช้ไปแนวคิดใหม่ซึ่งผู้จัดทำยังไม่ชำนาญ ดังนั้นผลการวิเคราะห์และออกแบบเชิงวัตถุที่ได้อาจจะยังไม่สมบูรณ์ หรือมีข้อผิดพลาดบ้าง
3. เครื่องมือที่ใช้ในการพัฒนาสามารถช่วยงานได้ระดับหนึ่ง แต่ไม่ทั้งหมด เช่น Rational Rose ใช้สร้างไคอะแกรมเพื่อการวิเคราะห์ และออกแบบระบบ แต่ไม่สนับสนุนการสร้างแอปพลิเคชันด้วย Developer ของอราเคิล อย่างที่สนับสนุนภาษา Java C++ และ VB ดังนั้นทำให้ต้องใช้เวลา และความพยายามมากกว่า ซึ่งการออกแบบคลาสในวิวเลเยอร์ ต้องกำหนดแอททริบิวต์ และการดำเนินการเอง และยูสเซอร์อินเตอร์เฟซที่ใช้นี้มีรายละเอียดมากมายที่จะสร้างได้สมบูรณ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้คัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4. การพัฒนาซอฟต์แวร์ด้วยผลิตภัณฑ์ของออรากิล ต้องอาศัยความรู้ และกฎเกณฑ์เฉพาะหลายอย่าง ทำให้ใช้เวลานานในการศึกษา

8.3 ประโยชน์ที่ได้รับ

ประโยชน์ที่ได้รับจากการวิเคราะห์และออกแบบ มีดังนี้

1. เนื่องจากการวิเคราะห์ และออกแบบเชิงวัตถุ สามารถใช้เครื่องมือของ Rational rose สร้างไคอะแกรมของยูเอ็มแอลเพื่ออธิบายการวิเคราะห์และออกแบบ ที่มีประสิทธิภาพ สามารถสืบค้นได้ และช่วยในการจัดทำเอกสาร
2. เป็นต้นแบบให้ผู้ที่จะทำการพัฒนาระบบต่อไป โดยไม่ต้องศึกษาระบบใหม่ทั้งหมด
3. มีการจัดการระบบสารสนเทศที่ดี สามารถค้นหา เข้าถึงข้อมูลได้สะดวกรวดเร็ว และตรวจสอบข้อมูลได้
4. ลดความซ้ำซ้อนจากระบบเดิมที่มีการเก็บข้อมูล และมีการทำงานซ้ำซ้อนกัน ได้ระบบฐานข้อมูลเป็นอันเดียว เรียกใช้บริการจากดาตาเบสเซิร์ฟเวอร์ โดยเฉพาะมีการออกแบบโมดูลที่ทำงานกับฐานข้อมูลไว้เป็นสตอโปรซีเจอร์

8.4 ข้อเสนอแนะ

หลังจากโครงการพัฒนาระบบซอร์ฟแวร์ผู้พัฒนามีข้อเสนอแนะซึ่งคิดว่าจะมีประโยชน์ต่อผู้สนใจ ดังนี้

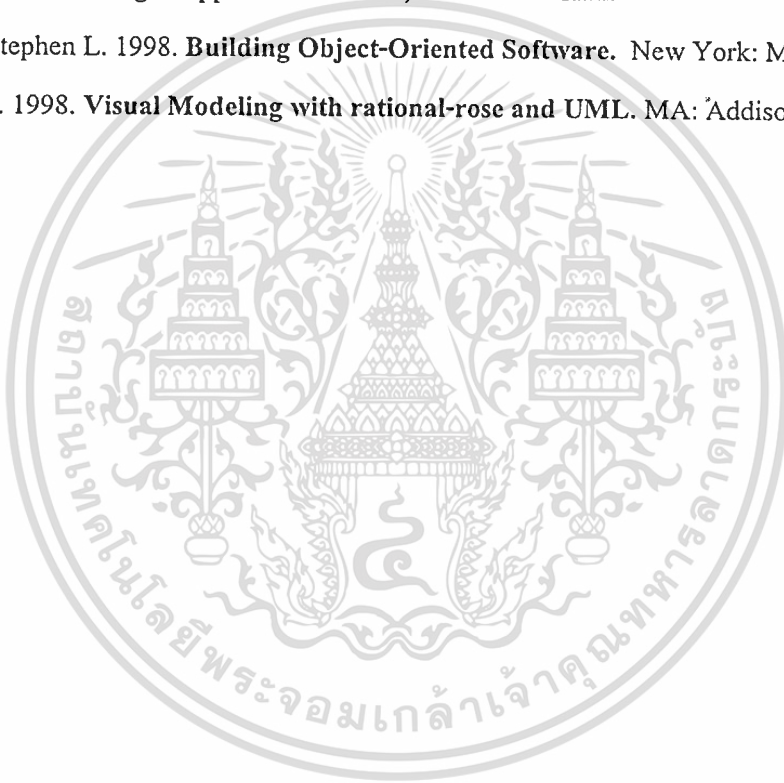
1. การพัฒนาซอฟต์แวร์ให้ประสบผลสำเร็จจะต้องให้ผู้ที่มีส่วนร่วม เพราะผู้ใช้คือเจ้าของระบบ เพื่อให้ได้ระบบที่ตรงความต้องการ และสามารถนำไปใช้ได้จริง
2. นอมนัลไลซ์คลาสไดอะแกรมดี ก็จะได้การออกแบบฐานข้อมูลที่ดียิ่ง และการใช้เครื่องมือของ Rational Rose ในการสร้างคำสังนิยามโครงสร้างฐานข้อมูลของออรากิล นอกจากจะใช้รันผ่าน Rational Rose แล้วสามารถ นำสคริปต์ไฟล์นามสกุล .sql นี้ไปรันเองต่างหาก โดยสามารถปรับเปลี่ยนตามความเหมาะสมได้
3. ควรมีการทำเอกสารควบคู่ไปกับการพัฒนา ในที่นี้คือระหว่างการวิเคราะห์ และออกแบบ เมื่อใช้ Rational Rose เป็นเคสทูล สามารถทำเอกสารไปในตัว ทำให้สามารถอ้างอิงได้ภายหลัง สามารถแก้ไข และนอกจากนี้ยังสามารถสร้างเอกสารด้วยเครื่องมือที่ชื่อ SoDa ซึ่งต้องติดตั้งไว้ด้วย

4. การพัฒนาต้องอาศัยประสิทธิภาพหลายอย่าง เช่นความรู้เกี่ยวกับการออกแบบฐานข้อมูล สามารถประกอบในการออกแบบคลาส ความรู้ในการโปรแกรมภาษาจะช่วยให้ออกแบบระบบที่สอดคล้องกับการสร้างแอปพลิเคชัน เป็นต้น
5. ทีมงาน และผู้ให้คำปรึกษา ช่วยแนะนำ และแลกเปลี่ยนเทคโนโลยีในการพัฒนา ทำให้การพัฒนาประสบความสำเร็จด้วยดี



บรรณานุกรม

- Bahrami, Ali. 1999. **Object Oriented Systems Development**. New York: McGraw-Hill
- Dittman, Kevin C. **Systems Analysis and Design Method**. New York: McGraw-Hill
- Eriksson, Hans-Erik and Magnus Penker . 1998. **UML toolkit**. New York: John Wiley & Sons, Inc.
- Larman, Craig. 1997. **Applying UML and Patterns : An Introduction to Object-Oriented Analysis and Design**. Upper Saddle River, NJ: Prentice Hall.
- Montgomery, Stephen L. 1998. **Building Object-Oriented Software**. New York: McGraw-Hill.
- Quatrani, Terry. 1998. **Visual Modeling with rational-rose and UML**. MA: Addison-Wesley.



ประวัติผู้เขียน

ชื่อผู้เขียน	นางสาววิวรรณ เหล่าตั้งจิตตรง
วันเดือนปีเกิด	24 พฤศจิกายน พ.ศ. 2514
สถานที่เกิด	โรงพยาบาลหมอเล็ก อ.เมือง จ.ชุมพร
สถานที่สำเร็จการศึกษาระดับปริญญาตรี	คณะวิทยาศาสตร์
	ภาควิชาคณิตศาสตร์
	มหาวิทยาลัยสงขลานครินทร์ วิทยาเขตหาดใหญ่
ปีที่สำเร็จการศึกษา	ปีการศึกษา 2537

