การออกแบบและการพัฒนาระบบตั๋วสัญญาใช้เงินด้วย Unified Modeling Language

Design and Develop Promissory Note System Using Unified Modeling Language

โดย

นาย อาคม คำทิพย์

รหัส 42067174

H001817

อาจารย์ที่ปรึกษา

อ. อัครินทร์ คุณกิตติ

รายงานนี้เป็นส่วนหนึ่งของวิชาโครงการพัฒนาระบบงาน

หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ

ภาคเรียนที่ 2 ปีการศึกษา 2547

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ชื่อหัวข้อ	การออกแบบและพัฒนาระบบตั๋วสัญญาใช้เงินด้วย Unified Modeling Language
นักศึกษา	นาย อาคม คำทิพย์
อาจารย์ที่ปรึกษา	อ. อัครินทร์ คุณกิตติ
ระดับการศึกษา	วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
แขนงวิชา	วิทยาการสารสนเทศ
ปีการศึกษา	2544

บทคัดย่อ

ระบบตั๋วสัญญาใช้เงินเป็นการทำธุรกรรมด้านการรับฝาก-ถอนเงินในรูปการออกตั๋วสัญญาใช้เงิน (Promissory Note) การพัฒนาระบบงานนี้นำเสนอการใช้เทคนิคของ Unified Modeling Language (UML) เข้ามาช่วยการพัฒนาระบบตั๋วสัญญาใช้เงินตามแนวคิดเชิงวัตถุ โดยการนำเอา Business Process ของระบบตั๋วสัญญาใช้เงิน มาเริ่มทำ Problem Definition ก่อนแล้วจึงทำ Requirement Analysis หลังจากนั้นจึงเริ่มทำ Design โดยการนำ requirement ที่ได้มาออกแบบด้วยไดอะแกรมของ UML ด้วยโปรแกรม Rational Rose 2000 ซึ่งจะครอบคลุมในเรื่องของ การเปิดบัญชี, การฝากเงิน, การถอนเงิน, การคิดดอกเบี้ยของตั๋วสัญญา รวมไปถึงการออกแบบรายงานที่สามารถนำไปใช้ในการบริหารเงินได้ เมื่อออกแบบเสร็จแล้วจึงนำสิ่งที่ได้ออกแบบมาทำการ Coding โดยนำไดอะแกรมที่ได้มาทำการเขียนโปรแกรมด้วย Centura เพื่อนำไปใช้เป็น application ที่สามารถทำงานได้จริงบน platform บน Window95/98 ต่อไป

Title	Design and Develop Promissory Note System using Unified Modeling Language
Student	Mr. Arkhom Khamthip
Advisor	Arjan Akharin Khunkitti
Level of study	Master of Science in Information Technology
Major	Information Science
Academic year	2001

ABSTRACT

This project presents the usage of Unified Modeling Language (UML) for developing the object-oriented analysis and design of Promissory Note system (PN). The first phase is the problem definition which is based on the business process of Promissory Note. The second phase is the requirement analysis. This is followed by designing the system using UML. The tool used for designing is Rational Rose 2000. The designing covers opening an account, deposit and withdraw of a PN, interest calculation and report generation which can use for asset management. After completing the design , mapping the class to be a schema in relational database will be implemented and also coding phase will be taken up (based on the UML) usingusing Centura to build an application that is supported on Window95/98 platform.

กิตติกรรมประกาศ

ในการศึกษาเรื่องการออกแบบและพัฒนาระบบตั๋วสัญญาใช้เงินด้วย Unified Modeling Language ผู้จัดทำขอขอบพระคุณท่านอาจารย์อัครินทร์ คุณกิตติ ที่ได้ให้คำปรึกษาและแนะนำแนวทางในการพัฒนาระบบจนเสร็จสมบูรณ์ และโครงงานนี้ได้มีผู้เกี่ยวข้อง ที่สนับสนุนและให้ความช่วยเหลือหลายท่าน ดังนี้

- ขอกราบขอบพระคุณ คุณพ่อ คุณแม่ ที่คอยดูแลเป็นกำลังใจ และสนับสนุนในการทำงานเสมอมา
- ขอกราบขอบพระคุณคณาจารย์ทุกท่านที่ได้อบรมสั่งสอนวิชาความรู้ในแขนงวิชาต่าง ๆ ตลอดระยะเวลาการศึกษาที่สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง
- ขอขอบคุณ คุณเบญจมาภรณ์ เดชดำรงค์กุล, คุณอิสรภาพ ถิระพัฒน์, คุณพงศ์เทพ จอมพงศ์ และ พ.ท. พิรุณ นยโกวิทย์ ที่คอยเตือนสติ และ เพื่อน IS8 (ภาคสมทบ) ที่ให้คำแนะนำและช่วยเหลืองานด้านอื่น ๆ ตลอดระยะเวลาที่ได้ศึกษาร่วมกัน
- ขอขอบคุณทุกทุกคนที่ไม่ได้กล่าวถึงข้างต้นที่ให้กำลังใจและช่วยเหลือผมตลอดมา

อาคม คำทิพย์

14 กันยายน 2544

สารบัญ

สารบัญรูป

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ความเป็นมา

ปัจจุบันแนวคิดเชิงวัตถุได้ถูกนำมาประยุกต์ใช้งานในด้านการออกแบบซอฟต์แวร์เป็นอย่างมาก เพราะแนวคิดเชิงวัตถุนั้นมีประโยชน์หลายอย่างเช่น การนำซอฟต์แวร์กลับมาใช้ใหม่, ความง่ายในการเพิ่มความซับซ้อนของซอฟต์แวร์, ความง่ายในการดูแลรักษา, ความสามารถโยกย้ายซอฟต์แวร์ไปทำงานกับระบบอื่นได้ง่ายและเป็นแนวคิดที่เพิ่มประสิทธิภาพให้กับซอฟต์แวร์แต่ว่าตั้งแต่ปี ค.ศ. 1989 ถึง 1994 ได้มีผู้คิดค้นวิธีคิดในเชิงวัตถุขึ้นมาประมาณ 50 วิธี ทำให้นักวิเคราะห์ระบบ พบปัญหาว่าไม่รู้จะใช้ภาษาสัญลักษณ์ (Modeling Language) แบบใดดีที่จะตรงกับความต้องการ ดังนั้นจึงเกิดการรวมตัวของผู้เชี่ยวชาญทางแนวคิดเชิงวัตถุคือ Grady Booch, Ivar Jacobson และ James Rambaugh ได้ร่วมกันคิดภาษาสัญลักษณ์ที่สามารถเข้าใจร่วมกันได้ทั้งหมดจึงเป็นที่มาของ Unified Modeling Language (UML) [1]

ส่วนระบบงานที่นำมาใช้ในวิชาโครงการพัฒนาระบบงานก็คือ ระบบตั๋วสัญญาใช้เงิน (Promissory Note) ที่ใช้ในการทำธุรกรรมด้านการรับฝาก-ถอนเงิน ในรูปการออกตั๋วสัญญาใช้เงิน (Promissory Note) มีการจัดทำ Daily Cash Position, การคำนวณดอกเบี้ย และภาษีหัก ณ ที่จ่าย, การพิมพ์ตั๋วสัญญาใช้เงิน,การพิมพ์เช็ค,การ Renew ตั๋วสัญญา และการพิมพ์รายงานต่าง ๆ ที่สามารถนำไปใช้ประโยชน์ในการบริหารเงินได้

1.2 วัตถุประสงค์ของการพัฒนาระบบงาน

วิชาโครงการพัฒนาระบบงาน นำเสนอการออกแบบโดยใช้หลักการเชิงวัตถุซึ่งตลอดระยะเวลาที่ผ่านมานั้น การวิเคราะห์ และออกแบบระบบมีความยุ่งยาก การจัดเก็บข้อมูลต่าง ๆ รวมกันเป็นแฟ้มข้อมูล (File) ส่วนที่เป็นโปรแกรมและข้อมูลจะแยกออกจากกัน ทำให้มีปัญหาเข้าใจยากและไม่สะดวกในการนำไปประยุกต์ใช้ ต่างจากการวิเคราะห์และออกแบบระบบใช้หลักการเชิงวัตถุซึ่งจะกำหนดสิ่งต่าง ๆ ภายในขอบเขตของระบบเป็น ออบเจค ซึ่งในแต่ละออบเจคจะประกอบด้วยข้อมูล (Data) และวิธีการ (Method) การเรียกใช้วิธีการหรือแมทธอดหรือสั่งให้กระทำการต่างๆ ตามที่กำหนดคือการส่งแมสเสจ (Message)ไปยังออบเจคนั้น ๆ ดังนั้นเมื่อมีการเปลี่ยนแปลงวิธีการหรือการกระทำในแต่ละออบเจคจึงไม่มีผลกระทบกับแมสเสจที่ส่งไปยังออบเจคนั้น ๆ หรือ

ออบเจคอื่น ๆ รวมถึงความสามารถในการนำ ออบเจคกลับมาใช้ใหม่ได้อีก ซึ่งเป็นผลดีในการพัฒนาระบบงานต่าง ๆ [2]

ซึ่งเมื่อได้วิเคราะห์และออกแบบเชิงวัตถุแล้วจะใช้ UML ในการแสดงการออกแบบของระบบ ซึ่งเป้าหมายของการพัฒนาโครงการพัฒนาระบบงานนี้ เพื่อออกแบบระบบตั๋วสัญญาใช้เงิน โดยต้องสามารถ สร้างแบบจำลอง (Model) ซึ่งจะประกอบด้วย Use Case Diagram, Classes Diagram, State Diagram, Activity Diagram แล้วนำไปสร้างเป็น Application Software เพื่อสามารถใช้งานได้จริงต่อไป

1.3 ขอบเขตของการพัฒนาระบบงาน

ระบบงานที่นำมาพัฒนาระบบงานนี้ เป็นระบบงานที่จะนำไปใช้ในสถาบันการเงินที่มีบริการทางการเงินในรูปแบบของตั๋วสัญญาใช้เงิน ซึ่งจะทำงานในรูปแบบของไคลน์/เซอฟเวอร์ โดยจะมีผู้ใช้งานในระบบอย่างน้อย 2 คนขึ้นไป โดยคนหนึ่งจะเป็นผู้กรอกข้อมูล และอีกคนหนึ่งจะเป็นผู้ตรวจสอบข้อมูล

การออกแบบ จะใช้แนวคิดเชิงวัตถุและใช้ไดอะแกรมของ UML เข้ามาช่วยการออกแบบ และใช้โปรแกรม Rational Rose 2000 เป็นเครื่องมือสำหรับการออกแบบไดอะแกรมของ UML การเขียนโปรแกรมจะใช้โปรแกรม Centura 1.5.1 โดยจะเขียนโปรแกรมในเชิงวัตถุเช่นเดียวกัน การพิมพ์รายงาน จะใช้โปรแกรม Centura Report Builder 1.5 Engine ส่วนระบบฐานข้อมูล ใช้ฐานข้อมูลของ Microsoft Access 97 ซึ่งเป็นฐานข้อมูลแบบรีเลชันเนล เป็นตัวเก็บข้อมูลของระบบ

1.4 ประโยชน์ที่คาดว่าจะได้รับจากการพัฒนาระบบ

ผู้จัดทำคาดว่าการทำวิชาโครงการพัฒนาระบบงานในหัวข้อการพัฒนาระบบตั๋วสัญญาใช้เงินด้วย Unified Modeling Language จะเป็นการนำเอาความรู้ความเข้าใจในหลักการเชิงวัตถุแสดงออกมาในรูปของแบบจำลองด้วย UML แล้วนำมาพัฒนาให้เป็นโปรแกรมประยุกต์ให้สามารถนำไปเป็นแนวทางในการใช้งานและพัฒนาต่อไป ทำให้สะดวกต่อการทำความเข้าใจ และช่วยทำให้การเขียน, การแก้ไข และ การเพิ่มขนาดของโปรแกรมทำให้สะดวกรวดเร็ว และสามารถนำไปประยุกต์ใช้งาน ในสถาบันการเงินที่มีบริการทางการเงินในเรื่องของตั๋วสัญญาใช้เงินได้

บทที่ 2

Unified Modeling Language

ในบทนี้จะกล่าวถึงหลักการเชิงวัตถุที่ได้นำมาใช้ในการออกแบบระบบตั๋วสัญญาใช้เงิน และการใช้ UML ในการแสดงสิ่งต่าง ๆ ที่ได้ออกแบบในเชิงวัตถุ การทำความเข้าใจถึงหลักการเชิงวัตถุ เป็นสิ่งจำเป็นอย่างยิ่งในการออกแบบโปรแกรมเชิงวัตถุ การนำ UML เข้ามาช่วยแสดงการออกแบบนั้นเป็นสิ่งสำคัญรองลงมา เพราะเป็นการถ่ายทอดความคิดของผู้ออกแบบ ออกมาให้เป็นรูปร่างในรูปแบบของไดอะแกรม เพื่อที่จะได้นำไดอะแกรมเหล่านี้ไปเป็นต้นแบบในการพัฒนาเขียนโปรแกรมต่อไป

2.1 หลักการเชิงวัตถุ

เนื่องจากระบบซอฟต์แวร์ในปัจจุบันมีขนาดใหญ่และซับซ้อนมากขึ้น จึงทำให้งานในขั้นตอนของการวิเคราะห์ระบบกลายเป็นงานที่ยุ่งยาก และใช้เวลามากขึ้นในการจัดทำ แนวความคิดใหม่ของนักวิเคราะห์และพัฒนาระบบปัจจุบัน ต้องการวิเคราห์ระบบที่สะดวกและง่ายต่อการทำความเข้าใจ การศึกษาและทำความเข้าใจระบบงานก่อนการดำเนินการออกแบบ ถือเป็นความจำเป็นหรือเป็นปัจจัยที่สำคัญในการที่จะทำให้ระบบงานนั้นประสบผลสำเร็จอย่างมีประสิทธิภาพ อีกทั้งยังต้องสามารถนำไปประยุกต์ในการออกแบบและเขียนโปรแกรม ดังนั้นการวิเคราะห์และออกแบบด้วยหลักการเชิงวัตถุ จึงเป็นวิธีหนึ่งที่สามารถช่วยให้นักวิเคราะห์และพัฒนาระบบ นำเสนอผลงานได้ตรงกับความต้องการมากที่สุด

2.1.1 วิวัฒนาการหลักการเชิงวัตถุ

หลักการเชิงวัตถุเริ่มต้น และพัฒนามาจากโปรแกรมภาษา Simula ในปี ค.ศ. 1976 ซึ่งเป็นภาษาที่ใช้ในการเลียนแบบเหตุการณ์ต่าง ๆ ต่อมาในปี ค.ศ. 1970 ได้มีการพัฒนาภาษา Smalltalk ให้ได้ใกล้กับแนวคิดของหลักการเชิงวัตถุยิ่งขึ้น ต่อมาในช่วงทศวรรษที่ 1980 บริษัทซีร็อกส์ ได้นำระบบยูสเซอร์อินเตอร์เฟสแบบ WIMP (Windows Icons Mice and Pointer) มาพัฒนาร่วมกับโปรแกรมภาษา Smalltalk ทำให้การพัฒนาระบบงานสะดวกมากยิ่งขึ้น โดยมีการอ้างอิงกราฟฟิกในการติดต่อกับผู้ใช้งาน ในปัจจุบัน เครื่องมือสำหรับพัฒนาโปรแกรมจะสนับสนุนการเขียนโปรแกรมเชิงวัตถุ ไม่ว่าจะเป็น Delphi, Visual Basic, Visual C++, Java, Centura และอื่น ๆ

2.1.2 แนวความคิดของหลักการเชิงวัตถุ

แนวความคิดหลักของหลักการเชิงวัตถุ มีความแตกต่างจากแนวความคิดเดิมในการพัฒนาระบบงาน เนื่องจากโปรแกรมภาษาแบบเดิมเป็นโปรแกรมแบบโครงสร้าง StructureProgramming) ส่วนที่เป็นโปรแกรมและข้อมูลถูกแยกออกจากกัน ขณะที่แนวคิดเชิงวัตถุจะมีออบเจคซึ่งรวมข้อมูลและวิธีดำเนินการไว้ด้วยกัน โดยมีคลาส เป็นตัวกำหนดคุณสมบัติของออบเจค

2.1.3 คุณสมบัติของหลักการเชิงวัตถุ

คุณสมบัติเบื้องต้นของหลักการเชิงวัตถุ จะต้องมีการกำหนดรายละเอียดและขั้นตอนของการประมวล ว่าขั้นตอนอะไรบ้าง เช่น

2.1.3.1 มีการกำหนดข้อมูลที่ใช้งาน เช่น วัตถุที่เราสนใจอยู่นั้น ใช้ข้อมูลอะไรบ้าง จำนวนเท่าไหร่

2.1.3.2 มีการปรับปรุงข้อมูลอย่างไร เช่น วัตถุจะมีการเปลี่ยนแปลงข้อมูล ภายในวัตถุนั้นได้อย่างไร, ด้วยวิธีใด หรือ จะต้องทำอะไรบ้างในการปรับปรุงข้อมูล

2.1.3.3 มีฟังก์ชันอะไรที่เกี่ยวข้อง เช่น วัตถุนั้นมีหน้าที่อะไรบ้าง หรือ มีการทำงานกับวัตถุอื่นได้อย่างไร

2.1.3.4 มีการแสดงผลของข้อมูลอย่างไร เช่น วัตถุนั้นจะมีการแสดงข้อมูลออกมาแบบใด หรือว่าอยู่ในรูปแบบใด

2.1.4 ส่วนประกอบของหลักการเชิงวัตถุ

หลักการเชิงวัตถุหมายถึง หลักการที่ทำให้ข้อมูลและวิธีการในการดำเนินงานสามารถรวมอยู่ด้วยกันในออบเจค การเข้าถึงข้อมูลต่าง ๆ ต้องผ่านโอเปอเรชัน หรือ แมทธอท ของออบเจคนั้น ๆ ซึ่งจะรวมถึงการถ่ายทอดคุณสมบัติต่าง ๆ ของออบเจคนั้น ๆ ด้วย

การวิเคราะห์ระบบ ด้วยหลักการเชิงวัตถุคือ การศึกษาโดเมนจำเพาะของปฏิกิริยา ของออบเจคเพื่อความเข้าใจ และจัดทำเอกสารเกี่ยวกับลักษณะเฉพาะของปฏิกิริยาระหว่าง ออบเจค ผลการวิเคราะห์งานที่ได้ จะมีความสัมพันธ์แบบตรงไปตรงมากับวัตถุที่มีอยู่จริง ซึ่งทำให้ง่ายต่อการทำความเข้าใจ และนำไปประยุกต์ใช้ในการออกแบบและการเขียนโปรแกรม ดังนั้นการวิเคราะห์ระบบด้วยหลักการเชิงวัตถุประกอบด้วย การศึกษา การทำความเข้าใจ และการจัดทำเอกสารประกอบการพัฒนาระบบ

ปฏิกิริยาระหว่างออบเจคสามารถเข้าใจได้โดยการตรวจสอบสิ่งต่อไปนี้

2.1.4.1 ความสัมพันธ์ระหว่างออบเจค เช่น การทำงานออบเจคนั้น จะมีความสัมพันธ์กันอยู่ในรูปแบบของ Dependency หรือ Generalization

2.1.4.2 พฤติกรรมของออบเจคแต่ละออบเจค คือ หน้าที่ของออบเจคนั่นเอง

2.1.4.3 พฤติกรรมที่มีร่วมกันของออบเจค คือ การทำงานของออบเจคที่มีการถ่ายทอดคุณสมบัติออกไป ซึ่งออบเจคที่ถ่ายทอดคุณสมบัติออกไปนั้นจะมีพฤติกรรมร่วมเหมือนกับตัวต้นแบบนั่นเอง

การวิเคราะห์ระบบด้วยการวิเคราะห์เชิงวัตถุ เป็นการวิเคราะห์ที่เน้นไปที่คำว่า What คือมีออบเจคอะไรอยู่จริงในระบบ มากกว่าเป็นการวิเคราะห์ที่เน้นไปที่คำว่า How คือต้องทำอย่างไร หรือว่าเป็นการซ่อนวิธีการทำงานต่าง ๆ ไว้ภายในออบเจค โดยสามารถใช้งานเฉพาะ method ที่กำหนดให้ได้ นั่นเอง

2.2 Unified Modeling Language (UML)

UML ย่อมาจาก Unified Modeling Language ซึ่งเป็นภาษาภาพ (Graphical Language) ใช้สำหรับ แสดงการออกแบบ (Visualizing), กำหนดคุณลักษณะระบบ (Specification), กำหนดการออกแบบและพัฒนา (Construction), บันทึกรายละเอียดทั้งหมดของโครงการ (Documentation). UML จะมีมาตรฐานในการกำหนดสัญลักษณ์, คำศัพท์, เทคนิค, กฎเกณฑ์ ที่ช่วยเราสามารถอธิบายสิ่งที่เป็น Conceptual เช่น กระบวนการทางธุรกิจ (Business Process) หรือเป็น concrete เช่น ภาษาโปรแกรมที่เขียนเป็นคลาส โดยจะแสดงแนวความคิดเป็น แบบจำลอง (Model) ที่สามารถสื่อสารเพื่อความเข้าใจกันระหว่างผู้ใช้งาน, นักวิเคราะห์ระบบ และโปรแกรมเมอร์ได้

UML ได้รวมเอาแบบจำลองต่าง ๆ มาไว้ด้วยกันเช่น Data Modeling concepts (Entity Relationship diagrams), Business Modeling (Work Flow), Object Modeling, Component Modeling. UML คือ ภาษาแบบจำลอง (Modeling Language) ซึ่ง UML ไม่ใช่ Method เพราะ Method ประกอบด้วย Principle และ Process แต่ ภาษาแบบจำลองเป็น สัญลักษณ์ (Notation) ที่ใช้แสดงสิ่งที่ได้ออกแบบมา

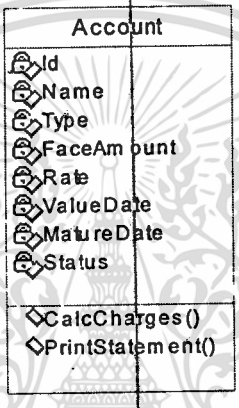
องค์ประกอบของ UML

- สิ่งของ (Thing)
- ความสัมพันธ์ (Relationship)
- แผนผัง (Diagram)

2.2.1 สิ่งของ (Thing) จะประกอบไปด้วย

2.2.1.1 สิ่งของที่มีโครงสร้าง (Structural Thing) คือสิ่งของใน UML ที่มีลักษณะ Static ประกอบไปด้วย 7 ชนิดคือ

- Class เป็นการแทนกลุ่มของ Object ที่มี Attribute เดียวกัน, มี Operation ชุดเดียวกัน และความสัมพันธ์และความหมายแบบเดียวกัน รูปร่างของ Class ในสัญลักษณ์สี่เหลี่ยม ภายในจะแบ่งเป็น 3 ส่วนซึ่งส่วนบนแทนชื่อ ซึ่งชื่อของ Class จะต้องไม่ซ้ำกับ Class ตัวอื่น ส่วนต่อมาแทน Attribute และส่วนล่างสุด UML ใช้แทน Operation



รูปที่ 2.1 Class

- **Interface** เป็นกลุ่มของ Operation ที่ Class หรือ Component ให้บริการ ดังนั้น Interface คือส่วนของ Class ที่ภายนอกสามารถใช้ได้และเห็นได้ แต่จะไม่ทราบถึงกลไกการทำงานภายใน ใช้สัญลักษณ์ วงกลมพร้อมชื่อ

รูปที่ 2.2 Interface

- **Use case** เป็นการแสดงพฤติกรรมของผู้ใช้งานกับระบบเพื่อให้ได้งานแบบใดแบบหนึ่ง Use case จะต้องอาศัย ความร่วมเมือ (Collaboration) ระหว่าง Actor กับ Class ต่าง ๆ ใช้สัญลักษณ์วงรีเส้นทึบพร้อมชื่อ

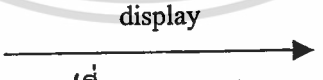
รูปที่ 2.3 Use case

- **Component** คือส่วนของระบบที่สามารถใช้แทนกันได้ (Replaceable) ซึ่งจะประกอบด้วยกลุ่มของ Interface ที่ใช้งานได้จริง มีสัญลักษณ์เป็นกล่องสี่เหลี่ยมและมี Tab คั่นอยู่พร้อมชื่อ

รูปที่ 2.4 Component

2.2.1.2 สิ่งของที่มีพฤติกรรม **(Behavioral Thing)** สิ่งของที่แสดงพฤติกรรมเป็นลักษณะ Dynamic ซึ่งจะแสดงในเวลาและสถานการณ์ต่าง ๆ สามารถแบ่งได้ 2 ประเภทคือ

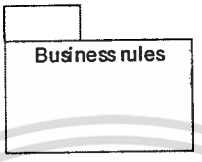
- **Interaction** เป็นพฤติกรรมที่ประกอบด้วยกลุ่มของ message ที่แลกเปลี่ยนระหว่างกลุ่มของ Object เพื่อให้บรรลุวัตถุประสงค์. Interaction จะเกี่ยวข้องกับ Message Action Sequence และ Link ระหว่าง Object. มีสัญลักษณ์คือ เส้นตรงมีหัวลูกศรพร้อมชื่อของ Operation และ Message



รูปที่ 2.5 Interaction

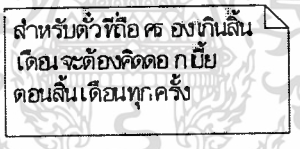
- **State Machine** เป็นการแสดงพฤติกรรมของ Object ในรูปแบบที่ต้องอยู่ใน State ต่าง ๆ เนื่องจากต้องตอบสนองต่อ Event และ ต้องทำงานตามสถานภาพและรูปแบบของ Event ที่ได้รับ แสดงสัญลักษณ์ ด้วยสี่เหลี่ยมมีมุมมนพร้อมชื่อ

2.2.1.3 สิ่งของที่เป็นกลุ่ม (Grouping Thing) เป็นการรวมชิ้นส่วนต่าง ๆ ของ UML เข้าด้วยกันเป็น Package แต่ Package จะไม่เหมือนกับ Component (ซึ่งจะพบได้ในขณะ Run Time) Package จะเป็น conceptual ล้วน ๆ (ซึ่งหมายความว่าจะพบเฉพาะเวลาพัฒนาโครงการเท่านั้น) มีสัญลักษณ์คือรูปแฟ้มพร้อมชื่อ



รูปที่ 2.6 Package

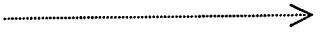
2.2.1.4 สิ่งของที่ใช้อธิบาย (Annotation Things) เป็นส่วนที่ใช้สำหรับการอธิบายขยายความของ UML หรือการให้ Comment ของสิ่งของใน UML เราเรียกอีกชื่อหนึ่งว่า Note มีสัญลักษณ์คือรูปสี่เหลี่ยมมุมพับ



รูปที่ 2.7 Annotation Thing

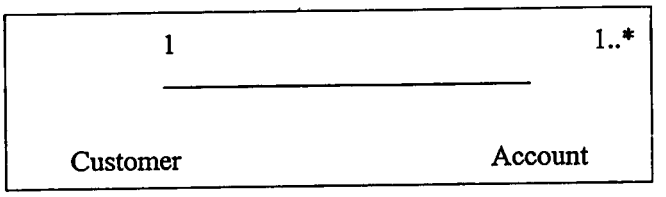
2.2.2 ความสัมพันธ์ (Relationship) ความสัมพันธ์ของสิ่งของใน UML จะประกอบไปด้วย

2.2.2.1 Dependency ให้ความหมายถึงความสัมพันธ์ระหว่างสิ่ง 2 สิ่ง ซึ่งถ้าสิ่งหนึ่งเปลี่ยนไปจะมีผลกระทบถึงอีกสิ่งหนึ่งด้วย สัญลักษณ์จะเป็นเส้นประที่มีหัวลูกศร (อาจไม่มีก็ได้) และชื่อ



รูปที่ 2.8 Dependency

2.2.2.2 Association แสดงโครงสร้างความสัมพันธ์ระหว่าง Object 2 ตัว มีสัญลักษณ์เป็นเส้นทึบมีชื่อ อาจมีหรือไม่มีหัวลูกศรก็ได้ นอกจากนั้นมักจะประกอบด้วย Adornments ซึ่งจะแสดงจำนวน (Multiplicity) และ ชื่อบทบาท (Role)



รูปที่ 2.9 Association และ Adorment

2.2.2.3 Generalization แสดงความสัมพันธ์ระหว่าง object ในเชิงทั่วไป/เฉพาะอย่าง Generalization / Specialization ซึ่ง Generalization คือส่วนของ Parent และ Specialization คือ Child ซึ่ง Child จะมีโครงสร้างและพฤติกรรมของ Parent สัญลักษณ์แทนด้วยเส้นทึบมีหัวลูกศรชี้ไปที่ Parent

รูปที่ 2.10 Generalization

2.2.3 แผนผังใน UML

UML จะแบ่ง diagram ได้เป็น model ดังนี้คือ

2.2.3.1 Functional Model เป็นโมเดลที่ใช้แสดงความต้องการของระบบทั้งหมด แสดงให้เห็นว่าระบบทำอะไรบ้าง แต่จะยังไม่บอกว่าทำอย่างไร เมื่อทำเสร็จแล้วเป็นอย่างไร เช่น การคิดดอกเบี้ยตอนสิ้นเดือน เราจะรู้ว่าใครที่รับผิดชอบงานนี้ เมื่อทำเสร็จแล้วใครจะเป็นผู้ที่มาตรวจสอบแล้ว อนุมัติการทำรายการให้สมบูรณ์ โดยเครื่องมือที่ใช้แสดง Functional Model คือ Use Case Diagram

- **Use Case Diagram** ประกอบด้วย 2 ส่วน

ก่อนที่จะทำความเข้าใจความหมายของ Use Case Diagram ควรเข้าใจกับความหมายของ scenario

scenario คือลำดับของขั้นตอนที่อธิบายปฏิสัมพันธ์ระหว่างผู้ใช้กับระบบ เช่นในระบบตั๋วสัญญาใช้เงิน เมื่อมีลูกค้าเข้ามาเปิดบัญชี เราจะมี scenario ว่า

"ลูกค้านำเอกสารที่จำเป็นในการเปิดบัญชีตั๋วสัญญาใช้เงิน พร้อมนำเงินสดมาให้กับพนักงานเปิดบัญชี แล้วพนักงานเปิดบัญชีจะตรวจหลักฐานที่นำมา แล้วออกตั๋วสัญญาใช้เงินให้กับลูกค้า"

- *Use Case* คือกลุ่มของ scenario ที่ทำงานร่วมกันโดยยึดเอาเป้าหมายของผู้ใช้ระบบเป็นหลัก
- *Actor* คือบทบาทของผู้ใช้ที่มีให้กับระบบ สิ่งที่ควรสังเกตคือต้องดูที่บทบาท แทนที่จะดูที่ตัวบุคคลหรือตำแหน่งของคน ๆ นั้น Actor สามารถเป็นได้ทั้งระบบอื่น หรืออุปกรณ์ฮาร์ดแวร์ ที่ติดต่อกับระบบปัจจุบัน. Actor จะใช้สัญลักษณ์รูปคน

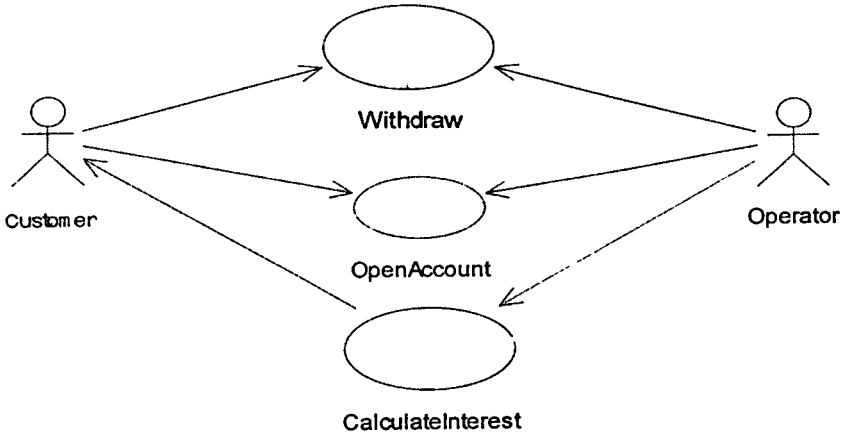
Customer

(from Use Case View)

รูปที่ 2.11 Actor

ความสัมพันธ์ใน Use Case Diagram ประกอบด้วย

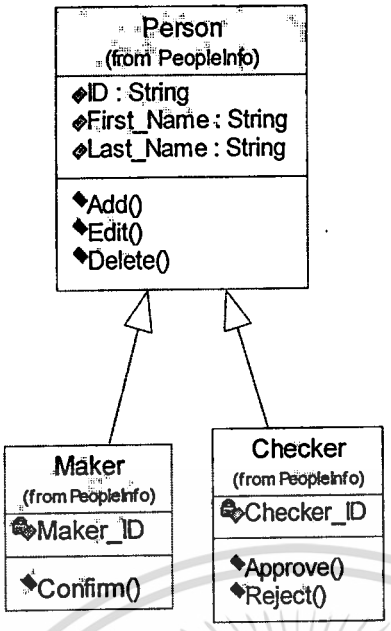
- **Include** ใช้เมื่อต้องการทำกิจกรรมที่เหมือนกันระหว่าง Use Case
- **Generalization** ใช้เมื่อมี Use Case ที่มีลักษณะคล้ายกัน ซึ่งมีความแตกต่างกันเพียงเล็กน้อย
- **Extend** จะคล้ายกับ Generalization แต่ต้องการอธิบายกฎเพิ่มเติม



รูปที่ 2.12 Use Case diagram

2.2.3.2 Object Model เป็นโมเดลที่ใช้แสดงโครงสร้างของระบบ โดยจะแสดงในรูปของ Class ที่ได้จากการพิจารณาความต้องการของระบบใน Functional Model โดยเครื่องมือที่ใช้แสดง Object Model คือ

- **Class Diagram** เป็นไดอะแกรมที่แสดงข้อมูลโครงสร้างของคลาสและความสัมพันธ์ระหว่างคลาส และข้อมูลของพฤติกรรมของข้อมูลที่แตกต่างกัน ความสัมพันธ์ระหว่างคลาสมีได้ 3 แบบ คือ
  - **Association** จะแสดงความสัมพันธ์ระหว่างคลาสในลักษณะของการรู้จักกันและมีบทบาทของคลาส ใน 2 ทิศทางคือ ถ้ามี Association จาก A ไป B ก็จะมีบทบาทของทั้ง B ไป A และ A ไป B ด้วย
  - **Depends on** หรือ **Coupling** เป็นรูปแบบของความสัมพันธ์ประเภทหนึ่งที่ใช้แสดงความสัมพันธ์กันระหว่างคลาส 2 คลาส ซึ่งมีการใช้งานที่ขึ้นตรงต่อกัน ซึ่งถ้ามีการแก้ไขคลาสหนึ่งจะส่งผลกระทบกับคลาสที่ Depends on ด้วย ซึ่งใช้สัญลักษณ์ดังนี้
  - **Generalization** หรือการสืบทอดความสัมพันธ์ (Inheritance) โดยจะเรียกคลาสที่ถูกถ่ายทอดว่าซูปเปอร์คลาส (Super Class) และเรียกคลาสที่ถ่ายทอดว่า ซับคลาส (Sub Class)



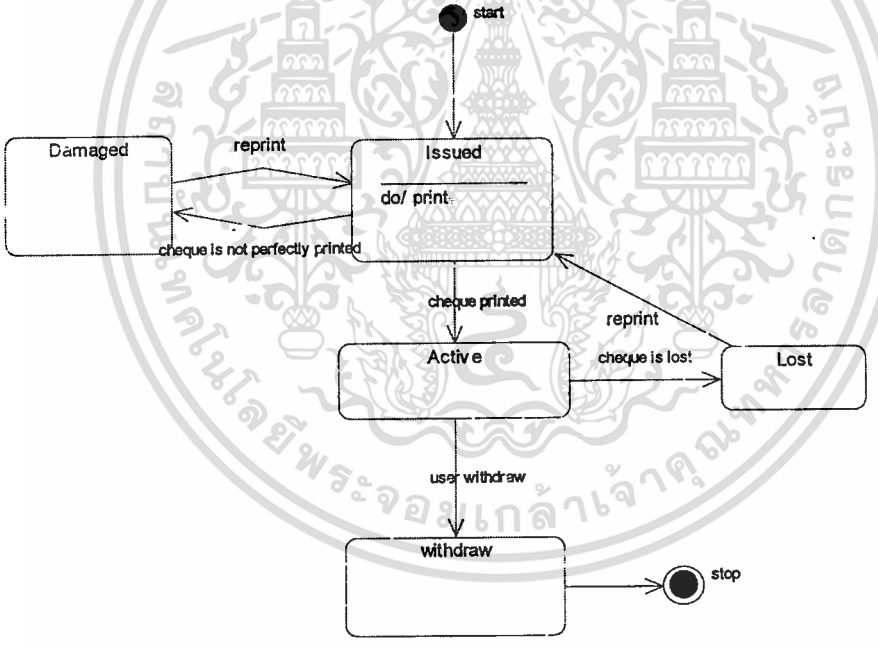
รูปที่ 2.13 Generalization

2.2.3.3 Dynamic Model เป็นโมเดลที่แสดงถึงการทำงานระหว่าง object ต่าง ๆ ตามการส่งข้อความ (Message) หรือเมื่อเหตุการณ์ (Event) ต่าง ๆ ได้เกิดขึ้น object ในที่นี้หมายถึงอินสแทนซ์ (Instace) ที่สร้างขึ้นจากคลาสที่ได้ออกแบบไว้ใน Object Model โดยแต่ละ Object จะมีคุณสมบัติเช่นเดียวกันกับต้นแบบ ที่ได้ออกแบบไว้ใน Object Model เมื่อ Object ทำการส่งข้อความไปมาระหว่าง Object ซึ่งจะทำให้ Object มีการเปลี่ยนสถานะ (State) ไปตามเหตุการณ์ที่เกิดขึ้น ซึ่งได้กำหนดไว้ใน Functional Model โดยเครื่องมือที่ใช้แสดง Dynamic Model คือ

- **Sequence Diagram** เป็นการอธิบายถึงความสัมพันธ์ของออบเจคในมิติของเวลา ซึ่งจะใช้สัญลักษณ์สี่เหลี่ยมผืนผ้าแทนออบเจคหรือคลาส (ในกรณีที่เป็นคลาสจะต้องขีดเส้นใต้) ประกอบไปด้วย
  - **Object's lifeline** สัญลักษณ์เส้นประในแนวตั้งเรียกเป็นการกำหนดขอบเขตการทำงานของออบเจคนั้น
  - **self-call** เป็นแมสเซสที่ออบเจคเรียกตัวเอง โดยจะใช้สัญลักษณ์ลูกศรที่วกกลับเข้าหาตัวเอง
  - **condition** เป็นตัวบ่งบอกว่าเมื่อใหร่ถึงจะมีการส่งแมสเซส ซึ่งจะส่งออกไปเมื่อ condition มีค่าเป็น true เท่านั้น แต่มีข้อสังเกตว่าการใช้ condition ควรใช้ในกรณีง่าย ๆ ถ้าหากมีเงื่อนไขที่ซับซ้อนแล้วควรใช้ Sequence Diagram เป็นตัวอธิบายจะดีกว่า

- **iteration marker** เมื่อต้องการแสดงว่ามีการส่งแมสเซสหลาย ๆ ครั้งไปถึงหลายออบเจค
- **return** เป็นตัวบ่งบอกสิ่งที่ได้คืนมาจากการส่งแมสเซส แต่ไม่ใช่แมสเซสใหม่ การวาด return สำหรับทุก ๆ แมสเซสไม่ใช่สิ่งที่จำเป็น เราจะวาด return ในกรณีที่คิดว่าทำให้การเข้าใจไดอะแกรมได้ดีขึ้นเท่านั้น

- **State Diagram** เป็นวงจรชีวิตของออบเจค, ซับซิสเต็ม และระบบต่าง ๆ ซึ่งสเตทจะเป็นตัวบ่งบอกถึงเหตุการณ์ต่าง ๆ ว่ามีอะไรเกิดขึ้นบ้าง สเตทไดอะแกรม จะทำหน้าที่เชื่อมต่อทุกคลาสที่มีพฤติกรรมอันซับซ้อนให้ชัดเจนขึ้น และเป็นการอธิบายถึงพฤติกรรมของระบบ ซึ่งแต่ละเสตทจะมีความแตกต่างกันขึ้นอยู่กับสถานะในปัจจุบัน รวมทั้งมีเหตุการณ์ใดบ้างที่มีผลกับการเปลี่ยนแปลงทั้งภายในและภายนอก โดยอาจจะมีจุดเริ่มต้นและจุดได้ในหลาย ๆ จุด



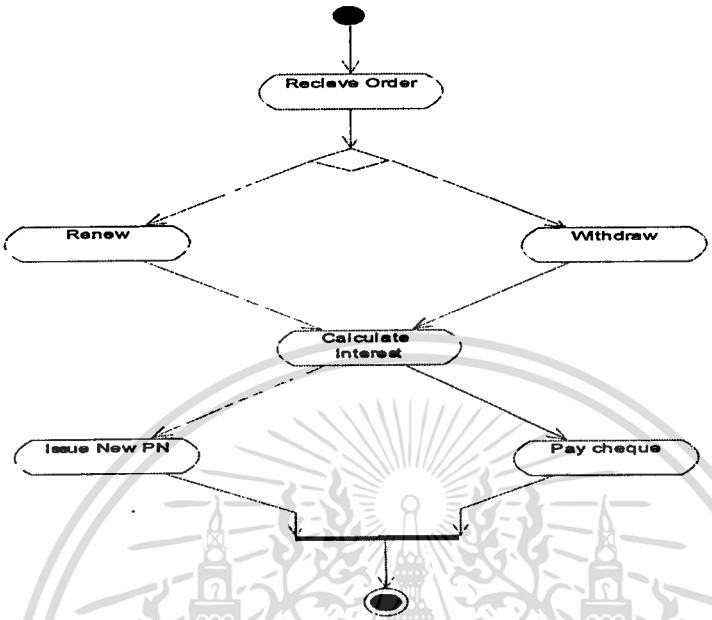
รูปที่ 2.14 State diagram

State Diagram จะมีส่วนประกอบอยู่ 3 ส่วนคือ

- **State's name** ชื่อของสเตท
- **Optional State** เป็นคุณสมบัติของคลาสใน State diagram

- **Optional Activity State** เป็นรายละเอียดของเหตุการณ์และ การกระทำต่าง ๆ (Action)สถานะ ที่ใช้เป็นมาตรฐาน ในการทำกิจกรรมต่าง มี 3 สถานะคือ
- **Initiator** คือการเข้าไปทำกิจกรรมต่าง ๆ ภายในสเตทต่าง ๆ หรือการเริ่มต้น จะใช้สัญลักษณ์วงกลมทึบ
- **Terminator** จะใช้ในกรณีที่ต้องการออกจากเสตท หรือสิ้นสุดการทำงาน จะใช้สัญลักษณ์วงกลมทึบ และมีอีก 1 วงล้อมรอบ
- **Do** ใช้ในการปฏิบัติงานต่าง ๆ ในเสตท เช่นการส่งเมสเซจ, การรอ หรือการคำนวณ

- **Activity Diagram** ใช้แสดงถึงลำดับขั้นตอนการทำงานของกิจกรรมในระบบ ซึ่งจะสามารถใช้เป็นแบบในการปฏิบัติงานในหน้าที่ต่าง ๆ รวมทั้งยังอธิบายถึงขั้นตอนการทำงานในขั้นตอนต่อ ๆ ไปของกิจกรรมอื่นด้วย Activity Diagram เป็นการพิจารณาถึงสถานะของการทำงานแต่ละ action ซึ่งเมื่อเริ่มต้นทำแล้วก็ต้องทำให้เสร็จก่อนที่จะเริ่มทำงานในสถานะต่อไปได้ ดังนั้นสิ่งที่จะใช้ควบคุมการทำงานในขั้นตอนต่าง ๆ ก็คือ การติดต่อสื่อสารกันหรือการเชื่อมโยงกันของสเตทต่าง ๆ ซึ่งสามารถกระทำได้ทั้งการส่งและรับข้อมูลข่าวสารได้พร้อม ๆ กัน โดยมีการนำแนวคิดมาจากหลาย ๆ เทคนิคเช่น Event Diagram, SDL State Modeling และ Petri Nets Diagram โดยนำมาใช้ประโยชน์ในการเชื่อมต่อกับ Flow และรายละเอียด ของพฤติกรรมที่มีการประมวลผลแบบขนาน (Parallel) คือสามารถทำงานได้หลายอย่างในเวลาเดียวกันได้เป็นจำนวนมาก จุดประสงค์หลักของไดอะแกรมนี้ก็คือ ใช้เพื่อแสดงถึงความเปลี่ยนแปลงอันเกิดจากการประมวลผลภายใน (ใช้แสดงเหตุการณ์ต่าง ๆ อันเป็นผลมาจากการกระทำภายในของระบบเอง)ในขณะที่สเตทไดอะแกรม จะใช้แสดงการกระทำที่เกิดขึ้นในลักษณะที่ไม่ต่อเนื่องกัน.



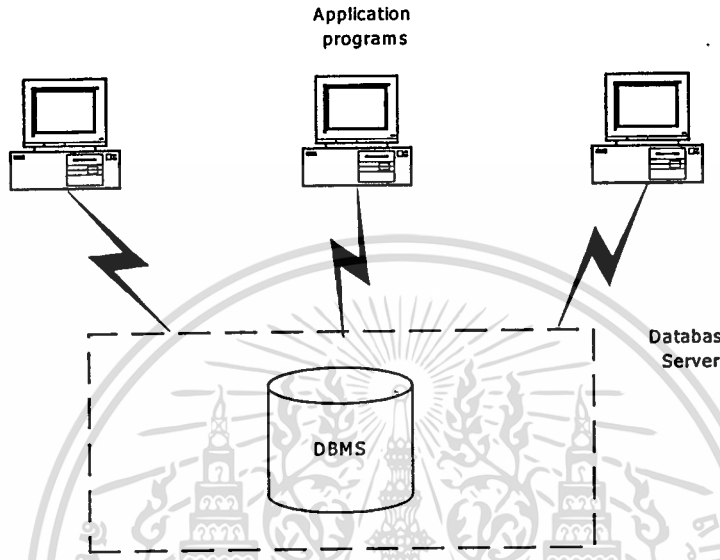
รูปที่ 2.15 Activity diagram

ทั้งหมดนี้เป็นการอธิบายถึงหลักการเชิงวัตถุและ UML ส่วนหัวข้อต่อไปเป็นการอธิบายถึงสถาปัตยกรรมของระบบไคลน์/เซอร์เวอร์แบบ SQL-Based ที่ได้นำมาใช้ในการพัฒนาโครงการนี้

2.3 ระบบไคลน์/เซอร์เวอร์แบบ SQL-Based

ระบบนี้จะเป็นการแยกส่วนโปรแกรมประยุกต์ออกจาก DBMS และ File Services โดยที่ส่วนโปรแกรมประยุกต์จะไปถูกปฏิบัติงานที่เครื่องลูกข่าย และส่วน DBMS และ File Services จะถูกนำไปปฏิบัติที่เครื่อง Database Server

ในการพัฒนาระบบงาน เมื่อเราได้ออกแบบระบบด้วยหลักการเชิงวัตถุ และแสดงด้วย UML แล้ว เราจะนำสิ่งที่ได้ออกแบบไปเขียนโปรแกรมให้เป็นโปรแกรมประยุกต์ โดยการทำงานจะเก็บข้อมูลลงในฐานข้อมูลที่ใช้ Microsoft Access 97 เป็น DBMS ดังนั้นจะเห็นได้ว่าโปรแกรมตู้สัญญาใช้เงินจะอยู่ในส่วนของ Application Program และ ฐานข้อมูล Microsoft Access 97 จะอยู่ในส่วนของ DBMS ดังรูปที่ 2.16



รูปที่ 2.16 สถาปัตยกรรมระบบไคลน์/เซอร์เวอร์แบบ SQL-Based

ในบทนี้ทำให้เราได้ทราบถึงหลักการเชิงวัตถุ ซึ่งเป็นหัวใจสำคัญในการออกแบบโปรแกรมเชิงวัตถุ นอกจากนั้นเรายังได้รู้ถึงการใช้ UML ในการแสดงไดอะแกรมต่าง ๆ ที่ได้ออกแบบมาในเชิงวัตถุ เพื่อให้ง่ายในติดต่อสื่อสารกับบุคคลในฝ่ายต่าง ๆ และยังทำให้เกิดความเข้าใจในระบบที่ตรงกันเพราะได้แสดงการออกแบบ มาให้เป็นรูปภาพด้วย UML และสุดท้ายได้ทราบถึงสถาปัตยกรรมของระบบไคลน์/เซอร์เวอร์แบบ SQL-Base ที่ได้นำมาประยุกต์ใช้กับการพัฒนาระบบด้วย ในบทต่อไปจะอธิบายถึงการนำเอากระบวนการทำงานทางธุรกิจของตั๋วสัญญาใช้เงิน มาออกแบบด้วยแนวคิดเชิงวัตถุ และใช้ UML ในการแสดงสิ่งที่ได้ออกแบบมาด้วย

บทที่ 3
การวิเคราะห์และออกแบบ ระบบตั๋วสัญญาใช้เงิน

3.1 ระบบตั๋วสัญญาใช้เงิน (Promissory Note)

ระบบตั๋วสัญญาใช้เงิน (Promissory Note) เป็นระบบที่ใช้ในการทำธุรกรรมด้านการรับฝาก-ถอนเงิน ในรูปการออกตั๋วสัญญาใช้เงิน (Promissory Note) มีการจัดทำ Daily Cash Position, การคำนวณดอกเบี้ย และภาษีหัก ณ ที่จ่าย,การพิมพ์ตั๋วสัญญาใช้เงิน,การพิมพ์เช็ค,การ Renew ตั๋วสัญญา และการพิมพ์รายงานต่าง ๆ ที่สามารถนำไปใช้ประโยชน์ในการบริหารเงินได้ตามประมวลกฎหมายแพ่งและพาณิชย์มาตรา 982 บัญญัติไว้ว่า อันตั๋วสัญญาใช้เงินนั้นคือ หนังสือตราสารซึ่งบุคคลหนึ่งเรียกว่าผู้ออกตั๋วให้คำมั่นสัญญาว่าจะใช้เงินจำนวนหนึ่งให้แก่บุคคลอีกคนหนึ่งหรือใช้ให้ตามคำสั่งของบุคคลอีกคนหนึ่ง เรียกว่าผู้รับเงิน ซึ่งเจ้าหนี้จะกำหนดวันที่ลูกหนี้ต้องชำระหนี้แล้วยังมักจะมีการกำหนดอัตราดอกเบี้ยอีกด้วย [3]

3.2 การทำงานของระบบเดิม

การทำทุกขั้นตอนเป็นการบันทึกข้อมูลลงในสมุดบัญชี โดยไม่มีการตรวจสอบข้อมูลจากผู้ตรวจทาน จึงมีปัญหาในเรื่องการโกง หรือ การบันทึกข้อมูลผิดพลาด การทำงานของระบบเริ่มจากลูกค้าเปิดบัญชีกับทางธนาคารก่อน หลังจากนั้นลูกค้าสามารถซื้อตั๋วสัญญาใช้เงินจากทางธนาคาร ซึ่งลูกค้าสามารถซื้อตั๋วสัญญาใช้เงินจากทางธนาคารได้ไม่จำกัดจำนวน ซึ่งทางลูกค้าจะใช้หมายเลขบัญชีที่เปิดไว้ติดต่อกับทางธนาคาร และเมื่อตั๋วสัญญาที่ซื้อไปครบอายุแล้ว ลูกค้าจะนำตั๋วสัญญาฉบับนั้นมาขึ้นเงินกับทางธนาคาร ทางธนาคารจะคิดจำนวนดอกเบี้ยให้ตามที่กำหนดไว้ในตั๋วสัญญาใช้เงิน พร้อมทั้งคืนเงินต้นให้ด้วย ในกรณีที่ลูกค้าต้องการฝากต่อ ลูกค้าจะแจ้งให้เจ้าหน้าของธนาคารทราบในวันถอน ว่าต้องการฝากต่อ (Renew) ทางเจ้าหน้าที่จะต้องถอนตั๋วใบนั้นก่อนแล้วทำการฝากตั๋วใบใหม่ให้แทน ส่วนการพิมพ์เช็ค พิมพ์ตั๋วสัญญาใช้เงิน พิมพ์ใบ Inward Voucher หรือ Outward Voucher จะใช้เครื่องพิมพ์ดีดพิมพ์ ในกรณีที่ลูกค้านำตั๋วสัญญาใช้เงินที่ครบกำหนดมาทำการฝากต่อ (renew) พนักงานจะต้องทำการปิดบัญชีก่อนแล้วจึงทำการฝากตั๋วสัญญาใช้เงินใหม่ ซึ่งเป็นการทำงานที่ซ้ำซ้อน และเสียเวลา

3.3 การทำงานของระบบใหม่

การทำงานของระบบใหม่จะใช้คอมพิวเตอร์เข้ามาช่วยในการทำงานแบบ On-line และเก็บข้อมูล และการนำระบบ Maker-Checker ซึ่งเป็นการทำงานที่ใช้คนร่วมกันตรวจสอบข้อมูลให้ถูกต้องก่อนที่จะนำข้อมูลไปใช้งานในระบบ ระบบ Maker-Checker คือ ผู้ใช้ที่มีสิทธิของ Maker จะมีสิทธิบันทึกข้อมูล, แก้ไขข้อมูล และ ลบข้อมูล ซึ่งการกระทำดังกล่าวจะสมบูรณ์ก็ต่อเมื่อมีผู้ใช้ที่มีสิทธิของ Checker มาทำการอนุมัติข้อมูลนั้น ๆ โดยที่ Maker และ Checker จะต้องไม่เป็นบุคคล ๆ เดียวกัน ในการพิมพ์เช็ค พิมพ์ตั๋วสัญญาใช้เงิน พิมพ์ใบ Inward Voucher หรือ Outward Voucher จะใช้คอมพิวเตอร์ควบคุมการพิมพ์ทั้งหมด ในกรณีที่ลูกค้านำตั๋วสัญญาใช้เงินที่ครบกำหนด มาทำการฝากต่อ (renew) จะให้ระบบคอมพิวเตอร์ทำให้อัตโนมัติ

3.4 ขั้นตอนการดำเนินการพัฒนาระบบ

หลังจากได้วิเคราะห์ระบบงานเดิมในเบื้องต้นแล้ว เราจึงเริ่มขึ้นตอนในการพัฒนาระบบงาน ซึ่งในการพัฒนาระบบงาน สามารถแบ่งขั้นตอนการดำเนินงานได้ดังนี้

3.4.1 Problem Definition

3.4.2 Requirement Analysis

3.4.3 Specification

3.4.4 Design

3.4.5 Coding

3.4.6 Testing

3.4.1 Problem Definition

เป็นขั้นตอนที่จะต้องหาปัญหาที่แท้จริงของผู้ใช้งาน และทำความเข้าใจกับปัญหานั้น จึงสรุปปัญหาของผู้ใช้ได้ดังนี้

- *มีการทุจริต* เนื่องจากระบบเดิม เป็นการทำงานด้วยการบันทึกบัญชี โดยคน ๆ เดียว ไม่มีการตรวจสอบในด้านต่าง ๆ จึงเกิดปัญหาในเรื่องการโกงเงินขึ้น

- *การเรียกใช้ข้อมูล* ปัจจุบันมีจำนวนของลูกค้าเพิ่มมากขึ้น การบันทึกข้อมูลด้วยระบบเดิม ทำให้เกิดปัญหาในเรื่องการค้นหาข้อมูลเพื่อการนำมาใช้ประโยชน์
- *การทำงานบางขั้นตอนซ้ำซ้อนกัน* ในกรณีที่ลูกค้านำตั๋วสัญญาใช้เงินที่ครบกำหนด มาทำการฝากต่อ (renew) พนักงานจะต้องทำการปิดบัญชีก่อนแล้วจึงทำการฝากตั๋วสัญญาใช้เงินใหม่ ซึ่งเป็นการทำงานที่ซ้ำซ้อนกัน

3.4.2 Requirement Analysis

เป็นขั้นตอนที่ต้องวิเคราะห์ ความต้องการของผู้ใช้ ในการแก้ปัญหาที่เกิดขึ้นกับผู้ใช้ ซึ่งสามารถสรุปได้ดังนี้

- ผู้ใช้ต้องการเปลี่ยนระบบการทำงานจากการทำด้วยมือไปเป็นระบบคอมพิวเตอร์เพื่อความถูกต้องและรวดเร็ว
- ผู้ใช้ต้องการทำ การฝากต่อตั๋วเดิมที่ทำการถอน (renew) เมื่อครบกำหนดโดยอัตโนมัติ
- ต้องการทำระบบ Maker-Checker เพื่อป้องกันการทุจริต
- ผู้ใช้ต้องการให้มีการทำงานแบบ On-line แต่มีขอบเขตอยู่ภายใน office เดียวกัน
- ผู้ใช้เห็นว่าแนวโน้มของการทำธุรกรรมนี้จะมีลูกค้ามากขึ้น จึงต้องการระบบที่มีความรวดเร็วในการทำงาน ไม่ว่าจะเป็นการ ค้นหาข้อมูล,การ สอบถามข้อมูล และ การพิมพ์รายงาน

3.4.3 Specification

เป็นขั้นตอนที่กำหนดว่าระบบทำอะไรได้บ้างเพื่อให้ได้ผลตามสิ่งที่ผู้ใช้ต้องการ ซึ่งสรุปได้ดังนี้

- สามารถควบคุมและตรวจสอบข้อมูลด้วยระบบ Maker-Checker
- สามารถพิมพ์ตั๋วสัญญาใช้เงิน, เช็ค, ใบ Inward Voucher, ใบ Outward Voucher ได้
- สามารถทำการฝากตั๋วสัญญาใช้เงินใหม่จากตั๋วสัญญาใช้เงินที่ครบอายุเดิมโดยอัตโนมัติได้

- เนื่องจากในอนาคต อาจมีการเปลี่ยนแปลงการทำงานของโปรแกรมให้เหมาะสมกับสถานะการณ์ ดังนั้นโปรแกรมจะออกแบบมาให้มีความยืดหยุ่นพอกับการแก้ไขเปลี่ยนแปลงได้ง่าย และใช้ระยะเวลาสั้น
- มีการทำงานแบบ Client/Server

3.4.4 Design

ออกแบบให้ระบบให้สามารถทำงานได้ตาม specification ในขั้นตอนนี้เราได้เลือกจึงกำหนดแนวทางการออกแบบได้ดังนี้

- ออกแบบระบบซอฟต์แวร์ด้วยแนวคิดเชิงวัตถุ โดยใช้ UML เข้ามาช่วยในการพัฒนา
- ใช้เครื่องมือในการออกแบบ UML คือ Rational Rose 2000
- ใช้เครื่องมือในเขียนโปรแกรม Centura 1.5.1
- ใช้ระบบฐานข้อมูลแบบ Relational Database โดยใช้ Microsoft Access 97

สำหรับไดอะแกรมใน UML ที่เหมาะสมสำหรับโครงการนี้มีดังต่อไปนี้

- Use Case Diagram
- Activity Diagram (activity-Oriented)
- Sequence Diagram (message-oriented)
- Classes Diagram
- Statechart Diagram (time-oriented)

โดยที่ได้ตัดเอาไดอะแกรมบางส่วนออกคือ

Deployment Diagram ซึ่งจะแสดงความสัมพันธ์กันระหว่างซอฟต์แวร์และฮาร์ดแวร์ในทางกายภาพ และเหมาะสำหรับแสดงเส้นทางของ object ในการทำงานในระบบ distributed เท่านั้น [6]

Component Diagram ซึ่งเอาไว้ใช้สำหรับแสดงความขึ้นต่อกัน (dependency) ของ component ต่าง ๆ ซึ่งเราเห็นว่าการที่จะใช้ไดอะแกรมนี้ควรจะเป็นระบบที่มีความซับซ้อนและมีระบบย่อยที่มีจำนวนมาก [6]

เริ่มด้วยการทำ Use case diagram ก่อน ซึ่งจะเป็นการแสดงความสัมพันธ์ของระบบกับผู้ใช้ และแสดงให้เห็นถึง function ของระบบด้วยว่ามีอะไรบ้าง โดยจะต้องกำหนด Actor และ Use case

Use Case Diagram

เป็นการแสดงหน้าที่ของระบบในมุมมองของผู้ใช้งานเพื่อให้ได้ผลตามที่ผู้ใช้ต้องการ โดยเราได้กำหนด Actor และ Use case ในระบบได้ดังนี้

Actor คือบทบาทของผู้ใช้ที่มีให้กับระบบ Actor สามารถเป็นได้ทั้ง คน, อุปกรณ์ฮาร์ดแวร์ หรือ ระบบอื่น ที่ทำงานอยู่กับระบบของเรา ซึ่งจะอธิบายได้ดังนี้

คำอธิบาย Customer : บุคคลที่มาทำธุรกรรมกับธนาคาร เช่น การเปิดบัญชี, การฝากตั๋วสัญญาใช้เงิน,การถอนตั๋วสัญญาใช้เงิน

ซึ่งต่อไปนี้เป็น Actor ในระบบพร้อมคำอธิบาย

Maker: เจ้าหน้าที่ของธนาคาร มีหน้าที่ติดต่อกับลูกค้าและบันทึกการทำธุรกรรม, การแก้ไข การลบข้อมูลของลูกค้ากับระบบตั๋วสัญญาใช้เงิน

Checker: เจ้าหน้าที่ของธนาคารที่ทำหน้าที่ตรวจสอบและอนุมัติข้อมูลที่บันทึกจาก Maker เพื่อให้เป็นข้อมูลที่ระบบตั๋วสัญญาใช้เงินสามารถนำไปใช้ได้ และต้องไม่ใช้คนคนเดียวกันกับ Maker

GL: เป็นระบบบัญชีที่จะคอยรับเอกสารทางการเงินจากระบบตั๋วสัญญาใช้เงิน เพื่อนำไปทำงานในระบบบัญชีของธนาคาร

Use Case

เป็นการแสดง function ของระบบที่ actor สามารถใช้งานได้

คำอธิบาย : use case นี้มีไว้เพื่อให้ Maker บันทึกข้อมูลการเปิดบัญชีของ Customer

ซึ่งต่อไปนี้เป็น use case ในระบบพร้อมคำอธิบาย

Open Account Approval: use case นี้มีไว้เพื่อให้ Checker ตรวจสอบข้อมูลการเปิดบัญชีของ Customer ที่ทำโดย Maker

Deposit P/N Entry: use case นี้มีไว้เพื่อให้ Maker บันทึกข้อมูลการฝาก P/N ของ Customer

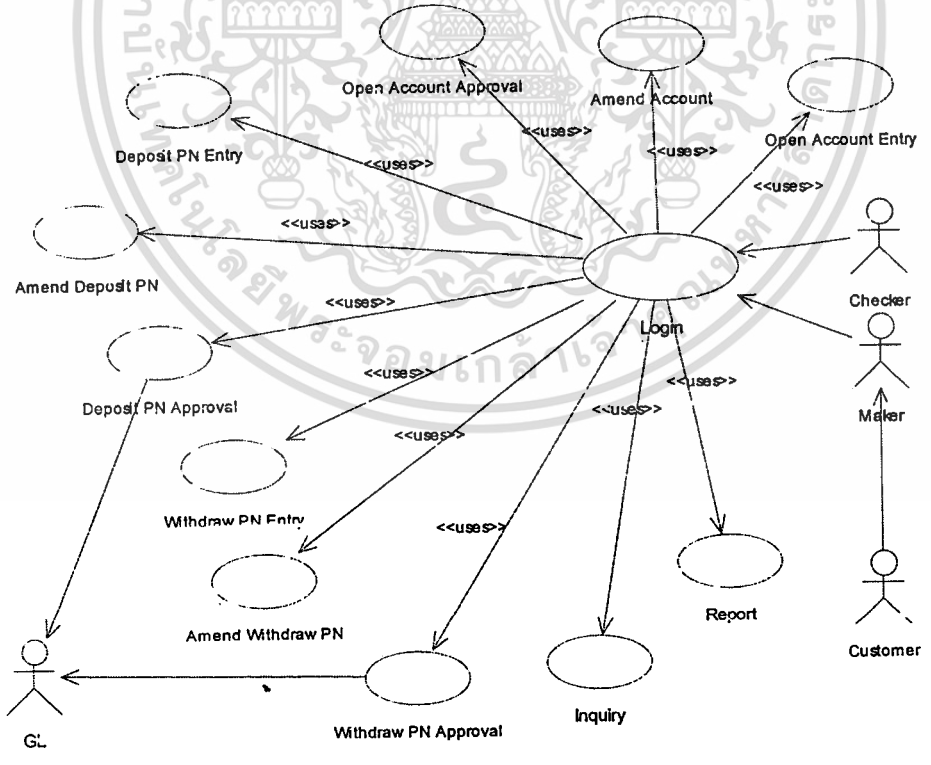
Deposit P/N Approval: use case นี้มีไว้เพื่อให้ Checker ตรวจสอบข้อมูลการฝาก P/N ของ Customer ที่ทำโดย Maker

Withdraw P/N Entry: use case นี้มีไว้เพื่อให้ Makeบันทึกข้อมูลการถอนเงินของ Customer

Withdraw P/N approval: use case นี้มีไว้เพื่อให้ Checker ตรวจสอบข้อมูลการถอนของ Customer ที่ทำโดย Maker

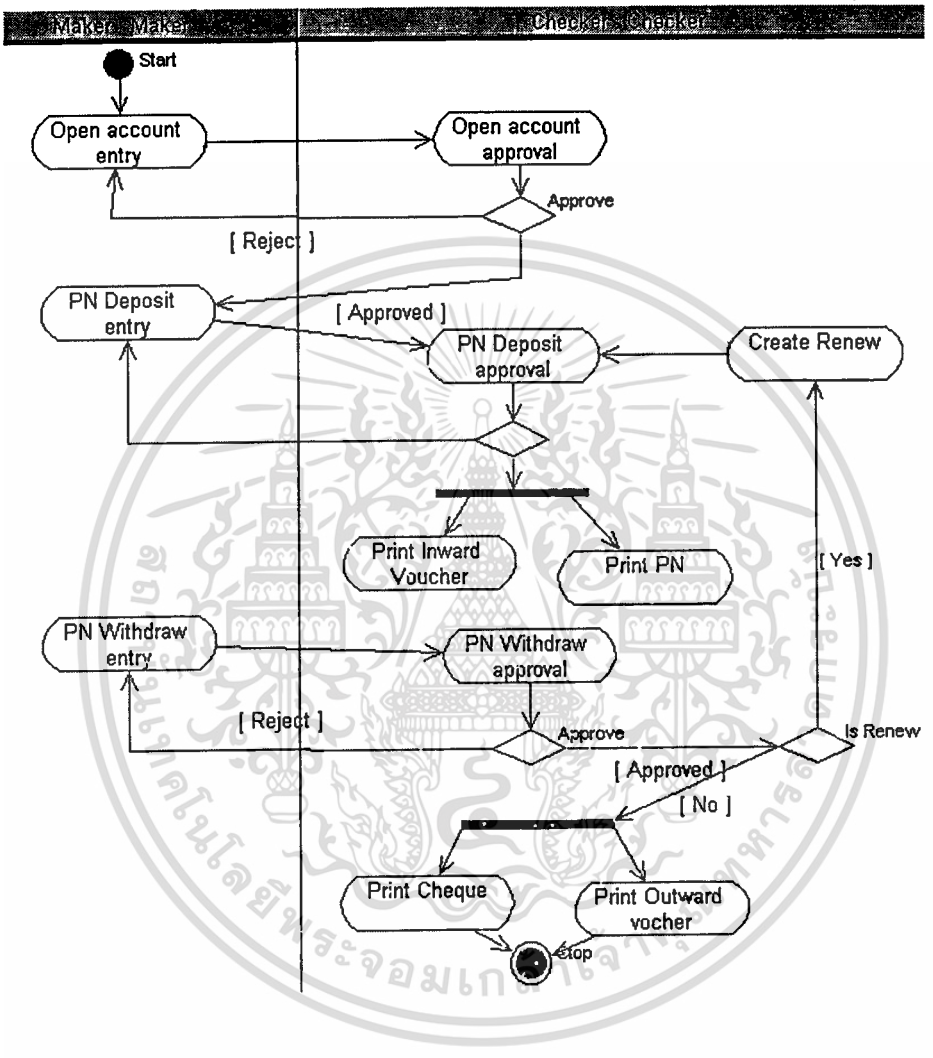
Inquiry: use case นี้มีไว้เพื่อให้ Checker และ Maker เรียก ข้อมูลในระบบได้ เช่น การเรียกดูข้อมูลตั๋วสัญญาใช้เงิน

Report: use case นี้มีไว้เพื่อให้ Checker และ Maker ออกรายงานต่าง ๆ เพื่อสามารถนำไปเป็น information ในการบริหารเงินได้



รูปที่ 3.1 Use Case Diagram ของระบบตั๋วสัญญาใช้เงิน

หลังจากที่เราได้กำหนด use case ต่าง ๆ ที่จำเป็นต่อระบบแล้ว เราจะอธิบายการทำงานของแต่ละ use case ว่ามีรายละเอียดอย่างไรบ้าง เช่น มี actor อะไรที่เกี่ยวข้อง, มีลำดับของเหตุการณ์อย่างไร พร้อมกับแสดง activity diagram ของ use case นั้นด้วย



รูปที่ 3.2 Activity Diagram แสดง System overview ของระบบ

ภาพที่ 3.2 แสดงภาพการทำงานรวมของระบบ และต่อไปนี้จะแสดงการทำงานของแต่ละ use case

LOGIN

Use case นี้อธิบายกระบวนการทำงาน โดยที่ผู้ใช้ระบบ Log in เข้ามาในระบบ P/N พร้อมทั้งทำการกำหนดสิทธิในการใช้งานต่าง ๆ ตามสิทธิของผู้ใช้

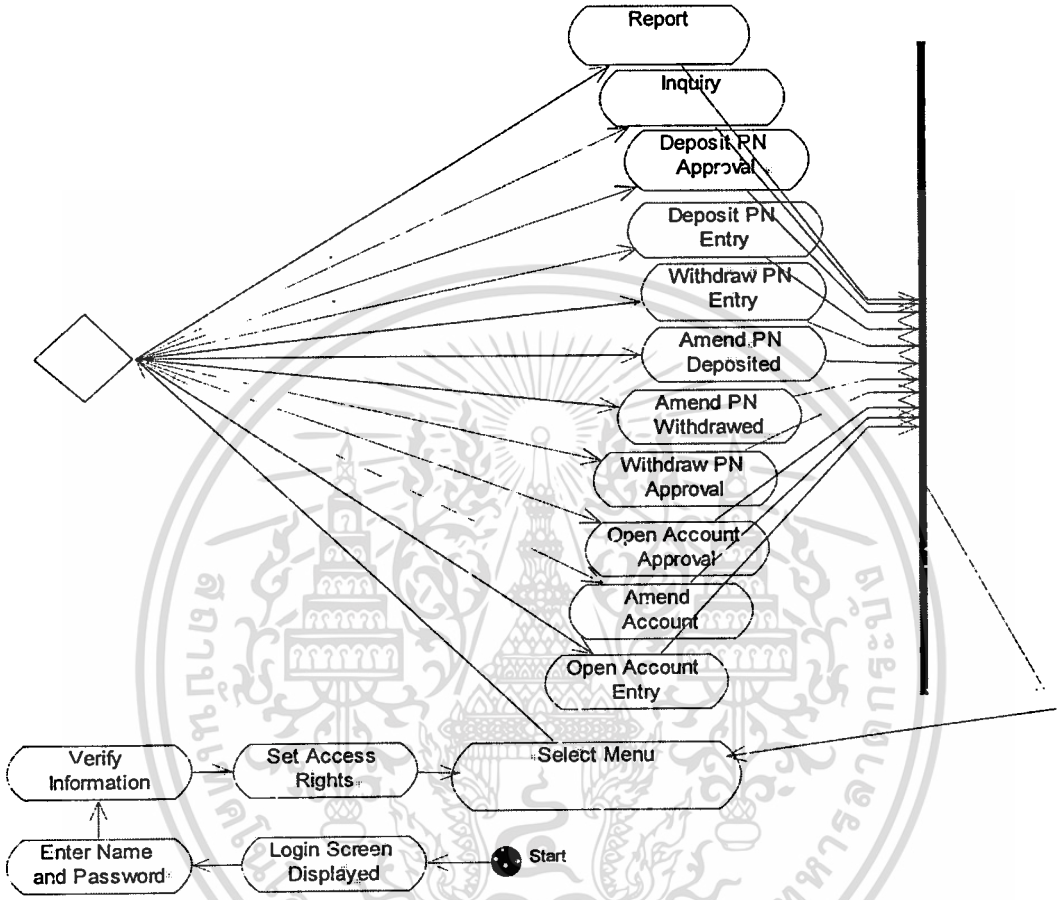
Actors

- Maker
- Checker

ลำดับเหตุการณ์

1. Use case เริ่มทำงานเมื่อผู้ใช้เปิดโปรแกรม
2. ระบบจะแสดงหน้าจอสำหรับกรอกชื่อผู้ใช้ และ รหัสผ่าน
3. ผู้ใช้กรอกชื่อ และ รหัสผ่าน
4. ระบบจะกำหนดสิทธิของผู้ใช้
5. ระบบจะแสดงหน้าจอหลัก
6. ผู้ใช้จะเลือกเมนูต่าง ๆ
7. ขณะที่ผู้ใช้ยังไม่จบการทำงาน ผู้ใช้สามารถเลือกใช้เมนูต่าง ๆ ได้

Activity Diagram



รูปที่ 3.3 Login

OPEN ACCOUNT ENTRY

Use case นี้อธิบายกระบวนการเปิดบัญชีของ Customer กับ Maker ซึ่งจะเป็นกระบวนการแรกในการทำงานในกรณีที่ Customer ยังไม่ได้เปิดบัญชีกับธนาคาร

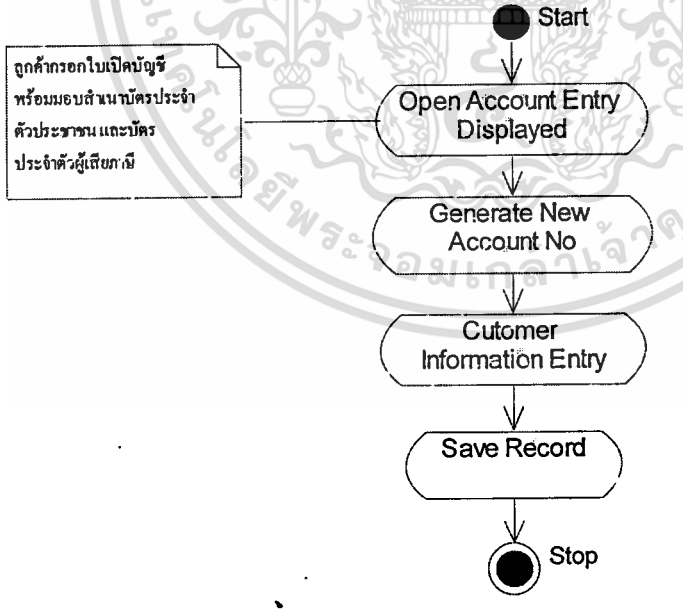
Actors

- Customer
- Maker

ลำดับเหตุการณ์

1. จะเริ่มการทำงานเมื่อมีคำสั่งเปิดบัญชีจากลูกค้า
2. Maker จะเปิดหน้าจอการเปิดบัญชี
3. Maker จะกรอกรายละเอียดของลูกค้า
4. Maker กดปุ่ม Save เมื่อกรอกรายละเอียดครบถ้วน
5. ระบบจะสร้างหมายเลขบัญชีใหม่อัตโนมัติ
6. ระบบจะบันทึกข้อมูล เพื่อรอการตรวจจาก Checker

Activity Diagram



รูปที่ 3.4 Open Account Entry

OPEN ACCOUNT APPROVAL

use case นี้มีไว้เพื่อให้ Checker ตรวจสอบข้อมูลการเปิดบัญชีของ Customer ที่ทำโดย Maker

Actors

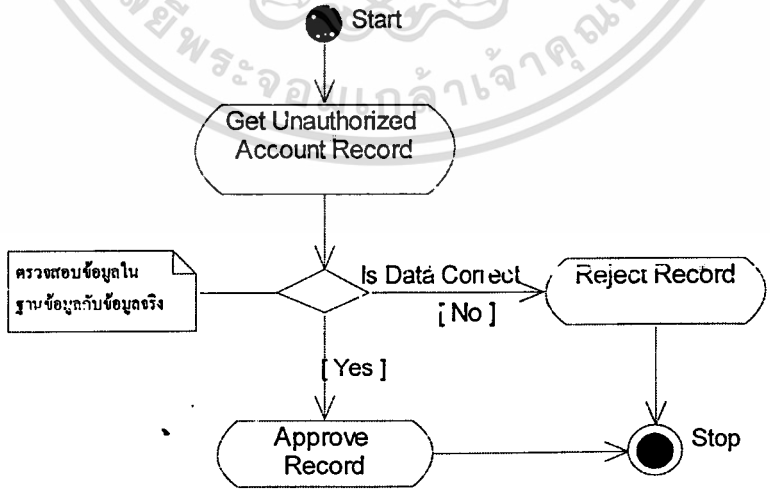
- Maker

ลำดับเหตุการณ์

1. use case จะเริ่มเมื่อ Maker คลิ๊กที่เมนู Open Account Approval
2. ระบบจะอ่านข้อมูลของ Account ที่มีสถานะเป็น UN
3. Checker จะเลือกข้อมูล Account ที่ต้องการตรวจสอบ
4. ระบบแสดงข้อมูลของ Account
5. Checker จะตรวจสอบข้อมูลบนหน้าจอ กับข้อมูลตามใบขอเปิดบัญชี
6. ถ้าข้อมูลถูกต้อง

   Checker กดปุ่ม Approve ระบบจะเปลี่ยนสถานะบัญชีเป็น LI
7. ถ้าไม่ถูกต้อง Checker กดปุ่ม Reject ระบบจะเปลี่ยนสถานะบัญชีเป็น RN

Activity Diagram



รูปที่ 3.5 Open Account Approval

DEPOSIT P/N ENTRY

use case นี้มีไว้เพื่อให้ Maker บันทึกข้อมูลการฝาก P/N ของ Customer

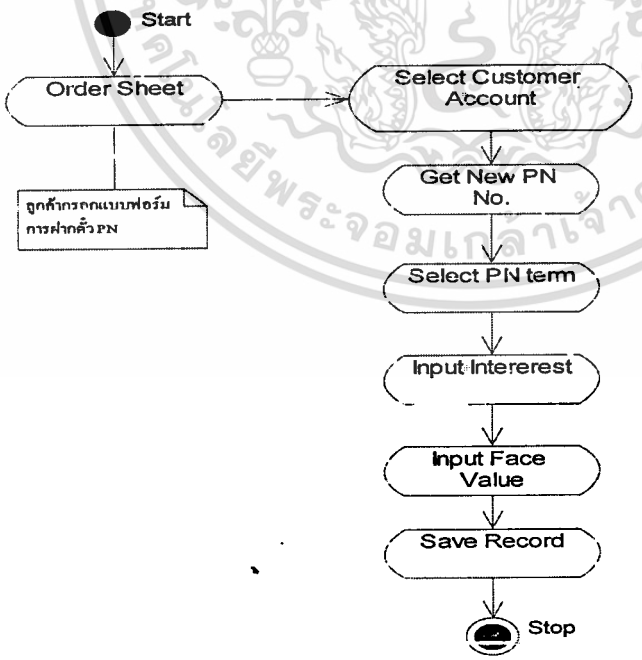
Actors

- Customer
- Maker

ลำดับเหตุการณ์

1. use case นี้จะเริ่มเมื่อลูกค้ากรอกแบบฟอร์มการฝาก P/N และมอบให้ Maker
2. Maker เปิดหน้าจอการฝาก P/N
3. Maker กรอกข้อมูลตามแบบฟอร์มการฝาก P/N เช่น ระยะเวลาการฝาก, อัตราดอกเบี้ย, จำนวนเงิน เป็นต้น
4. ระบบจะสร้างหมายเลข P/N ให้อัตโนมัติ
5. Make บันทึกข้อมูล

Activity Diagram



รูปที่ 3.6 Deposit P/N Entry

DEPOSIT P/N APPROVAL

use case นี้มีไว้เพื่อให้ Checker ตรวจสอบข้อมูลการฝากเงินของ Customer ที่ทำโดย Maker

Actor

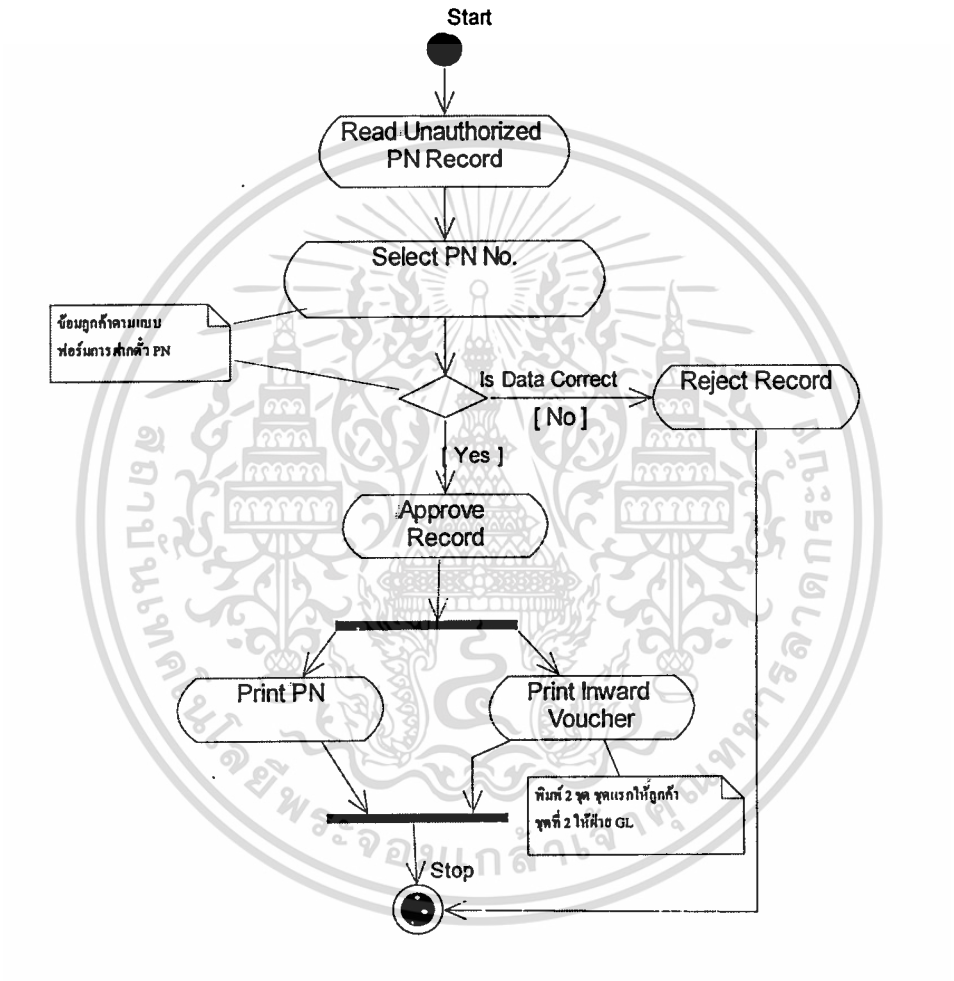
- Maker
- GL

ลำดับเหตุการณ์

1. Use case นี้เริ่มเมื่อ checker ต้องการ approve ข้อมูล P/N ที่ได้รายการโดย Maker
2. ระบบอ่านข้อมูล P/N ที่มีสถานะเป็น U
3. Maker เลือกข้อมูล P/N ที่ต้องการ
4. ระบบแสดงข้อมูล P/N
5. Maker ตรวจสอบข้อมูลจากระบบ กับแบบฟอร์มการฝาก P/N ของลูกค้า
6. ถ้าข้อมูลถูกต้อง

   Maker กดปุ่ม Approve บนหน้าจอ แล้วระบบ จะพิมพ์ตั๋วสัญญาใช้เงินออกทางเครื่องพิมพ์ พร้อมใบ Inward Voucher จำนวน 2 ชุด เพื่อให้ลูกค้า 1 ชุด และให้ระบบบัญชีเก็บไว้ลง GL อีก 1 ชุด
7. ถ้าไม่ถูกต้อง Make กดปุ่ม Reject ระบบจะเปลี่ยนสถานะ P/N เป็น Reject

Activity Diagram



รูปที่ 3.7 Deposit P/N Approval

WITHDRAW P/N ENTRY

use case นี้มีไว้เพื่อให้ Maker บันทึกข้อมูลการถอน P/N ของ Customer เมื่อ P/N ครบกำหนดการไถ่ถอน

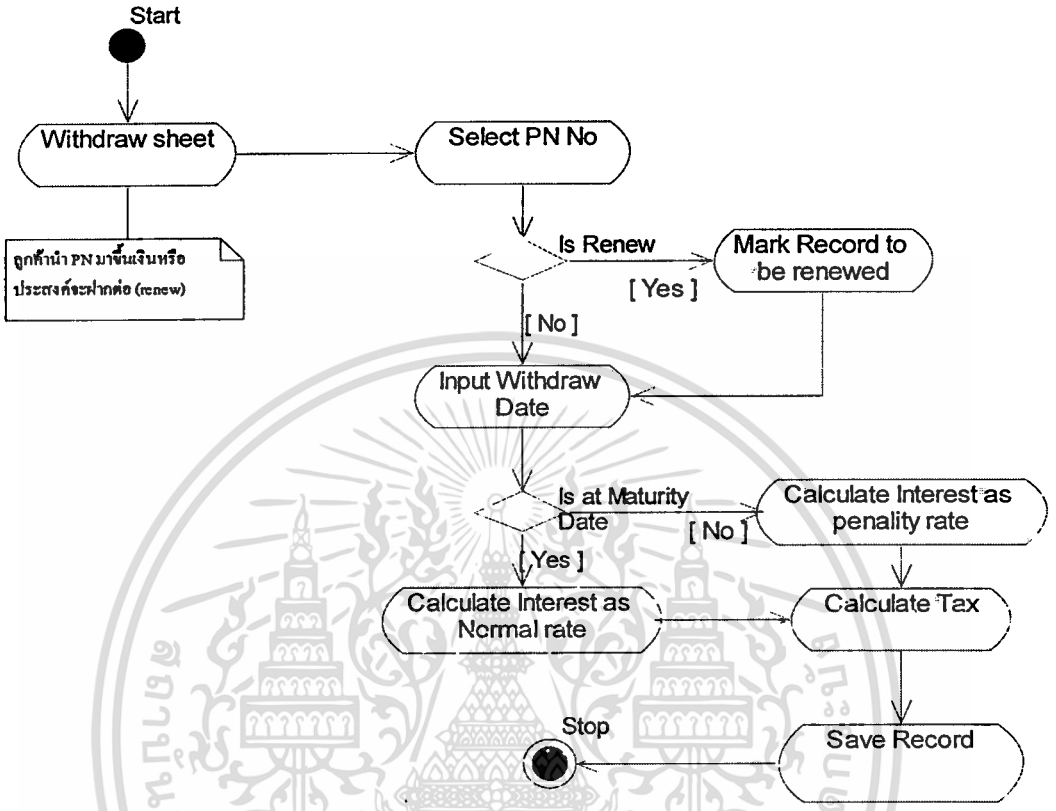
Actors

- Maker
- Customer

ลำดับเหตุการณ์

1. use case นี้เริ่มหลังจากที่ Customer ได้นำ P/N ที่ครบกำหนดมาไถ่ถอนและได้กรอกแบบฟอร์มการไถ่ถอน และนำมาให้ Maker เป็นผู้ทำรายการ
2. Maker เลือกรายการ P/N จากระบบ
3. Maker กรอกข้อมูลการไถ่ถอนของ P/N คือ วันที่ไถ่ถอนของ P/N
4. ระบบจะตรวจสอบว่า P/N ได้ถอนก่อนกำหนดหรือไม่
5. ถ้าเป็นการไถ่ถอนก่อนกำหนด
   ระบบจะคำนวณอัตราดอกเบี้ยให้น้อยกว่าที่ได้กำหนดไว้
6. ถ้าเป็นการไถ่ถอนตามกำหนด ระบบจะคำนวนดอกเบี้ยตามอัตราที่ได้กำหนดไว้
7. ระบบคำนวณเงินต้นพร้อมดอกเบี้ยที่ได้รับ พร้อมหักภาษี ณ ที่จ่าย
8. Maker กดปุ่ม Save บนหน้าจอ

Activity Diagram



รูปที่ 3.8 Withdraw P/N Entry

WITHDRAW P/N APPROVAL

Use case นี้มีไว้เพื่อให้ Checker ตรวจสอบข้อมูลการถอนของ Customer ที่ทำโดย Maker

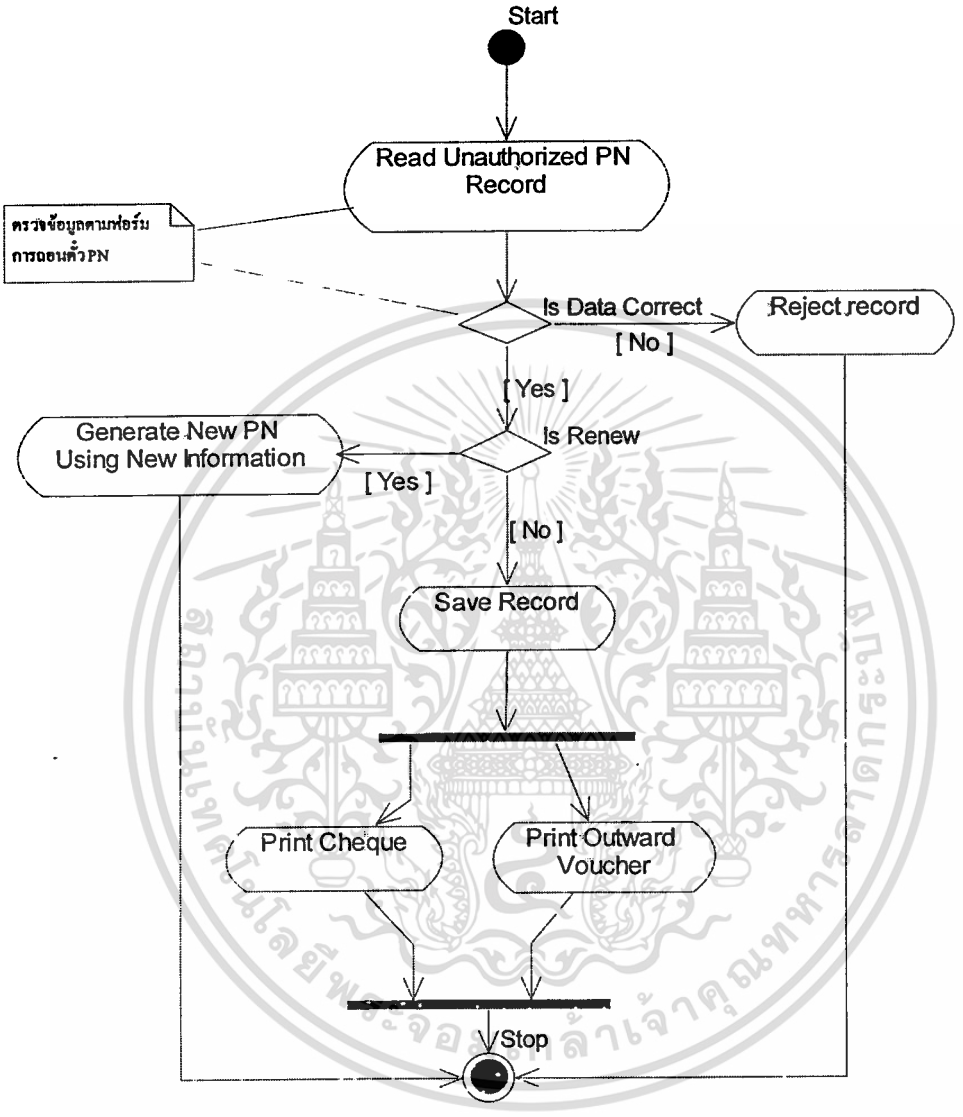
Actor

- Checker
- GL
- Customer

ลำดับเหตุการณ์

1. use case นี้เริ่มเมื่อ Checker ต้องการ Approve ข้อมูลการถอน P/N ที่ได้ทำรายการจาก Maker
2. ระบบอ่านข้อมูล P/N ที่มีสถานะเป็น U
3. Checker เลือกข้อมูล P/N ที่ต้องการตรวจสอบการไถ่ถอน
4. ระบบแสดงข้อมูล P/N
5. Checker ตรวจสอบข้อมูลต่างๆ เช่น ชื่อบัญชี จำนวนเงินต้นและดอกเบี้ย
6. ถ้าข้อมูลการไถ่ถอนถูกต้อง
   Checker กดปุ่ม Save บนหน้าจอ แล้วระบบจะพิมพ์ Cheque ให้ Customer และใบ Outward Voucher จำนวน 2 ชุด โดยให้ Customer 1 ชุด และฝ่ายบัญชีเพื่อนำไปลงรายการ GL อีก 1 ชุด
7. ถ้าข้อมูลไม่ถูกต้อง Checker กดปุ่ม Reject บนหน้าจอ

Activity Diagram



รูปที่ 3.9 Withdraw P/N Approval

AMEND ACCOUNT

Use case นี้มีไว้สำหรับ Maker เมื่อต้องการแก้ไขข้อมูล Account

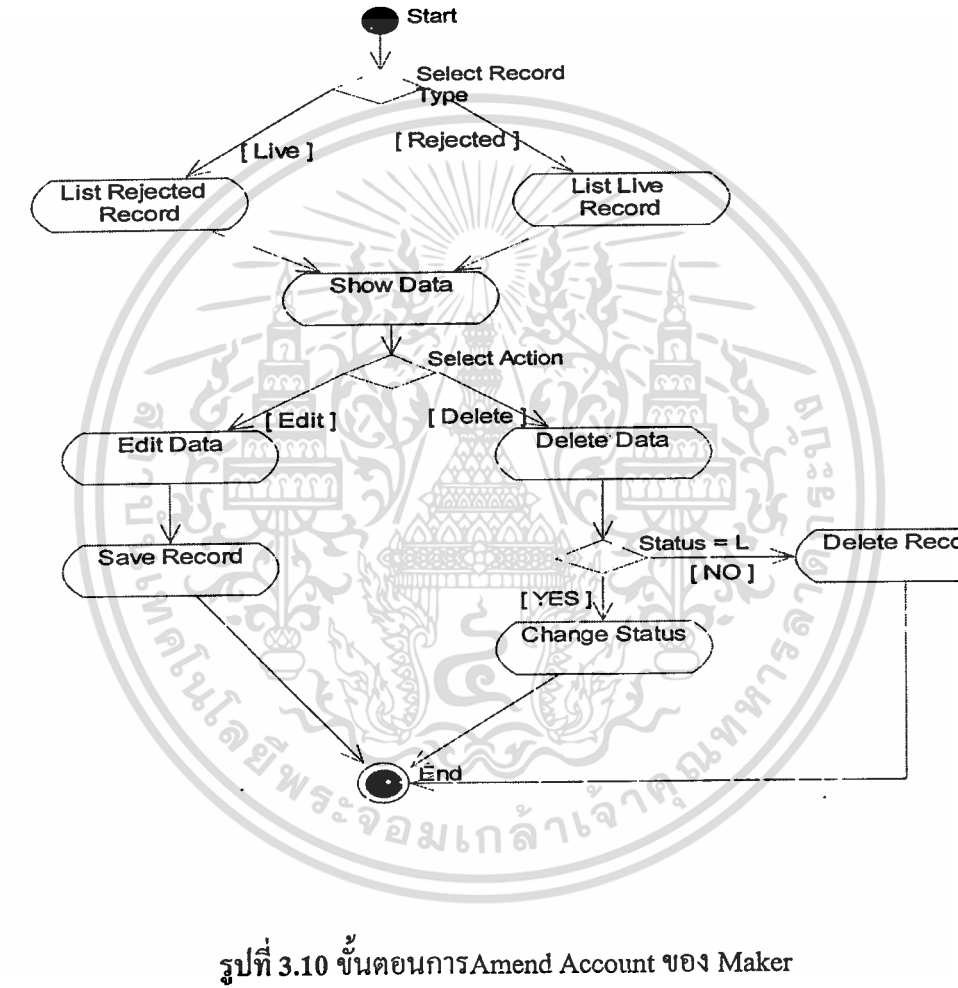
Actor

- Maker

ลำดับเหตุการณ์

1. use case นี้เริ่มเมื่อ Maker ต้องการแก้ไขข้อมูล Account
2. ระบบจะให้เลือกว่าต้องการแสดงรายการที่มีสถานะ Live หรือ Unauthorized
3. ระบบจะแสดงข้อมูลตามเงื่อนไขข้อ 2
4. Maker ตรวจสอบข้อมูล
5. ถ้า Maker ต้องการแก้ไขข้อมูล
   Maker แก้ไขข้อมูลตามที่ต้องการแล้วบันทึกข้อมูล
6. ถ้า Maker ต้องการลบข้อมูล
   Maker กดปุ่ม Delete บนหน้าจอ ระบบจะทำการลบข้อมูล
7. การ Delete ถ้าข้อมูลสถานะ L
   ระบบจะเปลี่ยนสถานะให้เป็น LD เพื่อรอการ approve ต่อไป
8. ถ้าสถานะข้อมูลเป็น R
   ระบบจะลบข้อมูลฐานข้อมูลอย่างถาวร

Activity Diagram



รูปที่ 3.10 ขั้นตอนการAmend Account ของ Maker

AMEND DEPOSIT P/N

Use case นี้มีไว้สำหรับ Maker เมื่อต้องการแก้ไขข้อมูลการฝาก P/N ที่ Checker ได้ตรวจสอบแล้วว่ามีข้อมูลการฝากไม่ถูกต้อง

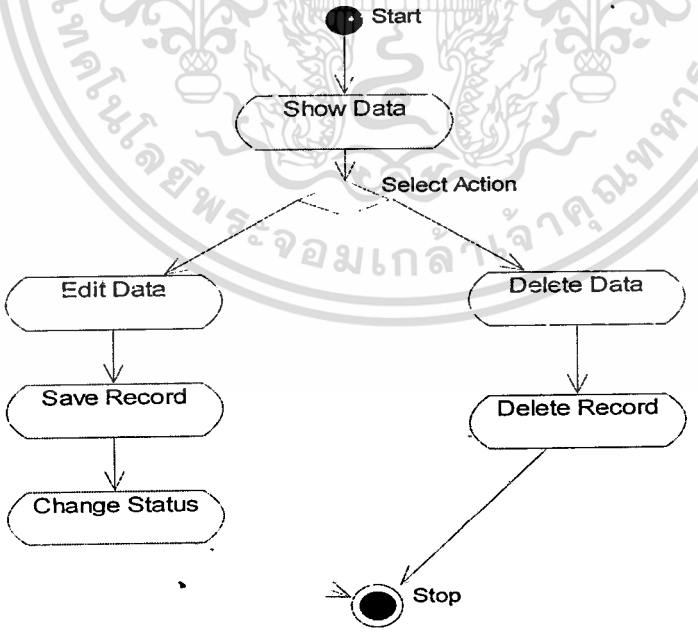
Actor

- Maker

ลำดับเหตุการณ์

1. use case นี้เริ่มเมื่อ Maker ต้องการแก้ไขข้อมูลการฝาก P/N
2. ระบบแสดงข้อมูล P/N ที่มีสถานะเป็น RN หรือ LI
3. Maker เลือกข้อมูล และ ตรวจสอบข้อมูล
4. ถ้า Maker ต้องการแก้ไขข้อมูล
   Maker แก้ไขข้อมูลตามที่ต้องการแล้วบันทึกข้อมูล
5. ถ้า Maker ต้องการลบข้อมูล Maker กดปุ่ม Delete บนหน้าจอ ระบบจะทำการลบข้อมูล ระบบจะลบข้อมูลฐานข้อมูลอย่างถาวร

Activity Diagram



รูปที่ 3.11 ขั้นตอนการAmend Deposit ของ Maker

AMEND WITHDRAW P/N

Use case นี้มีไว้สำหรับ Maker เมื่อต้องการแก้ไขข้อมูลการถอน P/N ที่ Checker ได้ตรวจสอบแล้วว่ามีข้อมูลการถอนไม่ถูกต้อง

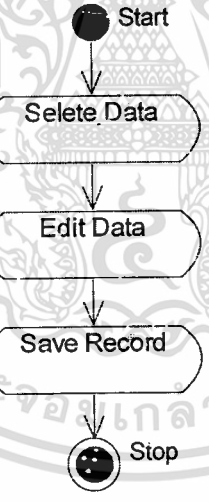
Actor

- Maker

ลำดับเหตุการณ์

1. use case นี้เริ่มเมื่อ Maker ต้องการแก้ไขข้อมูลการถอน P/N
2. ระบบแสดงข้อมูล P/N ที่มีสถานะเป็น R
3. Maker เลือกข้อมูล และ ตรวจสอบข้อมูล
4. Maker แก้ไขข้อมูลตามที่ต้องการแล้วบันทึกข้อมูล

Activity Diagram



รูปที่ 3.12 Amend Withdraw P/N

INQUIRY

Use case นี้ใช้เมื่อต้องการค้นหาข้อมูลของ P/N ที่มีอยู่ในระบบ

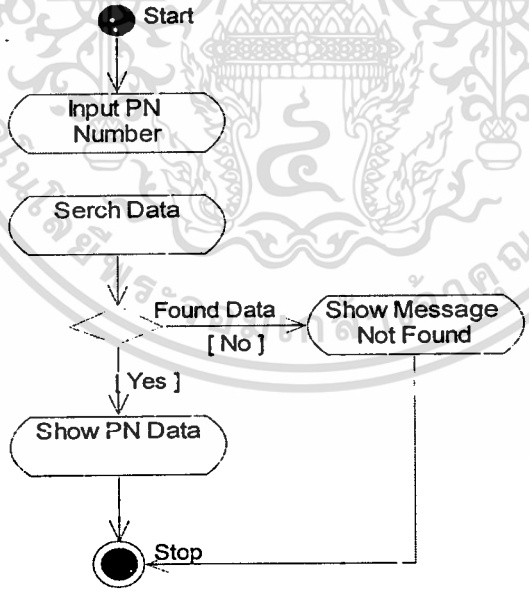
Actor

- **Maker**
- **Checker**

ลำดับเหตุการณ์

1. **Use case นี้เริ่มเมื่อผู้ใช้ต้องการค้นหาข้อมูลของ P/N**
2. **ระบบจะให้ผู้ใช้ป้อนหมายเลขของ P/N**
3. **ระบบจะแสดงข้อมูลของ P/N**
4. **หากไม่มีข้อมูลของ P/N ระบบจะแสดงข้อความให้ผู้ใช้ทราบว่าไม่พบข้อมูล**

Activity Diagram



รูปที่ 3.13 ขั้นตอนใน Inquiry

REPORT

Use case นี้ใช้เมื่อต้องการพิมพ์รายงานต่าง ๆ ในระบบ P/N ซึ่งรายงานต่าง ๆ จะประกอบไปด้วย

1. รายงานยอดคงเหลือสุทธิ
2. รายงาน P/N จะครบกำหนด
3. รายงาน P/N ตามหมายเลขบัญชี

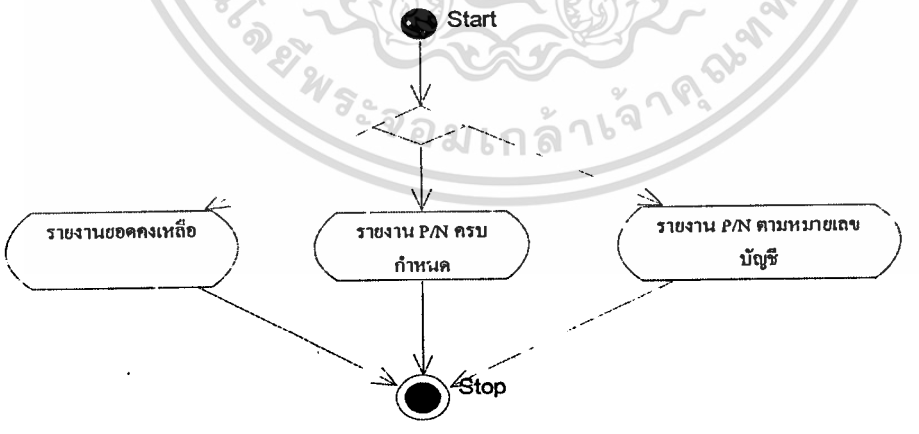
Actor

- Maker
- Checker

ลำดับเหตุการณ์

1. Use case นี้เริ่มเมื่อผู้ใช้ต้องการพิมพ์รายงานต่าง ๆ ของระบบ P/N
2. ระบบจะให้ผู้ใช้เลือกชนิดของรายงานต่าง ๆ
3. ระบบจะแสดงข้อมูลตามชนิดของรายงาน

Activity Diagram



รูปที่ 3.14 การพิมพ์รายงานในระบบ P/N

Class Diagram

Class คือ กลุ่มของ Object ที่มี Attribute เดียวกัน, มี Operation ชุดเดียวกัน และความสัมพันธ์และความหมายแบบเดียวกัน [1] นอกจากนั้นเรายังแบ่งประเภทของ class ด้วย Stereotype ซึ่ง Stereotype of class คือ ประเภทต่าง ๆ ของ class ที่แสดงเพิ่มเติมจาก class เดิมเพื่อประโยชน์ในการแบ่งกลุ่มเพื่อนำไปใช้งานได้ง่าย [5] เราจึงแบ่ง stereotype of class ได้ดังนี้

- Entity Classes

  เป็นแบบจำลองจาก real-world entity หรือเป็นสิ่งที่มีความจำเป็นต่อการทำงานของระบบ โดยทั่วไปจะเป็นอิสระต่อสิ่งที่อยู่รอบข้าง [5] ซึ่งต่อไปนี้เป็น entity class ในระบบพร้อมคำอธิบาย

  - **PN**: เป็นข้อมูลที่จำเป็นต่อ customer ในการฝากเงิน, ถอนเงิน และการคิดดอกเบี้ย รวมไปถึงการพิมพ์ใบ Inward voucher, และ Outword voucher ด้วย
  - **Account**: เป็นข้อมูลที่จำเป็นต่อ customer ในฝาก P/N, ถอน P/N และการคิดดอกเบี้ย
  - **Customer**: เป็นข้อมูลที่จำเป็นต่อการเปิดบัญชี, ปิดบัญชี เป็นบุคคลที่ฝากเงิน, ถอนเงิน จากระบบ P/N
  - **Maker**: เป็นข้อมูลที่จำเป็นต่อการ entry ข้อมูลทุกอย่างในระบบ PN
  - **Checker**: เป็นข้อมูลที่จำเป็นต่อการ approve ข้อมูลทุกอย่างในระบบ PN

- **Boundary Classes**

  มีหน้าที่ในการติดต่อสื่อสารระหว่างสิ่งรอบข้างกับการทำงานในระบบ [5] ซึ่งต่อไปนี้เป็น boundary class ในระบบพร้อมคำอธิบาย

  - **MakerEntryAccount**:มีหน้าที่ให้ Maker บันทึกข้อมูลการเปิดบัญชีของ customer
  - **CheckerApproveAcount**: มีหน้าที่ให้ Checker approve ข้อมูลการเปิดบัญชีของ customer
  - **MakerEntryDeposit**: มีหน้าที่ให้ Maker บันทึกข้อมูลการฝาก PN ของ customer
  - **CheckerApproveDeposit**: : มีหน้าที่ให้ Checker approve ข้อมูลการฝาก PN ของ customer
  - **MakerEntryWithdraw**: มีหน้าที่ให้ Maker บันทึกข้อมูลการถอน PN ของ customer

- **CheckerApproveWithdraw:** : มีหน้าที่ให้ Checker approve ข้อมูลการถอน PN ของ customer

- **Control Classes**

  ทำหน้าที่ในการควบคุมลำดับการทำงานของ use case ซึ่งอาจจะควบคุม use case ได้มากกว่า 1 use case ก็ได้ และยังทำหน้าที่จัดการเกี่ยวกับเหตุการณ์ต่าง ๆ ที่เราได้กำหนดไว้ใน use case โดยทั่วไปแล้วจะเป็นลักษณะ application-dependent class [5] ซึ่งต่อไปนี้เป็น control class ในระบบพร้อมคำอธิบาย

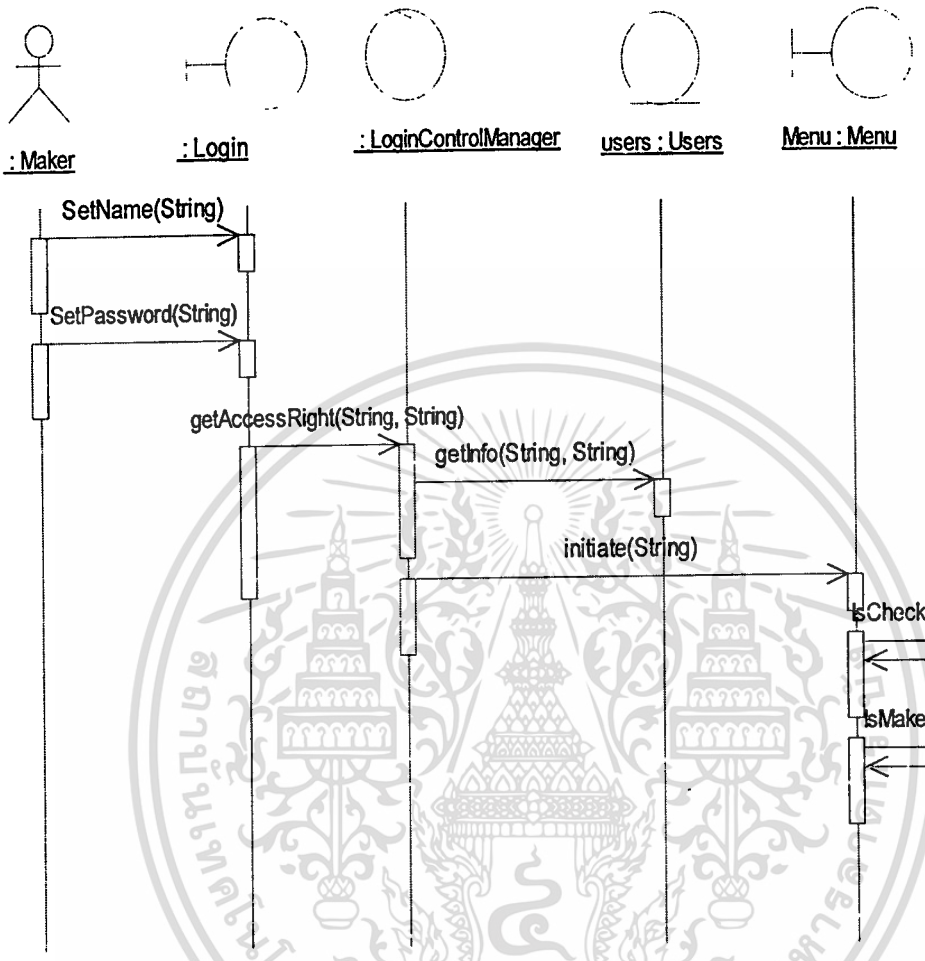
  - **LoginControlManager:** ทำหน้าที่ควบคุมการ Log in และการกำหนดสิทธิในการทำงานต่าง ๆ

หลังจากที่เราได้กำหนดคลาสต่าง ๆ ที่มีอยู่ในระบบแล้ว เราจะแสดงการทำงานระหว่างออปเจคตามลำดับเหตุการณ์ที่เกิดขึ้น เราจะใช้ Sequence Diagram เป็นสิ่งที่ใช้แสดงดังกล่าว

Sequence diagram

แสดงปฏิสัมพันธ์ระหว่างออปเจคตามลำดับของเหตุการณ์ที่เกิดขึ้น [6] ที่มี Message ต่าง ๆ ส่งไปมา จากแต่ละ use case เราสามารถ แสดงเหตุการณ์ที่เกิดขึ้น และความสัมพันธ์ ของออปเจค ที่เราได้ออกแบบมาในรูปแบบของลำดับเวลา ซึ่งจะเรียงตามลำดับของแต่ละ use case ได้ดังนี้

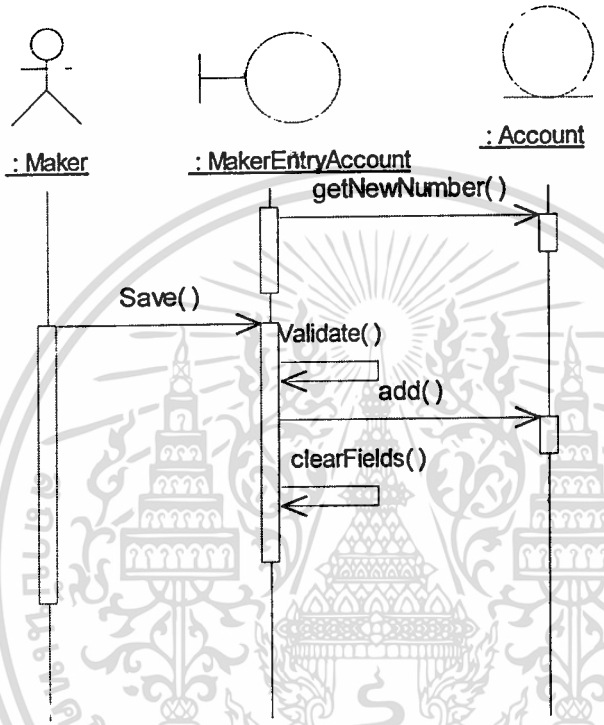
LOGIN



รูปที่ 3.15 Sequence diagram สำหรับการ Login

Use case เริ่มทำงานเมื่อผู้ใช้เปิดโปรแกรม ระบบจะแสดงหน้าจอสำหรับกรอกชื่อผู้ใช้ และรหัสผ่าน โดยใช้ออปเจค Log in แล้วออปเจค Log in จะเรียกใช้เมททอด getAccessRight โดยใช้ชื่อและรหัสผ่าน เป็นอาร์กิวเมนของออปเจค LoginControlManager แล้ว ออปเจคจะเรียกเมททอด getInfo จากออปเจค Users เพื่อตรวจสอบสิทธิของผู้ใช้ หากผู้ใช้ในพาสเวิร์ดถูกต้องระบบจะกำหนดสิทธิของผู้ใช้ และจะแสดงเมนูตามสิทธิของผู้ใช้ให้เลือกเมนูต่าง ๆ

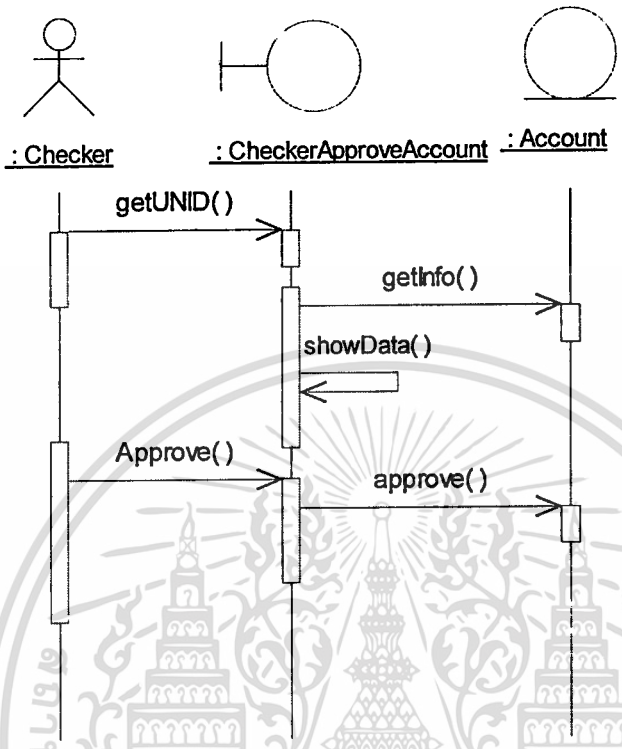
OPEN ACCOUNT ENTRY



รูปที่ 3.16 Sequence diagram สำหรับ Open Account Entry

เมื่อมีคำสั่งเปิดบัญชีจากลูกค้า Maker จะเปิดหน้าจอการเปิดบัญชีระบบจะสร้างหมายเลขบัญชีใหม่อัตโนมัติ Maker จะกรอกรายละเอียดของลูกค้า Maker กดปุ่ม Save เมื่อกรอกรายละเอียดครบถ้วน ระบบจะบันทึกข้อมูล เพื่อรอการตรวจจาก Checker

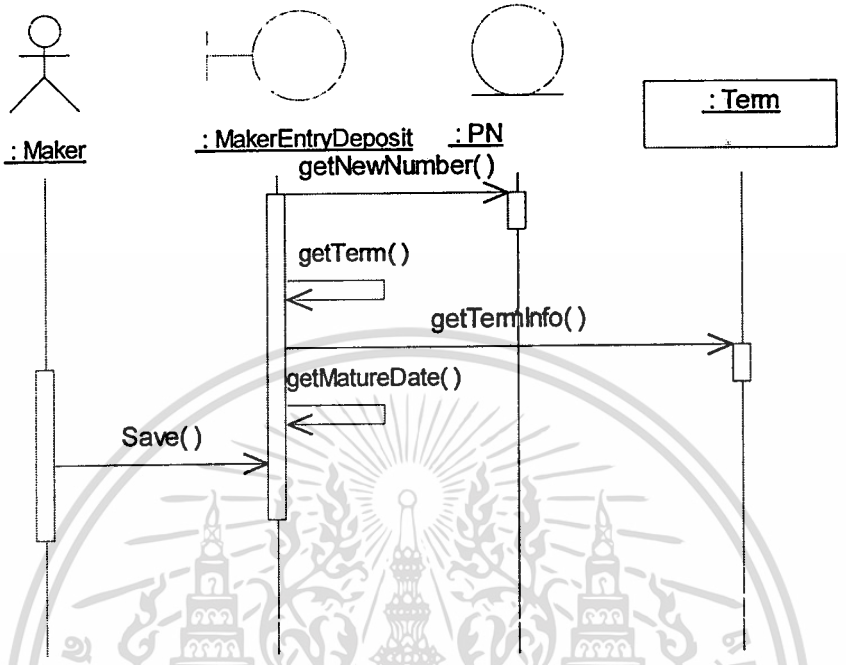
OPEN ACCOUNT APPROVAL



รูปที่ 3.17 Sequence diagram สำหรับ Open Account Approval

เมื่อ Maker คลิ๊กที่เมนู Open Account Approval ระบบจะอ่านข้อมูลของ Account ที่มีสถานะเป็น UN Checker จะเลือกข้อมูล Account ที่ต้องการตรวจสอบ ระบบแสดงข้อมูลของ Account. Checker จะตรวจสอบข้อมูลบนหน้าจอ กับข้อมูลตามใบขอเปิดบัญชี ถ้าข้อมูลถูกต้อง Checker กดปุ่ม Approve ระบบจะเปลี่ยนสถานะบัญชีเป็น LI ถ้าไม่ถูกต้อง Checker กดปุ่ม Reject ระบบจะเปลี่ยนสถานะบัญชีเป็น RN

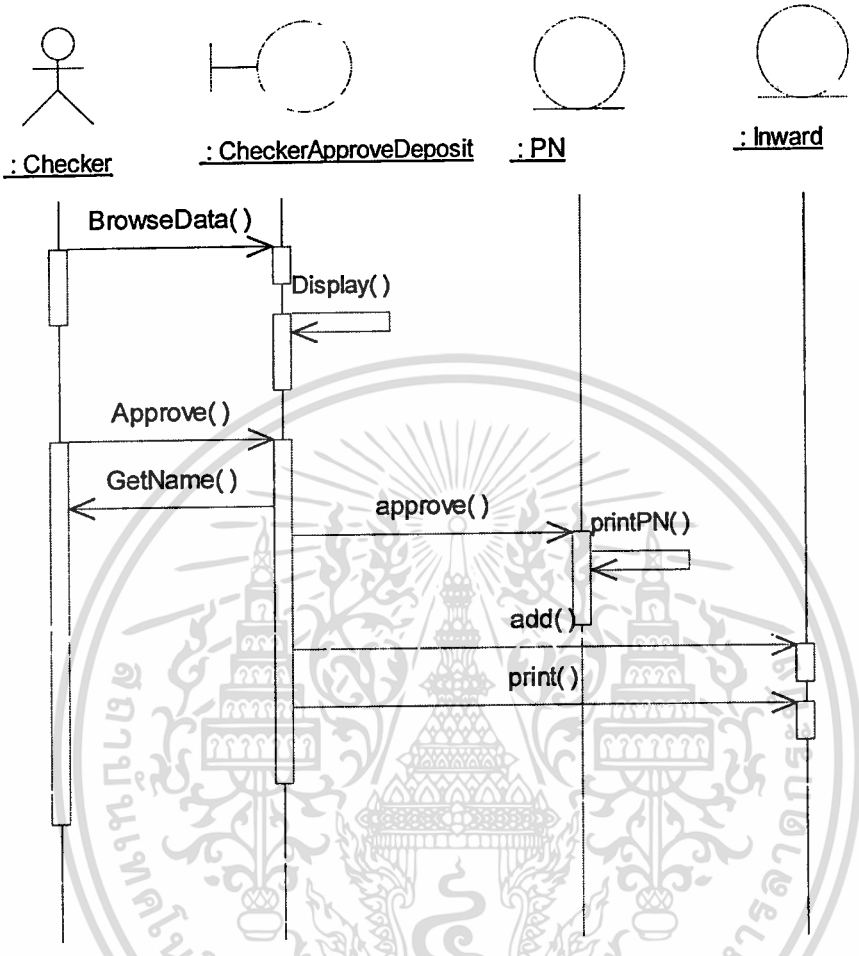
DEPOSIT P/N ENTRY



รูปที่ 3.18 Sequence diagram สำหรับ Deposit P/N Entry

ลูกค้ากรอกแบบฟอร์มการฝาก P/N และมอบให้ Maker. Maker เปิดหน้าจอการฝาก P/N ระบบจะสร้างหมายเลข P/N ให้อัตโนมัติ Maker กรอกข้อมูลตามแบบฟอร์มการฝาก P/N เช่น ระยะเวลาการฝาก โดยเรียกใช้ออปเจค Term และ อัตราดอกเบี้ย, จำนวนเงิน เป็นต้น Make บันทึกข้อมูล

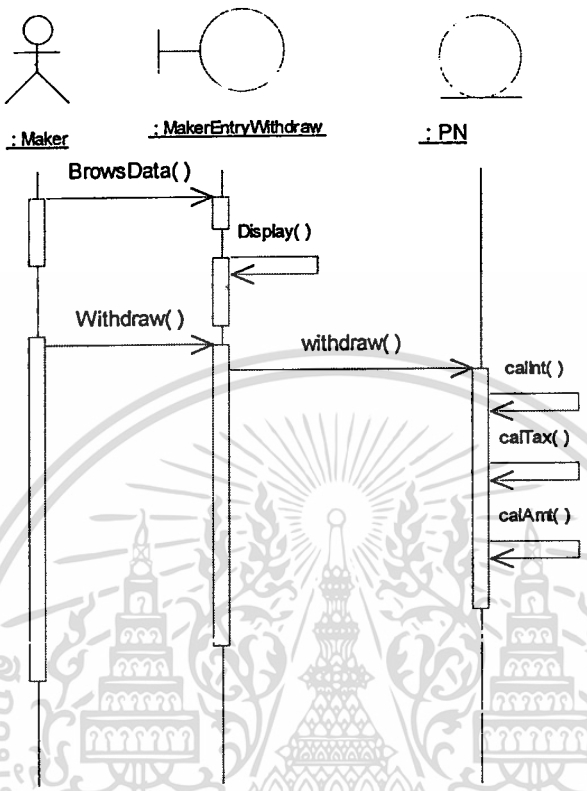
DEPOSIT P/N APPROVAL



รูปที่ 3.19 Sequence diagram สำหรับ Deposit P/N Approval

Use case นี้เริ่มเมื่อ checker ต้องการ approve ข้อมูล P/N ที่ได้รายการโดย Maker ระบบอ่านข้อมูล P/N ที่มีสถานะเป็น UN Maker เลือกข้อมูล P/N ที่ต้องการ ระบบแสดงข้อมูล P/N Maker ตรวจสอบข้อมูลจากระบบ กับแบบฟอร์มการฝาก P/N ของลูกค้า ถ้าข้อมูลถูกต้อง Maker กดปุ่ม Approve บนหน้าจอ แล้วระบบ จะพิมพ์ตั๋วสัญญาใช้เงินออกทางเครื่องพิมพ์ พร้อมใบ Inward Voucher จำนวน 2 ชุด เพื่อให้ลูกค้า 1 ชุด และให้ระบบบัญชีเก็บไว้ลง GL อีก 1 ชุด ถ้าไม่ถูกต้อง Make กดปุ่ม Reject ระบบจะเปลี่ยนสถานะ P/N เป็น Reject

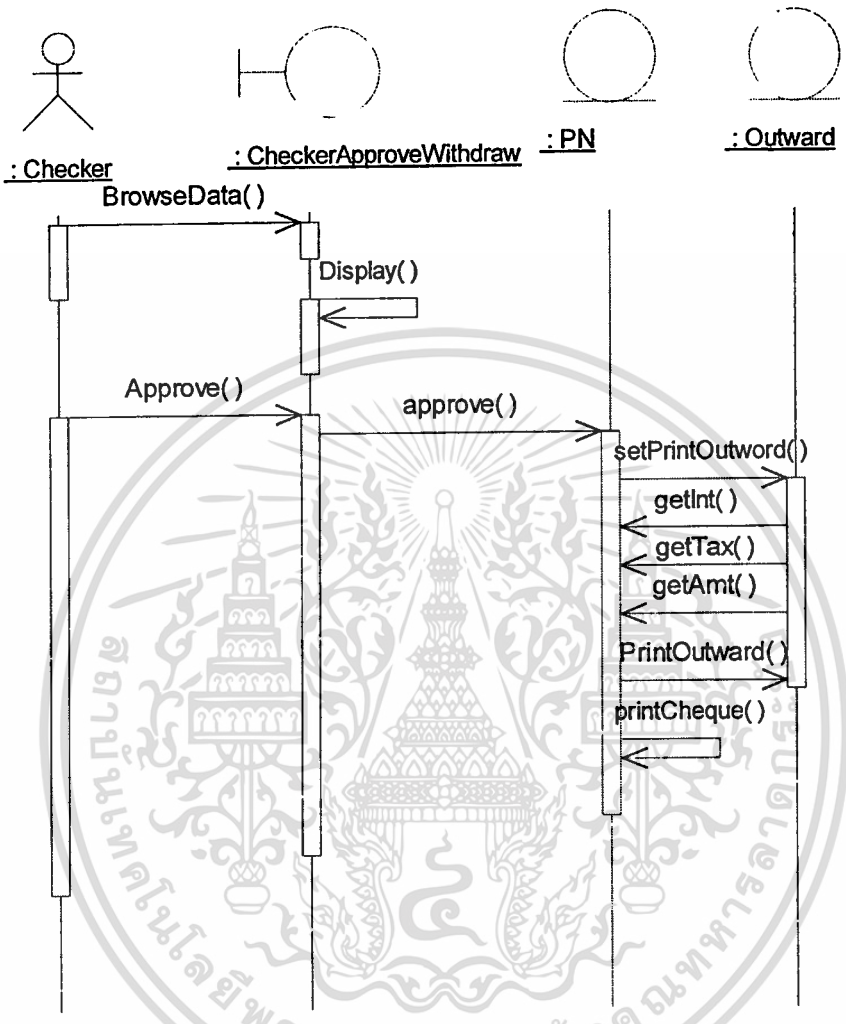
WITHDRAW P/N ENTRY



รูปที่ 3.20 Sequence diagram สำหรับ Withdraw P/N Entry

เริ่มจากที่ Customer ได้นำ P/N ที่ครบกำหนดมาไถ่ถอนและได้กรอกแบบฟอร์มการไถ่ถอน และนำมาให้ Maker เป็นผู้ทำรายการ Maker เลือกรายการ P/N จากระบบ แล้ว Maker กรอกข้อมูลการไถ่ถอนของ P/N คือ วันที่ไถ่ถอนของ P/N ระบบจะตรวจสอบว่า P/N ได้ถอนก่อนกำหนดหรือไม่ ถ้าเป็นการไถ่ถอนก่อนกำหนดระบบจะคำนวณอัตราดอกเบี้ยให้น้อยกว่าที่ได้กำหนดไว้ ถ้าเป็นการไถ่ถอนตามกำหนด ระบบจะคำนวนดอกเบี้ยตามอัตราที่ได้กำหนดไว้ ระบบคำนวณเงินต้นพร้อมดอกเบี้ยที่ได้รับ พร้อมหักภาษี ณ ที่จ่าย Maker กดปุ่ม Save บนหน้าจอ

WITHDRAW P/N APPROVAL



รูปที่ 3.21 Sequence diagram สำหรับ Withdraw P/N Approval

เมื่อ Checker ต้องการ Approve ข้อมูลการถอน P/N ที่ได้ทำรายการจาก Maker ระบบอ่านข้อมูล P/N ที่มีสถานะเป็น UN. Checker เลือกข้อมูล P/N ที่ต้องการตรวจสอบการไถ่ถอน ระบบแสดงข้อมูล P/N . Checker ตรวจสอบข้อมูลต่างๆ เช่น ชื่อบัญชี จำนวนเงินต้นและดอกเบี้ย ถ้าข้อมูลการไถ่ถอนถูกต้อง Checker กดปุ่ม Save บนหน้าจอ แล้วระบบจะพิมพ์ Cheque ให้ Customer และใบ Outward Voucher จำนวน 2 ชุด โดยให้ Customer 1 ชุด และฝ่ายบัญชี เพื่อนำไปลงรายการ GL อีก 1 ชุด ถ้าข้อมูลไม่ถูกต้อง Checker กดปุ่ม Reject บนหน้าจอ

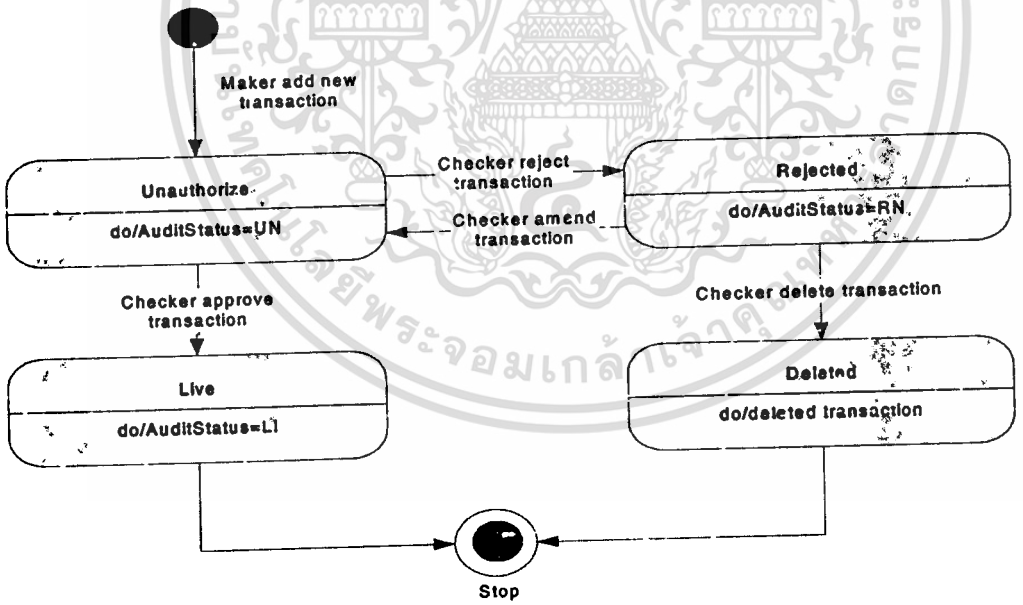
State Diagram

เป็นการแสดงวงจรชีวิตของออบเจค ซึ่งสเตทจะเป็นตัวบ่งบอกถึงเหตุการณ์ต่าง ๆ ว่ามีอะไรเกิดขึ้นบ้าง สเตทไดอะแกรม จะทำหน้าที่เชื่อมต่อทุกคลาสที่มีพฤติกรรมอันซับซ้อนให้ชัดเจนขึ้น และเป็นการอธิบายถึงพฤติกรรมของระบบ ซึ่งแต่ละเสตทจะมีความแตกต่างกันขึ้นอยู่กับสถานะในปัจจุบัน รวมทั้งมีเหตุการณ์ใดบ้างที่มีผลกับการเปลี่ยนแปลงทั้งภายในและภายนอก โดยอาจจะมีจุดเริ่มต้นและจุดได้ในหลาย ๆ จุด

ในระบบตั๋วสัญญาใช้เงิน เราได้นำระบบ Maker –Checker เข้ามาควบคุมการตรวจสอบความถูกต้องของข้อมูลโดยมีจะมีคนที่มีหน้าที่กรอกข้อมูลคือ Maker และผู้ที่ตรวจสอบข้อมูลคือ Checker ดังนั้น ออบเจค Account, PN จะต้องมีสถานะของออบเจคเป็นตัวบ่งบอกถึงการทำงานว่าอยู่ในสถานะใด

เราได้แบ่งสถานะของออปเจคออกเป็น 2 สถานะใหญ่ ๆ คือ สถานะที่ออปเจคเป็นข้อมูลใหม่ของระบบ (New record) และสถานะที่ออปเจคเป็นข้อมูลที่ใช้อยู่แล้วภายในระบบ (Live record) ซึ่งทั้งสองสถานะ หากมีการปรับปรุงข้อมูล ก็จะต้องผ่านกระบวนการ Maker-Checker เสียก่อน จึงแสดงได้ดังต่อไปนี้

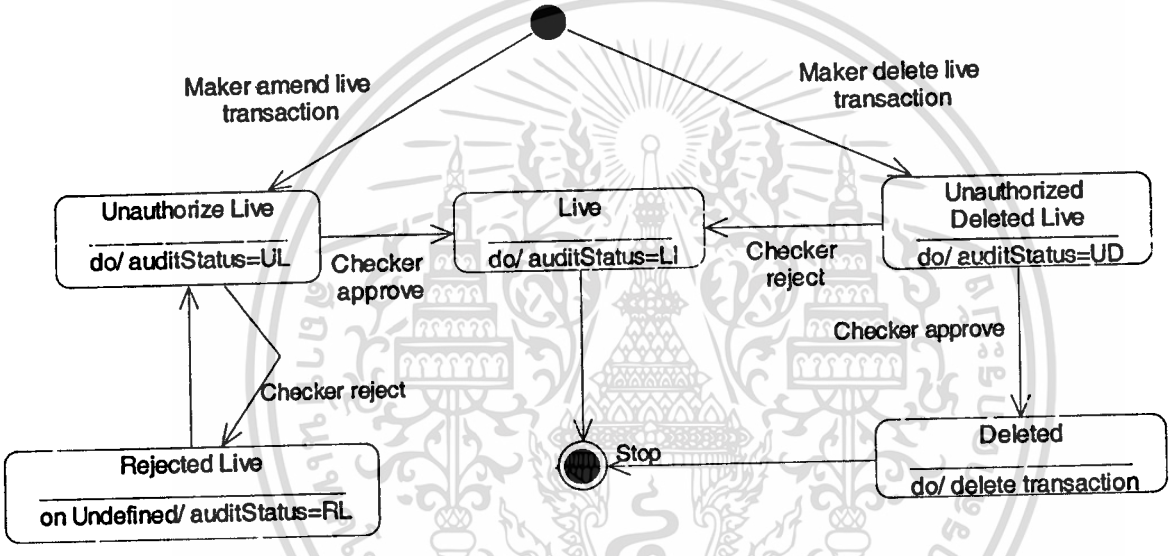
- State diagram ของการเพิ่มข้อมูล Account ใหม่เข้าไปในระบบ



รูปที่ 3.22 State diagram ของการเพิ่มข้อมูล Account ใหม่เข้าไปในระบบ

เมื่อ Maker ทำการเพิ่มข้อมูลใหม่เข้าไป ระบบจะกำหนดให้สถานะออบเจคเป็น UN เพื่อรอให้ Checker ทำการตรวจสอบข้อมูล หาก Checker ตรวจสอบข้อมูลแล้วเห็นว่าถูกต้องก็ทำการ approve ระบบก็จะกำหนดให้สถานะออบเจคเป็น LI แต่หากเห็นว่าข้อมูลไม่ถูกต้องก็ทำการ reject ระบบก็จะกำหนดสถานะออบเจคเป็น RN เพื่อรอให้ Maker ทำการแก้ไขข้อมูลให้ถูกต้อง หาก Maker แก้ไขข้อมูลแล้วจะทำการบันทึกข้อมูลเพื่อรอให้ Checker ตรวจสอบอีกครั้งซึ่งระบบจะกำหนดให้สถานะออบเจคเป็น UN อีกครั้งหนึ่ง แต่หาก Make rต้องการลบข้อมูลนั้นออกจากระบบก็สามารถลบข้อมูลได้ โดยระบบจะทำการลบข้อมูลนั้นอออกไปจากระบบ

- State diagram ของการปรับปรุงข้อมูลของ Account ที่มีอยู่แล้วในระบบ

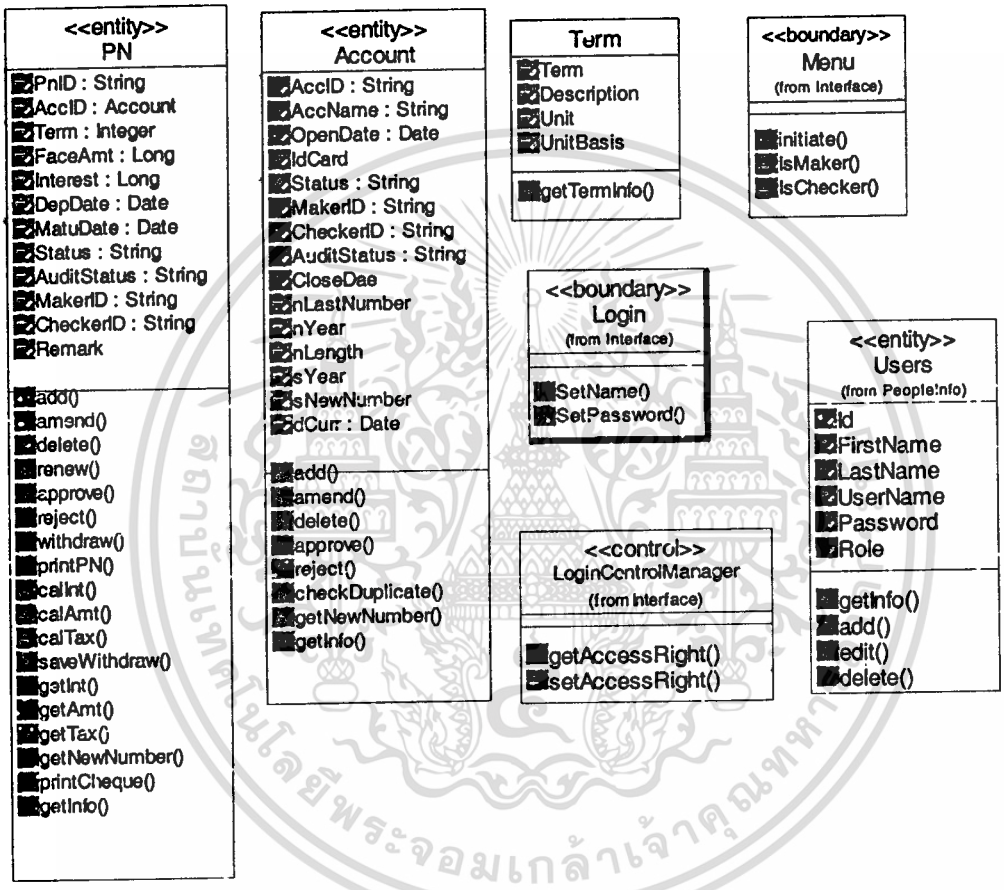


รูปที่ 3.23 State diagram ของการปรับปรุงข้อมูลของ Account ที่มีอยู่แล้วในระบบ

เมื่อ Maker แก้ไขข้อมูลที่ใช้อยู่ในระบบ ระบบก็จะกำหนดสถานะให้ออบเจคเป็นUL แล้วรอให้ Checker ทำการตรวจสอบหาก Checker ทำการตรวจสอบถูกต้องแล้ว approve ระบบก็จะกำหนดสถานะให้ออบเจคเป็น LI แต่หาก Checker ทำการ reject ระบบก็จะกำหนดสถานะให้ออบเจคเป็น RL เพื่อรอให้ Maker ทำการแก้ไขต่อไป

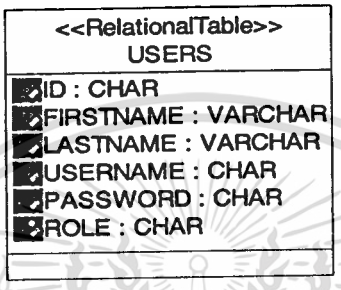
เมื่อ Maker ลบข้อมูลที่ใช้อยู่ในระบบ ระบบก็จะกำหนดสถานะให้ออบเจคเป็น UD แล้วรอให้ Checker ทำการตรวจสอบหาก Checker ทำการตรวจสอบถูกต้องแล้ว approve ระบบก็จะลบข้อมูลออกจากระบบ แต่หาก Checker ทำการ reject ระบบก็จะกำหนดสถานะให้ออบเจคเป็น LI

หลังจากที่เราได้กำหนดปฏิสัมพันธ์ต่าง ๆ ใน Sequence diagram กับทุก ๆ use case แล้ว เราจะ เพิ่มสิ่งที่เรียกว่า method และ attribute ให้กับแต่ละ class โดยสำหรับ Method เราจะนำเอา message ที่ส่งมาจาก object มาตั้งเป็นชื่อของ method [5] แต่สำหรับ attribute นั้นนักวิเคราะห์ระบบจะต้องวิเคราะห์ข้อมูลจากขั้นตอนการทำ Problem definition ว่าจะต้องมีข้อมูลอะไรบ้างสำหรับ class นั้น ๆ



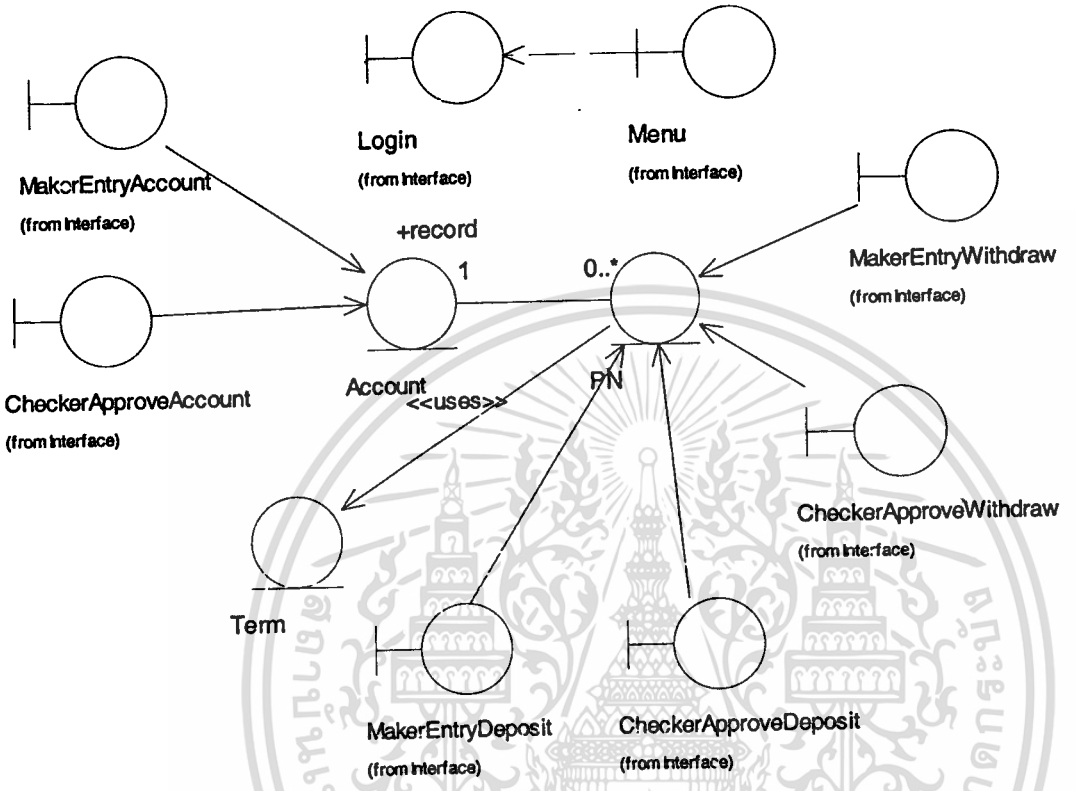
รูปที่ 3.24 คลาสต่าง ๆ ในระบบตั๋วสัญญาใช้เงิน

สำหรับActor Maker และ Checker นั้นในทางปฏิบัติได้ออกแบบมาให้เป็นคลาสในขั้นตอนการ implementation ให้เป็นเพียงคลาสเดียวคือคลาส User ที่มี attribute ต่าง ๆ ที่เหมือนกัน และจะใช้ attribute ที่มีชื่อว่า Role เป็นตัวบ่งบอกว่าออปเจคนั้นมีสถานะเป็น Maker หรือ Checker และมี stereotype เป็น RelationalTable เพื่อบ่งบอกถึงว่าคลาสนี้จะหมายถึงตารางในฐานข้อมูลเชิงสัมพันธ์ แต่สำหรับคลาสอื่น ๆ ที่มี Stereotype เป็น entity เช่น PN, Account นั้น ในขั้นตอน Implement นั้นจะไม่มีการเปลี่ยนแปลงจากคลาสเดิมที่ได้ออกแบบมา



รูปที่ 3.25 คลาส USER ที่มี Stereotype เป็น RelationalTable

สำหรับการแสดงความสัมพันธ์ระหว่าง Boundary Class กับ Entity Class นั้นเราสามารถแสดงความสัมพันธ์ระหว่างคลาสได้ดังนี้



รูปที่ 3.26 ความสัมพันธ์ของคลาสในระบบตู้สัญญาใช้เงิน

ในบทนี้เราได้ทราบถึงการแสดงลำดับเหตุการณ์ต่าง ๆ ว่ามีกิจกรรมใดเกิดขึ้นในระหว่างการทำงาน ซึ่งเราจะแสดงได้โดยใช้ Activity Diagram นอกจากนี้เราได้ทราบถึงคลาสต่าง ๆ ในระบบว่ามีอะไรบ้าง มีหน้าที่อะไรในระบบ หรือเก็บข้อมูลอะไรบ้างในระบบ ซึ่งเราใช้คลาสไดอะแกรมแสดง พร้อมทั้งความสัมพันธ์ของคลาส และยังได้แสดง Sequence Diagram แสดงลำดับการทำงานระหว่างออปเจคในเชิงเวลาว่าออปเจคใดติดต่อกับออปเจคอื่นอย่างไร นอกจากนั้นเรายังได้ทราบถึงสถานะของออบเจคที่เกิดขึ้นระหว่างการทำงานผ่านกระบวนการต่าง ๆ ด้วยการแสดงใน State Diagram ในบทต่อไปเป็นการนำเอาไดอะแกรมต่าง ๆ ที่ได้ออกแบบไปเขียนโปรแกรมและการสร้าง schema ในฐานข้อมูล เพื่อสร้างเป็นโปรแกรมประยุกต์ที่สามารถนำไปใช้ได้จริงต่อไป

บทที่ 4
การพัฒนาระบบงาน

การพัฒนาระบบงานเป็นการนำ UML ที่ได้ออกแบบมาทำการเขียนโปรแกรม ซึ่งสิ่งที่ได้ออกแบบมานั้นเป็นการออกแบบด้วยแนวคิดเชิงวัตถุ ดังนั้นภาษาที่ใช้สำหรับการเขียนโปรแกรมจะต้องมีคุณสมบัติในการสนับสนุนการเขียนโปรแกรมเชิงวัตถุด้วย สำหรับเรื่องของฐานข้อมูลที่เราใช้สำหรับเก็บข้อมูลนั้น จะใช้ฐานข้อมูลเชิงสัมพันธ์มาใช้งานในระบบ ถึงแม้ว่าในปัจจุบันจะมีโปรแกรมฐานข้อมูลที่สนับสนุนการทำงานเชิงวัตถุออกมาใช้งานบ้างแล้วก็ตาม แต่ความซับซ้อนในการออกแบบ, การใช้งาน หรือการเรียกใช้ข้อมูลยังมีอยู่มาก ดังนั้นโครงการพัฒนาระบบงานตั๋วสัญญาใช้เงินนี้จึงเน้นไปที่การออกแบบและการเขียนโปรแกรมให้สอดคล้องกับแนวคิดเชิงวัตถุมากกว่าการเน้นการเก็บข้อมูลลงในฐานข้อมูลเชิงวัตถุ

4.1 การพัฒนาโปรแกรมและเครื่องมือที่ใช้

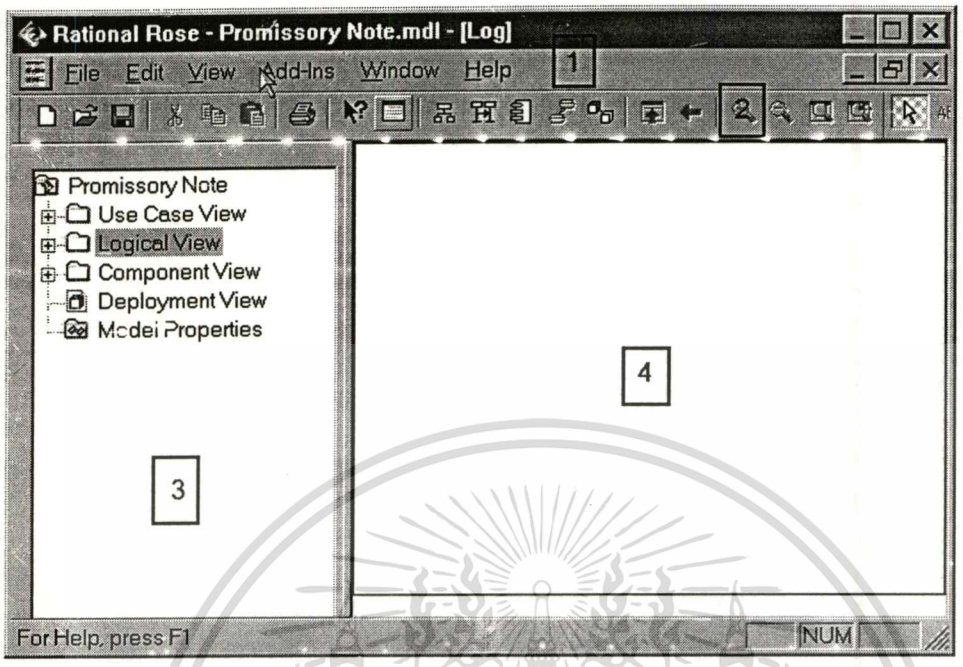
การทำงานของระบบตั๋วสัญญาใช้เงินจะประกอบด้วย 2 ส่วนหลัก คือ Application และ Database Server โดยจะอาศัย ODBC เป็นตัวกลางในการเชื่อมต่อระหว่าง Application และ Database Server

การใช้แนวคิดเชิงวัตถุและใช้ UML มาแสดงแนวคิดการออกแบบโปรแกรมนั้นเราใช้โปรแกรม Rational Rose 2000 Enterprise Edition เป็นเครื่องมือในการพัฒนา ส่วนการเขียนโปรแกรมจะใช้ Centura 1.5.1 สำหรับการเขียนโปรแกรมพร้อมกับส่วนรายงานของระบบ ส่วนฐานข้อมูลใช้โปรแกรม Microsoft Access 97 สำหรับการเก็บข้อมูล

4.1.1 Rational Rose 2000 Enterprise Edition

Rational Rose ได้ถูกออกแบบมาใช้งานสำหรับงานประเภท visual modeling เพื่อการพัฒนาโครงการต่าง ๆ ให้มีประสิทธิภาพ และยังใช้สัญลักษณ์ต่าง ๆ ที่เป็นมาตรฐานในการออกแบบด้วย UML เพื่อให้ผู้ใช้ที่ไม่จำเป็นต้องเขียนโปรแกรมเป็นก็สามารถใช้งานได้ จากที่กล่าวมา Rational Rose เป็นเครื่องมือสำหรับผู้ที่ออกแบบระบบเชิงวัตถุด้วย UML ดังนั้นจะอธิบายการใช้งาน Rational Rose เพื่อการออกแบบเชิงวัตถุด้วย UML

4.1.1.1 หน้าจอหลักของ Rational Rose



รูปที่ 4.1 หน้าจอหลักของโปรแกรม Rational Rose 2000 Enterprise Edition

เมื่อเราเริ่มใช้งานโปรแกรม Rational Rose แล้วเราจะพบกับหน้าจอของโปรแกรมดังภาพที่ 4.1 ซึ่งจะแบ่งส่วนประกอบหลักได้ 4 ส่วนดังนี้

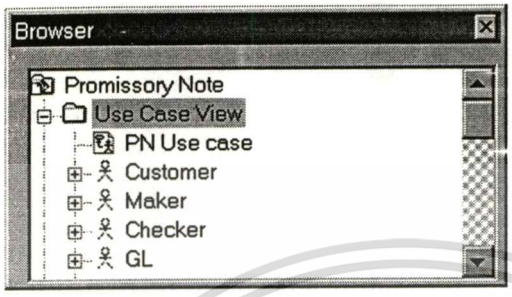
- ส่วนที่ 1 เมนูหลัก เป็นส่วนสำคัญสำหรับควบคุมการทำงานหลักของโปรแกรม
- ส่วนที่ 2 ทูลบาร์ เป็นไอคอนสำหรับใช้งานโปรแกรมตามหน้าที่ที่สำคัญ
- ส่วนที่ 3 บราวเซอร์ เป็นเครื่องมือสำหรับแสดงชื่อและไอคอนของไดอะแกรมต่าง ๆ ไม่ว่าจะเป็น Use Case, Class, Interface, Association และ Package
- ส่วนที่ 4 สำหรับแสดงไดอะแกรมต่าง ๆ ที่ได้เลือกจากบราวเซอร์

4.1.1.2 การสร้าง Actor

1. คลิ๊กขวาบน Use Case View package ในบราวเซอร์เพื่อให้เกิดเมนูขึ้น
2. เลือก New:Actor
3. ใส่ชื่อของ Actor ที่ต้องการ

4.1.1.3 การสร้าง Use Case

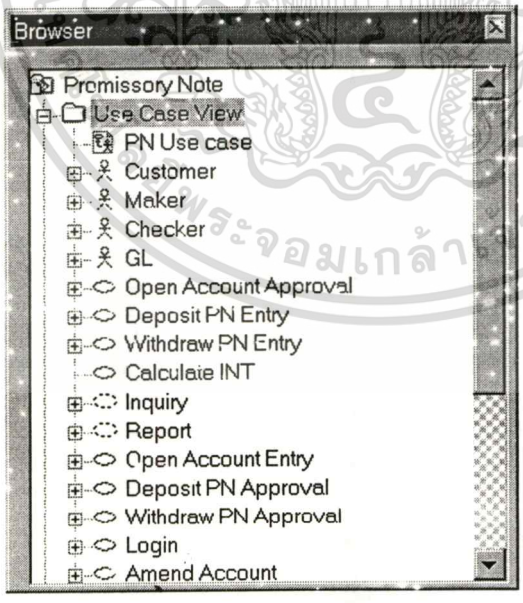
1. คลิ๊กขวาบน Use Case View package ในบราวเซอร์เพื่อให้เกิดเมนูขึ้น
2. เลือก New:Use Case
3. ใส่ชื่อของ Use Case ที่ต้องการ



รูปที่ 4.2 การสร้าง Actor

4.1.1.4 การสร้าง Use Case

1. คลิ๊กขวาบน Use Case View package ในบราวเซอร์เพื่อให้เกิดเมนูขึ้น
2. เลือก New:Use Case
3. ใส่ชื่อของ Use Case ที่ต้องการ



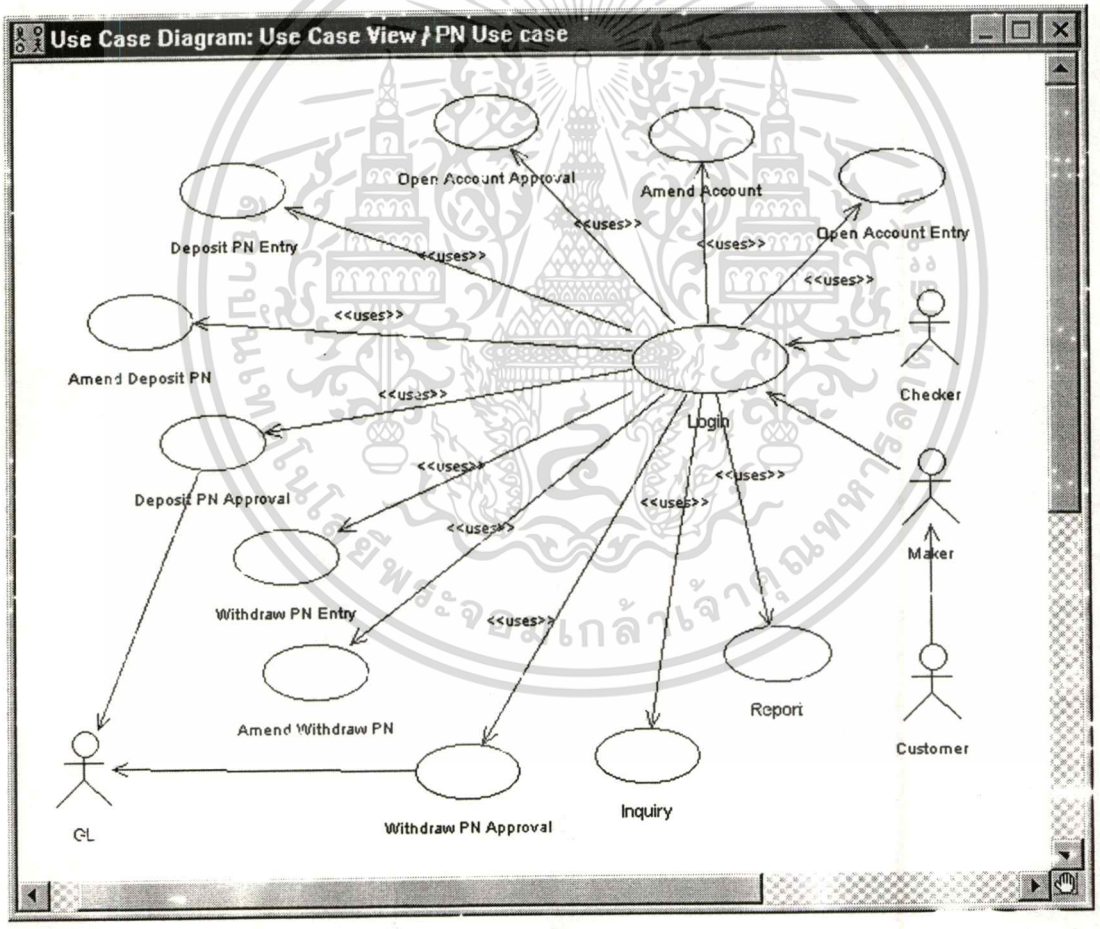
รูปที่ 4.3 การสร้าง Use Case

4.1.1.5 การสร้าง Use Case Diagram

1. ดับเบิ้ลคลิ๊กที่ Main diagram ใน Use Case View ในบราวเซอร์
2. คลิ๊กที่ actor เพื่อเลือกแล้วลากไปใส่ไว้ที่ diagram
3. เลือก actor อื่นไปไว้ตามที่ต้องการ
4. คลิ๊กที่ use caser เพื่อเลือกแล้วลากไปใส่ไว้ที่ diagram
5. เลือก use case อื่นไปไว้ตามที่ต้องการ

4.1.1.6 การเพิ่ม Relationship

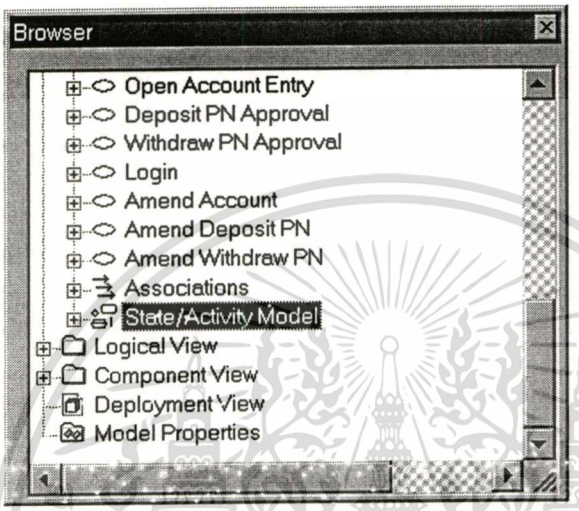
1. คลิ๊กที่ไอคอน Undirectional Association บนทูลบาร์
2. คลิ๊กที่ use case เพื่อเลือกแล้วลากไปหา use case อื่นที่ต้องการ



รูปที่ 4.5 การสร้าง Use Case Diagram และ Relationship

4.1.1.7 การสร้าง Activity Diagram

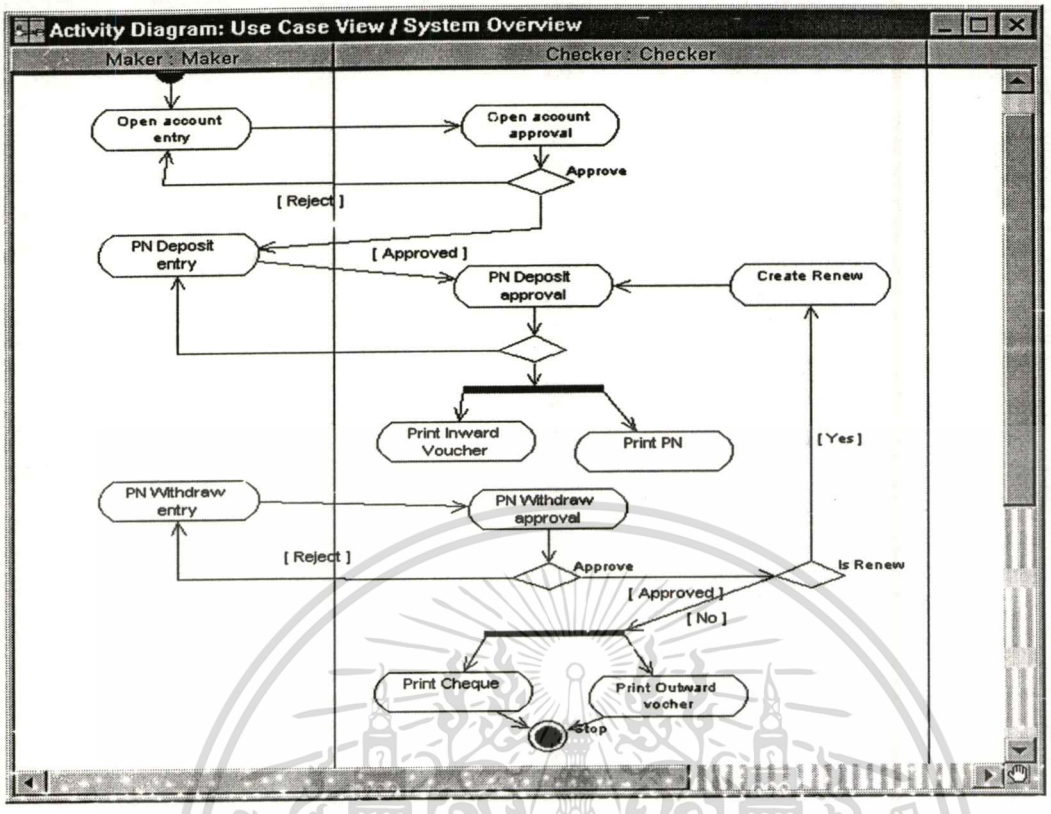
1. คลิ๊กขวาบน Use Case View ในบราวเซอร์เพื่อให้เกิดเมนูขึ้น
2. เลือก New:Activity Diagram
3. ใส่ชื่อของ Activity Diagram ที่ต้องการ
4. ดับเบิ้ลคลิ๊กที่ Activity Diagram ในบราวเซอร์เพื่อเปิดไดอะแกรม



รูปที่ 4.6 Activity Diagram ในบราวเซอร์

4.1.1.8 การสร้าง Activity, Transition, Decision, และ Synchronization Bars

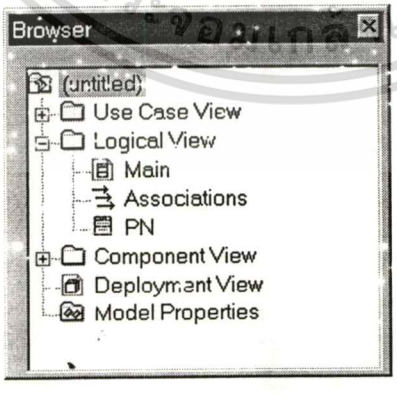
1. คลิ๊ก Activity บนทูลบาร์ แล้วคลิ๊กที่ไดอะแกรมเพื่อวางตามต้องการ
2. คลิ๊ก Transition บนทูลบาร์ แล้วลากระหว่าง Activity ตามต้องการ
3. คลิ๊ก Decision บนทูลบาร์ แล้วคลิ๊กที่ไดอะแกรมเพื่อวางตามต้องการ
4. คลิ๊ก Synchronization Bars บนทูลบาร์ แล้วคลิ๊กที่ไดอะแกรมเพื่อวางตามต้องการ



รูปที่ 4.7 Activity, Transition, Decision, และ Synchronization Bars

4.1.1.9 การสร้าง Class

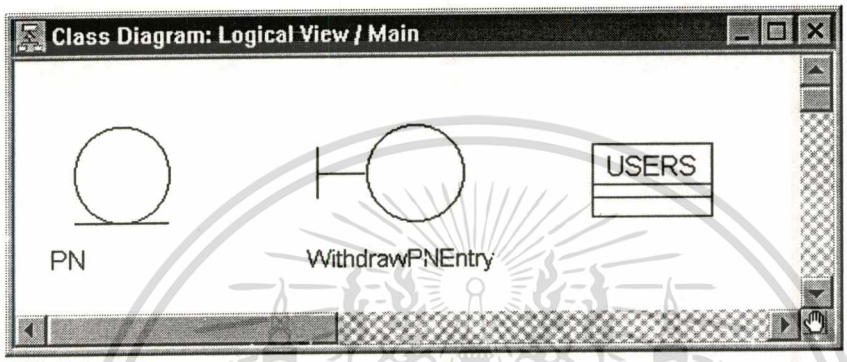
1. คลิ๊กขวาบน Logical View ในบราวเซอร์
2. เลือก New:Class
3. ใส่ชื่อของ Class ที่ต้องการ



รูปที่ 4.8 การสร้าง Class

4.1.1.10 การสร้าง Stereotype สำหรับ Class

1. คลิ๊กขวาที่ Class ในบราวเซอร์เพื่อให้เกิดเมนูขึ้น
2. เลือก Open Specification
3. เลือก General tab
4. คลิ๊ก Stereotype field เพื่อเลือก stereotype ที่ต้องการ
5. คลิ๊กที่ปุ่ม OK



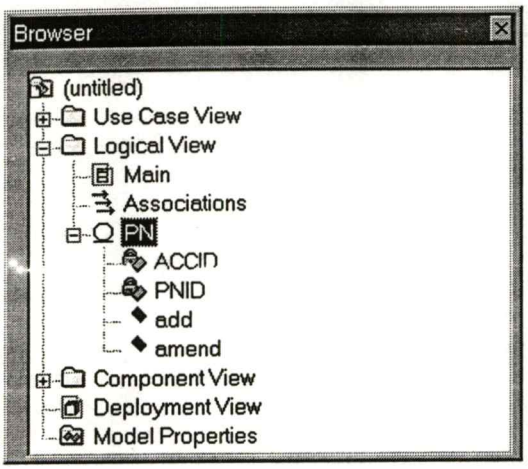
รูปที่ 4.9 การสร้าง Stereotype ของ Class

4.1.1.11 การสร้าง Attribute สำหรับ Class

1. คลิ๊กขวาเพื่อเลือก Class ในบราวเซอร์เพื่อให้เกิดเมนูขึ้น
2. เลือก New:Attribute
3. ใส่ชื่อของ Attribute ตามที่ต้องการ

4.1.1.12 การสร้าง Attribute สำหรับ Class

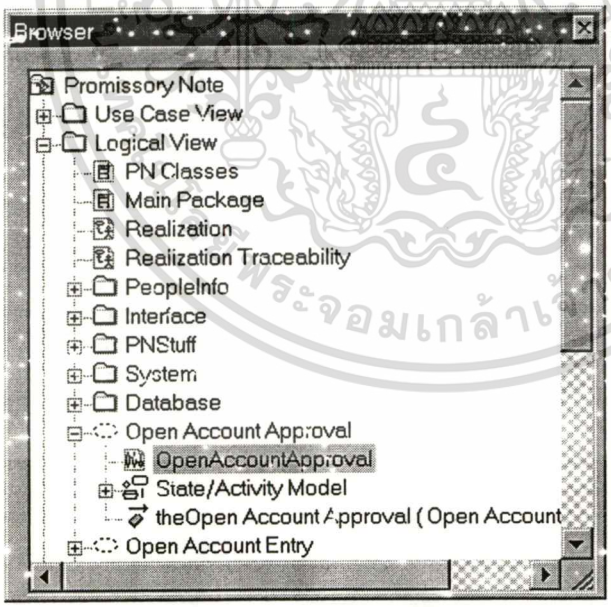
1. คลิ๊กขวาเพื่อเลือก Class ในบราวเซอร์เพื่อให้เกิดเมนูขึ้น
2. เลือก New:Operation
3. ใส่ชื่อของ Operation ตามที่ต้องการ



รูปที่ 4.10 การสร้าง Attribute และ Operation ของ Class

4.1.1.13 การสร้าง Sequence Diagram

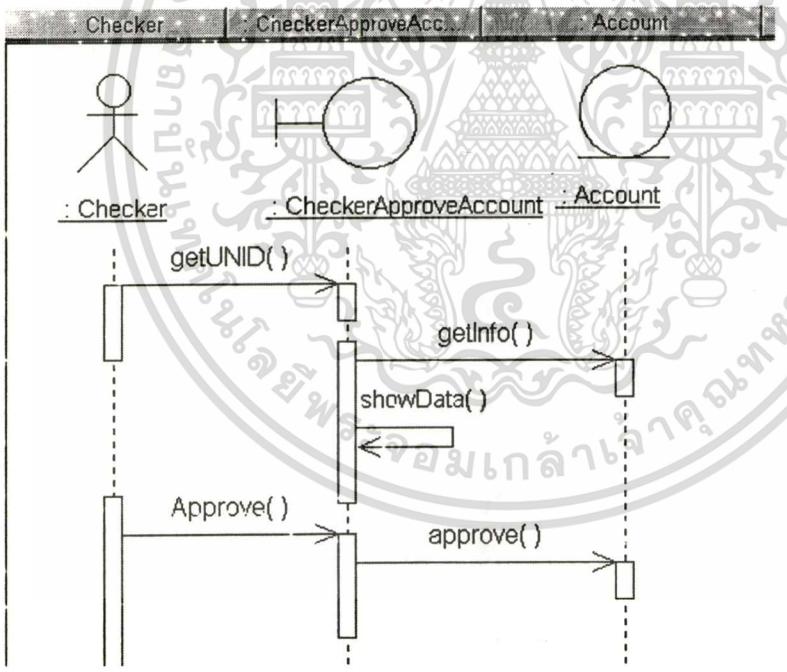
1. คลิ๊กขวาเพื่อเลือก Use case ใน Logical View ในบราวเซอร์เพื่อให้เกิดเมนูขึ้น
2. เลือก New:Sequence Diagram
3. ใส่ชื่อของ Sequence diagram ตามที่ต้องการ



รูปที่ 4.11 การสร้าง Sequence Diagram

4.1.1.14 การสร้าง Objects และ Message ใน Sequence Diagram

1. ดับเบิ้ลคลิ๊กบน Sequence Diagram บนบราวเซอร์
2. เลือก Actor ในบราวเซอร์
3. ลาก Actor ไปวางไปบน Sequence diagram
4. เลือก Object ไอคอนที่อยู่บนทูลบาร์
5. คลิ๊กบน Sequence Diagram เพื่อวาง Object
6. ใส่ชื่อของ Object
7. ทำซ้ำขั้นตอนข้างบนเพื่อให้ครบกับสิ่งที่เราวิเคราะห์
8. เลือกไอคอน Object Message จากทูลบาร์
9. คลิ๊กบน Actor หรือ Object เพื่อเลือกแล้วลาก Message ไปมาตามต้องการ
10. สามารถใส่ชื่อของ Message ได้ตามต้องการ
11. ทำซ้ำขั้นตอนที่ 7-9 จนครบตามที่เราได้วิเคราะห์



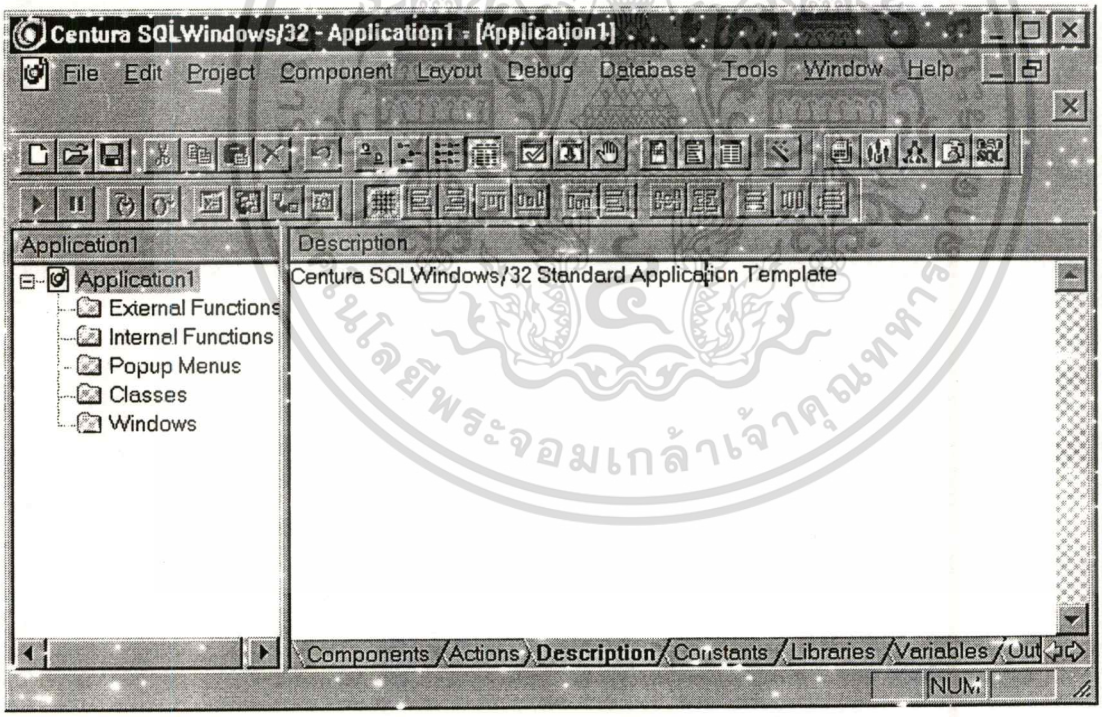
รูปที่ 4.12 การสร้าง Sequence Diagram ด้วย Object และ Message

4.1.2 Centura 1.5.1

เนื่องจากการออกแบบของระบบตั๋วสัญญาใช้เงิน เป็นการออกแบบเชิงวัตถุ ดังนั้นการเขียนโปรแกรมจึงจำเป็นที่จะต้องใช้ภาษาโปรแกรมที่สนับสนุนการเขียนโปรแกรมเชิงวัตถุด้วย โปรแกรม Centura 1.5.1 เป็นโปรแกรมสำหรับการเขียนโปรแกรมบนระบบปฏิบัติการวินโดวส์ที่สนับสนุนการเขียนโปรแกรมเชิงวัตถุ ซึ่งสามารถสร้างคลาส กำหนด Attribute, Property หรือแม้กระทั้งการถ่ายทอดคุณสมบัติจากคลาสหนึ่งไปยังอีกคลาสหนึ่งได้

ส่วนประกอบต่าง ๆ ของโปรแกรม Centura 1.5.1 อธิบายได้ดังนี้

1. เมนูหลัก มีหน้าที่สำหรับการควบคุมการทำงานของโปรแกรมในด้านต่าง ๆ เช่น การเปิด-ปิดไฟล์, การกำหนดคอมโพเนนต่าง ๆ, การ debug, การเปิดดูคู่มือการใช้งาน
2. Application Windows มีไว้สำหรับการแสดงรายละเอียดของ External Function, Internal Function, Classes, Windows
3. Application Views มีไว้สำหรับการจัดหน้าจอเพื่อให้เหมาะสม กับการทำงาน



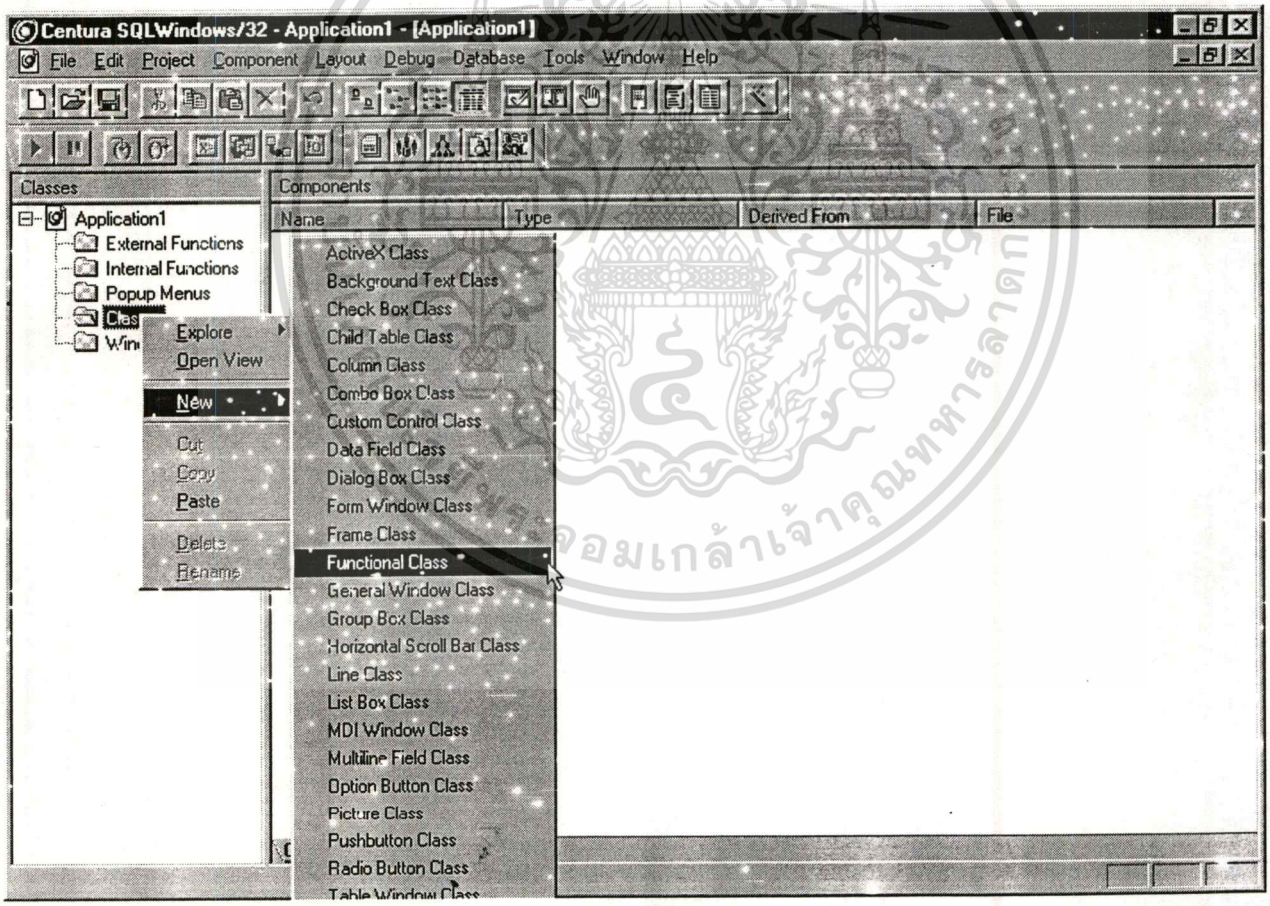
รูปที่ 4.13 หน้าจอหลักของโปรแกรม Centura 1.5.1

เมื่อเปิดโปรแกรม Centura โปรแกรมจะแสดง MDI frame ที่ประกอบด้วยวินโดวส์ย่อย ๆ ภายใน ซึ่งเราเรียก MDI นี้ว่า application view ส่วนวินโดวส์ที่อยู่ภายในด้านซ้ายเราเรียกว่า Application Windows ส่วนด้านขวาเรียกว่า Application Views

4.1.2.1 การสร้างคลาส

คลาสที่เราจะสร้างในโปรแกรม Centura เป็นคลาสประเภท Functional (โปรแกรม Centura สามารถสร้างคลาสได้ 3 ประเภทคือ วินโดวส์คลาส, General Window Class และ Functional Class) ซึ่ง Functional Class สามารถ

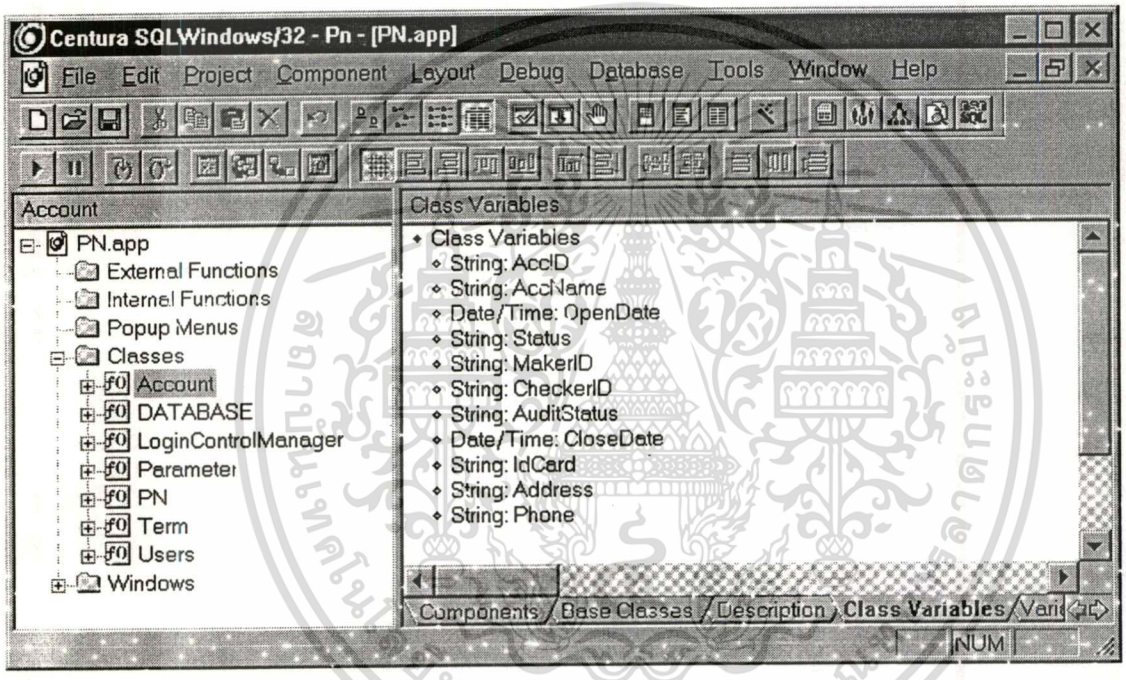
1. คลิ๊กขวาที่ไอคอน class ใน Tree view
2. เลือก New > Functional Class ดังรูปที่
3. ตั้งชื่อคลาสตามที่เราได้ออกแบบมา เช่น Account, PN



รูปที่ 4.14 การสร้างคลาสในโปรแกรม Centura 1.5.1

4.1.2.2 การสร้าง Attribute

1. คลิ๊กคลาสใน Tree view ที่เราต้องการจะสร้าง attribute
2. ใน Tab view คลิ๊กที่ Class Variables Tab
3. คลิ๊กขวาที่ Class Variables เพื่อสร้าง attribute ของคลาส
4. เลือก Add Next Level
5. เลือกชนิดของตัวแปร ตามที่เราได้ออกแบบไว้
6. ตั้งชื่อของ attribute ให้ตรงตามที่เราได้ออกแบบไว้
7. ทำซ้ำขั้นตอนที่ 4 ไปจนกระทั่งครบ



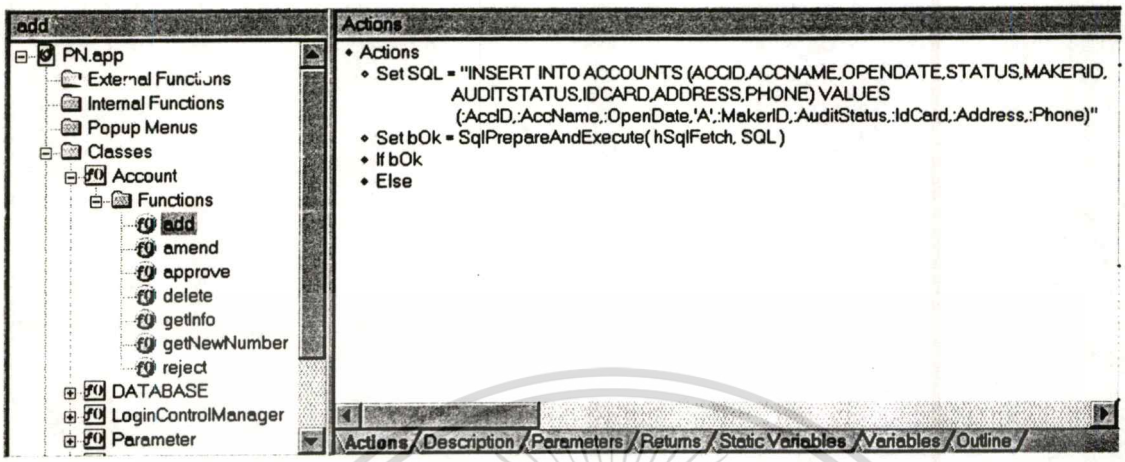
รูปที่ 4.15 การสร้าง attribute ในโปรแกรม Centura 1.5.1

4.1.2.3 การสร้าง Operation

1. ดับเบิ้ลคลิ๊กคลาสใน Tree view ที่เราต้องการจะสร้าง operation
2. ใน Tree view จะพบว่าในคลาสที่เราเลือกจะแสดงไอคอน Functions ขึ้นมา ดังรูปที่
3. คลิ๊กขวาที่ไอคอน Functions เพื่อสร้าง operation ของคลาส
4. เลือก New > Function
5. ตั้งชื่อของ operation ตามที่เราได้ออกแบบมา
6. คลิ๊กที่ไอคอนของ operation เพื่อเขียนโค้ดโปรแกรม

7. ใน Tab view คลิ๊กที่ Actions Tab
8. เขียนโค้ดโปรแกรมเพื่อให้ทำงานตามที่เราได้ออกแบบมา

ทำซ้ำขั้นตอน 4 ไปจนกระทั่งครบ



รูปที่ 4.16 การสร้าง operation ในโปรแกรม Centura 1.5.1

4.1.3 Microsoft Access 97

เราสามารถนำคลาสที่ได้ออกแบบมาสร้างตารางฐานข้อมูลในโปรแกรม Microsoft Access 97 ได้ โดยเลือกคลาสที่มี Stereotype ที่เป็น Entity และมีประเภทของ Attribute ที่เป็น Public เท่านั้นที่จะนำไปสร้างเป็น Field ในตารางฐานข้อมูล

4.1.3.1 วิธีการแปลงข้อมูลจากคลาสไปเป็นตารางในฐานข้อมูลเชิงสัมพันธ์

เทคโนโลยีของฐานข้อมูลเชิงสัมพันธ์ (Relation Database) มีใช้มานานและมีความสามารถสูงschema ของฐานข้อมูลเชิงสัมพันธ์จะเกิดจากหลาย ๆ ตาราง (Table)แต่ละตารางประกอบไปด้วยหลายๆ แถว (rows) และหลาย ๆคอลัมน์(columns)แต่ละคอลัมน์จะมีชื่อและชนิดของข้อมูลสิ่งที่คู่กับตารางในโมเดลเชิงวัตถุคือคลาสซึ่งคลาสมีชุดของattributesและอธิบายbehaviorของคลาสด้วยmethodการเปรียบเทียบกันระหว่างฐานข้อมูลเชิงสัมพันธ์กับแนวความคิดเชิงวัตถุได้ดังนี้

แถวของตารางในฐานข้อมูลเชิงสัมพันธ์เก็บข้อมูลแบบมีได้ค่าเดียว (single) เทียบเคียงได้กับ object ที่เป็น instantiate จาก คลาส ในแนวความคิดเชิงวัตถุ

- stored procedure ในฐานข้อมูลเชิงสัมพันธ์เทียบเคียงได้กับ method ในแนวความคิดเชิงวัตถุ stored procedure คือ โมดูล์ที่เก็บ SQL ที่ compile ไว้แล้วซึ่งจะเก็บอยู่ที่ฐานข้อมูล และประมวลผลบน server เพื่อที่บังคับ business rule ต่าง ๆ ที่กำหนดเกี่ยวกับข้อมูล แต่ในการพัฒนาโครงการตั๋วสัญญาใช้เงินนี้ ได้ใช้ฐานข้อมูลเป็น Microsoft Access 97 ซึ่งไม่สนับสนุนการทำงานแบบ stored procedure ดังนั้น method ต่าง ๆ

ก็จะถูกเขียนให้อยู่ในโปรแกรม Centura ดังที่กล่าวไปแล้ว

ดังนั้นการแปลงจึงเกิดระหว่างตารางกับ คลาส, ระหว่างคอลัมน์กับ attribute, ระหว่างแถวกับ object

การกำหนด Primary Key

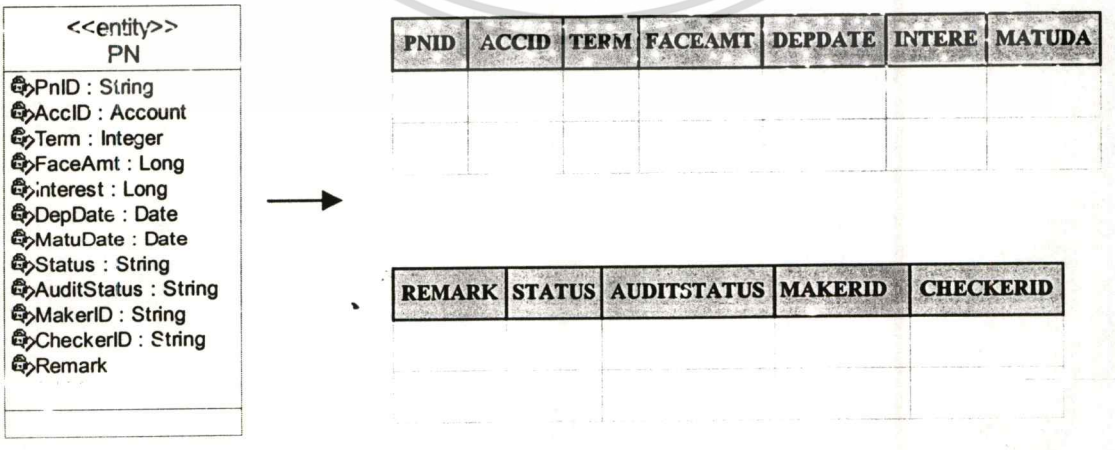
การกำหนด primary key ของตาราง มีอยู่ 2 วิธีคือ

- Existence-based Identity เพิ่ม identifier attribute ให้ในแต่ละ class แล้วใช้มันเป็น primary key ของตาราง และ primary key ของ association table ที่มาจาก association class ประกอบไปด้วย identifier attribute ของหนึ่งหรือมากกว่าหนึ่ง class ที่เกี่ยวข้อง ข้อดีของวิธีนี้ คือ primary key มีขนาดเล็ก และมีขนาดเดียวกัน ส่วนข้อเสีย คือ primary key จะไม่มีความหมายภายในตัวมันเองทำให้ยากต่อการบำรุงรักษา
- Value-based Identity นำ attribute ในแง่ความเป็นจริงมาเป็นตัว identify ของแต่ละ object ข้อดีของวิธีนี้ คือ primary key มีความหมายในตัวของมันทำให้ง่ายต่อการบำรุงรักษา ส่วนข้อเสีย คือ ยากที่จะเปลี่ยนแปลง primary key และบาง object ก็ไม่มี identifier ในแง่ความเป็นจริง

สำหรับการพัฒนาโครงการนี้ใช้วิธี Value-based Identity เพื่อให้ primary key มีความหมายต่อตัว object และทุก ๆ คลาสในระบบจะมี identifier ที่ออกแบบมาตั้งแต่แรกแล้ว

การแปลงข้อมูลจากคลาสไปเป็นตาราง

เป็นการแปลงข้อมูลตรง ๆ จากคลาสไปเป็นตาราง โดยชื่อของคลาสจะกลายไปเป็นชื่อของตาราง ส่วน attribute ของคลาสจะกลายไปเป็นคอลัมน์ในตาราง ข้อมูลในแต่ละแถวของตารางที่ได้จะแสดงถึง object แต่ละตัว ในการ mapping คลาส หนึ่ง คลาส จะ map ไปเป็นตารางได้ 1 ตาราง

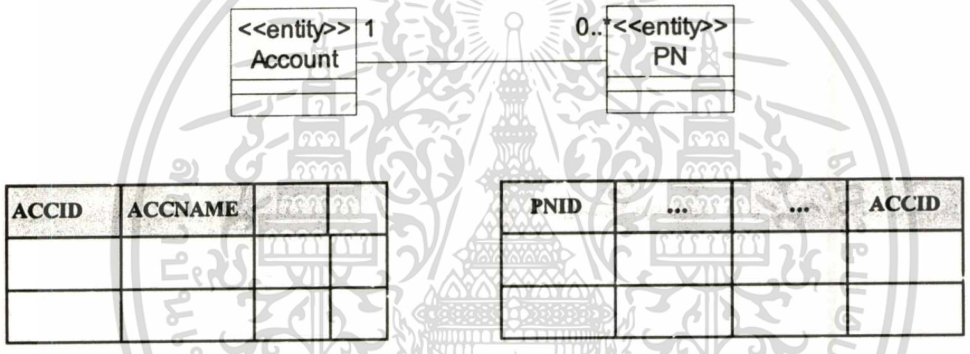


รูปที่ 4.17 การแปลงข้อมูลจากคลาสไปเป็นตาราง

การแปลงข้อมูลจากความสัมพันธ์ของคลาสไปเป็นข้อมูลในตาราง

นอกจากการแปลงแต่ละ attribute ของ class ไปเป็นคอลัมน์ แล้วยังมีการเพิ่มคอลัมน์จากการกำหนด primary key โดยได้มาจากความสัมพันธ์ของคลาสในลักษณะ Association

- Many-to-many Association กำหนด primary key ของ association เกิดจากการรวมกันของ primary key ของแต่ละคลาส
- One-to-many association กำหนด foreign key ในตารางฝั่ง many โดยชื่อ role จะกลายเป็นชื่อของ attribute ที่เป็น foreign
- Zero-or-one-to-exactly-on association กำหนด foreign ในตาราง ฝั่ง "zero-or-one"
- One-to-one association กำหนด foreign key ในตาราง ฝั่งใดฝั่งหนึ่ง



รูปที่ 4.18 การแปลงข้อมูลความสัมพันธ์ของคลาสไปเป็นตาราง แบบ One-to-many

4.14 Data Dictionary

เมื่อเราได้ Schema ของฐานข้อมูลเชิงสัมพันธ์แล้ว การทำงานขั้นต่อไปคือการแสดงความหมายและการแสดงชนิดของข้อมูลของแต่ละคอลัมภ์ในแต่ละตาราง นั่นคือการทำ Data dictionary ของระบบ โดยต่อไปนี้เป็นการแสดง Data dictionary ของระบบตั๋วสัญญาใช้เงิน

Table Name : ACCOUNT

Field Name	Type	Description	Primary Key
ACCID	TEXT	หมายเลขบัญชี	Y
ACCNAME	TEXT	ชื่อบัญชี	
IDCARD	TEXT	หมายเลขบัตรประจำตัว	

OPENDATE	DATE	วันเปิดบัญชี	
ADDRESS	TEXT	ที่อยู่	
STATUS	TEXT	สถานะของของบัญชี	
MAKERID	TEXT	ชื่อของ Maker	
CHECKERID	TEXT	ชื่อของ Checker	
AUDITSTATUS	TEXT	สถานะของเรคคอร์ด	
CLOSEDATE	DATE	วันที่ปิดบัญชี	

Table Name : PN

Field Name	Type	Description	Primary Key
PNID	TEXT	หมายเลขตั๋วสัญญาใช้เงิน	Y
ACCID	TEXT	หมายเลขบัญชี	
DEPOSITDATE	DATE	วันที่ฝาก	
TERM	TEXT	รหัสของช่วงเวลา	
MATUREDATE	DATE	วันครบอายุ	
FACEAMT	NUMBER	ราคาหน้าตั๋วสัญญาใช้เงิน	
INTEREST	NUMBER	อัตราดอกเบี้ย	
REMARK	TEXT	หมายเหตุการฝาก	
WREMARK	TEXT	หมายเหตุการถอน	
STATUS	TEXT	สถานะของตั๋วสัญญาใช้เงิน	
WITHDRAWDATE	DATE	วันที่ถอนตั๋วสัญญาใช้เงิน	
INTAMT	NUMBER	จำนวนดอกเบี้ยที่ได้รับ	
TAX	NUMBER	อัตราภาษีที่ต้องหักจากดอกเบี้ย	
TAXAMT	NUMBER	จำนวนภาษีที่ต้องหักจากดอกเบี้ย	
NETINT	NUMBER	จำนวนดอกเบี้ยสุทธิ (หลังหักภาษี)	
NETAMT	NUMBER	จำนวนสุทธิ (ราคาหน้าตั๋ว + ดอกเบี้ยสุทธิ)	
AUDITSTATUS	TEXT	สถานะของเรคคอร์ด	
MAKERID	TEXT	ชื่อของ Maker	
CHECKERID	TEXT	ชื่อของ Checker	

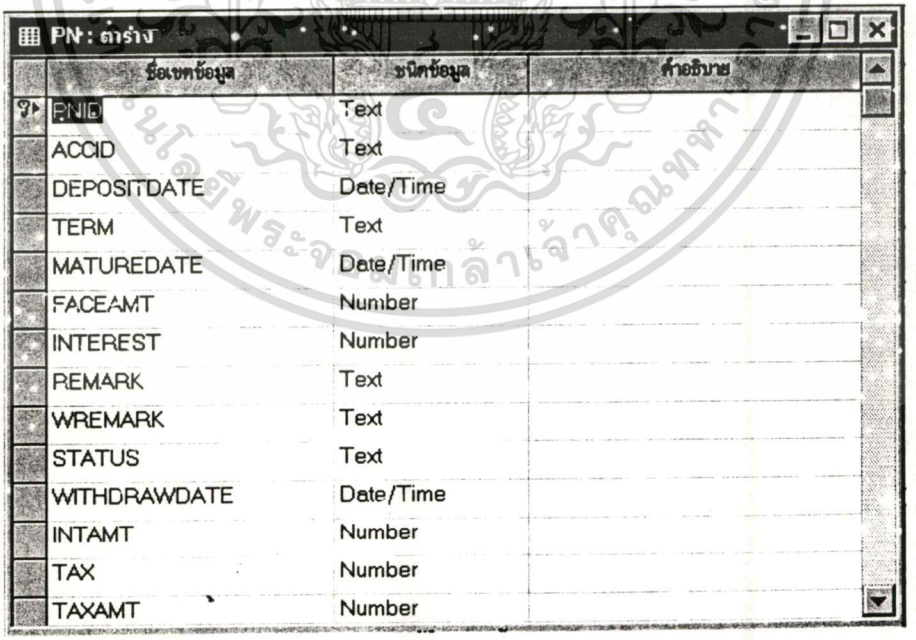
Table Name : TERM

Field Name	Type	Description	Primary Key
TERM	TEXT	รหัสของช่วงเวลา	Y
DESC	TEXT	คำอธิบายรายละเอียด	
UNITBASIS	NUMBER	จำนวนวันที่ใช้คำนวนดอกเบี้ย	

Table Name : USERS

Field Name	Type	Description	Primary Key
ID	AUTONUMBER	หมายเลขบัญชี	Y
FIRSTNAME	TEXT	ชื่อเต็มของผู้ใช้งาน	
LASTNAME	TEXT	ชื่อสกุลของผู้ใช้งาน	
USERNAME	TEXT	ชื่อที่ใช้ Log in	
PASSWORD	TEXT	รหัสผ่าน	
ROLE	TEXT	MAKER/CHECKER	

จาก Data dictionary มาสร้างเป็นตารางฐานข้อมูลในโปรแกรม Microsoft Access 97 ได้ดังภาพ

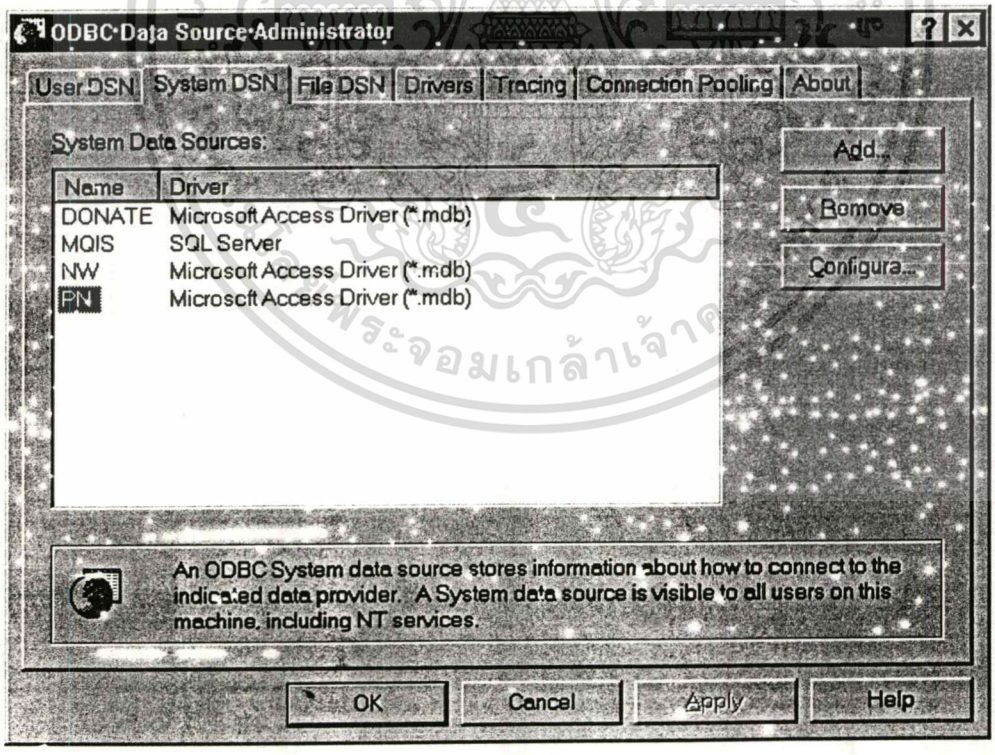


รูปที่ 4.19 การนำ attribute มาสร้างเป็น field ในโปรแกรม MS

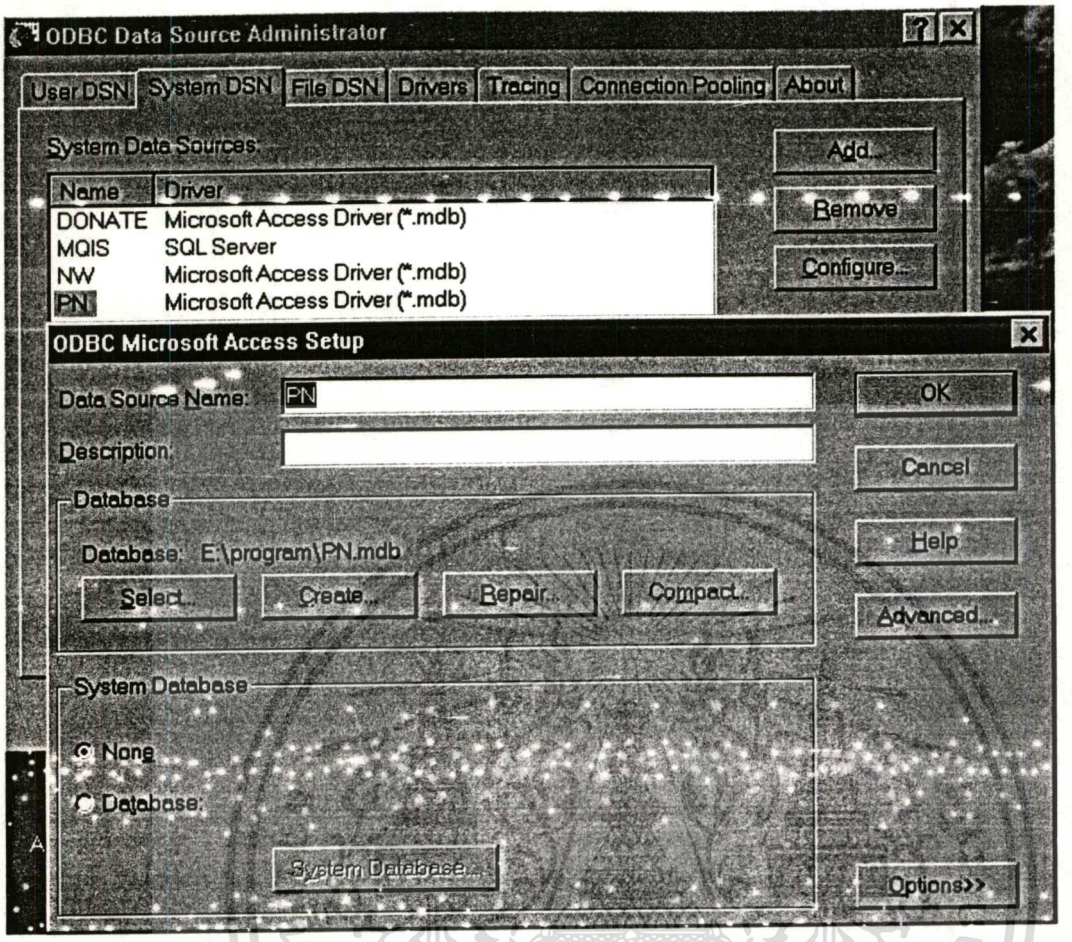
4.1.4 Microsoft ODBC

โปรแกรม Microsoft ODBC เป็นMiddlewareของบริษัทMicrosoftมีหน้าที่เป็นตัวกลางในการติดต่อจากapplication ไปยังฐานข้อมูล Microsoft Access ขั้นตอนต่อไปนี้เป็นขั้นตอนการติดตั้ง Microsoft ODBC เพื่อให้สามารถติดต่อกับโปรแกรมที่เราพัฒนาได้

1. ขณะทำงานบนวินโดวส์ คลิ๊ก Start>Settings>Control Panel
2. ดับเบิ้ลคลิ๊กที่ไอคอน ODBC Data Source (32 bit)
3. คลิ๊กที่ System DSN (ดังรูปที่ 4.17)
4. คลิ๊กที่ปุ่ม Add
5. เลือก Microsoft Access Driver (*.mdb)
6. คลิ๊กที่ปุ่ม Finish
7. พิมพ์ PN ที่ช่อง Data Source Name (ดังรูปที่ 4.18)
8. คลิ๊กที่ปุ่ม Select เพื่อเลือกฐานข้อมูลที่ได้สร้างไว้ด้วยโปรแกรม Microsoft Access 97
9. คลิ๊กที่ปุ่ม Ok



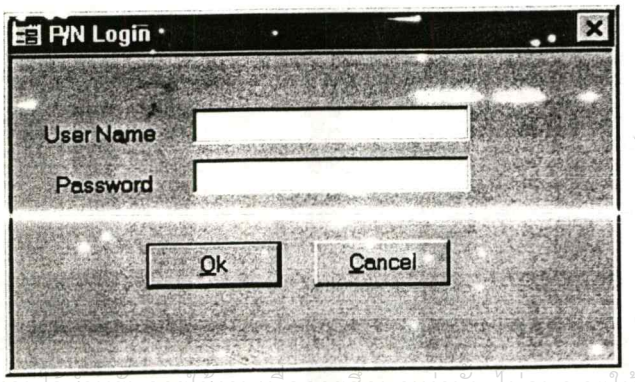
รูปที่ 4.20 การสร้าง Data Source ด้วยโปรแกรม Microsoft



รูปที่ 4.21 การใส่ Data Source Name

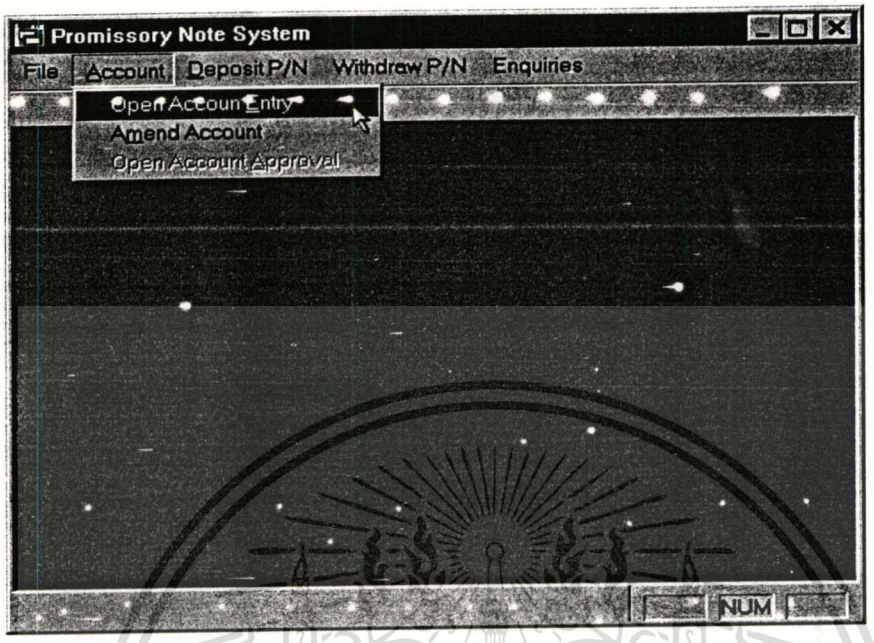
4.2 หน้าจอของโปรแกรมตัวสัญญาใช้เงิน

การเข้าสู่ระบบ เมื่อผู้ใช้เข้าสู่หน้าจอแรกของระบบแล้วผู้ใช้ต้ องป้ อน้ อูมล รหัสผู้ใช้และรหัสผ่านของตัวเองระบบ จะทำการตรวจสอบสิทธิของผู้ใช้นั้นกับฐานข้อมูลที่มีอยู่ หากชื่อผู้ใช้ไม่มีในฐานข้อมูลหรือรหัสผิด ระบบจะแสดงข้อความเตือน หากผู้ใช้กรอกผิดติดต่อกัน 3 ครั้ง ระบบจะออกจากการทำงานทันที



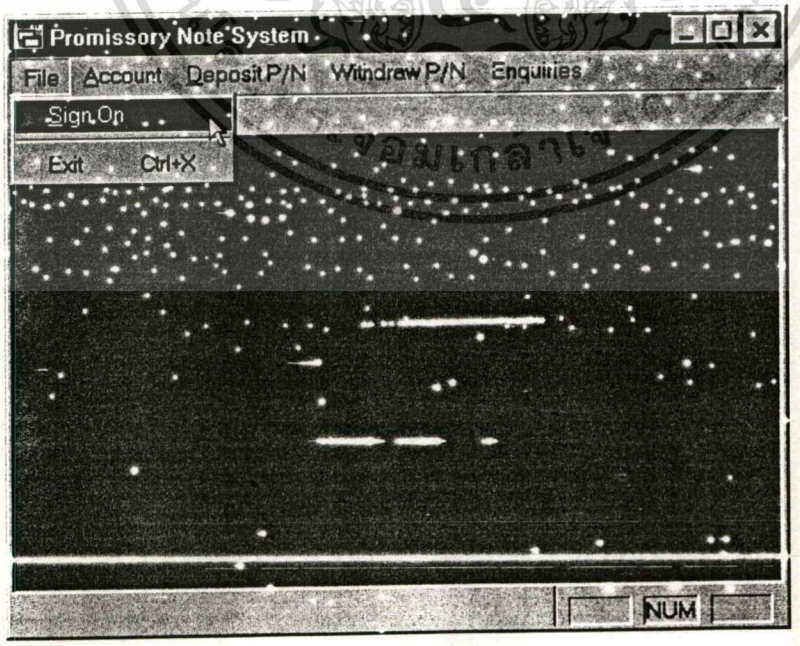
รูปที่ 4.18 หน้าจอ Login เพื่อเข้าสู่ระบบ

การเข้าสู่ระบบ ผู้ใช้ที่เข้าสู่ระบบจะมีสิทธิการทำงานต่างกันไป สิทธิที่ว่าก็คือ Maker และ Checker นั่นเอง



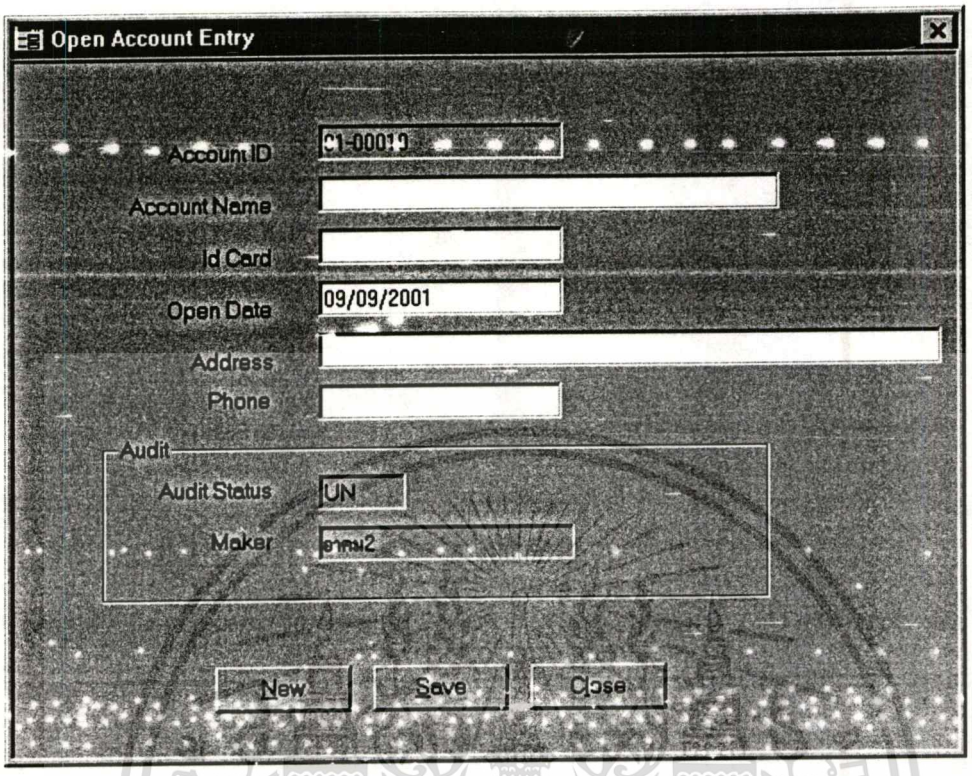
รูปที่ 4.23 หน้าจอหลักของระบบตั๋วสัญญาใช้เงิน

การทำงานของระบบตั๋วสัญญาใช้เงินสามารถใช้เมนู File->Sign On เพื่อทำการเปลี่ยนชื่อผู้ใช้



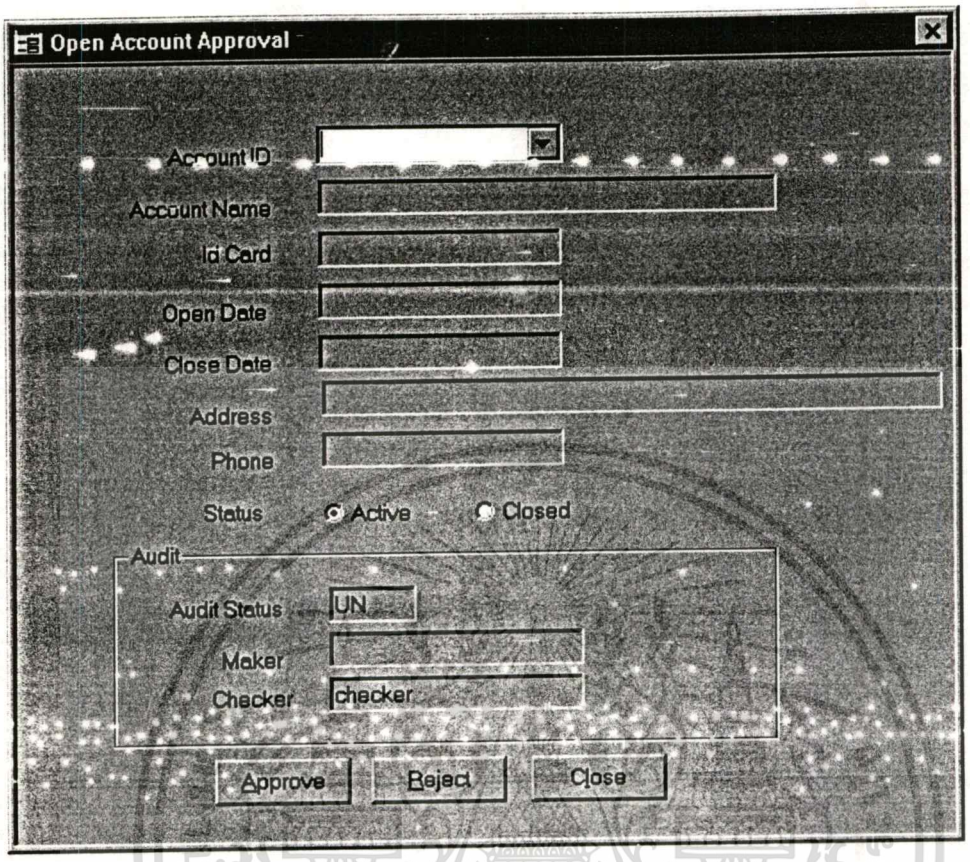
รูปที่ 4.24 แสดงเมนู Sign On

งานได้ โดยระบบจะแสดงหน้าจอ login ขึ้นมาเหมือนกับตอนที่เริ่มใช้งานของระบบ



รูปที่ 4.25 แสดงหน้าจอ Open Account Entry

หน้าจอ Open Account Entry ไว้เพื่อให้ Maker บันทึกข้อมูลการเปิดบัญชีของ Customer ซึ่งสามารถใช้ได้โดยผ่านเมนู Account->Open Account Entry สำหรับสีเหลืองมีความหมายว่า ผู้ใช้จะต้องกรอกข้อมูลใน field นั้น หากผู้ใช้ไม่กรอกแล้ว ระบบจะแสดงข้อความเตือน เมื่อผู้ใช้กดปุ่ม Save เพื่อทำการบันทึกข้อมูล ปุ่ม New ใช้เมื่อต้องการกรอกข้อมูลชุดใหม่ โดยที่ระบบจะสร้างหมายเลขบัญชีให้โดยอัตโนมัติ

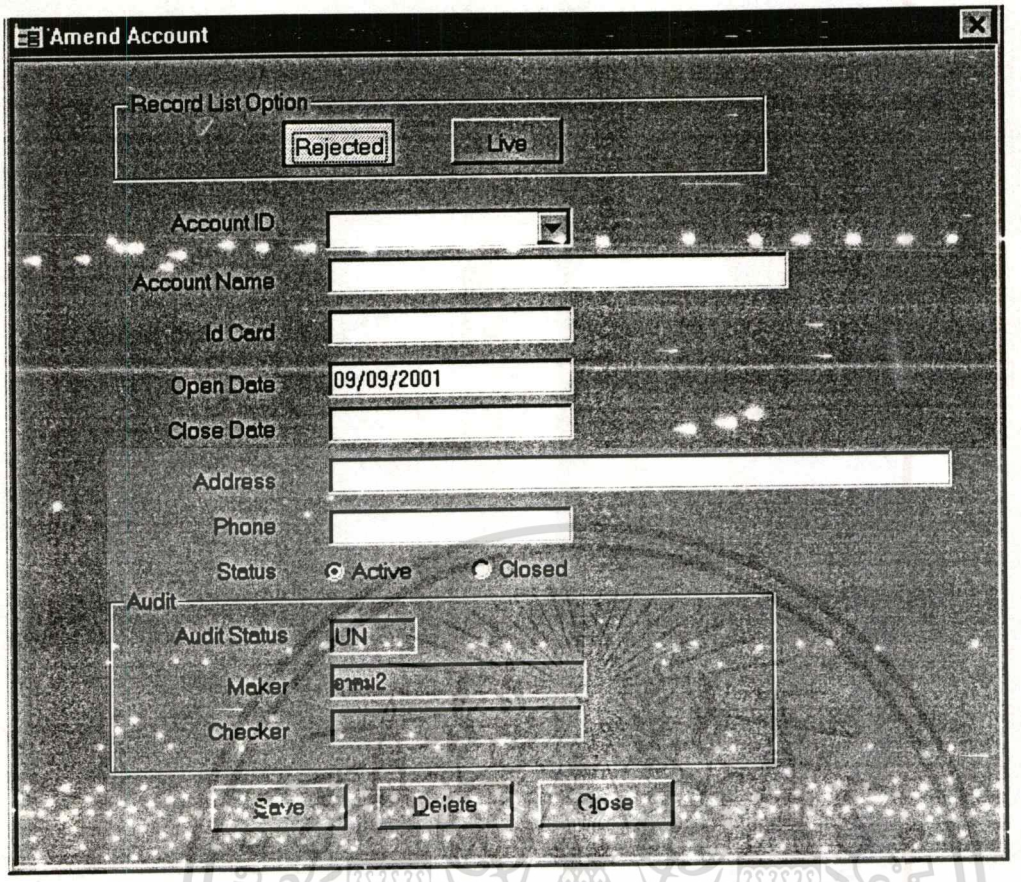


รูปที่ 4.26 แสดงหน้าจอ Open Account Approval

หน้าจอ Open Account Approval ไว้เพื่อให้ Checker ตรวจสอบทำการ approve ข้อมูลการเปิดบัญชีของ Customer ซึ่งสามารถใช้ได้โดยผ่านเมนู Account->Open Account Approval

ปุ่ม Approve ใช้ในกรณีที่ Checker ทำการตรวจสอบข้อมูลและเห็นว่าถูกต้องแล้ว

ปุ่ม Reject ใช้ในกรณีที่ Checker ทำการตรวจสอบข้อมูลและ พบว่าข้อมูลมีความผิดพลาด และจะต้องให้ Maker ทำการแก้ไขข้อมูลใหม่อีกครั้ง



รูปที่ 4.27 แสดงหน้าจอ Amend Account

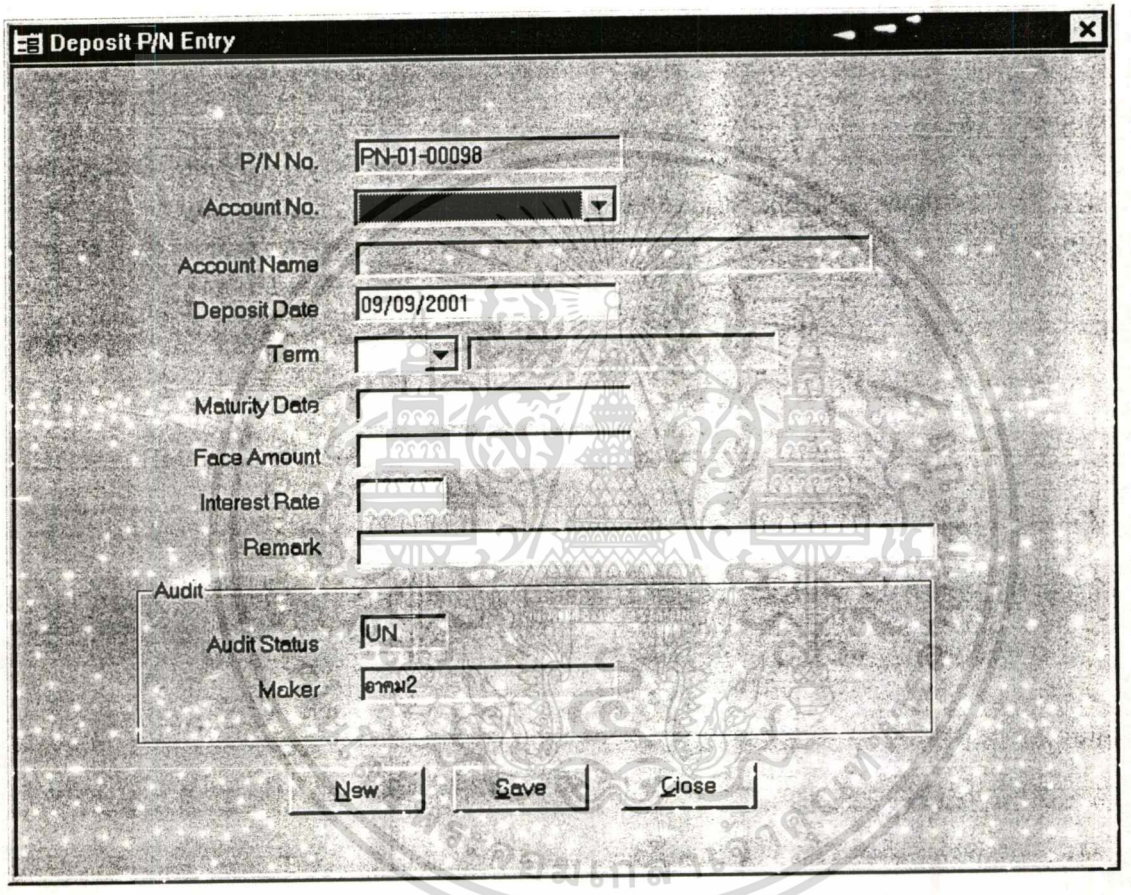
หน้าจอ Amend Account ไว้เพื่อให้ Maker แก้ไขข้อมูลการเปิดบัญชีของ Customer ที่ได้ถูก reject มาจาก Checker ซึ่งสามารถเรียกใช้ได้โดยกดเมนู Account->Amend Account ผู้ใช้สามารถกดปุ่ม Rejected และ Live ซึ่งจะอธิบายได้ดังนี้

Rejected หมายถึง ระบบจะแสดงข้อมูลที่ถูก reject มาจาก Checker เมื่อผู้ใช้เรียก ข้อมูลมาแสดงแล้วผู้ใช้มีสิทธิที่จะกระทำกับข้อมูลได้ 2 อย่างคือ

- แก้ไขข้อมูลให้ถูกต้องแล้วกด Save จะทำให้ข้อมูลนั้นส่งไปรอการ approve ให้กับ Checker
- ลบข้อมูล ใช้ในกรณีที่ผู้ใช้ไม่ต้องการแก้ไขข้อมูลที่ถูก reject มา และต้องการลบข้อมูลนั้นออกไปจากระบบ

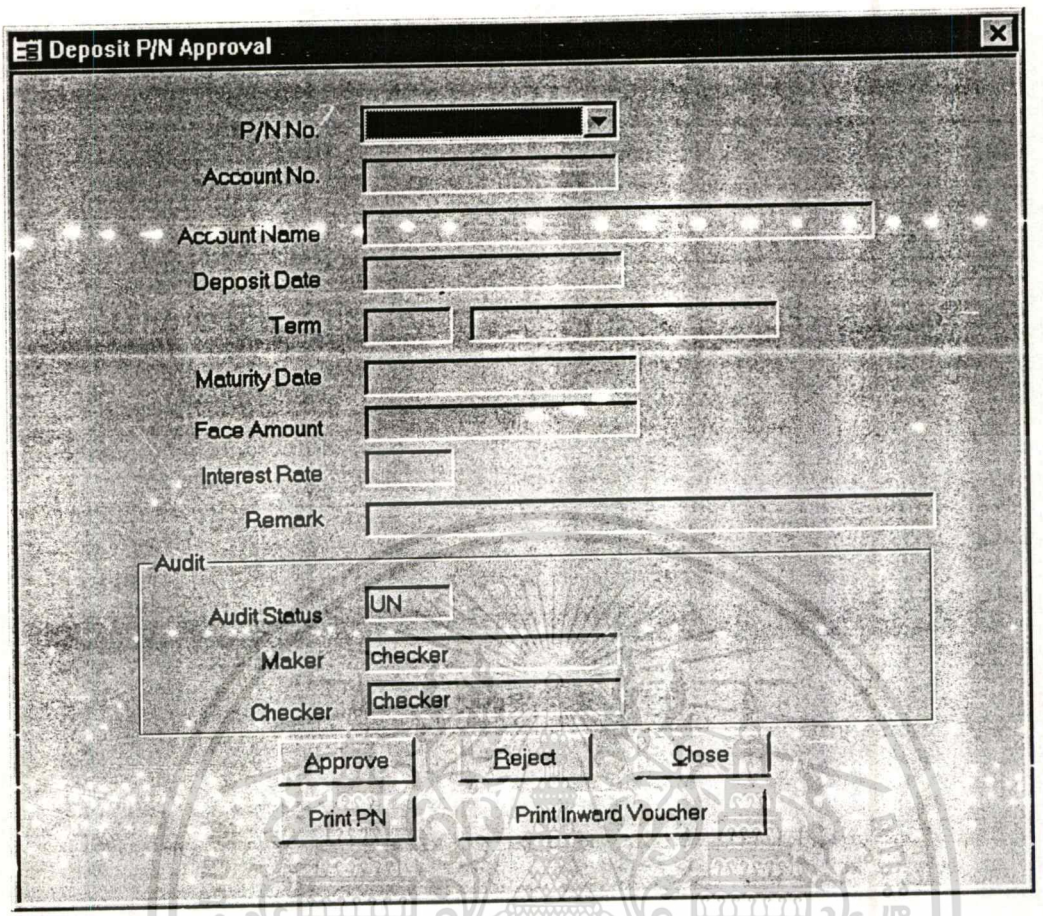
Live หมายถึง ระบบจะแสดงข้อมูลที่มีสถานะ Live ซึ่ง ผู้ใช้มีสิทธิที่จะกระทำกับข้อมูลได้ 2 อย่างคือ

- แก้ไขข้อมูล ในกรณีที่ลูกค้าอาจมีการเปลี่ยนแปลงข้อมูลบางอย่าง ก็สามารถทำได้โดยการแก้ไขข้อมูลที่เปลี่ยนแปลงให้ถูกต้องแล้วกด Save จะทำให้ข้อมูลนั้นส่งไปรอการ approve ให้กับ Checker
- ลบข้อมูล ในกรณีที่ต้องการลบข้อมูลนั้นออกไปจากระบบ ให้กดปุ่ม Delete จะทำให้ข้อมูลนั้นส่งไปรอการ approve ให้กับ Checker



รูปที่ 4.28 แสดงหน้าจอ Deposit P/N Entry

Deposit P/N Entry หน้าจอ นี้มีไว้เพื่อให้ Maker บันทึกข้อมูลการฝากเงินของ Customer ซึ่งสามารถเรียกใช้ได้โดยกดเมนู Deposit P/N-> Deposit P/N Entry ปุ่ม New ใช้เมื่อต้องการกรอกข้อมูลชุดใหม่ โดยที่ระบบจะสร้างหมายเลขตั๋วสัญญาให้โดยอัตโนมัติ



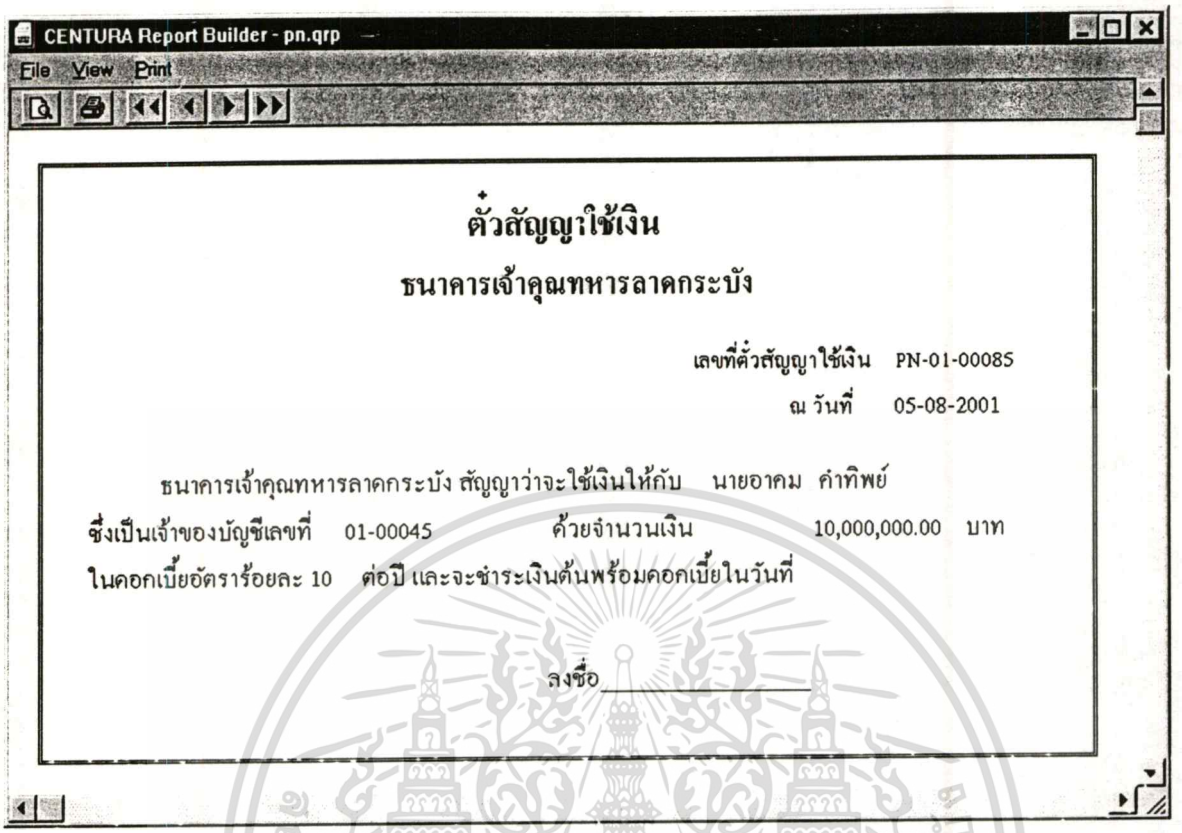
รูปที่ 4.19 แสดงหน้าจอ Deposit P/N Approval

หน้าจอ Deposit P/N Approval มีไว้เพื่อให้ Checker ตรวจสอบทำการ approve ข้อมูลการฝากตั๋วสัญญาใช้เงินของ Customer ซึ่งสามารถใช้ได้โดยผ่านเมนู Deposit P/N->Deposit P/N Approval

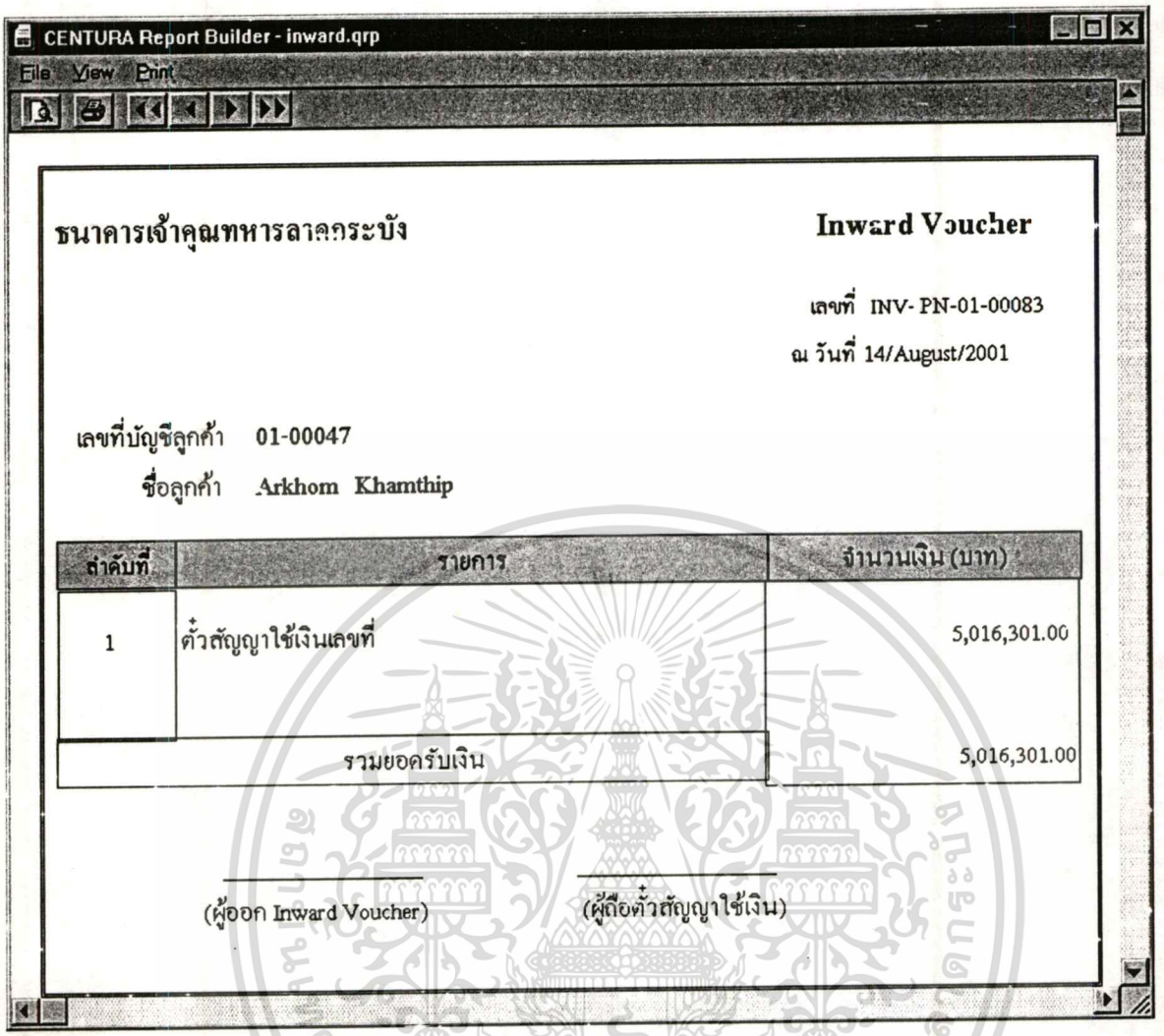
ปุ่ม Approve ใช้ในกรณีที่ Checker ทำการตรวจสอบข้อมูลและเห็นว่าถูกต้องแล้ว

ปุ่ม Reject ใช้ในกรณีที่ Checker ทำการตรวจสอบข้อมูลและพบว่าข้อมูลมีความผิดพลาดและจะต้องให้ Maker ทำการแก้ไขข้อมูลใหม่อีกครั้ง

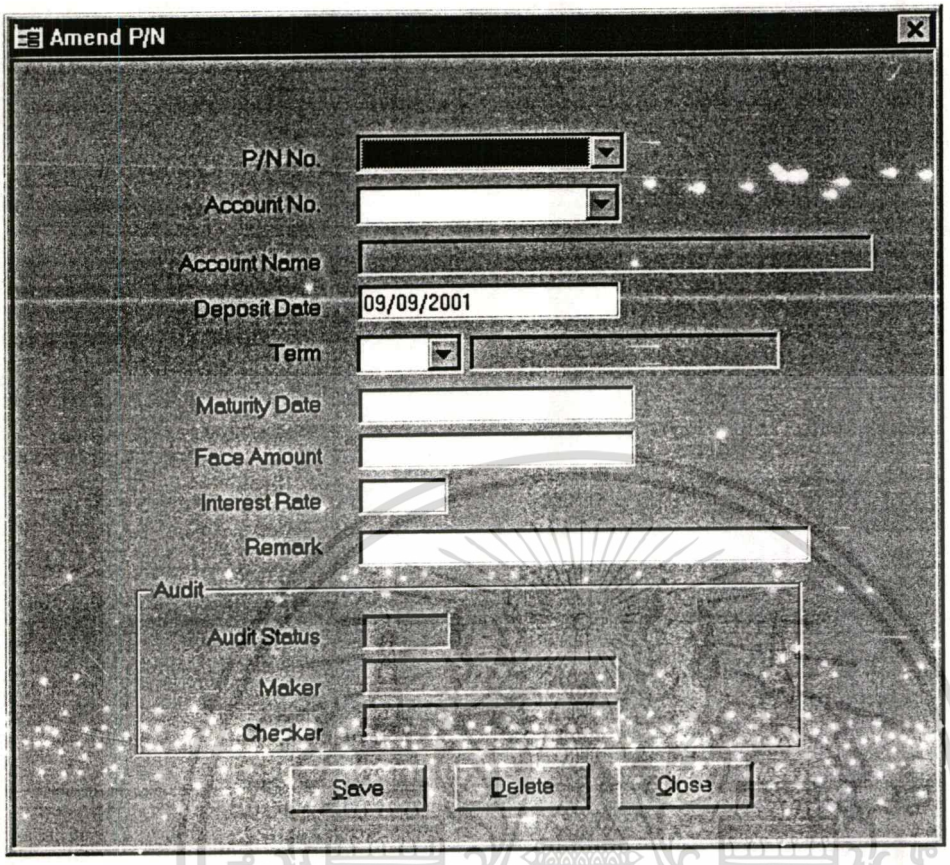
ในการทำงานเมื่อ Checker ทำการ approve ข้อมูลตั๋วสัญญาใช้เงิน ระบบจะแสดงหน้าจอสำหรับพิมพ์ตั๋วสัญญาใช้เงินและใบ Inward Voucher ขึ้นมาให้โดยอัตโนมัติ



รูปที่ 4.30 แสดงหน้าจอสำหรับพิมพ์ตั๋วสัญญาใช้เงิน



รูปที่ 4.31 แสดงหน้าจอสำหรับพิมพ์ใบ Inward Voucher

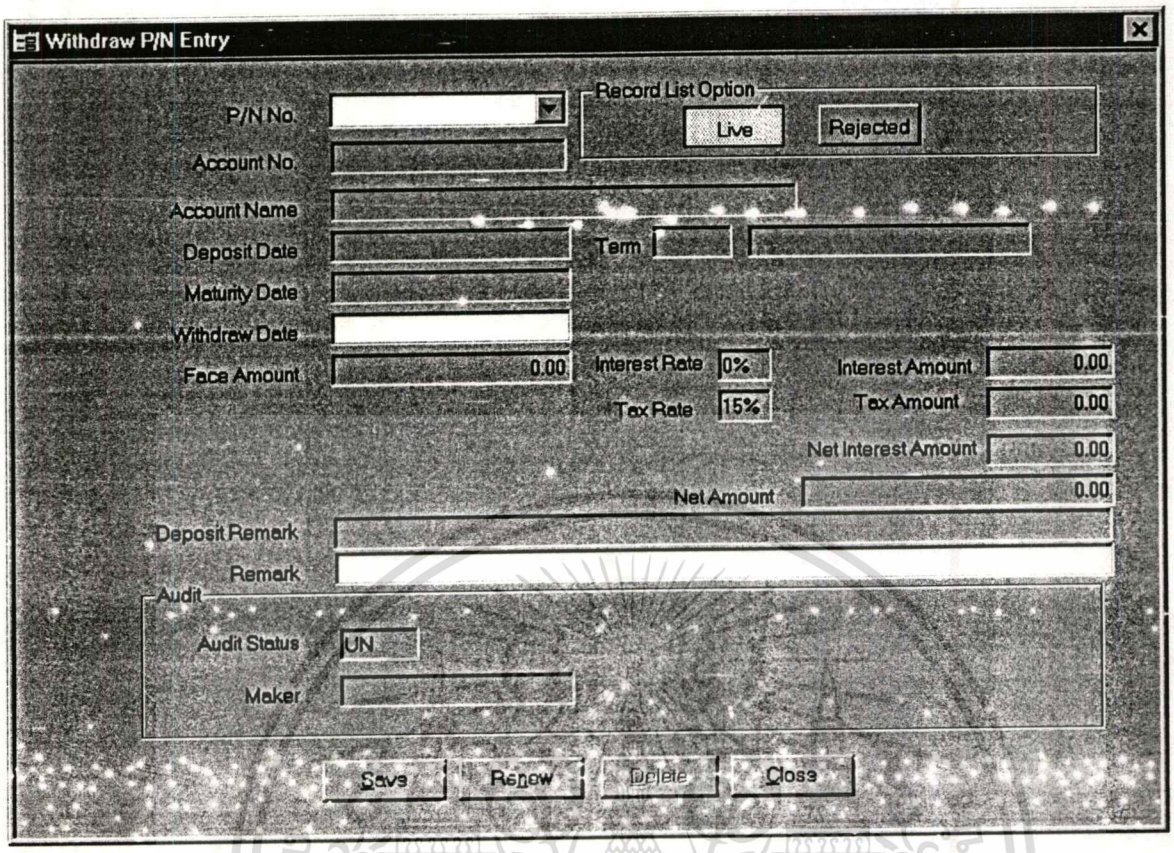


รูปที่ 4.20 แสดงหน้าจอ Amend P/N

หน้าจอ Amend P/N ไว้เพื่อให้ Maker แก้ไขข้อมูลการฝากตั๋วของ Customer ที่ได้ถูก Reject มาจาก Checker ซึ่งสามารถเรียกใช้ได้โดยกดเมนู Deposit P/N->Amend P/N

ผู้ใช้มีสิทธิที่จะกระทำกับข้อมูลได้ 2 อย่างคือ

- แก้ไขข้อมูลให้ถูกต้องแล้วกด Save จะทำให้ข้อมูลนั้นส่งไปรอการ approve ให้กับ Checker
- ลบข้อมูล ใช้ในกรณีที่ผู้ใช้ไม่ต้องการแก้ไขข้อมูลที่ถูก reject มา

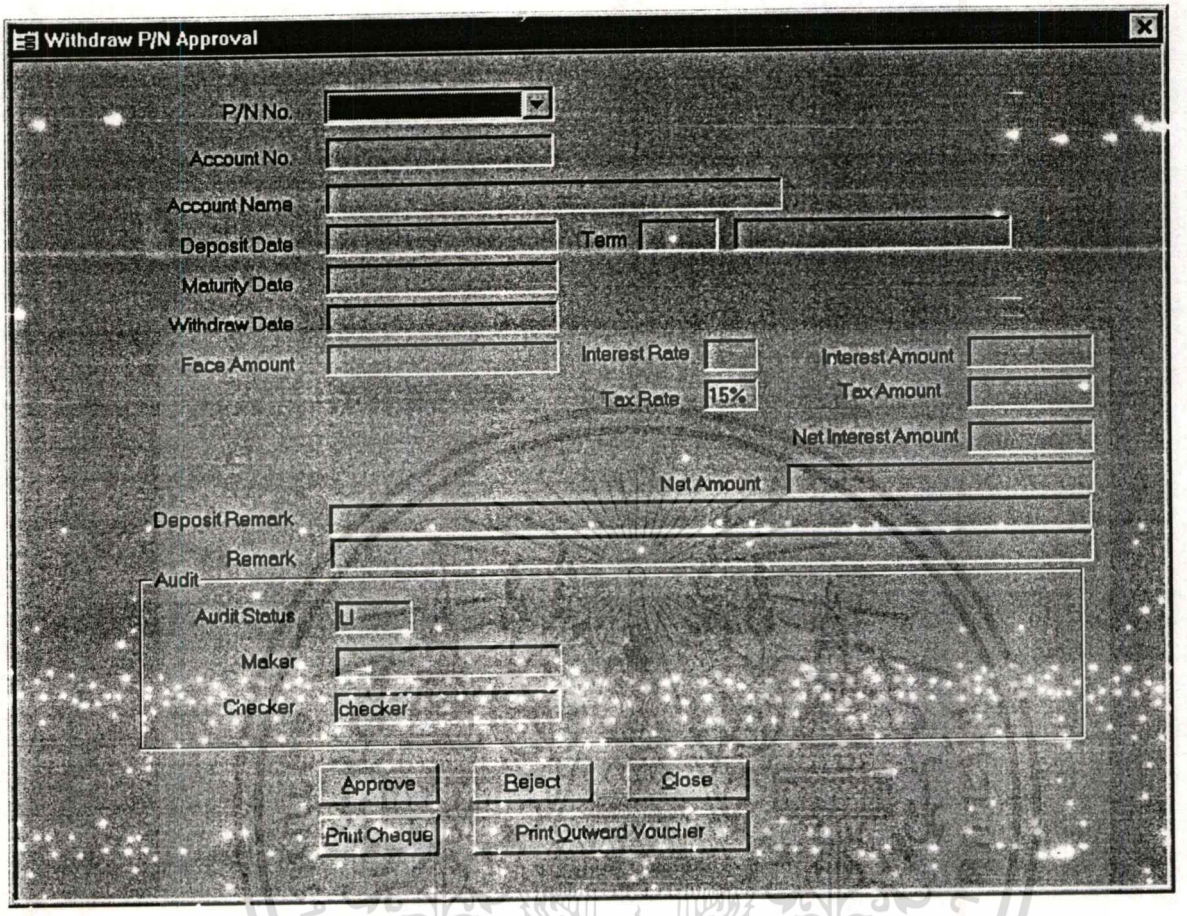


รูปที่ 4.33 แสดงหน้าจอ Withdraw P/N Entry

หน้าจอ Withdraw P/N Entry มีไว้เพื่อให้ Makerทำการบันทึกข้อมูลการถอนตั๋วสัญญาใช้เงินของ Customer ซึ่งสามารถใช้ได้โดยผ่านเมนู Withdraw P/N->Withdraw P/N Entry สำหรับหน้าจอนี้สามารถทำการแก้ไขข้อมูลการถอนที่ถูก Reject มาได้ด้วยการกดปุ่ม Rejected ผู้ใช้มีสิทธิ์ที่จะกระทำกับข้อมูลได้ 2 อย่างคือ

- แก้ไขข้อมูลให้ถูกต้องแล้วกด Save จะทำให้ข้อมูลนั้นส่งไปรอการ approve ให้กับ Checker
- ลบข้อมูล ใช้ในกรณีที่ผู้ใช้ไม่ต้องการแก้ไขข้อมูลที่ถูก reject มา และเมื่อกดปุ่ม Delete แล้วตั๋วสัญญาใช้เงินนั้นจะกลับเป็นสถานะปกติ

ในกรณีที่ต้องการทำ Renew คือการถอนตั๋วสัญญาใช้เงินที่ครบกำหนดแล้วต้องการฝากต่อด้วยจำนวนเงินที่ได้รับมา ก็สามารถกดปุ่ม Renew



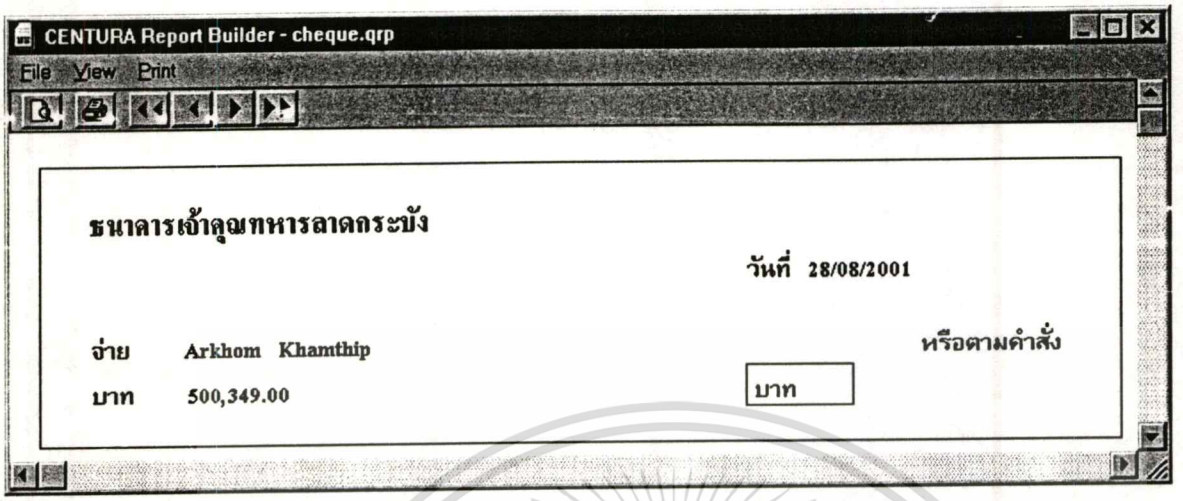
รูปที่ 4.34 แสดงหน้าจอ Withdraw P/N Approval

หน้าจอ Withdraw P/N Approval มีไว้เพื่อให้ Checker ตรวจสอบทำการ approve ข้อมูลการถอนตั๋วสัญญาใช้เงินของ Customer ซึ่งสามารถใช้ได้โดยผ่านเมนู Withdraw P/N->Withdraw P/N Approval ในการทำงานเมื่อ checker ทำการ approve ข้อมูลแล้วระบบจะแสดงหน้าจอเพื่อพิมพ์ Cheque และ Outward Voucher ให้โดยอัตโนมัติ

เมื่อผู้ใช้ทำการเลือกข้อมูลผู้ใช้มีสิทธิที่จะกระทำกับข้อมูลได้ ดังนี้

ปุ่ม Approve ใช้ในกรณีที่ Checker ทำการตรวจสอบข้อมูลและเห็นว่าถูกต้องแล้ว

ปุ่ม Reject ใช้ในกรณีที่ Checker ทำการตรวจสอบข้อมูลและพบว่าข้อมูลมีความผิดพลาดและจะต้องให้ Maker ทำการแก้ไขข้อมูลใหม่อีกครั้ง

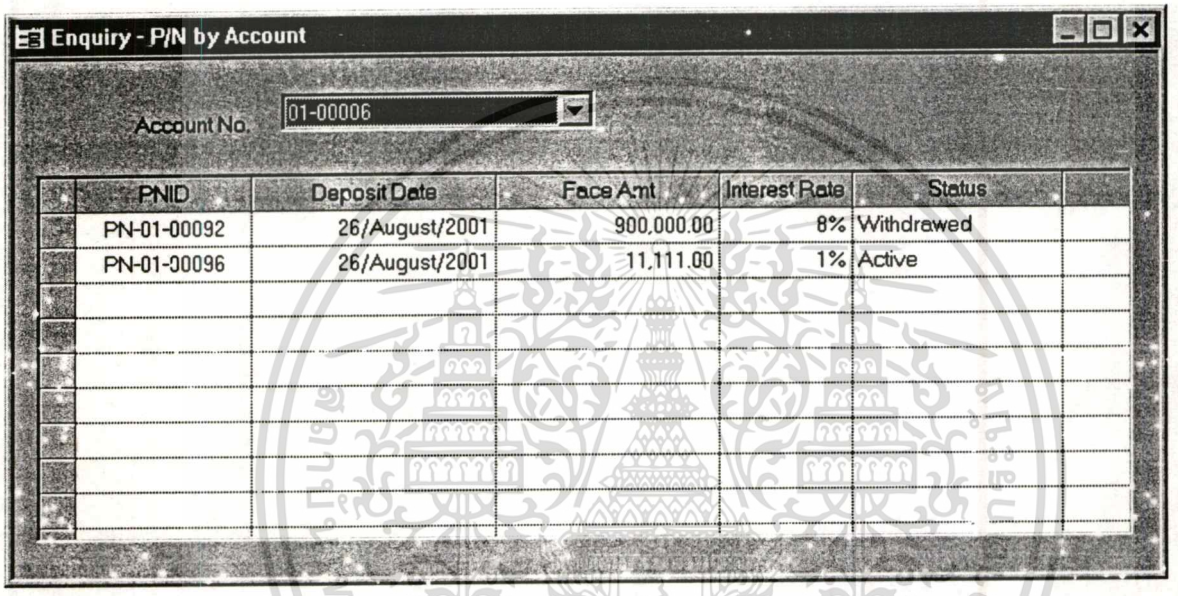


รูปที่ 4.35 แสดงหน้าจอการพิมพ์เช็ค

รูปที่ 4.21 แสดงหน้าจอการพิมพ์ใบ Outward

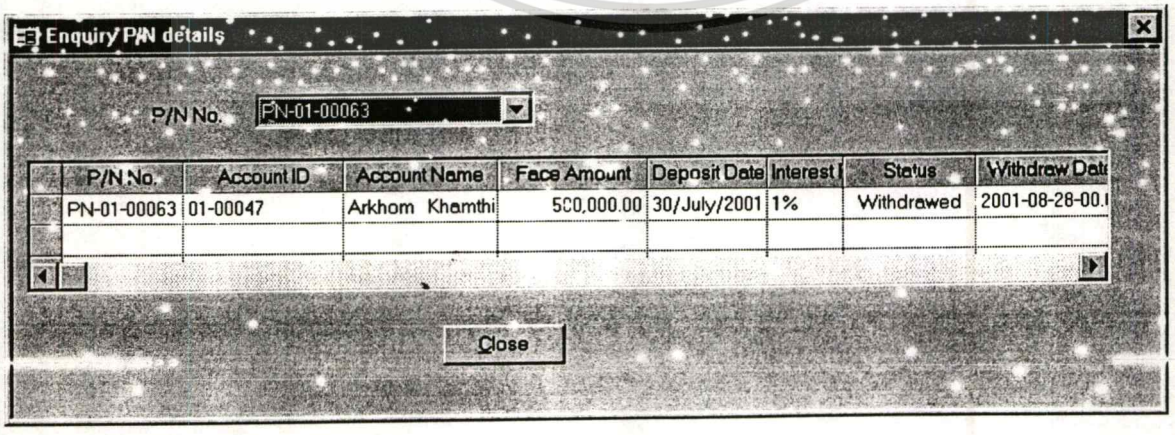
การใช้งานในเมนู Enquiries เป็นการแสดงข้อมูลต่าง ๆ ในระบบตั๋วสัญญาใช้เงิน เมื่อผู้ใช้ต้องการทราบข้อมูลทางด้าน บัญชีลูกค้า, ข้อมูลตั๋วสัญญาใช้เงิน, ข้อมูลแสดงยอดรวมของตั๋วสัญญาใช้เงินประเภท ต่างๆ ก็สามารถเรียกใช้เมนูนี้ได้

หากต้องการทราบรายละเอียดว่าบัญชีแต่ละบัญชีมีการฝาก-ถอน P/N อย่างไรบ้าง ก็สามารถใช้เมนู Enquiries->P/N by Account



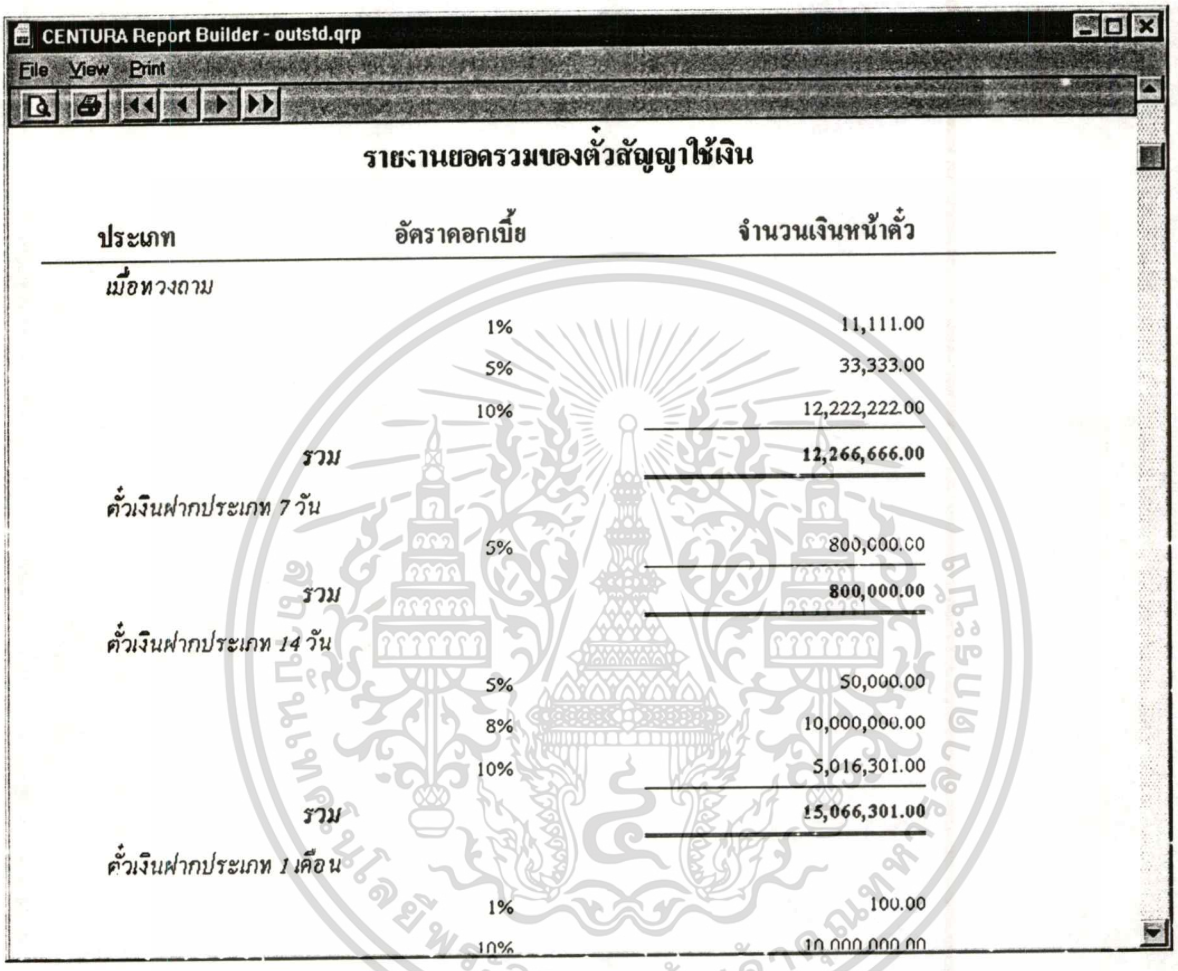
รูปที่ 4.22 แสดงหน้าจอ Enquiry ของ P/N by Account

หากต้องการทราบรายละเอียดข้อมูลของ P/N ก็สามารถใช้เมนู Enquiries->P/N Details



รูปที่ 4.38 แสดงหน้าจอ Enquiry แสดง P/N

หากหน้าจอแสดงยอกรวมของตั๋วสัญญาใช้เงินทั้งหมดในระบบโดยแยกแสดงตามประเภทของตั๋วสัญญาใช้เงิน ซึ่งสามารถใช้เมนู Enquiries->Outstanding



รูปที่ 4.39 แสดงยอดรวมของตั๋วสัญญาใช้เงิน

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 สรุปผลการทดลอง

จากการที่ได้ออกแบบระบบในเชิงวัตถุแล้ว นำไปเขียนโปรแกรมเพื่อให้โปรแกรมทำงานตามที่ได้ออกแบบเชิงวัตถุ แล้วพบว่าได้ผลดีอย่างยิ่ง โดยในขั้นตอนแรก เราจะสามารถทำความเข้าใจระบบด้วยการใช้ไดอะแกรมของ UML แสดงการทำงานของระบบ ซึ่งการทำความเข้าใจกับระบบว่าระบบมีหน้าที่ทำอะไรบ้างนั้น เราสามารถทำความเข้าใจกับ Use Case Diagram ซึ่งจะแสดงรายละเอียดว่ามี Use Case อะไรบ้าง หลังจากนั้น หากต้องการทราบว่าในแต่ละ Use Case นั้นมีขั้นตอนการทำงานอะไรบ้าง เราสามารถใช้ Activity Diagram แสดงถึงขั้นตอนการทำงานของแต่ละ Use Case ว่าได้เกิดกิจกรรมอะไรขึ้น ถึงตรงนี้หากบุคคลที่เป็น Business Consultant ก็จะสามารถทราบได้ว่าระบบที่สนใจอยู่นั้นทำงานอะไรได้บ้าง สามารถทำงานได้ตามที่ต้องการหรือไม่ หลังจากนั้น เราสามารถใช้ Sequence Diagram และ Class Diagram เพื่อให้นักวิเคราะห์ระบบ และ นักเขียนโปรแกรมสามารถทราบว่า จะต้องเขียนโปรแกรมอย่างไรบ้างเพื่อให้ระบบมีออปเจคครบสมบูรณ์และสามารถทำงานได้ตาม ที่ไดอะแกรมได้แสดงไว้

5.2 ปัญหาที่พบในการทำงานและข้อเสนอแนะแนวทางการพัฒนาระบบ

การออกแบบยังไม่ยืดหยุ่นพอกับการนำการไปใช้งานได้จริง เนื่องจากปัญหาเรื่องเวลาในการพัฒนาโครงการ การพัฒนาโครงการนี้มีวัตถุประสงค์เพื่อนำ UML เข้ามาช่วยในการออกแบบโปรแกรมเชิงวัตถุเพื่อสามารถใช้สัญลักษณ์ต่าง ๆ ใน UML สื่อให้ผู้อ่านไดอะแกรมได้เข้าใจการทำงานของระบบในเชิงวัตถุ และยิ่งไปกว่านั้นยังนำการออกแบบในเชิงวัตถุด้วย UML มาทำการเขียนโปรแกรมเชิงวัตถุให้สามารถทำงานได้ตามที่ได้ออกแบบไว้ ซึ่งโดยการทำงานพื้นฐานแล้วระบบสามารถทำงานได้อย่างดี แต่ที่ดังที่ได้กล่าวไว้ว่าระบบยังไม่ยืดหยุ่นพอที่จะนำไปใช้งานได้จริงได้ก็เพราะว่าในการทำงานในธนาคารหรือสถาบันการเงินก็ดี จะมีความซับซ้อนของการทำงานมากกว่านี้ ดังนั้นหากจะนำไปใช้งานจริงแล้วจะต้องทำการเพิ่มคลาสหรือแอทริบิวให้มากกว่านี้ ข้อสังเกตุอีกข้อหนึ่งก็คือว่าการออกแบบโปรแกรมเชิงวัตถุนี้ได้ใช้ฐานข้อมูลเชิงสัมพันธ์ ซึ่งอาจทำให้เกิดความสับสนว่าทำไมจึงไม่ใช้ฐานข้อมูลเชิงวัตถุ แต่ในการออกแบบเชิงวัตถุนั้น เราจะใช้คลาสเป็นสิ่งที่ใช้บ่งบอกว่าวัตถุมีหน้าที่อย่างไรบ้าง ซึ่งจะดูได้จาก operation ของคลาสนั้น ๆ ซึ่งการทำงานภายใน operation นั้นเราไม่จำเป็นต้องรู้ว่าคลาสนั้นทำงานอย่างไร ภายในคลาสจะเก็บข้อมูลในฐานข้อมูลที่เป็นเชิงวัตถุหรือเชิงสัมพันธ์หรือไม่ก็ไม่ใช่สิ่งสำคัญ แต่สิ่งที่สำคัญคือคลาสนั้นจะต้องสามารถทำงานตามที่ได้ออกแบบไว้ หากเป็นฐานข้อมูลเชิงสัมพันธ์เราจะใช้ภาษา SQL เป็นสิ่งที่ใช้ในการจัดการกับข้อมูลโดยเราจะต้องเขียนภาษา SQL ให้ไป

ทำงานอยู่ใน Operation ของคลาสจึงจะสามารถทำงานได้ แต่กระนั้นท่านอาจารย์อัครินทร์ ผู้ที่เป็นอาจารย์ที่ปรึกษา ได้กรุณาให้คำแนะนำที่เป็นประโยชน์ได้มาก ซึ่งท่านเองก็ได้ให้แง่คิดว่าทำไมการออกแบบใช้แนวคิดเชิงวัตถุแต่ฐานข้อมูลเป็นฐานข้อมูลเชิงสัมพันธ์ ซึ่งเหตุผลที่ไม่นำฐานข้อมูลเชิงวัตถุมาพัฒนาก็เพราะว่าฐานข้อมูลเชิงวัตถุเป็นเทคโนโลยีที่ใหม่ในวงการซอฟต์แวร์ในประเทศไทย ซึ่งยังไม่มีการใช้กันอย่างแพร่หลาย การศึกษาการทำงานของฐานข้อมูลเชิงวัตถุคงจะต้องใช้เวลาอีกสักระยะ และยังไม่สามารถยืนยันได้ว่ามันสามารถสนันสนุนการทำงานของโปรแกรม Centura ที่ใช้เขียนโปรแกรมหรือไม่ ดังกล่าวจึงเป็นผลสรุปของการพัฒนาโครงการของระบบตั๋วสัญญาใช้เงิน

บรรณานุกรม

[1] ดวงดาว จิตสุขปาลพรหม. "การพัฒนาระบบงานเวชระเบียนสำหรับโรงพยาบาลทหารผ่านศึก" วิทยาศาสตร์มหาบัณฑิต. 2542

[2] เพชรี ขุมทรัพย์. **หลักการบริหารการเงิน.** กรุงเทพ : โรงพิมพ์มหาวิทยาลัยธรรมศาสตร์. **2532.**

[3] สมรัฐ เชตนุช. "วาดฝันด้วย UML ตอนที่ 2" **PC Magazine,** พฤษภาคม 2543. หน้า 97-100.

[4] สุชาย ธนวเสถียร และ สมนึก ดีรีโต, **การวิเคราะห์และออกแบบโปรแกรมด้วยวิธี Object-oriented ที่ใช้ภาษา UML.** กรุงเทพ : สถาบันไฟฟ้าและอิเลคทรอนิกส์ ร่วมกับบริษัทซัม ซิสเท็ม จำกัด. 2542.

[5] Grady Booch, James Rambaugh and Ivar Jacobson 1999. **The Unified Modeling Language User Guide.** 3rd Printing. Massachusetts : Addison Wesley Longman.

[6] Martin Fowler, **UML Distilled Seccond Edtion.** 1st Printing. Massachusetts : Addison Wesley Longman, Inc. 1999.

[7] Michale R. Blaha and William J. Premerlani. "Object-Oriented Concepts for Database Design" **Fifth Annual Software Technology Conference.** April 1993.

[8] Terry Quatrani. **Visual Modeling with Rational Rose 2000 and UML. .** Massachusetts : Addison Wesley Longman, Inc. 2000.

ประวัติผู้เขียน

ชื่อผู้เขียน	นายอาคม คำทิพย์
วันเดือนปีเกิด	4 มกราคม พ.ศ. 2519
สถานที่เกิด	จังหวัดน่าน
วุฒิการศึกษาระดับปริญญาตรี	วิทยาศาสตร์บัณฑิต สาขาเทคโนโลยีสารสนเทศ
สถานที่สำเร็จการศึกษา	คณะเทคโนโลยีสารสนเทศ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี
ปีที่สำเร็จการศึกษา	ปีการศึกษา 2541
อาชีพปัจจุบัน	พนักงานบริษัท ตำแหน่ง Programmer Analyst บริษัท ทีมีนอส(ประเทศไทย) จำกัด