ตัวตรวจทานชุดคำสั่งภาษาจาวาเพื่อทวนสอบข้อกำหนดการออกแบบในแผนภาพคลาส

นายคณิษฐ์ จินโต

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2552

JAVA CODE REVIEWER FOR VERIFYING AGAINST DESIGN SPECIFICATIONS IN CLASS DIAGRAM

Mister Kanit Jinto

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science Program in Computer Science

Department of Computer Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 2009

หัวข้อวิทยานิพนธ์	ตัวตรวจทานชุดคำสั่งภาษาจาวาเพื่อทวนสอบข้อกำหนดการออกแบบในแผนภาพคลาส
โดย	นายคณิษฐ์ จินโต
สาขาวิชา	วิทยาศาสตร์คอมพิวเตอร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	ผู้ช่วยศาสตราจารย์ดร.ญาใจ ลิ่มปิยะกรณ์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์
(รองศาสตราจารย์ ดร.บุญสม เลิศหิรัญวงศ์)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.สุกรี สินธุภิญโญ)

..... อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(ผู้ช่วยศาสตราจารย์ ดร.ญาใจ ลิ่มปิยะกรณ์)

..... กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.วิษณุ โคตรจรัส)

..... กรรมการภายนอกมหาวิทยาลัย
(ดร.ภาสกร อภิรักษ์วรพินิต)

คณิษฐ์ จินโต : ตัวตรวจทานชุดคำสั่งภาษาจาวาเพื่อทวนสอบข้อกำหนดการออกแบบในแผนภาพคลาส. (JAVA CODE REVIEWER FOR VERIFYING AGAINST DESIGN SPECIFICATIONS IN CLASS DIAGRAM) อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก: ผศ.ดร.ญาใจ ลิ่มปิยะกรณ์, 121 หน้า.

กระบวนการทวนสอบและตรวจสอบความสมเหตุสมผลมีบทบาทสำคัญในการควบคุมคุณภาพบนหลักการที่ว่า ยิ่งข้อบกพร่องถูกตรวจจับแต่เนิ่นๆ ภาระการแก้ไขงานจะยิ่งลดน้อยลง จากงานวรรณกรรมต่างๆ ได้มีการค้นพบว่า ข้อบกพร่องส่วนมากมักเกิดขึ้นในระยะการออกแบบและการเขียนโค้ด การตรวจจับข้อบกพร่องแบบอัตโนมัติเหล่านี้อาจช่วยบรรเทาปัญหาดังกล่าวได้ ดังนั้น งานวิจัยนี้จึงได้คิดค้นตัวตรวจทานชุดคำสั่งอัตโนมัติเพื่อตรวจสอบแฟ้มรหัสคำสั่งจาวากับการออกแบบเชิงวัตถุที่ปรากฏในแผนภาพคลาสยูเอ็มแอล โดยก่อนกระบวนการตรวจทานจะเริ่มขึ้น แผนภาพคลาสต่างๆ จะถูกแปลงให้อยู่ในรูปแบบเอ็กซ์เอ็มแอล เพื่อให้สามารถสกัดสารสนเทศของคลาสและความสัมพันธ์ระหว่างคลาส สำหรับใช้สร้างรายการตรวจทาน ซึ่งตัวตรวจทานชุดคำสั่งจะใช้ทวนสอบรหัสคำสั่งตามรายการไอเท็มตรวจทานว่า ทุกคลาสที่นิยามไว้ปรากฏในรหัสคำสั่งทั้งหมดหรือไม่ เมท็อดและพารามิเตอร์ที่ถูกห่อหุ้มในโครงสร้างคลาสถูกอิมพลีเมนต์อย่างถูกต้องหรือไม่ ทุกความสัมพันธ์ระหว่างคลาสถูกต้องหรือไม่ ท้ายที่สุด รายงานสรุปจะถูกสร้างขึ้นเพื่อแจ้งผลการตรวจทาน

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต Kanit Jito
สาขาวิชา วิทยาศาสตร์คอมพิวเตอร์ ลายมือชื่ออาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
ปีการศึกษา 2552

5171405721 : MAJOR COMPUTER SCIENCE

KEYWORDS : CODE REVIEW / VERIFICATION / CLASS DIAGRAM

KANIT JINTO: JAVA CODE REVIEWER FOR VERIFYING AGAINST DESIGN SPECIFICATIONS IN CLASS DIAGRAM. THESIS ADVISOR: ASST. PROF. YACHAI LIMPIYAKORN, Ph.D.,121 pp.

Verification and Validation processes play an important role in quality control based on the notion that the earlier defects are detected, the less rework incurs. According to the findings from literature, most of the defects occurred during the design and coding phases. Automatic detection of these defects would alleviate the problem. This research therefore invented an automatic code reviewer to examine Java source files against the object-oriented design described in UML class diagrams. Prior to the review process, the class diagrams are converted into XML format so that the information of classes and relations could be extracted and used to generate the review checklists. The code reviewer will then follow the checklist items to verify whether all defined classes exist in the code, the class structures with encapsulated methods and parameters are correctly implemented, all relations of associated classes are valid. Finally, the summary report will then be generated to notify the results.

Department : Computer Engineering Student's Signature Kanit Jinto

Field of Study : Computer Science Advisor's Signature Y. Limpiyakorn

Academic Year : 2009

กิตติกรรมประกาศ

วิทยานิพนธ์นี้เป็นการศึกษาเรื่อง ตัวตรวจทานชุดคำสั่งภาษาจาวาเพื่อทวนสอบข้อกำหนดการออกแบบในแผนภาพคลาส สามารถสำเร็จลุล่วงได้ด้วยความกรุณาจาก ผู้ช่วยศาสตราจารย์ดร.ญาใจ ลิ่มปิยะกรณ์ (อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก) ที่ได้ให้ความรู้ คำปรึกษา คำแนะนำ ตลอดจนการตรวจสอบและแก้ไขข้อบกพร่องต่างๆ จนกระทั่งเสร็จสมบูรณ์ไปได้ด้วยดี ผู้วิจัยจึงขอกราบขอบพระคุณเป็นอย่างสูงไว้ ณ โอกาสนี้

ขอขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร.สุกรี สินธุภิญโญ ผู้ช่วยศาสตราจารย์ ดร.วิษณุ โคตรจรัส และอาจารย์ ดร.ภาสกร อภิรักษ์วรพินิต กรรมการสอบวิทยานิพนธ์ ที่กรุณาเสียสละเวลาให้คำแนะนำ ตรวจสอบ และแก้ไขวิทยานิพนธ์ฉบับนี้

ขอขอบคุณทุกท่านที่มีส่วนช่วยเหลือในการทำวิทยานิพนธ์ครั้งนี้เสร็จสมบูรณ์ ซึ่งมิได้กล่าวนามในที่นี้ทั้งหมด

ท้ายสุดนี้ หากมีสิ่งใดขาดตกบกพร่องหรือข้อผิดพลาดประการใด ผู้วิจัยขออภัยเป็นอย่างสูงในข้อบกพร่องและความผิดพลาดนั้น และหวังว่าวิทยานิพนธ์ฉบับนี้จะเป็นประโยชน์บ้างไม่มากก็น้อยสำหรับผู้ที่สนใจจะศึกษารายละเอียดต่อไป

สารบัญ

หน้า

สารบัญตาราง

หน้า

สารบัญภาพ

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ปัจจุบันเทคโนโลยีสารสนเทศได้เข้ามามีบทบาทต่อองค์กรมากขึ้น ทำให้ซอฟต์แวร์ที่เป็นตัวผลักดันธุรกิจต่างๆ นั้นจำเป็นต้องมีมาตรฐานและคุณภาพอย่างมาก การได้มาซึ่งซอฟต์แวร์นั้นมีด้วยกันหลากหลายวิธีไม่ว่าจะเป็นการใช้บุคลากรภายในองค์กรเป็นผู้พัฒนาขึ้น การจัดซื้อโปรแกรมสำเร็จรูป การว่าจ้างบริษัทอื่นเพื่อช่วยในการพัฒนาซอฟต์แวร์ หรืออาจเกิดจากความร่วมมือของบุคลากรภายในองค์กร กับการว่าจ้างบริษัทที่รับพัฒนาซอฟต์แวร์ ทั้งนี้ทั้งนั้นการลงทุนในการพัฒนาซอฟต์แวร์ยังขึ้นกับงบประมาณขององค์กรอีกด้วย

การพัฒนาซอฟต์แวร์ไม่ว่าจะด้วยวิธีการใดก็ตาม ควรคำนึงถึงคุณภาพของซอฟต์แวร์ โดยคุณภาพของซอฟต์แวร์นั้นสามารถถูกควบคุม คาดเดา ได้จากความสามารถหรือความมีวุฒิภาวะของกระบวนการที่ใช้ในการพัฒนา ซึ่งการได้มาของซอฟต์แวร์ที่มีคุณภาพจำเป็นต้องควบคุมการพัฒนาระบบตั้งแต่การเก็บความต้องการจากทางลูกค้าจนถึงการดูแลรักษาระบบ

การออกแบบและการเขียนโปรแกรมที่ดีนับว่าเป็นสิ่งที่สำคัญต่อการพัฒนาระบบอย่างมาก โดยการออกแบบระบบที่ดีนั้นควรมีการใช้ Design Patterns, เอกสารที่ใช้ในการออกแบบ เช่น UML Diagrams ต่าง ๆ เพื่อเป็นพิมพ์เขียวอ้างอิงสำหรับการเขียนโปรแกรมในขั้นตอนต่อไป การใช้เอกสารการออกแบบจะช่วยให้นักพัฒนาโปรแกรมสามารถทำความเข้าใจต่อระบบงานง่ายขึ้น มีมุมมองการเขียนโปรแกรมเชิงวัตถุ (OOP) [1] และเขียนโปรแกรมที่มีตรรกะถูกต้องตามข้อกำหนดความต้องการที่ระบุในเอกสารการออกแบบ สำหรับการเขียนโปรแกรมที่มีการใช้ Design Patterns จะช่วยให้โค้ดทำงานได้อย่างมีประสิทธิภาพมากยิ่งขึ้นตลอดจนบำรุงรักษาง่าย

การควบคุมคุณภาพซอฟต์แวร์ให้สามารถทำงานได้ถูกต้องตรงตามข้อกำหนดความต้องการ ควรทำแต่เนิ่นๆ ไม่ควรรอจนถึงขั้นตอนการทดสอบซอฟต์แวร์ ด้วยหลักการ V & V (Verification and Validation) [2] โครงการจึงควรมีการทวนสอบ (Verify) ความถูกต้องตามข้อกำหนดซอฟต์แวร์ด้วยเทคนิคการทบทวน (Review) ที่ระยะไมล์สโตนสำคัญๆ ได้แก่ Software Requirements Specification Reviews, Design Reviews, Code Reviews เป็นต้น โดยกิจกรรมการทบทวนจำเป็นต้องมีการสร้างรายการตรวจทาน (Review Checklist) [3] เพื่อตรวจทานข้อบกพร่อง (defect detection) ที่สำคัญหรือที่พบบ่อย ข้อดีของกิจกรรมการทบทวนคือทำให้สามารถพบและแก้ไขข้อบกพร่องเหล่านั้นทันที ไม่ให้หลุดผ่านไปยังเฟสต่อๆ ไป และยังลดช่องว่างการ Implement ที่ไม่ตรงตาม Specification หรือที่มีการเรียกว่า fault of Ommition, fault

Commisio [4] อีกด้วย ซึ่งจะทำให้โครงการล่าช้าเสียหาย สิ้นเปลืองงบประมาณโดยไม่จำเป็น และส่งผลถึงคุณภาพซอฟต์แวร์ได้ในที่สุด

1.2 วัตถุประสงค์ของการวิจัย

งานวิจัยนี้จะมุ่งเน้นการตรวจจับ และลดจำนวน Defect ที่เกิดขึ้นในช่วงขั้นตอน Design Phase จนถึงขั้น Implementation Phase ซึ่งในขั้นตอนเหล่านี้ เป็นส่วนที่ทำให้เกิด Defect มากที่สุด ทำให้ส่งผลกระทบต่อคุณภาพซอฟต์แวร์และเป็นปัจจัยหลักที่ทำให้ส่งมอบโปรเจคล่าช้าในที่สุด

งานวิจัยนี้มีจุดประสงค์เพื่อออกแบบและพัฒนาซอฟต์แวร์ตรวจทานชุดคำสั่งโปรแกรมภาษาจาวาที่สามารถตรวจสอบความถูกต้องของโปรแกรมตามข้อกำหนดความต้องการที่สกัดได้จากแผนภาพคลาสของยูเอ็มแอลร่วมกับ Design patterns ที่กำหนดโดยนักวิเคราะห์ระบบ ซึ่งการตรวจทานชุดคำสั่งแบบอัตโนมัตินี้จะช่วยลดเวลา แรงงานที่ใช้ในการควบคุมคุณภาพซอฟต์แวร์

1.3 ขอบเขตของการวิจัย

ระบบที่พัฒนาขึ้นสามารถตรวจทานชุดคำสั่งโปรแกรมที่เขียนด้วยภาษาจาวา (Java Programming Language) โดยสามารถตรวจทานความถูกต้องโดยตรวจทานรูปแบบการเขียนโปรแกรมที่เป็นช่องโหว่ และไม่เป็นไปตามข้อกำหนดในแผนภาพคลาสที่นักวิเคราะห์ระบบออกแบบไว้ รวมทั้งสามารถตรวจทาน design pattern ได้อย่างน้อย 6 แบบ

ในส่วนของการออกแบบยูเอ็มแอลและการสร้างข้อมูลเอ็กซ์เอ็มแอล จะใช้เครื่องมือ Rational Rose

ระบบตัวตรวจทานชุดคำสั่งภาษาจาวา สามารถแบ่งกลุ่มผู้ใช้งานระบบที่เกี่ยวข้องออกเป็น 2 กลุ่ม คือ

1. ผู้ดูแลระบบหรือนักวิเคราะห์ระบบ
    - 1.1 สามารถเข้าสู่ระบบผู้ดูแลระบบได้
    - 1.2 สามารถเพิ่ม ลบ และแก้ไขข้อมูลพื้นฐานของระบบได้
        - 1.2.1 การจัดการข้อมูลความสัมพันธ์ของ Class
        - 1.2.2 การจัดการข้อมูลระดับการเข้าถึง ของ Class
        - 1.2.3 การจัดการข้อมูลวัตถุประสงค์ของ Design Pattern
        - 1.2.4 การจัดการข้อมูลประเภทการตรวจสอบของระบบ

1.2.5 การจัดการข้อมูลสิทธ์การใช้งานระบบ

1.3 สามารถเพิ่ม ลบ และแก้ไขข้อมูลพนักงานโดยผู้ดูแลระบบยังสามารถแบ่งสิทธ์การใช้งานของพนักงานได้อีกด้วย

1.4 สามารถเค้นหา เพิ่ม ลบ และแก้ไขข้อมูลโครงการได้

1.4.1 สามารถจัดวางพนักงานไปยังโครงการที่มีอยู่ในระบบได้

1.4.2 สามารถตรวจสอบจำนวนโครงการที่พนักงานได้รับมอบหมายได้

1.5 สามารถค้นหาเพิ่ม ลบ และแก้ไขข้อมูลDesign Pattern ได้

1.5.1 ผู้ดูแลระบบสามารถตรวจสอบเพิ่ม ลบ และแก้ไขข้อมูล Class ได้

1.5.2 ผู้ดูแลระบบสามารถตรวจสอบ เพิ่ม ลบ และแก้ไขข้อมูล Attribute Class ได้

1.5.3 ผู้ดูแลระบบสามารถตรวจสอบ เพิ่ม ลบ และแก้ไขข้อมูล Method Class

1.5.4 ผู้ดูแลระบบสามารถตรวจสอบ เพิ่ม ลบ และแก้ไขข้อมูล ความสัมพันธ์ระหว่าง Class ได้

1.6 สามารถค้นหาคำร้อง ตรวจสอบ ยอมรับ หรือยกเลิกคำร้องของนักพัฒนาระบบได้ โดยระบบสามารถแจ้งเตือนเมื่อมีการเปลี่ยนแปลงข้อมูลหรือสร้างรายการตรวจทานชุดคำสั่งโปรแกรม เพื่อให้ผู้ใช้ระบบรับทราบและดำเนินการต่อไป

1.7 สามารถค้นหารายการตรวจทานชุดคำสั่งเพื่อตรวจสอบ เพิ่ม ลบ และแก้ไขข้อมูลได้ โดยระบบจะทำการตรวจสอบ และบันทึกข้อมูลเอ็กซ์เอ็มแอลลงฐานข้อมูล

1.8 สามารถค้นหารายการชุดคำสั่งโปรแกรมภาษาจาวา เพื่อตรวจสอบ เพิ่ม ลบ และแก้ไขข้อมูลได้

1.9 สามารถค้นหา แสดงผลรายละเอียด ตรวจสอบรายการตรวจทานชุดคำสั่ง กับรายการชุดคำสั่งโปรแกรมภาษาจาวาได้

1.10 พิมพ์รายงานข้อมูลต่างๆ ข้างต้นได้

2. นักพัฒนาระบบ

2.1 สามารถเข้าสู่ระบบนักพัฒนาระบบได้

2.2 สามารถตรวจสอบ และแก้ไขข้อมูลส่วนตัวได้

2.3 สามารถศึกษาข้อมูลรูปแบบ โครงสร้าง Design Pattern ผ่านระบบได้

2.4 สามารถค้นหา ตรวจสอบโครงการที่ตนเองได้รับมอบหมายได้

2.5 สามารถทำการร้องขอ และตรวจสอบสถานะการร้องขอ จากระบบได้ โดยนักพัฒนาระบบจำเป็นต้องจัดเก็บชุดคำสั่งโปรแกรมภาษาจาวาเรียบร้อยก่อน ถึงจะทำการร้องขอให้ตรวจสอบข้อมูลได้

2.6 สามารถตรวจสอบรายละเอียดรายการผิดปกติของตนเองได้

2.7 สามารถพิมพ์รายงานข้อมูลต่างๆ ข้างต้นได้

1.4 คำจำกัดความที่ใช้ในการวิจัย

“Fault of omission” [4] หมายถึง โปรแกรม specification มีการระบุฟังก์ชันต่าง ๆ เอาไว้อย่างชัดเจน แต่นักพัฒนาโปรแกรม ยังพัฒนาฟังก์ชันต่าง ๆ ที่มีการระบุไว้ไม่ครบถ้วน

“Fault of commission” [4] หมายถึง นักพัฒนาโปรแกรม มีการพัฒนาโปรแกรมไม่ตรงตามขอบเขตที่ระบุเอาไว้ใน Specification

1.5 ประโยชน์ที่คาดว่าจะได้รับ

ระบบการตรวจทานชุดคำสั่งโปรแกรมแบบอัตโนมัติที่พัฒนาขึ้น สามารถช่วยให้การควบคุมคุณภาพซอฟต์แวร์มีประสิทธิภาพมากขึ้น โดยที่ไม่จำเป็นต้องรอถึงขั้นตอนการทดสอบ รวมทั้งช่วยลดเวลาและแรงงานที่ใช้ในโครงการซอฟต์แวร์

1.6 วิธีดำเนินการวิจัย

1. ศึกษาและทำความเข้าใจในหลักการ ข้อกำหนด และรายละเอียดอื่นๆ ของมาตรฐานการพัฒนาโปรแกรมเชิงวัตถุ

2. ศึกษาและทำความเข้าใจในหลักการ ข้อกำหนด และรายละเอียดอื่นๆ ของมาตรฐานการสร้างชุดข้อมูลตรวจทาน

3. ศึกษาและทำความเข้าใจในหลักการ ข้อกำหนด และรายละเอียดอื่นๆ ของมาตรฐานการออกแบบ Model Class Diagram, Design Pattern

4. ศึกษาและทำความเข้าใจในหลักการ ข้อกำหนด และรายละเอียดอื่นๆ ของมาตรฐาน IEEE 1208-1997 Standard for Software Reviews

5. ศึกษางานวิจัยที่เกี่ยวข้อง

6. ออกแบบ และพัฒนาระบบ

7. ทดสอบและประเมินผลระบบที่พัฒนา

8. สรุปผลการวิจัยและข้อเสนอแนะ

9. ตีพิมพ์บทความวิชาการ

10. จัดทำวิทยานิพนธ์

1.7 ลำดับขั้นตอนในการเสนอผลการวิจัย

วิทยานิพนธ์นี้แบ่งเนื้อหาทั้งหมดออกเป็น 6 บทดังต่อไปนี้ บทที่ 1 เป็นบทนำซึ่งกล่าวถึงความเป็นมาและความสำคัญของปัญหา วัตถุประสงค์ของการวิจัย รวมถึงประโยชน์ที่คาดว่าจะได้รับ บทที่ 2 กล่าวถึงแนวคิดทฤษฎีและงานวิจัยที่เกี่ยวข้อง บทที่ 3 กล่าวถึงวิธีการออกแบบขั้นตอนการดำเนินงาน บทที่ 4 กล่าวถึงการพัฒนาเครื่องมือ บทที่ 5 กล่าวถึงการทดสอบระบบ บทที่ 6 กล่าวถึงการสรุปผลและข้อเสนอแนะ

1.8 ผลงานที่ตีพิมพ์จากวิทยานิพนธ์

ส่วนหนึ่งของวิทยานิพนธ์นี้ได้รับการตีพิมพ์เป็นบทความวิจัยในหัวข้อเรื่อง “Java Code Reviewer for Verifying Object-Oriented Design in Class Diagrams” โดย Kanit Jinto, and Yachai Limpiyakorn ในวารสารโครงการ IEEE International Conference on Information Management and Engineering (IEEE ICIME 2010) เฉินตู สาธารณรัฐประชาชนจีน ระหว่างวันที่ 16 – 18 เมษายน 2553

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

งานวิจัยนี้ ผู้วิจัยได้ค้นคว้าศึกษาเอกสาร แหล่งความรู้ทางอินเตอร์เน็ต งานวิจัย รวมทั้งแนวคิดทฤษฎีต่างๆ ที่เกี่ยวข้อง ได้ดังนี้

2.1.1 การเขียนโปรแกรมเชิงวัตถุ และยูเอ็มแอล (UML - Unified Modeling Language)

2.1.1.1 Grady Booch

ซึ่งเป็นผู้นำแนวคิดแบบ Booch method ซึ่งเป็นวิธีการที่มีชื่อเสียงมาก มี Diagram จำนวนมากสำหรับใช้งาน แต่มีข้อเสียคือมีมากเกินความจำเป็น และยุ่งยากมากในการวาด diagram ด้วยมือ แนวความคิดของ Booch จะทำการวิเคราะห์ทั้งแบบ Micro - และ Micro Development และอยู่บนพื้นฐานของการพัฒนาระบบงานแบบ Iteration and Incremental Process

2.1.1.2 Jame Rumbaugh

Object modeling Techniques (OMT) แนวความคิดนี้ถูกพัฒนาขึ้นที่ General Eletric ซึ่งเป็นที่ทำงานเดิมของ Jame Rumbaugh ประกอบด้วยโมเดลจำนวนมาก ครอบคลุมถึง Object Model, Dynamic Model, Functional Model and Use-case Model

2.1.1.3 Ivar Jacobson

Object Oriented Software Engineer (OOSE) [1] เป็นรูปแบบวิธีการทำงานที่เน้น Requirement ด้วย มีพื้นฐานการทำงานอยู่บน Use-Case Model ซึ่ง Use-Case Model นี้จะถูกใช้ตลอดทุกระยะในการพัฒนาระบบงาน

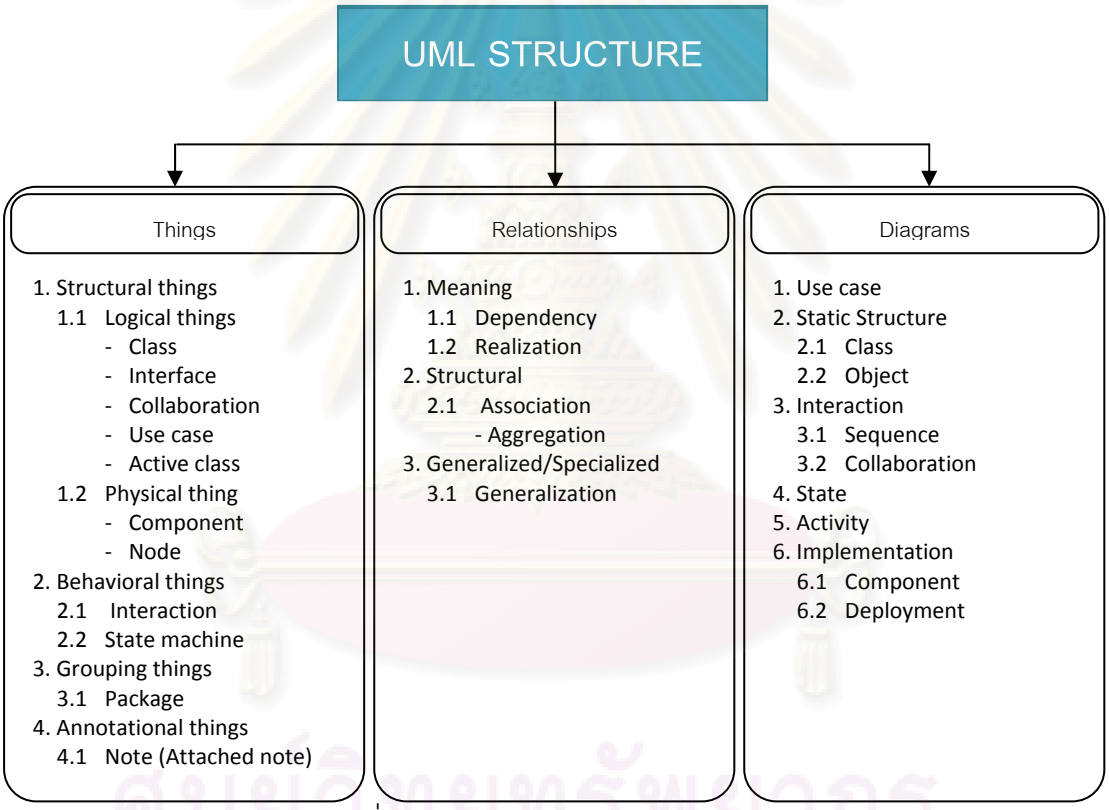
UML (Unified Modeling Language) [5] เป็นเครื่องมือใหม่ที่ได้รับการยอมรับเพิ่มขึ้นตลอดเวลา เริ่มประยุกต์ใช้กับระบบงานมากขึ้น เพราะเป็นเครื่องมือที่มีความหลากหลายในการแสดงแบบซอฟต์แวร์ เป็นโมเดลมาตรฐานที่ใช้หลักการออกแบบ OOP (Object Oriented Programming) รูปแบบของภาษามี Notation เป็นสัญลักษณ์สำหรับสื่อความหมาย มีกฎระเบียบที่มีความหมายต่อการเขียนโปรแกรม (Coding) ดังนั้นการใช้ UML จะต้องทราบความหมายของ Notation เช่น generalize, association, dependency, class และ package สิ่งเหล่านี้มีความจำเป็นต่อการตีความการออกแบบ ก่อนนำไป Implement ระบบงานจริง

UML ประกอบไปด้วย 3 ส่วนหลักคือ Things, Relationships และ Diagrams แสดงดังรูปที่ 1 [5]

- ส่วน Things แบ่ง 4 ประเภท คือ Structural things, Behavioral things, Group things และ An notational things

- ส่วน Relationships แบ่งออกเป็น 3 ประเภท คือ Meaning, Structural และ Generalized/Specialized

- ส่วน Diagrams แบ่งออกเป็น 6 ประเภท คือ Use case, Static structure, Interaction, State, Activity และ Implementation



รูปที่ 1 แผนภาพโครงสร้างยูเอ็มแอล (UML)

ประโยชน์ของยูเอ็มแอล (UML Advantage)

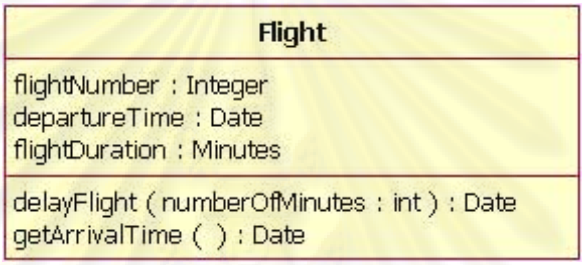
1. วงจรการพัฒนาที่สั้นที่สุด (Shortest Development life cycle)
2. เพิ่มผลผลิต (Increase productivity)
3. ปรับปรุงคุณภาพซอฟต์แวร์ (Improve software quality)
4. สนับสนุนระบบสืบทอดมรดก (Support legacy system)
5. ปรับปรุงการเชื่อมต่อทีมงาน (Improve team connectivity)

2.1.2 UML Class Diagram

Model Class Diagram [5]

Class Diagram คือ แผนภาพที่ใช้แสดง Class และ ความสัมพันธ์ระหว่าง Class ของระบบที่สนใจ (Problem Domain)

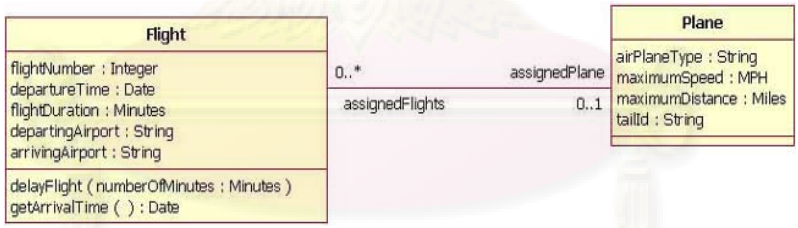
Class ประกอบไปด้วย Class Name ชื่อของ Class, Attributes คือ คุณลักษณะของ Class และ Operations หรือ Methods คือ กิจกรรมที่สามารถกระทำกับObject นั้นๆ ได้ แสดงในรูปที่ 2



รูปที่ 2 ตัวอย่าง Class

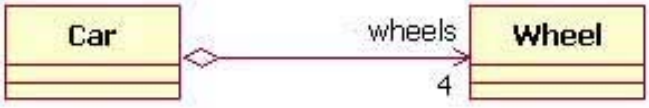
ความสัมพันธ์ระหว่าง Object ประกอบด้วย

1. Association เป็นความสัมพันธ์ระหว่าง Object หรือ Class สองทิศทางแสดงในรูปที่ 3



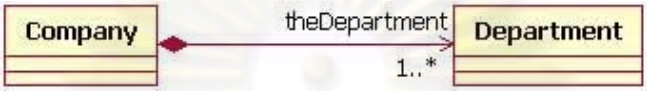
รูปที่ 3 ตัวอย่าง Association relation

2. Aggregation เป็นความสัมพันธ์ระหว่าง Object หรือ Class แบบ “Whole-Part” หรือ “is part of” โดยจะมี Class ที่ใหญ่ที่สุดที่เป็น Object หลัก และมี Class อื่นเป็นส่วนประกอบ แสดงในรูปที่ 4



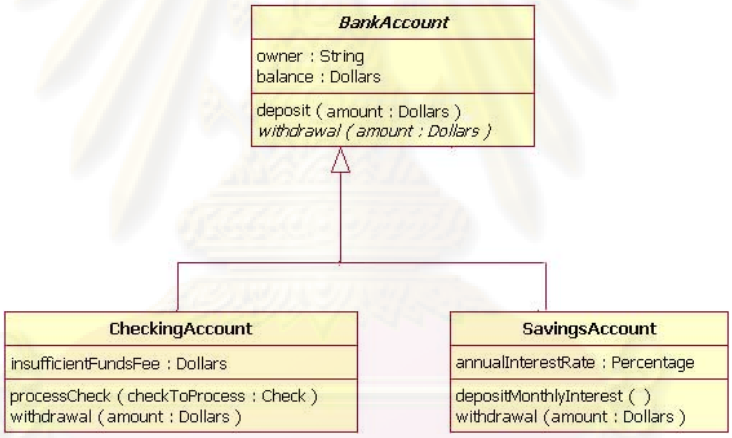
รูปที่ 4 ตัวอย่าง Aggregation relation

3. Composition เป็นความสัมพันธ์ระหว่าง Object หรือ Class แบบขึ้นต่อกันและมีความเกี่ยวข้องกันเสมอ โดยจะมี Class ซึ่งเป็นองค์ประกอบของ Class อื่นที่ใหญ่กว่า เมื่อ Class ที่ใหญ่กว่าถูกทำลาย Class ที่เป็นองค์ประกอบก็จะถูกทำลายไปด้วยแสดงในรูปที่ 5



รูปที่ 5 ตัวอย่าง Composition relation

4. Generalization เป็นความสัมพันธ์ระหว่าง Object หรือ Class ในลักษณะของการสืบทอดคุณสมบัติจาก Class หนึ่ง (Superclass) ไปยังอีก Class หนึ่ง (Subclass)แสดงในรูปที่ 6



รูปที่ 6 ตัวอย่าง Generalization relation

2.1.3 Design Patterns [6]

การออกแบบซอฟต์แวร์เชิงวัตถุ (Object Oriented Design – OOD) [1] แม้ว่ายาก แต่การออบแบบให้ได้มาซึ่งซอฟต์แวร์ที่สามารถนำกลับมาใช้ใหม่ได้อย่างมีประสิทธิภาพกลับยากยิ่งกว่า เพราะในขณะที่ทำการออกแบบรายละเอียด Class ความสัมพันธ์ระหว่าง Class และกิจกรรมระหว่าง Class เพื่อตอบสนองความต้องการในการแก้ปัญหาหนึ่งๆ นั้น ยังต้องคำนึงถึงผลลัพธ์ ที่เป็นกลุ่มของ Class ที่มีความเป็นทั่วไป มากพอที่จะนำไปใช้งานอื่นๆ ต่อไปได้อีก เพื่อตอบสนองต่อความต้องการ Class ที่มีคุณสมบัติ Reusable

หากมิได้คำนึงถึงคุณสมบัติตั้งแต่เริ่มต้นการออกแบบ ก็แทบจะเป็นไปไม่ได้เลยในการจะได้มาซึ่ง Reusable Class ซึ่งบ่อยครั้งที่ต้องย้อนกลับมาออกแบบ Class ใหม่หลังจากที่ออกแบบไปแล้ว เพราะ Class ที่ได้ยังขาดคุณสมบัติ Reusable หรือยังไม่ยืดหยุ่นพอ

การออกแบบที่ให้ได้ Reusable Class มาดีที่สุด คือ การลอกเลียนแบบวิธีการออกแบบที่เคยได้ผลมาแล้วในอดีต หรือวิธีการที่เป็นที่ยอมรับ ซึ่งมีความเสี่ยงน้อยกว่าการคิดค้นด้วยตนเอง ซึ่งการลอกเลียนแบบ และทำซ้ำวิธีเดิมๆ จะช่วยให้เกิดความเชี่ยวชาญ

ซึ่งที่มาของ Design Patterns คือการรวบรวมเอาวิธี สำหรับการออกแบบที่ได้ผลและเป็นที่ยอมรับ ซึ่งวิธีเหล่านี้ไม่จำเพาะเจาะจงสำหรับปัญหาอย่างใดอย่างหนึ่งโดยเฉพาะ กล่าวได้ว่า Design Patterns คือหลักการหรือวิธีการที่สามารถถูกนำไปใช้ซ้ำๆ และรับประกันผลที่จะได้ (หากใช้อย่างได้ถูกต้อง) ไม่ว่าผู้ใช้จะเป็นผู้พัฒนาระบบที่มีประสบการณ์สูง หรือผู้พัฒนาระบบที่มีประสบการณ์น้อย ผลลัพธ์ที่ได้โดยการใช้ Design Patterns เดียวกัน ก็จะออกมาแบบเดียวกัน ซึ่งเป็นการช่วยรักษามาตรฐานของซอฟแวร์ที่จะถูกพัฒนาขึ้น และลดช่องว่างที่เกิดจากการมีประสบการณ์ที่แตกต่างกัน โดย 4 องค์ประกอบของ Design Patterns มีดังต่อไปนี้

1. ชื่อ (Pattern Name) เป็นสิ่งภายนอกใช้เพื่ออ้างอิง Design Pattern ตัวใดตัวหนึ่ง ชื่อของ Design Pattern ตัวหนึ่งจะต้องไม่ซ้ำกับชื่อของ Design Pattern ตัวอื่นๆ และต้องเป็นชื่อสั้นๆ และสื่อความหมายถึงคุณลักษณะของ Design Pattern ตัวนั้นๆ

2. ปัญหา (Problem) เป็นสิ่งที่อธิบายว่า เมื่อใดที่ Design Pattern ตัวนั้นจะถูนำกลับมาใช้งาน ซึ่งการอธิบายปัญหานั้น อาจจะอธิบายได้ในเชิงสถิติ และในเชิงกิจกรรม หรืออาจหมายถึงการอธิบายเงื่อนไขที่ควรจะเป็นก่อนและหลังการใช้ Design Pattern ตัวนั้นๆ

3. วิธีแก้ปัญหา (Solution) เป็นสิ่งที่อธิบายว่า Design Pattern ตัวนั้น จะต้องประกอบขึ้นจาก Class ที่มีโครงสร้าง ความสัมพันธ์อย่างไร มีกิจกรรมใดบ้าง ลำดับของกิจกรรมเป็นอย่างไร และจะนำเอา Pattern ที่ได้ไปใช้อย่างไรเพื่อแก้ปัญหา (Problem) ที่มี

4. ผลที่ตามมา (Consequence) เป็นสิ่งที่จะอธิบายผลที่ตามมาจากการใช้ Design Pattern ตัวหนึ่งๆ ซึ่งจะมีทั้งผลทางบวกและผลทางลบ Consequence เป็นเสมือนข้อมูลให้แก่เราในการจะประเมินและเลือกใช้ Design Pattern ตัวที่เหมาะสม

Gang of Four Design Patterns (GoF Design Patterns) เป็นชุดของ Design Patterns ที่ได้รับการยอมรับสูงในปัจจุบัน โดย GoF Design Patterns จะประกอบไปด้วย Design Pattern ที่มีคุณลักษณะและคุณประโยชน์ที่แตกต่างกันทั้งสิ้น 23 Design Patterns ซึ่งแสดงความสัมพันธ์

ที่มีอยู่ทั้ง 23 ตัวแสดงดังรูปที่ 7 [6] โดย Design Pattern สามารถจัดเป็น 3 หมวด ตามวัตถุประสงค์การใช้งานที่เรียกว่า Design Pattern Catalog แสดงดังตารางที่ 1 [6] ได้แก่

1. Creational Patterns กลุ่มของ Design Patterns ที่ใช้สำหรับแก้ปัญหาในการสร้าง Object โดย Creational Patterns ประกอบด้วย 5 Design Patterns ได้แก่ Abstract Factory, Builder, Prototype และ Singleton

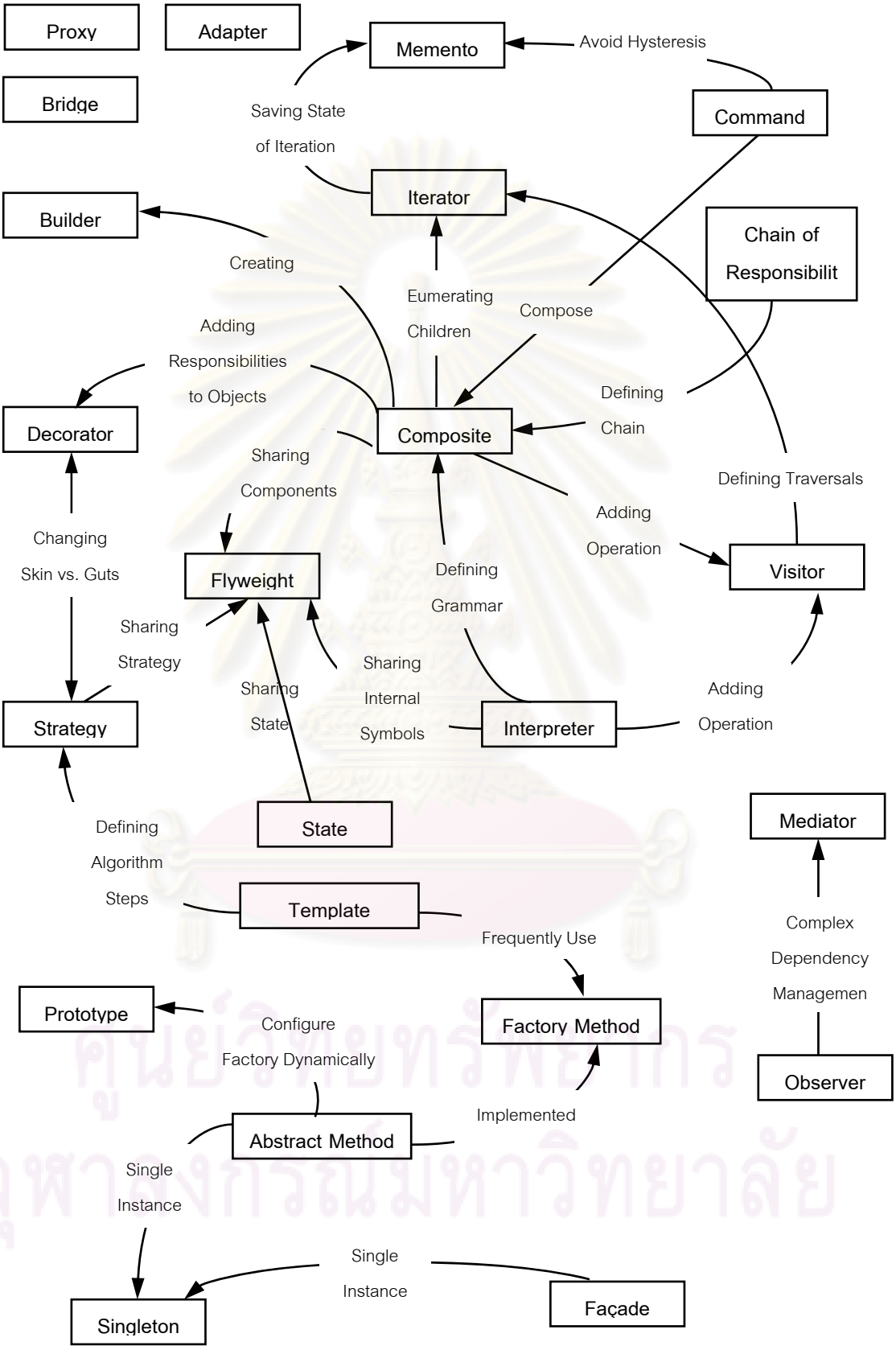
2. Structural Patterns กลุ่มของ Design Patterns ที่ใช้แก้ปัญหาเกี่ยวกับการวางโครงสร้างความสัมพันธ์ระหว่าง Class ซึ่ง Structural Patterns ประกอบด้วย 7 Design Patterns ได้แก่ Adapter, Bridge, Composite, Decorator, Façade, Flyweight และ Proxy

3. Behavioral Patterns กลุ่มของ Design Patterns ที่ใช้แก้ปัญหาเกี่ยวกับพฤติกรรมของ Object และมีปฏิสัมพันธ์ระหว่าง Object ทั้งนี้ Behavioral Patterns ประกอบด้วย 11 Design Patterns ได้แก่ Chain of Responsibility, Command, Iterator, Mediator, Memento, Observer, State, Strategy และ Visitor

แต่ Design Pattern จะมีระดับของ Abstraction ที่เข้าไปเกี่ยวข้องแตกต่างกันไป บาง Design Patterns จะถูกใช้งานในระดับ Class ในขณะที่บาง Design Pattern จะถูกใช้งานในระดับ Object ซึ่งเรียกการแบ่งระดับของ Abstraction ที่เข้าไปเกี่ยวข้องกับ Design Pattern นี้ว่า Design Pattern Scope

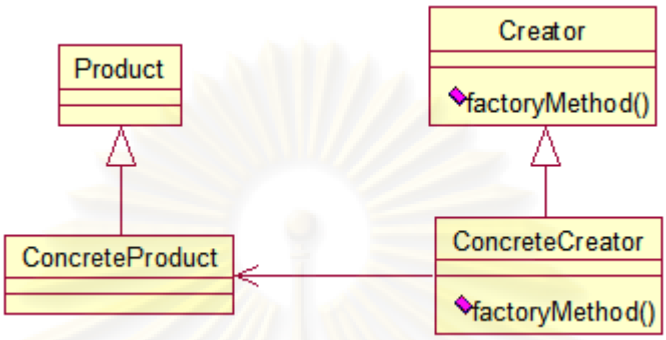
ตารางที่ 1 Design Pattern Catalogs

Design Patterns Catalogs		Purpose		
		Creational	Structural	Behavioral
Scope	Class	Factory Method	Adapter	Interpreter, Template Method
	Object	Abstract Factory, Builder, Prototype, Singleton	Adapter, Bridge, Composite, Decorator, Façade, Flyweight, Proxy	Chain of Responsibility, Command, Iterator, Mediator, Memento, Observer, State, Strategy, Visitor



รูปที่ 7 ภาพรวมของ Design Pattern

ตัวอย่าง Factory Method ที่อยู่ในวัตถุประสงค์ Creational และ Scope ของ Class แสดงดังรูปที่ 8 [6]



รูปที่ 8 ตัวอย่าง Class Diagram ของ Factory Method

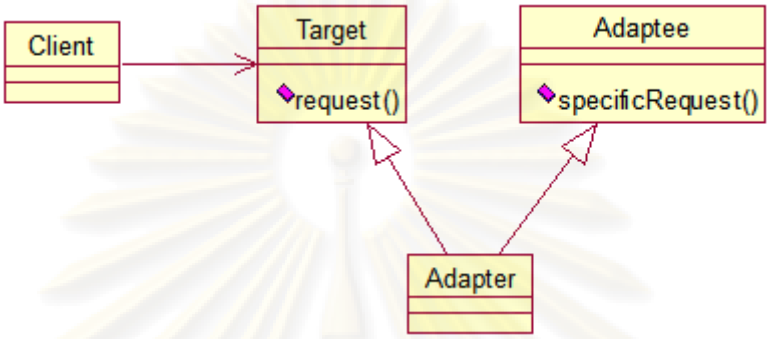
การเขียนโปรแกรมแบบ OOP นั้น Object จะถูกสร้างขึ้นจาก Class ซึ่งในโปรแกรมหรือระบบหนึ่งๆ ก็จะมี Class เป็นจำนวนมากหรือน้อยแตกต่างกันออกไป บ่อยครั้ง Class ที่แตกต่างกันเป็น Class ที่มาจาก Super Class เดียวกัน โดย ณ เวลาใดเวลาหนึ่ง Object ที่ถูกสร้างขึ้นจะเป็นของ Class ใดขึ้นอยู่กับเงื่อนไขของโปรแกรม ณ ขณะนั้น

ข้อดีของ Factory Method

1. Object ที่ต้องการนำเอา Product ไปใช้งาน ไม่จำเป็นต้องรู้โครงสร้างทั้งหมดของทุกๆ Sub Class ของ Product แต่ Object นั้นๆ จำเป็นต้องมีความรู้ 2 ประการเท่านั้นคือ เรียนรู้ว่า Product มี Service อะไรให้ใช้บ้าง และเรียนรู้ว่า ProductFactory มี Method ใดเป็น Factory Method และต้องใส่Parameter อะไรบ้าง

สรุปได้ว่าการสร้าง Object ด้วย Factory Method จะมีความยืดหยุ่นกว่าการสร้าง Object โดยตรงจาก Class ผู้ใช้ เพียงแต่เรียกใช้ Factory Method แล้วนำเอา Service ของ Object ที่ถูกสร้างขึ้นโดยผู้ใช้ไม่ทราบด้วยซ้ำว่า Object ที่ตนได้นั้น ในความจริงแล้วเป็น Object ของ Class ใด ซึ่งผลดีที่ตามมาคือ เราสามารถซ่อนรายละเอียดของ ConcreteProduct ไว้ ไม่ให้ภายนอกมองเห็นได้ และอีกประการคือผู้ใช้งานไม่จำเป็นต้องกังวลถึงโครงสร้างของ Object ของ ConcreteProduct ที่สร้างขึ้น เพียงแต่ใช้งานตามข้อตกชงที่กำหนดไว้ใน Interface Product ก็พอ

ตัวอย่าง Adapter ที่อยู่ในวัตถุประสงค์ Structural และ Scope ของ Class แสดงดังรูปที่ 9 [6]



รูปที่ 9 ตัวอย่าง Class Diagram ของ Adapter

Adapter Pattern เป็น Design Pattern ที่ใช้เพื่อแปลง (Adapt) interface หรือ Class หนึ่งๆ ให้มีความสามารถเหมือนกับ Class อื่น เราจะใช้ Adapter Pattern เมื่อต้องการให้ Class สอง Class ที่ไม่ได้มีส่วนเกี่ยวข้องสัมพันธ์กันให้สามารถติดต่อหรือทำงานร่วมกันได้

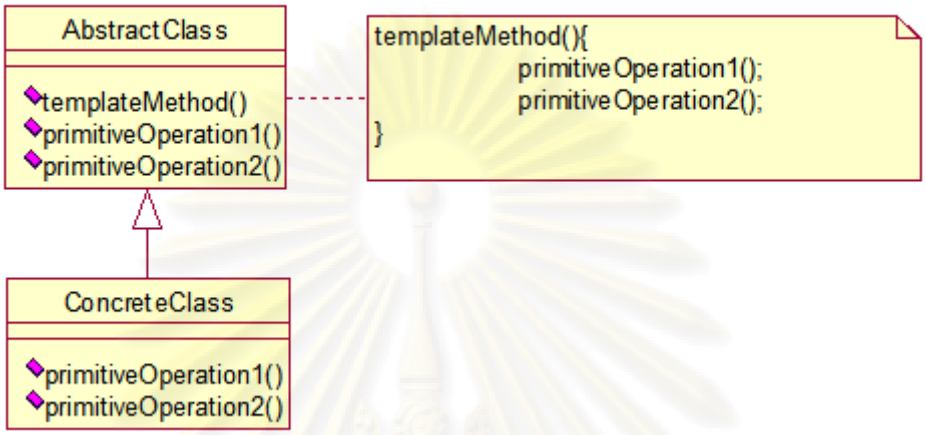
เราสามารถสร้าง Adapter ได้ 2 วิธี ได้แก่ การใช้ Inheritance และการใช้ Object Composition ซึ่ง Adapterที่ได้มาด้วยวิธีการ Inheritance เราจะเรียกว่า “Class Adapter” ส่วน Adapter ที่ได้มาด้วยวิธีการ Object Composition เรียกว่า “Object Adapter”

ข้อดีของ Adapter

1. การทำให้ Class หนึ่งๆ มีขีดความสามารถมากขึ้น โดยการ Adapt ความสามารถของ Class อื่นๆ มาใช้ ซึ่งทำได้สองวิธี คือ ใช้ Class Adapter และใช้ Object Adapter หากจะเปรียบเทียบระหว่าง Class Adapter กับ Object Adapter

นอกจากการสร้าง Adapter ตามปกติ ด้วยหลักการ Class Adapter และ Object Adapter ตามปกติแล้ว ยังสามารถประยุกต์ Class Adapter และ/หรือ Object Adapter เพื่อการสร้าง Adapter ให้มีโครงสร้างและการใช้งานที่มีประสิทธิภาพขึ้น และยืดหยุ่นกว่า Adapter ปกติด้วย อันได้แก่ Two-way Adapter และ Pluggable Adapter

ตัวอย่าง Template Method ที่อยู่ Behavioral และ Scope ของ Class แสดงดังรูปที่ 10 [6]



รูปที่ 10 ตัวอย่าง Class Diagram ของTemplate Method

การออกแบบระบบด้วยหลักการ OO นั้น สามารถใช้ Interface หรือ Abstract Class เพื่อประกาศโครงสร้างของ Method ที่จำเป็นต้องมีใน Class ได้โดยไม่จำเป็นต้องใส่ใจเรื่องของการ Implement Method ต่างๆ เหล่านั้น

แต่ Interface หรือ Abstract Class ตามปกตินั้น ไม่ได้ช่วยเราในการวางโครงของกิจกรรม ซึ่งกิจกรรมในที่นี้คือ ลำดับ ก่อน-หลัง ของการใช้ Method ต่างๆ ตัวอย่างเช่นในการ Copy File คือ การเกิดลำดับของกิจกรรมต่างๆ ดังนี้ สร้าง File ใหม่ อ่านเนื้อหาของ File ต้นทาง เขียนเนื้อหาที่อ่านได้ลงไปใน File ใหม่

ซึ่งการกำหนดกิจกรรมเหล่านี้ไม่สามารถกระทำได้ใน Abstract Class หรือ Interface ดังนั้นจึงเกิดแนวคิดในการสร้าง Design Pattern “Template Method” ที่เอื้ออำนวยให้สามารถกำหนดโครงสร้างของ Method ของ Class ได้พร้อมๆ กับการวางโครงของกิจกรรมที่จะเกิดขึ้นใน Method ได้ โดยที่สามารถโยนหน้าที่ในการ Implement Method เหล่านี้ให้กับ Class อื่นๆ ต่อไปได้

ข้อดีของ Template Method

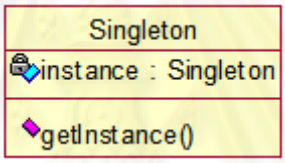
1. Template Method สนับสนุนหลักการ Modularity และ Reusability

AbstractClass จะกำหนดโครงร่างลำดับ (Skeleton) ของกิจกรรมไว้ใน Template Method โดย Template Method จะมีลำดับของกิจกรรมย่อยๆ ไว้อย่างแน่นอนตายตัว และ

AbstractClass จะยกหน้าที่ในการ Implement แต่ละลำดับของกิจกรรมไปให้กับ ConcreteClass ด้วยหลักการดังนี้ จะช่วยให้เราสามารถแยกแยะประเภทของกิจกรรมได้อย่างชัดเจน ตามจำนวนของ AbstractClass

ด้วยหลักการแบ่งย่อยลำดับของกิจกรรมนี้เอง ในกรณีที่เกิดข้อผิดพลาดในการทำงาน ก็จะสามารถตรวจสอบและระบุ ConcreteClass ที่มีปัญหาได้ทันที ซึ่งตรงตามวัตถุประสงค์ของ Modularity นอกจากนี้ ConcreteClass และ AbstractClass ยังสามารถถูกใช้งานแบบแยกจากกันได้

ตัวอย่าง Singleton ที่อยู่ในวัตถุประสงค์ Creational และ Scope ของ Object แสดงดังรูปที่ 11 [6]



รูปที่ 11 ตัวอย่าง Class Diagram ของ Singleton

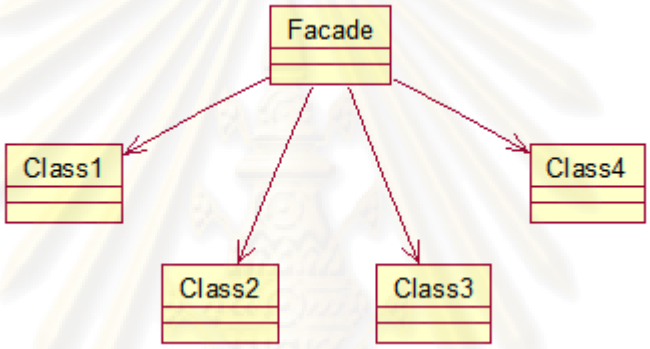
ในบางกรณี เราอาจต้องการให้มี Object หรือ Instance ของ Class ใด Class หนึ่ง เพียงตัวเดียวเท่านั้น ตัวอย่างเช่น ในระบบ client-server system หรือ distributed system ที่ยอมให้ผู้ใช้งานหลายๆ คนสามารถเข้ามาใช้งานระบบได้พร้อมๆ กัน การที่ผู้ใช้งานทุกๆ คนสามารถสร้าง Object จาก Class เดียวกันได้ อาจทำให้มี Object ของ Class เกินความจำเป็น ทั้งๆ ที่ในบางกรณีผู้ใช้งานหลายๆ คน สามารถใช้งาน Object เดียวกันได้ ตัวอย่างเช่น ในระบบ distributed system ที่มี Factory ที่ใช้ Factory Method Pattern เพื่อสร้าง Object ของ Class ซึ่งดูจะเกินความจำเป็นหากมี Object ของ Factory หนึ่งตัว สำหรับหนึ่งผู้ใช้งาน ทั้งๆ ที่ Factory เพียงตัวเดียว ก็สามารถให้บริการสร้าง Object ให้กับทุกๆ ผู้ใช้งานได้ ในกรณีนี้ Factory ควรจะมี Instance ได้เพียงตัวเดียว และหากมีผู้ใช้งานร้องขอให้สร้าง Instance ของ Factory ขึ้นอีก ระบบควรจะให้ผู้ใช้งานคนดังกล่าวใช้งาน Factory ที่สร้างขึ้นมาแล้ว แทนที่จะสร้าง Factory ขึ้นอีกตัวหนึ่ง โดยไม่จำเป็น

จากคุณสมบัติของ Singleton ไม่เพียงแค่จำกัดให้สร้าง Instance ของ Class ให้มีได้แค่ตัวเดียวเท่านั้น เราสามารถประยุกต์ใช้ Singleton เพื่อจำกัดจำนวนของ Instance (ไม่ใช่เพียงแค่ 1 ตัว)

ข้อดีของ Singleton

1. Singleton ช่วยให้การจัดการทรัพยากรของระบบได้อย่างมีประสิทธิภาพ
2. การควบคุมการเข้าถึงของ Object สามารถทำได้ดีด้วย Singleton Pattern
3. การเปลี่ยนแปลงจำนวนของ Instance ของ Singleton Class สามารถทำได้ง่าย

ตัวอย่าง Façade ที่อยู่ในวัตถุประสงค์ Structural และ Scope ของ Object แสดงดังรูปที่ 12 [6]



รูปที่ 12 ตัวอย่าง Class Diagram ของ Façade

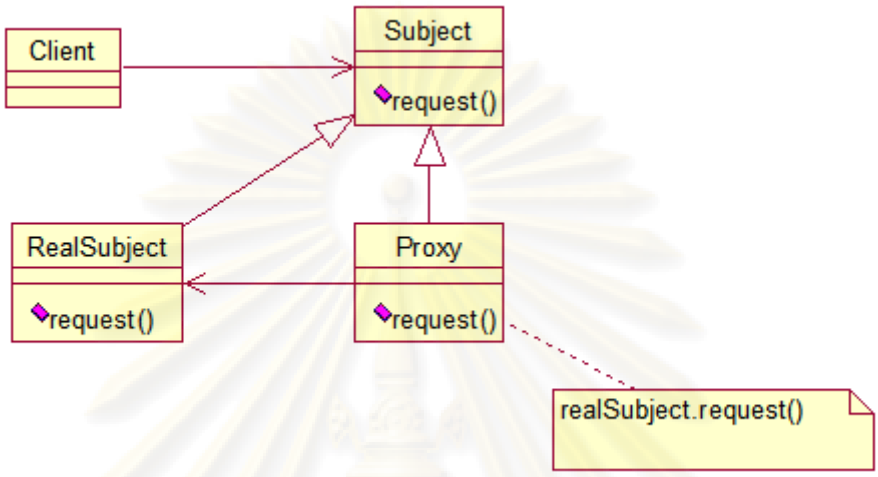
Encapsulation เป็นหลักการสำคัญข้อหนึ่งของ Object Oriented Software Engineering (OOSE) ซึ่ง Encapsulation คือ การทำให้ความยุ่งยากต่างๆ อันได้แก่ Attributes และ Method ได้รับการห่อหุ้มไว้ภายใน Class โดย Class จะทำหน้าที่ตัดสินใจว่าจะเปิดเผยส่วนหนึ่งส่วนใดให้ภายนอกรู้ จะปิดบังส่วนหนึ่งส่วนใดจากภายนอก และจัดเตรียมกลไกเพื่อให้ภายนอกสามารถติดต่อกับ Class ได้ด้วยหลักการ Encapsulation นี้เองทำให้สามารถติดต่อกับ Class ได้ด้วยความเรียบง่ายไม่จำเป็นต้องเข้าใจความยุ่งยากซับซ้อนของ Class แต่อย่างใด

Façade จะทำหน้าที่ในการเตรียม Interface ที่ทำให้สามารถติดต่อกับระบบที่ซับซ้อนได้โดยง่าย และไม่จำเป็นต้องเข้าใจถึงความซับซ้อนของระบบ

ข้อดีของ Façade

1. การใช้ Façade ช่วยลดภาระของ Client
2. Façade ช่วยจัดระบบในการเข้าถึง System

ตัวอย่าง Proxy ที่อยู่ในวัตถุประสงค์ Structural และ Scope ของ Object แสดงดังรูปที่ 13 [6]



รูปที่ 13 ตัวอย่าง Class Diagram ของ Proxy

ในการพัฒนาโปรแกรมแบบ OO เมื่อต้องการใช้งาน Object ต้องดำเนินการสร้าง Object จาก Class โดยเรียกกระบวนการนี้ว่า การ Instantation ในบางกรณีจำเป็นต้องใช้ Object หลายๆ ประเภท แต่ Object แต่ละประเภทไม่ได้ถูกเปิดใช้พร้อมๆ กัน และเพื่อให้การดำเนินการดังกล่าวสัมฤทธิ์ผล จะต้องใช้ Object ตัวหนึ่งที่ทำหน้าที่เหมือนเป็นตัวแทนของ Object รูปภาพ ซึ่งทำหน้าที่ดูแลว่าเมื่อใดภาพควรจะถูกแสดงขึ้น โดยตัวแทนดังกล่าวทำหน้าที่เก็บสิ่งที่ใช้อ้างอิงไปยัง Object รูปภาพดังกล่าว และจะสร้าง Object ของภาพขึ้นเมื่อถึงคราวจำเป็น โดยสิ่งที่ทำหน้าที่เป็นตัวแทนนี้จะเรียกว่า Proxy

ข้อดีของ Proxy

1. ช่วยรักษาความเป็นส่วนตัวและความปลอดภัยของ Object ได้ (Remote Proxy และ Protection Proxy)

2. Proxy ช่วย Optimize การสร้างและใช้งาน Object ได้

2.1.4 Tools Rational Rose

ปัจจุบันมีเครื่องมือที่เรียกว่า Visual Modeling Tool ซึ่งใช้ในการสร้าง UML Diagram มีอยู่มากมาย เช่น Together, Visual Paradigm, Rational Rose เป็นต้น สำหรับงานวิจัยนี้ได้มีการเลือกใช้ Tool ในการออกแบบ UML Diagram ที่มีชื่อว่า Rational Rose ซึ่งเป็นเครื่องมือที่มี

ชื่อเสียงและเป็นที่นิยมใช้กันมากในปัจุบันเนื่องจากมีคุณสมบัติมากมายในการช่วยพัฒนา Software ให้สมบูรณ์และรวดเร็วยิ่งขึ้น

Rational Rose เป็น Case Tool ที่ช่วยให้ System Analyst สร้าง Model ของระบบได้อย่างสะดวกและง่ายยิ่งขึ้น ซึ่งรองรับระบบที่ใช้การวิเคราะห์และออกแบบระบบเชิงวัตถุ (Object-Oriented System Analyst and Design) [1] โดยคุณสมบัติและจุดเด่นคือ

1. รองรับมาตรฐาน UML 2.0 , Booch , OMT
2. ประกอบด้วยเครื่องมือ ต่าง ๆ มากมายที่ใช้ในการสร้าง Object-Oriented Diagram ตามมาตรฐานของภาษา UML ช่วนในการสร้าง Diagram ทั้งหมดได้อย่างสะดวกและง่ายขึ้น
3. สามารถ Convert Model เป็นรูปแบบเอ็กซ์เอ็มแอลได้
4. สามารถแปลง Model เพื่อสร้างโครงสร้างของ Programming language ต่าง ๆ ได้ดังนี้ Corba , DDL , JAVA , C++ , Visual C++ ,Visual Basic6.0 , Visual Basic 5.0

2.1.5 **เอ็กซ์เอ็มแอล** (XML) [7]

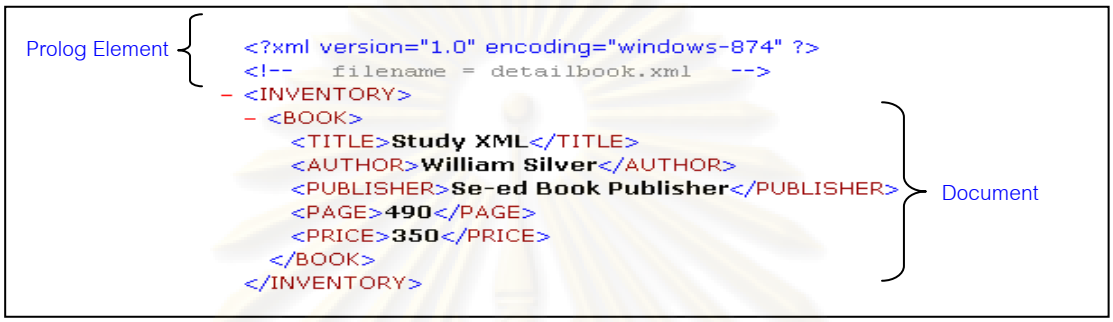
เอ็กซ์เอ็มแอล หรือ Extensible Markup Language (XML) เป็นภาษา Markup (ภาษาที่ใช้วิธีระบุเนื้อหาและจัดรูปแบบด้วย Text file) ที่มีแท็ก (Tag) คล้ายภาษา HTML มีความสามารถในการแสดงผลผ่าน Web Browser จึงถูกนำมาใช้เป็นสื่อกลางในการแลกเปลี่ยนข้อมูลบนอินเทอร์เน็ต เพราะมีความสามารถในการอธิบายความหมายของข้อมูลได้ นอกจากนี้เอ็กซ์เอ็มแอล ยังอนุญาตให้ผู้พัฒนากำหนดแท็กได้ตามต้องการ ดังนั้นเอ็กซ์เอ็มแอล จึงมีความยืดหยุ่นและใช้งานได้หลากหลายกว่า HTML ทั้งนี้ แนวคิดของเอ็กซ์เอ็มแอลยังนับว่าเป็นพื้นฐานสำคัญในการเรียนรู้ SOAP WSDL และUDDI อีกด้วย

2.1.5.1 **ส่วนประกอบของเอกสารเอ็กซ์เอ็มแอล**

แท็ก (Tag) เป็นส่วนประกอบสำคัญของภาษา Markup การกำหนดแท็กเริ่มต้น (Start Tag) ชื่อแท็กจะอยู่ภายในเครื่องหมาย “<” และ “>” เช่น <INVENTORY>” ส่วนการกำหนดแท็กสิ้นสุด (End Tag) จะกำหนดชื่อของ Tag อยู่ภายในเครื่องหมาย “</” และ “>” เช่น </INVENTORY> โดยจะมีเครื่องหมาย “/” แทรกอยู่หน้าชื่อ ส่วนข้อมูลต่างๆ จะอยู่ระหว่าง Start Tag กับ End Tag โดยส่วนประกอบหลักของเอกสารเอ็กซ์เอ็มแอลมีดังนี้

2.1.5.2 องค์ประกอบของเอกสาร เอ็กซ์เอ็มแอล

เอกสารเอ็กซ์เอ็มแอลมีองค์ประกอบหลัก 2 ส่วน คือ Prolog Element และ Document Element (หรือ Root Element) ดังรูปที่ 14



รูปที่ 14 องค์ประกอบของเอกสารเอ็กซ์เอ็มแอล

- **Prolog Element**

เป็นส่วนของการประกาศเอกสารเอ็กซ์เอ็มแอลสามารถบรรจุส่วนต่างๆ ดังนี้

1) เอ็กซ์เอ็มแอล Declaration คือ ส่วนที่แจ้งให้ทราบว่าเอกสารนี้ เป็นเอกสารเอ็กซ์เอ็มแอลโดยระบุเวอร์ชันของเอ็กซ์เอ็มแอล และระบุการเข้ารหัสถ้าต้องการให้แสดงผลภาษาไทยสามารถกำหนดรหัสเป็น “windows-874” หรือ “tis-620”

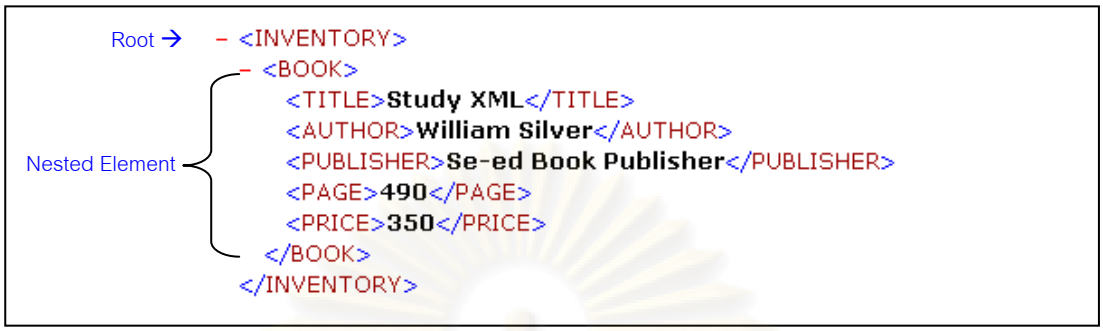
2) Comment คือ ข้อความอธิบายเอกสาร โดยประมวลผลของเอ็กซ์เอ็มแอล (XML Parser) สามารถกำหนดข้อความไว้ระหว่างเครื่องหมาย “<!—“ และ “-->” ได้

3) DTD คือ เอกสารเอ็กซ์เอ็มแอลรูปแบบหนึ่งที่ใช้กำหนดความถูกต้องให้กับข้อมูลใน Element

4) Procession Instruction คือ ส่วนประมวลผลข้อมูลที่ XML Parser จะส่งให้กับภาษาที่ใช้แสดงผลข้อมูล เช่น Style Sheet, HTML, XSL และภาษาอื่นๆ ที่ใช้ร่วมกับเอ็กซ์เอ็มแอลได้

- **Document Element**

คือ Element ที่ประกอบด้วย Element ย่อย (Nested Element) อื่นๆ ซ้อนกันอยู่เป็นลำดับ โดยคำอธิบายที่สามารถบอกได้ว่าข้อมูลใน Element คือข้อมูลอะไร ดังนั้นการกำหนดชื่อของ Element จึงควรกำหนดให้สอดคล้องกับข้อมูล ดังรูปที่ 15



รูปที่ 15 ส่วนประกอบของ Document Element

2.1.5.3 XML Element

คือ ส่วนที่ใช้แสดงโครงสร้างและเนื้อหาต่างๆ ของเอกสารเอ็กซ์เอ็มแอล โดยอีลีเมนต์บนสุดเรียกว่า Root Element ซึ่งสามารถมีได้เพียงอีลีเมนต์เดียวเท่านั้น ส่วนอีลีเมนต์อื่นๆ เรียกว่า “Nested Element” ทั้งนี้ อีลีเมนต์สามารถแบ่งตามโครงสร้างได้ 3 ประเภท ดังนี้

1) Simple Element คือ อีลีเมนต์ที่ไม่มีอีลีเมนต์อื่นอยู่ภายใน โดยทั่วไปนิยมกำหนดให้กับข้อมูลที่ไม่มีองค์ประกอบย่อยอื่นๆ

2) Mixed Element คือ อีลีเมนต์ที่มีอีลีเมนต์อื่นอยู่ภายใน โดยทั่วไปนิยมกำหนดให้กับข้อมูลที่มีองค์ประกอบย่อย

3) Empty Element คือ อีลีเมนต์ที่ไม่ได้บรรจุข้อมูล หรือช่องว่างใด ๆ ไว้ภายใน

2.1.5.4 XML Attribute

คือ การระบุคุณสมบัติบางประการให้กับอีลีเมนต์ โดยแอททริบิวต์ไม่นับว่าเป็นส่วนหนึ่งของข้อมูลจริง แต่มีประโยชน์ในการอธิบายส่วนเพิ่มเติมให้กับอีลีเมนต์แต่ละตัว โดยค่าของแอททริบิวต์ต้องกำหนดไว้ระหว่างเครื่องหมาย Double Quote (“) หรือ Single Quote (‘) เช่น <BOOK ISBN = ‘974-94136-8-7’> ……</BOOK> โดยสมมติให้อีลีเมนต์ “BOOK” เก็บข้อมูลที่ต้องการทราบเกี่ยวกับหนังสือต่างๆ ไว้ เช่น หนังสือ ผู้แต่ง จำนวนหน้า และราคา เป็นต้น โดยมีแอททริบิวต์ชื่อ “ISBN” เป็นข้อมูลที่บอกความหมายเพิ่มเติมของอีลีเมนต์นี้ ทำให้ทราบว่าหนังสือเล่มดังกล่าวมีรหัสเป็นเลขอะไร เป็นต้น โดยประโยชน์ที่สำคัญของแอททริบิวต์ คือ ช่วยในการค้นหา หรือกรองข้อมูลก่อนการแสดงผล ทำให้ XML Parser สามารถประมวลผลได้ถูกต้อง และรวดเร็วมากยิ่งขึ้น

2.1.5.5 XML Entity

คือ กลุ่มของอักขระที่ถูกกำหนดความหมายไว้ โดย XML Parser จะประมวลผลกลุ่มอักขระนั้นๆ แล้วส่งค่าออกมาเป็นผลลัพธ์ ซึ่ง Entity ที่กำหนดโดยองค์กร W3C ทำให้ XML Parser สามารถประมวลผล Entity นั้นๆ ได้อย่างถูกต้อง ที่สำคัญมี ดังนี้

- ⁢ มีค่าเท่ากับ <
- > มีค่าเท่ากับ >
- &amp มีค่าเท่ากับ &
- &qout; มีค่าเท่ากับ “
- ' มีค่าเท่ากับ ‘

จาก Entity ข้างต้น สังเกตได้ว่า ผลลัพธ์ของ Entity จะเป็นอักขระพิเศษที่มีผลต่อการทำงานของ XML Parser เช่น ถ้าใส่เครื่องหมาย < ลงในส่วนของข้อมูลจะทำให้เกิดข้อผิดพลาดขึ้นทันที เนื่องจากเอ็กซ์เอ็มแอลตีความหมายสัญลักษณ์ดังกล่าวเป็นเครื่องหมายของแท็กที่ใช้กำหนดอีลีเมนต์ เป็นต้น

2.1.6 IEEE1012: IEEE Standard for Software Verification and Validation [2]

กระบวนการตรวจสอบความถูกต้องของระบบงาน เป็นกระบวนการที่เกิดขึ้นอย่างต่อเนื่องตลอดช่วงระยะเวลาในการพัฒนาระบบ จนถึงช่วงเวลาของการติดตั้ง และใช้งานระบบ กระบวนการตรวจสอบความถูกต้องของระบบงาน จะช่วยให้ผู้พัฒนาแน่ใจว่าระบบงานที่ได้พัฒนาขึ้นมานั้น สามารถทำงานและตอบสนองความต้องการ ได้ตรงตามความต้องการของผู้ใช้และตรงตามข้อตกลงหรือไม่ โดยที่ความหมายของ Verification และ Validation นั้น มีความหมายที่แตกต่างกันออกไป ดังนี้

- Verification คือ การตรวจสอบ และประเมินว่าระบบหรือโปรแกรมที่พัฒนาขึ้นมานั้นได้พัฒนาให้ตรงตามข้อกำหนดความต้องการที่ได้รับมาหรือไม่

- Validation คือ การตรวจสอบ และประเมินว่าระบบหรือโปรแกรมที่พัฒนาขึ้นมานั้นได้พัฒนาให้ตรงตามความคาดหวัง ตามที่ผู้ใช้งานระบบได้คาดหวังไว้หรือไม่

วัตถุประสงค์ของการทำ Verification และ Validation นั้น คือเพื่อค้นหาข้อบกพร่อง และค้นหาข้อผิดพลาด ที่อาจเกิดได้จากการใช้งานระบบ อันเนื่องมากจากปัจจัยหลายอย่าง ตั้งแต่การวิเคราะห์ การออกแบบ จนกระทั่งถึงการเขียนโปรแกรม และเพื่อเป็นการประเมินว่าระบบที่

พัฒนาขึ้นมานั้น สามารถทำงานได้จริง สามารถประมวลผลได้อย่างถูกต้อง แม่นยำ และตอบสนองความต้องการของผู้ใช้งานได้ ในสภาพการทำงานจริงหรือไม่

เทคนิควิธีการที่ใช้ใน Software Verification แบ่งออกเป็น 2 ประเภท คือ

- Review เป็นวิธีการทบทวนความถูกต้องของผลิตภัณฑ์งาน (work product) ระหว่างกระบวนการพัฒนา

- Testing เป็นการทดสอบการทำงานของโปรแกรมว่าถูกต้อง ตรงตามข้อกำหนด

2.1.7 IEEE1028: IEEE Standard for Software Reviews [3]

มาตรฐานนี้ อธิบายถึง 5 ประเภทของการตรวจทาน รวมถึงความแตกต่างของวัตถุประสงค์ วิธีการในการตรวจทาน ผู้เข้าร่วมและบทบาทหน้าที่ กล่าวคือ

- Management reviews เป็นการตรวจทานในส่วนของการจัดการ สถานะของโครงการ และการจัดการทรัพยากร โดยที่ Management reviews นี้จะถูกนำมาช่วยในการตัดสินใจ เพื่อให้การจัดการมีประสิทธิภาพ ไม่ว่าจะเป็นการจัดสรรทรัพยากร หรือการเปลี่ยนแปลงขอบเขตของโครงการ

- Technical reviews เป็นการตรวจทานเพื่อเป็นการตรวจดูคุณภาพของระบบ ว่าสามารถทำงานได้ตามรายละเอียด หรือตามมาตรฐานหรือไม่ โดยที่การตรวจทานนั้น จะเน้นไปที่ด้านเทคนิคการพัฒนา

- Inspections เป็นการตรวจทานเพื่อเป็นการตรวจดูระบบ ว่าสามารถทำงานตามรูปแบบของระบบ ไม่มีข้อผิดพลาด หรือสิ่งผิดปกติใดๆ เพื่อให้การทำงานของระบบนั้นมีคุณภาพและเป็นที่น่าพอใจ

- Walk-troughs เป็นการตรวจทานอย่างไม่เป็นทางการเพื่อหาข้อผิดพลาด หรือสิ่งผิดปกติโดยเจ้าของงาน หรือเป็นการตรวจทานเพื่อปรับปรุงคุณภาพการทำงานของระบบ และผู้พัฒนานั้น สามารถขอคำแนะนำ เกี่ยวกับการแก้ไขปัญหา หรือเพื่อขอความยินยอมเห็นชอบ

- Audits เป็นการตรวจทานเชิงวัตถุวิสัยของผลิตภัณฑ์งานหรือชุดของผลิตภัณฑ์งานเทียบกับเกณฑ์ที่กำหนด ปฏิบัติโดยบุคคลผู้ได้รับอนุญาต จุดประสงค์เพื่อจัดเตรียมและทำการประเมินผลิตภัณฑ์ซอฟต์แวร์และกระบวนการ อย่างเป็นอิสระ ในการประเมินความสอดคล้องกับข้อกำหนดความต้องการ กระบวนการที่เป็นระบบ มีอิสระ และมีเอกสารแสดง สำหรับการได้มาซึ่งหลักฐานการตรวจสอบและประเมินหลักฐานนั้นอย่างมีวัตถุประสงค์ เพื่อระบุว่าเกณฑ์การตรวจสอบถูกกระทำอย่างครบถ้วน

การตรวจทานโปรแกรมจัดอยู่ในประเภท Inspection หรือปัจจุบันเรียกว่า Peer Review เป็นขั้นตอนหนึ่งในการทวนสอบความถูกต้องของผลิตภัณฑ์งานระหว่างการพัฒนาระบบซอฟต์แวร์ มักกระทำโดยเจ้าหน้าที่ผู้ชำนาญการเพื่อหาข้อบกพร่อง เพื่อแก้ไข หรือเป็นการตรวจดูว่า การเขียนโปรแกรมนั้นเป็นที่ยอมรับได้หรือไม่

2.2 งานวิจัยที่เกี่ยวข้อง

จากการศึกษาค้นคว้าพบรายงานการวิจัยที่เกี่ยวข้องดังต่อไปนี้

1. Automatic Generation of Java Code from UML Diagrams using UJECTOR [8]

งานวิจัยชิ้นนี้ได้นำเสนอเครื่องมือที่มีชื่อว่า “UJECTOR” โดยเครื่องมือนี้สามารถทำการสร้างชุดคำสั่งภาษา จาวาขึ้นโดยอัตโนมัติ จาก UML Diagram ตัวอย่าง Diagram ที่เครื่องมือสามารถสร้างชุดคำสั่งได้โดยอัตโนมัติได้แก่ Class Diagram, Sequence Diagram และ Activity Diagram

งานวิจัยนี้ยังทำการเปรียบเทียบเครื่องมือ UJECTOR กับเครื่องมือที่ใช้ในการพัฒนาโปรแกรมทั่ว ๆ ไปโดยผลลัพธ์ที่ได้คือเครื่องมือ UJECTOR สามารถสร้างฟังก์ชันการทำงานได้ครบถ้วน และทำความเข้าใจการทำงานได้มากกว่าอีกด้วย

2. What Makes a Code Review Trustworthy? [9]

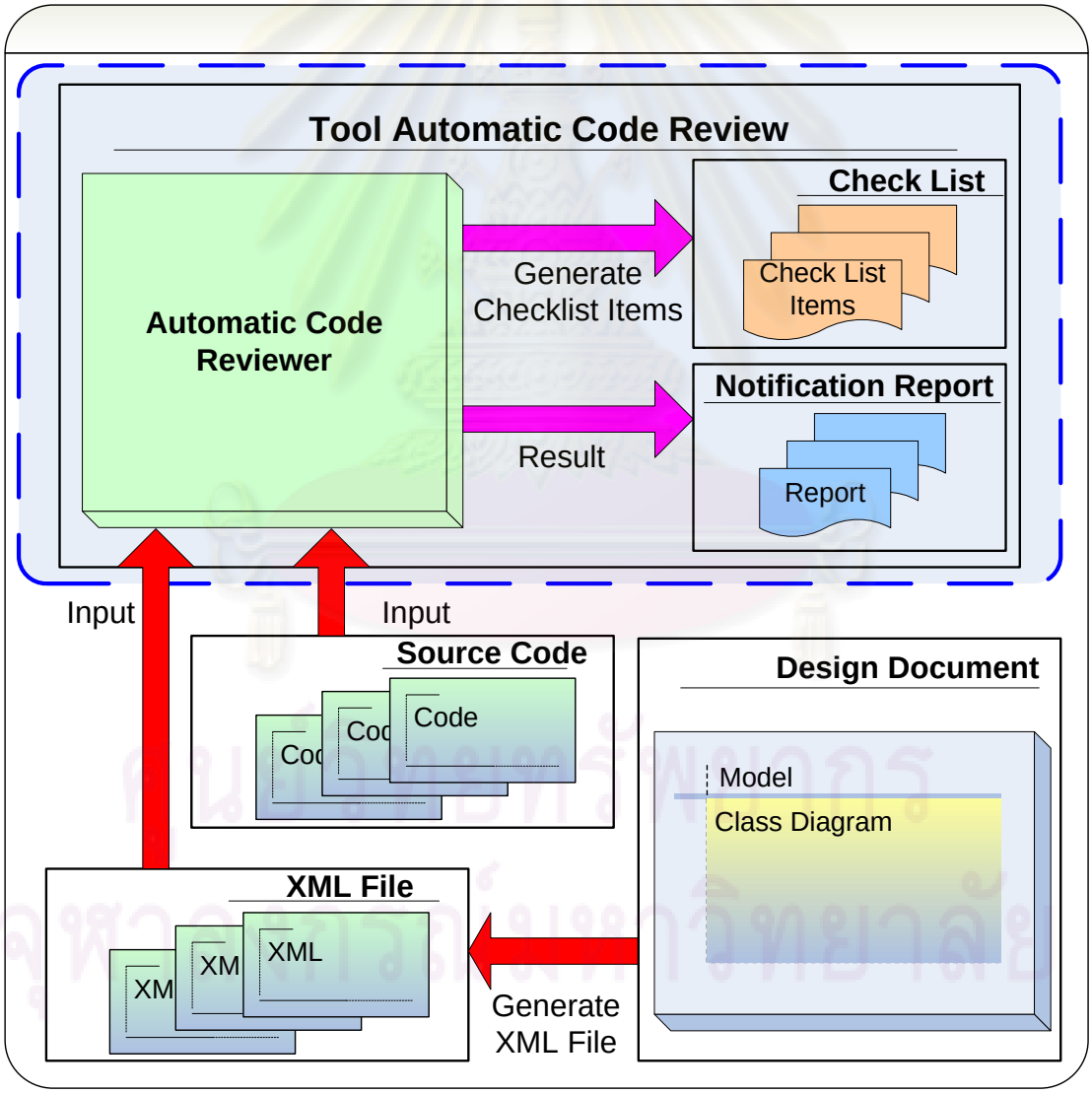
งานวิจัยนี้ ได้กล่าวถึงความสำคัญของการทำ Code Review ว่า เป็นขั้นตอนสำคัญที่จะทำการรับรองได้ว่า ระบบที่พัฒนาขึ้นมานั้น มีความปลอดภัยหรือไม่ โดยผู้ที่ทำหน้าที่เป็น Reviewer นั้น จำเป็นที่จะต้องมีทักษะ และประสบการณ์อย่างสูง เนื่องจาก Reviewer นั้น จะเป็นผู้จัดทำรายการตรวจสอบ (Checklist) ในการ review จาก high-level requirements ไปสู่ minute language details เพื่อใช้ในการทบทวน (review)

งานวิจัยนี้ ยังได้มุ่งประเด็นไปที่คำถามที่ว่า “ทำอย่างไร ถึงจะมั่นใจได้ว่า การทำ Code Review นั้น เป็นที่น่าเชื่อถือ” โดยที่ผู้จัยได้ทำการค้นคว้า โดยการสัมภาษณ์ผู้เชี่ยวชาญเพื่อหาถึงประเด็นสำคัญ ในขั้นตอนของการทำ Code Review เพื่อที่จะวิเคราะห์หาประเด็นที่คล้ายกัน มาจัดทำเป็น common properties

ในงานวิจัยนี้ยังได้สรุปถึง การค้นคว้าหาจุดสำคัญของการทำ Code Review เพื่อเป็นข้อมูลการวิเคราะห์สำหรับการหาวิธีและเครื่องมือในการเพิ่มความน่าเชื่อถือของระบบ

บทที่ 3
การออกแบบขั้นตอนการดำเนินงาน

งานวิจัยนี้มีแนวคิดที่จะพัฒนาระบบตัวตรวจทานชุดคำสั่งอัตโนมัติเพื่อช่วยค้นหาการออกแบบ Class Diagram ที่มีโครงสร้างแบบ Design Pattern เข้ามาประยุกต์ ตรวจจับหาข้อผิดพลาดของชุดโปรแกรมภาษาจาวา รวมถึงยังช่วยลดจำนวนการเกิด Defect ตั้งแต่ Design Phase อีกด้วย โดยเครื่องมือที่ทำการพัฒนานั้นจะตรวจสอบจากข้อมูลตั้งต้นที่เป็นไฟล์เอ็กซ์เอ็มแอล ที่ได้มาจากเครื่องมือ Rational Rose โดยภาพรวมการทำงานของเครื่องมือแสดงดังรูปที่ 16



รูปที่ 16 สถาปัตยกรรมของระบบตัวตรวจทานชุดคำสั่งอัตโนมัติ

ระบบที่พัฒนาขึ้นมีการออกแบบขั้นตอนการทำงานหลักๆ กล่าวคือ

- วิเคราะห์ความต้องการระบบเพื่อออกแบบฟังก์ชันการทำงานต่างๆ ของระบบรวมถึง Design Pattern ที่ครอบคลุมการตรวจทานในเครื่องมือที่พัฒนาขึ้น
- แปลงแผนภาพคลาสยูเอ็มแอลให้เป็นไฟล์เอ็กซ์เอ็มแอลด้วยเครื่องมือ Rational Rose
- ดึงรายละเอียดข้อมูลในไฟล์เอ็กซ์เอ็มแอล เพื่อสร้างรายการตรวจทานตามเกณฑ์การตรวจที่กำหนดไว้
- อ่านชุดคำสั่งภาษาจาวาที่ต้องการตรวจทานเข้าระบบ
- สร้างรายงานสรุปผลการตรวจทานชุดคำสั่ง

3.1 วิเคราะห์ความต้องการระบบและกำหนด Design Patterns ที่ระบบสามารถตรวจทานได้

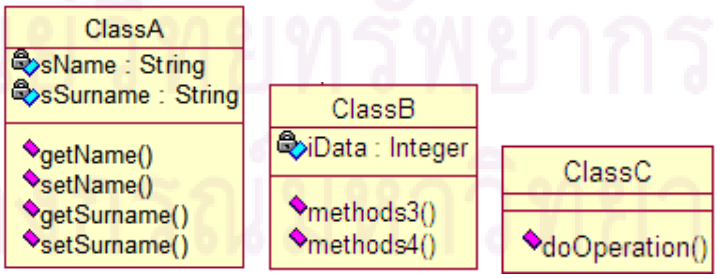
โดย Design Patterns ที่ระบบสามารถตรวจทานได้ประกอบไปด้วย Factory Method, Adapter, Template Method, Singleton, Façade และ Proxy โดยฟังก์ชันหน้าที่การทำงานของระบบจะมีการอธิบายอยู่ในบทที่ 4

3.2 แปลงแผนภาพคลาสเป็นไฟล์เอ็กซ์เอ็มแอล

การแปลงแผนภาพคลาสเป็นไฟล์เอ็กซ์เอ็มแอล สำหรับงานวิจัยนี้สามารถแบ่งออกเป็น 2 รูปแบบ คือ การแปลงข้อมูลคลาสให้อยู่ในรูปแบบเอ็กซ์เอ็มแอล และการแปลงข้อมูลความสัมพันธ์ระหว่างคลาสให้อยู่ในรูปแบบเอ็กซ์เอ็มแอล

3.2.1 การแปลงข้อมูลคลาสให้อยู่ในรูปแบบเอ็กซ์เอ็มแอล

ตัวอย่างการแปลง Class A, Class B, Class C ดังรูปที่ 17

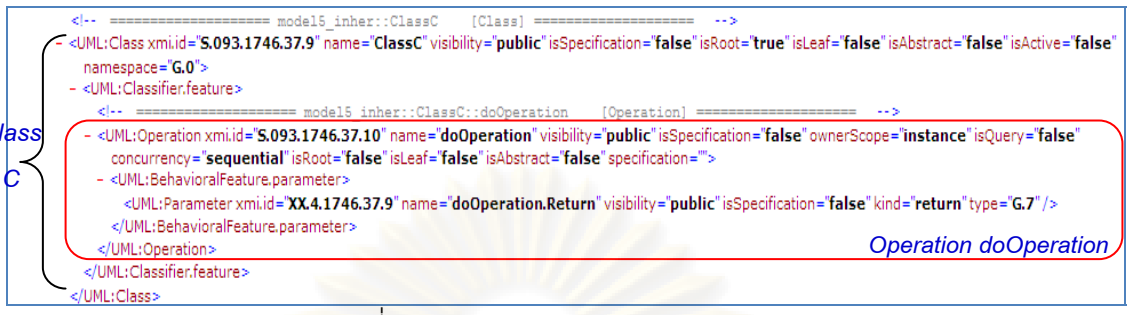


รูปที่ 17 ตัวอย่าง Class A, B, C

คลาสต่างๆ ข้างต้น เมื่อใช้ Rational Rose เพื่อแปลงเป็นไฟล์เอ็กซ์เอ็มแอล จะได้ผลลัพธ์ดังแสดงในรูปที่ 18, 19, 20, 21 ตามลำดับ

รูปที่ 18 ตัวอย่างเอ็กซ์เอ็มแอล Class A

รูปที่ 19 ตัวอย่างเอ็กซ์เอ็มแอล Class B

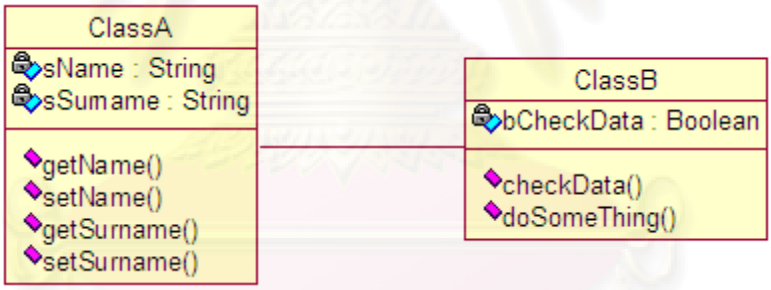


รูปที่ 20 ตัวอย่างเอ็กซ์เอ็มแอล Class C

รูปที่ 21 ตัวอย่างประเภทตัวแปรที่มีการคืนค่าของ Class A และ Class B และ Class C

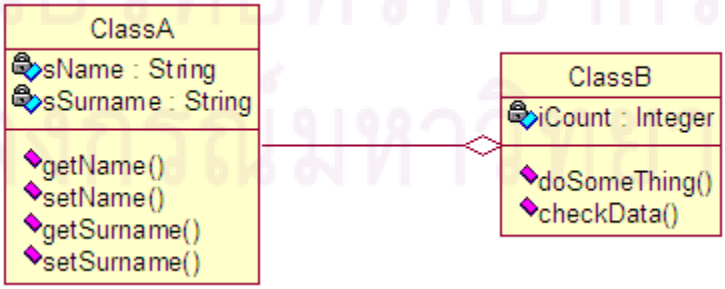
3.2.2 การแปลงข้อมูลความสัมพันธ์ระหว่างคลาสให้อยู่ในรูปแบบเอ็กซ์เอ็มแอล

- Association Relation ตัวอย่างดังแสดงในรูปที่ 22



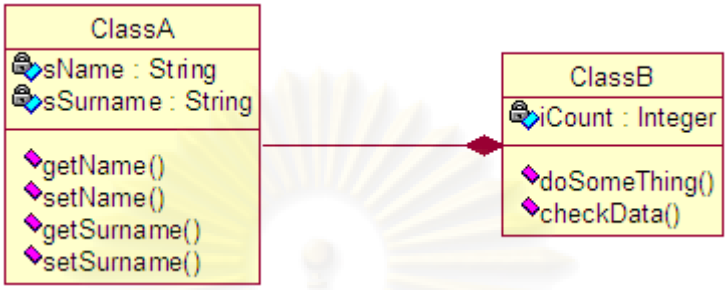
รูปที่ 22 ตัวอย่าง Class A กับ Class B ที่มีความสัมพันธ์แบบ Association

- Aggregation Relation ตัวอย่างดังแสดงในรูปที่ 23



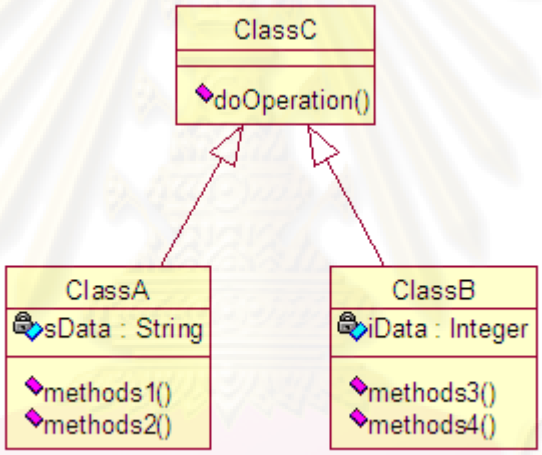
รูปที่ 23 ตัวอย่าง Class A กับ Class B ที่มีความสัมพันธ์แบบ Aggregation

- Composition Relation ตัวอย่างดังแสดงในรูปที่ 24



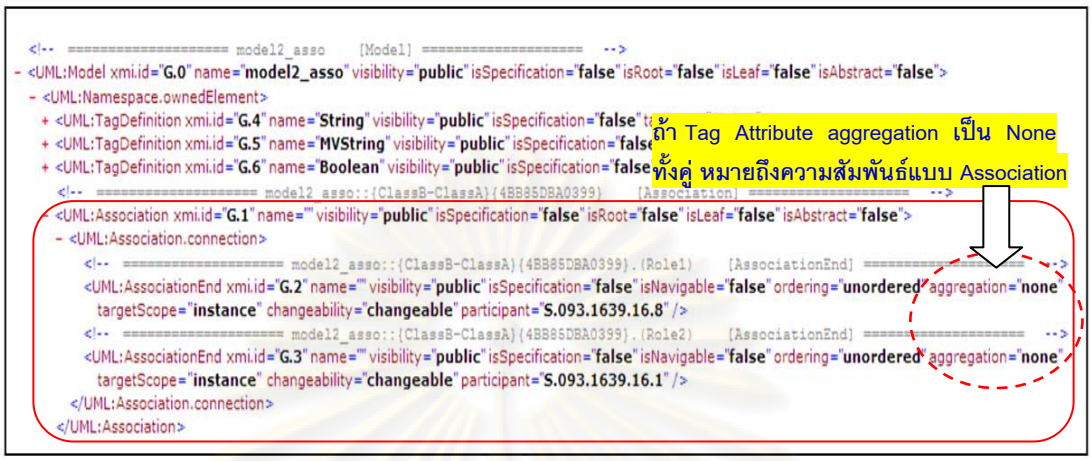
รูปที่ 24 ตัวอย่าง Class A กับ Class B ที่มีความสัมพันธ์แบบ Composition

- Generalization Relation ตัวอย่างดังแสดงในรูปที่ 25



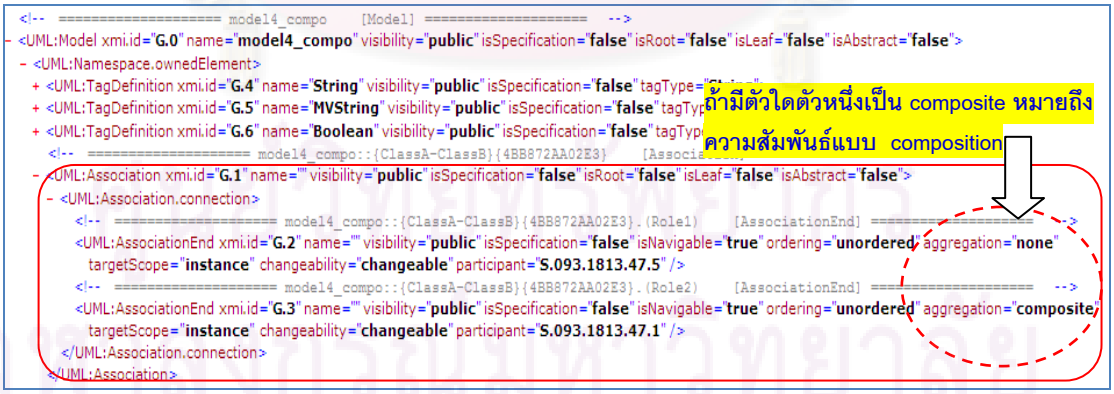
รูปที่ 25 ตัวอย่าง Class A กับ Class B ที่มีความสัมพันธ์แบบ Generalization

Relationships ต่างๆ ข้างต้น เมื่อใช้ Rational Rose เพื่อแปลงเป็นไฟล์เอ็กซ์เอ็มแอล จะได้ผลลัพธ์ดังแสดงในรูปที่ 26, 27, 28, 29 ตามลำดับ

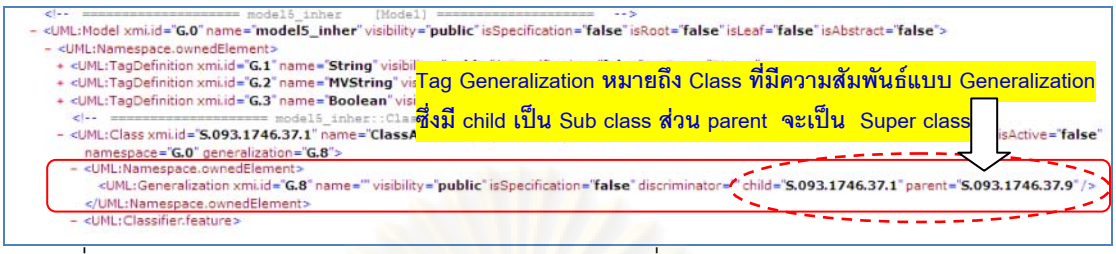


รูปที่ 26 ตัวอย่างเอ็กซ์เอ็มแอล Class A กับ Class B ที่มีความสัมพันธ์แบบ Association

รูปที่ 27 ตัวอย่างเอ็กซ์เอ็มแอลClass A กับ Class B ที่มีความสัมพันธ์แบบ Aggregation



รูปที่ 28 ตัวอย่างเอ็กซ์เอ็มแอล Class A กับ Class B ที่มีความสัมพันธ์แบบ Composition



รูปที่ 29 ตัวอย่างเอ็กซ์เอ็มแอล Class A กับ Class B ที่มีความสัมพันธ์แบบ Generalization

3.3 สร้างรายการตรวจทานจากรายละเอียดในไฟล์เอ็กซ์เอ็มแอล ตามเกณฑ์การตรวจทานที่กำหนดไว้

ในการสร้างเกณฑ์การตรวจทานและรายการตรวจทานจากรายละเอียดในไฟล์เอ็กซ์เอ็มแอล จำเป็นจะต้องมีการกำหนดรูปแบบคุณสมบัติต่างๆ ดังต่อไปนี้

3.3.1 เกณฑ์การตรวจทาน

งานวิจัยนี้ได้มีการแบ่งเกณฑ์การตรวจทานออกเป็น 2 รูปแบบ คือ เกณฑ์การตรวจทานโครงสร้างของคลาส ความสัมพันธ์ของคลาสทั่วไป และเกณฑ์การตรวจทานของ Design Pattern ดังต่อไปนี้

3.3.1.1 เกณฑ์การตรวจทานโครงสร้างของคลาส และความสัมพันธ์ของคลาสทั่วไป

เกณฑ์การตรวจทานโครงสร้างของคลาส และความสัมพันธ์ของคลาส แสดงดังตารางที่ 2 และ 3 ตามลำดับ

ตารางที่ 2 คุณสมบัติของโครงสร้างคลาสทั่วไป

เกณฑ์การตรวจทาน	คำอธิบาย
1. Class Name	ชื่อ Class
2. Attribute Name	ชื่อ Attribute
3. Attribute Visibility	ระดับการมองเห็นของ Attribute ประกอบไปด้วย private , public และ protected
4. Attribute Type	ประเภท Data Type ของ Attribute เช่น String, int ,Boolean เป็นต้น
5. Method Name	ชื่อ Method
6. Method Visibility	ระดับการมองเห็นของ Method ประกอบไปด้วย private , public และ protected

ตารางที่ 2 (ต่อ) คุณสมบัติของโครงสร้างคลาสทั่วไป

เกณฑ์การตรวจทาน	คำอธิบาย
7. Method Parameter Name	ชื่อ Method Parameter
8. Method Parameter Type	ประเภท Data Type ของ Method Parameter เช่น String , int ,Boolean เป็นต้น

ตารางที่ 3 คุณสมบัติความสัมพันธ์ของคลาส

หัวข้อเกณฑ์การตรวจทาน	คำอธิบาย
1. ความสัมพันธ์ Association	แสดงความสัมพันธ์ระหว่าง Object หรือ Class สองทิศทาง
2. ความสัมพันธ์ Aggregation	แสดงความสัมพันธ์ระหว่าง Object หรือ Class แบบ “Whole-Part” หรือ “is part of” โดยจะมี Class ที่ใหญ่ที่สุดที่เป็น Object หลัก และมี Class อื่นเป็นส่วนประกอบ
3. ความสัมพันธ์ Composition	แสดงความสัมพันธ์ระหว่าง Object หรือ Class แบบขึ้นต่อกันและมีความเกี่ยวข้องกันเสมอ โดยจะมี Class ซึ่งเป็นองค์ประกอบของ Class อื่นที่ใหญ่กว่า เมื่อ Class ที่ใหญ่กว่าถูกทำลาย Class ที่เป็นองค์ประกอบก็จะถูกทำลายไปด้วย
4. ความสัมพันธ์ Generalization	แสดงความสัมพันธ์ระหว่าง Object หรือ Class ในลักษณะของการสืบทอดคุณสมบัติจาก Class หนึ่ง (Superclass) ไปยังอีก Class หนึ่ง (Subclass)

3.3.1.2 เกณฑ์การตรวจทานโครงสร้าง Design Pattern

เกณฑ์การตรวจทาน Design Pattern สำหรับงานวิจัยนี้มี 6 รูปแบบดังนี้

3.3.1.2.1 เกณฑ์การตรวจทาน Factory Method

จาก Design Pattern รูปที่ 8 สามารถนำมาสร้างเกณฑ์การตรวจทานของ Factory Method ได้ดังตารางที่ 4

ตารางที่ 4 เกณฑ์การตรวจทานของ Factory Method

เกณฑ์การตรวจทาน	**คำอธิบาย**
คุณสมบัติหลักของ Factory Method	1. Class Product เป็น Class หรือ Interface ที่กำหนดคุณสมบัติ (ร่วม) ของ Object ที่จะถูกสร้างขึ้นโดย Factory Method 2. Class ConcreteProduct เป็น Sub Class ของ Product 3. Class Creator เป็น Class หรือ Interface ที่กำหนดคุณสมบัติ (ร่วม) ของ Object ที่จะทำหน้าที่สร้าง Sub Class หรือ Implementation Class ของ Product 4. Method ของ Class Creator ที่ใช้สร้าง Object ของ Class Product เรียกว่า factoryMethod() 5. Class ConcreteCreator เป็น Class ที่ทำหน้าที่ Override Method factoryMethod() ของ Class Creator
คุณสมบัติหลักของความสัมพันธ์ระหว่างคลาสใน Factory Method	1. ความสัมพันธ์ Generalization ระหว่าง Class Creator กับ Class ConcreteCreator โดยมีการสืบทอดคุณสมบัติจาก Class Creator ไปยัง Class ConcreteCreator 2. ความสัมพันธ์ Generalization ระหว่าง Class Product กับ Class ConcreteProducct โดย มีการสืบทอดคุณสมบัติจาก Class Product ไปยัง Class ConcreteProduct 3. ความสัมพันธ์ Association แบบมีทิศทาง ระหว่าง Class ConcreteCreator กับยัง Class ConcreteProduct โดย Class ConcreteCreator มีการเรียกใช้ Object จาก Class ConcreteProduct

3.3.1.2.2 เกณฑ์การตรวจทาน Adapter

จาก Design Pattern รูปที่ 9 สามารถนำมาสร้างเกณฑ์การตรวจทานของ Adapter ได้ดังตารางที่ 5

ตารางที่ 5 เกณฑ์การตรวจทานของ Adapter

เกณฑ์การตรวจทาน	คำอธิบาย
คุณสมบัติหลักของ Adapter	1. Class Target เป็น Class หรือ Interface ที่มี method ต่าง ๆ ที่ Class Client สามารถเรียกใช้งานได้ 2. Class Client ทำหน้าที่เป็นผู้เรียกใช้ method ต่าง ๆ ที่ประกาศไว้โดย Class Target 3. Class Adaptee เป็น Class ที่มี method ที่เป็นที่ต้องการของ Class Target และจะถูกเรียกใช้โดย Class Target ผ่านทาง Class Adapter 4. Class Adapter ทำหน้าที่เรียกใช้ method ของ Class Adaptee โดย method ของ Class Adaptee จะเรียกอยู่ภายใน method ของ Class Adapter ซึ่ง method ของ Adapter จะถูกเรียกใช้โดย Class Client
คุณสมบัติหลักของความสัมพันธ์ระหว่างคลาสใน Adapter	1. ความสัมพันธ์ Generalization ระหว่าง Class Target กับ Class Adapter โดยมีการสืบทอดคุณสมบัติจาก Class Target ไปยัง Class Adapter 2. ความสัมพันธ์ Generalization ระหว่าง Class Adaptee กับ Class Adapterโดยมีการสืบทอดคุณสมบัติจาก Class Adapteeไปยัง Class Adapter 3. ความสัมพันธ์ Association แบบมีทิศทาง ระหว่าง Class Client กับ Class Target โดยClass Client มีการเรียกใช้ Object จาก Class Target

3.3.1.2.3 เกณฑ์การตรวจทาน Template Method

จาก Design Pattern รูปที่ 10 สามารถนำมาสร้างเกณฑ์การตรวจทานของ Template Method ได้ดังตารางที่ 6

ตารางที่ 6 เกณฑ์การตรวจทานของ Template Method

เกณฑ์การตรวจทาน	คำอธิบาย
คุณสมบัติหลักของ Template Method	1. Class AbstractClass เป็น Class ที่ประกาศ Primitive Method และ Template Method ไว้โดยมีข้อกำหนดว่า Primitive Method ต้องเป็น Abstract Method เท่านั้น 2. Class ConcreteClass เป็น Sub class ของ Class AbstractClass 3. Class ConcreteClass เป็น Class ซึ่งทำหน้าที่ Implement Primitive Method ของ Class AbstracClass
คุณสมบัติหลักของความสัมพันธ์ระหว่างคลาสใน Template Method	1. ความสัมพันธ์ Generalization ระหว่าง Class Abstract Class กับ Class ConcreteClass โดยมีการสืบทอดคุณสมบัติจาก Class Abstract ไปยัง Class ConcreteClass

3.3.1.2.4 เกณฑ์การตรวจทาน Singleton

จาก Design Pattern รูปที่ 11 สามารถนำมาสร้างเกณฑ์การตรวจทานของ Singleton ได้ดังตารางที่ 7

ตารางที่ 7 เกณฑ์การตรวจทานของ Singleton

เกณฑ์การตรวจทาน	คำอธิบาย
คุณสมบัติหลักของ Singleton	1. Class Singleton เป็น Class ที่มี Method getInstance() ซึ่งใช้เพื่อการสร้าง Object โดย Method getInstance() จะทำหน้าที่ จำกัดจำนวนของ Object ที่จะถูกสร้างขึ้นได้
คุณสมบัติหลักของความสัมพันธ์ระหว่างคลาสใน Singleton	ไม่มี

3.3.1.2.5 เกณฑ์การตรวจทาน Facade

จาก Design Pattern รูปที่ 12 สามารถนำมาสร้างเกณฑ์การตรวจทานของ Facade ได้ดังตารางที่ 8

ตารางที่ 8 เกณฑ์การตรวจทานของ Facade

เกณฑ์การตรวจทาน	คำอธิบาย
คุณสมบัติหลักของ Facade	1. Class N เป็น Class ต่างๆ ในระบบที่เราสนใจ 2. Class Facade เป็น Class ที่มีความรู้ว่าแต่ละ Class ในระบบทำหน้าที่อะไร ให้ผลลัพธ์อะไร และจะรับคำสั่งจากภายนอกเพื่อไปสั่งงาน Class ต่างๆ ในระบบ
คุณสมบัติหลักของความสัมพันธ์ระหว่างคลาสใน Facade	1. ความสัมพันธ์ Association แบบมีทิศทาง ระหว่าง Class Facade กับ Class1 โดย Class Facade มีการเรียกใช้ Object จาก Class Class1 2. ความสัมพันธ์ Association แบบมีทิศทาง ระหว่าง Class Facade กับ Class2 โดย Class Facade มีการเรียกใช้ Object จาก Class Class2 3. ความสัมพันธ์ Association แบบมีทิศทาง ระหว่าง Class Facade กับ Class3 โดย Class Facade มีการเรียกใช้ Object จาก Class Class3 4. ความสัมพันธ์ Association แบบมีทิศทาง ระหว่าง Class Facade กับ Class4 โดย Class Facade มีการเรียกใช้ Object จาก Class Class4

3.3.1.2.6 เกณฑ์การตรวจทาน Proxy

จาก Design Pattern รูปที่ 13 สามารถนำมาสร้างเกณฑ์การตรวจทานของ Proxy ได้ดังตารางที่ 9

ตารางที่ 9 เกณฑ์การตรวจทานของ Proxy

เกณฑ์การตรวจทาน	คำอธิบาย
คุณสมบัติหลักของ Proxy	1.Class Subject เป็น Class Interface ที่จัดเตรียมไว้เพื่อ Implements Class RealSubject และ Class Proxy 2.Class RealSubject เป็น Class ที่ถูกเรียกใช้งานโดย Class Proxy โดยผ่านทาง Interface Subject

ตารางที่ 9 (ต่อ) เกณฑ์การตรวจทานของ Proxy

เกณฑ์การตรวจทาน	คำอธิบาย
	3.Class Proxy เรียกใช้งาน Class RealSubject ผ่าน Class Subject ได้ 4.Class Proxy เป็น Class ที่ทำหน้าที่เก็บรักษาการอ้างอิงที่ทำให้ Class Proxy สามารถเข้าถึง Class Real Subject ได้ 5.Class Proxy เป็น Class ที่จัดเตรียม Interface เพื่อการเข้าถึง Class Subject ซึ่งในการใช้งานจริง Class Subject จะถูกแทนที่ได้ด้วย Class RealSubject 6.Class Proxy เป็น Class ที่ควบคุมการเข้าถึง Class RealSubject และอาจทำหน้าที่ในการสร้างและลบ Class RealSubject ด้วย
คุณสมบัติหลักของความสัมพันธ์ระหว่างคลาสใน Proxy	1. ความสัมพันธ์ Generalization ระหว่าง Class Subject กับ Class Proxy โดยมีการสืบทอดคุณสมบัติจาก Class Subject ไปยัง Class Proxy 2. ความสัมพันธ์ Generalization ระหว่าง Class Subject กับ Class RealSubject โดยมีการสืบทอดคุณสมบัติจาก Class Subject ไปยัง Class RealSubject 3. ความสัมพันธ์ Association แบบมีทิศทาง ระหว่าง Class Proxy กับ Class RealSubject โดยClass Proxy มีการเรียกใช้ Object จาก Class RealSubject 4. ความสัมพันธ์ Association แบบมีทิศทาง ระหว่าง Class Client กับ Class Subject โดยClass Client มีการเรียกใช้ Object จาก Class Subject

3.3.2 **สร้างรายการตรวจทานจากข้อมูลเอ็กซ์เอ็มแอล**

จากเกณฑ์การตรวจทานข้างต้น งานวิจัยนี้จะดึงรายละเอียดแผนภาพคลาสจากไฟล์เอ็กซ์เอ็มแอลโดยค้นหาจากแท็กต่างๆ กล่าวคือ

ใน Main Class จะทำการสังเกตที่ชื่อของ Class, Attribute Class, Operation Class และ Operation Parameter โดยการตรวจสอบโครงสร้างคลาสจะสังเกตจากแท็ก สรุปได้ดังตารางที่ 10

ตารางที่ 10 เอ็กซ์เอ็มแอลแท็กสำหรับการตรวจสอบโครงสร้างคลาส

XML Elements	Attributes
UML:Class	Xmi.id , Class Name, Visibility, isSpecification, isRoot, isLeaf, isAbstract, isActive ,namespace , genezalization
UML:Generalization	Xmi.id , Generalization Name , Visibility, isSpecification , diseriminator , child , parent
UML:Attribute	Xmi.id , AttributeName ,Visibility , isSpecification , ownerScope ,changeable , targetScope , ordering ,type
UML:Operation	Xmi.id , OperationName ,Visibility, isSpecification , ownerScope isQuery , concurrency , isRoot , isLeaf , isAbstract , Specification
UML:Parameter	Xmi.id , ParameterName, Visibility, isSpecification, kind, type

Relation Class จะทำการสังเกตเริ่มต้นที่ชื่อความสัมพันธ์ และจะลงไปที่ Class โดย 1 Class จะประกอบไปด้วย จุดเริ่มต้น และจุดสิ้นสุด แสดงในตารางที่ 11 และ 12 ตามลำดับ

ตารางที่ 11 เอ็กซ์เอ็มแอลแท็กสำหรับการตรวจสอบความสัมพันธ์

XML Elements	Attribute
UML:Association	name , visibility , isSpecification , isRoot , isLeaf , isAbstract
UML:Association. connection	name , visibility , isSpecification, isNavigable ,ordering , aggregation, targetScope , changeability ,participant
Association.connection	name , visibility , isSpecification, isNavigable ,ordering , aggregation, targetScope , changeability ,participant

ตารางที่ 12 เอ็กซ์เอ็มแอลแท็กสำหรับการตรวจสอบประเภทตัวแปล

XML Elements	Attribute
UML:DataType	Xmi.id , name , Visibility , isSpecification , isRoot , isleaf , isAbstract
UML:Association. connection	name , visibility , isSpecification, isNavigable ,ordering , aggregation, targetScope , changeability ,participant

ตารางที่ 12 (ต่อ) เอ็กซ์เอ็มแอลแท็กสำหรับการตรวจสอบประเภทตัวแปล

XML Elements	Attribute
Association.connection	name , visibility , isSpecification, isNavigable ,ordering , aggregation, targetScope , changeability ,participant

จากข้อมูลตัวอย่าง Association Relation (รูปที่ 22) เราสามารถดึงรายละเอียดจากไฟล์เอ็กซ์เอ็มแอลที่มีความสัมพันธ์ในรูปแบบ Association ออกมาจาก Tag Element ได้ดังตารางที่ 13, 14, 15 และ 16 ตามลำดับ

ตารางที่ 13 ตัวอย่างการดึงข้อมูลเอ็กซ์เอ็มแอลของ Class B

โครงสร้างชื่อ	Details of Class B
Class	ClassName = 'ClassB' visibility = 'public' isSpecification = 'false' isRoot = 'true' isLeaf = 'true' isAbstract = 'false' isActive = 'false'
Attribute	Attribute name = 'iCount' visibility = 'private' isSpecification = 'false' ownerScope = 'instance' changeability = 'changeable' targetScope = 'instance' ordering = 'unordered' type = 'G.7'
Method	Operationname = 'doSomeThing' visibility = 'public' isSpecification = 'false' ownerScope = 'instance' isQuery = 'false' concurrency = 'sequential' isRoot = 'false' isLeaf = 'false' isAbstract = 'false' specification = '' Operationname = 'checkData' visibility = 'public' isSpecification = 'false' ownerScope = 'instance' isQuery = 'false' concurrency = 'sequential' isRoot = 'false' isLeaf = 'false' isAbstract = 'false' specification = ''

ตารางที่ 14 ตัวอย่างการดึงข้อมูลเอ็กซ์เอ็มแอลของ Class A

โครงสร้างชื่อ	*Details of Class A*
Class	ClassName = 'ClassA' visibility = 'public' isSpecification = 'false' isRoot = 'true' isLeaf = 'true' isAbstract = 'false' isActive = 'false'

ตารางที่ 14 (ต่อ) ตัวอย่างการดึงข้อมูลเอ็กซ์เอ็มแอลของ Class A

โครงสร้างชื่อ	*Details of Class A*
Attribute	Attribute name = 'sName' visibility = 'private' isSpecification = 'false' ownerScope = 'instance' changeability = 'changeable' targetScope = 'instance' ordering = 'unordered' type = 'G.7' Attribute name = ‘sSurname’ visibility = 'private' isSpecification = 'false' ownerScope = 'instance' changeability = 'changeable' targetScope = 'instance' ordering = 'unordered' type = 'G.7'
Method	Operationname = 'getName' visibility = 'public' isSpecification = 'false' ownerScope = 'instance' isQuery = 'false' concurrency = 'sequential' isRoot = 'false' isLeaf = 'false' isAbstract = 'false' specification = '' Parametername = 'getName.Return' visibility = 'public' isSpecification = 'false' kind = 'return' type = 'G.7' > Operation name = 'setName' visibility = 'public' isSpecification = 'false' ownerScope = 'instance' isQuery = 'false' concurrency = 'sequential' isRoot = 'false' isLeaf = 'false' isAbstract = 'false' specification = '' Operationname = 'getSurname' visibility = 'public' isSpecification = 'false' ownerScope = 'instance' isQuery = 'false' concurrency = 'sequential' isRoot = 'false' isLeaf = 'false' isAbstract = 'false' specification = '' Parametername = 'getSurname.Return' visibility = 'public' isSpecification = 'false' kind = 'return' type = 'G.7' > Operation name = 'setSurname' visibility = 'public' isSpecification = 'false' ownerScope = 'instance' isQuery = 'false' concurrency = 'sequential' isRoot = 'false' isLeaf = 'false' isAbstract = 'false' specification = ''

ตารางที่ 15 ตัวอย่างการดึงข้อมูลเอ็กซ์เอ็มแอลของความสัมพันธ์รหว่าง Class A กับ Class B

Details of Relation
xmi.id = 'G.1' name = '' visibility = 'public' isSpecification = 'false' isRoot = 'false' isLeaf = 'false' isAbstract = 'false' xmi.id = 'G.2' name = '' visibility = 'public' isSpecification = 'false' isNavigable = 'false' ordering = 'unordered' aggregation = 'none' targetScope = 'instance' changeability = 'changeable' xmi.id = 'G.3' name = '' visibility = 'public' isSpecification = 'false' isNavigable = 'false' ordering = 'unordered' aggregation = 'none' targetScope = 'instance' changeability = 'changeable'

ตารางที่ 16 ตัวอย่างการดึงข้อมูลเอ็กซ์เอ็มแอลประเภทข้อมูลตัวแปล

Details of Return Type
xmi.id = 'G.7' name = 'Integer' visibility = 'public' isSpecification = 'false' isRoot = 'false' isLeaf = 'false' isAbstract = 'false' xmi.id = 'G.8' name = 'String' visibility = 'public' isSpecification = 'false' isRoot = 'false' isLeaf = 'false' isAbstract = 'false'

3.4 อ่านชุดคำสั่งภาษาจาวาที่ต้องการตรวจทานเข้าระบบ

การทำงานจะอาศัย Regular Expression ซึ่งคำสั่งจะอธิบายดังตารางที่ 17 เพื่อค้นหาชื่อ Class, Attribute, Method Prameter และ Relationship ที่ปรากฏอยู่ในไฟล์ชุดคำสั่งจาวา

ตารางที่ 17 อธิบายคำสั่ง Regular Expression

Regular Expression	**คำอธิบาย**
^	คำ/อักษรที่อยู่หน้าเครื่องหมายนี้ ต้องเป็นคำขึ้นต้นของข้อความที่นำมาตรวจสอบเช่น “^การ” เป็นการกำหนดว่า คำที่นำมาตรวจสอบต้องขึ้นต้นด้วยคำว่า การ เช่น “การทำดี” “การบ้าน” เป็นต้น คำพวกนี้จะผ่านการทดสอบ

ตารางที่ 17 (ต่อ) อธิบายคำสั่ง Regular Expression

Regular Expression	คำอธิบาย
$	คำ/อักษรที่อยู่หน้าเครื่องหมายนี้ ต้องอยู่ตอนท้ายของข้อความที่นำมาตรวจสอบ เช่น “มา$” จะถือว่าคำต่อไปนี้ถูกตามเงื่อนไข “ตามา” “ขอขมา” หรือแม้แต่คำว่า “หมา” แต่คำว่า “ทำดี” จะไม่ผ่าน เพราะไม่ได้ลงท้ายด้วยคำว่า “มา” ตามเงื่อนไขนั่นเอง
+	คำ/อักษรที่อยู่หน้าเครื่องหมายนี้ ต้องมีปรากฏในคำที่นำมาตรวจสอบ อย่างน้อย 1 ตัว เช่น “ท+” จะถือว่าคำต่อไปนี้ผ่านการตรวจสอบ เช่น “ทองจุล” “วันทนา” “ถนนหนทางทุกแห่ง”
?	คำ/อักษรที่อยู่หน้าเครื่องหมายนี้ อาจจะมีปรากฏในคำที่นำมาตรวจสอบ หรือไม่ก็ได้ ถ้ามีจะมีกี่ตัวก็ได้ “ก?ข+$” หมายถึง อาจจะมีด้วยตัว ก และอักษรตัวสุดท้ายต้องมีตัว ข อย่างน้อย 1 ตัว (เครื่องหมาย + แสดงว่ามีอย่างน้อย 1 และ เครื่องหมาย $ แสดงว่าเป็นตัวสุดท้าย)
*	เหมือนกับ ?
\s	ช่องว่าง หรือ whitespace
.	ใช้แทนตัวอักษรอะไรก็ได้ ● “ก.[0-9]” หมายถึง ตัว ก ตามด้วยตัวอักษรอะไรก็ได้ และต่อด้วยเลขอารบิค เลข 0-9 ● “^.{3}$” หมายถึง ต้องมีตัวอักษรเพียง 3 ตัวเท่านั้น เป็นตัวเลข ตัวอักษรภาษาไทย ภาษาอังกฤษ ได้ทั้งนั้น
[ ]	ใช้ระบุตำแหน่งในคำว่า ในตำแหน่งนี้จะมีตัวอักษรอะไรได้บ้าง เช่น ● “[นร]” เป็นการกำหนดว่า คำที่นำมาตรวจสอบ ต้องเป็นตัว น หรือ ตัว ร เท่านั้นจึงจะผ่าน มีความหมายเช่นเดียวกับ “น|ร” ● “[ก-ค]” เป็นการบอกว่า คำที่นำมาจะต้องเป็น ตัว ก ข ค เท่านั้น เช่น ในกรณีเลขประจำตัวที่ขึ้นต้นด้วย ก ข หรือ ค เท่านั้น ถ้าพิมพ์ตัวแรกเป็นตัวอักษรตัวอื่นก็แสดงว่าพิมพ์ผิด เราจะเขียนได้ดังนี้ ^[ก-ค]

ตารางที่ 17 (ต่อ) อธิบายคำสั่ง Regular Expression

Regular Expression	**คำอธิบาย**
	• "^[a-zA-Z]" เป็นการบอกว่า คำที่นำมาตรวจสอบต้องขึ้นต้นด้วยตัวอักษร จะเป็นตัวเล็ก คือ a ถึง z หรือ ตัวใหญ่ คือ A ถึง Z ก็ได้ • "[0-9๐-๙]%" เป็นการบอกว่า ให้มีตัวเลข 1 ตัว เลขอะไรก็ได้ เลข 0 ถึง เลข 9 เป็นได้ทั้งเลขไทยและอารบิค ต่อด้วยเครื่องหมาย % • [ก-๙] ตัว ก ถึง ฮ รวมทั้งสระทุกตัว และ ตัวเลขไทย ๐ ถึง ๙ • [0-9๐-๙] เลข 0-9 ทั้งเลขไทยและฝรั่ง • ^[0-9๐-๙]+$ ให้มีเฉพาะตัวเลข 0-9 เลขไทยหรือเลขฝรั่งก็ได้ แต่ห้ามมีตัวอักษรใด ๆ • "^[กข]{3}[-][0-9]$" ขึ้นต้นด้วยตัว ก หรือ ข จำนวน 3 ตัว ต่อด้วยเครื่องหมาย – และจบด้วยตัวเลขอารบิค เลข 0-9 เช่น "กขก-5" "กกก-3" เป็นต้น สิ่งต่อไปนี้จะไม่ผ่านหรือเป็นเท็จ เช่น "กกกขข" เพราะ ตัวที่ 4 ไม่ใช้เครื่องหมาย – และตัวสุดท้ายไม่ใช่ตัวเลข "ขขข-๘" ตัวเลขสุดท้ายเป็นเลขไทย ไม่ว่าตัวอักษร หรือสัญลักษณ์ใด ๆ ที่อยู่ภายในเครื่องหมาย [ ] จะกลายเป็นสัญลักษณ์ธรรมดา เช่น + กลายเป็นเครื่องหมายบวก แทนที่จะหมายถึงว่า ต้องมีตัวอักษรอย่างน้อย 1 ตัว
{ }	แสดงจำนวนครั้งที่ซ้ำกัน เช่น • "กข{2}" หมายถึงให้มีตัว ข จำนวน 2 ตัว เช่น "กขข" • "กข{2,} หมายถึงให้มีตัว ข อย่างน้อย 2 ตัว เช่น "กขขขข" • "กข{3,5} หมายถึงให้มีตัว ข จำนวน 3-5 ตัวเท่านั้น คือ "กขขข" "กขขขข" และ "กขขขขข"

ตารางที่ 17 (ต่อ) อธิบายคำสั่ง Regular Expression

Regular Expression	คำอธิบาย
( )	ใช้รวมกลุ่มเข้าด้วยกันเป็นส่วนเดียวกัน เช่น ● “ก(ขค)*” หมายถึง ตัว ก และอาจจะตามด้วยตัว ขค หรือไม่มีตัว ขค ก็ได้ เครื่องหมาย * แสดงว่าจะมีหรือไม่ก็ได้ ● “ก(ขค){1,5}” หมายถึง ตัว ก แล้วจะตามด้วย ขค จำนวน 1-5 ชุด เช่น “กขคขคขค” หรือ “กขคขค” ก็ได้
|	เสนอทางเลือกอย่างใดอย่างหนึ่ง เช่น ● “การ|ความ” เป็นการบอกว่า จะใช้คำว่า การ หรือ ความ ก็ได้ ● “(ก|ขค)งจ” เช่น กงจ หรือ ขคงจ ก็ได้

3.4.1 การค้นหารายละเอียดของคลาส

การค้นหารายละเอียดและความสัมพันธ์ของคลาสสามารถแบ่งออกเป็น 5 แบบดังนี้

3.4.1.1 การค้นหาชื่อคลาส

รูปแบบ Regular Expression เพื่อใช้ในการค้นหาชื่อคลาส แสดงดังรูปที่ 30

```
^public(\\s+)(abstract|interface|class)(\\s+)[a-zA-Z0-9].+\\{$
```

รูปที่ 30 Regular Expression ในการค้นหาชื่อคลาส

เพื่อค้นหาชื่อคลาสจากลักษณะการประกาศคลาสแบบต่างๆ ต่อไปนี้

3.4.1.1.1 public abstract class “ClassName”

3.4.1.1.2 public interface “ClassName”

3.4.1.1.3 public class “ClassName”

3.4.1.1.4 public class “ClassName” implements “SuperClass”

3.4.1.1.5 public class “ClassName” extends “SuperClass”

3.4.1.2 ค้นหาข้อมูล Attribute ของ Class

รูปแบบ Regular Expression เพื่อใช้ในการค้นหา Attribute ของ Class แสดงดังรูปที่ 31

```
(private|public|protected)*(\\s+)*([a-zA-Z0-9]+|static)(\\s+)[a-zA-Z0-9].+;$
```

รูปที่ 31 Regular Expression ในการค้นหาข้อมูล Attribute

รูปแบบต่างๆ ของชุดคำสั่งโปรแกรมที่มีการประกาศชื่อ Attribute คือ

3.4.1.2.1 [private/ public/ protected] DataType “AttributeName”

3.4.1.2.2 [private/ public/ protected] static DataType “AttributeName”

3.4.1.2.3 DataType “AttributeName”

3.4.1.3 ค้นหาข้อมูล Method ของ Class

รูปแบบ Regular Expression เพื่อใช้ในการค้นหาข้อมูลการประกาศ Method ทั่วไปและการประกาศเฉพาะเจาะจงในรูปแบบ Abstract Method ของ Class แสดงดังรูปที่ 32 และรูปที่ 33 ตามลำดับ

```
(public|private|protected)(\\s+)([a-zA-Z0-9]+(\\s+)*|(static\\s+)*[a-zA-Z0-9]+(\\s+)[a-zA-Z0-9]+(\\s+)*)\\(.+\\{$
```

รูปที่ 32 Regular Expression ในการค้นหาข้อมูล Method ทั่วไป

```
(public|private|protected)(\\s+)([a-zA-Z0-9]+(\\s+)*|(abstract\\s+)*[a-zA-Z0-9]+(\\s+)[a-zA-Z0-9]+(\\s+)*)\\(.+\\;$
```

รูปที่ 33 Regular Expression ในการค้นหาข้อมูล Abstract Method

รูปแบบต่างๆ ของชุดคำสั่งโปรแกรมที่มีการประกาศชื่อ Method คือ

3.4.1.3.1 [private/ public/ protected] DataType “MethodName”

3.4.1.3.2 [private/ public/ protected] [static/ abstract] DataType “MethodName”

3.4.1.3.3 Public “MethodName”

3.4.1.4 ค้นหาข้อมูล Method Parameter ของ Class

รูปแบบ Regular Expression ในการค้นหาข้อมูล Method Parameter ทั่วไปและรูปแบบที่เจาะจงเฉพาะ Interface Method Parameter กับ Abstract Method Parameter แสดงดังรูปที่ 34 และรูปที่ 35 ตามลำดับ

```
.+\\(([a-zA-Z0-9]+|(\\s+)*)\\s[a-zA-Z0-9]+((\\s+)|,)*.+\\{$
```

รูปที่ 34 Regular Expression ในการค้นหาข้อมูล Method Parameter ทั่วไป

```
.+\\(([a-zA-Z0-9]+|(\\s+)*)\\s[a-zA-Z0-9]+((\\s+)|,)*.+\\;$
```

รูปที่ 35 Regular Expression ในการค้นหาข้อมูล Interface Method Parameter และ Abstract Method Parameter

รูปแบบต่าง ๆ ของชุดคำสั่งโปรแกรมที่มีการประกาศ Method Parameter คือ

3.4.1.4.1 () ไม่มีการประกาศพารามิเตอร์

3.4.1.4.2 (DataType “ParameterName” [,])

3.4.1.5 การค้นหาข้อมูลความสัมพันธ์ของ Class

รูปแบบ Regular Expression ในการค้นหาข้อมูลความสัมพันธ์แบบการประกาศ New Instance และการเรียกใช้งาน Method โดยผ่านชื่อคลาสโดยตรงระหว่าง Class ต้นทาง ไปยัง Class ปลายทาง แสดงดังรูปที่ 36 และรูปที่ 37 ตามลำดับ

```
.+new(\\s+)(“Class Name”)(\\s+)*\\(.+;$
```

รูปที่ 36 Regular Expression ในการค้นหาแบบ New Instance

```
.+("Class Name")\\..+
```

รูปที่ 37 Regular Expression ในการค้นหาแบบใช้ชื่อคลาสเรียก Method โดยตรง

โดยการความสัมพันธ์ระหว่าง Class จะมีเพียง 2 มุมมองได้แก่

มุมมองที่ 1 Inheritance ความสัมพันธ์แบบการสืบทอดคุณสมบัติของ Class แม่

มุมมองที่ 2 Association เนื่องการเขียนชุดคำสั่งโปรแกรมจะมองแค่ระดับการเรียกใช้งานเท่านั้น ดังนั้นการค้นหาความสัมพันธ์จึงไม่ได้ลงถึงขั้น Aggegation และ Composition

รูปแบบการค้นหาความสัมพันธ์ระหว่าง Class ต้นทาง ไปยัง Class ปลายทาง

3.4.1.5.1 public class “ClassName” implements “SuperClass”

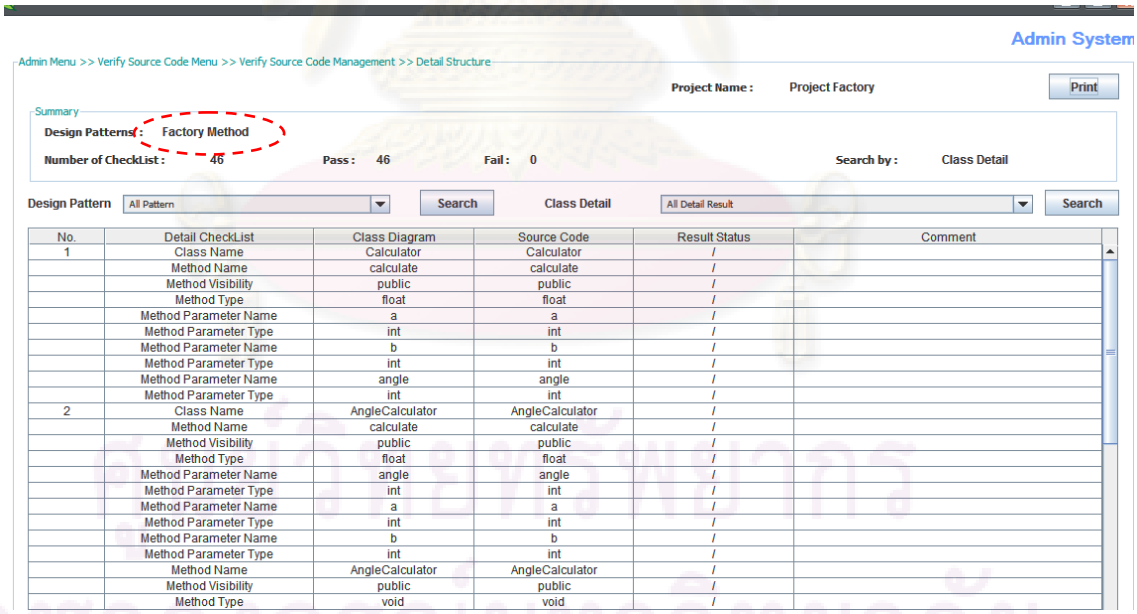
3.4.1.5.2 public class “ClassName” extends “SuperClass”

3.4.1.5.3 การ “New Instance” หรือมีการเรียกใช้ ClassName มาใช่งานเช่น โดยตรงเช่น ClassName MyClass = New ClassName(); หรือ ClassName.doSomeThing(); เป็นต้น

3.5 สร้างรายงานสรุปผลการตรวจทานชุดคำสั่ง

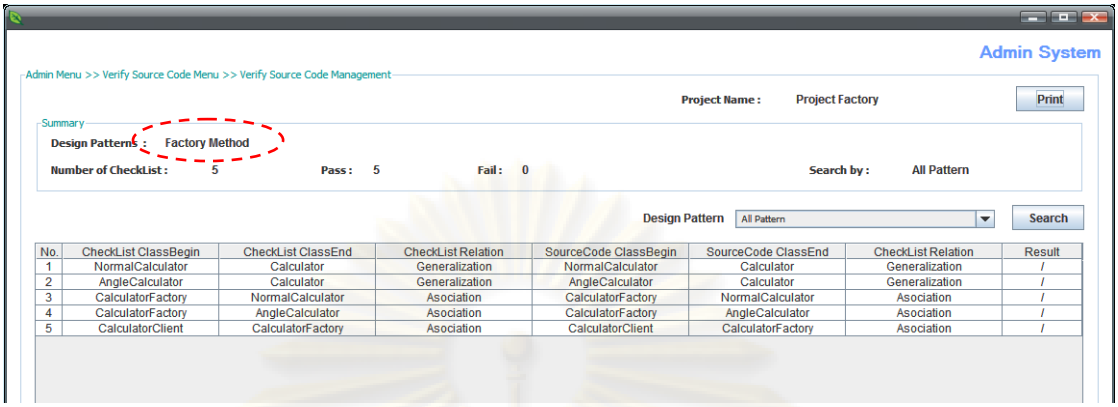
กระบวนการตรวจสอบชุดคำสั่งโปรแกรมภาษาจาวานั้น ระบบจะทำการวิ่งอ่าน ไฟล์ชุดคำสั่งโปรแกรมที่ละ Class โดยจะทำการตรวจสอบ โครงสร้างของ Class, Attribute, Methods, Parameter รวมถึงความสัมพันธ์ระหว่าง Class ที่อยู่ภายในระบบโครงการ จัดเก็บข้อมูลลงฐานข้อมูลเพื่อทำการ ตรวจสอบว่าตรงตามชุดข้อมูลตรวจทานหรือไม่ ถ้าการตรวจสอบชุดคำสั่งโปรแกรมไม่เป็นไม่ตามการออกแบบระบบจะทำการแจ้งเตือนไปยังนักพัฒนาโปรแกรมเพื่อทำการแก้ไขให้ถูกต้องต่อไป

3.5.1 **ตัวอย่างรายงานการตรวจทาน** Design Pattern: Factory Method แสดงดังรูปที่ 38 และรูปที่ 39 ตามลำดับ



No.	Detail CheckList	Class Diagram	Source Code	Result Status	Comment
1	Class Name	Calculator	Calculator	/	
	Method Name	calculate	calculate	/	
	Method Visibility	public	public	/	
	Method Type	float	float	/	
	Method Parameter Name	a	a	/	
	Method Parameter Type	int	int	/	
	Method Parameter Name	b	b	/	
	Method Parameter Type	int	int	/	
	Method Parameter Name	angle	angle	/	
	Method Parameter Type	int	int	/	
2	Class Name	AngleCalculator	AngleCalculator	/	
	Method Name	calculate	calculate	/	
	Method Visibility	public	public	/	
	Method Type	float	float	/	
	Method Parameter Name	angle	angle	/	
	Method Parameter Type	int	int	/	
	Method Parameter Name	a	a	/	
	Method Parameter Type	int	int	/	
	Method Parameter Name	b	b	/	
	Method Parameter Type	int	int	/	
	Method Name	AngleCalculator	AngleCalculator	/	
	Method Visibility	public	public	/	
	Method Type	void	void	/	

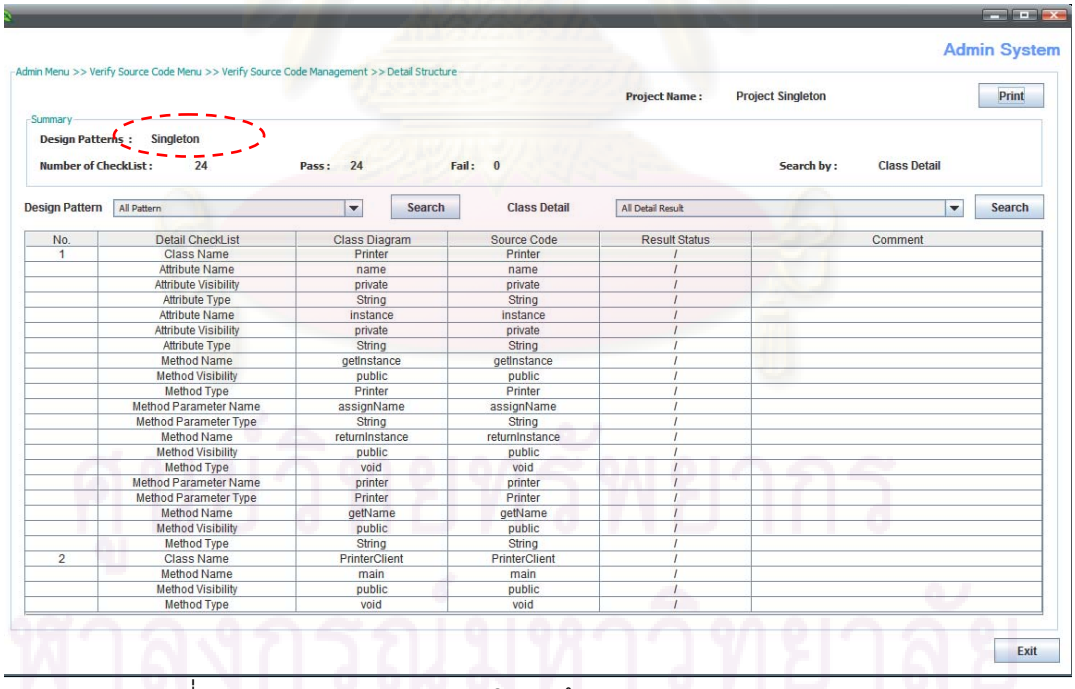
รูปที่ 38 รายงานการตรวจทานโครงสร้าง Design Pattern: Factory Method



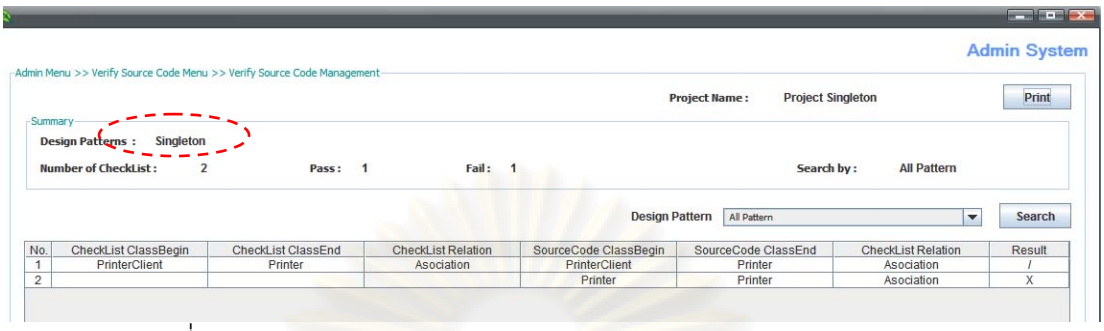
รูปที่ 39 รายงานการตรวจทานความสัมพันธ์ Design Pattern: Factory Method

จากรายงานการตรวจทานชุดคำสั่งในรูปที่ 38 และรูปที่ 39 พบว่าระบบสามารถตรวจทาน Design Pattern: Factory Method ตามเกณฑ์การตรวจทานได้อย่างถูกต้อง

3.5.2 **ตัวอย่างรายงานการตรวจทาน** Design Pattern: Singleton แสดงดังรูปที่ 40 และรูปที่ 41



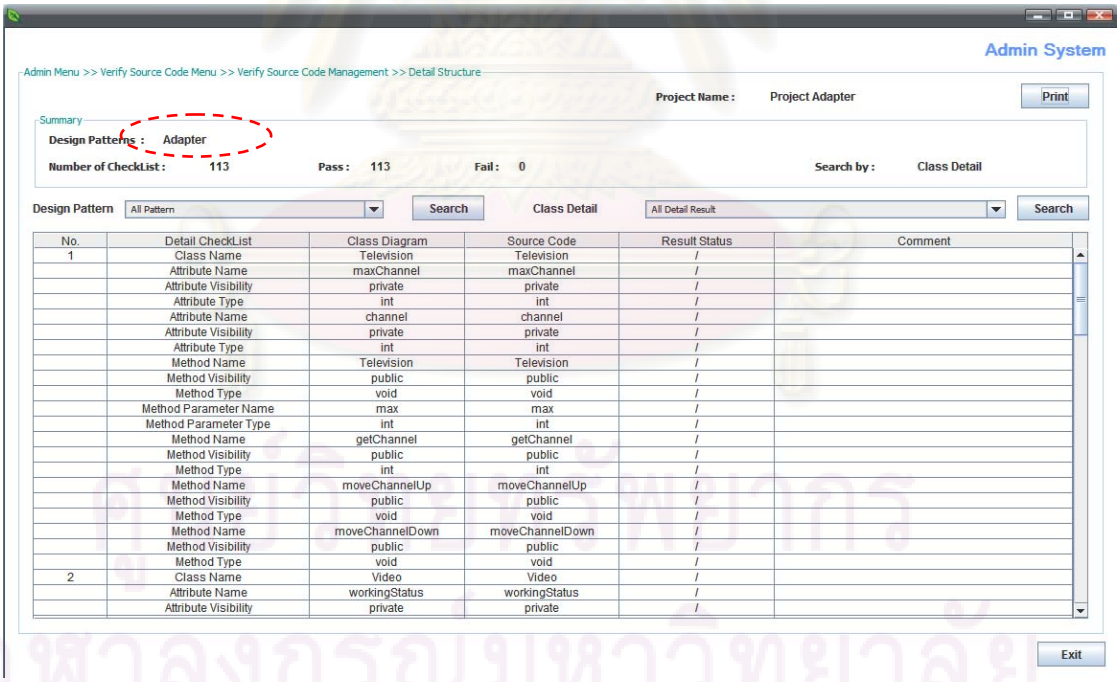
รูปที่ 40 รายงานการตรวจทานโครงสร้าง Design Pattern: Singleton



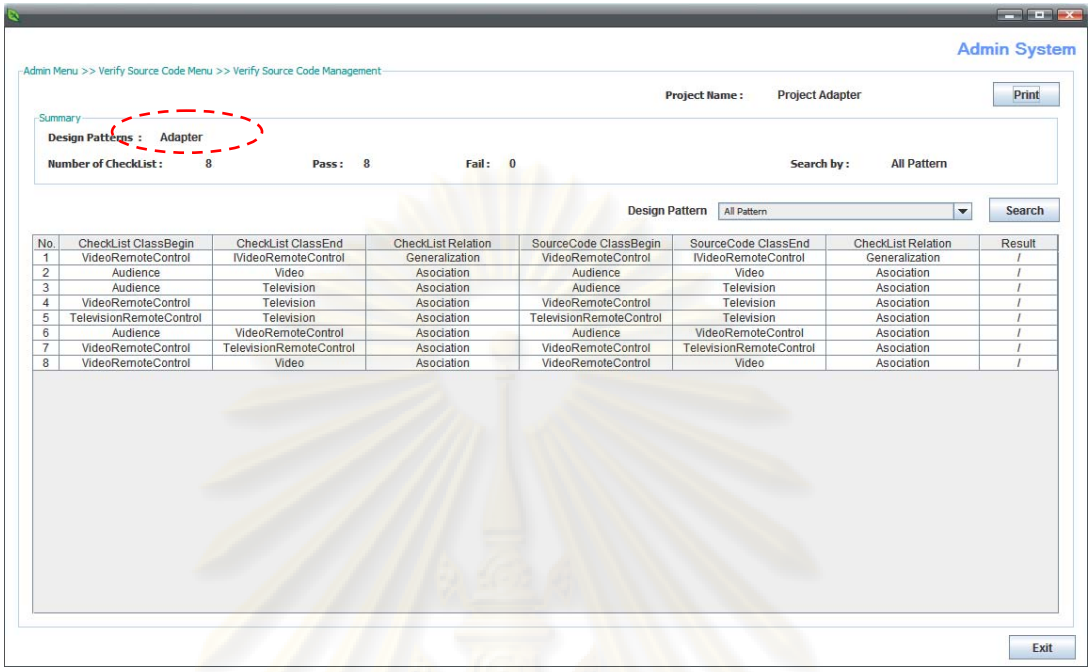
รูปที่ 41 รายงานการตรวจทานความสัมพันธ์ Design Pattern: Singleton

จากรายงานการตรวจทานชุดคำสั่งในรูปที่ 40 และรูปที่ 41 พบว่าระบบสามารถตรวจทาน Design Pattern: Singleton ตามเกณฑ์การตรวจทานได้อย่างถูกต้อง โดยในรายงานการตรวจทานชุดคำสั่งในรูปที่ 41 จะมีการแสดงผลลัพธ์ที่ไม่ถูกต้อง 1 ตัว เพราะเกณฑ์การตรวจสอบในตารางที่ 7 ไม่มีความสัมพันธ์ของคลาสใน Design Pattern: Singleton

3.5.3 **ตัวอย่างรายงานการตรวจทาน** Design Pattern: Adapter แสดงดังรูปที่ 42 และรูปที่ 43



รูปที่ 42 รายงานการตรวจทานโครงสร้าง Design Pattern: Adapter

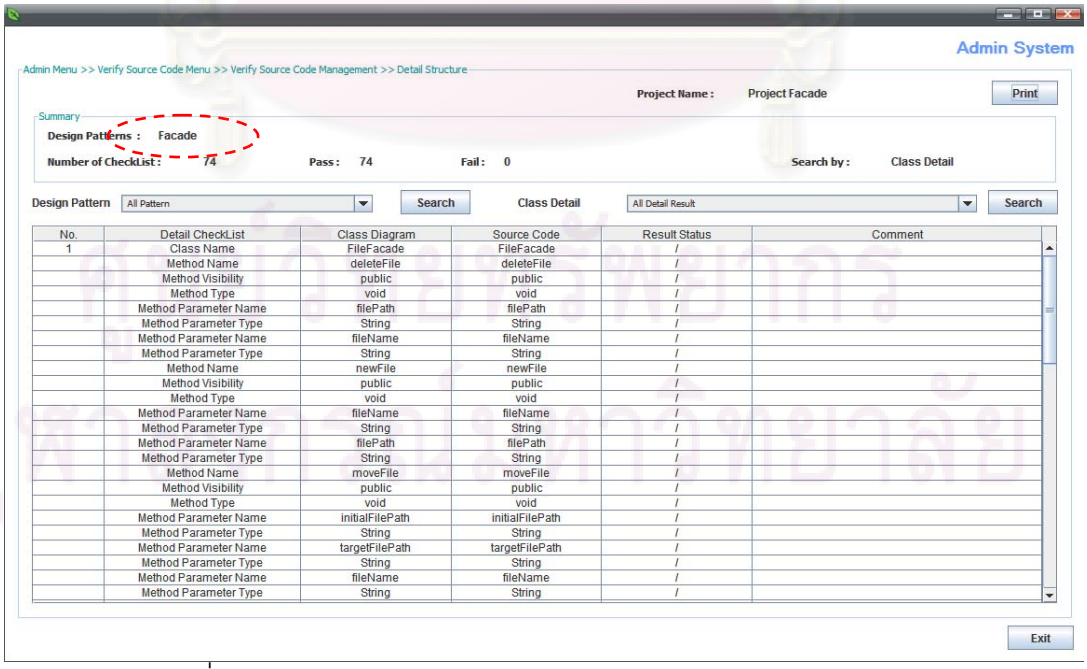


No.	CheckList ClassBegin	CheckList ClassEnd	CheckList Relation	SourceCode ClassBegin	SourceCode ClassEnd	CheckList Relation	Result
1	VideoRemoteControl	IVideoRemoteControl	Generalization	VideoRemoteControl	IVideoRemoteControl	Generalization	/
2	Audience	Video	Association	Audience	Video	Association	/
3	Audience	Television	Association	Audience	Television	Association	/
4	VideoRemoteControl	Television	Association	VideoRemoteControl	Television	Association	/
5	TelevisionRemoteControl	Television	Association	TelevisionRemoteControl	Television	Association	/
6	Audience	VideoRemoteControl	Association	Audience	VideoRemoteControl	Association	/
7	VideoRemoteControl	TelevisionRemoteControl	Association	VideoRemoteControl	TelevisionRemoteControl	Association	/
8	VideoRemoteControl	Video	Association	VideoRemoteControl	Video	Association	/

รูปที่ 43 รายงานการตรวจทานความสัมพันธ์ Design Pattern: Adapter

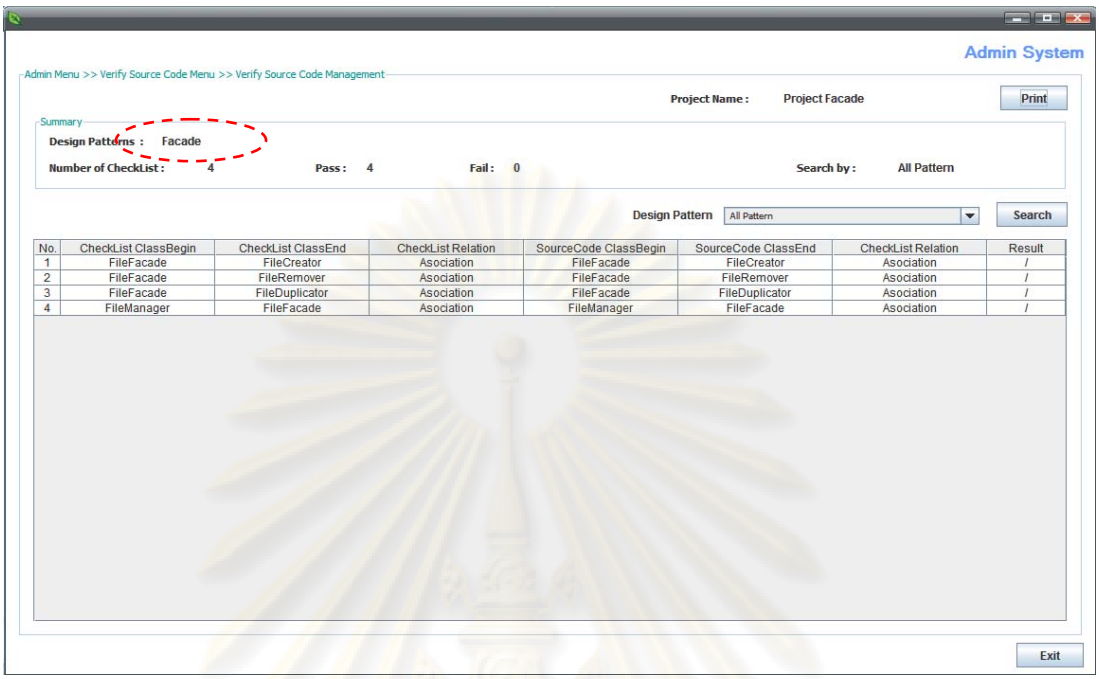
จากรายงานการตรวจทานชุดคำสั่งในรูปที่ 42 และรูปที่ 43 พบว่าระบบสามารถตรวจทาน Design Pattern: Adapter ตามเกณฑ์การตรวจทานได้อย่างถูกต้อง

3.5.4 **ตัวอย่างรายงานการตรวจทาน** Design Pattern: Facade แสดงดังรูปที่ 44 และรูปที่ 45



No.	Detail CheckList	Class Diagram	Source Code	Result Status	Comment
1	Class Name	FileFacade	FileFacade	/	
	Method Name	deleteFile	deleteFile	/	
	Method Visibility	public	public	/	
	Method Type	void	void	/	
	Method Parameter Name	filePath	filePath	/	
	Method Parameter Type	String	String	/	
	Method Parameter Name	fileName	fileName	/	
	Method Parameter Type	String	String	/	
	Method Name	newFile	newFile	/	
	Method Visibility	public	public	/	
	Method Type	void	void	/	
	Method Parameter Name	fileName	fileName	/	
	Method Parameter Type	String	String	/	
	Method Parameter Name	filePath	filePath	/	
	Method Parameter Type	String	String	/	
	Method Name	moveFile	moveFile	/	
	Method Visibility	public	public	/	
	Method Type	void	void	/	
	Method Parameter Name	initialFilePath	initialFilePath	/	
	Method Parameter Type	String	String	/	
	Method Parameter Name	targetFilePath	targetFilePath	/	
	Method Parameter Type	String	String	/	
	Method Parameter Name	fileName	fileName	/	
	Method Parameter Type	String	String	/	

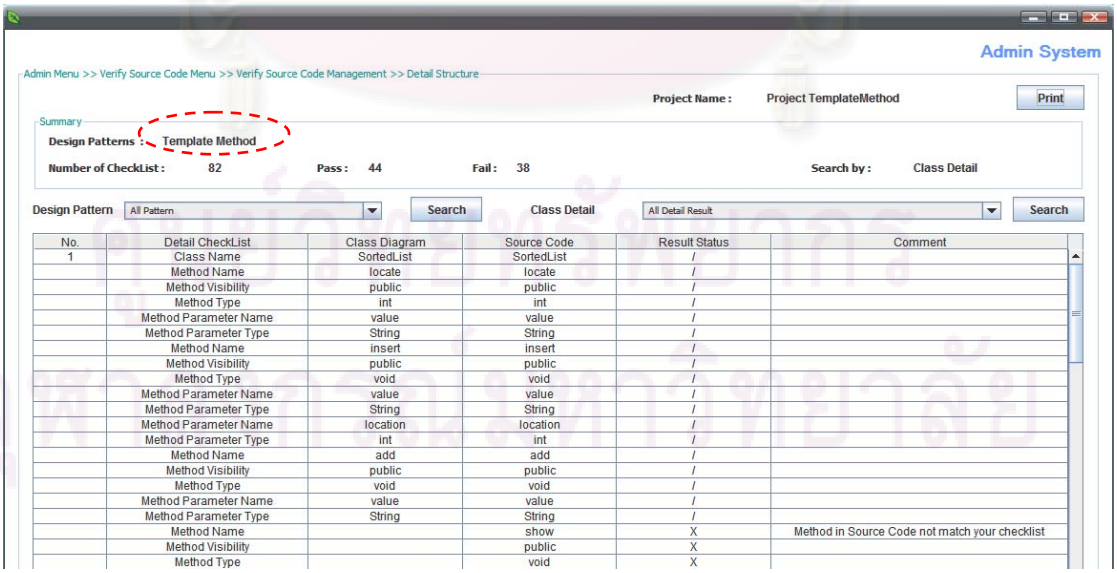
รูปที่ 44 รายงานการตรวจทานโครงสร้าง Design Pattern: Facade



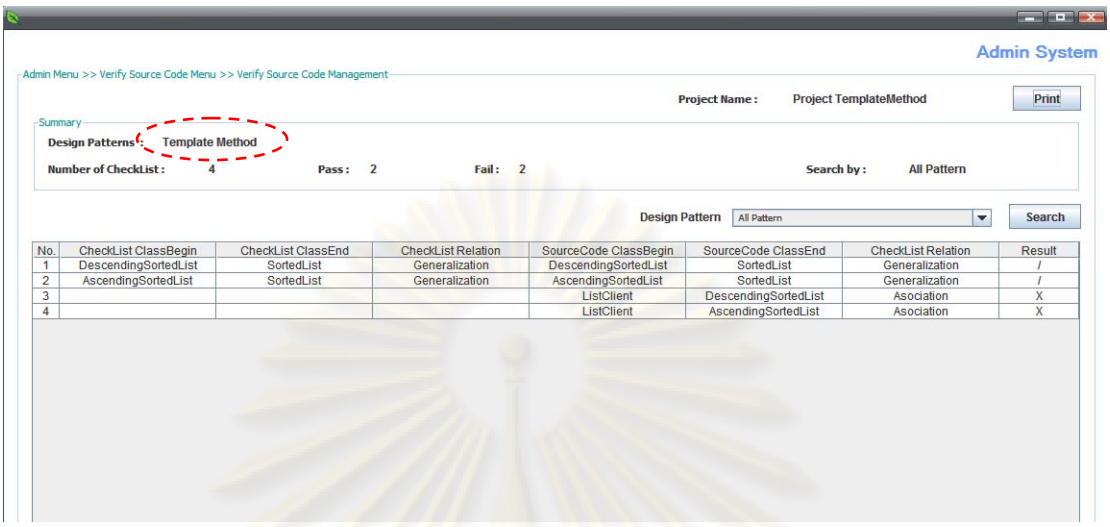
รูปที่ 45 รายงานการตรวจทานความสัมพันธ์ Design Pattern: Facade

จากรายงานการตรวจทานชุดคำสั่งในรูปที่ 44 และ 45 พบว่าระบบสามารถตรวจทาน Design Pattern: Facade ตามเกณฑ์การตรวจทานได้อย่างถูกต้อง

3.5.5 **ตัวอย่างรายงานการตรวจทาน** Design Pattern: Template Method แสดงดังรูปที่ 46 และรูปที่ 47



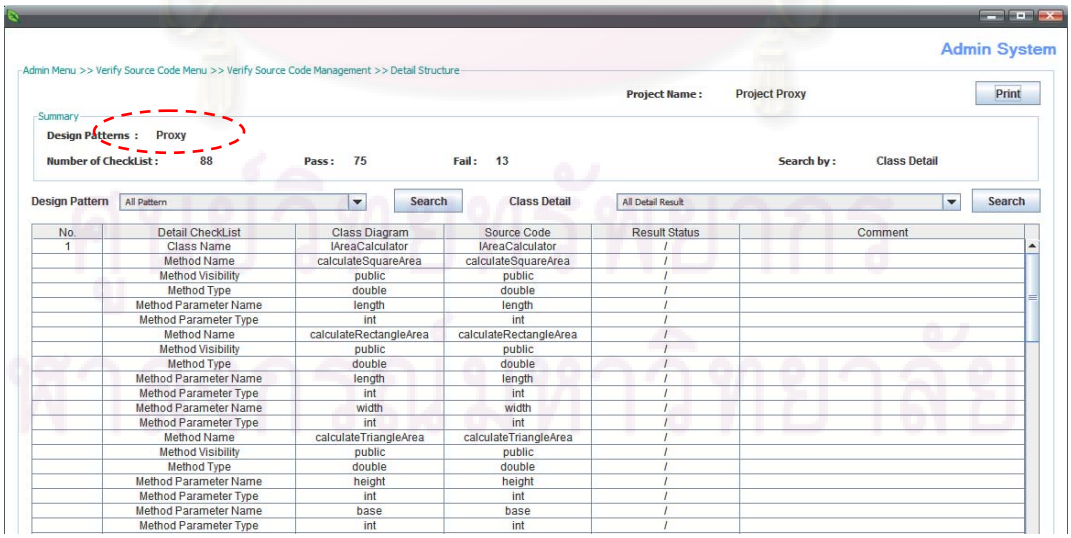
รูปที่ 46 รายงานการตรวจทานโครงสร้าง Design Pattern: Template Method



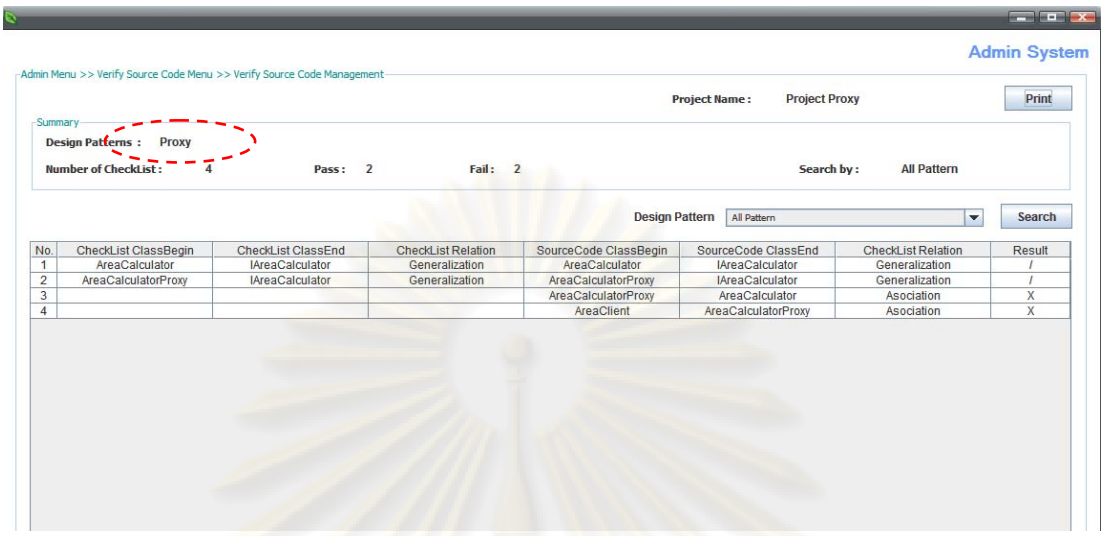
รูปที่ 47 รายงานการตรวจทานความสัมพันธ์ Design Pattern: Template Method

จากรายงานการตรวจทานชุดคำสั่งในรูปที่ 46 และ 47 พบว่าระบบสามารถตรวจทาน Design Pattern: Template Method เป็นไปตามเกณฑ์การตรวจทานของ Template Method โดยผลจากรายงานพบว่ามีข้อบกพร่องเกิดขึ้น เนื่องจากนักพัฒนาโปรแกรมมีการเขียนชุดคำสั่งภาษาเกินกว่าการออกแบบของนักวิเคราะห์ระบบ

3.5.6 **ตัวอย่างรายงานการตรวจทาน Design Pattern: Proxy** แสดงดังรูปที่ 48 และรูปที่ 49



รูปที่ 48 รายงานการตรวจทานโครงสร้าง Design Pattern: Proxy



รูปที่ 49 รายงานการตรวจทานความสัมพันธ์ Design Pattern: Proxy

จากรายงานการตรวจทานชุดคำสั่งในรูปที่ 48 และ 49 พบว่าระบบสามารถตรวจทาน Design Pattern: Proxy เป็นไปตามเกณฑ์การตรวจทานของ Proxy โดยผลจากรายงานพบว่ามีข้อบกพร่องเกิดขึ้น เนื่องจากนักพัฒนาโปรแกรมมีการเขียนชุดคำสั่งภาษาเกินกว่าการออกแบบของนักวิเคราะห์ระบบ

บทที่ 4
การพัฒนาระบบ

จากการศึกษาและออกแบบขั้นตอนการดำเนินงานการพัฒนาซอฟต์แวร์ตามที่ได้นำเสนอในบทที่ 3 ผู้เสนอวิทยานิพนธ์ได้ทำการสรุปหน้าที่ความรับผิดชอบ รวมถึงขั้นตอนการทำงานดังต่อไปนี้

4.1 ความต้องการที่เป็นฟังก์ชันการทำงาน

ความต้องการที่เป็นฟังก์ชันการทำงาน (Functional Requirements) ของระบบตัวตรวจทานชุดคำสั่งอัตโนมัติ มีดังตารางที่ 18

ตารางที่ 18 ความต้องการที่เป็นฟังก์ชันการทำงาน

รหัส	ชื่อ	คำอธิบาย
F01	หน้าแรก	แสดงรายละเอียดหน้าจอหลักของผู้ใช้งานทั้งหมด
F02	การจัดการข้อมูลพื้นฐานของระบบ	จัดการค้นหา เพิ่ม/แก้ไข/ลบ ข้อมูลพื้นฐานของระบบ
F03	การจัดการข้อมูลผู้ใช้งานระบบ	จัดการค้นหา เพิ่ม/แก้ไข/ลบ ข้อมูลผู้ใช้งานระบบ
F04	การจัดการข้อมูล Design Pattern ของระบบ	จัดการค้นหา เพิ่ม/แก้ไข/ลบ ข้อมูล Design Pattern ของระบบ
F05	การจัดการข้อมูลโครงการของระบบ	จัดการค้นหา เพิ่ม/แก้ไข/ลบ โครงการของระบบ
F06	การจัดสรรนักพัฒนาระบบเพื่อทำการพัฒนาโครงการ	จัดการ เพิ่ม/แก้ไข/ลบ ข้อมูลนักพัฒนาระบบที่ประจำอยู่ในโครงการ
F07	การจัดการข้อมูลการร้องขอของนักพัฒนาระบบ	แสดงรายละเอียด ยอมรับ/ยกเลิก ข้อมูลการร้องขอของนักพัฒนาระบบ

ตารางที่ 18 (ต่อ) ความต้องการที่เป็นฟังก์ชันการทำงาน

รหัส	ชื่อ	คำอธิบาย
F08	การจัดการข้อมูลรายการตรวจทาน	แสดงรายละเอียด เพิ่ม/แก้ไข/ลบ ข้อมูลรายการตรวจทานของระบบ
F09	การค้นหา Design Pattern จากข้อมูลรายการตรวจทาน	แสดงผลลัพธ์การค้นหาข้อมูล Design Pattern
F10	การจัดการข้อมูลรายการตรวจสอบชุดคำสั่งโปรแกรมภาษาจาวา	แสดงรายละเอียด เพิ่ม/แก้ไข/ลบ ข้อมูลรายการตรวจชุดคำสั่งโปรแกรมภาษาจาวา
F11	การจัดการตรวจสอบรายการตรวจทานกับข้อมูลชุดตรวจสอบคำสั่งโปรแกรมภาษาจาวา	แสดงรายละเอียดข้อมูลผลการตรวจสอบ
F12	การแจ้งเตือนตรวจสอบ	แสดงสถานะการแจ้งเตือนไปยังนักพัฒนาระบบ
F13	การแสดงผล ทางรายงาน	แสดงข้อมูลต่าง ๆ ผ่านรายงาน

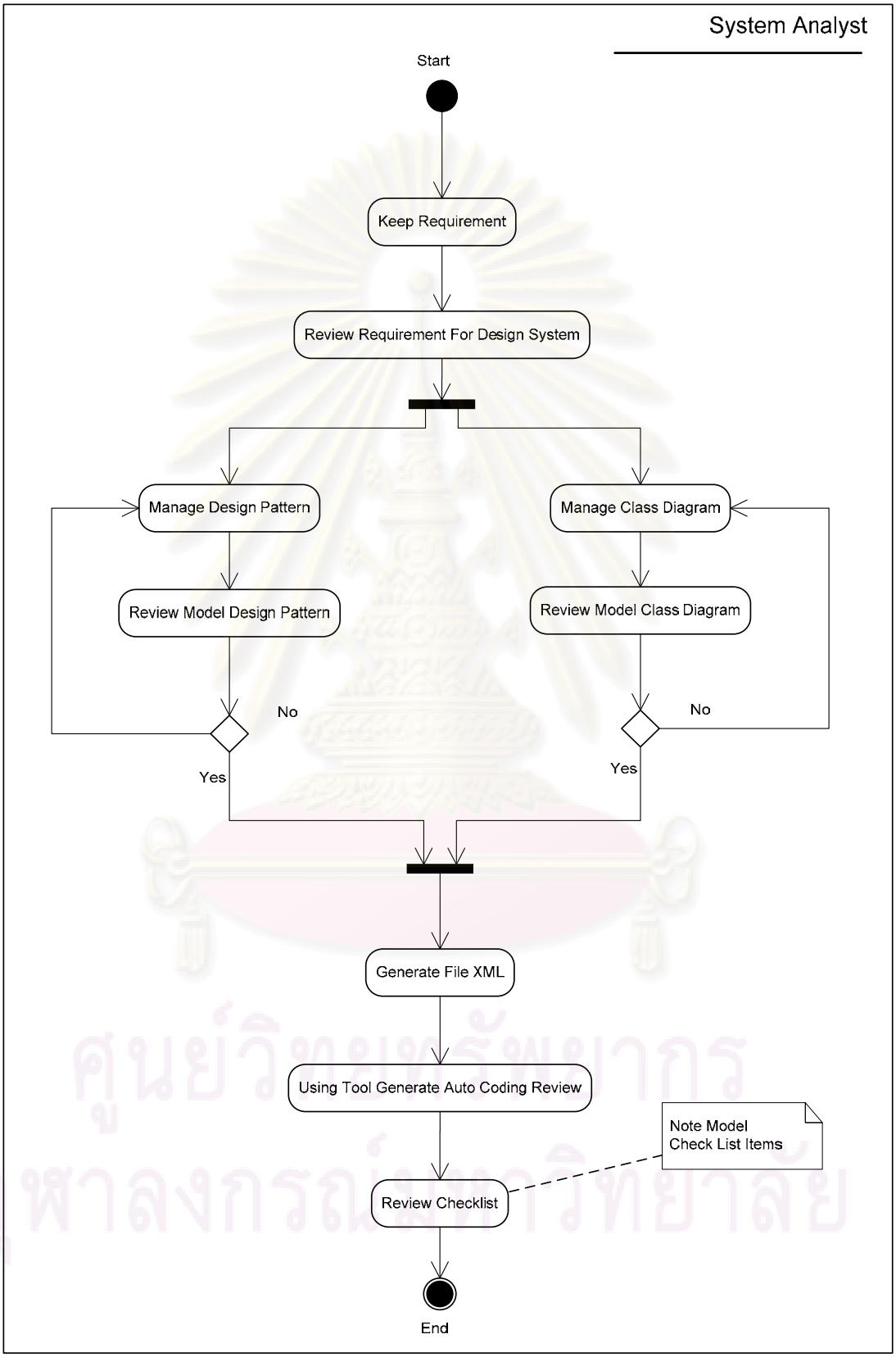
4.2 บทบาทหน้าที่ของผู้เกี่ยวข้องกับระบบ

งานวิจัยนี้ได้แบ่งกลุ่มบทบาทหน้าที่ของผู้เกี่ยวข้องกับระบบ ดังนี้

4.2.1 นักวิเคราะห์ หรือผู้ดูแลระบบ (System Analyst) รูปภาพประกอบรูปที่ 50

- วิเคราะห์ปัญหาและทำการออกแบบ Class Diagram ผ่านเครื่องมือ Rational Rose
- สร้างไฟล์ข้อมูล XML จากเครื่องมือ Rational Rose

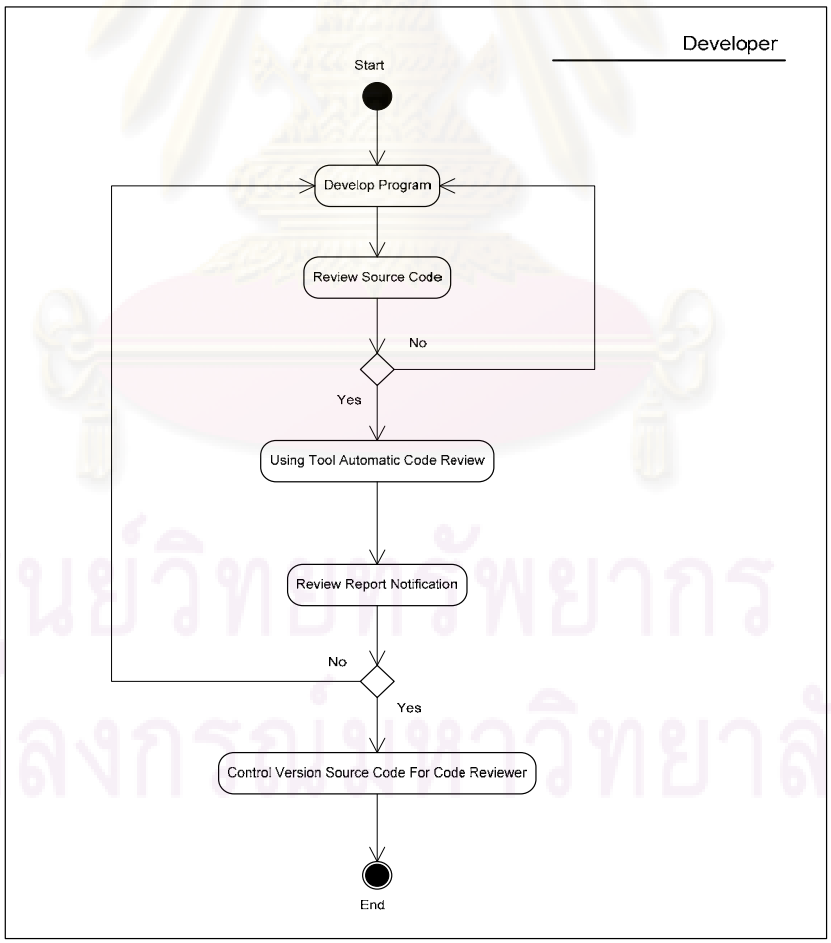
- การจัดการค่าตั้งต้นของระบบ เช่น กำหนดชื่อความสัมพันธ์ของ Class กำหนดประเภทที่ต้องการตรวจสอบ กำหนดขอบเขตการมองเห็นของ Attribute เป็นต้น (รายละเอียดภาคผนวก Application)
- การจัดการโปรเจคที่ต้องการตรวจสอบให้กับระบบ
- การจัดการพนักงานที่ใช้งานเครื่องให้กับระบบ
- การจัดการแบ่งกลุ่มพนักงานกับโปรเจคภายในระบบ
- การจัดการข้อมูลตั้งต้น Design Pattern ให้กับระบบ
- การจัดการข้อมูลรายการตรวจทาน (Check List) และ ชุดคำสั่งโปรแกรมภาษาจาวาของนักพัฒนาระบบ
- ตรวจสอบ และบันทึกข้อมูลชุดคำสั่งโปรแกรม ของพนักงานที่มีการร้องขอทำการแก้ไขชุดโปรแกรมภายในโปรเจคได้
- ตรวจสอบความถูกต้องของ ชุดข้อมูล Design Pattern ,ชุดข้อมูลตรวจทานและชุดคำสั่งโปรแกรมของพนักงานได้
- เรียกดูและตรวจสอบชุดข้อมูลตรวจทานกับชุดคำสั่งโปรแกรมของนักพัฒนาระบบได้
- แจ้งเตือนการตรวจสอบแก่นักพัฒนาระบบ
- ออกรายงานต่าง ๆ ได้เช่น รายงานพนักงานในแต่ละโปรเจค , รายงานผลการตรวจสอบของพนักงาน เป็นต้น



รูปที่ 50 แผนภาพกิจกรรมการทำงานของนักวิเคราะห์ระบบ

4.2.2 นักพัฒนาระบบ (Developer) (รูปที่ 51)

- พัฒนาโปรแกรมตาม Design และ Program Specification ตามที่นักวิเคราะห์ระบบออกแบบไว้
- จัดการข้อมูลส่วนตัวได้
- ตรวจสอบโปรเจคที่ตนเองรับผิดชอบได้
- ทำความเข้าใจเกี่ยวกับ Design Pattern ได้
- ทำการร้องขอให้นักวิเคราะห์ระบบทำการบันทึกชุดคำสั่งโปรแกรม เข้าสู่ระบบและทำการตรวจสอบได้
- ตรวจสอบผลที่ได้รับจากนักวิเคราะห์ระบบ และทำการแก้ไขชุดคำสั่งโปรแกรมจากข้อมูลรายงานได้
- ออกรายงานต่างๆ ได้ เช่น รายงานโปรเจคที่ได้มอบหมาย รายงานผลการตรวจสอบชุดคำสั่งโปรแกรมของตนเองได้



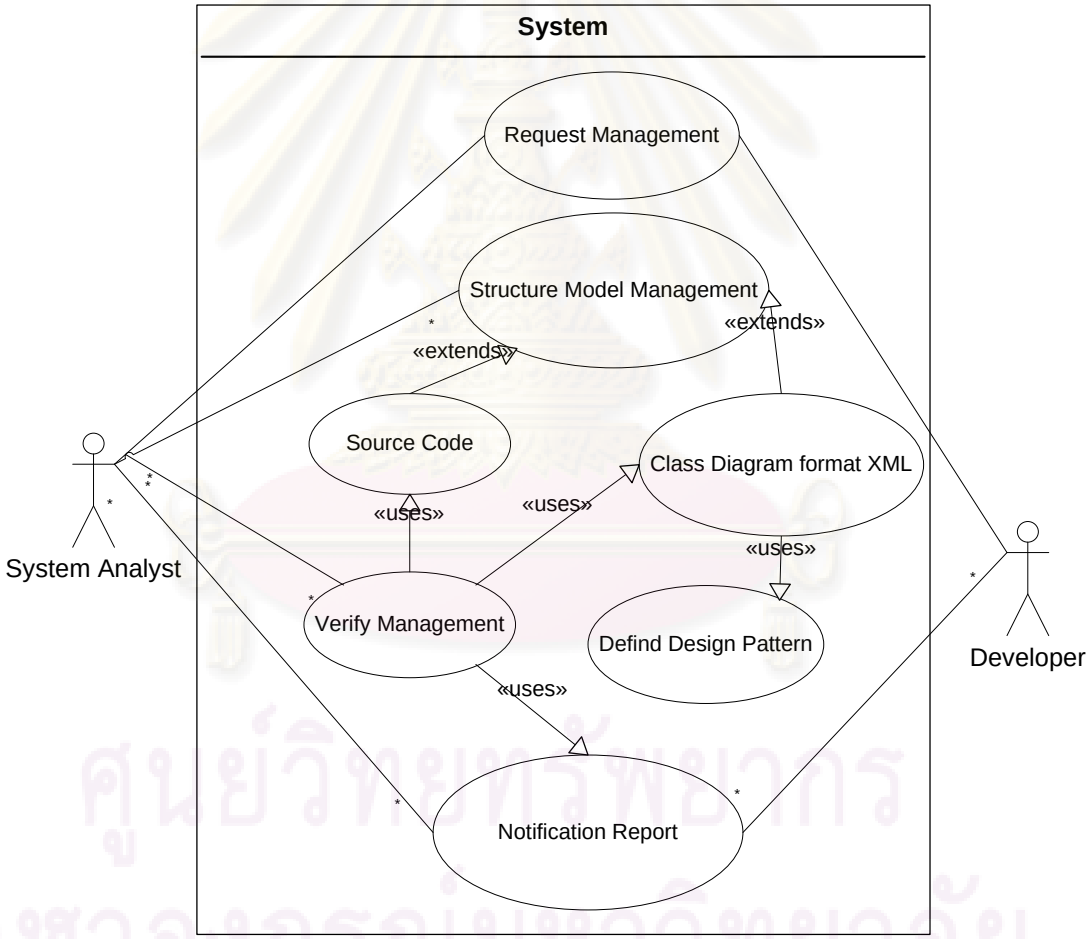
รูปที่ 51 แผนภาพกิจกรรมการทำงานของนักพัฒนาระบบ

4.3 การออกแบบการพัฒนาระบบ

จากการวิเคราะห์ฟังก์ชันการทำงานต่างๆ ของระบบ ผู้วิจัยได้ออกแบบแผนภาพยูสเคส รวมถึงฐานข้อมูลเพื่อแสดงการทำงานหลัก และการเก็บข้อมูลของระบบ โดยมีการอธิบายหัวข้อต่างๆ ดังต่อไปนี้

4.3.1 แผนภาพยูสเคส (Use Case Diagram)

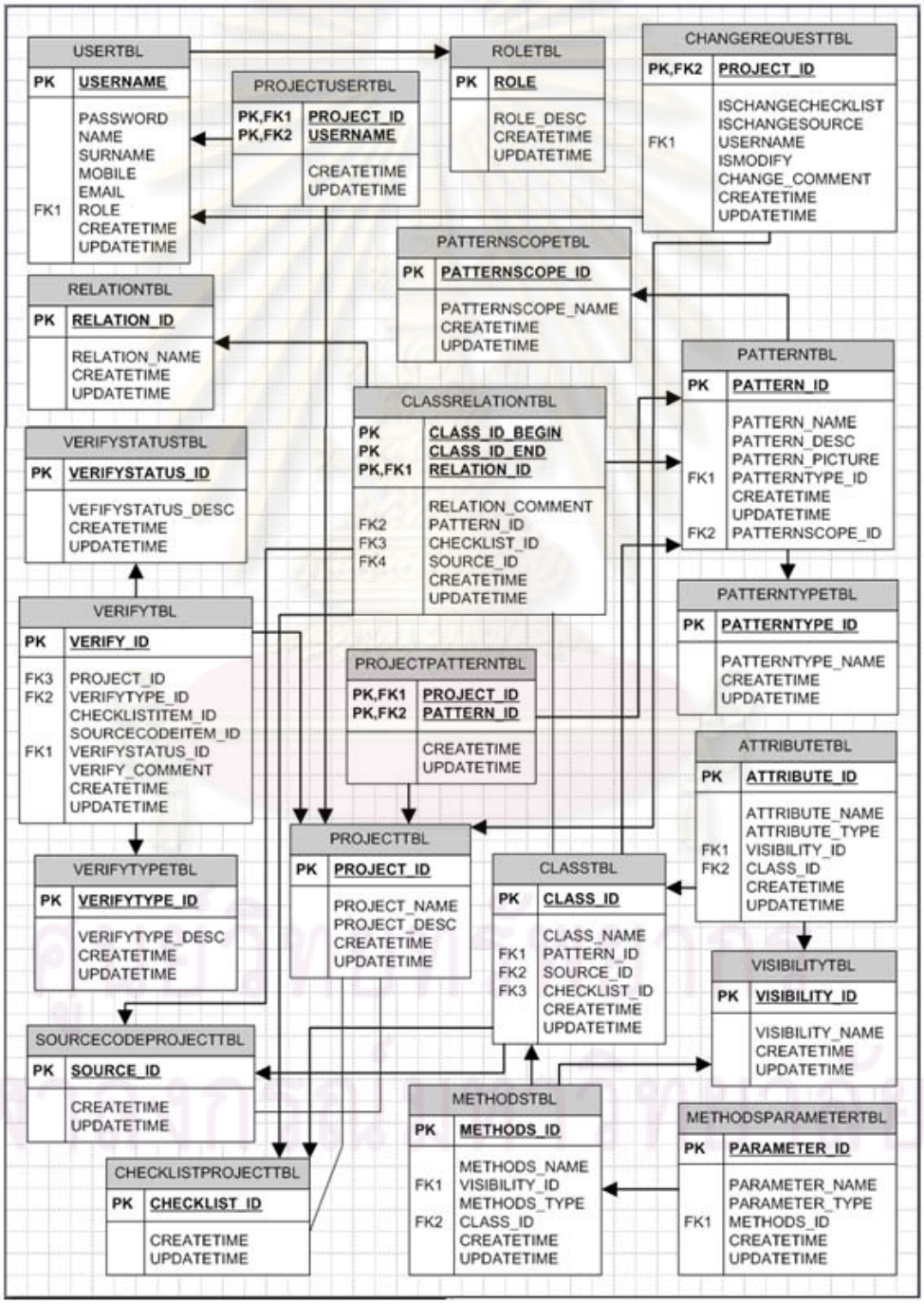
แผนภาพยูสเคสนำมาใช้อธิบายหน้าที่การทำงานหลักของระบบแสดงดังรูปที่ 52 โดยตารางคำอธิบายการทำงานจะแสดงในภาคผนวก ก



รูปที่ 52 แผนภาพยูสเคสการพัฒนาระบบการทำงานหลักของตัวตรวจทานชุดคำสั่งอัตโนมัติ

4.3.2 การออกแบบฐานข้อมูล

โครงสร้างและความสัมพันธ์ระหว่างข้อมูลภายในฐานข้อมูลของระบบตัวตรวจทานชุดคำสั่งอัตโนมัติ สามารถแสดงด้วยแผนภาพอีอาร์ ดังรูปที่ 53 สำหรับรายละเอียดของพจนานุกรมข้อมูล แสดงในภาคผนวก ข



รูปที่ 53 การออกแบบฐานข้อมูลระบบตัวตรวจทานชุดคำสั่งอัตโนมัติด้วยแผนภาพอีอาร์

4.4 สภาพแวดล้อมและเครื่องมือที่ใช้ในการพัฒนา

สภาพแวดล้อมที่ใช้ในการพัฒนาระบบมีสภาพแวดล้อมทางด้านฮาร์ดแวร์และซอฟต์แวร์ดังต่อไปนี้

ฮาร์ดแวร์

1. หน่วยประมวลผล อินเทลเพนเทียม 4 2.2 กิกะเฮิร์ต (Pentium 4 2.2 GHz.)
2. หน่วยความจำ (RAM) 2 กิกะไบต์ (2 GB)
3. ฮาร์ดดิสก์ (Hard Disk) 80 กิกะไบต์ (80 GB)

ซอฟต์แวร์

1. ระบบปฏิบัติการ วินโดวส์เอ็กซ์พี โปรเฟสชั่นนอล (Windows XP Professional)
2. โปรแกรมประมวลผลจาวา 6.0 (Java Runtime 6.0)
3. เครื่องมือพัฒนาโปรแกรมจาวา เน็ตบีนส์ เวอร์ชั่น 6.5 (Net Beans IDE 6.5)
4. โปรแกรมสร้างรายงาน iReport 3.7.1 สำหรับโปรแกรมจาวา เน็ตบีนส์
5. โปรแกรมระบบฐานข้อมูลออราเคิล 10g (Oracle 10g)

4.5 การติดตั้งซอฟต์แวร์ในการพัฒนาระบบ

เมื่อเตรียมเครื่องมือสำหรับการพัฒนาระบบเรียบร้อยแล้ว ขั้นต่อมาจึงเป็นส่วนของการติดตั้งเครื่องมือทั้งหมดลงในเครื่องคอมพิวเตอร์ที่ใช้พัฒนาระบบ โดยมีลำดับการติดตั้งเครื่องมือเป็นไปตามขั้นตอนต่อไปนี้

1. ติดตั้งระบบปฏิบัติการ วินโดวส์เอ็กซ์พี โปรเฟสชั่นนอล
2. ติดตั้งโปรแกรมระบบฐานข้อมูลออราเคิล 10g
3. สร้างฐานข้อมูลระบบ codereview
4. ติดตั้งเครื่องมือพัฒนาโปรแกรมจาวา เน็ตบีนส์เวอร์ชั่น 6.5
5. ติดตั้งโปรแกรมประมวลผลจาวา 6.0

4.6 การพัฒนาส่วนต่อประสานผู้ใช้

การพัฒนาในส่วนนี้จะใช้เทคโนโลยี Swing ของภาษาจาวาเพื่อช่วยในการหน้าจอการทำงาน โดยการนำเทคโนโลยี Swing มาช่วยในการออกแบบ และสร้างแอพพลิเคชั่นนั้น ทำให้หน้าจอการใช้งานมีความสวยงานและกระชับมากกว่าเทคโนโลยี AWT ทำให้ผู้ใช้งานสามารถใช้งานแอพพลิเคชั่นได้อย่างสบายตา

ผู้ใช้งานสามารถใช้งานและทำความเข้าใจกับระบบแอพพลิเคชั่นได้ง่ายเนื่องรูปแบบการใช้งานมีขั้นตอนที่ชัดเจน

บทที่ 5
การทดสอบระบบ

การทดสอบระบบตัวตรวจทานชุดคำสั่งอัตโนมัติ มีจุดประสงค์เพื่อตรวจทานข้อผิดพลาดของชุดคำสั่งโปรแกรมภาษาจาวา ที่ไม่ตรงตามการออกแบบ (Design Phase) โดยการตรวจสอบจะทำการมุ่งเน้นไปที่ผลลัพธ์จากเครื่องมือว่าตรงตามชุดข้อมูลตรวจทาน (Checklist) หรือไม่ ซึ่งเครื่องมือจะทำการแสดงผลออกมาเป็นรายงานเพื่อให้นักพัฒนาโปรแกรมนำไปตรวจสอบ และแก้ไขชุดคำสั่งโปรแกรมของตนเองต่อไป

5.1 ขั้นตอนปฏิบัติและผลการทดสอบระบบด้วยกรณีทดสอบ

การดำเนินการทดสอบใช้หลักการทดสอบหน้าที่การทำงาน (Black Box Testing) ตามกรณีทดสอบที่ได้ออกแบบไว้ โดยจะแบ่งการทดสอบออกเป็น 2 ส่วนมีขั้นตอนปฏิบัติและผลการทดสอบดังตารางที่ 19, 20, 21, 22, 23, 24 และ 25 ตามลำดับ

5.1.1 ทดสอบการทำงานฟังก์ชันพื้นฐานของระบบ

ตารางที่ 19 การเข้าสู่ระบบ และลำดับขั้นตอนการทดสอบระบบ

ลำดับ	การทดสอบ	คำอธิบาย	ผลการทดสอบที่คาดหวัง	ผลการทดสอบจริง
1.	เข้าสู่ระบบ	ทำการล็อคอินเพื่อเข้าใช้งานระบบ	หากรหัสผู้ใช้และรหัสผ่านถูกต้องให้เข้าใช้งานได้ ถ้าผิดพลาดต้องไม่สามารถเข้าใช้งานได้	ถูกต้อง

ตารางที่ 20 การจัดการข้อมูล Design Pattern และลำดับขั้นตอนการทดสอบระบบ

ลำดับ	การทดสอบ	คำอธิบาย	ผลการทดสอบที่คาดหวัง	ผลการทดสอบจริง
1.	สร้างข้อมูล Design Pattern ตั้งต้น	ทำการสร้างข้อมูล Design Pattern ลงในระบบ	สามารถแสดงข้อมูลชื่อ Design Pattern	ถูกต้อง
2.	แก้ไขข้อมูล Design Pattern ตั้งต้น	ทำการแก้ไขข้อมูล Design Pattern	สามารถแสดงรายการแก้ไขข้อมูล Design Pattern	ถูกต้อง
3.	ลบข้อมูล Design Pattern ตั้งต้น	ทำการลบข้อมูล Design Pattern	สามารถลบรายการข้อมูล Design Pattern ที่ทำการเลือกได้	ถูกต้อง
4.	จัดการโครงสร้าง Design Pattern	ทำการเลือก Design Pattern เพื่อจัดการโครงสร้างภายใน	แสดงข้อมูล หน้าจอ Class Management โดยมีการอ้างถึงข้อมูล Design Pattern ที่เลือกอีกด้วย	ถูกต้อง
5.	สร้างข้อมูล Class ของ Design Pattern	ทำการสร้างข้อมูล Class ของ Design Pattern ลงในระบบ	สามารถแสดงข้อมูลชื่อ Class ของ Design Pattern	ถูกต้อง
6.	แก้ไขข้อมูล Class ของ Design Pattern	ทำการแก้ไขข้อมูล Class ของ Design Pattern	สามารถแสดงรายการแก้ไขข้อมูล Class ของ Design Pattern	ถูกต้อง
7.	ลบข้อมูล Class ของ Design Pattern	ทำการลบข้อมูล Class ของ Design Pattern	สามารถลบรายการข้อมูล Class ของ Design Pattern ที่ทำการเลือกได้	ถูกต้อง

ตารางที่ 20 (ต่อ) การจัดการข้อมูล Design Pattern และลำดับขั้นตอนการทดสอบระบบ

ลำดับ	การทดสอบ	คำอธิบาย	ผลการทดสอบที่คาดหวัง	ผลการทดสอบจริง
8.	สร้างข้อมูล Attribute ของ Class	ทำการสร้างข้อมูล Attribute ของ Class ที่อยู่ใน Design Pattern ลงในระบบ	สามารถแสดงข้อมูลชื่อ Attribute ของ Class ที่อยู่ใน Design Pattern ได้	ถูกต้อง
9.	แก้ไขข้อมูล Attribute ของ Class	ทำการแก้ไขข้อมูล Attribute ของ Class ที่อยู่ใน Design Pattern	สามารถแสดงรายการแก้ไขข้อมูล Attribute ของ Class ที่อยู่ใน Design Pattern ได้	ถูกต้อง
10.	ลบข้อมูล Attribute ของ Class	ทำการลบข้อมูล Attribute ของ Class ที่อยู่ใน Design Pattern	สามารถลบรายการข้อมูล Attribute ของ Class ที่อยู่ใน Design Pattern ได้	ถูกต้อง
11.	สร้างข้อมูล Method ของ Class	ทำการสร้างข้อมูล Method ของ Class ที่อยู่ใน Design Pattern ลงในระบบ	สามารถแสดงข้อมูลชื่อ Method ของ Class ที่อยู่ใน Design Pattern ได้	ถูกต้อง
12.	แก้ไขข้อมูล Method ของ Class	ทำการแก้ไขข้อมูล Method ของ Class ที่อยู่ใน Design Pattern	สามารถแสดงรายการแก้ไขข้อมูล Method ของ Class ที่อยู่ใน Design Pattern ได้	ถูกต้อง
13.	ลบข้อมูล Method ของ Class	ทำการลบข้อมูล Method ของ Class ที่อยู่ใน Design Pattern	สามารถลบรายการข้อมูล Method ของ Class ที่อยู่ใน Design Pattern ได้	ถูกต้อง

ตารางที่ 20 (ต่อ) การจัดการข้อมูล Design Pattern และลำดับขั้นตอนการทดสอบระบบ

ลำดับ	การทดสอบ	คำอธิบาย	ผลการทดสอบที่คาดหวัง	ผลการทดสอบจริง
14.	จัดการโครงสร้าง Parameter ของ Method	ทำการเลือก Method เพื่อจัดการ โครงสร้าง Parameter ภายใน	แสดงข้อมูล หน้าจอ Parameter Management โดยมีการอ้างถึงข้อมูล Method ที่เลือกอีกด้วย	ถูกต้อง
15.	สร้างข้อมูล Parameter ของ Method	ทำการสร้างข้อมูล Parameter ของ Method ที่อยู่ใน Class ลงในระบบ	สามารถแสดงข้อมูลชื่อ Parameter ของ Method ที่อยู่ใน Class ได้	ถูกต้อง
16.	แก้ไขข้อมูล Parameter ของ Method	ทำการแก้ไขข้อมูล Parameter ของ Method ที่อยู่ใน Class	สามารถแสดงรายการแก้ไขข้อมูล Parameter ของ Method ที่อยู่ใน Class ได้	ถูกต้อง
17.	ลบข้อมูล Parameter ของ Method	ทำการลบข้อมูล Parameter ของ Method ที่อยู่ใน Class	สามารถลบรายการข้อมูล Parameter ของ Method ที่อยู่ใน Class ได้	ถูกต้อง
18.	ผูกความสัมพันธ์ระหว่าง Class	ทำการเลือก Class ต้นทาง Class ปลายทาง และความสัมพันธ์ ของ Class ทั้ง 2 เพื่อทำการบันทึก	สามารถแสดงข้อมูลการผูกความสัมพันธ์ได้	ถูกต้อง

ตารางที่ 20 (ต่อ) การจัดการข้อมูล Design Pattern และลำดับขั้นตอนการทดสอบระบบ

ลำดับ	การทดสอบ	คำอธิบาย	ผลการทดสอบที่คาดหวัง	ผลการทดสอบจริง
19.	แก้ไขความสัมพันธ์ระหว่าง Class	ทำการเลือกความสัมพันธ์ ของClass ทั้ง 2 เพื่อทำการแก้ไข	สามารถแสดงข้อมูลการผูกความสัมพันธ์ที่แก้ไขเรียบร้อยได้	ถูกต้อง
20.	ลบข้อมูลความสัมพันธ์ระหว่าง Class	ทำการลบข้อมูลความสัมพันธ์ระหว่าง Class	สามารถลบรายการความสัมพันธ์ระหว่าง Class ได้	ถูกต้อง

ตารางที่ 21 การจัดการข้อมูลพนักงาน และลำดับขั้นตอนการทดสอบระบบ

ลำดับ	การทดสอบ	คำอธิบาย	ผลการทดสอบที่คาดหวัง	ผลการทดสอบจริง
1.	สร้างข้อมูลพนักงาน ของระบบ	ทำการสร้างข้อมูลพนักงานระบบ	สามารถแสดงข้อมูลรายละเอียดของพนักงานที่สร้างได้	ถูกต้อง
2.	แก้ไขข้อมูลพนักงาน ของระบบ	ทำการแก้ไขข้อมูลพนักงานระบบ	สามารถแสดงข้อมูลรายละเอียดของพนักงานที่แก้ไขได้	ถูกต้อง
3.	ลบข้อมูลพนักงานของระบบ	ทำการลบข้อมูลพนักงานระบบ	สามารถลบรายการข้อมูลที่ทำการเลือกได้	ถูกต้อง

ตารางที่ 22 การจัดการข้อมูลโครงการ และลำดับขั้นตอนการทดสอบระบบ

ลำดับ	การทดสอบ	คำอธิบาย	ผลการทดสอบที่คาดหวัง	ผลการทดสอบจริง
1.	สร้างข้อมูลโครงการ ของระบบ	ทำการสร้างข้อมูลโครงการระบบ	สามารถแสดงข้อมูลชื่อโครงสร้างที่สร้างได้	ถูกต้อง
2.	แก้ไขข้อมูลโครงการ ของระบบ	ทำการแก้ไขข้อมูลโครงการที่เลือก	สามารถแสดงรายการแก้ไขข้อมูล โครงการได้	ถูกต้อง
3.	ลบข้อมูลโครงการของระบบ	ทำการลบข้อมูลโครงการ	สามารถลบรายการข้อมูล โครงการที่ทำการเลือกได้	ถูกต้อง
4.	ระบุพนักงานที่พัฒนาระบบในโครงการ	ระบุพนักงานประจำโครงการ หรือมอบหมายโครงการให้แก่พนักงาน	สามารถเลือกข้อมูลพนักงาน ประจำโครงการได้	ถูกต้อง
5.	แก้ไขพนักงานที่พัฒนาระบบในโครงการ	ปรับเปลี่ยนพนักงานประจำโครงการ	สามารถแก้ไขข้อมูลพนักงาน ประจำโครงการได้	ถูกต้อง
6.	ลบพนักงานที่พัฒนาระบบในโครงการ	ลบข้อมูลพนักงานประจำโครงการ	สามารถลบข้อมูลพนักงาน ประจำโครงการได้	ถูกต้อง
7.	ตรวจสอบจำนวนและรายละเอียดโครงการ ของพนักงานในระบบ	ตรวจสอบโครงการที่ได้มอบหมาย ให้กับพนักงาน	สามารถเลือกพนักงานและดูข้อมูลชื่อโครงการที่ได้รับมอบหมายได้	ถูกต้อง

ตารางที่ 23 การจัดการข้อมูลการร้องขอของพนักงาน และลำดับขั้นตอนการทดสอบระบบ

ลำดับ	การทดสอบ	คำอธิบาย	ผลการทดสอบที่คาดหวัง	ผลการทดสอบจริง
1.	ตรวจสอบการร้องขอของนักพัฒนาระบบ	ทำการเลือกเมนู Monitor Activity Change	สามารถแสดงรายการและสถานะการร้องขอจากนักพัฒนาระบบได้	ถูกต้อง
2.	ยอมรับการร้องขอ	ทำการกดปุ่ม Accept เพื่อยอมรับการร้องขอ	สามารถแสดงข้อมูลสถานะการร้องขอเป็น Accept และ สถานะโปรแกรมจะเป็น Change	ถูกต้อง
3.	ยกเลิกการร้องขอ	ทำการกดปุ่ม Del เพื่อยกเลิกการร้องขอ	สามารถแสดงข้อมูลสถานะการร้องขอเป็น Reject	ถูกต้อง

ตารางที่ 24 การจัดการข้อมูลรายการตรวจทาน ชุดคำสั่งโปรแกรม และลำดับขั้นตอนการทดสอบระบบ

ลำดับ	การทดสอบ	คำอธิบาย	ผลการทดสอบที่คาดหวัง	ผลการทดสอบจริง
1.	สร้างรายการตรวจทานตั้งต้น	กดปุ่ม Add และทำการเลือกโปรเจค กับข้อมูล XML ที่ต้องการสร้างรายการตรวจทาน	สามารถแสดงข้อมูลชื่อโครงการ จำนวน Class และจำนวนความสัมพันธ์	ถูกต้อง
2.	แก้ไขรายการตรวจทานตั้งต้น	เลือกรายการที่ต้องการแก้ไข และทำการกดปุ่ม Edit เพื่อบันทึกข้อมูลรายการตรวจทานใหม่	สามารถแสดงข้อมูลชื่อโครงการ จำนวน Class และจำนวนความสัมพันธ์	ถูกต้อง

ตารางที่ 24 (ต่อ) การจัดการข้อมูลรายการตรวจทาน ชุดคำสั่งโปรแกรม และลำดับขั้นตอนการทดสอบระบบ

ลำดับ	การทดสอบ	คำอธิบาย	ผลการทดสอบที่คาดหวัง	ผลการทดสอบจริง
3.	ลบข้อมูลรายการตรวจทานตั้งต้น	ทำการลบข้อมูลรายการตรวจทาน	สามารถลบรายการข้อมูลรายการตรวจทานที่ทำการเลือกได้	ถูกต้อง
4.	ตรวจสอบความถูกต้องของโครงสร้างรายการตรวจทาน	เลือกโครงการที่ต้องการดูรายละเอียด Class และกดปุ่ม Project Structure	สามารถแสดงข้อมูลชื่อ Class จำนวน Attribute และ จำนวน Methods ได้ครบทุก Class	ถูกต้อง
5.	ตรวจสอบความถูกต้อง Class ของรายการตรวจทาน	เลือก Class ที่ต้องการดูรายละเอียด และกดปุ่ม Class Detail	สามารถแสดงรายละเอียด Attribute และ รายละเอียด Methods ได้ครบถ้วน	ถูกต้อง
6.	ตรวจสอบความถูกต้อง Methods Parameter ของรายการตรวจทาน	เลือก Method ที่ต้องการดูรายละเอียดของ Parameter และกดปุ่ม Parameter	สามารถแสดงรายละเอียด Parameter ที่อยู่ใน Method ได้อย่างครบถ้วน	ถูกต้อง
7.	ตรวจสอบความถูกต้องของความสัมพันธ์ระหว่างClass ต้นทางไปยัง Class ปลายทาง จากรายการตรวจทาน	เลือกโครงการที่ต้องการดูรายละเอียดของความสัมพันธ์ระหว่าง Class และกดปุ่ม Detail Relation	สามารถแสดงข้อมูลชื่อ Class ต้นทาง Class ปลายทาง และความสัมพันธ์ระหว่าง Class ได้อย่างครบถ้วน	ถูกต้อง

ตารางที่ 24 (ต่อ) การจัดการข้อมูลรายการตรวจทาน ชุดคำสั่งโปรแกรม และลำดับขั้นตอนการทดสอบระบบ

ลำดับ	การทดสอบ	คำอธิบาย	ผลการทดสอบที่คาดหวัง	ผลการทดสอบจริง
8.	สร้างรายการตรวจสอบข้อมูลชุดคำสั่งโปรแกรม	กดปุ่ม Add และทำการเลือกโปรเจค กับ Folder ข้อมูลชุดคำสั่งโปรแกรมเพื่อสร้างข้อมูลสำหรับตรวจสอบ	สามารถแสดงข้อมูลชื่อโครงการ จำนวน Class และจำนวนความสัมพันธ์ได้อย่างครบถ้วน	ถูกต้อง
9.	แก้ไขรายการตรวจสอบข้อมูลชุดคำสั่งโปรแกรม	เลือกรายการที่ต้องการแก้ไข และทำการกดปุ่ม Edit เพื่อบันทึกข้อมูลรายการตรวจสอบใหม่	สามารถแสดงข้อมูลชื่อโครงการ จำนวน Class และจำนวนความสัมพันธ์ได้อย่างครบถ้วน	ถูกต้อง
10.	ลบข้อมูลรายการตรวจสอบข้อมูลชุดคำสั่งโปรแกรม	ทำการลบข้อมูลรายการตรวจสอบ	สามารถลบรายการข้อมูลรายการตรวจสอบที่ทำการเลือกได้	ถูกต้อง
11.	ตรวจสอบความถูกต้องของโครงสร้างรายการตรวจสอบข้อมูลชุดคำสั่งโปรแกรม	เลือกโครงการที่ต้องการดูรายละเอียด Class และกดปุ่ม Project Structure	สามารถแสดงข้อมูลชื่อ Class จำนวน Attribute และ จำนวน Methods ได้ครบทุก Class	ถูกต้อง
12.	ตรวจสอบความถูกต้อง Class ของรายการตรวจสอบข้อมูลชุดคำสั่งโปรแกรม	เลือก Class ที่ต้องการดูรายละเอียด และกดปุ่ม Class Detail	สามารถแสดงรายละเอียด Attribute และ รายละเอียด Methods ได้ครบถ้วน	ถูกต้อง

ตารางที่ 24 (ต่อ) การจัดการข้อมูลรายการตรวจทาน ชุดคำสั่งโปรแกรม และลำดับขั้นตอนการทดสอบระบบ

ลำดับ	การทดสอบ	คำอธิบาย	ผลการทดสอบที่คาดหวัง	ผลการทดสอบจริง
13.	ตรวจสอบความถูกต้อง Methods Parameter ของรายการตรวจสอบข้อมูลชุดคำสั่งโปรแกรม	เลือก Method ที่ต้องการดูรายละเอียดของ Parameter และกดปุ่ม Parameter	สามารถแสดงรายละเอียด Parameter ที่อยู่ใน Method ได้อย่างครบถ้วน	ถูกต้อง
14.	ตรวจสอบความถูกต้อง ของความสัมพันธ์ระหว่าง Class ต้นทางไปยัง Class ปลายทาง จากรายการตรวจสอบข้อมูลชุดคำสั่งโปรแกรม	เลือกโครงการที่ต้องการดูรายละเอียด ของความสัมพันธ์ระหว่าง Class และกดปุ่ม Detail Relation	สามารถแสดงข้อมูลชื่อ Class ต้นทาง Class ปลายทาง และความสัมพันธ์ระหว่าง Class ได้อย่างครบถ้วน	ถูกต้อง

ตารางที่ 25 การตรวจสอบรายการตรวจทานกับรายการชุดคำสั่งโปรแกรมภาษาจาวา และลำดับขั้นตอนการทดสอบระบบ

ลำดับ	การทดสอบ	คำอธิบาย	ผลการทดสอบที่คาดหวัง	ผลการทดสอบจริง
1.	ตรวจสอบรายการที่ต้องการตรวจสอบ	ทำการเลือกรายการที่ต้องการตรวจสอบและกดปุ่ม Verify	สามารถแสดงสถานะการ ตรวจสอบที่สำเร็จได้ โดยสถานะจะแสดงผล Verify Completed	ถูกต้อง
2.	ดูรายละเอียดที่ได้จากการตรวจสอบ	กดปุ่ม Verify Detail	สามารถแสดงข้อมูลที่ได้จากการตรวจสอบ	ถูกต้อง

5.1.2 ทดสอบการทำงานฟังก์ชันหลักของระบบ

การทดสอบระบบเพื่อวัดผลความถูกต้องในการตรวจสอบชุดคำสั่งโปรแกรมภาษาจาวากับชุดข้อมูลรายการตรวจทาน แสดงดังตารางที่ 26

ตารางที่ 26 การทดสอบเครื่องมือเพื่อวัดผลความถูกต้อง

ลำดับ	การทดสอบ	คำอธิบาย	ผลการทดสอบที่คาดหวัง	ผลการทดสอบจริง
1.	ตรวจสอบ Class Name	ตรวจสอบ Class Name จากชุดตรวจทาน กับชุดคำสั่งโปรแกรมภาษาจาวา ผ่านระบบ	ชื่อข้อมูล Class ตรงกัน ถ้าไม่ตรงกันจะมี Comment ว่าข้อมูลส่วนชุดตรวจทาน หรือข้อมูลชุดคำสั่งโปรแกรมภาษาจาวาใดขาดหายไป	ถูกต้อง

ตารางที่ 26 (ต่อ) การทดสอบเครื่องมือเพื่อวัดผลความถูกต้อง

ลำดับ	การทดสอบ	คำอธิบาย	ผลการทดสอบที่คาดหวัง	ผลการทดสอบจริง
2.	ตรวจสอบ Attribute Name	ตรวจสอบ Attribute Name จากชุดตรวจทาน กับชุดคำสั่งโปรแกรมภาษาจาวาผ่านระบบ	ชื่อข้อมูล Attribute Name ตรงกัน ถ้าไม่ตรงกันจะมี Comment ว่าข้อมูลส่วนชุดตรวจทาน หรือข้อมูลชุดคำสั่งโปรแกรมภาษาจาวาใดขาดหายไป	ถูกต้อง
3.	ตรวจสอบ Attribute Visibility	ตรวจสอบ Attribute Visibility จากชุดตรวจทานกับชุดคำสั่งโปรแกรมภาษาจาวาผ่านระบบ	ชื่อข้อมูล Attribute Visibility ตรงกัน ถ้าไม่ตรงกันจะมี Comment ว่าข้อมูลส่วนชุดตรวจทาน หรือข้อมูลชุดคำสั่งโปรแกรมภาษาจาวาใดขาดหายไป	ถูกต้อง
4.	ตรวจสอบ Attribute Type	ตรวจสอบ Attribute Type จากชุดตรวจทาน กับชุดคำสั่งโปรแกรมภาษาจาวาผ่านระบบ	ชื่อข้อมูล Attribute Type ตรงกัน ถ้าไม่ตรงกันจะมี Comment ว่าข้อมูลส่วนชุดตรวจทาน หรือข้อมูลชุดคำสั่งโปรแกรมภาษาจาวาใดขาดหายไป	ถูกต้อง

ตารางที่ 26 (ต่อ) การทดสอบเครื่องมือเพื่อวัดผลความถูกต้อง

ลำดับ	การทดสอบ	คำอธิบาย	ผลการทดสอบที่คาดหวัง	ผลการทดสอบจริง
5.	ตรวจสอบ Method Name	ตรวจสอบ Method Name จากชุดตรวจทาน กับชุดคำสั่งโปรแกรมภาษาจาวาผ่านระบบ	ชื่อข้อมูล Method Name ตรงกัน ถ้าไม่ตรงกันจะมี Comment ว่าข้อมูลส่วนชุดตรวจทาน หรือข้อมูลชุดคำสั่งโปรแกรมภาษาจาวาใดขาดหายไป	ถูกต้อง
6.	ตรวจสอบ Method Visibility	ตรวจสอบ Method Visibility จากชุดตรวจทานกับชุดคำสั่งโปรแกรมภาษาจาวาผ่านระบบ	ชื่อข้อมูล Method Visibility ตรงกัน ถ้าไม่ตรงกันจะมี Comment ว่าข้อมูลส่วนชุดตรวจทาน หรือข้อมูลชุดคำสั่งโปรแกรมภาษาจาวาใดขาดหายไป	ถูกต้อง
7.	ตรวจสอบ Method Type	ตรวจสอบ Method Type จากชุดตรวจทาน กับชุดคำสั่งโปรแกรมภาษาจาวาผ่านระบบ	ชื่อข้อมูล Method Type ตรงกัน ถ้าไม่ตรงกันจะมี Comment ว่าข้อมูลส่วนชุดตรวจทาน หรือข้อมูลชุดคำสั่งโปรแกรมภาษาจาวาใดขาดหายไป	ถูกต้อง

ตารางที่ 26 (ต่อ) การทดสอบเครื่องมือเพื่อวัดผลความถูกต้อง

ลำดับ	การทดสอบ	คำอธิบาย	ผลการทดสอบที่คาดหวัง	ผลการทดสอบจริง
8.	ตรวจสอบ Method Parameter Name	ตรวจสอบ Method Parameter Name จากชุดตรวจทาน กับชุดคำสั่งโปรแกรมภาษาจาวาผ่านระบบ	ชื่อข้อมูล Method Parameter Name ตรงกัน ถ้าไม่ตรงกันจะมี Comment ว่าข้อมูลส่วนชุดตรวจทาน หรือข้อมูลชุดคำสั่งโปรแกรมภาษาจาวาใดขาดหายไป	ถูกต้อง
9.	ตรวจสอบ Method Parameter Type	ตรวจสอบ Method Parameter Type จากชุดตรวจทาน กับชุดคำสั่งโปรแกรมภาษาจาวากับผ่านระบบ	ชื่อข้อมูล Method Parameter Type ตรงกัน ถ้าไม่ตรงกันจะมี Comment ว่าข้อมูลส่วนชุดตรวจทาน หรือข้อมูลชุดคำสั่งโปรแกรมภาษาจาวาใดขาดหายไป	ถูกต้อง

ตารางที่ 26 (ต่อ) การทดสอบเครื่องมือเพื่อวัดผลความถูกต้อง

ลำดับ	การทดสอบ	คำอธิบาย	ผลการทดสอบที่คาดหวัง	ผลการทดสอบจริง
10.	ตรวจสอบความสัมพันธ์ระหว่าง Class	ตรวจสอบความสัมพันธ์ระหว่าง Class จากชุดตรวจทาน กับชุดคำสั่งโปรแกรมภาษาจาวากับผ่านระบบ	ข้อมูลความสัมพันธ์ตรงกัน ถ้าไม่ตรงกันจะมี Comment ว่าข้อมูลส่วนชุดตรวจทาน หรือข้อมูลชุดคำสั่งโปรแกรมภาษาจาวาใดขาดหายไป	ถูกต้อง
11.	ค้นหา Design Pattern	ค้นหา Design Pattern จากชุดข้อมูลตรวจทาน	ตรวจสอบผลลัพธ์ที่ได้ว่าตรงตามโครงสร้างของ Design Pattern จริงหรือไม่ จากข้อมูล Design Pattern ตั้งต้นในระบบ Automatic Code Review ถ้าตรวจสอบแล้วตรงตามที่มีการแสดงผลแสดงว่าการค้นหาผลลัพธ์ถูกต้อง	ถูกต้อง

รายงานการแสดงผลการตรวจทานดังรูปที่ 54และ รูปที่ 55

Report : Verify Structure Detail Data : 05/05/2553

Project Name : Project Factory Page 1 of 4

Pattern Name : Factory Method

Pass : 46 Fail : 0 Total Result : 46

No.	Checklist Detail	Class Diagram	Source Code	Result	Comment
1	Class Name	Calculator	Calculator	/	
	Method Name	calculate	calculate	/	
	Method Visibility	public	public	/	
	Method Type	float	float	/	
	Method Parameter	a	a	/	
	Method Parameter	int	int	/	
	Method Parameter	b	b	/	
	Method Parameter	int	int	/	

Page 1 of 4

รูปที่ 54 ตัวอย่างรายงานการแสดงผลการตรวจทานโครงสร้างของ Class

Report : Verify Structure Detail Data : 05/05/2553

Project Name : Project Factory Page 1 of 1

Pattern Name : Factory Method

Pass : 5 Fail : 0 Total Result : 5

No.	CheckList Class Begin	CheckList Class End	CheckList Relation	SourceCode Class Begin	SourceCode Class End	SourceCode Relation	Result
1	NormalCalculator	Calculator	Generalization	NormalCalculator	Calculator	Generalization	/
2	AngleCalculator	Calculator	Generalization	AngleCalculator	Calculator	Generalization	/
3	CalculatorFactory	NormalCalculator	Asociation	CalculatorFactory	NormalCalculator	Asociation	/
4	CalculatorFactory	AngleCalculator	Asociation	CalculatorFactory	AngleCalculator	Asociation	/
5	CalculatorClient	CalculatorFactory	Asociation	CalculatorClient	CalculatorFactory	Asociation	/

Page 1 of 1

รูปที่ 55 ตัวอย่างรายงานการแสดงผลการตรวจทานความสัมพันธ์ระหว่าง Calss

5.2 สรุปผลการทดลอง

จากผลการทดลองข้างต้นสรุปได้ว่าระบบตัวตรวจทานชุดคำสั่งอัตโนมัติ ตามที่ได้กำหนดไว้นี้สามารถบันทึก เปลี่ยนแปลง และจัดเก็บข้อมูลต่างๆ รวมถึงการวัดผลความถูกต้องในการตรวจสอบชุดคำสั่งโปรแกรมภาษาจาวาได้อย่างแม่นยำ นอกจากนี้ผู้วิจัยยังได้ทำการทดสอบเครื่องมือนี้โดยการนำไปทดสอบกับโปรเจค Pilot ของบริษัท Optimussoft เพื่อช่วยในการตรวจทาน Defect ตั้งแต่ขั้นตอนการออกแบบ ทำให้ช่วยลดปัญญาการออกแบบที่ไม่ตรงกับชุดโปรแกรมที่นักพัฒนาโปรแกรมได้พัฒนาขึ้นมา

บทที่ 6
สรุปผลการวิจัย

6.1 สรุปผลการวิจัย

การลดจำนวนเหตุที่ทำให้เกิด Defect ได้นั้นเป็นส่วนสำคัญของความสำเร็จในการบรรลุเป้าหมายของโปรเจคอย่างมาก ดังนั้นขั้นตอนการพัฒนาระบบในแต่ละ Phase ที่เกิดขึ้น จึงจำเป็นอย่างยิ่งต่อการนำหลักการ V&V เข้ามา Verify กระบวนการทำงาน

ในงานวิจัยนี้ได้มีการนำเสนอ แนวความคิดและเครื่องมือ ที่ช่วยในการตรวจทาน เพื่อลด Defect ผ่านระบบตัวตรวจทานชุดคำสั่งอัตโนมัติ ทำให้การพัฒนาระบบมีความผิดพลาดน้อยลง และยังสามารถช่วยให้การพัฒนาซอฟต์แวร์ยังมีประสิทธิภาพสูง เนื่องจากมีการนำหลักการ OOP เข้ามาช่วยในการออกแบบผ่าน UML Class diagram หรือจะเป็นการเลือกใช้ Class diagram ผ่านทาง Design Patterns นอกจากนี้รายงานจากระบบยังเป็นการสร้าง Review Checklist ขึ้นมาเพื่อช่วยในการตรวจสอบความถูกต้องของ Requirement ได้อีกด้วย

จากการทดสอบผลระบบตัวตรวจทานชุดคำสั่งอัตโนมัติ กับโปรเจคระดับเล็ก ระดับกลาง หรือระดับใหญ่ได้ผลการตรวจจับ Defect ที่ถูกต้องแม่นยำ ทำให้นักพัฒนาโปรแกรมแก้ไข Source Code ได้ทันถ่วงที ไม่เสียเวลากับการแก้ไข Source Code ที่ไม่ตรงกัน Design Document

6.2 ข้อจำกัด

จากการดำเนินงานวิจัยนี้พบปัญหาและข้อจำกัด ดังต่อไปนี้

1. ระบบสามารถค้นหาข้อมูล Design Patterns จากชุดข้อมูลตรวจทานได้ เท่าที่มีข้อมูล Design Patterns อยู่ในระบบเท่านั้น
2. ระบบรองรับข้อมูลไฟล์เอ็กซ์เอ็มแอลจากเครื่องมือ Rational Rose เท่านั้น

รายการอ้างอิง

[1] Booch, G., Maksimchuk, R.A., Engel, M.W., Young, R.J., Conallen, J., Houston, K.A., and other. Object-Oriented Analysis and Design with Applications third edition. Addison-Wesley Professional, 2004.

[2] The Institute of Electrical and Electronics Engineers, Inc., IEEE 1012-2004- IEEE Standard for Software Verification and Validation, 2004.

[3] IEEE Standard for Software Reviews-IEEE Computer Society IEEE Std. IEEE Standard for Software Reviews, 2008.

[4] Wiegers K.E. Peer Reviews in Software. Addison-Wesley, 2002.

[5] Object Management Group. UML Semantics. [Online]. Available from http://www.omg.org/docs/ad/97-08-04.pdf [1997, August 4].

[6] Gamma, E., Helm, T., Johnson, R. and Vlissides, J. Design patterns: elements of reusable Object-Oriented Software, Addison-Wesley Longman Publishing Co., Inc., Boston, 1995.

[7] Object management Group. XML Metadata Interchange [Online]. Available from http://www.omg.org/docs/ad/98-07-01.pdf [1998, July 1].

[8] Muhanmad U.,and Aamer N. Automatic Generation of Java Code from UML Diagrams using UJECTOR, 2009.

[9] Stacy N., Johann S. What Makes a Code Review Trustworthy?, USA: IEEE Computer Society Washington, DC, 2004.

ภาคผนวก

ภาคผนวก ก.
คำอธิบายยูสเคส

ตารางที่ 27 อธิบายยูสเคสการจัดการคำร้องของนักวิเคราะห์ระบบ

หมายเลขยูสเคส : UC1	**ชื่อยูสเคส** : การจัดการคำร้องของนักวิเคราะห์ระบบ
ผู้เกี่ยวข้องหลัก : นักวิเคราะห์ระบบ	
รายละเอียด : เพื่ออธิบายขั้นตอนการตรวจสอบคำร้องจากนักพัฒนาระบบ	
ความสัมพันธ์ : Association : นักวิเคราะห์ระบบ Use : Extend : Generalization :	
ขั้นตอนการทำงานหลัก : 1. เลือกเมนู Activity Change Request 2. ระบบแสดงรายละเอียดการร้องขอ และสถานะการร้องของทุกโครงการที่เกิดขึ้น 3. ยอมรับ หรือยกเลิกการร้องขอ	
ขั้นตอนการทำงานกรณีพิเศษ :	

ตารางที่ 28 อธิบายยูสเคสการจัดการคำร้องของนักพัฒนาระบบ

หมายเลขยูสเคส : UC2	**ชื่อยูสเคส** : การจัดการคำร้องของนักพัฒนาระบบ
ผู้เกี่ยวข้องหลัก : นักพัฒนาระบบ	
รายละเอียด : เพื่ออธิบายขั้นตอนตรวจสอบคำร้องของนักพัฒนาระบบ	
ความสัมพันธ์ : Association : นักพัฒนาระบบ Use : Extend : Generalization :	
ขั้นตอนการทำงานหลัก : 1. เลือกเมนู Request Verify 2. ระบบแสดงรายละเอียดการร้องขอ และสถานะการร้องขอ ในแต่ละโครงที่ทำการร้องขออยู่ a. เพิ่มการร้องขอโปรเจคใหม่ b. ส่งการร้องขอซ้ำ c. ลบรายการที่มีการร้องขอสำเร็จ	
ขั้นตอนการทำงานกรณีพิเศษ :	

ตารางที่ 29 อธิบายยูสเคสการจัดการรายการตรวจทาน

หมายเลขยูสเคส : UC3	**ชื่อยูสเคส** : การจัดการรายการตรวจทาน
ผู้เกี่ยวข้องหลัก : นักวิเคราะห์ระบบ	
รายละเอียด : เพื่อทำการบันทึกข้อมูลรายการตรวจทานของระบบ	
ความสัมพันธ์ : Association : นักวิเคราะห์ระบบ Use : Extend : Generalization :	
ขั้นตอนการทำงานหลัก : 1. เลือกเมนู CheckList Management 2. ระบบแสดงรายระเอียดโครงการที่มีรายการตรวจทานทั้งหมด a. เพิ่มข้อมูลรายการตรวจทาน b. แก้ไขข้อมูลรายการตรวจทาน c. ลบข้อมูลรายการตรวจทาน	
ขั้นตอนการทำงานกรณีพิเศษ :	

ตารางที่ 30 อธิบายยูสเคสการจัดการชุดคำสั่งโปรแกรมภาษาจาวา

หมายเลขยูสเคส : UC4	**ชื่อยูสเคส** : การจัดการชุดคำสั่งโปรแกรมภาษาจาวา
ผู้เกี่ยวข้องหลัก : นักวิเคราะห์ระบบ	
รายละเอียด : เพื่อทำการบันทึกข้อมูลรายการตรวจสอบชุดคำสั่งโปรแกรมของระบบ	
ความสัมพันธ์ : Association : นักวิเคราะห์ระบบ Use : Extend : Generalization :	
ขั้นตอนการทำงานหลัก : 1. เลือกเมนู Source Code Management 2. ระบบแสดงรายระเอียดโครงการที่มีรายการตรวจสอบชุดคำสั่งโปรแกรมทั้งหมด 2.1 เพิ่มข้อมูลรายการตรวจสอบชุดคำสั่งโปรแกรม 2.2 แก้ไขข้อมูลรายการตรวจสอบชุดคำสั่งโปรแกรม 2.3 ลบข้อมูลรายการตรวจสอบชุดคำสั่งโปรแกรม	
ขั้นตอนการทำงานกรณีพิเศษ :	

ตารางที่ 31 อธิบายยูสเคสการจัดการ การตรวจสอบรายการตรวจทานกับรายการตรวจสอบชุดคำสั่งโปรแกรม

หมายเลขยูสเคส : UC5	**ชื่อยูสเคส** : การตรวจสอบรายการตรวจทานกับรายการตรวจสอบชุดคำสั่งโปรแกรม
ผู้เกี่ยวข้องหลัก : นักวิเคราะห์ระบบ	
รายละเอียด : เพื่อทำการประมวลผลการตรวจสอบรายการที่ผิดปกติ	
ความสัมพันธ์ : Association : นักวิเคราะห์ระบบ Use : Extend : Generalization :	
ขั้นตอนการทำงานหลัก : 1.เลือกเมนู Verify Management 2.ระบบแสดงรายระเอียดโครงการทั้งหมด และสถานะของการตรวจสอบ 2.1 ถ้าระบบมีข้อมูลรายการตรวจทาน และข้อมูลรายการตรวจสอบชุดคำสั่งโปรแกรมเรียบร้อยแล้ว จะสามารถทำการตรวจสอบข้อผิดพลาดได้ 2.2 ระบบจะมีการแจ้งเตือนไปยังนักพัฒนาระบบโดยอัตโนมัติ	
ขั้นตอนการทำงานกรณีพิเศษ :	

ตารางที่ 32 อธิบายยูสเคสการตรวจสอบผลลัพธ์ที่ได้จากการตรวจสอบของนักวิเคราะห์ระบบ

หมายเลขยูสเคส : UC6	**ชื่อยูสเคส** : การตรวจสอบผลลัพธ์ที่ได้จากการตรวจสอบของนักวิเคราะห์ระบบ
ผู้เกี่ยวข้องหลัก : นักวิเคราะห์ระบบ	
รายละเอียด : เพื่อทำการวิเคราะห์ข้อมูลที่ผิดปกติ และหาวิธีการแก้ไขตั้งแต่เนิ่น ๆ	
ความสัมพันธ์ : Association : นักวิเคราะห์ระบบ Use : Extend : Generalization :	
ขั้นตอนการทำงานหลัก : 1.เลือกเมนู Verify Management 2.ระบบแสดงรายระเอียดโครงการทั้งหมด และสถานะของการตรวจสอบ 2.1 เลือกโครงการที่ต้องการตรวจสอบ 2.2 แสดงผลที่ได้จากการตรวจสอบ 2.3 ออกรายงาน	
ขั้นตอนการทำงานกรณีพิเศษ :	

.

ตารางที่ 33 อธิบายยูสเคสการตรวจสอบผลลัพธ์ที่ได้จากการตรวจสอบของนักพัฒนาระบบ

หมายเลขยูสเคส : UC7	**ชื่อยูสเคส** : การตรวจสอบผลลัพธ์ที่ได้จากการตรวจสอบของนักพัฒนาระบบ
ผู้เกี่ยวข้องหลัก : นักของนักพัฒนาระบบ	
รายละเอียด : เพื่อทำการวิเคราะห์ข้อมูลที่ผิดปกติ และหาวิธีการแก้ไขตั้งแต่เนิ่น ๆ	
ความสัมพันธ์ : Association : นักพัฒนาระบบ Use : Extend : Generalization :	
ขั้นตอนการทำงานหลัก : 1.เลือกเมนู My Result Verify 2.ระบบแสดงรายระเอียดโครงการทั้งหมด และสถานะของการตรวจสอบของตนเอง 2.1 เลือกโครงการที่ต้องการตรวจสอบ 2.2 แสดงผลที่ได้จากการตรวจสอบ 2.3 ออกรายงาน	
ขั้นตอนการทำงานกรณีพิเศษ :	

ภาคผนวก ข
พจนานุกรมข้อมูลของระบบฐานข้อมูล

ตารางที่ 34 ตัวอย่าง ATTRIBUTETBL

ชื่อตาราง	ATTRIBUTETBL		
คำอธิบายตาราง	ข้อมูลประเภทลักษณะของ Class		
คีย์หลัก	ATTRIBUTE_ID		
ความสัมพันธ์	VISIBILITY_ID , CLASS_ID		
ชื่อสดมภ์	ประเภทข้อมูล	ค่าเป็น null	คำอธิบายสดมภ์
ATTRIBUTE_ID	NUMBER	ไม่ได้	รหัสข้อมูลลักษณะของ Class
ATTRIBUTE_NAME	VARCHAR2(200)	ไม่ได้	ชื่อข้อมูลลักษณะของ Class
ATTRIBUTE_TYPE	VARCHAR2(200)	ไม่ได้	ประเภทตัวแปลข้อมูลลักษณะของ Class
VISIBILITY_ID	NUMBER	ไม่ได้	รหัสการเข้าถึงของข้อมูล
CLASS_ID	NUMBER	ไม่ได้	รหัส Class
CREATETIME	DATE	ไม่ได้	วันที่สร้างข้อมูล
UPDATETIME	DATE	ไม่ได้	วันที่แก้ไขข้อมูลล่าสุด

ตารางที่ 35 ตัวอย่าง CHANGEREQUESTTBL

ชื่อตาราง	CHANGEREQUESTTBL		
คำอธิบายตาราง	ข้อมูลการร้องขอต่างๆ จากพนักงานในระบบ		
คีย์หลัก	PROJECT_ID		
ความสัมพันธ์	USERNAME		
ชื่อสดมภ์	ประเภทข้อมูล	ค่าเป็น null	คำอธิบายสดมภ์
PROJECT_ID	NUMBER	ไม่ได้	รหัสโปรเจค
ISCHANGECHECKLIST	CHAR(1)	ไม่ได้	สถานะการแก้ไขชุดข้อมูลตรวจทาน
ISCHANGESOURCE	CHAR(1)	ไม่ได้	สถานะการแก้ไขโปรแกรม
USERNAME	VARCHAR2(10)	ไม่ได้	รหัสพนักงาน
ISMODIFY	CHAR(1)	ไม่ได้	สถานะการตรวจสอบจากระบบ Auto Code Review
CHANGE_COMMENT	VARCHAR2(200)	ได้	หมายเหตุการเปลี่ยนแปลงข้อมูลโปรแกรม หรือข้อมูลชุดตรวจทาน
CREATETIME	DATE	ไม่ได้	วันที่สร้างข้อมูล
UPDATETIME	DATE	ไม่ได้	วันที่แก้ไขข้อมูลล่าสุด

ตารางที่ 36 ตัวอย่าง CHECKLISTPROJECTTBL

ชื่อตาราง	CHECKLISTPROJECTTBL		
คำอธิบายตาราง	ข้อมูลโปรเจคที่ถูกทำการสร้างชุดตรวจทาน		
คีย์หลัก	CHECKLIST_ID		
ความสัมพันธ์	PROJECT_ID		
ชื่อสดมภ์	ประเภทข้อมูล	ค่าเป็น null	คำอธิบายสดมภ์
CHECKLIST_ID	NUMBER	ไม่ได้	รหัสโปรเจคที่ถูกสร้างชุดข้อมูลตรวจทาน
CREATETIME	DATE	ไม่ได้	วันที่สร้างข้อมูล
UPDATETIME	DATE	ไม่ได้	วันที่แก้ไขข้อมูลล่าสุด

ตารางที่ 37 ตัวอย่าง CLASSRELATIONTBL

ชื่อตาราง	CLASSRELATIONTBL		
คำอธิบายตาราง	ข้อมูลความสัมพันธ์ระหว่าง Class เริ่มต้นกับ Class ปลายทาง		
คีย์หลัก	ATTRI CLASS_ID_BEGIN, CLASS_ID_END, RELATION_ID		
ความสัมพันธ์	CLASS_ID , PATTERN_ID , CHECKLIST_ID , SOURCE_ID , RELATION_ID		
ชื่อสดมภ์	ประเภทข้อมูล	ค่าเป็น null	คำอธิบายสดมภ์
CLASS_ID_BEGIN	NUMBER	ไม่ได้	รหัส Class เริ่มต้น
CLASS_ID_END	NUMBER	ไม่ได้	รหัส Class ปลายทาง
RELATION_ID	NUMBER	ไม่ได้	รหัสความสัมพันธ์
RELATION_COMMENT	VARCHAR2(200)	ได้	หมายเหตุความสัมพันธ์
PATTERN_ID	NUMBER	ได้	รหัส Design Pattern
CHECKLIST_ID	NUMBER	ได้	รหัสข้อมูลตรวจทาน
SOURCE_ID	NUMBER	ได้	รหัสข้อมูลโปรแกรม
CREATETIME	DATE	ไม่ได้	วันที่สร้างข้อมูล
UPDATETIME	DATE	ไม่ได้	วันที่แก้ไขข้อมูลล่าสุด

ตารางที่ 38 ตัวอย่าง CLASSTBL

ชื่อตาราง	CLASSTBL		
คำอธิบายตาราง	ข้อมูล Class หรือแม่พิมพ์		
คีย์หลัก	CLASS_ID		
ความสัมพันธ์	PATTERN_ID , CHECKLIST_ID , SOURCE_ID		
ชื่อสดมภ์	ประเภทข้อมูล	ค่าเป็น null	คำอธิบายสดมภ์
CLASS_ID	NUMBER	ไม่ได้	รหัส Class
CLASS_NAME	VARCHAR2(200)	ไม่ได้	ชื่อ Class
PATTERN_ID	NUMBER	ได้	รหัส Design Pattern
CHECKLIST_ID	NUMBER	ได้	รหัสข้อมูลตรวจทาน
SOURCE_ID	NUMBER	ได้	รหัสข้อมูลโปรแกรม
CREATETIME	DATE	ไม่ได้	วันที่สร้างข้อมูล
UPDATETIME	DATE	ไม่ได้	วันที่แก้ไขข้อมูลล่าสุด

ตารางที่ 39 ตัวอย่าง METHODSPARAMETERTBL

ชื่อตาราง	METHODSPARAMETERTBL		
คำอธิบายตาราง	ข้อมูลตัวแปลของ Methods หรือ Operation		
คีย์หลัก	PARAMETER_ID		
ความสัมพันธ์	METHODS_ID		
ชื่อสดมภ์	ประเภทข้อมูล	ค่าเป็น null	คำอธิบายสดมภ์
PARAMETER_ID	NUMBER	ไม่ได้	รหัสตัวแปล
PARAMETER_NAME	VARCHAR2(200)	ไม่ได้	ชื่อตัวแปล
PARAMETER_TYPE	VARCHAR2(200)	ไม่ได้	ประเภทตัวแปล
METHODS_ID	NUMBER	ไม่ได้	รหัส Method หรือ รหัสOperation
CREATETIME	DATE	ไม่ได้	วันที่สร้างข้อมูล
UPDATETIME	DATE	ไม่ได้	วันที่แก้ไขข้อมูล ล่าสุด

ตารางที่ 40 ตัวอย่าง METHODSTBL

ชื่อตาราง	METHODSTBL		
คำอธิบายตาราง	ข้อมูล Method หรือข้อมูล Operation		
คีย์หลัก	METHODS_ID		
ความสัมพันธ์	CLASS_ID , VISIBILITY_ID		
ชื่อสดมภ์	ประเภทข้อมูล	ค่าเป็น null	คำอธิบายสดมภ์
METHODS_ID	NUMBER	ไม่ได้	รหัส Methods หรือรหัสOperation
METHODS_NAME	VARCHAR2(200)	ไม่ได้	ชื่อ Method หรือชื่อ Operation
VISIBILITY_ID	NUMBER	ไม่ได้	รหัสการเข้าถึงของข้อมูล
METHODS_TYPE	VARCHAR2(200)	ไม่ได้	ประเภทของ Method
CLASS_ID	NUMBER	ไม่ได้	รหัส Class
CREATETIME	DATE	ไม่ได้	วันที่สร้างข้อมูล
UPDATETIME	DATE	ไม่ได้	วันที่แก้ไขข้อมูลล่าสุด

ตารางที่ 41 ตัวอย่างPATTERNTBL

ชื่อตาราง	PATTERNTBL		
คำอธิบายตาราง	ข้อมูล Design Pattern		
คีย์หลัก	PATTERN_ID		
ความสัมพันธ์	PATTERNTYPE_ID		
ชื่อสดมภ์	ประเภทข้อมูล	ค่าเป็น null	คำอธิบายสดมภ์
PATTERN_ID	NUMBER	ไม่ได้	รหัส Design Pattern
PATTERN_NAME	VARCHAR2(200)	ไม่ได้	ชื่อ Design Pattern
PATTERN_DESC	VARCHAR2(200)	ได้	คำอธิบาย Design Pattern
PATTERN_PICTURE	VARCHAR2(200)	ได้	รูปภาพประกอบ Design Pattern
PATTERNTYPE_ID	NUMBER	ไม่ได้	รหัสประเภทของ Design Pattern
CREATETIME	DATE	ไม่ได้	วันที่สร้างข้อมูล
UPDATETIME	DATE	ไม่ได้	วันที่แก้ไขข้อมูลล่าสุด

ตารางที่ 42 ตัวอย่าง PATTERNSCOPETBL

ชื่อตาราง	PATTERNSCOPETBL		
คำอธิบายตาราง	ข้อมูลขอบเขตของ Design Pattern		
คีย์หลัก	PATTERNTYPE_ID		
ความสัมพันธ์	-		
ชื่อสดมภ์	ประเภทข้อมูล	ค่าเป็น null	คำอธิบายสดมภ์
PATTERNSCOPE_ID	NUMBER	ไม่ได้	รหัสขอบเขตของ Design Pattern
PATTERNSCOPE_NAME	VARCHAR2(200)	ไม่ได้	ชื่อขอบเขตของ Design Pattern

CREATETIME	DATE	ไม่ได้	วันที่สร้างข้อมูล
UPDATETIME	DATE	ไม่ได้	วันที่แก้ไขข้อมูลล่าสุด

ตารางที่ 43 ตัวอย่าง PATTERNTYPETBL

ชื่อตาราง	PATTERNTYPETBL		
คำอธิบายตาราง	ข้อมูลวัตถุประสงค์ของ Design Patten		
คีย์หลัก	PATTERNTYPE_ID		
ความสัมพันธ์	-		
ชื่อสดมภ์	ประเภทข้อมูล	ค่าเป็น null	คำอธิบายสดมภ์
PATTERNTYPE_ID	NUMBER	ไม่ได้	รหัสวัตถุประสงค์ของ Design Pattern
PATTERNTYPE_NAME	VARCHAR2(200)	ไม่ได้	ชื่อวัตถุประสงค์ของ Design Pattern
CREATETIME	DATE	ไม่ได้	วันที่สร้างข้อมูล
UPDATETIME	DATE	ไม่ได้	วันที่แก้ไขข้อมูลล่าสุด

ตารางที่ 44 ตัวอย่าง PROJECTPATTERNTBL

ชื่อตาราง	PROJECTPATTERNTBL		
คำอธิบายตาราง	ข้อมูลความสัมพันระหว่างโปรเจคกับ Design Patten		
คีย์หลัก	PROJECT_ID , PATTERN_ID		
ความสัมพันธ์	PATTERN_ID		
ชื่อสดมภ์	ประเภทข้อมูล	ค่าเป็น null	คำอธิบายสดมภ์
PROJECT_ID	NUMBER	ไม่ได้	รหัสโปรเจค
PATTERN_ID	NUMBER	ไม่ได้	รหัส Design Pattern
CREATETIME	DATE	ไม่ได้	วันที่สร้างข้อมูล
UPDATETIME	DATE	ไม่ได้	วันที่แก้ไขข้อมูลล่าสุด

ตารางที่ 45 ตัวอย่าง PROJECTTBL

ชื่อตาราง	PROJECTTBL		
คำอธิบายตาราง	ข้อมูลโปรเจค		
คีย์หลัก	PROJECT_ID		
ความสัมพันธ์	-		
ชื่อสดมภ์	ประเภทข้อมูล	ค่าเป็น null	คำอธิบายสดมภ์
PROJECT_ID	NUMBER	ไม่ได้	รหัสโปรเจค
PROJECT_NAME	VARCHAR2(200)	ไม่ได้	ชื่อโปรเจค
PROJECT_DESC	VARCHAR2(200)	ได้	คำอธิบายโปรเจค
CREATETIME	DATE	ไม่ได้	วันที่สร้างข้อมูล
UPDATETIME	DATE	ไม่ได้	วันที่แก้ไขข้อมูลล่าสุด

ตารางที่ 46 ตัวอย่าง PROJECTUSERTBL

ชื่อตาราง	PROJECTUSERTBL		
คำอธิบายตาราง	ข้อมูลความสัมพันธ์ระหว่างโปรเจคกับพนักงาน		
คีย์หลัก	PROJECT_ID , USERNAME		
ความสัมพันธ์	USERNAME		
ชื่อสดมภ์	ประเภทข้อมูล	ค่าเป็น null	คำอธิบายสดมภ์
PROJECT_ID	NUMBER	ไม่ได้	รหัสโปรเจค
USERNAME	VARCHAR2(20)	ไม่ได้	รหัสพนักงาน
CREATETIME	DATE	ไม่ได้	วันที่สร้างข้อมูล
UPDATETIME	DATE	ไม่ได้	วันที่แก้ไขข้อมูลล่าสุด

ตารางที่ 47 ตัวอย่าง RELATIONTBL

ชื่อตาราง	RELATIONTBL		
คำอธิบายตาราง	ข้อมูลความสัมพันธ์ของ Class		
คีย์หลัก	RELATION_ID		
ความสัมพันธ์	-		
ชื่อสดมภ์	ประเภทข้อมูล	ค่าเป็น null	คำอธิบายสดมภ์
RELATION_ID	NUMBER	ไม่ได้	รหัสความสัมพันธ์
RELATION_NAME	VARCHAR2(200)	ไม่ได้	ชื่อความสัมพันธ์
CREATETIME	DATE	ไม่ได้	วันที่สร้างข้อมูล
UPDATETIME	DATE	ไม่ได้	วันที่แก้ไขข้อมูลล่าสุด

ตารางที่ 48 ตัวอย่าง ROLETBL

ชื่อตาราง	ROLETBL		
คำอธิบายตาราง	ข้อมูลสิทธิ์การใช้งานระบบ		
คีย์หลัก	ROLE		
ความสัมพันธ์	-		
ชื่อสดมภ์	ประเภทข้อมูล	ค่าเป็น null	คำอธิบายสดมภ์
ROLE	CHAR(1)	ไม่ได้	รหัสสิทธิ์การใช้งาน
ROLE_DESC	VARCHAR2(100)	ไม่ได้	คำอธิบายรหัสการใช้งาน
CREATETIME	DATE	ไม่ได้	วันที่สร้างข้อมูล
UPDATETIME	DATE	ไม่ได้	วันที่แก้ไขข้อมูลล่าสุด

ตารางที่ 49 ตัวอย่าง SOURCECODEPROJECTTBL

ชื่อตาราง	SOURCECODEPROJECTTBL		
คำอธิบายตาราง	ข้อมูลโปรเจคที่ถูกทำการจัดเก็บ		
คีย์หลัก	SOURCE_ID		
ความสัมพันธ์	-		
ชื่อสดมภ์	ประเภทข้อมูล	ค่าเป็น null	คำอธิบายสดมภ์
SOURCE_ID	NUMBER	ไม่ได้	รหัสโปรเจคที่ถูกสร้างจากการบันทึกข้อมูลโปรแกรมพนักงาน
CREATETIME	DATE	ไม่ได้	วันที่สร้างข้อมูล
UPDATETIME	DATE	ไม่ได้	วันที่แก้ไขข้อมูลล่าสุด

ตารางที่ 50 ตัวอย่าง USERTBL

ชื่อตาราง	USERTBL		
คำอธิบายตาราง	ข้อมูลพนักงาน		
คีย์หลัก	USERNAME		
ความสัมพันธ์	ROLE		
ชื่อสดมภ์	ประเภทข้อมูล	ค่าเป็น null	คำอธิบายสดมภ์
USERNAME	VARCHAR2(20)	ไม่ได้	รหัสพนักงาน
PASSWORD	VARCHAR2(20)	ไม่ได้	รหัสผ่านพนักงาน
NAME	VARCHAR2(150)	ไม่ได้	ชื่อพนักงาน
SURNAME	VARCHAR2(150)	ไม่ได้	นามสกุลพนักงาน
MOBILE	VARCHAR2(20)	ได้	เบอร์โทรศัพท์
EMAIL	VARCHAR2(150)	ได้	อีเมล
ROLE	CHAR(1)	ไม่ได้	รหัสสิทธิ์การทำงาน
CREATETIME	DATE	ไม่ได้	วันที่สร้างข้อมูล
UPDATETIME	DATE	ไม่ได้	วันที่แก้ไขข้อมูลล่าสุด

ตารางที่ 51 ตัวอย่าง VERIFYSTATUSTBL

ชื่อตาราง	VERIFYSTATUSTBL		
คำอธิบายตาราง	ข้อมูลจำกัดความผลลัพธ์การตรวจสอบ		
คีย์หลัก	VERIFYSTATUS_ID		
ความสัมพันธ์	-		
ชื่อสดมภ์	ประเภทข้อมูล	ค่าเป็น null	คำอธิบายสดมภ์
VERIFYSTATUS_ID	NUMBER	ไม่ได้	รหัสผลลัพธ์การตรวจสอบ
VEFIFYSTATUS_DESC	VARCHAR2(200)	ไม่ได้	คำอธิบายผลลัพธ์การตรวจสอบ
CREATETIME	DATE	ไม่ได้	วันที่สร้างข้อมูล
UPDATETIME	DATE	ไม่ได้	วันที่แก้ไขข้อมูลล่าสุด

ตารางที่ 52 ตัวอย่าง VERIFYTBL

ชื่อตาราง	VERIFYTBL		
คำอธิบายตาราง	ข้อมูลการตรวจสอบของข้อมูลตรวจทานกับโปรแกรม		
คีย์หลัก	VERIFY_ID		
ความสัมพันธ์	PROJECT_ID , VERIFYTYPE_ID , CHECKLISTITEM_ID , SOURCECODEITEM_ID , VERIFYSTATUS_ID		
ชื่อสดมภ์	ประเภทข้อมูล	ค่าเป็น null	คำอธิบายสดมภ์
VERIFY_ID	NUMBER	ไม่ได้	รหัสตรวจสอบ
PROJECT_ID	NUMBER	ไม่ได้	รหัสโปรเจค
VERIFYTYPE_ID	NUMBER	ไม่ได้	รหัสประเภทการตรวจสอบ
CHECKLISTITEM_ID	NUMBER		รหัสชุดข้อมูลตรวจทาน
SOURCECODEITEM_ID	NUMBER	ไม่ได้	รหัสข้อมูลโปรแกรม

VERIFYSTATUS_ID	NUMBER	ไม่ได้	รหัสผลการตรวจสอบ
VERIFY_COMMENT	VARCHAR2(200)	ได้	หมายเหตุการตรวจสอบ
CREATETIME	DATE	ไม่ได้	วันที่สร้างข้อมูล
UPDATETIME	DATE	ไม่ได้	วันที่แก้ไขข้อมูลล่าสุด

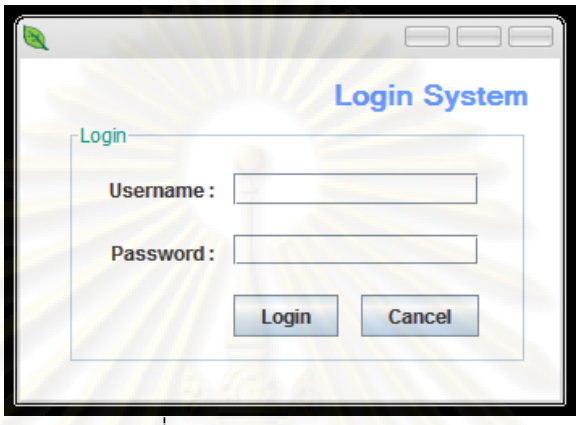
ตารางที่ 53 ตัวอย่าง VERIFYTYPETBL

ชื่อตาราง	VERIFYTYPETBL		
คำอธิบายตาราง	ข้อมูลจำกัดความการตรวจสอบ		
คีย์หลัก	VERIFYTYPE_ID		
ความสัมพันธ์	-		
ชื่อสดมภ์	ประเภทข้อมูล	ค่าเป็น null	คำอธิบายสดมภ์
VERIFYTYPE_ID	NUMBER	ไม่ได้	รหัสประเภทการตรวจสอบ
VERIFYTYPE_DESC	VARCHAR2(200)	ไม่ได้	คำอธิบายการตรวจสอบ
CREATETIME	DATE	ไม่ได้	วันที่สร้างข้อมูล
UPDATETIME	DATE	ไม่ได้	วันที่แก้ไขข้อมูลล่าสุด

ตารางที่ 54 ตัวอย่าง VISIBILITYTBL

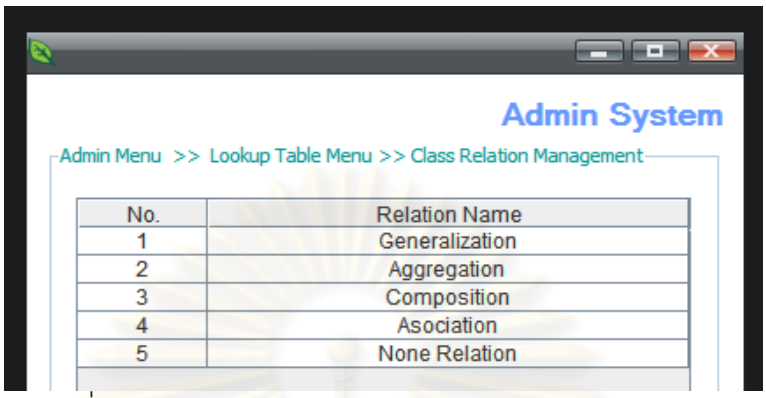
ชื่อตาราง	VISIBILITYTBL		
คำอธิบายตาราง	ข้อมูลจำกัดความการเข้าถึงข้อมูล		
คีย์หลัก	VISIBILITY_ID		
ความสัมพันธ์	-		
ชื่อสดมภ์	ประเภทข้อมูล	ค่าเป็น null	คำอธิบายสดมภ์
VISIBILITY_ID	NUMBER	ไม่ได้	รหัสการเข้าถึงของข้อมูล
VISIBILITY_NAME	VARCHAR2(200)	ไม่ได้	ชื่อจำกัดความของการเข้าถึงข้อมูล
CREATETIME	DATE	ไม่ได้	วันที่สร้างข้อมูล
UPDATETIME	DATE	ไม่ได้	วันที่แก้ไขข้อมูลล่าสุด

ภาคผนวก ค
ตัวอย่างหน้าจอของผู้ใช้งานระบบ

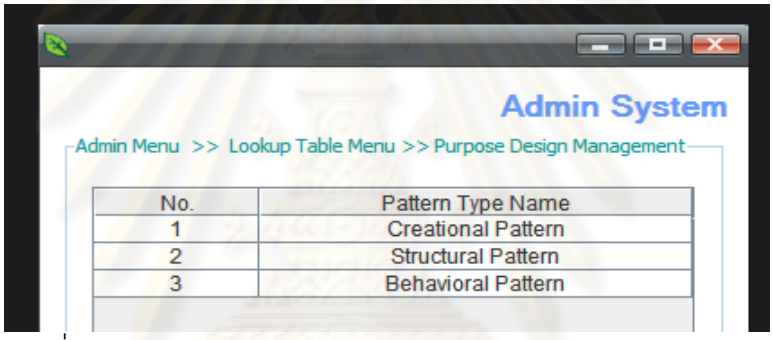


รูปที่ 56 ตัวอย่างหน้าจอ login

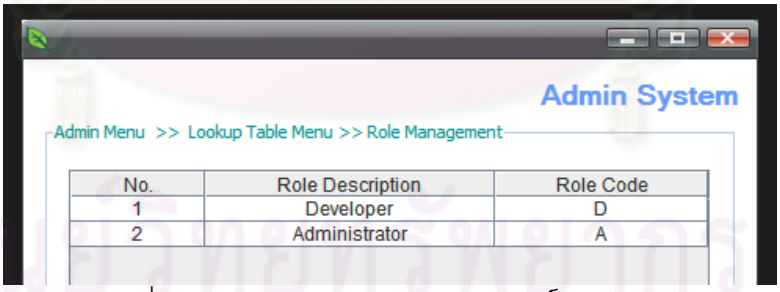
รูปที่ 57 ตัวอย่างหน้าจอเมนูผู้ดูแลระบบ



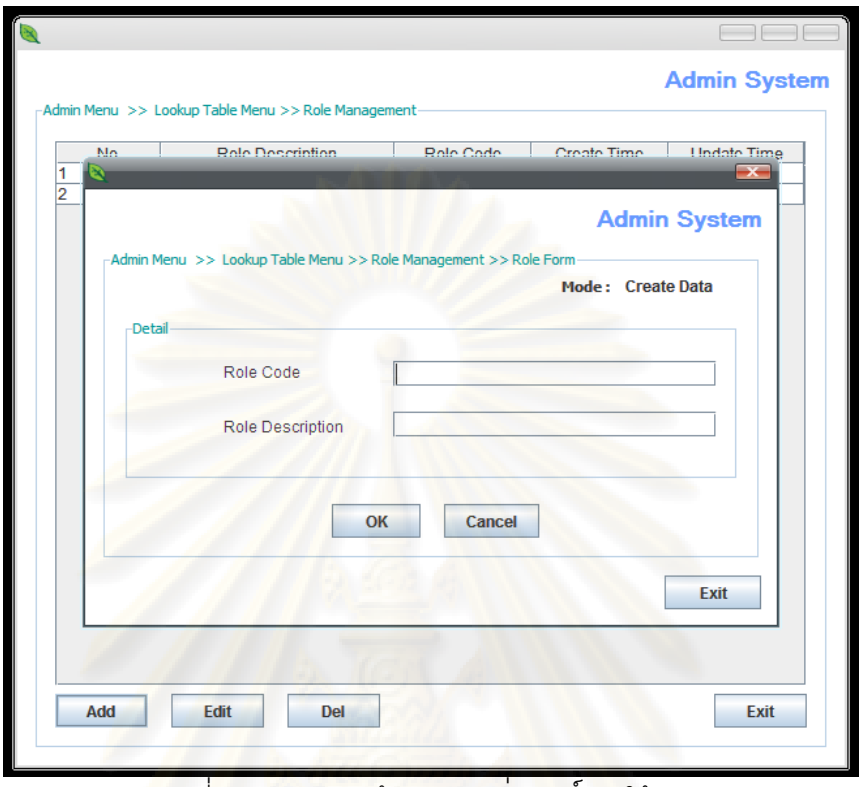
รูปที่ 58 ตัวอย่างหน้าจอการจัดการความสัมพันธ์ของ Class



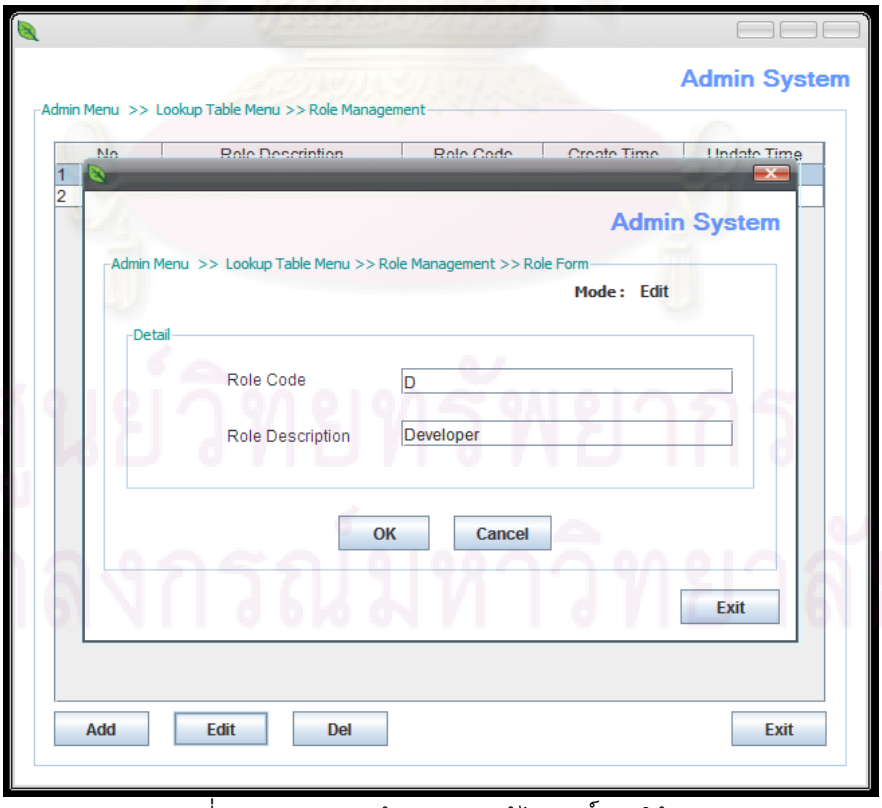
รูปที่ 59 ตัวอย่างหน้าจอการจัดการวัตถุประสงค์ Design Pattern



รูปที่ 60 ตัวอย่างหน้าจอการจัดการสิทธิ์การใช้งาน

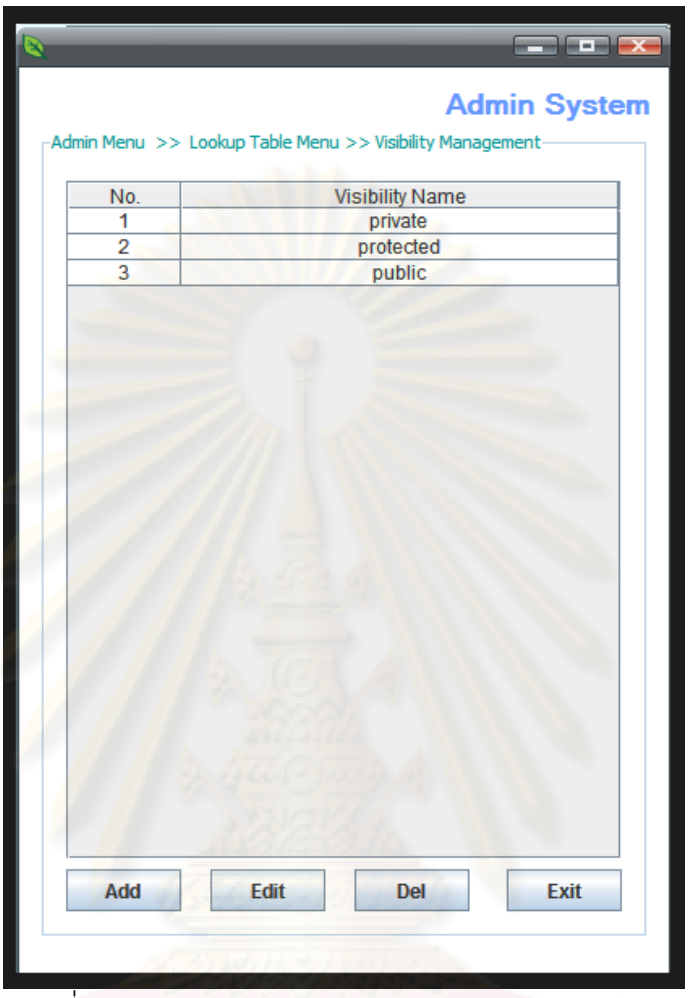


รูปที่ 61 ตัวอย่างหน้าจอการเพิ่มสิทธิ์การใช้งาน

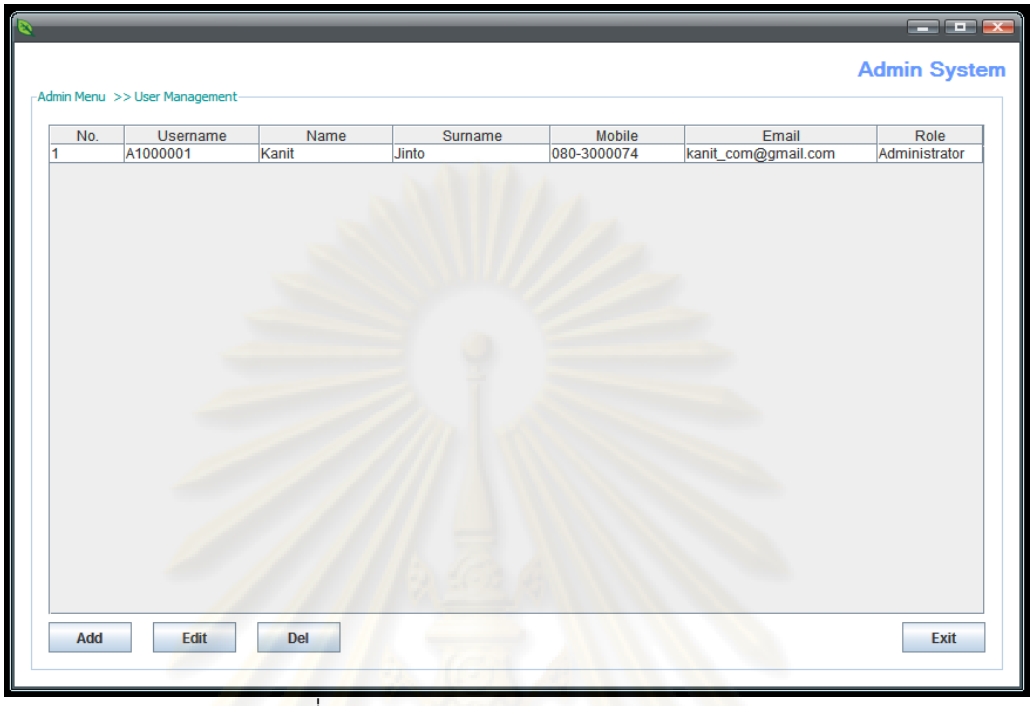


รูปที่ 62 ตัวอย่างหน้าจอการแก้ไขสิทธิ์การใช้งาน

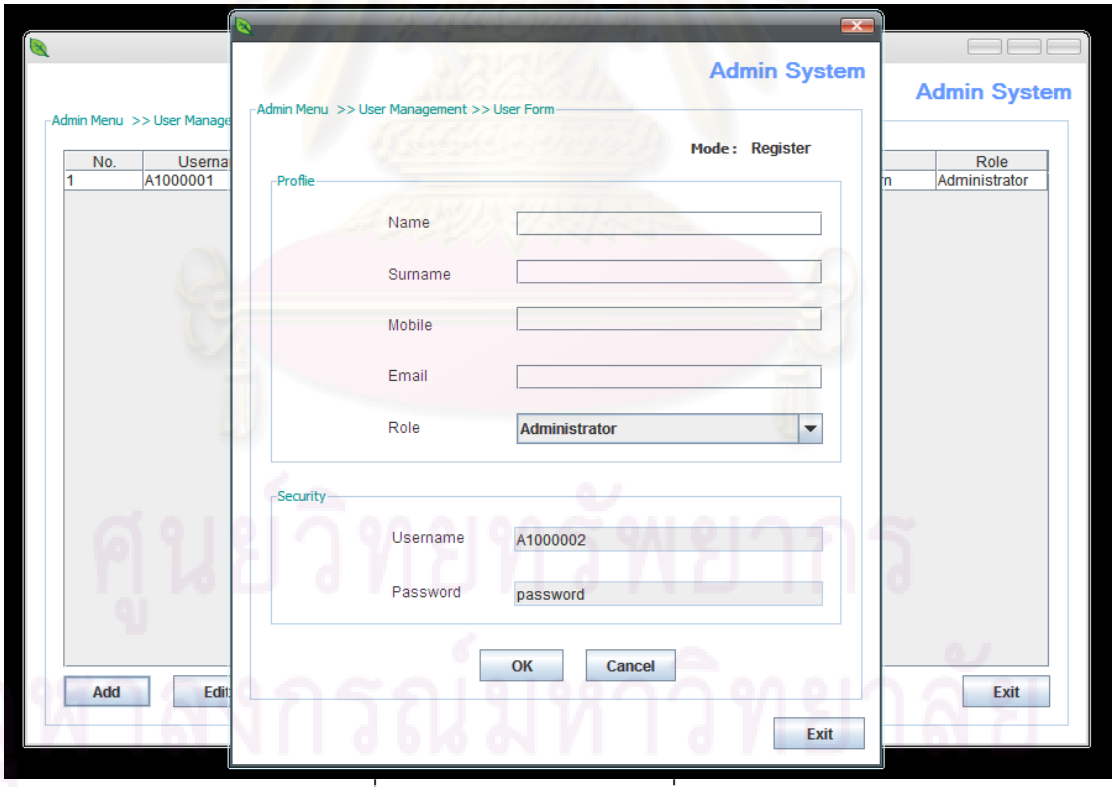
รูปที่ 63 ตัวอย่างหน้าจอการจัดการประเภทข้อมูลที่ต้องการตรวจสอบ



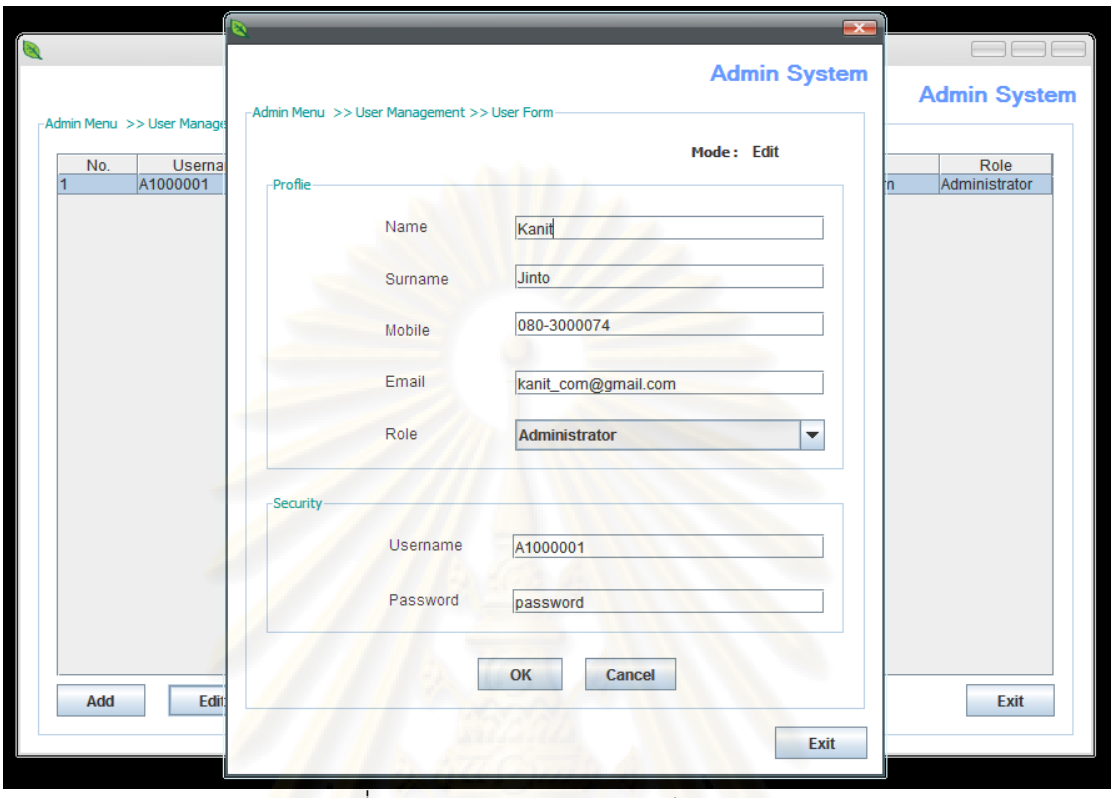
รูปที่ 64 ตัวอย่างหน้าจอการจัดการขอบเขตการมองเห็น



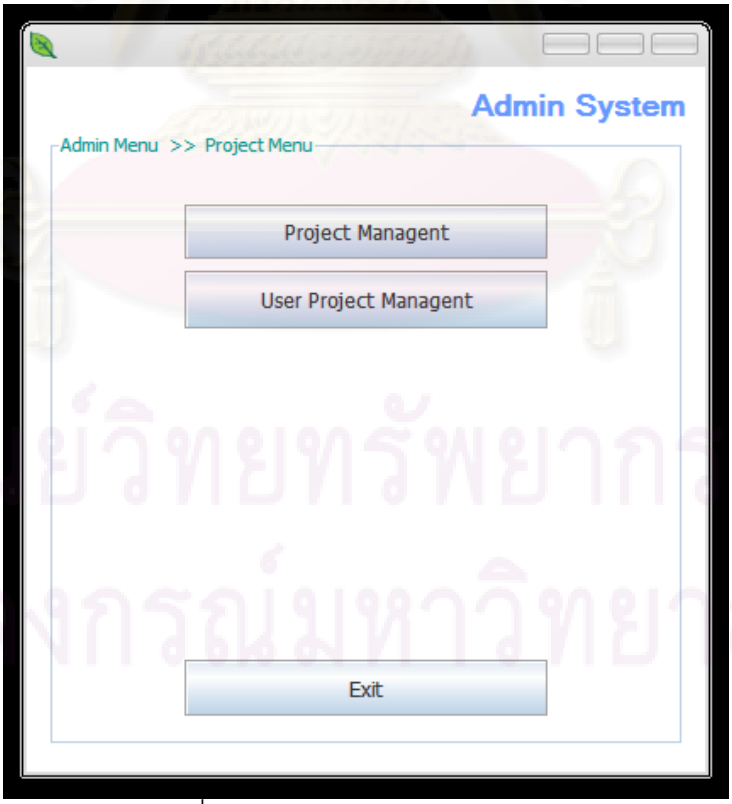
รูปที่ 65 ตัวอย่างหน้าจอการจัดการผู้ใช้งาน



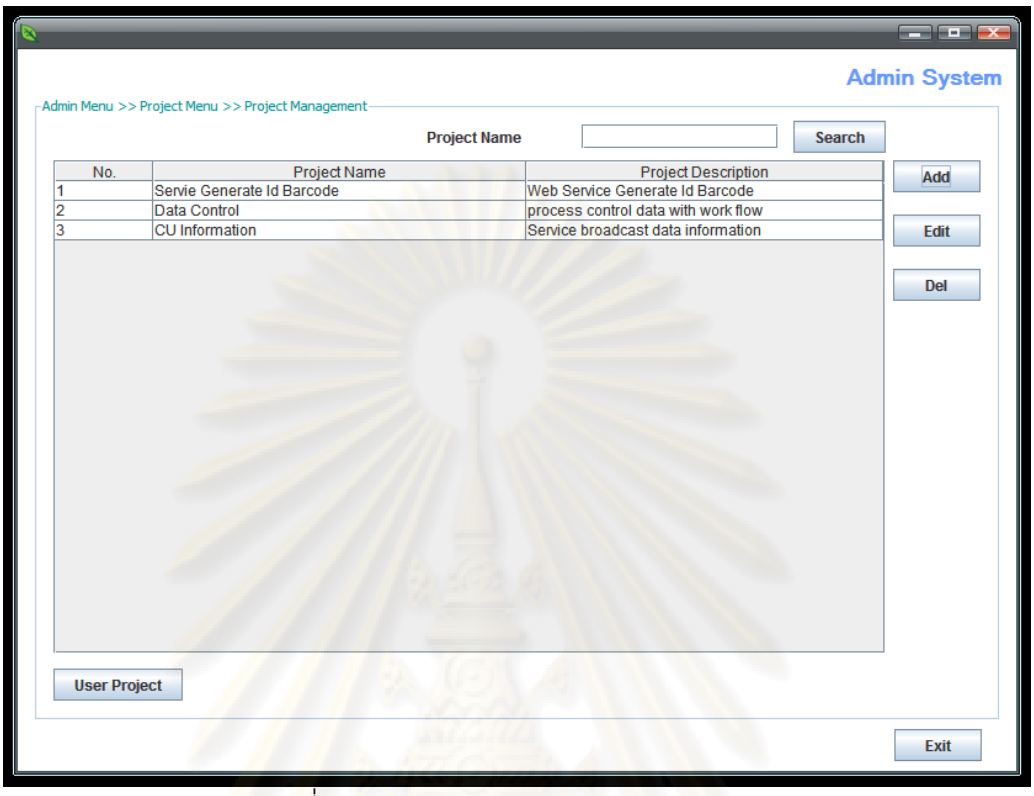
รูปที่ 66 ตัวอย่างหน้าจอการเพิ่มผู้ใช้งาน



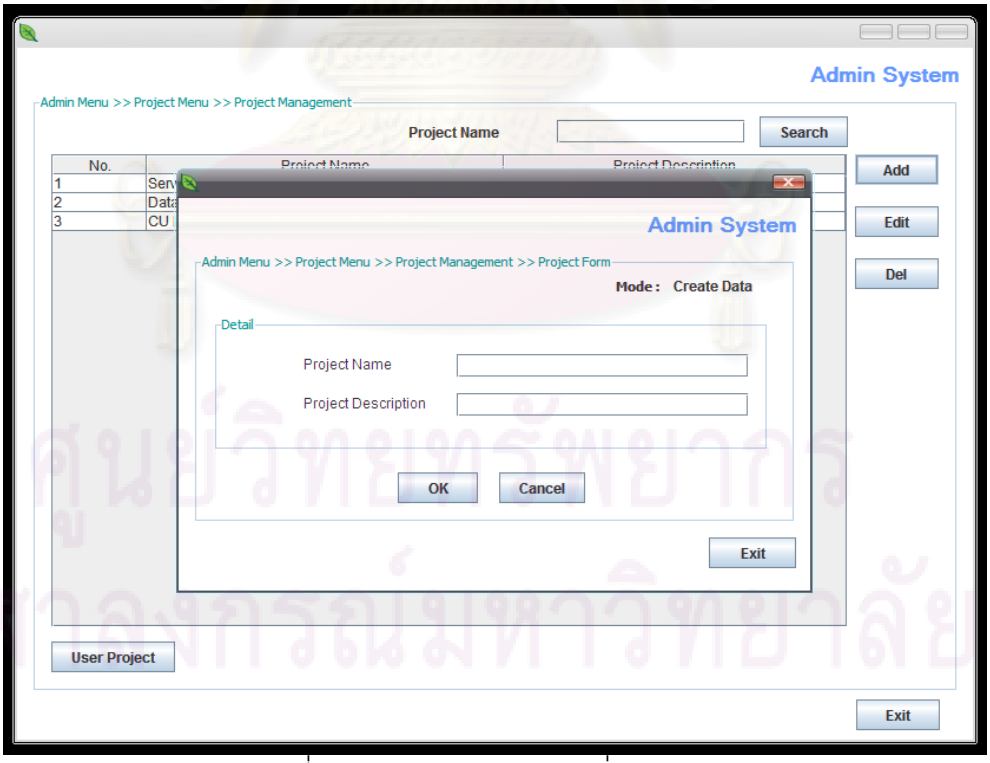
รูปที่ 67 ตัวอย่างหน้าจอการแก้ไขผู้ใช้งาน



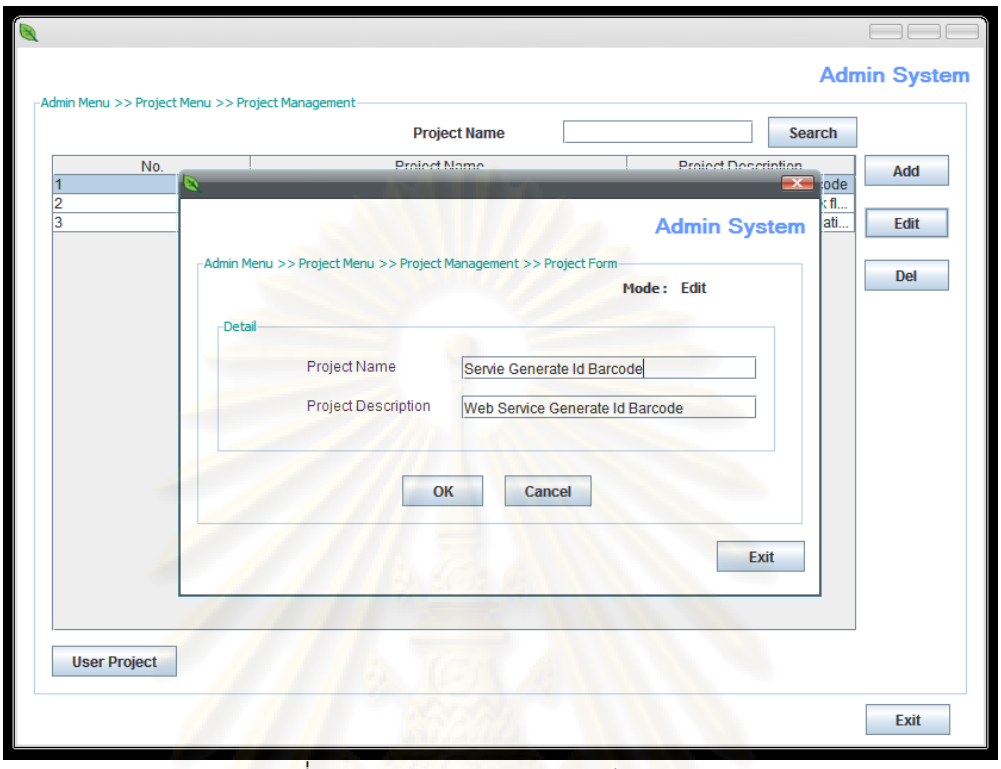
รูปที่ 68 ตัวอย่างหน้าจอเมนูโครงการ



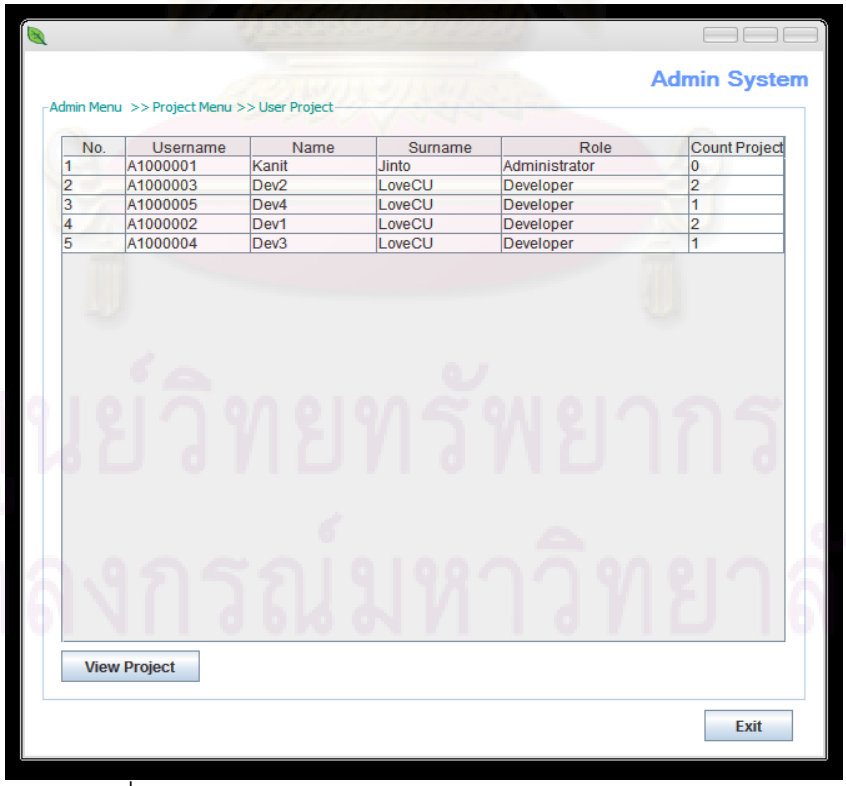
รูปที่ 69 ตัวอย่างหน้าจอการจัดการโครงการ



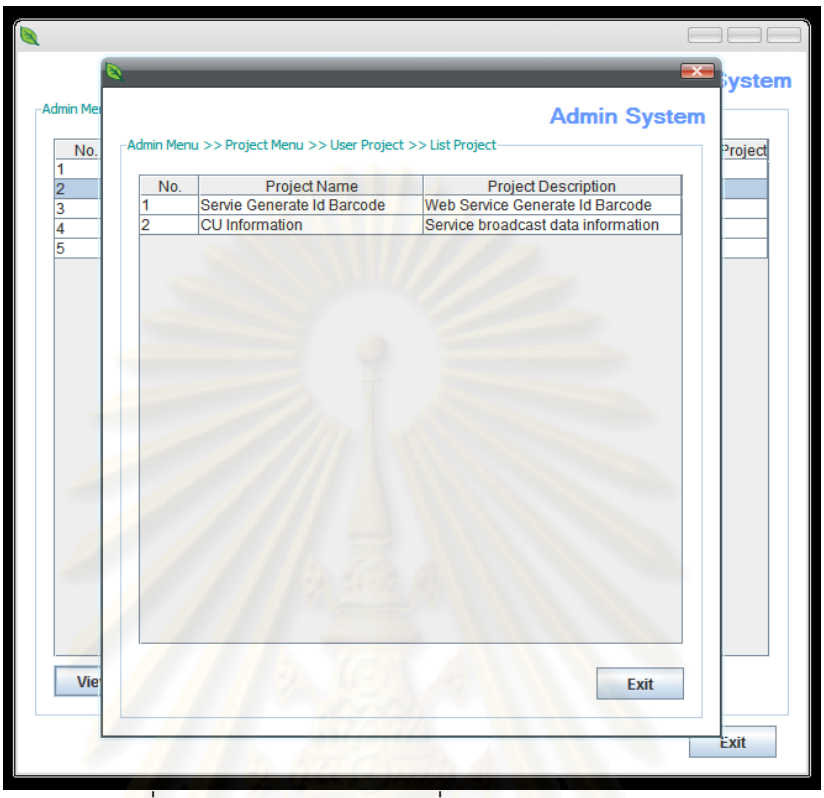
รูปที่ 70 ตัวอย่างหน้าจอการเพิ่มโครงการ



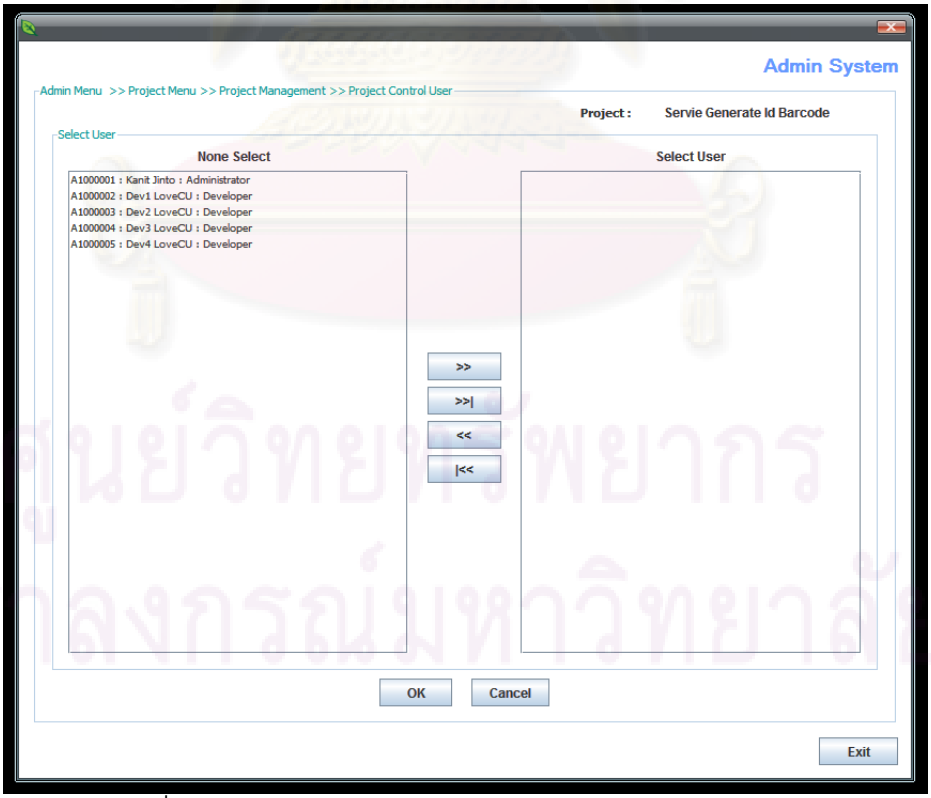
รูปที่ 71 ตัวอย่างหน้าจอการแก้ไขโครงการ



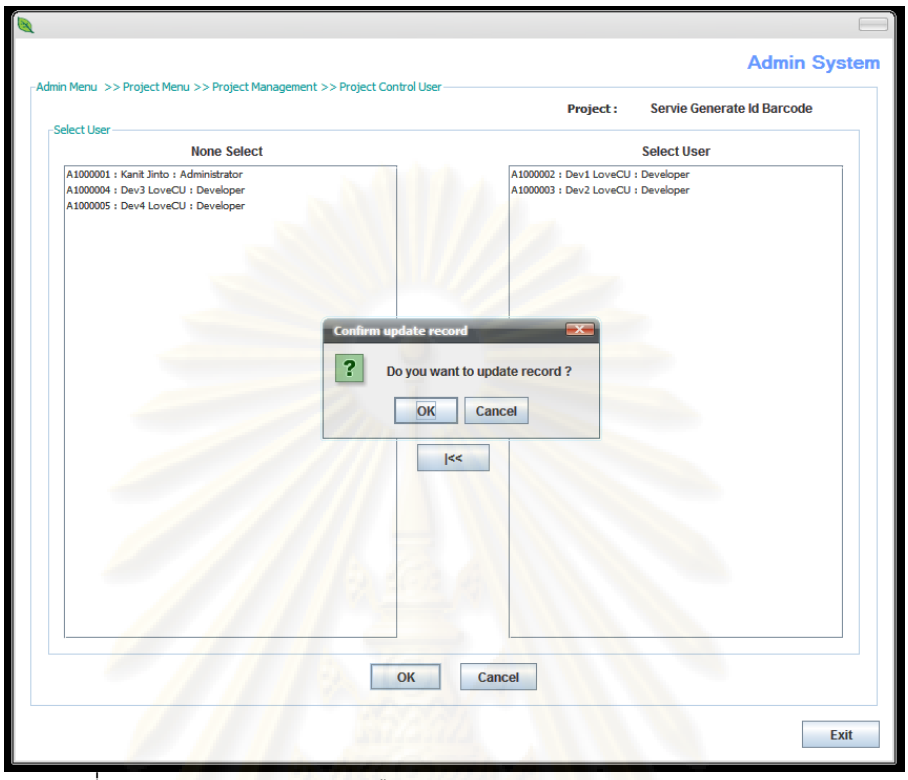
รูปที่ 72 ตัวอย่างหน้าจอการตวจสอบโครงการของผู้ใช้ระบบ



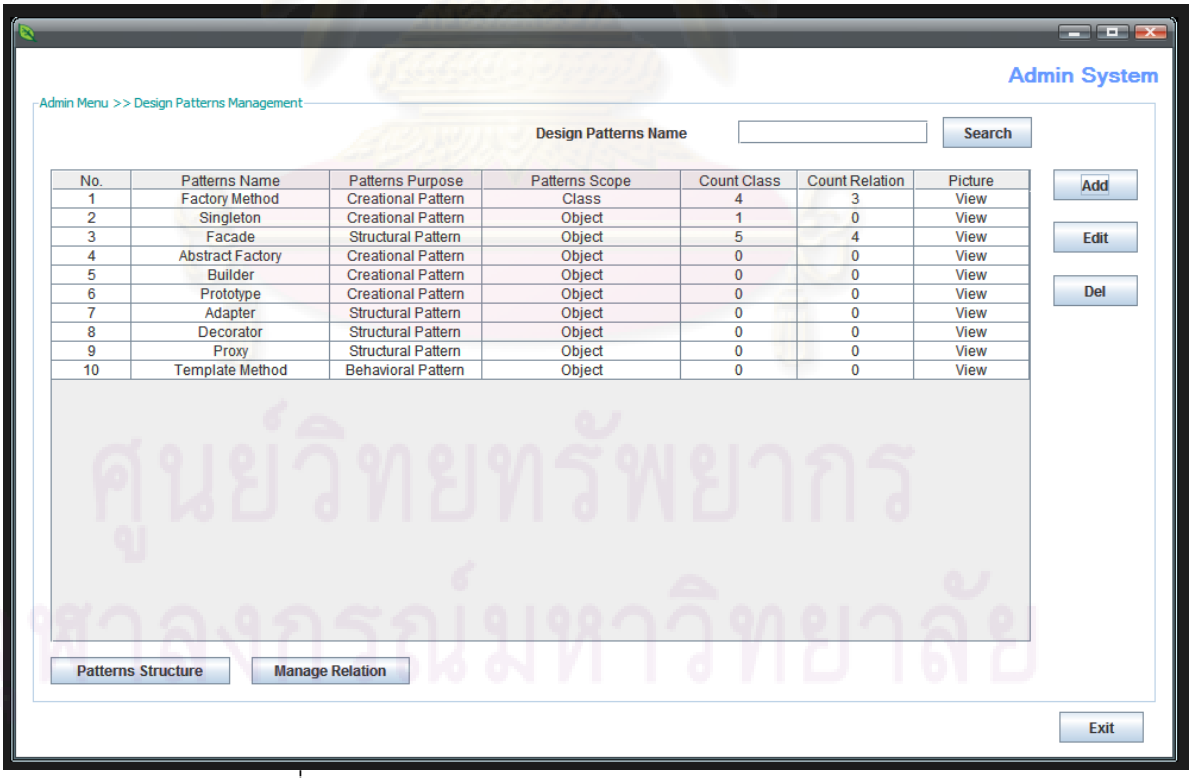
รูปที่ 73 ตัวอย่างหน้าจอรายชื่อโครงการของผู้ใช้ระบบ



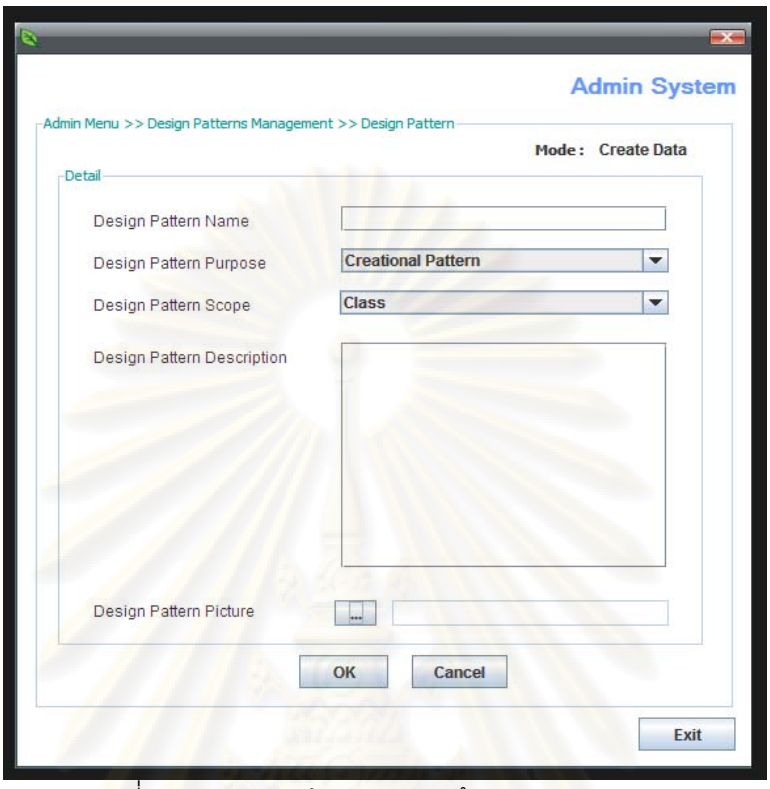
รูปที่ 74 ตัวอย่างหน้าจอการจัดการผู้ใช้งานเข้ามาในโครงการ



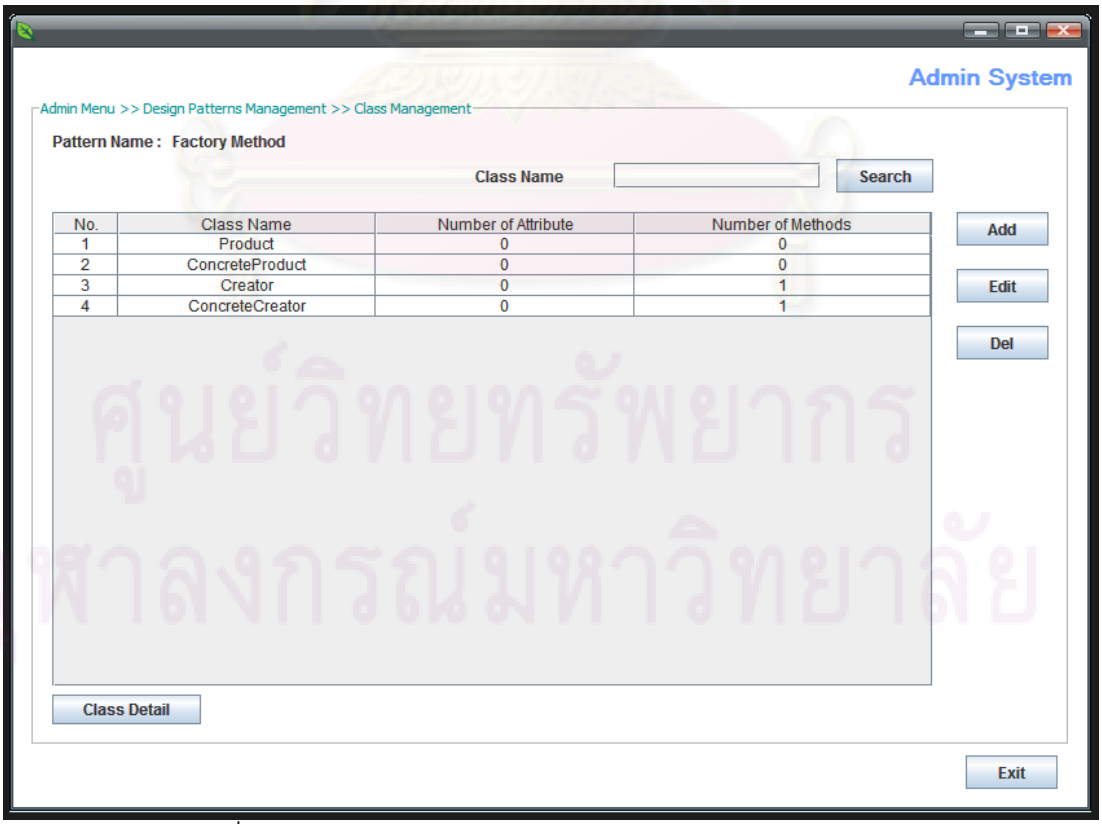
รูปที่ 75 ตัวอย่างหน้าจอการยืนยันการจัดการผู้ใช้งานเข้ามาในโครงการ



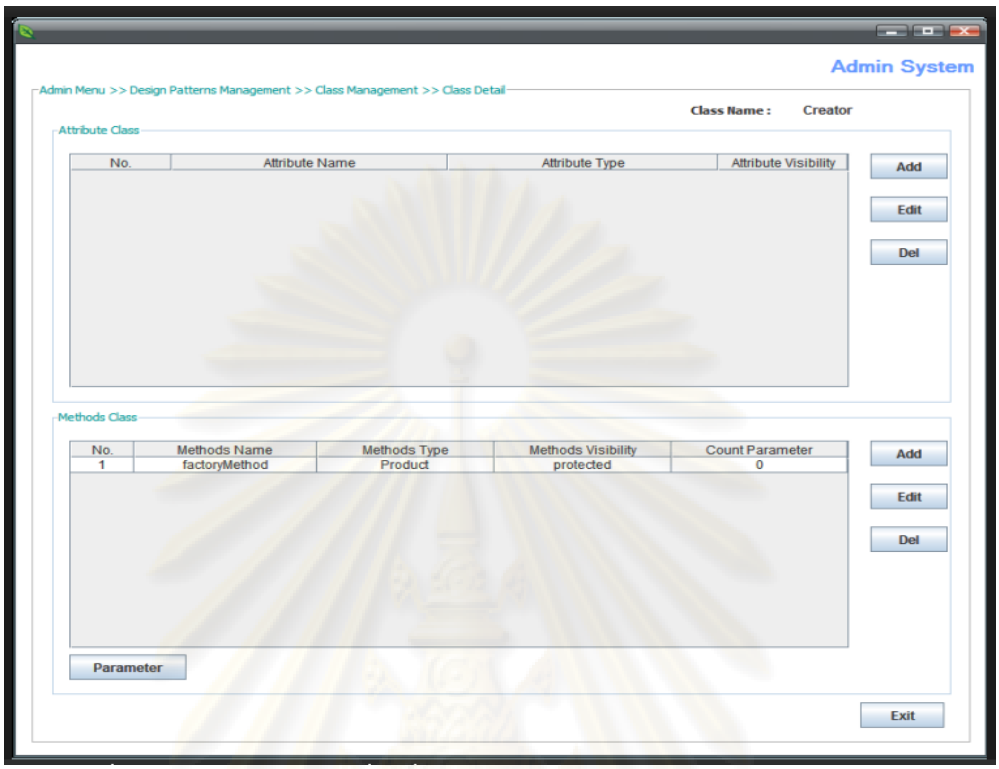
รูปที่ 76 ตัวอย่างหน้าจอการจัดการ Design Patterns



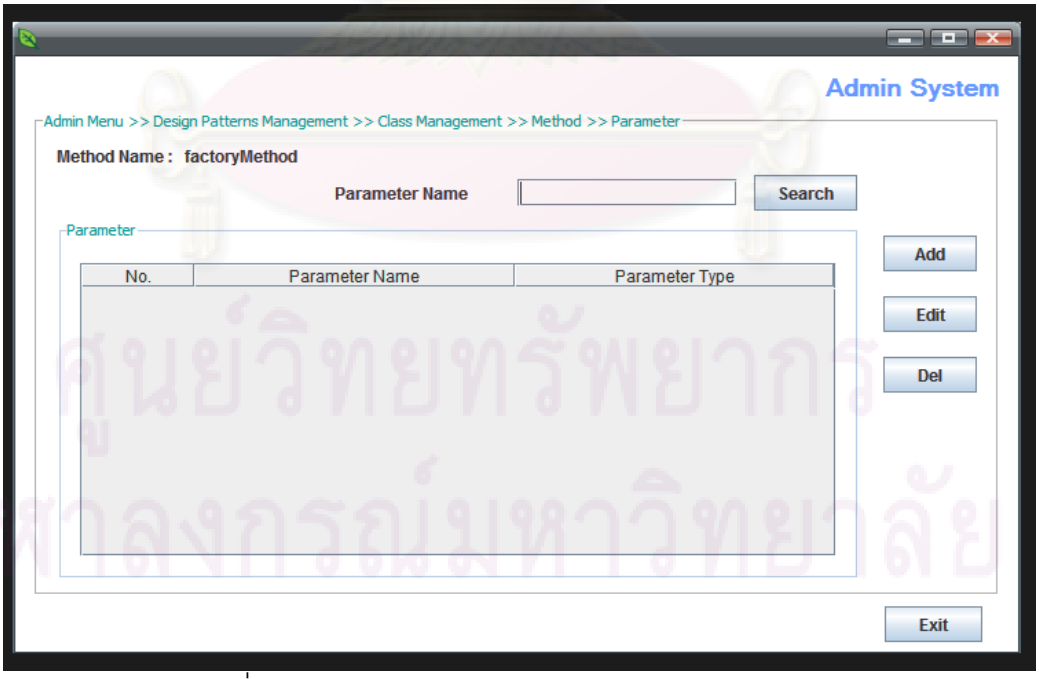
รูปที่ 77 ตัวอย่างหน้าจอการบันทึก Design Pattern



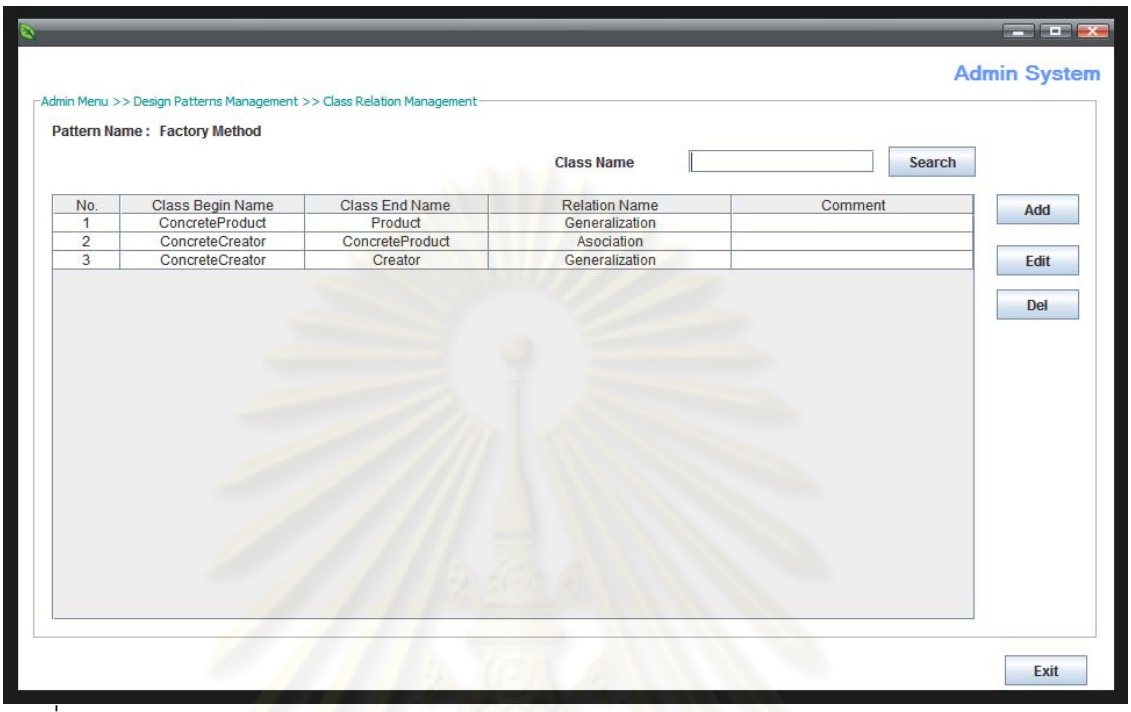
รูปที่ 78 ตัวอย่างหน้าจอการจัดการ Classของ Design Patterns



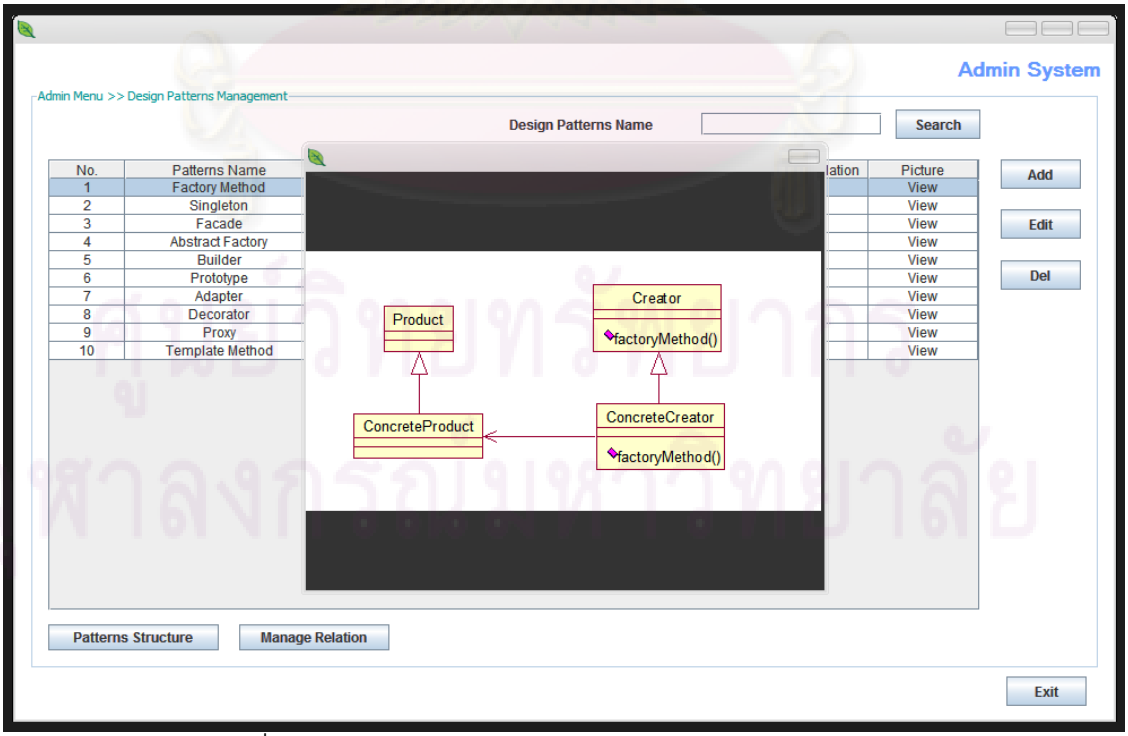
รูปที่ 79 ตัวอย่างหน้าจอการจัดการ Attribute และ Method ของ Design Patterns



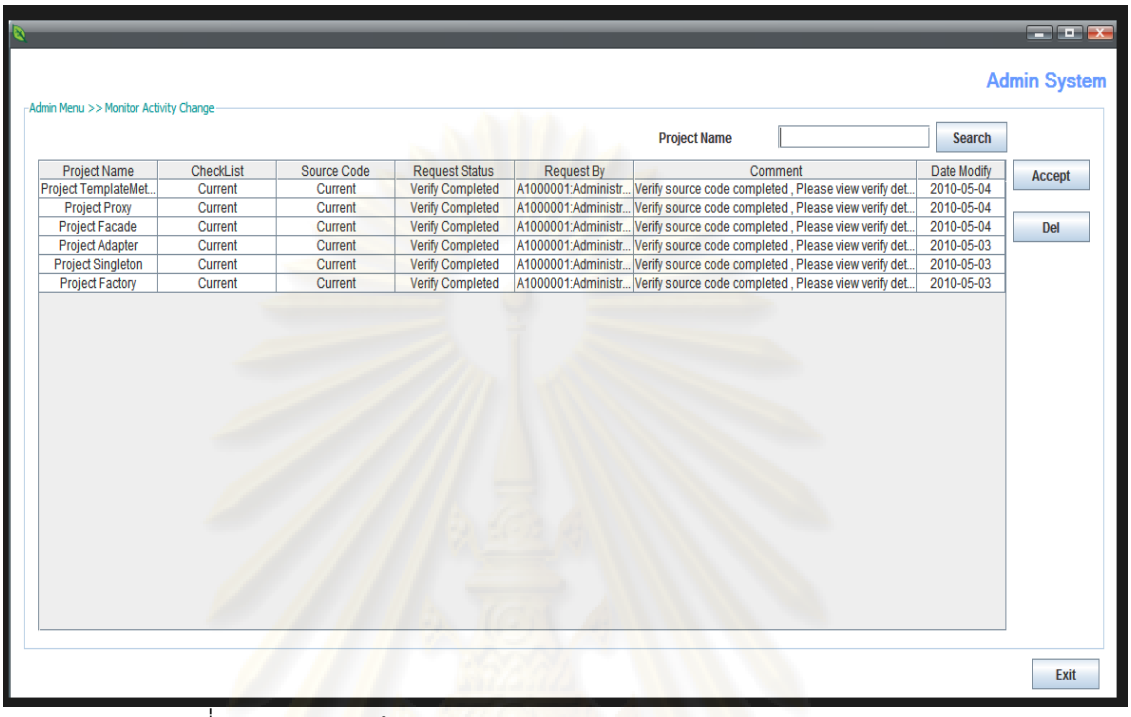
รูปที่ 80 ตัวอย่างหน้าจอการจัดการ Method Parameter



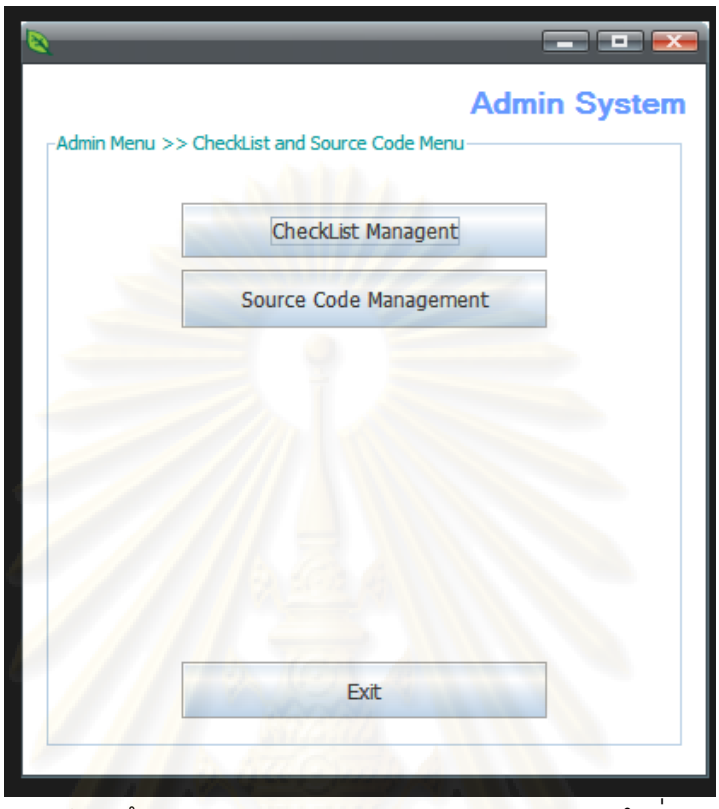
รูปที่ 81 ตัวอย่างหน้าจอการจัดการความสัมพันธ์ระหว่าง Class ต้นทางกับ Class ปลายทางของ Design Patterns



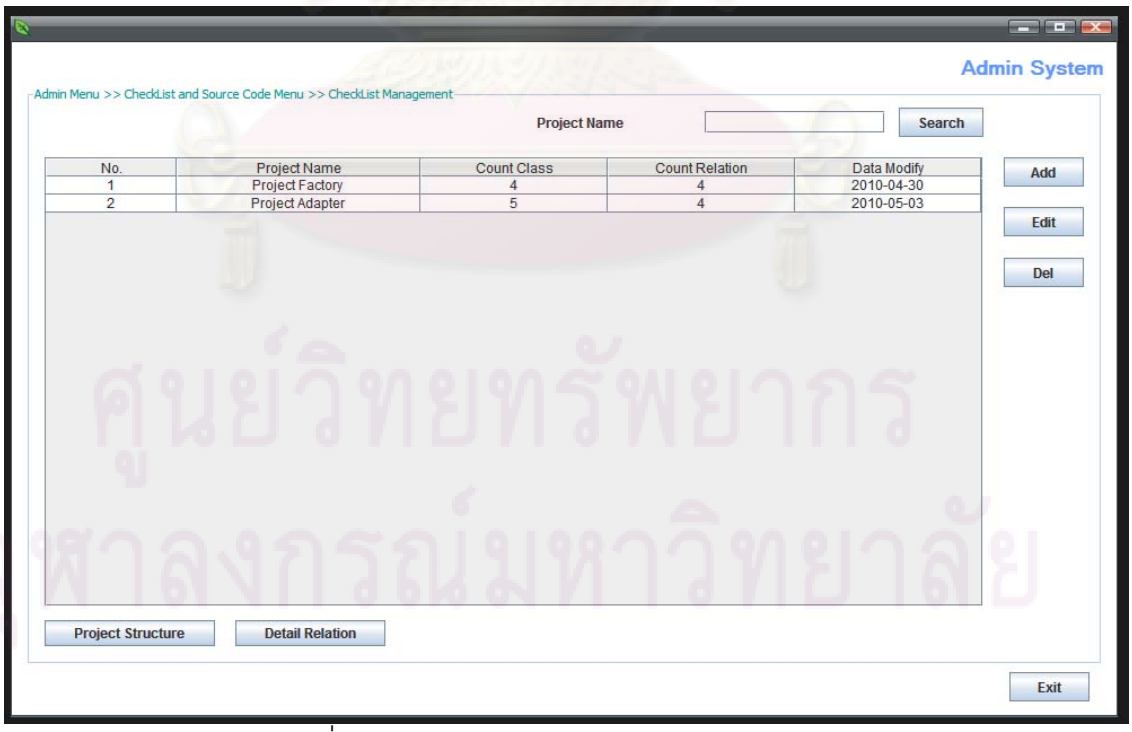
รูปที่ 82 ตัวอย่างหน้าจอการแสดงรูปภาพ Design Patterns



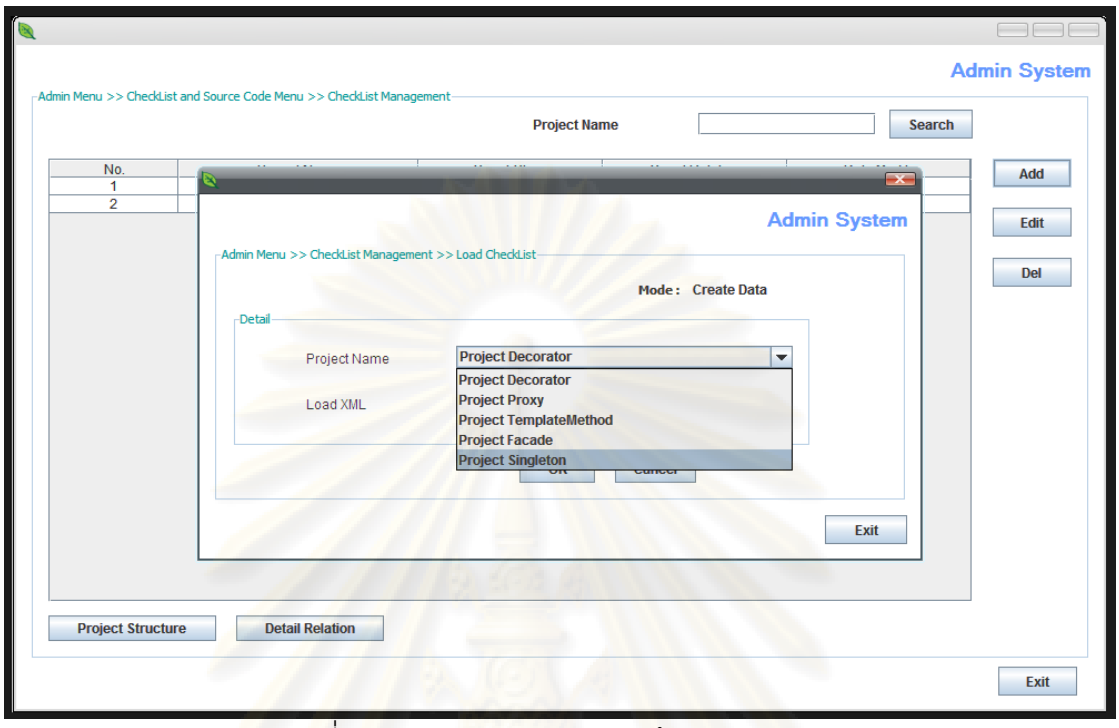
รูปที่ 83 ตัวอย่างหน้าจอการตรวจสอบสถานะของนักพัฒนาโปรแกรม



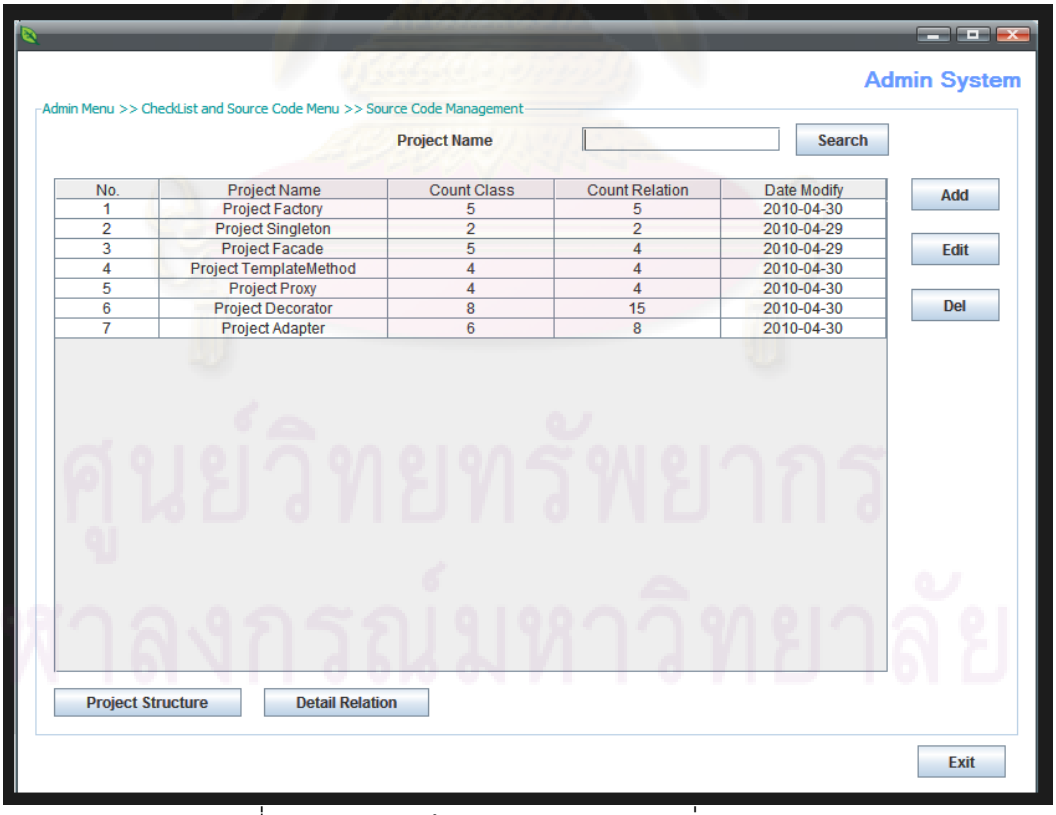
รูปที่ 84 ตัวอย่างหน้าจอเมนูการจัดการตรวจทานละจัดการชุดคำสั่งภาษาจาวา



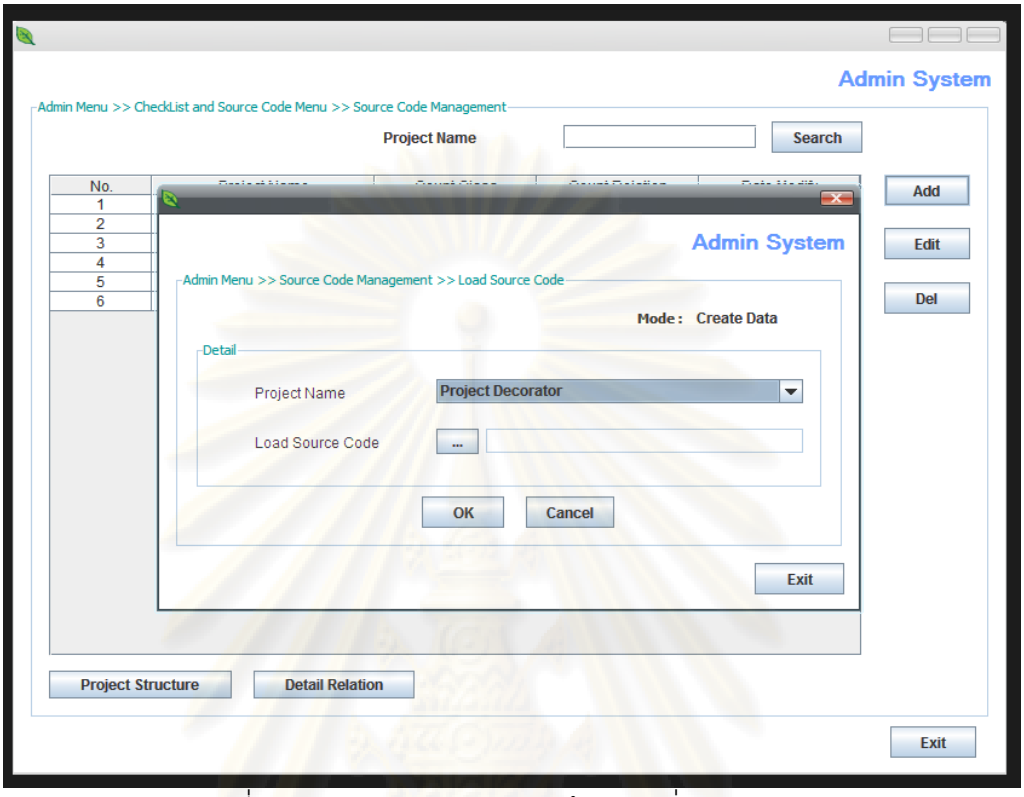
รูปที่ 85 ตัวอย่างหน้าจอการจัดการตัวตรวจทาน



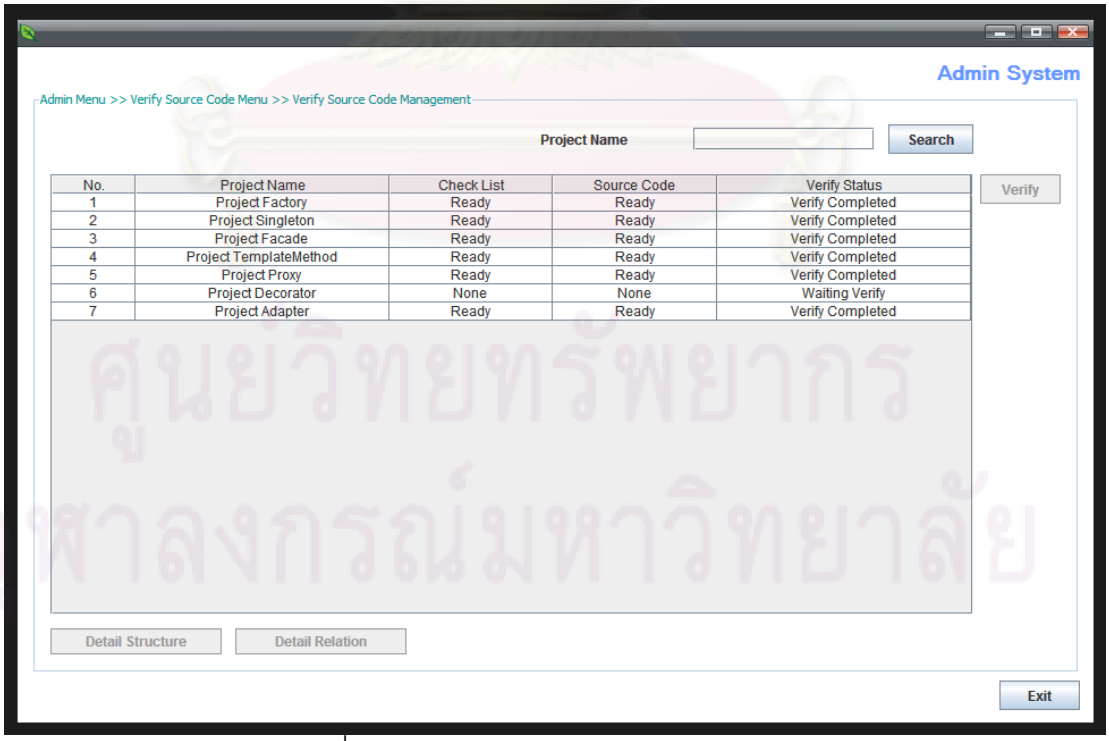
รูปที่ 86 ตัวอย่างหน้าจอการบันทึกตัวตรวจทาน



รูปที่ 87 ตัวอย่างหน้าจอการจัดการชุดคำสั่งภาษาจาวา



รูปที่ 88 ตัวอย่างหน้าจอการบันทึกชุดคำสั่งภาษาจาวา



รูปที่ 89 ตัวอย่างหน้าจอการจัดการตรวจทาน

ประวัติผู้เขียนวิทยานิพนธ์

นายคณิษฐ์ จินโต เกิดวันที่ 23 ตุลาคม พ.ศ.2524 สถานที่เกิดกรุงเทพมหานคร วุฒิการศึกษาระดับปริญญาตรี วิทยาศาสตรบัณฑิต (วท.บ.) มหาวิทยาลัยธรรมศาสตร์ ปี พ.ศ. 2548 ประสบการณ์การทำงาน ปี พ.ศ. 2548-2551 ตำแหน่ง Programmer บริษัท ยิบอินซอย จำกัด ปี พ.ศ. 2551-2552 ตำแหน่ง Developer บริษัท เอ็น ซีอาร์ จำกัด และปี พ.ศ. 2552-ปัจจุบัน ตำแหน่ง System Analyst บริษัท อ๊อฟติมัสซอฟต์ จำกัด