การให้บริการทางเว็บสำหรับระบบบัญชี

นายวศิน ตรีสินธุรส

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์
มหาวิทยาลัยเทคโนโลยีสุรนารี
ปีการศึกษา 2552

WEB SERVICE FOR ACCOUNTING SYSTEM

Wasin Treesinthuros

A Thesis Submitted in Partial Fulfillment of the Requirements for the

Degree of Master of Engineering in Computer Engineering

Suranaree University of Technology

Academic Year 2009

การให้บริการทางเว็บสำหรับระบบบัญชี

มหาวิทยาลัยเทคโนโลยีสุรนารี อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

คณะกรรมการสอบวิทยานิพนธ์

_____

(รศ. ดร.กิตติศักดิ์ เกิดประสพ)
ประธานกรรมการ

_____

(ผศ. ดร.พิชโยทัย มหัทธนาภิวัฒน์)
กรรมการ (อาจารย์ที่ปรึกษาวิทยานิพนธ์)

_____

(รศ. ดร.นิตยา เกิดประสพ)
กรรมการ

_____

(ผศ. ดร.ปรเมศวร์ ห่อแก้ว)
กรรมการ

_____

(ศ. ดร.ชูกิจ ลิมปีจำนงค์)
รองอธิการบดีฝ่ายวิชาการ

_____

(รศ. น.อ. ดร.วรพจน์ ขำพิศ)
คณบดีสำนักวิชาวิศวกรรมศาสตร์

วศิน ตรีสินธุรส : การให้บริการทางเว็บสำหรับระบบบัญชี (WEB SERVICE FOR COUNTING SYSTEM) อาจารย์ที่ปรึกษา : ผู้ช่วยศาสตราจารย์ ดร.พิชโยทัย มหัธนาภิวัฒน์, 99 หน้า.

การทำงานในบริษัท หรือองค์กรต่าง ๆ ในปัจจุบันนั้น ล้วนมีความเกี่ยวข้องกับการเงินทั้งสิ้น และการที่จะต้องดูแลสภาวะทางการเงินขององค์กรนั้น จำเป็นต้องมีผู้เชี่ยวชาญเฉพาะทางมาดูแลเป็นพิเศษ โดยผู้ดูแลนั้นเป็นเพียงส่วนหนึ่งของระบบการดูแลทั้งหมด แต่ส่วนที่มีผลต่อการดูแลมากคือ ตัวโปรแกรมบัญชีขององค์กรนั้น ๆ ผู้วิจัยจึงมีความสนใจที่จะสร้างระบบบริการทางเว็บสำหรับช่วยงานผู้เขียนโปรแกรมเพื่อทำงานในงานบัญชี เพื่อช่วยในการเขียนโปรแกรมสำหรับงานทางบัญชีทำได้สะดวกมากยิ่งขึ้น

ผู้วิจัยได้ทำการพัฒนาระบบบัญชีโดยใช้เทคโนโลยี Web Service โดยทำการสร้างโมดูล ซึ่งประกอบด้วยการคำนวณงบต่าง ๆ เช่น งบทดลอง งบดุล งบกำไรขาดทุน เป็นต้น จะทำงานโดยการรับค่าเพื่อนำมาประมวลผลของงบการเงิน และการส่งผลลัพธ์การคำนวณกลับคืนมายังโปรแกรมเตรียมเอาไว้ให้ผู้พัฒนาเอารูปแบบไปพัฒนาระบบบัญชีของตนเอง รูปแบบการทำงานคือให้ผู้ใช้ทำการระบุข้อมูลทางการเงินผ่านทางโปรแกรมของผู้ใช้เอง และทำการเรียกใช้การให้บริการทางเว็บสำหรับระบบบัญชีซึ่งอยู่ที่ฝั่งเซิฟเวอร์ ระบบจะทำการประมวลผลและส่งค่ากลับมาเป็นผลลัพธ์ทางบัญชีที่ต้องการ หรืองบการเงินในรูปแบบของ XML เพื่อที่ผู้พัฒนาระบบบัญชีไม่จำเป็นต้องเขียนระบบการทำงานทั้งหมด แต่เรียกใช้การบริการจากระบบที่ผู้วิจัยพัฒนาขึ้นให้ทำงานร่วมกับโปรแกรมของนักพัฒนาระบบได้ตลอดเวลา เป็นการประหยัดเวลา และเป็นมาตรฐานร่วมกัน

สาขาวิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนักศึกษา_____

ปีการศึกษา 2552 ลายมือชื่ออาจารย์ที่ปรึกษา_____

WASIN TREESINTHUROS : WEB SERVICES FOR ACCOUNTING SYSTEM. THESIS ADVISOR : ASST. PROF. PITCHAYOTHAI MAHATTHANAPIWAT, Ph.D., 99 PP.

Nowadays, working in companies or organizations is entirely related with finance and it is necessary to use some specialist to manage financial conditions of organization. Such specialist is only a part of whole management system, but the most affective part of system is accounting application software of those organizations. Therefore, the researcher paid an attention on creating a web service system to help programmers, who work with accounting, for more convenience on accounting system programming.

The researcher developed accounting system with Web Service technology by creating a module consisted of calculated statements such as trial balances, balance sheets, profit and loss statements etc. This system was operated by receiving all values for processing financial statements and then sent calculated results back to the program for programmers' developing as their own accounting systems. Operation format was to let users specify financial data via their own programs and request servers for Web Service System for Accounting. The system would process and sent back financial results or statements as needed in format of XML for accounting system developers not to program entire working system, but only request services from this system that developed by the research and operate together with their own developed system for saving times and being co-standard.

School of Computer Engineering Student's Signature_____

Academic Year 2009 Advisor's Signature_____

กิตติกรรมประกาศ

วิทยานิพนธ์นี้สามารถสำเร็จลุล่วงด้วยดี ผู้วิจัยขอขอบพระคุณ คณะบุคคลดังต่อไปนี้ ที่ได้ร่วมเป็นส่วนหนึ่ง ที่ทำให้วิทยานิพนธ์สำเร็จได้ทั้งในด้านทางวิชาการ ด้านดำเนินงานวิจัย และในด้านอื่น ๆ

ผู้ช่วยศาสตราจารย์ ดร.พิชโยทัย มหัทธนาภิวัฒน์ อาจารย์ที่ปรึกษาวิทยานิพนธ์

รองศาสตราจารย์ ดร.กิตติศักดิ์ เกิดประสพ หัวหน้าสาขาวิชา รองศาสตราจารย์ ดร.นิตยา เกิดประสพ อาจารย์ประจำสาขาวิชาวิศวกรรมคอมพิวเตอร์ สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

ขอขอบคุณ คุณสุพรรษา น้อยไทย สำหรับความรู้เรื่องบัญชี

ขอขอบคุณ คุณปัทมา ตรีสินธุรส ภรรยาที่เป็นกำลังใจให้ตลอดเวลา

ขอขอบคุณ คุณจริยาพร ศรีวิไลลักษณ์ ที่ช่วยตรวจสอบรูปแบบวิทยานิพนธ์

และสุดท้าย ขอขอบคุณ คุณพ่อคมกฤช และคุณแม่วนิดา ตรีสินธุรส ที่ให้การเลี้ยงดูอบรมสั่งสอน และส่งเสริมให้การศึกษาเล่าเรียนเป็นอย่างดีมาโดยตลอด จนทำให้วันนี้ผู้วิจัยสามารถประสบความสำเร็จได้

วศิน ตรีสินธุรส

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญรูป

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของปัญหาการวิจัย

รูปแบบการทำงานขององค์กรในปัจจุบัน สามารถแบ่งงานได้เป็นสองส่วน ส่วนแรกเป็นงานหลักที่ติดต่อระหว่างหน่วยงานในองค์กร ส่วนที่สองเป็นงานภายในองค์กรย่อยของบริษัท เช่น งานบริหารจัดการบุคคล งานจัดการทางการเงิน งานการจัดการทางบัญชี งานการจัดซื้อ งานทางพัสดุ เป็นต้น ส่วนใหญ่การจัดการระบบงานขึ้นมาขององค์กรหนึ่ง ๆ มุ่งหวังให้การติดต่อประสานงานระหว่างหน่วยงานและการดำเนินงานภายในหน่วยงาน เป็นไปด้วยความเรียบร้อย ราบรื่น ไม่ติดขัด มีความถูกต้อง ครบถ้วน ระบบงานมีเสถียรภาพ มีการบริหารจัดการที่ดี บุคลากรมีระดับความสามารถได้มาตรฐาน ลดปัญหาความผิดพลาดที่เกิดจากตัวบุคคล (Human Error) ให้มากที่สุด ตัวชี้วัดความสำเร็จในระดับนี้ได้อย่างหนึ่งคือ บุคลากรสามารถหมุนเวียนทำงานแทนกันได้ในยามจำเป็น โดยไม่ทำให้เวลาการทำงานช้าลง ไม่กระทบต่อเวลารวมของการทำงานทั้งระบบ

ระบบงานที่มีความละเอียดอ่อนที่สุด ในการบริหารจัดการคือ ระบบงานที่เกี่ยวกับการเงินและงานทางบัญชี เนื่องจากเป็นหัวใจสำคัญขององค์กร ทั้งการควบคุมค่าใช้จ่าย การทราบสถานะทางการเงินและบัญชีจะทำให้บริษัททราบสถานะของตัวเอง และคาดการณ์ปัญหาต่าง ๆ สามารถป้องกันหรือแก้ไขปัญหาทางด้านการเงินได้ทัน งานทางการเงินและบัญชีมีขอบเขตของงานโดยสรุปดังนี้

- งานการเงิน มีขอบเขตการทำงานเกี่ยวกับการบริหารเงินขององค์กร โดยมีอำนาจในการจัดการตามระเบียบการเงินของหน่วยงาน การรับ-จ่ายในกิจการต่าง ๆ ของบริษัท รวมไปถึงการจัดการรายได้ของพนักงานด้วย
- งานทางบัญชี เป็นบันทึกร่องรอยต่าง ๆ ของกิจกรรมทางการเงินที่เกิดขึ้น เพื่อจัดทำรายงาน ตรวจสอบสถานะและความถูกต้อง เป็นตัวบ่งชี้ความสามารถ การจัดการในระบบการเงินของหน่วยงาน

ในยุคปัจจุบัน ได้มีการนำเอาคอมพิวเตอร์เข้ามาทำงานภายในองค์กรมากขึ้น จากเดิมที่เป็นเพียงงานบางงาน แต่บางองค์กรในปัจจุบันได้ใช้โปรแกรมคอมพิวเตอร์บริหารงานทั้งระบบทั้งหมด ทำให้พบว่าการใช้ระบบคอมพิวเตอร์กลายเป็นเรื่องธรรมดาของการจัดการองค์กรไปแล้ว ทำให้งานบางอย่างที่เคยใช้ตัวบุคคลเป็นผู้บริหารจัดการได้มอบหมายให้เป็นภาระของโปรแกรม

คอมพิวเตอร์จัดการ ส่งผลให้เกิดความสะดวกสบายต่อผู้ใช้โปรแกรม และที่สำคัญยังมีความแม่นยำสูงกว่าบุคลากรที่ทำงานในงานเดียวกัน ทำให้ในตอนนี้การเขียนโปรแกรมบริหารการจัดการขององค์กรที่เกิดขึ้นใหม่ได้รับความนิยมเป็นอย่างมาก ในระบบงานที่กล่าวถึงระบบงานบัญชีเป็นระบบแรก ๆ ที่มีการนำโปรแกรมคอมพิวเตอร์ ที่เขียนขึ้นมาใช้งานช่วยการทำงานของมนุษย์ เนื่องจากการดำเนินการตาม แบบดั้งเดิมด้วยการ บันทึกลงสมุดบัญชี มักจะพบความผิดพลาดของการบันทึกบ่อยครั้ง เป็นเพราะประสิทธิภาพของคนมีจำกัด มีความเหนื่อยล้าได้เมื่อทำงานนาน ๆ และการเขียนด้วยลายมือของแต่ละคน ที่อาจทำให้เกิดปัญหาความผิดพลาดโดยไม่ตั้งใจ เช่นการเขียนที่ไม่ชัดเจน เลข 1 ที่คล้าย เลข 7 เป็นต้น นอกจากนี้ การใช้กระดาษก็ก่อให้เกิดความสิ้นเปลืองอย่างต่อเนื่องไม่สิ้นสุด และตัวระบบงานบัญชีก็มีความซับซ้อน มีขั้นตอนที่มาก ต้องการความแม่นยำสูง ระบบมีการตรวจสอบย้อนกลับได้ และผลลัพธ์ต้องไม่ผิดพลาดในทุกขั้นตอน ความผิดพลาดในการคำนวณ หรือการทำงานที่ไม่ถูกต้องตามขั้นตอนแต่เพียงเล็กน้อย ทำให้ผลลัพธ์ไม่สามารถนำมาใช้งานได้ และเสียเวลาในการค้นหาข้อผิดพลาดที่เกิดในระบบงาน ความจำเป็นดังกล่าวจึงทำให้แนวโน้มของหน่วยงานต่าง ๆ พยายามนำโปรแกรมมาช่วยทำงานในระบบบัญชี และปรับปรุงระบบการทำงานให้เป็นไปตามมาตรฐาน

การนำระบบคอมพิวเตอร์มาใช้งานมีอยู่ 2 แนวทาง คือ (1) การใช้โปรแกรมสำเร็จรูป และ (2) การพัฒนาโปรแกรมขึ้นมาใหม่

การใช้งานทั้งสองแบบ มีข้อแตกต่างกัน ดังนี้

การใช้โปรแกรมสำเร็จรูป

ข้อดี

- สามารถจัดหาได้ง่าย มีจำหน่ายตั้งแต่ราคาถูก จนถึงราคาแพง
- มีความสามารถในการคำนวณอย่างดี ไม่มีผิดพลาดหากผู้ใช้กรอกข้อมูลถูกต้อง
- ส่งต่อผลการทำงานได้ง่าย
- สับเปลี่ยนการทำงานของบุคลากรได้

ข้อเสีย

- บางระบบงานอาจไม่เหมาะกับการทำงาน เนื่องจากส่วนใหญ่โปรแกรม สำเร็จรูปจะออกแบบการทำงานในลักษณะทั่ว ๆ ให้สามารถใช้งานได้กับทุก ๆ หน่วยงาน งานที่มีลักษณะเฉพาะตัวจึงอาจไม่เหมาะกับโปรแกรมสำเร็จรูป
- ถ้าองค์กรมีขนาดใหญ่อาจมีปัญหาเรื่องค่าใช้จ่ายเกี่ยวกับลิขสิทธิ์ที่ต้องจ่ายเป็นจำนวนมาก เนื่องจากการคำนวณค่าใช้จ่ายขึ้นกับจำนวนเครื่องใช้งาน

- ความเข้ากันได้ระหว่างระบบบัญชีแต่ละระบบ หรือระบบบัญชีกับระบบงานอื่นอาจมีปัญหาได้ เพราะไม่ได้พัฒนามาจากที่เดียวกัน

การพัฒนาโปรแกรมขึ้นมาใหม่

ข้อดี

- สามารถทำงานได้ตรงตามความต้องการของหน่วยงาน
- สามารถทำงานร่วมกับระบบอื่น ๆ ได้เนื่องจากมีการออกแบบและพัฒนาระบบงานร่วมกัน การทำงานโดยรวมจะมีประสิทธิภาพ
- กำหนดแพลตฟอร์มที่ใช้ในการพัฒนาได้
- ไม่มีปัญหาเรื่องการกำหนดจำนวนสิทธิ์ในการใช้งาน สามารถขยาย จำนวนบุคลากรที่ทำงานได้ตามต้องการ

ข้อเสีย

- ต้องมีค่าใช้จ่ายในการดูแลรักษาและปรับปรุงโปรแกรมตลอดระยะเวลาการใช้งาน ยกเว้นหน่วยงานนั้นมีความสามารถที่จะทำเองได้
- ในช่วงการพัฒนาต้องใช้เวลานาน อาจใช้เวลาตั้งแต่ 1 ปีขึ้นไป
- ต้องการความร่วมมือร่วมใจจากบุคลากรที่ทำงานในการให้ข้อมูล ความต้องการต่าง ๆ แก่ผู้พัฒนาระบบอย่างครบถ้วน ถ้าบุคลากรไม่ให้ความร่วมมือ หรือไม่มีเวลาให้ข้อมูล ระบบงานที่พัฒนาอาจล้มเหลวได้
- ถ้าเป็นเทคโนโลยีที่ทีมพัฒนาไม่คุ้นเคย อาจใช้เวลาในการพัฒนา

ในปัจจุบัน หน่วยงานที่มีงบประมาณพอเพียง มักจะเลือกใช้โปรแกรมที่พัฒนาขึ้นมาใช้เฉพาะหน่วยงานของตนแทนการซื้อโปรแกรมสำเร็จรูป เนื่องจากต้องการระบบงานที่เข้ากันได้ กับระบบการทำงานขององค์ที่มีอยู่ก่อนหน้า จะได้ไม่ต้องปรับตัวมาก และต้องการให้ระบบทั้งหมดทำงานเข้ากันได้เป็นเนื้อเดียวกัน ไม่สะดุดติดขัด ไม่ต้องเพิ่มขั้นตอนปรับแก้ข้อมูลก่อนการส่งต่อให้ระบบงานอื่น ๆ ที่ทำงานกันต่อเนื่อง ขณะเดียวกันในมุมมองของผู้พัฒนาระบบมาใช้งาน ความที่มีเทคโนโลยีให้เลือกใช้มากกมาย หลากหลายเทคโนโลยีแต่ละอย่าง มีรายละเอียดมากกว่าจะศึกษาให้ชำนาญ นำมาใช้งานได้คล่องแคล่ว ใช้งานได้จริง ต้องใช้เวลาศึกษาค่อนข้างนาน โปรแกรมเมอร์ส่วนมากจึงถนัดและมีความชำนาญ มีความเข้าใจลึกซึ้งในเทคโนโลยีเดียว หากเทคโนโลยีนั้นไม่ทันสมัย หรือมีการปรับเปลี่ยนไปใช้เทคโนโลยีอื่น ทีมงานพัฒนาจะต้องปรับตัว ศึกษาใหม่ ทำให้มีค่าใช้จ่ายและเสียเวลา อาจเป็นสาเหตุให้เวลาในการพัฒนาไม่เป็นไปตามตารางเวลา ไม่สามารถส่งงานได้ตามกำหนด เป็นเรื่องเสียหายทั้งผู้พัฒนาและหน่วยงานที่ต้องการใช้โปรแกรม ตัวอย่างการปรับเปลี่ยนเทคโนโลยี เช่น การเปลี่ยนจากโปรแกรมที่ทำงานในสถาปัตยกรรมแบบสองเทียร์ (Two

Tiers Architecture) มาเป็นสถาปัตยกรรมแบบสามเทียร์ (Three Tiers Architecture) ผู้พัฒนาต้องปรับตัว ปรับแนวคิดและเทคโนโลยีทำงาน สร้างความรู้ความชำนาญขึ้นมาใหม่ และหากในอนาคตเทคโนโลยีมีการปรับเปลี่ยนไปอีก ก็จะเกิดปัญหาในการปรับตัว ระบบการทำงาน และเวลาในการปรับตัวอีกเช่นเดิม

ด้วยสาเหตุดังกล่าว ผู้วิจัยมีความต้องการจะลดระยะเวลาที่ใช้ในการพัฒนาโปรแกรม และรองรับการนำกลับมาใช้ใหม่ อีกทั้งยังเป็นแนวคิดที่ครอบคลุมไปถึงการพัฒนาเพื่อองค์กรขนาดใหญ่ จึงเลือกวิจัยวิธีการนำเทคโนโลยีเว็บเซอร์วิส (เว็บเซอร์วิส) มาใช้ในการพัฒนาระบบบัญชี เนื่องจากรูปแบบของเทคโนโลยีเป็นการเรียกใช้บริการผ่านทางโปรโตคอล http ซึ่งเป็นโปรโตคอล ที่ได้รับความนิยมสูงในปัจจุบัน อีกทั้งรูปแบบการส่งข้อมูลเป็นภาษา xml ซึ่งเป็นภาษาที่มีโครงสร้างมาตรฐาน สามารถใช้งานได้ร่วมกับทุก ๆ เทคโนโลยีทั้งในปัจจุบัน และคาดว่าในอนาคต ก็ยังคงใช้การต่อไปได้เนื่องจากความเป็นมาตรฐาน นอกจากนี้ ระบบบริการเว็บเซอร์วิส ยังมีประโยชน์ ในแง่ของการ นำกลับมากใช้ใหม่ (reusable) เนื่องจากระบบมีการติดตั้งไว้ที่ Server เมื่อมีโปรแกรมที่สร้างขึ้นมาจากเทคโนโลยีใดก็ตาม ต้องการใช้ข้อมูลจากการประมวล ผลระบบงานหนึ่ง ๆ ที่ให้บริการ ก็สามารถมาเรียกใช้บริการเว็บเซอร์วิสนี้ได้ โดยไม่มีปัญหาความไม่เข้ากันของเทคโนโลยีที่ใช้ในการพัฒนาระบบ

งานวิจัยนี้ ได้พัฒนาระบบการให้บริการทางเว็บสำหรับระบบบัญชีขึ้น ระบบงานที่ให้บริการคือ การคำนวณงบต่าง ๆ เช่น งบทดลอง งบดุล งบกำไร ขาดทุน เป็นการให้บริการด้วยเทคโนโลยีเว็บเซอร์วิส และได้พัฒนาโปรแกรมอีกส่วนหนึ่งเป็นระบบงานที่ให้ผู้ใช้โปรแกรมได้ทำการกรอกข้อมูลทางบัญชีลงไป หรือเรียกว่าสมุดรายวัน เป็นโปรแกรมตัวอย่างและใช้ในการทดสอบแนวคิดการนำเทคโนโลยีเว็บเซอร์วิสมาใช้

1.2 วัตถุประสงค์ของการวิจัย

1.2.1 เพื่อพัฒนาการให้บริการทางเว็บสำหรับระบบบัญชี

1.2.2 เพื่ออำนวยความสะดวกกับนักพัฒนาที่พัฒนาระบบบัญชีหลาย ๆ แพลตฟอร์ม

1.2.3 เพื่อให้สามารถนำกลับมาใช้ระบบใหม่ได้ โดยเสียเวลาในการพัฒนาน้อยที่สุด (Reusable)

1.2.4 เพื่อเป็นแนวทางการพัฒนาระบบไปสู่การนำแนวคิด SOA (Service Oriented Architecture) มาปรับใช้ในองค์กร

1.3 ข้อตกลงเบื้องต้น

1.3.1 ระบบที่พัฒนาขึ้นสามารถใช้งานได้ในระบบปฏิบัติการ Windows, Linux และ Mac OS X ได้

1.3.2 ระบบสามารถทำงานได้โดยใช้ Web Browser ในการทำงาน แต่ในส่วนที่เป็นเว็บเซอร์วิส จำเป็นต้องใช้บนระบบที่ใช้ Web Server เป็น IIS เท่านั้น

1.3.3 สิ่งแวดล้อมในการพัฒนาได้แก่ ระบบปฏิบัติการ Mac OS X Leopard 10.5.5 เครื่องคอมพิวเตอร์ที่ใช้ในการพัฒนาคือ Macbook Pro, CPU Intel 2.4 Ghz Core 2 Duo, Ram 2GB

1.3.4 เครื่องมือพัฒนาได้แก่ Netbeans 6.5, MAMP (Mac, Apache, MySQL, PHP) ภาษาที่ใช้พัฒนาคือภาษา PHP

1.4 ขอบเขตการวิจัย

1.4.1 สร้างระบบการให้บริการทางเว็บสำหรับระบบบัญชี

1.4.2 สร้างโปรแกรมขึ้นมาทดสอบเรียกใช้งานระบบการให้บริการทางเว็บสำหรับระบบบัญชี

1.4.3 ได้ทำการคำนวณงบทั้งหมด 3 งบได้แก่ งบทดลอง, งบดุล และงบกำไรขาดทุน

1.4.4 ระบบที่พัฒนาขึ้นจะครอบคลุมเฉพาะการเรียกใช้งานทดสอบระบบเท่านั้น มิได้สร้างขึ้นมา เพื่อการนำไปใช้งานจริง เป็นเพียงแนวคิดของงานวิจัย โดยสิ่งที่ระบบสามารถ ทำได้คือ สามารถบันทึกรายการทางบัญชีลงสมุดรายวันได้ สามารถคำนวณงบการเงินได้ 3 งบ ได้แก่ (1) งบทดลอง (2) งบกำไรขาดทุน (3) งบดุล โดยจะส่งการคำนวณผ่านเว็บเซอร์วิส

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1.5.1 ช่วยให้นักพัฒนาสามารถนำระบบเดิมมาพัฒนาได้ ไม่จำกัดแพลตฟอร์ม

1.5.2 ช่วยลดเวลาของนักพัฒนาในการเขียนโปรแกรม

1.5.3 ช่วยเป็นแนวการการพัฒนาต่อไปสู่ระบบงานใหญ่ ๆ ได้

1.5.4 เป็นตัวอย่างการนำเอาสถาปัตยกรรมเชิงบริการ (Service-Oriented Architecture : SOA) มาประยุกต์ใช้ในการพัฒนาแอปพลิเคชั่น และเป็นแนวทางการพัฒนาต่อไปในอนาคต

บทที่ 2
ปริทัศน์วรรณกรรมและงานวิจัยที่เกี่ยวข้อง

2.1 แนะนำ Service-Oriented Architecture (SOA)

ปัจจุบันภายในองค์กรต่าง ๆ อาจประกอบด้วยงานหลายชนิด เช่น CRM (ระบบบริหารความสัมพันธ์ลูกค้า : Customer Relationship Management), ERP (ระบบการบริหารทรัพยากร ขององค์กร : Enterprise Resource Planning) และ SCM (ระบบการจัดการกระบวนการผลิต : Supply Chain Management) เป็นต้น ดังนั้น การนำระบบงานต่าง ๆ มาเชื่อมต่อกันภายในและภายนอกองค์กร เพื่อแบ่งปันทรัพยากรและเทคโนโลยีระหว่างกัน จึงเป็นสิ่งจำเป็นเพราะจะช่วยเพิ่มความคล่องตัวในการทำงาน เพิ่มประสิทธิภาพในการติดต่อสื่อสาร และรองรับรูปแบบของธุรกิจที่มีการเปลี่ยนแปลงไปอย่างรวดเร็ว แต่การเชื่อมต่อของซอฟต์แวร์ขององค์กรกับซอฟต์แวร์ของคู่ค้าทางธุรกิจนั้นทำได้ยาก และต้องใช้ต้นทุนสูง จึงมีการนำ “สถาปัตยกรรมเชิงบริการ(Service-Oriented Architecture : SOA)” มาปรับใช้เพื่อแก้ไขปัญหาดังกล่าว ด้วยการสร้างให้ซอฟต์แวร์ของระบบมีลักษณะเชิงบริการ โดย “หน่วยของลอจิก (Unit of Logic)” ที่อยู่ภายในคอมโพเน้นท์ของซอฟต์แวร์จะมีลักษณะเป็น “Service” ซึ่งจัดเตรียมบริการต่าง ๆ ไว้ให้คอมโพเน้นท์หรือ Service อื่นเรียกใช้ โดยที่แต่ละ Service จะมีความเป็นอิสระต่อกัน และสามารถติดต่อสื่อสารกันได้ทั้ง Service ที่อยู่ภายในและภายนอกองค์กร ลักษณะดังกล่าวเป็นการแก้ไขปัญหาเรื่องความแตกต่างด้านเทคโนโลยี ทำให้สามารถเชื่อมต่อระบบกับองค์กรคู่ค้าได้ผ่านเทคโนโลยีที่มีอยู่ โดยไม่จำเป็นต้องพัฒนาระบบใหม่ทั้งหมด แต่การนำ SOA มาใช้งานต้องคำนึงถึงปัจจัยต่าง ๆ ทั้งทางด้านความพร้อมขององค์กรและทางด้านเทคนิค มิฉะนั้นอาจทำให้การนำซอฟต์แวร์เชิงบริการมาใช้งานประสบความล้มเหลวได้

แนวคิดเชิงบริการ (Service-Orientation) คือแบบแผนในการออกแบบและพัฒนาซอฟต์แวร์ที่อยู่บนพื้นฐานของ Service ซึ่งเป็นแนวคิดที่ตอบสนองความต้องการทางธุรกิจในอนาคต โดยทำงานอยู่บนมาตรฐานเปิดที่ได้รับการยอมรับแนวคิดดังกล่าว ถูกนำไปประยุกต์ใช้เพื่อสนับสนุนการออกแบบคุณลักษณะ (Characteristic) ต่าง ๆ ของ Service ซึ่งเป็นปัจจัยสำคัญที่ใช้ในการสนับสนุนสถาปัตยกรรมเชิงบริการ Service-Oriented Architecture (SOA) โดยแนวคิดเชิงบริการจะมององค์ประกอบต่าง ๆ ของซอฟต์แวร์เป็น Service คล้ายกับแนวคิดเชิงวัตถุ (Object-Orientation) ที่มององค์ประกอบต่าง ๆ เป็น Object สถาบัน OASIS (Organization

for the Advancement of Structured Information Standards) ได้ให้ความหมายของ SOA ไว้ดังนี้

“สถาปัตยกรรมเชิงบริการ (Service-Oriented Architecture : SOA) คือแบบจำลอง ของการจัดระเบียบและการใช้ประโยชน์จากความสามารถที่ถูกแจกจ่ายออกมาโดยผู้ที่เป็นเจ้าของความสามารถดังกล่าวที่แตกต่างกันจำนวนมาก ความสามารถที่ถูกแจกจ่ายจะอยู่ในรูปแบบ ที่พร้อมนำเสนอออกสู่ภายนอก เพื่อให้ผู้ที่ต้องการใช้ความสามารถค้นหา ทำงานร่วมกัน และ เรียกใช้ความสามารถเหล่านั้น เพื่อนำไปสู่ผลลัพธ์ที่ต้องการได้”

Thomas Erl เป็นนักวิชาการทางด้านคอมพิวเตอร์ และระบบการจัดการบริการ ได้ให้คำจำกัดความของ SOA (Service-Oriented Architecture Concept Technology and Design , Thomas Erl, 2006) ไว้ดังนี้

“สถาปัตยกรรมเชิงบริการ (Service-Oriented Architecture : SOA) คือ แบบจำลองที่แสดงให้เห็นถึงการแบ่งส่วนลอจิกการทำงานที่สามารถทำงานเองอัตโนมัติออกเป็นหน่วยย่อย (Unit of Logic) แต่ละหน่วยของลอจิก ประกอบไปด้วย ลอจิกการทำงานทาง ธุรกิจแบบอัตโนมัติ ที่สามารถกระจายไปใช้งานข้ามระบบได้”

จากแนวคิดทั้ง 2 แนวคิด และจากคำจำกัดความของ SOA จากแหล่งต่าง ๆ ที่ไม่ได้กล่าวถึง สามารถนำมาสรุปได้คือ

สถาปัตยกรรมเชิงบริการ (Service-Oriented Architecture : SOA) คือ สถาปัตยกรรม ของซอฟต์แวร์ (Software Architecture) ซึ่งอยู่บนพื้นฐานของแนวคิดเชิงบริการ (Service-Orientation) โดยจะนิยามถึงวิธีการใช้ Service ที่มีความเป็นอิสระต่อกัน (Loosely Coupled) หรือการเชื่อมโยง Service แบบหลวม ๆ ทำให้การปรับเปลี่ยน หรือประกอบ Service ทำได้อย่างอิสระ และสามารถทำงานร่วมกันข้ามระบบได้ นอกจากนี้ยังช่วยสนับสนุนการนำ Service ที่พัฒนาไว้แล้วกลับมาใช้ใหม่ (Reuse) ทำให้สามารถรองรับความต้องการของผู้ใช้ซอฟต์แวร์ และกระบวนการทางธุรกิจ (Business Process) ซึ่งเกิดจากการทำงานร่วมกันของ Service ตามลำดับขั้นตอนที่กำหนดไว้ได้

SOA วิวัฒนาการมาจากเทคโนโลยีการประมวลผลแบบกระจาย (Distributed Computing) จุดเด่นของ SOA คือ ไม่ยึดติดกับภาษาที่ใช้ในการพัฒนาโปรแกรม มีความยืดหยุ่นสูง และสามารถทำงานในระบบปฏิบัติการที่ต่างกันได้ แต่ SOA ไม่ได้ถูกออกแบบมาเพื่อเทคโนโลยีใดเทคโนโลยีหนึ่ง และไม่ยึดติดกับมาตรฐานใด ๆ กล่าวคือ SOA เป็นเพียงหลักการวิเคราะห์ และหลักการออกแบบซอฟต์แวร์ที่สามารถนำมา Implement ร่วมกับเทคโนโลยีต่าง ๆ ได้ เช่น DCOM, CORBA และ RPC เป็นต้น แต่การนำเอาเทคโนโลยีดังกล่าวมา Implement ต้องเสียค่าใช้จ่ายค่อนข้างสูง และใช้งานได้เฉพาะบาง Platform ทำให้ในปัจจุบันนิยมเอาเทคโนโลยีเว็บเซอร์วิส (Web Service) มาใช้

กับ SOA เนื่องจากเป็นเทคโนโลยีที่มีประสิทธิภาพ มีความยืดหยุ่นสูงไม่ขึ้นกับ Platform ใด ๆ และสนองตอบความต้องการทางธุรกิจได้เป็นอย่างดี

2.1.1 จุดเด่นของ SOA

การพัฒนา SOA จะอยู่บนมาตรฐานแบบเปิด ซึ่งช่วยให้ธุรกิจสามารถ เชื่อมโยงข้อมูลต่าง ๆ ระหว่างลูกค้า หุ้นส่วน และซับพลายเออร์เข้ากันได้อย่างเป็นระบบ โดย Service ทำหน้าที่เชื่อมโยงข้อมูลด้วยสถาปัตยกรรม SOA ไม่ว่าข้อมูลจะอยู่ใน Platform หรือแอปพลิเคชั่นใด ด้วยการแบ่งโครงสร้างพื้นฐานทั้งระบบออกเป็นส่วน ๆ ซึ่งสามารถ นำมาใช้งานร่วมกัน หรือผสมกันเพื่อดำเนินการตามวัตถุประสงค์ของธุรกิจได้ โดย SOA เปรียบเสมือนแนวคิดที่ทำให้เห็น “โครงร่างการทำงาน (Framework)” ที่ใช้ในการพัฒนาระบบ

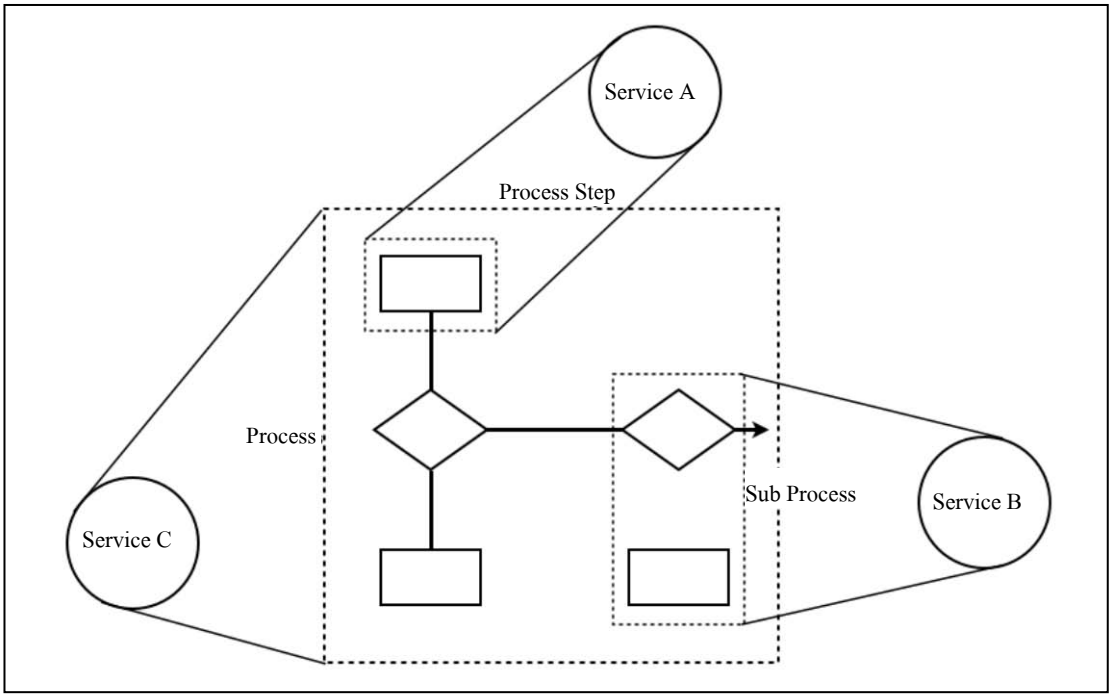
จุดเด่นของ SOA คือความสามารถในการทำงานร่วมกัน (Interoperability) โดยไม่สนใจระบบ โครงสร้าง และเทคโนโลยีของผู้ที่ต้องการติดต่อ เช่น สร้างฟังก์ชั่น “Check Inventory” สำหรับให้บริการแก่แอปพลิเคชั่นหนึ่งเพื่อควบคุมรายการสินค้า ในขณะเดียวกัน ก็สามารถนำฟังก์ชั่น “Check Inventory” ไปใช้สนับสนุนเครื่องมือควบคุมสินค้าแบบ Web Base ได้โดยไม่ต้องแก้ไข การนำฟังก์ชั่นดังกล่าวมาใช้เป็นข้อได้เปรียบ เนื่องจากช่วยลดต้นทุน การพัฒนางาน นอกจากนี้จุดมุ่งหมายระยะยาวของการนำฟังก์ชั่นมาใช้ซ้ำ คือ ลดฟังก์ชั่นที่เกิน ความจำเป็น ลดความซ้ำซ้อนของโครงสร้างพื้นฐาน และลดค่าใช้จ่ายในการบำรุงรักษา ทำให้มีข้อเชื่อมโยงระบบที่รวดเร็วและยืดหยุ่นยิ่งขึ้น สามารถตรวจเช็คระบบได้อย่างรวดเร็วเมื่อเปรียบเทียบกับการใช้งานระบบเดิม (การพัฒนาระบบ และสถาปัตยกรรมด้วย Web Sevice : สุธี พงศาสกุลชัย)

2.1.2 ความหมายของ Service

เซอร์วิส (Service) หรือบริการ คือฟังก์ชั่นการทำงานที่ถูกสร้างขึ้นใน ลักษณะของการให้บริการ ซึ่งอยู่ในรูปแบบของ Interface ที่กำหนดข้อตกลงในการเรียกใช้บริการไว้ โดยสามารถห่อหุ้ม (Encapsulate) ลอจิก (Logic) ที่อยู่ภายในโซลูชั่น (Solution) ไว้ และมีขอบเขตการทำงานชัดเจน Service จะทำงานตามขอบเขตดังกล่าวได้ด้วยตนเอง เนื่องจากฟังก์ชั่นในการทำงานต่าง ๆ จะถูกบรรจุไว้ในตัว Service

2.1.3 การห่อหุ้มลอจิกของเซอร์วิส (Service)

ลอจิกของ Service หนึ่งสามารถนำไปเป็นส่วนประกอบของ Service อื่น ๆ ได้ ตัวอย่างเช่น Solution หนึ่งที่ประกอบไปด้วยลอจิกต่าง ๆ ที่เป็นคำสั่งควบคุมการทำงานดังรูป

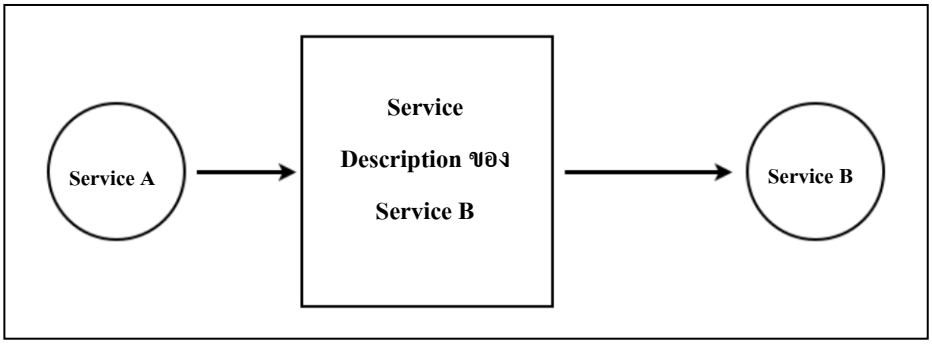


รูปที่ 2.1 แสดงลอจิกที่อยู่ใน Service

รูปข้างต้นแสดง Solution ที่ประกอบไปด้วยกลุ่มของ Service ต่าง ๆ แต่ละ Service สามารถห่อหุ้ม (Encapsulate) ลอจิกต่าง ๆ ไว้ภายใน โดย Service A จะห่อหุ้มขั้นตอนของกระบวนการ (Process Step) ไว้ขั้นตอนเดียว แต่ Service B จะห่อหุ้มกระบวนการย่อย (Sub Process) ซึ่งประกอบไปด้วยขั้นตอนของกระบวนการต่าง ๆ ไว้ภายใน ส่วน Service C จะห่อหุ้มลอจิกทั้งหมดไว้ภายใน โดยทุก Service ที่ห่อหุ้มลอจิกไว้ภายในสามารถดำเนินกิจกรรมร่วมกันได้

2.1.4 ความสัมพันธ์ของ Service

การที่ Service หนึ่งจะเรียกใช้บริการจาก Service อื่นได้นั้น จะต้องทราบรายละเอียดของ Service ที่ต้องการติดต่อสื่อสารก่อน เช่น ลักษณะของบริการ ตำแหน่ง หรือวิธีเรียกใช้ Service เป็นต้น โดยรายละเอียดดังกล่าวจะถูกเก็บไว้ในเอกสารที่เรียกว่า “Service Description”



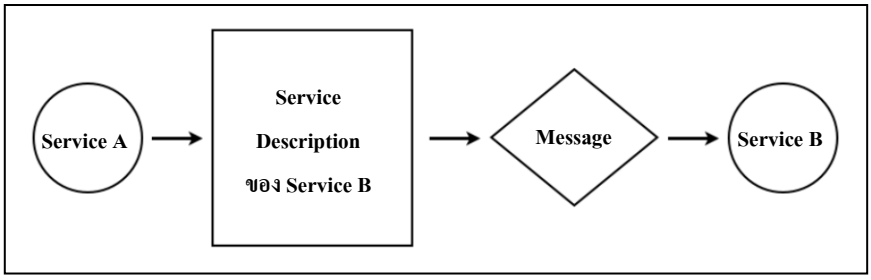
รูปที่ 2.2 แสดงภาพ Service A เข้าถึง Service Description ของ Service B

Service Description ใช้กำหนดชื่อและตำแหน่งของ Service ทำหน้าที่เหมือนกับตัวกลาง สำหรับการร้องขอเพื่อแลกเปลี่ยนข้อมูล ทำให้เกิดความเป็นอิสระต่อกัน (Loosely Couple) ระหว่าง Service ดังรูปข้างต้น (แสดงให้เห็นว่า Service A จะทราบข้อมูลทั้งหมดที่จำเป็น ในการติดต่อสื่อสารกับ Service B ผ่าน Service Description ของ Service B นั่นเอง)

การติดต่อสื่อสารกันของ Service เพื่อให้บรรลุเป้าหมายบางอย่างจะต้องมีการแลกเปลี่ยนข้อมูลข่าวสารระหว่างกัน โดยโครงร่างของการติดต่อสื่อสาร (Communication Framework) ที่ใช้จะต้องรักษาความเป็นอิสระต่อกันของ Service ไว้

2.1.5 การติดต่อสื่อสารของ Service

เมื่อ Service ส่ง Message ไปยังปลายทางแล้ว Service นั้นจะไม่สามารถควบคุม หรือทราบถึงสิ่งที่เกิดขึ้นกับ Message ได้ ดังนั้น Message จึงต้องมีความสามารถในการควบคุมตนเอง กล่าวคือ Message มีลักษณะเป็นตัวกลางอิสระ สำหรับใช้ในการติดต่อสื่อสารโดยมีการทำงานแบบอัตโนมัติ

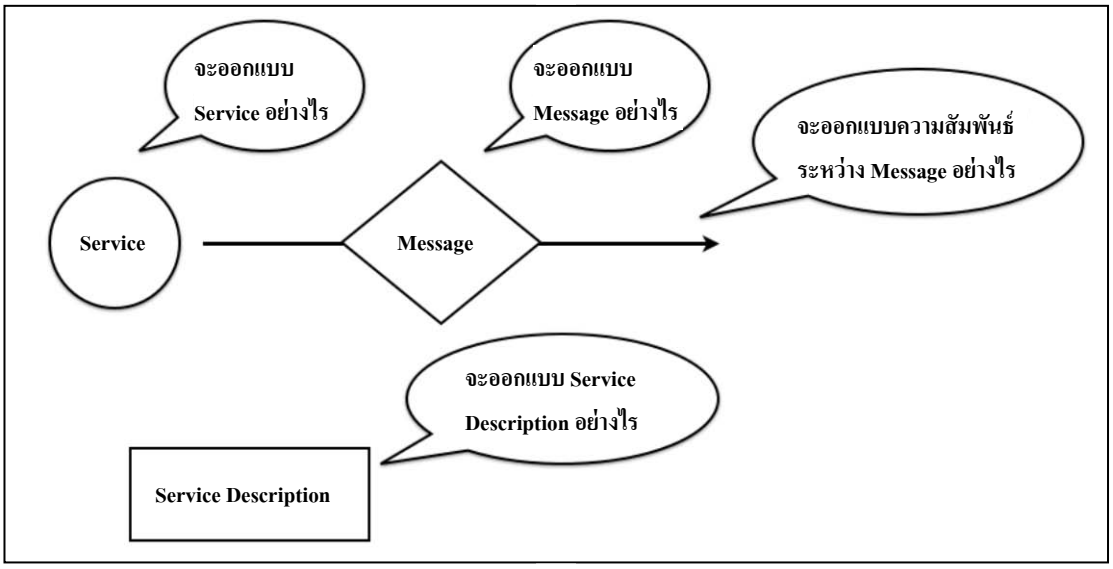


รูปที่ 2.3 แสดงภาพของ Message ซึ่งเหมือนเครื่องมือที่ทำให้การติดต่อสื่อสารเป็นอิสระต่อกัน

Service จัดเตรียม Service Description และการติดต่อสื่อสารผ่าน Message ด้วยสถาปัตยกรรมที่เรียบง่ายไว้ ซึ่งมีลักษณะคล้ายกับ Distribute Architecture ที่สนับสนุน Messaging ด้วยการแยกส่วนของการเชื่อมต่อ (Interface) ออกจากส่วนลอจิกของการประมวลผล (Processing Logic) โดยแนวคิดของ Service-Orientation ได้แบ่งส่วนประกอบต่าง ๆ ออกเป็น 3 ส่วนหลัก คือ Service (ส่วนที่คอยให้บริการ) Description (ส่วนอธิบายรายละเอียดของบริการ) และ Message (ส่วนที่ใช้สำหรับติดต่อสื่อสารระหว่างกัน)

2.1.6 การออกแบบ Service

การสร้างบริการโดยอาศัยหลักการของ Service-Orientation จำเป็นต้องออกแบบส่วนประกอบต่าง ๆ ของ Service ให้ทำงานอยู่บน “หลักการเบื้องต้นของ SOA” โดยสามารถอธิบายได้ดังรูป



รูปที่ 2.4 แสดงภาพส่วนประกอบพื้นฐานของ Service

2.1.7 หลักการเบื้องต้นของ SOA

ในหัวข้อนี้จะกล่าวถึงหลักเบื้องต้นของ Service-Orientation ที่อธิบายถึงคุณสมบัติและกฎเกณฑ์พื้นฐานของแอปพลิเคชั่นที่นำ SOA ไปประยุกต์ใช้งานเพื่อสร้าง Service (Service Oriented Architecture for Dummies) หลักการเหล่านี้จะมีผลกระทบต่อพฤติกรรมการทำงาน และการออกแบบระบบ โดยหลักการสำคัญของ Service-Orientation ที่จะพิจารณามีดังนี้

- Loose Coupling	Service จะรักษาความสัมพันธ์ระหว่างกันไว้แต่พยายามให้มีความอิสระต่อกันมากที่สุด โดยทราบเพียงบริการที่ต้องการใช้งานเท่านั้น และต้องไม่ขึ้นกับ Service อื่น
- Service Contract	Service ต้องมีข้อตกลงสำหรับการติดต่อสื่อสาร โดยเก็บรวบรวมไว้ใน Service Description
- Autonomy	Service จะควบคุมลอจิกต่าง ๆ ที่อยู่ภายในได้ด้วยตัวเอง
- Abstraction	Service จะต้องแยกลอจิก (หรือส่วนในการทำงาน) ต่าง ๆ ออกจากข้อตกลงที่กำหนดไว้ใน Service Contract เพื่อซ่อนลอจิกจากแอปพลิเคชั่น หรือ Service อื่นที่อยู่ภายนอก
- Reusability	ลอจิกต่าง ๆ ที่อยู่ภายใน Service ต้องสนับสนุนการนำกลับมาใช้ใหม่ (Reuse)
- Composability	กลุ่มของ Service สามารถทำงานร่วมกันได้ โดยนำมาประกอบกันเพื่อสร้างเป็น Service ใหม่
- Statelessness	ควรลดการเก็บข้อมูลของกิจกรรม หรือสถานะต่าง ๆ ไว้ให้น้อยที่สุด เพื่อลดการเชื่อมโดยกันระหว่าง Service
- Discoverability	Service จะถูกประกาศสู่ภายนอก และสามารถค้นหาได้ผ่านกลไกการค้นหา (Discovery Mechanism)

การนำหลักการเบื้องต้นของ SOA มาใช้ในการพัฒนา Web Service แบ่งออกเป็น 2 ส่วน คือหลักการที่ใช้ในการออกแบบ Service และหลักการที่เกี่ยวข้องกับเทคโนโลยี โดยหลักการที่นำไปใช้ในการออกแบบได้แก่ (1) Loose Coupling (2) Autonomy (3) Abstraction และ (4) Statelessness ส่วนหลักการที่เกี่ยวข้องกับเทคโนโลยี ได้แก่ Service Contract คือ WSDL ที่ใช้บอกรายละเอียด และวิธีการติดต่อกับ Service Composability เกี่ยวข้องกับภาษาที่สร้างองค์ประกอบร่วม

ของ Web Service Discoverability คือ UDDI ที่ใช้ลงทะเบียน Service สถาปัตยกรรมพื้นฐาน และหลักการต่าง ๆ ที่กล่าวมาเป็นส่วนประกอบ ในการนำไปใช้ เพื่อจัดการ และกำหนดมาตรฐานของการให้บริการ แต่เนื้อหาที่กล่าวมาทั้งหมดยังไม่ได้ รวมส่วนของการ Implement ด้วยเทคโนโลยีต่าง ๆ ซึ่งสามารถทำได้หลาย Platform เช่น J2EE หรือ .NET เป็นต้น

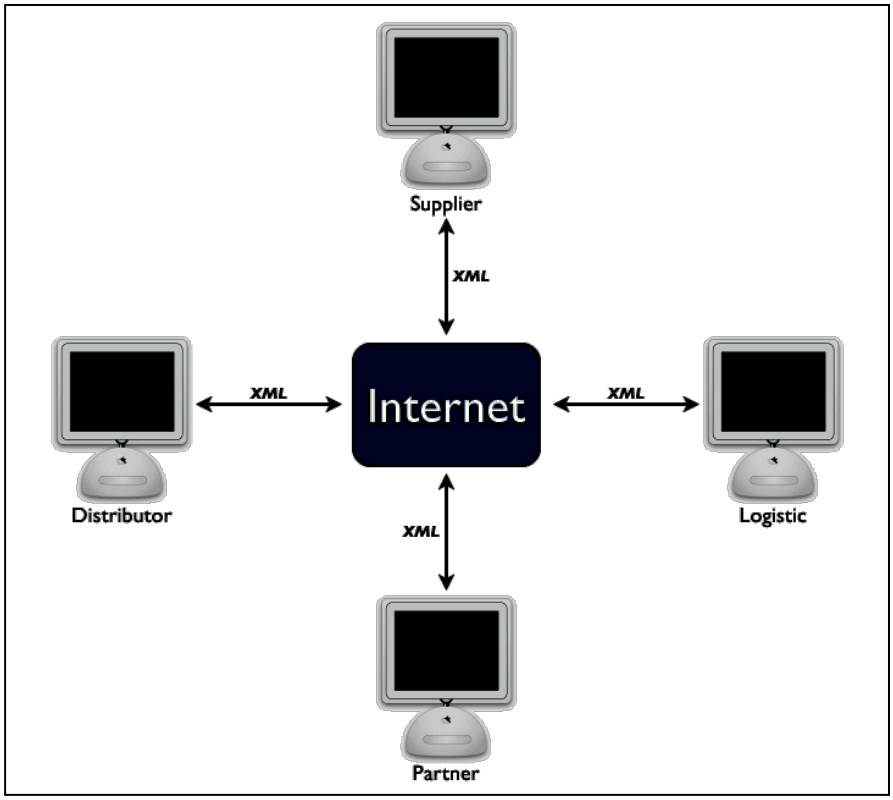
2.2 ประวัติของ SOA

2.2.1 ความเป็นมาของ XML (www.thaixml.com)

E-Commerc เป็นช่องทางในการดำเนินธุรกิจโดยใช้อินเตอร์เน็ต ซึ่งมีวิวัฒนาการ มาทั้งหมด 3 ยุค คือ ยุคแรกเป็นการเสนอข้อมูลในรูปแบบที่ไม่เปลี่ยนแปลง (Static) โดยใช้ HTML และไคลเอนต์ติดต่อกับเซิร์ฟเวอร์ ผ่านทางโปรโตคอล HTTP เพื่อร้องขอข้อมูลมาแสดงบน Browser ส่วนยุคที่ 2 มีลักษณะคล้ายกับยุคแรก แต่มีการเชื่อมโยงฐานข้อมูล และการประมวลผลบางอย่างบนเซิร์ฟเวอร์ เช่น ลูกค้าสามารถตรวจสอบ และสั่งสินค้าที่ต้องการจากเว็บไซต์ ของบริษัทได้โดยตรง ลักษณะดังกล่าวทำให้องค์กรธุรกิจสามารถจัดการกับข้อมูลภายในของตัวเองได้โดยตรง แล้วเชื่อมต่อกับระบบอินเตอร์เน็ตเพื่อให้บริการกับลูกค้า ทำให้ต้นทุนต่าง ๆ ลดลง และอำนวยความสะดวกให้กับลูกค้า โดยเป็นการดำเนินธุรกิจแบบ B2C ซึ่งธุรกิจส่วนใหญ่ในอินเตอร์เน็ตจะอยู่ในรูปแบบดังกล่าว

ในปัจจุบันการดำเนินธุรกิจแบบ B2C ซึ่งนำเสนอข้อมูลแก่ลูกค้าโดยตรง เริ่มไม่เพียงพอต่อความต้องการขององค์กรธุรกิจต่าง ๆ ดังนั้นการทำธุรกิจ E-Commerce จึงเกิดการเปลี่ยนแปลงเข้าสู่ยุคที่ 3 โดยองค์กรธุรกิจไม่เพียงเสนอข้อมูลต่าง ๆ แก่ลูกค้าโดยตรงเท่านั้น แต่ยังสามารถให้ลูกค้าเรียกใช้บริการจากองค์กรธุรกิจอื่น ๆ ผ่านทางแอปพลิเคชั่นของตนได้อีกด้วย ซึ่งถือว่าเป็นการดำเนินธุรกิจแบบ B2B เช่น เมื่อแอปพลิเคชั่นของผู้ขายได้รับข้อมูลการปรับปรุงสินค้าที่ส่งมาจากตัวแทนจำหน่าย ข้อมูลดังกล่าวจะส่งให้กับลูกค้าด้วย เป็นต้น วิธีดังกล่าวเป็นการนำเอาอินเตอร์เน็ต มาใช้ในเชิงพาณิชย์ ทั้งภายในองค์กร และระหว่างธุรกิจ โดยให้แอปพลิเคชั่นติดต่อกันเอง หรือเรียกว่า “Program to Program (P2P)”

Extensible Markup Language หรือ XML เป็นเทคโนโลยีที่สำคัญนำมาใช้กับ SOA เพื่อสร้างบริการในรูปแบบของ Web Service โดยภาษา XML เป็นภาษา Markup ที่มีแท็ก (Tag) คล้ายกับภาษา HTML ซึ่งมีจุดมุ่งหมายเพื่อสร้างเว็บเพจ และแสดงผลผ่าน Browser ภาษา XML มีจุดประสงค์เพื่อการสื่อความหมายของข้อมูลบนเว็บไซต์ โดยสามารถกำหนด Tag ขึ้นใช้งานเองได้



รูปที่ 2.5 แสดงภาพการแลกเปลี่ยนข้อมูลด้วย XML

รูปข้างต้นแสดงตัวอย่างการใช้ XML เพื่อแลกเปลี่ยนข้อมูลระหว่างหุ้นส่วน องค์กรธุรกิจในรูปแบบ B2B โดยแต่ละองค์กรตั้งอยู่ในสถานที่ต่างกัน รวมทั้งใช้ระบบ และแอปพลิเคชั่นทางธุรกิจที่ต่างกัน แต่ทุกองค์กรมีพื้นฐานการแลกเปลี่ยนข้อมูลด้วย XML เหมือนกัน

XML มีลักษณะคล้ายกับภาษา HTML ได้รับการพัฒนาโดย W3C โดยมีพื้นฐานมาจาก Standard Generalized Markup Language (SGML) ซึ่งถูกคิดค้นในปี ค.ศ. 1960 เพื่อนำมาใช้แทนข้อมูล แต่ SGML เป็นภาษาที่เรียนรู้ยากและเสียเวลามาก เนื่องจาก SGML ได้รวบรวมมาตรฐานต่าง ๆ ไว้มากมาย ด้วยเหตุผลนี้แม้ว่าภาษา SGML จะมีประสิทธิภาพเพียงใดก็ตาม แต่ก็ใช้งานลำบาก ทำให้ไม่ได้รับความนิยมเท่าที่ควร

XML ถูกนำมาใช้กับธุรกิจ E-Business เพื่อแทนข้อมูลอย่างแพร่หลาย และใช้ Internet Protocol เพื่อรับส่งข้อมูล การนำ XML มาใช้ทำให้แอปพลิเคชั่นสามารถสื่อสารกันได้อย่างอิสระมากขึ้น ผู้พัฒนาสามารถแนบความหมาย หรือคำอธิบายต่าง ๆ ของข้อมูลด้วย XML และส่งข้ามระบบเครือข่ายได้โดยใช้ Internet Protocol แต่ XML ไม่ได้ถูกนำมาใช้ เพื่อแทนข้อมูลเพียงอย่างเดียว โดย XML ยังถูกใช้เพื่อแทนกลุ่มของข้อกำหนดต่าง ๆ มากมายเช่น XML Schema

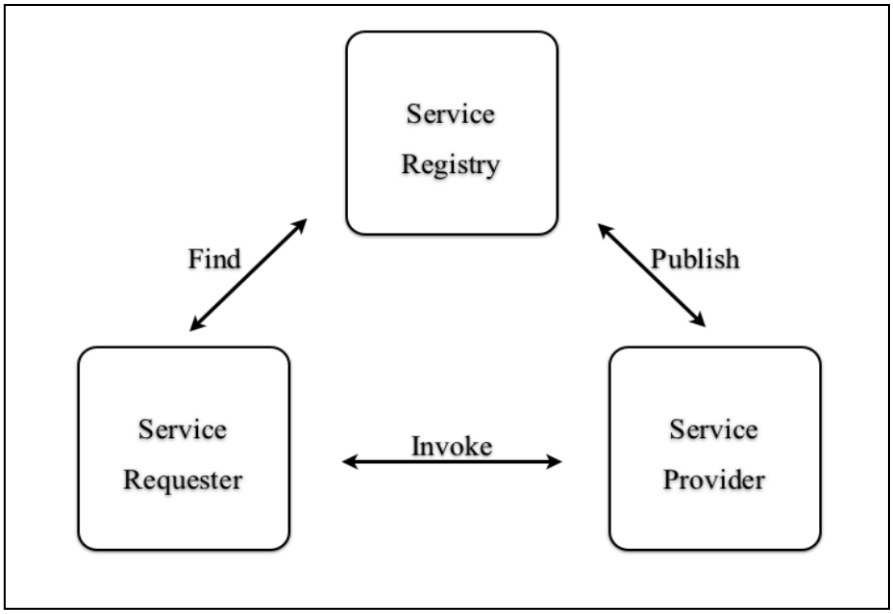
Defination (XSD) หรือ XSL Transformation (XSLT) ซึ่งเป็นภาษาที่พัฒนามาจาก XML ข้อกำหนดข้างต้นเป็นส่วนประกอบสำคัญของกลุ่มเทคโนโลยี XML

XML เป็นโครงสร้างสำคัญที่นำมาใช้กับ SOA เพื่อพัฒนา Web Service โดย XML จะกำหนดรูปแบบ และโครงสร้างของ Message ที่ Service ใช้ติดต่อสื่อสารกัน ตัวอย่างเช่น XML Schema ใช้สำหรับคงความถูกต้อง และความสมบูรณ์ของข้อมูล ส่วน XSLT ทำให้ข้อมูล ที่แตกต่างกันติดต่อสื่อสารกันได้ด้วยวิธี Schema Mapping

2.2.2 รูปแบบการทำงานของ SOA

SOA ถูกแบ่งเป็นประเภทต่าง ๆ ตามเทคโนโลยีที่นำมา Implement เพื่อสร้าง Service โดยแบบจำลองของ SOA (SOA Model) ในช่วงแรก ๆ เป็นจุดเริ่มต้นมาตรฐานที่สำคัญของ Web Service จุดเริ่มต้นของ SOA เกิดจาก 3 ส่วนประกอบสำคัญคือ ผู้ให้บริการ (Service Provider) ผู้ร้องขอบริการ (Service Requester) และหน่วยสืบค้นบริการ (Service Registry) โดยมีรายละเอียดดังนี้

- ผู้ให้บริการ (Service Provider) คือแอปพลิเคชั่นที่จัดเตรียมบริการไว้ให้แอปพลิเคชั่นอื่นเรียกใช้
- ผู้ร้องขอบริการ (Service Requester) คือแอปพลิเคชั่นที่ขอบริการจากแอปพลิเคชั่นที่ประกาศ (Publish)
- หน่วยสืบค้นบริการ (Service Registry) คือฐานข้อมูลกลางซึ่งรวบรวมบริการต่าง ๆ ที่ผู้ให้บริการประกาศไว้
- ให้พิจารณา SOA Model ดังภาพ



รูปที่ 2.6 แสดงภาพของ SOA Model

SOA มีรูปแบบการทำงานคือ ผู้ให้บริการ (Service Provider) จะประกาศ (Publish) บริการ (Service) ของตนเองไปยังหน่วยสืบค้นบริการ (Service Registry) เพื่อให้ผู้ร้องขอบริการ (Service Requester) ค้นหาบริการผ่าน Service Registry เมื่อพบบริการที่ต้องการแล้ว จึงเรียกใช้ (Invoke) บริการดังกล่าวผ่าน Service Provider ต่อไป เมื่อนำแบบจำลองดังกล่าวมาพิจารณา สามารถกำหนดมาตรฐานของ Web Service ได้ดังนี้

- WSDL (Web Service Definition Language) คือ คำอธิบายรายละเอียดของบริการ
- SOAP (Simple Object Access Protocol) คือ ผู้จัดเตรียมรูปแบบของ Message ที่ใช้ระหว่างผู้ให้บริการ กับผู้ร้องขอบริการ
- UDDI (Universal Description, Discovery and Integration) คือ ผู้กำหนดรูปแบบมาตรฐาน สำหรับลงทะเบียน Service

แบบจำลองของ Primitive SOA ยังมีการใช้งานจนถึงปัจจุบัน โดยผู้ผลิต Platform แต่ละชนิดต่างก็ได้วางแผนเพื่อรองรับการเปลี่ยนแปลงไปสู่ SOA การนำคุณลักษณะหลายอย่างของ Contemporary SOA มาผสมผสานกันทำให้เกิดข้อกำหนดเพิ่มเติมของ Web Service ในยุคแรก โดยเรียกว่า Web Service ยุคที่สอง (Second-Generation Web Service) หรือ “WS-*” ข้อกำหนดเพิ่มเติมนี้ได้ระบุถึงขอบเขตหน้าที่กับเป้าหมายทั้งหมดของการยกระดับเทคโนโลยี Web Service ไปสู่ระดับองค์กร (Enterprise)

2.2.3 ความเป็นมาของ Web Service

W3C ได้นำเสนอข้อกำหนดของ Simple Object Access Protocol (SOAP) ขึ้นในปี 2000 แรกเริ่มข้อกำหนดนี้ถูกออกแบบมาเพื่อการติดต่อสื่อสารแบบ RPC (Remote Procedure Call) เป็นไปในทิศทางเดียวกัน แนวความคิดนี้ใช้เพื่อส่งผ่านพารามิเตอร์ของข้อมูลระหว่าง Component ที่ถูกจัดเก็บในรูปแบบ XML แล้วส่งออกไปจากนั้นจึงแปลงกับไปเป็นรูปแบบเดิม

ต่อมาบริษัท และผู้ผลิตซอฟต์แวร์ต่าง ๆ เริ่มเห็นประโยชน์มากมายของระบบ ธุรกิจที่สร้างอยู่บนโครงสร้างการติดต่อสื่อสารที่เป็นอิสระผ่านอินเตอร์เน็ต ทำให้เกิดเทคโนโลยี Distribute ขึ้นมา จนกลายเป็นโครงสร้างมาตรฐาน สำหรับการติดต่อสื่อสารที่ใช้เพื่อแก้ปัญหาที่เกิดจากความแตกต่างของการสื่อสารข้อมูลทั้งภายใน และระหว่างองค์กร โดยเรียกแนวคิดนี้ว่า Web Service

การประกาศวิธีติดต่อสื่อสารกับ Service เป็นกลไกการทำงานที่สำคัญของ การพัฒนา SOA ด้วยเทคโนโลยี Web Service โดยจะใช้ Web Service Description Language (WSDL) ซึ่งอยู่ในรูปแบบของ XML เป็นแหล่งเก็บข้อมูลที่แสดงรายละเอียด และการเรียกใช้งาน Service ไว้ โดย W3C ได้นำเสนอภาษา WSDL เป็นครั้งแรกในปี ค.ศ. 2001 และ ได้ปรับปรุงข้อกำหนดต่าง ๆ มาอย่างต่อเนื่องจนถึงปัจจุบัน ส่วนการติดต่อสื่อสารระหว่าง Service จะอยู่ในรูปแบบของ Messaging โดยใช้เทคโนโลยี SOAP ซึ่งเป็นโปรโตคอลสื่อสารสำหรับรับส่ง Message ระหว่าง Service

UDDI คือ เทคโนโลยีที่ทำให้แนวคิดของ Web Service ในยุคแรก สามารถทำงานได้อย่างสมบูรณ์ โดยได้รับการพัฒนาจากความร่วมมือกันของ UDDI.org และ OASIS ให้เป็นข้อกำหนดมาตรฐานของการลงทะเบียน Service Description (WSDL) ทำให้ผู้จัดเตรียมบริการ (Service Provider) สามารถลงทะเบียน Service ไว้ในสถานที่ ซึ่งผู้ร้องขอบริการ (Service Requester) สามารถค้นหาบริการได้ (การพัฒนาระบบและ สถาปัตยกรรมด้วย Web Sevice : สุธี พงศาสกุลชัย,2550)

รูปที่ 2.7 แสดงความสัมพันธ์ระหว่าง UDDI WSDL SOAP และ Web Service

Web Service ที่สร้างขึ้นสามารถเรียกใช้ Web Service อื่นได้ กล่าวคือ แต่ละ Web Service สามารถเป็นได้ทั้ง Client และ Server ลักษณะดังกล่าวคล้ายระบบ Peer-to-Peer มากกว่าระบบ Client-Server ดังรูป

แอปพลิเคชั่นในรูปแบบต่าง ๆ ขึ้นมาใหม่ ทำให้แอปพลิเคชั่นมีการเปลี่ยนแปลงไปอย่างรวดเร็ว องค์กรทางด้าน IT เริ่มเห็นความสำคัญ และต้องการแม่แบบของแอปพลิเคชั่น เพื่อใช้เป็นบรรทัดฐานให้กับแอปพลิเคชั่นอื่นได้ โดยแม่แบบจะต้องมีข้อกำหนดเกี่ยวกับเทคโนโลยี ขอบเขต กฎเกณฑ์ ข้อจำกัด และการออกแบบคุณลักษณะต่าง ๆ เพื่อนำไปประยุกต์ใช้ในการแก้ปัญหาที่เกิดขึ้น บนพื้นฐานของแม่แบบ ทำให้เกิดมาตรฐานสถาปัตยกรรมซอฟต์แวร์ในระดับต่าง ๆ ซึ่งใช้อธิบายถึงโครงสร้าง ข้อบังคับ และวิธีการเชื่อมต่อของซอฟต์แวร์

2.3.1 สถาปัตยกรรมระดับแอปพลิเคชั่น (Application Architecture)

สถาปัตยกรรมระดับแอปพลิเคชั่น (Application Architecture) คือ รายละเอียด ของการพัฒนาแอปพลิเคชั่นที่นำไปสู่การพัฒนาแอปพลิเคชั่นอย่างเป็นระบบ ด้วยการแบ่งทีมพัฒนาและอธิบายรายละเอียดของแผนงานไว้ในพิมพ์เขียว (Blueprint) โดยจัดเตรียมรายละเอียด ทางเทคนิคทางด้านต่าง ๆ ของแต่ละแอปพลิเคชั่นไว้ เช่น แผนภาพในการแสดงการติดต่อสื่อสาร รูปแบบของข้อมูล หรือรายละเอียดด้านความปลอดภัย เป็นต้น

โดยทั่วไปองค์กรต่าง ๆ จะมีแอปพลิเคชั่นทำงานร่วมกันมากกว่า 1 แอปพลิเคชั่น และอาจทำงานบน Platform ที่ต่างกัน เช่น ระบบภายในองค์กรอาจประกอบด้วย แอปพลิเคชั่นที่ทำงานอยู่บน .NET และ J2EE Platform ดังนั้น การกำหนดสถาปัตยกรรม ระดับแอปพลิเคชั่นที่ชัดเจนจะแสดงให้เห็นถึงการแก้ไขปัญหาที่ถูกต้อง หัวใจสำคัญของระดับแอปพลิชั่น คือ การสะท้อนให้เห็นถึงวิธีการแก้ปัญหาของแต่ละแอปพลิเคชั่นอย่างชัดเจน ส่วนรายละเอียดการประกอบและรวบรวมแอปพลิเคชั่นต่าง ๆ ที่อยู่ภายในองค์กร จะถูกกำหนดไว้ใน “สถาปัตยกรรมระดับองค์กร (Enterprise Architecture)”

2.3.2 สถาปัตยกรรมระดับองค์กร (Enterprise Architecture)

โดยทั่วไปสภาพแวดล้อมขององค์กรที่มีขนาดใหญ่จะประกอบด้วยแอปพลิเคชั่นที่มีสถาปัตยกรรมที่ต่างกันทำให้ระบบมีความซับซ้อนมากขึ้น จึงจำเป็นต้องมีการควบคุมโครงสร้างต่าง ๆ ของระบบอย่างรอบคอบ เพื่อให้แอปพลิเคชั่นต่าง ๆ ทำงานร่วมกันได้อย่างมีประสิทธิภาพ ดังนั้น ควรกำหนดแบบแผนเพื่อใช้เป็นแม่แบบให้กับแอปพลิเคชั่นต่าง ๆ ที่ทำงานอยู่ภายใต้ระบบองค์กร

สถาปัตยกรรมระดับองค์กร (Enterprise Architecture) เป็นแบบแผนหรือ ข้อกำหนดขององค์กรที่มีลักษณะคล้ายกับการวางผังเมือง มีจุดประสงค์เพื่อนำแอปพลิเคชั่นต่าง ๆ ที่พัฒนาด้วยสถาปัตยกรรมระดับแอปพลิเคชั่นมาทำงานร่วมกัน กล่าวคือ การวางแบบแผน (Plan) หรือแม่แบบเพื่อพัฒนาระบบในองค์กรเปรียบเทียบได้กับสถาปัตยกรรมระดับองค์กร การเปลี่ยนแปลงสถาปัตยกรรมระดับองค์กร จะส่งผลกระทบโดยตรงต่อสถาปัตยกรรมระดับแอปพลิเคชั่น ดังนั้น

ไม่ควรเปลี่ยนแปลงข้อกำหนดของสถาปัตยกรรมระดับองค์กรโดยไม่จำเป็น นอกจากนี้ สถาปัตยกรรมระดับองค์กรยังแสดงถึงการวางแผนองค์กรในระยะยาว เพื่อนำเทคโนโลยีและสภาพแวดล้อมใหม่ ๆ มาใช้งาน โดยเอกสารนี้อาจกำหนดเทคโนโลยีและนโยบายที่ใช้เป็นมาตรฐานด้านความปลอดภัย (Security) อีกด้วย แต่นิยมเก็บแยกไว้ในแต่ละข้อกำหนดของโครงสร้างด้านความปลอดภัย (Security Architecture)

2.3.3 สถาปัตยกรรมเชิงบริการ (Service-Oriented Architecture)

สถาปัตยกรรมเชิงบริการ (Service-Oriented Architecture หรือ SOA) เป็นสถาปัตยกรรมที่ครอบคลุมทั้งระดับองค์กร และระดับแอปพลิเคชั่น อีกทั้งยังมีความก้าวหน้ามากกว่า จุดประสงค์สำคัญของ SOA คือการนำไปประยุกต์ใช้เพื่อทำให้เกิดการทำงานข้ามสภาพแวดล้อมของ Solution ที่ต่างกันได้ ลักษณะดังกล่าวเป็นการลงทุนเพื่อสร้างบริการที่สามารถนำกลับมาใช้ใหม่ (Reusable) และทำงานร่วมกันได้ (Interoperable) ซึ่งอยู่บนพื้นฐานของรูปแบบการติดต่อสื่อสารที่เป็นกลาง (Neutral) โดยไม่ขึ้นกับ Platform ใด

SOA เป็นแนวคิดที่สนับสนุนความสามารถในการนำมาประกอบกัน (Composable) ดังนั้น โครงสร้างของแต่ละแอปพลิเคชั่นที่ทำงานร่วมกันภายใน SOA สามารถใช้เป็นส่วนเพิ่มเติม (Extension) และเทคโนโลยีที่แตกต่างกันได้ ทำให้ภายในองค์กรอาจมีระบบของ SOA สามารถทำงานร่วมกันมากกว่า 1 ระบบ แต่ระบบภายในองค์กรไม่จำเป็นต้องใช้ SOA พัฒนาทั้งหมด

2.4 วิวัฒนาการของ SOA

วิวัฒนาการของ SOA จะเริ่มจาก Client-Server Architecture และ Distrubute Internet Architecture ไปจนถึงการนำ SOA ไปใช้กับสถาปัตยกรรมแบบเก่าที่เรียกว่า “Hybrid Web Service Architecture” (การพัฒนาระบบ และสถาปัตยกรรมด้วย Web Service : สุธี พงศาสกุลชัย,2550)

2.4.1 Client-Server Architecture

ภายในที่ประกอบไปด้วยกลุ่มของซอฟต์แวร์ที่ทำงานร่วมกันจะต้องมีการรับส่งข้อมูลข่าวสารระหว่างกัน โดยทั่วไปแอปพลิเคชั่นจะใช้วิธีการรับส่งข้อมูลที่เรียกว่า “Client-Server” ซึ่งยังคงได้รับความนิยมมาตั้งแต่อดีตจนถึงปัจจุบัน ในที่นี้ความหมายของคำว่า Client-Server Architecture คือ สภาพแวดล้อม หรือลักษณะการทำงานระหว่างผู้ร้องขอบริการ (Client) และผู้ให้บริการ (Server) ในยุคแรก ๆ ที่มีกฎเกณฑ์เฉพาะ และมีลักษณะการนำไปใช้งานที่ชัดเจน โดยระบบนี้แบ่งชั้นของการทำงานออกเป็น 3 ระดับ

- Presentation Logic เป็นชั้นที่กำหนดรูปแบบ และวิธีการติดต่อกับแอปพลิเคชั่นโดยทั่วไปจะอยู่ในรูปแบบของ Graphic User Inteface (GUI)

- Business Logic เป็นชั่นที่เกี่ยวข้องกับวิธีการทำงานของแอปพลิเคชั่น ที่ใช้สำหรับแก้ไขปัญหาต่าง ๆ ทางธุรกิจ
- Data Access Logic เป็นชั้นที่เกี่ยวข้องกับการจัดการข้อมูล

รู้จักกับ Client-Server Architecture

ระบบ Mainframe เป็นจุดเริ่มต้นของหลักการ Client-Server Architecture โดย Mainframe เป็นระบบที่ค่อนข้างใหญ่ สิ่งเปลืองค่าใช้จ่าย และดูแลรักษายาก เนื่องจากต้องคอยให้บริการ Thin Client จำนวนมาก โดยอาศัย “การประมวลผลแบบสู่ศูนย์กลาง (Centralized Computing)” นั่นคือ การประมวลผลข้อมูลทั้งหมดจะกระทำบนเครื่อง Server เท่านั้น ทำให้ต้องใช้เครื่อง Server ที่มีสมรรถนะสูง ซึ่งสถาปัตยกรรมดังกล่าวมีลักษณะแบบ Single-Tier Architecture

เพื่อให้ Server สามารถรับข้อมูลจากปลายทางได้อย่างต่อเนื่อง ต่อมาระบบ Mainframe เริ่มได้รับความนิยมลดลง เนื่องจากวิวัฒนาการของ Two-Tier Architecture ซึ่งเป็นการต่อยอดจากระบบ Client-Server

เปรียบเทียบ Client-Server Architecture กับ SOA

โดยทั่วไปแล้วระบบ Client-Server จะเก็บรักษา Business Rule ที่มีความสัมพันธ์กับข้อมูลไว้ภายใน Stored Procedure และ Trigger บนฐานข้อมูล นอกจากนี้การกำหนด Business Logic บางส่วนไว้บน Client ทำให้การเขียนโปรแกรม เพื่อเข้าถึงข้อมูล ได้ง่ายขึ้น แต่การเข้าถึงข้อมูลของ SOA นั้น Server จะเก็บ Application Logic ไว้ภายใน และกระจายไปยังผู้ที่ต้องการขอใช้บริการผ่าน Service Description โดย Client จะอาศัยการแลกเปลี่ยน SOAP Message เพื่อขอ Service Description สำหรับเรียกใช้บริการ ผู้ร้องขอบริการไม่ต้องประมวลผล Application Logic และไม่ต้องรู้โครงสร้างหรือแบบแผนการทำงานของ แอปพลิเคชั่นบนฝั่งผู้ให้บริการ เพียงแต่ทราบรูปแบบของ SOAP Message ที่ใช้สำหรับติดต่อ สื่อสารกับ Service เท่านั้น

Application Logic ของระบบ Client-Server จะอยู่ภายใน Component ของ Client ที่เป็น Workstation ซึ่งจะรับผิดชอบการประมวลผลต่าง ๆ ประมาณ 80% ส่วน Database Server จะทำหน้าที่ประมวลผลประมาณ 20% แม้ว่า Database Server จะทำหน้าที่ ประมวลผลเพียง 20% แต่ถ้ามีการเรียกใช้ฐานข้อมูลบ่อยครั้ง จะทำให้ประสิทธิภาพ การให้บริการของ Database Server ลดลง แต่ SOA จะใช้วิธีกระจาย Service ต่าง ๆ ภายในระบบ ซึ่ง Server แต่ละเครื่องจะบรรจุกลุ่มของ Service ที่ทำงานร่วมกันไว้ภายใน และแต่ละ Service อาจถูกประกอบจาก Service ที่อยู่บน Server ต่าง ๆ กันได้ ดังนั้น จึงสามารถกระจายการประมวลผล Service เพื่อแบ่งเบาภาระการประมวลผลของ Server ได้

ในระบบ Two-Tier เมื่อมีผู้ใช้จำนวนมากร้องขอการเชื่อมต่อ (Connection) กับฐานข้อมูล ซึ่งวิธีการเชื่อมต่อแบบ Synchronous ที่ต้องคงสถานะการเชื่อมต่อของทุก Client ไว้ ทำให้ต้องใช้ทรัพยากรจำนวนมาก อาจทำให้เกิดความเสียหาย หรือข้อผิดพลาดต่อการทำงานของ Database Server จนไม่สามารถให้บริการแอปพลิเคชั่นต่าง ๆ ได้ แต่ใน SOA การติดต่อสื่อสารระหว่าง Service กับผู้ร้องขอสามารถเป็นได้ทั้งแบบ Synchronous และ Asynchronous รวมทั้งไม่ต้องคงสถานะการเชื่อมต่อไว้ตลอดเวลา ทำให้เกิดความยืดหยุ่นในการประมวลผล และเกิดประสิทธิภาพมากขึ้น โดยเฉพาะกับการติดต่อสื่อสารแบบ Asynchronous

ในระบบ Two-Tier นั้น Client ถูกกำหนดให้รับผิดชอบการประมวลผลข้อมูล และเรียกใช้ทรัพยากรจากผู้ให้บริการต่าง ๆ ที่อยู่ในระบบเครือข่าย ทำให้ซอฟต์แวร์ฝั่ง Client ต้องใช้หน่วยความจำเป็นจำนวนมาก แต่ใน SOA ผู้ร้องขอบริการไม่ต้องประมวลผลการทำงานด้วยตัวเอง

และมีลักษณะการประมวลผลแบบกระจาย (Distributed) ซึ่งมีวิธีจัดการกับทรัพยากรต่าง ๆ ของ Server อย่างเหมาะสมทำให้แต่ละ Service มีขอบเขตหน้าที่ และการใช้ทรัพยากรที่ชัดเจน

การพัฒนาแอปพลิเคชั่นของระบบ Client-Server จะใช้ภาษาโปรแกรมยุคที่ 4 (4GL Programming Language) เช่น Visual Basic หรือ PowerBuider เป็นต้น สภาพแวดล้อมการพัฒนาดังกล่าวได้นำจุดเด่นของระบบปฏิบัติการ Windows มาใช้งาน โดยเตรียมความสามารถในการสร้าง User-Interface ที่ใช้งานง่าย สวยงาม และมีประสิทธิภาพไว้ นอกจากนี้ผู้ผลิตซอฟต์แวร์หลักที่พัฒนาฐานข้อมูล เช่น Oracle IBM Infomix และ Microsoft ได้หันมาให้ความสำคัญกับการพัฒนาระบบจัดการ ฐานข้อมูลเชิงสัมพันธ์ (RDBMS) ที่สามารถจัดการได้หลาย Connection โดยจัดเตรียมเครื่องมือสำหรับจัดเก็บ และบริหารข้อมูลไว้มากมาย แต่เทคโนโลยีต่าง ๆ ที่นำมาใช้ SOA ไม่ได้มีการเปลี่ยนแปลงมากนัก แต่ถูกขยายออกไปเป็นเทคโนโลยีต่าง ๆ มากมาย ภาษาสำหรับพัฒนาโปรแกรมที่ได้รับการพัฒนามาจากเวอร์ชั่นก่อน เช่น Visual Basic ยังคงถูกใช้เพื่อพัฒนา Web Service และส่วนของฐานข้อมูลเชิงสัมพันธ์ (Relational Database)

เหตุผลที่ทำให้ระบบ Client-Server ได้รับความนิยมน้อยลงคือ ต้องใช้ค่าใช้จ่ายจำนวนมากสำหรับการบำรุงรักษาแอปพลิเคชั่นของ Client และทำได้ยากลำบาก เนื่องจากเครื่อง Client อาจติดตั้งซอฟต์แวร์ภายในที่มีการทำงานแตกต่างกัน หรือมีอุปกรณ์ฮาร์ดแวร์จากหลายผู้ผลิต นอกจากนี้เมื่อ Client มีจำนวนมากขึ้น ทำให้เกิดความต้องการเรียกใช้ข้อมูลจาก Database Server เพิ่มขึ้น จึงยากต่อการจัดการ แต่ SOA สามารถบำรุงรักษา และขยายระบบได้สะดวกรวดเร็ว รวมทั้งเสียค่าใช้จ่ายน้อยกว่า เนื่องจากแต่ละแอปพลิเคชั่นจะทำงานเป็นอิสระต่อกัน

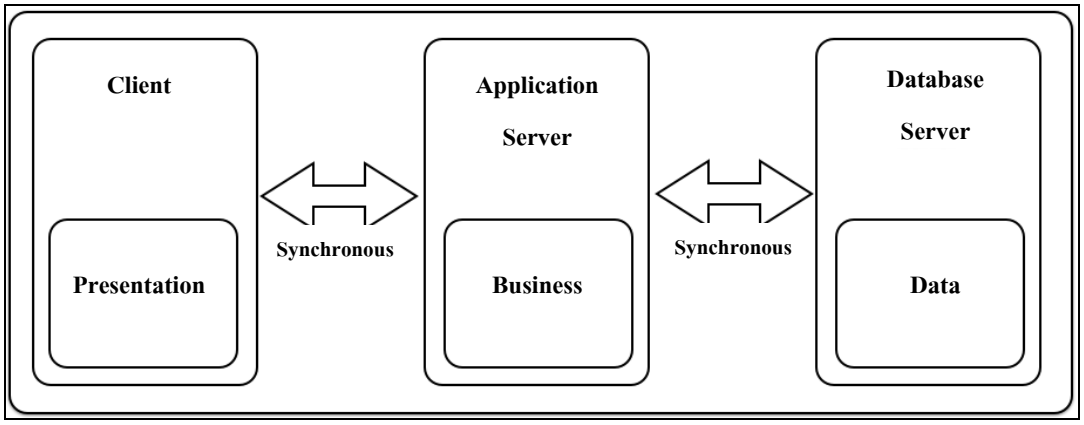
2.4.2 Distributed Internet Architecture

รู้จักกับ Distribute Internet Architecture

Two-Tier Client-Server Architecture ได้รับผลกระทบจากปัญหาเรื่องต้นทุนที่สูงและข้อจำกัดต่าง ๆ ทำให้หลักการสร้างแอปพลิเคชั่นที่ประกอบด้วยกลุ่มคอมโพเนนท์ (Component-Based Application) นั้นมีแนวโน้มที่จะได้รับความนิยมเพิ่มขึ้น และได้มีการนำเสนอ Multi-Tier Client-Server Architecture นำไปสู่หลักการออกแบบเชิง Component ที่ได้รับอิทธิพลมาจากแนวคิดเชิงวัตถุ (Object-Oriented)

ลอจิกของ Distributed Application ที่กระจายอยู่ภายใน Component ต่าง ๆ ของ Client และ Server ช่วยลดความยุ่งยาก และแบ่งเบาภาระที่เกิดจากการรวบรวมลอจิกทั้งหมดไว้บน Server เพื่อใช้เป็นจุดศูนย์กลาง (Centralizing) โดย Server-Side Component ถูกเก็บไว้บนแอปพลิเคชั่นของ Server ซึ่งทำหน้าที่แบ่งปัน (Share) และจัดการกับส่วนกลางของการเชื่อมต่อฐานข้อมูล (Database Connection Pool) ช่วยลดภาระของ Database Server เมื่อมีการเรียกใช้ข้อมูลเป็นจำนวนมาก โดยการ

เชื่อมต่อแบบ Single Connection ทำให้การจัดการกับผู้ใช้จำนวนมาก (Multiple User) ทำได้ง่ายขึ้น อย่างไรก็ตามลักษณะดังกล่าว ทำให้ต้องใช้ต้นทุนมากขึ้น การจัดการกับกระบวนการบริหาร และการพัฒนาที่เหมาะสมก็ทำได้ยาก กล่าวคือการสร้าง Component ที่ประมวลผลได้หลายงาน เพื่อให้สามารถรองรับหลายคำร้องขอ (Request) ได้พร้อมกัน ส่งผลให้การพัฒนามีความยุ่งยากซับซ้อนมากว่าแบบ Single User



รูปที่ 2.11 แสดงภาพของ Distribute Internet Architecture

จากรูปข้างต้น Browser ของ Client จะไม่มีการติดตั้ง Logic ใด ๆ ไว้ภายใน โดยทำหน้าที่แสดงผลข้อมูลเท่านั้น ส่วน Web Server จะบรรจุ Presentation Logic Business Rule และ Connection Pooling ไว้ภายใน โดยทำหน้าที่เป็นศูนย์กลางของ Application Logic ทำให้มีการประมวลข้อมูลจำนวนมาก

Distribute Internet Architecture ทำให้เกิด Physical Tier ที่เรียกว่า Web Server ขึ้น โดยเป็นผลมาจากการนำโปรโตคอล HTTP มาใช้ในการติดต่อสื่อสารระหว่าง Client และ Server แทนโปรโตคอล RPC โดย Distribute Internet Architecture มีส่วนแสดงให้เห็นถึงรูปแบบของ Computing Platform ที่นำไปพัฒนาแอปพลิเคชั่นระดับองค์กร การที่โปรแกรมเมอร์มีความชำนาญในการเขียนโปรแกรมแบบ Component-Based มากขึ้นประกอบกับวิวัฒนาการของ Middleware ที่มีประสิทธิภาพมากขึ้น ทำให้ลดความซ้ำซ้อนในการพัฒนา

เปรียบเทียบ Distribute Internet Architecture กับ SOA

Service-Oriented Architecture มีลักษณะคล้ายกับ Distribute Internet Architecture อย่างมาก โดย Distribute Internet Architecture ประกอบด้วยกลุ่มของ Component ที่อยู่ภายใน Application Server ใด ๆ Component ถูกออกแบบมาให้มีหน้าที่แตกต่างกัน ขึ้นอยู่กับงานที่ปฏิบัติ Component ที่อยู่บน Server เดียวกันจะติดต่อสื่อสารกันผ่าน API โดยอาศัย Public Interface ส่วนโปรโตคอล RPC ถูกใช้เพื่อทำให้สามารถติดต่อสื่อสารข้ามขอบเขตของ Server ได้ ด้วยการใช้ Proxy Sub ที่ใช้แทน Component บน Remote

Distribute Internet Architecture นั้นสนับสนุนโปรโตคอลต่าง ๆ ที่ไม่เป็นกลาง เช่น การแลกเปลี่ยนข้อมูลระยะไกลของ DCOM หรือ CORBA ทำให้เกิดการแข่งขันกันทางเทคโนโลยีทั้งในด้านประสิทธิภาพและความน่าเชื่อถือ โดยเฉพาะเรื่องของประสิทธิภาพในการเชื่อมต่อ และไม่สามารถทำงานข้าม Platform กันได้ แต่ SOA สนับสนุนโปรโตคอลที่เป็นกลาง โดยวิธีการติดต่อสื่อสารด้วย SOAP Message ซึ่งในรูปแบบของเอกสาร XML

Distribute Internet Architecture มีวิวัฒนาการเริ่มต้นจากแนวคิดของ Component Server-Side Script และเทคโนโลยีของ Web ตัวอย่างเช่น HTML และ HTTP เป็นต้น ซึ่งมีข้อจำกัดมากมาย จึงได้มีการปรับปรุง Middleware เพื่อเพิ่มประสิทธิภาพการประมวลผล และการควบคุมการติดต่อสื่อสารให้ดีขึ้น โดยการนำ XML มาใช้แทนข้อมูลที่มีความซับซ้อนเพื่อถ่ายทอดผ่าน Internet Protocol ทำให้เกิดการพัฒนาเป็น Web Service ที่ทำให้ Distribute Internet Architecture สามารถติดต่อสื่อสารข้าม Platform กันได้ แต่เมื่อ Application Logic ถูกกระจายข้ามไปยังขอบเขตต่าง ๆ การนำหลักการด้านความปลอดภัยมาใช้งาน เช่น การพิสูจน์ตัวตน (Authentication) หรือการกำหนดสิทธิ์การเข้าถึงข้อมูล (Authorization) จะทำได้ยากขึ้น

2.4.3 Service-Orientation และ Object-Orientation

แม้ว่า Service-Oriented และ Object-Oriented จะมีความสัมพันธ์กัน แต่ก็มีส่วนที่แตกต่างกัน ในส่วนนี้จะกล่าวถึงความแตกต่างระหว่างหลักการของ Object-Oriented Programming และ Service-Orientation เพื่อให้ผู้ใช้สามารถเข้าใจและนำมาใช้งานร่วมกันได้ดียิ่งขึ้น การเปรียบเทียบ Service-Oriented และ Object-Oriented จะอยู่ในลักษณะของการออกแบบ โดย Service-Oriented จะอยู่บนพื้นฐานการออกแบบเพื่อสร้าง Service ส่วน Object-Oriented จะอยู่บนพื้นฐานการสร้าง Object

Service-Oriented และ Object-Oriented จะมีความสัมพันธ์ในแง่ที่ว่า “การเขียนโปรแกรมเชิงวัตถุถูกนำมาใช้สร้างคำสั่งควบคุมการทำงาน (Application Logic) ที่ถูกห่อหุ้มไว้ภายใน Service” โดย Object-Oriented จะให้ความสำคัญกับการสร้าง Object ส่วน Service-Oriented จะอยู่บนพื้นฐานของการออกแบบ Service เทคโนโลยีทั้งสองมีหลักการที่แตกต่างดังนี้

Service-Orientation ให้ความสำคัญกับความเป็นอิสระต่อกัน (Loose Coupling) ระหว่าง Service ทำให้ Service มีความเป็นอิสระต่อกันอย่างแท้จริง แต่ Object-Orientation จะสนับสนุนการเขียนโปรแกรมในลักษณะของการนำกลับมาใช้ใหม่ (Reuseable) ซึ่งอยู่บนพื้นฐานของความเป็นอิสระต่อกัน (Loose Coupling) ในรูปแบบของ Class ทำให้ Object ที่ได้รับการสืบทอดต่อกันมา ยังคงรักษาความสัมพันธ์ที่แนบแน่นระหว่างกันไว้

Service-Orientation สนับสนุนหลักการของ Interface ที่ไม่ซับซ้อน โดยนำไปใช้เพื่อสร้าง Service Description เท่านั้น ส่วน Object-Oriented Programming จะรองรับการใช้งาน Interface อย่างเต็มรูปแบบ (APIs)

Service-Orientation สนับสนุนสารนำ Service ต่าง ๆ ที่เป็นอิสระต่อกันมาประกอบ (Composition) เพื่อทำงานร่วมกัน ส่วน Object-Oriented นอกจากจะสนับสนุนหลักการ Composition แล้ว ยังสนับสนุนการสืบทอดคุณสมบัติ (Inheritance) ระหว่าง Object อีกด้วย

เมื่อ SOA ได้พัฒนาความสนใจมากขึ้น จึงจำเป็นต้องมีองค์กรสำหรับกำหนดมาตรฐานให้ชัดเจนให้กับเทคโนโลยีที่จะนำมาใช้พัฒนาระบบด้วย SOA ซึ่งประกอบด้วย 3 องค์กรสำคัญ คือ (1) World Wide Web Consortium : W3C (2) Organization for the Advancement of Structured Information Standards (OASIS) และ (3) Web Service Interoperability Organization (WS-I) โดยองค์กรเหล่านี้จะกำหนดมาตรฐานที่เกี่ยวข้องกับเทคโนโลยีของ XML และ Web Service ที่นำไปใช้เพื่อพัฒนาระบบด้วย SOA แต่การกำหนดมาตรฐานต่าง ๆ ต้องได้รับการยอมรับจากผู้ผลิตซอฟต์แวร์ด้วย

SOA ได้รับการพัฒนามาจากแนวคิดของสถาปัตยกรรมแบบเก่า มีจุดประสงค์เพื่อลดข้อจำกัดของการทำงานในด้านต่าง ๆ โดยสถาปัตยกรรมสำคัญที่เป็นจุดเริ่มต้นของ SOA ได้แก่ Client-Server Architecture และ Distribute Internet Architecture ผู้พัฒนาระบบสามารถนำ Web Service มาใช้งานร่วมกับสถาปัตยกรรมแบบเก่าได้ เพื่อเพิ่มประสิทธิภาพการทำงาน และลดค่าใช้จ่ายในการเปลี่ยนแปลงระบบเดิมที่มีอยู่

2.5 องค์ประกอบสำคัญในการพัฒนา Service

SOA สามารถนำไปประยุกต์ได้กับหลายเทคโนโลยี แต่ Web Service เป็นเทคโนโลยีที่สามารถนำเอาหลักการของเทคโนโลยีเชิงบริการ (Service-Oriented) มาใช้ได้เป็นรูปธรรมมากที่สุด ดังนั้นองค์ประกอบในการพัฒนา Service ที่จะกล่าวถึงต่อไปนี้จะใช้ Web Service เป็นตัวแทนในการอธิบายโครงร่างการทำงานขององค์ประกอบต่าง ๆ ของ Service ได้แก่ Service Service Description (WSDL) และ Messaging (SOAP)

2.5.1 Web Service Framework

โครงร่างการทำงาน หรือกรอบการทำงาน (Framework) คือการรวบรวมความคิดต่าง ๆ ที่ประกอบด้วยกลุ่มของสถาปัตยกรรม (Architecture) เทคโนโลยี (Technology) แนวคิด (Concept) และแบบจำลอง (Model) ต่าง ๆ เพื่อใช้เป็นข้อกำหนดในการพัฒนาสิ่งใดสิ่งหนึ่ง โดย Framework ของ Web Service มีลักษณะต่าง ๆ ดังนี้

- ส่วนประกอบหลักของ Web Service ประกอบด้วย Web Service Service Description และ Message

- กำหนดมาตรฐานโดย Standard Organization โดยไม่ขึ้นอยู่กับผู้ผลิตซอฟต์แวร์ใด ๆ แต่สามารถนำไป Implement บน Platform ของผู้ผลิตซอฟต์แวร์ต่าง ๆ แล้วนำมาทำงานร่วมกันได้

- ข้อตกลงในการติดต่อสื่อสารถูกกำหนดโดย Service Description ที่อยู่ในรูปแบบของ WSDL

- Messaging Framework ที่ใช้ในการติดต่อสื่อสารอยู่ในรูปแบบของ SOAP

ใช้สถาปัตยกรรมของ UDDI ในการลงทะเบียน และค้นหา Service Description Framework ของ Web Service เป็นแนวคิดที่ถูกหลอมรวม หรืออยู่บนแนวทางเดียว กับหลักการของแนวคิดเชิงบริการ (Service-Orientation)

2.5.2 แนวคิดการพัฒนา Service

Service อาจมีบทบาทที่แตกต่างกันขึ้นอยู่กับลักษณะการแลกเปลี่ยน Message และสภาพแวดล้อมที่ถูกเรียกใช้ โดยอาจเป็นทั้งผู้ให้บริการ (Service Provider) ผู้ร้องขอบริการ (Service Requester) หรือสื่อกลางบริการ (Intermediary) นอกจากนี้ Service ยังสามารถแบ่งได้ หลายประเภทขึ้นอยู่กับวิธีการนำไปใช้ และลักษณะของ Application Logic ที่อยู่ภายใน แต่มี 3 ชนิดที่สำคัญ คือ (1) Business Service Model (2) Unity Service Model และ (3) Controller Service Model

การติดต่อสื่อสารระหว่าง Service จะอาศัยเอกสาร Service Description เพื่อบอกรายละเอียดและวิธีการเรียกใช้ Service (Web Service จะใช้ WSDL มีการสร้างเอกสาร Service Description) หาก Service ใดต้องการเรียกใช้บริการจาก Service อื่นจะต้องค้นหา Service Description ของ Service นั้นผ่านหน่วยสืบค้นบริการ หรือ Service Registry (Web Service จะใช้ UDDI สำหรับลงทะเบียน Service ไว้ในบัญชี) เพื่อให้ผู้ร้องขอบริการเข้ามาค้นหา และทุกการติดต่อสื่อสารระหว่าง Service จะอยู่ในรูปแบบของ Message และใช้โปรโตคอลที่ติดต่อสื่อข้าม Platform กันได้ (Web Service จะใช้ SOAP เป็นโปรโตคอลขนส่ง)

2.6 หลักการของ Service-Oriented

2.6.1 Enterprise Logic

ลอจิกต่าง ๆ ที่ถูกรวบรวมไว้ภายในระบบขององค์กร หรือ Enterprise Logic จะต้องเปลี่ยนรูปแบบการทำงานไปตามความต้องการที่เกิดขึ้นจากภายใน และภายนอกองค์กร โดย Enterprise Logic แบ่งได้เป็น 2 ประเภท คือ Business Logic และ Application Logic

- Service Autonomous คือ Service จะต้องควบคุมลอจิกต่าง ๆ ที่อยู่ภายในขอบเขตของตนเองได้โดยไม่ต้องอาศัยการทำงานจาก Service อื่น

- Service Statelessness คือ Service ควรเก็บข้อมูลของสถานะ หรือกิจกรรมที่ได้ดำเนินการไปแล้วให้น้อยที่สุด

- Service Discoverable คือ Service ต้องจัดเตรียมส่วนอธิบายที่เรียกว่า Service Description ไว้เพื่อให้ Service Requester ค้นหา เพื่อเรียกใช้บริการ

สามารถสรุปได้คือ การนำหลักการของ SOA มาประยุกต์ใช้กับระบบภายในองค์กร จำเป็นต้องเข้าใจรูปแบบลอจิกต่าง ๆ ที่ถูกรวบรวมไว้ภายในองค์กร หรือเรียกว่า Enterprise Logic ซึ่งมี 2 ประเภทคือ Business Logic และ Application Logic โดย Business Logic จะแสดงรายละเอียดของกระบวนการทำงานต่าง ๆ ส่วน Application Logic จะนำ Business Logic มาประยุกต์ใช้งานบนสภาพแวดล้อมหรือเทคโนโลยีที่ต่างกัน เมื่อนำเอาหลักการของ SOA มาประยุกต์ใช้งานทำให้เกิดแนวคิดใหม่ที่เรียกว่า Service ซึ่งถูกวางไว้ระหว่างชั้น ของ Business Logic กับ Application Logic เรียกว่า Service Interface Layer

2.7 Service Layer และการวางแผนในการพัฒนา SOA (Service-Oriented Architecture Concept Technology and Design : Thomas Erl)

2.7.1 ความสำคัญของ Service Layer

Service Interface Layer จะอยู่ระหว่าง Business Layer และ Application Layer เพื่อทำหน้าที่เป็นตัวเชื่อมต่อ Layer ทั้งสอง โดย Service Interface Layer เป็นส่วนที่ใช้กำหนดคุณลักษณะต่าง ๆ ของ Contemporary SOA ให้กับระบบ แต่การนำคุณลักษณะต่าง ๆ มา Implement เพื่อใช้ประโยชน์นั้นจะต้องกระทำผ่าน “Service Layer” ซึ่งเป็น Layer ย่อยที่อยู่ภายใน Service Interface Layer

Service Layer ทำให้เกิดลักษณะสำคัญต่าง ๆ ดังนี้

- Service Layer ทำให้เกิดความเป็นอิสระต่อกัน

Service Interface Layer สามารถแบ่งออกเป็น Layer ย่อย ๆ ของ Service ได้อีก โดยเรียกว่า Service Layer เพื่อให้ระบบมีความเป็นอิสระต่อกันมากขึ้น โดยกลุ่มของ Service ที่อยู่ในแต่ละ Service Layer จะทำหน้าที่รวบรวมลอจิกต่าง ๆ ไว้ภายใน ทำให้แต่ละส่วนของระบบไม่ขึ้นต่อส่วนอื่น ๆ ลอจิกที่อยู่ใน Business Layer จะทำให้เกิดความเป็นอิสระและไม่ขึ้นกับเทคโนโลยีใด ๆ โดยส่วนที่ระบุถึงเทคโนโลยีจะกำหนดไว้ใน Application Logic แทน ทำให้เกิดความสัมพันธ์ที่เป็นอิสระต่อกันระหว่าง Business Logic และ Application Logic

- Service Layer สามารถใช้ Service แทน Application Logic ของระบบเดิมได้

การนำ Application Logic ของระบบเดิมมาใช้งาน จำเป็นต้องนำมาใช้ผ่าน Service รวมทั้งการพัฒนาลอจิกใหม่ ๆ ขึ้นมาใช้งาน ก็จำเป็นต้องสนับสนุนการทำงานของ Service เช่นกัน โดยสามารถกำหนดข้อบังคับ ข้อจำกัด และความต้องการของระบบเดิมได้ เพื่อนำไปใช้พิจารณา ในขั้นตอนของการออกแบบ Service ต่อไป ในการใช้ Service Layer เพื่อควบคุมแอปพลิเคชั่น ต่าง ๆ ของระบบเดิมนั้นจำเป็นต้องนำหลักการบางอย่างของ SOA มาใช้งาน กลุ่มของ Service เหล่านี้จะถูกออกแบบมาเพื่อใช้ทำหน้าที่แทน Application Logic และเก็บไว้ใน Service Layer ที่จัดเตรียมไว้ เรียกว่า “Application Service Layer”

- Service Layer สามารถใช้ Service แทน Business Logic ได้

การใช้ Service เพื่อบรรจุ Business Logic ของกระบวนการทางธุรกิจ (Business Process) หรือแบบจำลองธุรกิจ (Business Model) ที่ออกแบบไว้ ต้องแน่ใจว่าการออกแบบ Service นั้น จะสามารถทำหน้าที่แทนลอจิกเหล่านั้นได้อย่างถูกต้องตาม Business Model ที่กำหนดไว้ โดย Service ที่ถูกออกแบบตามลักษณะดังกล่าวจะถูกจัดเก็บไว้ใน Service Layer ที่เรียกว่า “Business Service Layer”

- Service Layer ทำให้การพัฒนา Service มีความคล่องตัว (Agility)

ปัจจัยสำคัญในการสร้างความคล่องตัว (Agility) ให้กับระบบของ SOA คือ ลดการ พึ่งพากันระหว่าง Service ให้เหลือน้อยที่สุด ด้วยวิธีเก็บลอจิกที่ใช้ดำเนินงานต่าง ๆ ไว้ภายในตัวเอง โดย Service จะบรรจุ Business Rule ที่ใช้เป็นข้อบังคับสำหรับการดำเนินการต่าง ๆ ในขณะ Runtime ข้อบังคับเหล่านี้ใช้เพื่อจำกัดการใช้งาน Service จากภายนอกระบบ โดย Service จะทำหน้าที่เหมือนเป็น Controller Service ซึ่งภายในประกอบด้วย Service ย่อย โดย Controller Service สามารถพัฒนา Service ที่อยู่ภายในได้อย่างอิสระ แต่ต้องรักษาโครงสร้างของ Service ที่นำมาประกอบกันไว้ ด้วยลักษณะดังกล่าวจึงทำให้ Service มีความคล่องตัวยิ่งขึ้นนั่นเอง

2.7.2 วิธีวางแผนการพัฒนาระบบด้วย SOA

การพัฒนาระบบด้วย SOA จะมีลักษณะคล้ายกับการพัฒนา Distribute Application กล่าวคือ Service จะถูกออกแบบ (Design), พัฒนา (Develop) และนำไปใช้ (Deploy) ให้เหมาะสมกับ Component มาตรฐาน และเทคโนโลยีต่าง ๆ ที่มีอยู่

รูปที่ 2.17 แสดงขั้นตอนของ Service-Oriented Analysis

ขั้นตอนที่ 1 : รวบรวมความต้องการ

ขั้นตอนแรกเป็นการรวบรวมความต้องการทางธุรกิจที่เกิดขึ้น เพื่อใช้เป็นเอกสาร ในการวิเคราะห์กระบวนการต่าง ๆ และกำหนดขอบเขต ในการวิเคราะห์ความต้องการ เพื่อใช้ในการสร้าง Service โดยกระบวนการต่าง ๆ ที่กำหนดขึ้นจะนำไปใช้เป็นข้อมูล ในการกำหนดแบบจำลองที่ใช้อธิบายกระบวนการทำงานของระบบต่อไป

ขั้นตอนที่ 2 : วางแผนการทำงานของระบบเก่า และระบบใหม่

ความต้องการต่าง ๆ ที่กำหนดไว้ในขั้นตอนที่ 1 จะมีผลต่อการนำ Apllication Logic ต่าง ๆ ของระบบเดิมมาใช้งาน แต่ Service-Oriented Analysis ไม่ได้กำหนดว่าจะต้องนำ Web Service มาใช้งานแทนที่ หรือห่อหุ้มแอปพลิเคชั่นเดิมที่มีอยู่ โดยขั้นตอนนี้จะรวบรวมข้อมูลที่เกี่ยวข้องกับความสัมพันธ์ระหว่าง Web Service กับระบบเดิม ทำให้เข้าใจถึงผลกระทบต่าง ๆ ที่อาจเกิดขึ้นกับระบบเดิม หลังจากนำ Web Service มาใช้งาน ส่วนรายละเอียดที่อธิบายถึงวิธีการทำงานร่วมกันระหว่าง Web Service กับระบบเดิมจะถูกกำหนดไว้ใน “ขั้นตอนการออกแบบ (Design)”

ขั้นตอนที่ 3 : กำหนด Service Modeling

ส่วนนี้เป็นการนำเอาข้อมูลต่าง ๆ จากทั้ง 2 ขั้นตอนที่ผ่านมา มาใช้สำหรับกำหนดแบบจำลองการทำงานต่าง ๆ ที่เรียกว่า “Service Modeling” เพื่อระบุถึง Operation ของ Service และจัดกลุ่มการทำงานให้กับแต่ละ Service

ขั้นตอนการออกแบบ (Design)

เมื่อวิเคราะห์ความต้องการต่าง ๆ และทราบรายละเอียดของ Service ที่ต้องการสร้างแล้ว ขั้นตอนต่อไป คือ ออกแบบ (Design) โครงสร้างต่าง ๆ ของระบบ การออกแบบจะต้องเป็นไปตามแบบแผนของมาตรฐานต่าง ๆ ที่นำมาใช้ และอยู่บนหลักการของ SOA ด้วย โดย Service Layer จะถูกออกแบบในขั้นตอนนี้ เรียกขั้นตอนนี้ว่า “Service-Oriented Design”

Service-Oriented Design คือ กระบวนการออกแบบ Service ที่ได้วิเคราะห์ไว้ โดยมีวัตถุประสงค์เพื่อออกแบบส่วนประกอบต่าง ๆ ได้แก่ กำหนดกลุ่มของสถาปัตยกรรม สำคัญที่ต้อง

ใช้ขอบเขตของสถาปัตยกรรมกำหนดมาตรฐานการออกแบบ และออกแบบ ส่วนติดต่อกับ Service กระบวนการออกแบบจะต้องนำข้อมูลต่าง ๆ ที่ได้จัดเตรียมไว้ในขั้นตอนการวิเคราะห์มาใช้งาน โดย Service-Oriented Design ประกอบด้วย 3 ขั้นตอนสำคัญ ได้แก่

รูปที่ 2.18 แสดงขั้นตอนของ Service-Oriented Design

ขั้นตอนที่ 1 : กำหนดเทคโนโลยีมาตรฐาน

ขั้นตอนแรกเป็นการกำหนดเทคโนโลยี และสถาปัตยกรรมต่าง ๆ ที่จะนำมาใช้งานร่วมกัน โดยประกอบด้วย 3 ขั้นตอนสำคัญ คือ (1) การกำหนด Service Layer (2) เลือกมาตรฐานที่นำมาใช้ และ (3) กำหนดข้อกำหนดเพิ่มเติม

ขั้นตอนที่ 2 : ออกแบบ Service

เมื่อกำหนดเทคโนโลยีต่าง ๆ และเลือกมาตรฐานที่ต้องการใช้งานแล้ว ขั้นตอนต่อไปจะเป็นการออกแบบ Service ซึ่งประกอบด้วย 3 กระบวนการสำคัญ คือ (1) การออกแบบ Entity-Centric Business Service (2) การออกแบบ Application Service และ (3) การออกแบบ Task-Centric Business Service

ขั้นตอนที่ 3 : กำหนดขั้นตอนการทำงานของ Service

Service ต่าง ๆ ที่ได้ออกแบบไว้ จะต้องนำมาสร้าง Orchestration Layer โดยการผูก Service กับลอจิกต่าง ๆ เพื่อกำหนดขั้นตอนการดำเนินงาน (Workflow) ของลอจิก

ขั้นตอนการพัฒนา (Development)

เมื่อออกแบบ Service แล้ว ขั้นตอนต่อไปเป็นการพัฒนา Service ที่ได้ออกแบบไว้ด้วยการใช้ภาษาในการพัฒนา และ Platform ที่ระบุไว้เช่น J2EE หรือ .NET เป็นต้น โดยในขั้นตอนนี้ จะไม่คำนึงถึงชนิดของ Service มากนัก แต่จะเน้นไปที่การกำหนดรูปแบบของ Service ในเชิงกายภาพมากกว่า

แนวคิด ข้อกำหนด และเทคโนโลยีเปิดต่าง ๆ ที่ได้กล่าวถึงจำเป็นต้องนำไปใช้พัฒนา เป็นซอฟต์แวร์ด้วย Platform ต่าง ๆ ที่รองรับเทคโนโลยีของ Web Service โดย Platform

เหล่านี้ ได้จัดเตรียมเครื่องมือไว้สำหรับรองรับความต้องการเพื่อพัฒนาซอฟต์แวร์ของ Web Service ซึ่งตอบสนองต่อความต้องการต่าง ๆ ดังนี้

1. จัดเตรียมเครื่องมือสำหรับรองรับการพัฒนาโปรแกรมด้วยภาษาที่ใช้ในการพัฒนาต่าง ๆ

2. จัดเตรียม API ที่มีเครื่องมือและฟังก์ชั่นการทำงานต่าง ๆ ที่สนับสนุนการทำงานแบบ Runtime ทำให้สามารถสร้างซอฟต์แวร์ได้อย่างมีประสิทธิภาพมากขึ้น

3. ซอฟต์แวร์ API และ Runtime ต้องได้รับการสนับสนุนจาก OS (Operating System)

ลักษณะดังกล่าว ทำให้สามารถขยายรายละเอียดในชั้น (Layer) ได้ดังภาพ

เมื่อทำความเข้าใจกับสถาปัตยกรรมของซอฟต์แวร์ข้างต้นแล้ว ต่อไปให้พิจารณาภาพแสดง Layer ต่าง ๆ ของ SOA ที่พัฒนาบน Platform ของ J2EE และ .NET (.Net Web Service : Ballinger)

ขั้นตอนการบริหารและจัดการ (Administration)

เมื่อติดตั้ง Service แล้ว สิ่งที่ควรให้ความสำคัญอย่างยิ่ง คือ การบริหาร และการจัดการระบบ โดยการตรวจสอบร่องรอยเพื่อจัดการกับ Message ตรวจสอบการใช้ Service และตรวจสอบ ประสิทธิภาพในการให้บริการของระบบ ให้เป็นไปตามมาตรฐานที่กำหนดไว้ โดยใช้วิธีเหมือนกับ การบริหารและจัดการ Distributed Application และ Component-Base Application

Service Layer คือ Layer ย่อยที่อยู่ภายใน Service Interface Layer ประกอบด้วย Application Service Layer Business Service Layer และ Orchestration Service Layer เรียก Service ที่ทำงานอยู่ภายใต้แต่ละ Service Layer ว่า Application Service Business Service และ Process Service ตามลำดับ การนำ Service ของแต่ละ Layer มาทำงานร่วมกัน สามารถทำได้หลายรูปแบบขึ้นอยู่กับความต้องการของระบบ โดยแต่ละวิธีการจะมีรูปแบบ และขนาดที่แตกต่างกันขึ้นอยู่กับชนิดของ Service ที่นำมาประกอบ โดย Service ที่อยู่ในแต่ละ Layer จะต้องทำงานสัมพันธ์กัน ประโยชน์ของ Service Layer มีหลายประการ เช่น ทำให้เกิดความเป็นอิสระต่อกัน สนับสนุนการนำกลับมาใช้ใหม่ ทำให้ระบบมีความคล่องตัวยิ่งขึ้น และเชื่อมโยงการทำงานของลอจิกต่าง ๆ

การวางแผนเพื่อพัฒนาระบบด้วย SOA ประกอบด้วย 6 ขั้นตอนสำคัญ ได้แก่ (1) ขั้นตอนการวิเคราะห์ (2) ออกแบบ (3) พัฒนา (4) ทดสอบ (5) นำไปใช้งาน และ (6) บริหารจัดการ โดยขั้นตอน การวิเคราะห์และออกแบบ ถือเป็นหัวใจสำคัญในการพัฒนาระบบด้วย SOA เนื่องจากเป็นขั้นตอนที่นำหลักการของ SOA มาประยุกต์ใช้เพื่อกำหนดโครงสร้างการทำงานของ ระบบในรูปแบบของ SOA ให้สามารถตอบสนองความต้องการขององค์กรได้

2.8 Characteristic ของ Contemporary SOA

ในอนาคตธุรกิจมีแนวโน้มการเปลี่ยนแปลงรูปแบบ และการพัฒนาไปตามแนวทางของ SOA มากขึ้น ผู้ผลิตซอฟต์แวร์ต่าง ๆ ได้พัฒนาข้อกำหนดของ Web Service ขึ้นมา และสร้างภาษา XML ที่มีประสิทธิภาพสูงเพื่อนำมาใช้กับเทคโนโลยี Web Service ซึ่ง Platform ส่วนใหญ่ที่มีอยู่ในปัจจุบันต่างรองรับเทคโนโลยี Web Service ทำให้เกิดส่วนเพิ่มเติมของ SOA ที่เรียกว่า “Contemporary SOA” หรือ SOA ร่วมสมัย

Contemporary SOA ถูกสร้างขึ้นโดยยึด “Primitive SOA” เป็นต้นแบบ ซึ่งเป็นผลจากอิทธิพลของวงการธุรกิจและความก้าวหน้าทางเทคโนโลยีทำให้เกิดแนวคิดใหม่ ๆ เพิ่มขึ้น อย่างไรก็ตาม ความต้องการเทคโนโลยีที่นำมา Implement ได้เปลี่ยนแปลงไป ดังนั้น จึงต้องมีการกำหนดขอบเขตของการพัฒนา Comtemporary SOA ให้อยู่บนรากฐานของ SOA แบบดั้งเดิม กล่าวคือ ให้อยู่ในทิศทางที่สัมพันธ์กับกลุ่มของ “คุณลักษณะ” หรือ “Characteristic” หมายถึง ลักษณะประจำตัว

ลักษณะเฉพาะ หรือลักษณะพิเศษ ดังนั้น คำว่า Common Characteristic หมายถึง ลักษณะเฉพาะที่ใช้เป็นแนวทางร่วมกันในการพัฒนา Service ของ SOA ซึ่งลักษณะเฉพาะที่สำคัญของ Contemporary SOA มีดังนี้

- ใช้ Platform ในการประมวลผลแบบ Service-Oriented
- ช่วยเพิ่มคุณภาพของ Service
- อยู่บนรากฐานของความมีอิสระในการทำงาน (Autonomy)
- อยู่บนมาตรฐานเปิด (Open Standard)
- สนับสนุนการทำงานบน Platform ของ Vendor ที่ต่างกัน
- สนับสนุนกลไกการค้นหา Service
- สนับสนุนการทำงานร่วมกัน (Interoperability) อย่างแท้จริง
- สนับสนุนการรวม (Federation) แอปพลิเคชั่นเข้าด้วยกัน
- สนับสนุนการนำส่วนต่าง ๆ มาประกอบกัน (Composability)
- สนับสนุนการนำกลับมาใช้ใหม่ (Reusability)
- สนับสนุนการขยายความสามารถใหม่ ๆ
- สนับสนุนให้ทุกส่วนขององค์กรมีความเป็นอิสระต่อกัน
- เพิ่มความคล่องตัวให้กับระบบขององค์กร
- เป็นการพัฒนามาจากเทคโนโลยีแบบเดิม

2.9 ประโยชน์ของ SOA

SOA มีประโยชน์ต่อองค์กรหลายรูปแบบขึ้นอยู่กับจุดประสงค์และวิธีการในการนำ SOA ไปใช้งาน และการสนับสนุนของ ผลิตภัณฑ์ รวมถึงเทคโนโลยีที่นำมาประยุกต์ใช้ด้วย

2.9.1 ระบบทำงานร่วมกันได้ดีขึ้น

SOA ทำให้สามารถสร้างระบบที่เกิดจากการนำ Service ต่าง ๆ มาทำงานร่วมกันได้ เป็นการสร้าง Solution อยู่บนพื้นฐานของการทำงานร่วมกันของ Service

SOA ที่ใช้เทคโนโลยี Web Service นั้น ไม่ยึดติดกับโครงสร้างของการสื่อสารแบบใดแบบหนึ่ง เมื่อนำคุณสมบัติดังกล่าวมาประยุกต์ใช้กับ Service Description และการรับส่งข้อมูลด้วย Message จะได้ผลลัพธ์คือทำให้เกิดการทำงานร่วมกันอย่างแท้จริง โดยการนำแอปพลิเคชั่นที่ทำงานอย่างอิสระต่อกันมาทำงานร่วมกัน

2.9.2 การสืบทอดเพื่อนำกลับมาใช้ใหม่

Service-Oriented สนับสนุนการออกแบบ Service ด้วยการสืบทอด (Inheritance) เพื่อนำกลับมาใช้ใหม่ ลักษณะดังกล่าวทำให้สามารถนำลอจิกต่าง ๆ ที่พัฒนาไว้แล้วกลับมาใช้ใหม่ได้ ช่วยลดระยะเวลาในการพัฒนาได้เป็นอย่างมาก และช่วยลดค่าใช้จ่ายในการพัฒนาระบบด้วย Service-Oriented ได้เป็นอย่างดี

2.9.3 ปรับปรุง Architecture และ Solution ได้ดียิ่งขึ้น

คุณลักษณะพื้นฐานของ Contemporary SOA เป็นรูปแบบสำคัญที่นำมาใช้เป็นแนวทางในการปรับปรุงสถาปัตยกรรมของ SOA ให้ดียิ่งขึ้น เช่น แนวคิดของความสามารถในการนำมาประกอบกัน (Composition) คือ หลักการที่เป็นส่วนประกอบหนึ่งของ SOA ด้วยการนำ Service ต่าง ๆ มารวมกันไว้ภายใน Service อื่น เพื่อใช้ทำงานในด้านต่าง ๆ

การนำ Contemporary SOA มาปรับปรุงสถาปัตยกรรมของ Service-Oriented Architecture สามารถนำไปใช้ประโยชน์ในการสร้างระบบต่าง ๆ ได้เป็นอย่างดี โดยเทคโนโลยีที่นำมาใช้เป็นเพียงส่วนประกอบหนึ่งของสถาปัตยกรรมเท่านั้น ประโยชน์ของการปรับปรุงสถาปัตยกรรม คือ ลดขั้นตอนการดำเนินงาน และทำให้ความต้องการรวมถึงปัญหาต่าง ๆ ของ Solution น้อยลง

2.9.4 เพิ่มอำนาจการลงทุน

ในปัจจุบันเทคโนโลยีของ Web Service ได้รับการยอมรับอย่างกว้างขวาง ทำให้เกิดการปรับตัวของกลไกทางการตลาด โดยสามารถนำระบบเดิมที่มีอยู่มาทำงานร่วมกับระบบใหม่ที่พัฒนาด้วย SOA ช่วยให้ระบบงานที่ถูกพัฒนาด้วยเทคโนโลยีที่ต่างกันสามารถทำงานร่วมกันได้โดยมีค่าใช้จ่ายต่ำ และทำให้ประสิทธิภาพการทำงานของระบบดีขึ้น แม้ว่าอาจจะมีปัญหาเรื่องความเสี่ยงในการจัดการระบบเดิม และปริมาณการใช้งานที่มากขึ้น รวมทั้งความพร้อมในการนำ Service-Oriented มาใช้สร้าง Solution ต่าง ๆ ก็ตาม

2.9.5 เพิ่มประสิทธิภาพในการสื่อสารข้อมูลด้วย XML

หลักการของ SOA ถูกสร้างอยู่บนพื้นฐานของ XML โดยมีวัตถุประสงค์เพื่อเพิ่มประสิทธิภาพในการสื่อสารข้อมูลของระบบ เนื่องจาก XML สามารถทำงานได้ทุก Platform รูปแบบมาตรฐานของข้อมูลที่ถูกกำหนดด้วย XML ช่วยลดความซับซ้อนต่าง ๆ ของระบบ

- เอกสาร XML และ XML Schema ภายใน SOAP Message ซึ่งถูกส่ง ให้กับแอปพลิเคชั่นต่าง ๆ ที่ต้องการติดต่อสื่อสารกัน จะแสดงถึงรูปแบบมาตรฐาน และชนิดของข้อมูลที่ใช้สำหรับติดต่อสื่อสารกัน ทำให้สามารถขยาย และปรับปรุงการติดต่อสื่อสารผ่านระบบเครือข่ายได้มากขึ้น

- XML อนุญาตให้ผู้พัฒนากำหนดแท็กได้ตามต้องการ ทำให้สามารถถอดความหมายของข้อมูลโดยนักออกแบบ นักวิเคราะห์ หรือผู้พัฒนาได้ง่ายยิ่งขึ้น ช่วยเพิ่มประสิทธิภาพให้กับข้อมูลที่อยู่ใน Message ทำให้ดูแลรักษา สืบค้น และทำความเข้าใจได้ง่ายขึ้น

2.9.6 โครงสร้างของการติดต่อสื่อสารที่มีความยืดหยุ่นสูง

Web Service ถูกพัฒนาโดยใช้โครงสร้างการติดต่อสื่อสารที่เป็นกลางของ SOA ทำให้สามารถใช้เป็นส่วนประกอบมาตรฐานของโครงสร้างระบบ เพื่อใช้ติดต่อสื่อสารกันภายใน หรือระหว่างแอปพลิเคชั่นได้ ช่วยลดค่าใช้จ่าย และทำให้องค์กรต่าง ๆ มีโครงสร้างของการติดต่อสื่อสารบนเทคโนโลยีเดียวกัน

2.9.7 เป็นวิธีขยายระบบที่มีประสิทธิภาพ

ข้อจำกัดของการนำเทคโนโลยีในรูปแบบที่ต่างกันมาใช้งานร่วมกันเพื่อตอบสนองต่อความต้องการของระบบอัตโนมัติในองค์กรธุรกิจถือเป็นปัญหาสำคัญ เนื่องจากมีค่าใช้จ่ายสูงและนำมาพัฒนาได้ยาก แม้ว่า SOA จะไม่สามารถแก้ปัญหาดังกล่าวได้หมด แต่ก็เป็นทางเลือกที่ช่วยให้การดำเนินงานของธุรกิจ และระบบสารสนเทศมีประสิทธิภาพมากขึ้น หัวใจสำคัญของ Service-Oriented คือ การสนับสนุนการขยายระบบอย่างมีประสิทธิภาพ เนื่องจาก SOA ได้รับการ พัฒนาอยู่บนโครงสร้างของการติดต่อสื่อสารที่เป็นกลาง โดยไม่ยึดติดกับวิธีการพัฒนา หรือ Platform ของ Middleware ใด ๆ โดยใช้วิธีรวบรวมกลุ่มของระบบอัตโนมัติ และแสดงให้ทราบผ่าน Service Interface เพื่อเป็นทางเลือกในการนำไป Implement ให้กับ Service ทำให้เกิดความยืดหยุ่น และสามารถขยายระบบได้อย่างมีประสิทธิภาพ

2.9.8 เพิ่มความคล่องตัว (Agility) ให้กับองค์กร

ประโยชน์สำคัญที่ได้จากการนำ SOA มาใช้กับระบบขององค์กร คือ การป้องกันผลกระทบจากวิวัฒนาการของเทคโนโลยีที่เปลี่ยนแปลงไปต่อระบบขององค์กร เนื่องจากในปัจจุบันเทคโนโลยีมีแนวโน้มเปลี่ยนโครงสร้างการพัฒนาให้อยู่ในรูปแบบของ Distributed Solution มากยิ่งขึ้น เช่น การนำกลับมาใช้ใหม่ (Reuse) และการทำงานร่วมกัน (Interoperability) เป็นต้น ซึ่งช่วยเพิ่มระดับความน่าเชื่อถือ และยกระดับองค์กร แต่การนำรูปแบบดังกล่าวมาใช้งานจะประสบผลสำเร็จได้ก็ต่อเมื่อได้รับการออกแบบ และกำหนดมาตรฐานอย่างเหมาะสม โดย SOA สามารถสร้างความสัมพันธ์แบบ Loosely Coupled ระหว่างองค์กร ที่ยอมให้แต่ละระบบขององค์กรมีการพัฒนาได้อย่างอิสระ และปรับตัวได้เหมาะสมกับการเปลี่ยนแปลง ทำให้เกิดความคล่องตัวในการพัฒนาระบบมากขึ้น

2.10 ข้อควรระวังสำหรับการนำ SOA มาใช้งาน

หัวข้อนี้จะพิจารณาถึงขอบเขตการเปลี่ยนแปลงเทคโนโลยี และการนำ SOA มาใช้ภายในองค์กรอย่างเต็มรูปแบบ ซึ่งปัญหาบางเรื่องอาจถูกมองข้ามไป ทำให้เกิดผลเสียจากการนำ Service-Oriented Archirtecture มาใช้งาน

2.10.1 การสร้าง SOA เหมือนกับ Distribute Architecture

อุปสรรคหนึ่งที่พบในการนำ SOA มาใช้คือการนำ Distribute Architecture ซึ่งเป็นสถาปัตยกรรมแบบเก่ามาใช้งานร่วมกับ Contemporary SOA ตัวอย่างปัญหาที่เกิดขึ้นประกอบด้วย

- การนำ RPC (Remote Procedure Call) มาใช้กับ Service Description เป็นการเพิ่มปริมาณของการแลกเปลี่ยน Message ย่อยเป็นจำนวนมาก ทำให้ระบบมีความซับซ้อนมากขึ้น
- ไม่สามารถนำเครื่องมือต่าง ๆ ที่จัดเตรียมโดยข้อกำหนดของ WS-* มาใช้งานได้
- การแยกขอบเขตหน้าที่ต่าง ๆ ภายใน Service ไม่ถูกต้อง
- การติดต่อสื่อสารบางส่วนต้องอยู่ในรูปแบบ Synchronous ทำให้เกิดข้อจำกัดในการรับส่งข้อมูล

การทำความเข้าใจความแตกต่างของหลักการ SOA กับสถาปัตยกรรมแบบเก่า เป็นหัวใจที่สำคัญที่ทำให้สามารถหลีกเลี่ยงข้อผิดพลาด

2.10.2 SOA ยังไม่มีมาตรฐานการออกแบบที่แน่นอน

สิ่งสำคัญในการรวมแอปพลิเคชั่นต่าง ๆ เข้าด้วยกัน คือ ข้อกำหนด หรือมาตรฐาน เนื่องจากถ้าแต่ละโครงการมีการออกแบบแอปพลิเคชั่นในรูปแบบที่แตกต่างกัน เมื่อต้องการนำมารวมกันจะเกิดความยุ่งยาก เสียค่าใช้จ่ายสูงและระบบเสียหายง่าย แม้จุดเด่นของ SOA คือ ความสามารถในการรวมระบบต่าง ๆ ที่มีสภาพแวดล้อมต่างกันให้ทำงานร่วมกันได้ก็ตาม แต่โดยทั่วไปการรวมระบบลักษณะดังกล่าวถือเป็นเรื่องยาก

สมมติว่ามีโครงการหนึ่งที่พัฒนาแอปพลิเคชั่นในรูปแบบของ Service-Oriented ซึ่งมีความแตกต่างอย่างมากกับแอปพลิเคชั่นอื่นในโครงการที่ต้องทำงานร่วมกัน ลักษณะข้างต้นอาจนำไปสู่ปัญหาต่าง ๆ ดังต่อไปนี้

- การเข้ากันไม่ได้ของข้อมูลอันเนื่องมาจากการแยก Schemas ซึ่งใช้แทนข้อมูลชนิดเดียวกัน
- รูปแบบของ Service Description ที่ไม่แน่นอนอาจทำให้ลักษณะของส่วนที่ใช้ติดต่อกับ Service รวมทั้งรายละเอียดของ Service เปลี่ยนแปลงไป
- รองรับความสามารถเพิ่มเติม (Extension) แตกต่างกัน หรือ Extension ถูก Implement ด้วยวิธีการที่ต่างกัน

- SOA สนับสนุนการพัฒนาสภาพแวดล้อมที่เป็นอิสระภายในแต่ละแอปพลิเคชั่น โดยมีข้อจำกัด คือ ต้องมีมาตรฐานเพื่อทำให้เกิดความมั่นใจว่าการออกแบบ และการโต้ตอบต่าง ๆ ของ Service ที่ถูกรวบรวมไว้ในลอจิกนั้นสอดคล้องกัน ตัวอย่างของมาตรฐานที่ใช้ในการออกแบบ Service Description ได้แก่ WSDL

2.10.3 ควรวางแผนงานการเปลี่ยนแปลง (Transition Plan)

Transition Plan หรือแผนงานการเปลี่ยนแปลง ทำให้สามารถควบคุมรูปแบบ และผลกระทบจากการนำ Service-Oriented มาใช้กับระบบได้ การสร้าง Transition Plan ช่วยยับยั้งปัญหาต่าง ๆ ที่เกิดจากการนำ SOA มาใช้งาน โดยแต่ละแผนการจะถูกกำหนด โดยยึดความต้องการข้อบังคับ และเป้าหมายขององค์กรเป็นหลัก ตัวอย่างของการวางแผนงานการเปลี่ยนแปลง

วิเคราะห์ผลกระทบที่อาจเกิดขึ้นเมื่อมีการเปลี่ยนแปลงไปสู่ SOA จะต้องพิจารณาบนพื้นฐานของทรัพยากร กระบวนการ มาตรฐาน และเทคโนโลยีที่องค์กรมีอยู่

วิเคราะห์ผลตอบแทนที่จะได้รับในอนาคตจาก Web Service และการสนับสนุนเทคโนโลยีนี้

วิเคราะห์ถึงความเปลี่ยนแปลงของเทคโนโลยีต่าง ๆ ที่อาจส่งผลกระทบต่อระบบในอนาคตรวมถึงจัดเตรียมช่องทางในการเพิ่มเติมเทคโนโลยีใหม่ ๆ ให้กับระบบ

2.10.4 ไม่ได้เริ่มต้นการพัฒนา SOA จากมาตรฐาน XML

XML เป็นมาตรฐานที่องค์กรต่าง ๆ ได้ร่วมกันพัฒนาเพื่อใช้สำหรับแทนข้อมูล และเป็นมาตรฐานหลักที่ช่วยผลักดันให้สามารถนำหลักการของ SOA มาใช้งานได้จริง โดยใช้ในการสร้างข้อกำหนดต่าง ๆ ของ Web Service

สิ่งที่น่าสนใจในหลักการของ SOA คือจะขนส่งข้อมูลระหว่าง Service อย่างไร และจะกำหนดโครงสร้างของข้อมูลที่ใช้รับส่งระหว่าง Service ให้มีรูปแบบเหมือนกันได้อย่างไร โดยปัญหาเหล่านี้เป็นสิ่งที่ไม่ควรมองข้าม เพราะอาจส่งผลกระทบต่อคุณภาพของการประมวลผลข้อมูล เช่น ในเวลาที่ต่างกันข้อมูลที่เหมือนกันไม่จำเป็นต้องใช้งานได้เหมือนกัน หรือ การประมวลผลข้อมูลเป็นจำนวนมากอาจทำให้เกิดข้อผิดพลาดในขณะก่อนหรือหลังที่ Service รับส่ง Message ได้ เป็นต้น โดยข้อกำหนดต่าง ๆ ที่มีเทคโนโลยี XML เป็นพื้นฐาน ถูกใช้เพื่อแทนข้อมูลที่ส่งผ่านไปยังสภาพแวดล้อมของแอปพลิเคชั่นทั้งที่อยู่ภายใน และระหว่าง Service ทำให้เกิดการทำงานร่วมกัน

2.10.5 ไม่ได้ออกแบบระบบให้รองรับการทำงานใหม่ ๆ ในอนาคต

โดยทั่วไปเมื่อเริ่มต้นการพัฒนา Solution ด้วยหลักการของ Service-Oriented ระบบจะมีขนาดเล็กโดยมีหน้าที่การทำงานที่เรียบง่ายไม่ซับซ้อน แต่เมื่อขอบเขตความต้องการเพิ่มขึ้น ทำให้ต้องเพิ่มหน้าที่การทำงานใหม่ ๆ ส่งผลให้ระบบมีขนาดใหญ่ขึ้น แต่ไม่ได้มีการออกแบบสภาพแวดล้อมของระบบให้สามารถรองรับหน้าที่การทำงานใหม่ ๆ ดังนั้นก่อนการพัฒนาควรวางแผนเพื่อรองรับการขยายตัวของระบบ และทดสอบประสิทธิภาพของระบบว่าเพียงพอต่อความต้องการในอนาคตหรือไม่

ความน่าเชื่อถือของ Contemporary SOA บน Web Service อยู่ที่การแทนข้อมูลด้วย XML ตัวอย่างเช่น การวัดความปลอดภัยของ Web Service เช่น การเข้ารหัส (Encryption) การลงลายเซ็นอิเล็คทรอนิกส์ (Digital Signature) เป็นต้น สิ่งที่ต้องระวังในการสร้าง Service-Oriented คือความเข้าใจในเรื่องความต้องการประสิทธิภาพของ Solution และข้อจำกัดในด้านประสิทธิภาพของโครงสร้างในอนาคต โดยพิจารณาจากการทดสอบ ดังนี้

ทดสอบประสิทธิภาพการประมวลผล Message บนสภาพแวดล้อมของตนเองก่อนจะเริ่มดำเนินการลงทุนเพื่อพัฒนา Web Service เพื่อนำข้อมูลที่ได้ไปพิจารณาความคุ้มค่าในการลงทุน และโอกาสในการขยายระบบ

ทดสอบตัวประมวลผล (Processor) ของผู้ผลิตซอฟต์แวร์ เช่น XML XSLT หรือ SOAP ที่ต้องการใช้งาน

ค้นหาตัวประมวลผล หรือเทคโนโลยีอื่น ๆ ที่จำเป็นไว้ เพื่อเป็นทางเลือกสำหรับนำมาใช้ในการพัฒนาระบบ

2.10.6 ความไม่เข้าใจในเรื่องความปลอดภัยของ Web Service

การวิเคราะห์การเติบโต และวิวัฒนาการใหม่ ๆ ของเทคโนโลยีด้านความปลอดภัยที่เกี่ยวของกับ Web Service ช่วยให้ผู้พัฒนาและผู้ออกแบบสามารถเข้าใจโครงสร้างของ เทคโนโลยีที่จะนำมาใช้ในการสร้างระบบความปลอดภัยได้ดียิ่งขึ้น รวมทั้งเลือกเทคโนโลยีที่สนับสนุนข้อกำหนดของ WS-Security ซึ่งเป็นแบบจำลองด้านความปลอดภัย (Security Model) เมื่อองค์กรตัดสินใจนำ Security Model มา Implement ควรทำความเข้าใจเกี่ยวกับโครงสร้างและเงื่อนไขต่าง ๆ เพื่อปรับให้เข้ากับสถาปัตยกรรมและแอปพลิเคชั่นที่มีอยู่ เนื่องจากการนำเทคโนโลยี หรือ Platform ที่ไม่สนับสนุน WS-Security มาใช้ อาจต้องเสียค่าใช้จ่ายในการเปลี่ยนแปลงระบบค่อนข้างสูง

2.11 อนาคตของ SOA

SOA เป็นสถาปัตยกรรมที่มีความโดดเด่น ทั้งในเรื่องของประสิทธิภาพ และความยืดหยุ่นในการทำงาน ดังนั้น SOA จึงได้รับความสนใจจากองค์กรต่าง ๆ มากมาย เนื่องจากซอฟต์แวร์ในรูปแบบของ SOA สามารถนำกลับมาใช้ได้ใหม่ (Reuse) ทำให้องค์กรอื่น ๆ สามารถใช้งาน Service ร่วมกันได้ ซึ่งลักษณะดังกล่าวเป็นการสร้างความสัมพันธ์ และประสานการทำงานระหว่างองค์กรธุรกิจ ทำให้องค์กรไม่ต้องทำงานเพียงลำพัง นอกจากนี้ยังช่วยเพิ่มความรวดเร็วในการสร้างบริการอีกด้วย ด้วยการนำ Service ต่าง ๆ มาประกอบกัน และสามารถเพิ่มเติมหน้าที่การทำงานใหม่ ๆ ได้อีกด้วย นอกจากนี้ยังปรับเปลี่ยนแก้ไขแอปพลิเคชั่นได้อย่างรวดเร็ว ยืดหยุ่น และไม่ส่งผลกระทบต่อ Service อื่น โดยใช้วิธีถอดประกอบบริการที่ต้องการแก้ไข ดังนั้น SOA จึงเป็นแนวทางการพัฒนาระบบที่สอดคล้องกับความต้องการขององค์กรต่าง ๆ ทั้งในปัจจุบันและอนาคต เพราะจะช่วยลดต้นทุนและเพิ่ม ความสามารถในการแข่งขันทางการตลาดได้อย่างมีประสิทธิภาพ

แม้ว่าการพัฒนาซอฟต์แวร์ตามแนวทางของ SOA และเทคโนโลยี Web Service จะมีประสิทธิภาพสูง แต่การนำเอา SOA ไปใช้ประโยชน์กับระบบขององค์กร จำเป็นต้องศึกษาโครงสร้างการทำงานต่าง ๆ ของ SOA และ Web Service อย่างละเอียด เพื่อนำข้อมูลดังกล่าวไปวิเคราะห์และออกแบบฟังก์ชั่นบริการต่าง ๆ ให้สามารถทำงานร่วมกันได้อย่างมีประสิทธิภาพ ซึ่งต้องอาศัยความรู้ความเข้าใจทั้งในเรื่องของวิศวกรรมซอฟต์แวร์ SOA และเทคโนโลยี Web Service อย่างเป็นระบบ

2.12 ลักษณะทั่วไปของการบัญชี

จะเห็นได้ว่าสังคมปัจจุบัน บุคคลในทุกอาชีพมีส่วนเกี่ยวข้องกับเหตุการณ์ในการบริหารและข้อมูลทางการเงินและมีความผูกพันทางด้านการเงิน ทั้งในชีวิตส่วนตัว และการประกอบ หน้าที่งานในอาชีพของตัวเอง การบัญชีจึงเข้ามามีบทบาทสำคัญในสังคมธุรกิจ ดังนั้นหน้าที่สำคัญ ทางบัญชีคือ การจัดเก็บ รวบรวมข้อมูลอย่างเป็นระบบ เพื่อให้บุคคลที่สนใจจะนำข้อมูลไปใช้ประโยชน์ต่อไป

คำจำกัดความของคำว่า “การบัญชี”

โดยทั่ว ๆ ไป จุดมุ่งหมายของการบันทึกรายการทางบัญชี ก็เพื่อจัดให้มีข้อมูลที่เกี่ยวข้องกับทรัพยากรที่ธุรกิจเป็นเจ้าของ หนี้สินที่กิจการได้ทำไว้ และทุนของเจ้าของกิจการ ที่นำมาลงตลอดจนผลการดำเนินงานของกิจการ

สมาคมบัญชี และผู้ตรวจสอบบัญชีรับอนุญาตในสหรัฐอเมริกา คือ The America Institute of Certified Public Accountants (AICPA) ได้ให้คำจำกัดความของคำว่า “การบัญชี” ไว้ดังนี้

“การบัญชี” เป็นศิลปะของการจดบันทึกรายการหรือเหตุการณ์ที่เกี่ยวข้องกับการเงินไว้ในรูปแบบของเงินตรา จัดแยกหมวดหมู่ของรายการที่บันทึก สรุปผลและวิเคราะห์ความหมาย ของรายการที่จดบันทึกไว้ โดยจัดทำในรูปแบบของรายการการเงิน

Accounting is the art of recording, classifying and summarizing in a significant manners and in terms of money, transactions and events witch are, in part at least, of a financial character, and interpreting the results there of

จากคำจำกัดความนี้จะสามารถอธิบายความหมายของแต่ละขั้นตอนได้ดังนี้

1. การจดบันทึกรายการ (Recording) รายการทางบัญชี (Transactions) ต้องเป็นรายการการค้าที่เกิดขึ้นแล้ว การบันทึกรายการอาจทำโดยใช้ปากกาบันทึกรายการ ในสมุดบัญชี ตามปกติ แต่ในปัจจุบันกิจการขนาดใหญ่ ต้องการความรวดเร็วในการบันทึกรายการ อาจบันทึกรายการนั้นด้วยโปรแกรมคอมพิวเตอร์เข้ามาช่วย

2. นักบัญชีเงินตรา เป็นหน่วยในการบันทึกรายการทางบัญชี แต่ถือว่าเงินตราแต่ละหน่วยมีค่าคงที่ โดยจะไม่มีการปรับปรุงจำนวนเงินที่ได้บันทึกไว้ แม้ค่าของเงินจะเปลี่ยนไป

3. การจัดหมวดหมู่ของรายการ เมื่อได้บันทึกรายการในสมุดบัญชีแล้ว จะต้องมีการจัดหมวดหมู่ของรายการ โดยแยกเป็นประเภทสินทรัพย์ หนี้สิน ทุน รายจ่าย รายได้ เมื่อสิ้นระยะเวลาหนึ่ง ก็จะนำรายการดังกล่าวที่ได้แยกไว้แล้วมาสรุปผลการดำเนินงาน และฐานะของกิจการโดยจัดทำงบการเงินซึ่งประกอบด้วย งบกำไรขาดทุน และงบดุล

4. วิเคราะห์งบการเงินเมื่อจัดทำงบการเงินเสร็จแล้ว ก็นำงบการเงินนั้นมา วิเคราะห์ให้ผู้ที่สนใจทราบ การวิเคราะห์นั้นอาจทำเป็นเปอร์เซ็นต์ หรืออัตราส่วนก็ได้ เปรียบเทียบผลการดำเนินงานในงวดบัญชีต่าง ๆ กัน เป็นต้น

งบการเงิน (Financial Statement)

งบการเงิน คือ รายงานทางการเงินที่กิจการจัดทำขึ้นมาเพื่อแสดงผลการดำเนินงานของกิจการ ตลอดจนฐานะของกิจการในวันใดวันหนึ่ง ซึ่งบุคคลที่เกี่ยวข้องในธุรกิจให้ความสนใจ เพราะงบนี้แสดงให้เห็นความสามารถในการดำเนินของผู้บริหารกิจการว่าดำเนินมาแล้วในช่วงระยะเวลาหนึ่ง ได้ผลเป็นอย่างไร กำไร เสมอตัว หรือขาดทุน นอกจากนี้ยังแสดงให้เห็นฐานะของกิจการในวันใดวันหนึ่งว่า กิจการมีฐานะเป็นอย่างไร มีสินทรัพย์ หนี้สิน และขาดทุนเป็นสัดส่วนอย่างไร

งบการเงินประกอบด้วย

- งบกำไรขาดทุน (Income Statement)
- งบดุล (Balance Sheet)

2.12.1 ขั้นตอนการทำบัญชี

เนื่องด้วยงานวิจัยนี้เป็นการประยุกต์นำเอาการทำงานงานทางด้านบัญชีมาประยุกต์กับสิ่งแวดล้อมแบบ SOA บนเทคโนโลยี Web Service ดังนั้น จะนำเอามาใช้เฉพาะงานพื้นฐานทางการบัญชีเท่านั้น โดยมีขั้นตอนการทำงานดังภาพ

- การใช้โปรแกรมสำเร็จรูป เช่น Easy-ACC หรือ Accounting Express เป็นต้น ผู้ที่ใช้โปรแกรมสำเร็จรูปเป็นประจำคือ สำนักงานบัญชี หรือองค์กรทีมีขนาดย่อม เช่น SME เป็นต้น เหตุผลที่ใช้เนื่องจากราคาไม่สูงมาก สามารถทำงานพื้นฐานทางบัญชีได้ทั้งหมด
- การเขียนโปรแกรมขึ้นมาใหม่โดยองค์กรที่ใช้โปรแกรมรูปแบบนี้ ส่วนใหญ่มีขนาดขององค์กรใหญ่ และมีหลายส่วนงาน รวมไปถึงการแลกเปลี่ยน ข้อมูลระหว่างองค์กร บางครั้งอาจมี Solution เก่าอยู่แล้วในองค์กร แต่มีต้องการขยับขยาย หรือเพิ่มความเข้ากันได้ผ่านทางระบบโปรแกรมมากขึ้น เลยต้องมีการออกแบบใหม่ทั้งระบบ รวมไปถึงระบบบัญชี ซึ่งมีความจำเป็นต้องมีในทุก ๆ องค์กรอยู่แล้ว เนื่องด้วยมีความจำเป็นต้องวิเคราะห์ระบบใหม่ จนไปถึงทดสอบระบบ ทำให้การใช้โปรแกรมรูปแบบนี้มีค่าใช้จ่ายที่สูง

การออกแบบงานวิจัยนี้ ได้เขียนเพื่อลดระยะเวลาการออกเขียนโปรแกรมของโปรแกรมเมอร์ ที่เมื่อมีการเขียนโปรแกรมซ้ำ ๆ กัน สามารถนำเอาส่วนที่เขียนเอาไว้แล้ว นำกลับมาใช้ใหม่ได้ง่ายมากขึ้น กอปรกับเพื่อรองรับการเขียนโปรแกรมแบบไม่อิง Platform จึงได้มีการนำเอาเทคโนโลยี Web Service มาประยุกต์ใช้ในงานวิจัยนี้

บทที่ 3
วิธีดำเนินงานวิจัย

งานวิจัยนี้มีจุดประสงค์ เพื่อพัฒนาระบบการใช้บริการทางเว็บสำหรับงานบัญชี เพื่อให้นักพัฒนาสามารถนำเอาแนวคิดของ SOA (Service-Oriented Architecture) และเทคโนโลยีเว็บเซอร์วิสมาประยุกต์ใช้กับงานทางบัญชีที่ทุก ๆ องค์กรมีความจำเป็นต้องมี และยังคงเป็นแนวคิด ที่ลดระยะเวลาการพัฒนาให้กับนักพัฒนาอีกด้วย

3.1 ระเบียบวิธีวิจัย

3.1.1 ศึกษาและค้นคว้างานวิจัยที่เกี่ยวข้อง

3.1.2 ศึกษาระบบงานบัญชี ศึกษาการทำบัญชีเบื้องต้น โดยการคำนวณด้วยมือ (หลักการบัญชี 1, เชาวลีย์ พงศ์ผาติโรจน์, ดร.วรศักดิ์ ทุมมานนท์, 2547)

1) ศึกษาการทำบัญชีผ่านโปรแกรม Easy-Acc Accounting System (www.businesssoft.com, 2008)

2) ศึกษาการใช้งานและซอร์สโค้ดของระบบบัญชีเทศบาลนครนครราชสีมา วิเคราะห์ปัญหาของการพัฒนาระบบบัญชีขึ้นมาเอง โดยใช้ระบบบัญชีของเทศบาลนครนครราชสีมาเป็นหลัก

3.1.3 ศึกษาแนวคิดของสถาปัตยกรรมเชิงบริการ (SOA) และเทคโนโลยีที่นำมาใช้

1) ศึกษาถึงธรรมชาติของ SOA และวัตถุประสงค์ของแนวคิด

2) ศึกษาแนวทางการเขียน Web Service บนภาษา PHP โดยใช้ NuSoap

3.1.4 กำหนดปัญหา (Problem Definition) เป็นการทำระบบงานบัญชี โดยหลังจากวางแผนและพัฒนา จะให้นักบัญชีทดลองใช้โปรแกรม เพื่อตรวจสอบหาข้อผิดพลาด

3.1.5 ศึกษาความเป็นไปได้ (Feasibility Study) ศึกษาความเป็นไปได้ในการนำเอาแนวคิดสถาปัตยกรรมเชิงบริการ มาประยุกต์ใช้

3.1.6 การวิเคราะห์ระบบ (System Analysis) ทดสอบการทำงานของบัญชีด้วยการทำด้วยมือ แล้วศึกษาการใช้โปรแกรมบัญชีทั่วไป จากนั้นนำวิธีการทำงานมาแบ่งส่วนของงานออกเป็น Process ต่าง ๆ จากนั้นเอาข้อมูลมาวิเคราะห์ ออกแบบและพัฒนาระบบที่เป็นมาตรฐานตามหลักการของ Software Engineering

3.1.7 การออกแบบระบบ (System Design) นำผลจากการศึกษารูปแบบวิธีการทำงานที่ได้มาทำการออกแบบระบบในส่วนต่าง ๆ คือ

1) การออกแบบผังระบบ (System Flowchart)
2) การออกแบบข้อมูลนำเข้า (Input Design)
3) การออกแบบส่วนของบริการ (Service Design)
4) การออกแบบการรายงานผล (Output Design)

3.1.8 การทดสอบระบบ (System Testing) โดยการทดสอบจะแบ่งเป็น 2 ส่วนดังนี้

1) ทดสอบการทำงานบนเครื่องที่ใช้ในการพัฒนา (Stand Alone Test)
2) ทดสอบการทำงานบนเครื่องเซิร์ฟเวอร์ (Network Test)

3.1.9 สรุปผลการทำงาน

จากการศึกษาค้นคว้ารูปแบบการทำงานของงานบัญชีในแต่ละองค์กร พบว่าการทำงานบัญชีภายใต้องค์กรที่มีขนาดเล็ก จะพบปัญหาที่เกิดจากตัวของบุคลากรเป็นหลัก ไม่ว่าจะเป็น การบันทึกข้อมูลผิดพลาด การจัดทำข้อมูลล่าช้า เป็นต้น ส่วนองค์กรที่มีขนาดใหญ่ปัญหาที่เกิดส่วนใหญ่ มักเป็นปัญหาของการส่งข้อมูล ไม่ว่าจะเป็นรูปแบบผิดพลาด หรือประเภทของข้อมูลคนละประเภท ทำให้ส่วนใหญ่มักจัดการปัญหาแบบรายครั้ง กล่าวคือมีปัญหาอย่างไรก็แก้ไปอย่างนั้น ไม่ได้แก้ที่ต้นเหตุ หรือตัวของระบบ ปัญหาคือ หากองค์กรมีขนาดใหญ่ และมีองค์กรเล็ก ๆ อยู่มากมาย ระบบยังคงใช้รูปแบบดั้งเดิม คือ มีการแบ่งงานต่าง ๆ ออกเป็นโมดุล เช่น ฝ่ายบุคคล ฝ่ายเทคโนโลยีสารสนเทศ ฝ่ายการเงิน ฝ่ายบัญชี และอีกมากมาย การจัดการข้อมูลต่าง ๆ มักจะเขียนระบบหลักขึ้นมาแล้วเขียนระบบย่อย ๆ หรือโมดุลต่าง ๆ เข้ามาติดตั้งรูปแบบนี้ ได้มีใช้เป็นต้นแบบในการพัฒนาเป็นระบบ CMS (Content Menagement System) ในปัจจุบัน จุดเด่นคือ มีการทำงานเป็นระบบอย่างมาก แยกการปรับปรุงแก้ไขได้ ส่วนข้อจำกัดคือ ระบบมักอยู่ที่ส่วนกลางขององค์กร และให้องค์กรย่อยเข้ามาเรียกใช้ระบบกันผ่านทางเครือข่ายรูปแบบต่าง ๆ ไม่ว่าจะเป็น Ethernet หรือ Internet การทำงานแบบนี้ หากใช้ข้อมูลอยู่เพียงภายในองค์กร ก็ถือว่าสามารถทำได้ แต่ในยุคปัจจุบัน การที่หลาย ๆ องค์การเป็นพันธมิตรทางด้วยธุรกิจกัน ทำให้ต้องมีการแลกเปลี่ยนข้อมูลระหว่าง 2 องค์กรกันอยู่เสมอ ปัญหาคือเรื่องของเทคโนโลยีที่ใช้ เนื่องจากเป็นองค์กรที่ไม่ได้เกิดขึ้นมาพร้อมกัน ทำให้โอกาสที่จะใช้ เทคโนโลยีที่ต่างกันเป็นไปได้สูง เช่น องค์กร A ได้ใช้ Solution การบริหารภายในองค์กรของ Microsoft เป็นหลัก ส่วนองค์กร B เป็นองค์กรที่ไม่แสวงหากำไรจึงใช้ Linux เป็นหลัก จากนวัตกรรมที่ต่างกัน ก่อให้เกิดปัญหาของการแลกเปลี่ยนข้อมูลขึ้นมามากมาย ส่วนใหญ่มักจะหา Solution ส่วนกลาง แล้วปรับทั้ง 2 องค์กรเข้ามาใช้ร่วมกัน แต่นั่นก่อให้เกิดการลงทุนที่สูงมาก เนื่องจากต้องพัฒนาขึ้นมาใหม่ทั้งหมด เมื่อลงมามองที่ระบบงานบัญชี หากมี

การแลกเปลี่ยนทางข้อมูลกัน ก็จะประสบปัญหาของชนิดของข้อมูล และในมุมมองของ นักพัฒนาก็ต้องมีการเขียนซ้ำอยู่บ่อยครั้ง เพราะระบบนี้เป็นระบบที่ทุก ๆ องค์กรต้องใช้ แต่มีการปรับรูปแบบองค์กรอยู่ตลอดเวลาในยุคปัจจุบัน

จากปัญหาดังกล่าวงานวิจัยชิ้นนี้จึงได้พัฒนาการให้บริการทางเว็บสำหรับระบบบัญชี (Web Service for Accounting System)

3.2 การกำหนดปัญหา (Problem Definition)

ได้ลองศึกษาการบัญชีจากแบบเรียน และจากการเอาข้อมูลทางบัญชี มาทดลองปฏิบัติ จากนั้นทดสอบกับโปรแกรมทางบัญชีสำเร็จรูป ไม่ว่าจะเป็น Windows Application หรือ Web Applicantion เพื่อดูรูปแบบการทำงาน โครงสร้างของระบบ การแบ่งส่วนของการทำงานว่ามีการแบ่งอย่างไร โครงสร้างเป็นแบบไหน มีลักษณะการทำงานแบบไหน มีปัญหาอย่างไร และข้อจำกัดการใช้งานเป็นอย่างไร ทั้งนี้เพื่อนำข้อมูลที่ได้มาวิเคราะห์

จากการวิเคราะห์ปัญหาพบว่ารูปแบบของโปรแกรมสำเร็จรูปในการบัญชีมีข้อจำกัดอยู่คือ ไม่สามารถปรับแต่งตามความต้องการได้มากเท่าที่ควร ส่วนโปรแกรมที่มีประสิทธิภาพจะมีราคาค่อนข้างสูงตามไปด้วย และทุก ๆ โปรแกรมที่เป็น Windows Application มักจะต้องจ่ายเพิ่ม เมื่อมีการติดตั้งหลายเครื่อง ส่วนโปรแกรมที่เป็น Web Application ที่มีคุณภาพ มักจะ Bundle มาพร้อม ๆ กับ Solution อื่น ๆ ด้วย ทำให้ราคาสูง หากมีการติดตั้งระบบบัญชีในองค์กร ที่มีสาขามากมาย ค่า Licence เลยต้องสูงเป็นเงาตามตัว เช่น ธุรกิจเฟรนชายส์ ทุกสาขาต้องส่งข้อมูลทางการเงินให้กับบริษัทแม่ทุกเดือน บางแห่งเป็นรายวัน แล้วทุก ๆ ที่ก็มีความจำเป็นที่ต้องมีโปรแกรมบัญชีเอาไว้ในแต่ละสาขาของตัวเองอีกด้วย ยกตัวอย่าง 7-11 มีทั้งหมด 3,000 สาขาทั่วไทย หากมีการซื้อระบบบัญชีสำเร็จรูปมาต้องจ่ายค่า Licence เยอะเป็น 3,000 เท่า ดังนั้น โปรแกรมบัญชีสำเร็จรูป เมื่อมีการนำไปใช้จริง มักจะมีปัญหากับการลงทุนเป็นหลัก ทางออกอีกทาง คือ การเขียนระบบการบัญชีขึ้นมาใช้งานเอง โดยอาจมีการจ้างนักพัฒนามาพัฒนาระบบให้ ข้อดีของการพัฒนาระบบขึ้นมาเองคือสามารถกำหนดได้ว่าต้องการรูปแบบโปรแกรมเป็นแบบไหน รองรับอะไรได้บ้าง จากการที่ผู้วิจัยได้ทำการสอบถามกับองค์กรต่าง ๆ พบว่าองค์กรขนาดกลางไปถึงใหญ่ จะนิยมใช้ระบบบัญชีที่พัฒนาขึ้นมาเฉพาะองค์กรมากกว่าใช้โปรแกรมสำเร็จรูป ส่วนโปรแกรมสำเร็จรูปมักพบที่สำนักงานบัญชี หรือองค์กรขนาดย่อม หรือ SMEs (Small and Medium Enterprises) เพราะสามารถดูแลการจัดกรได้อย่างทั่วถึง

ดังนั้น แนวโน้มการใช้งานระบบโปรแกรมทางบัญชี มีโอกาสไปทางการเขียนขึ้นมาใหม่มากกว่า เพราะแต่ละองค์กรมักต้องการโปรแกรมที่เข้ากับพฤติกรรมการใช้งานของบุคลากรประจำ

ส่วนงานนั้น ๆ ในมุมของนักพัฒนาพบว่าการเขียนโปรแกรมทางบัญชีมีรูปแบบเดิมตลอด จะต่างกันก็เพียงรูปแบบหน้าตา (Theme) และการแสดงผลเท่านั้น แต่การคำนวนเหมือนเดิมส่งผลให้การเขียนโปรแกรมขึ้นมาใหม่ก็มีความจำเป็นต้องเขียนอัลกอลิทึ่มขึ้นมาใหม่ด้วย หรือหากเขียนเป็นฟังก์ชั่นเก็บไว้ ผู้ที่เข้าใจก็จะเป็นโปรแกรมเมอร์ที่เขียนเพียงคนเดียว ไม่สามารถส่งงานให้คนอื่นพัฒนาต่อได้ง่าย

จากปัญหาดังกล่าว ผู้วิจัยได้ทำการศึกษาแนวคิดการสถาปัตยกรรมเชิงบริการ หรือ SOA (Service-Oriented Architecture) เนื่องจากเป็นยุคที่ต่อเนื่องจากการเขียนโปรแกรมเชิงวัตถุ (Object-Oriented Programming) โดยสามารถสรุปแนวคิดการปรับมาใช้งานจาก OOP มายัง SOA คือ อนาคตการเขียนโปรแกรมจะไม่จำกัดอยู่ที่การเรียกใช้งานฟังก์ชั่นที่อยู่ใน Class ต่าง ๆ ใน Libraly แล้ว แต่การเรียกใช้งานฟังก์ชั่นในยุดต่อไปจะไม่มีพรมแดน และไม่มีข้อจำกัด เพราะ SOA จะมองเป็น Class เป็น Service ที่มีอยู่มากมายในอินเตอร์เน็ต มีทั้งแจกฟรี และทำการค้า (Commercial) อีกอย่างแนวคิดของ SOA จะใช้เทคโนโลยีที่เป็นสื่อกลางคือ Web Service ในการติดต่อสื่อสาร และพัฒนา ทำให้ลดข้อจำกัดเรื่องเทคโนโลยีที่ใช้พัฒนา เพราะ Web Service สามารถเขียนได้หลายภาษาแล้วในปัจจุบัน และการเรียกใช้งานก็สามารถใช้ภาษาอะไรก็ได้ในการเรียกใช้ เพราะ Web Service ใช้ภาษา XML ในการติดต่อสื่อสาร ซึ่งเป็นภาษาที่เป็นมาตรฐานที่สุดภาษาหนึ่ง จากจุดเด่นต่าง ๆ เหล่านี้ ทำให้ผู้วิจัยมีแนวความคิดเอา SOA มาประยุกต์กับงานบัญชี เนื่องจากงานบัญชีเป็นงานที่มีมาตรฐานเหมือนกันทั่วโลก และการคำนวณทางบัญชีนั้น มักประสบปัญหากับโปรแกรมเมอร์เสมอ ๆ เพราะจากการคำนวณที่ซับซ้อนนั่นเอง จึงได้สร้างการให้บริการทางเว็บสำหรับระบบบัญชี (Web Service for Accounting System) เพื่ออำนวยความสะดวกแก่โปรแกรมเมอร์รุ่นถัดไป ก็สามารถเรียกใช้ส่วนที่เป็น Service ได้เลย ไม่ต้องเขียนซ้ำ และยังลดความเสี่ยงจากการคำนวณผิดของโปรแกรมเมอร์ เนื่องจากมีคนออกแบบส่วนของการคำนวณเอาไว้แล้ว และยังสามารถนำแนวคิดไปศึกษาเพื่อพัฒนาต่อได้อีก

เพื่อให้ระบบนี้มีรูปแบบการทำงานให้ถูกต้อง จึงต้องมีกำหนดข้อมูลตัวอย่างขึ้น เพื่อนำ มาทดสอบกับตัวของโปรแกรม จากนั้นเอาผลการทำงานไปวิเคราะห์ เพื่อพัฒนาให้เกิดความเป็นมาตรฐานมากที่สุด ตามหลักการของ Software Engineering

3.3 การศึกษาความเป็นไปได้ (Feasibily Study)

ในส่วนนี้ผู้วิจัยได้ศึกษาความเป็นไปได้ของการพัฒนาระบบการให้บริการทางเว็บสำหรับระบบบัญชี ซึ่งมีปัจจัยในการพิจารณาดังนี้

3.3.1 ความเป็นไปได้ทางเทคนิค (Technical Feasibility)

ผู้วิจัยได้ตัดสินใจใช้ภาษา PHP ในการพัฒนาระบบ เนื่องจากเป็นภาษาสคริปที่ทำงานบนเว็บบราวเซอร์ และยังคงเป็นภาษาที่ได้รับความนิยมสูงมาก ในการพัฒนาเว็บไซต์ในปัจจุบันมีการเขียนโค้ดที่ยืดหยุ่น อีกทั้งยังเป็นโอเพนซอร์สอีกด้วย สาเหตุที่ไม่ได้ใช้ .NET ในการพัฒนาเนื่องจากไม่ว่ารูปแบบการเขียนโปรแกรมจะง่ายเท่าไร แต่ระบบการติดตั้งที่ค่อนข้างจะยุ่งยาก และไม่สามารถของรับระบบปฏิบัติการอื่น ๆ ได้

ส่วนที่ใช้ในการพัฒนา Web Service ผู้วิจัยได้เลือกใช้ NuSoap ในการพัฒนา เพราะเป็นชุด Libraly PHP ที่ใช้สำหรับเขียน Web Service ที่นิยมอย่างแพร่หลายในปัจจุบัน

3.3.2 ความเป็นไปได้ในทางปฏิบัติ (Operational Feasility)

จากการศึกษาพบว่า เกือบทุก ๆ องค์กรสามารถเชื่อมต่อระบบอินเตอร์เน็ตได้แล้ว แต่ยังคงใช้ระบบแลนในการทำงานมากกว่า เหตุเพราะควบคุมได้ง่าย เพราะเป็นเครือข่ายในที่เดียวกัน แต่ในยุคปัจจุบันเป็นยุคของอินเตอร์เน็ตความเร็วสูง ทำให้การใช้อินเตอร์เน็ตในปัจจุบัน มีความเร็วสูงได้ใกล้กับระบบของแลนในสมัยก่อนแล้ว การเริ่มต้นของการก้าวออกมาสู่ภายนอกนับเป็นการเริ่มที่ดีของการปรับเปลี่ยนระบบขององค์กรไปสู่ความเป็นมหาชน (Public) ทำให้เกิดโอกาสมากมาย ไม่ว่าจะเป็นการแลกเปลี่ยนข้อมูลข้ามแพลตฟอร์ม การลดเวลาการบำรุงรักษา (Maintainance) ลดเวลาพัฒนาโปรแกรม โดยที่ผู้ใช้งานไม่จำเป็นต้องมีความรู้เรื่องดังกล่าวเลย ผู้วิจัยจึงคิดว่าน่าจะเป็นประโยชน์ต่อการเริ่มต้นปรับไปสู่แนวคิด SOA

3.3.3 ความเป็นไปได้ในการลงทุน (Economic Feasibility)

เนื่องจากเครื่องมือที่ใช้ในการพัฒนาระบบ การให้บริการทางเว็บสำหรับ ระบบบัญชี นั้นเป็นโอเพนซอฟต์แวร์ ที่พัฒนาภายใต้ข้อกำหนดของ GPL (General Public License) จึงทำให้ไม่มีค่าใช้จ่ายในการพัฒนาแต่อย่างใด

3.4 การวิเคราะห์ระบบ (System Analysis)

จากการศึกษาจากแบบเรียนทางบัญชี พบว่ามีขั้นตอนการทำงานดังนี้

3.4.1 จัดเตรียมรายการทางการบัญชี (Business Transactions)

เมื่อเริ่มกิจการต้องมีรายการค้าเกิดขึ้น พร้อมที่จะนำมาบันทึกรายการทางการบัญชี รายการค้าในขั้นต้น เริ่มด้วยการนำเงินสด และสินทรัพย์มาลงทุนเริ่มแรก ขั้นต่อไปก็มีการจัดซื้อสินทรัพย์ เช่น ที่ดิน อาคาร อุปกรณ์ เครื่องใช้สำนักงาน และอาจจะมีนอกเหนือจากนี้แล้วแต่ขนาดและรูปแบบของกิจการ จากนั้นสามารถนำข้อมูลดังกล่าวมาทำงบแสดงฐานะของกิจการขึ้นมา

3.4.2 จัดหมวดหมู่และการให้เลขที่บัญชี

โดยมีชื่อว่า ผังบัญชี (Chart of General Ledger Account) คือ รายการบัญชีแยกประเภทเกี่ยวกับสินทรัพย์ หนี้สิน ทุน รายได้ และรายจ่าย ที่กิจการมีอยู่ในระบบบัญชีของกิจการ โดยจัดให้มีชื่อและเลขที่บัญชีเป็นหมวดหมู่อย่าง เป็นระเบียบ ผังบัญชีของแต่ละกิจการไม่จำเป็นต้องมีชื่อบัญชีที่เหมือนกัน ทั้งนี้ขึ้นอยู่กับ ลักษณะการดำเนินงานของธุรกิจ ขนาดของธุรกิจ การดำเนินงาน ความละเอียดของรายการ ในบัญชี เพื่อใช้ในการตัดสินใจของฝ่ายบริหาร เป็นต้น การจัดหมวดหมู่ และการให้เลขที่บัญชี โดยทั่วไปที่นิยมกันจะเริ่มที่บัญชีแยกประเภทสินทรัพย์เริ่มต้นด้วยเลข 1 หนี้สินเลข 2 ส่วนของ เจ้าของเลข 3 รายได้เริ่มด้วยเลข 4 และบัญชีแยกประเภทรายจ่ายเริ่มด้วยเลข 5 ในงานวิจัยนี้ ได้จำลองตัวอย่างของผังบัญชีดังนี้

ตารางที่ 3.1 ผังบัญชี สินทรัพย์หมุนเวียน (Current Assets)

เลขที่	ชื่อบัญชี
101	เงินสด (Cash)
102	เงินฝากธนาคาร (Bank)
103	ลูกหนี้ (Accounts Receivable)
104	สินค้าคงเหลือ (Inventories)
105	ตั๋วเงินรับ (Notes Receivable)
106	วัสดุสำนักงาน (Office Supplies)
107	อุปกรณ์สำนักงาน (Office Equipment)
108	อาคาร (Buiding)
109	ที่ดิน (Land)

ตารางที่ 3.2 ผังบัญชี หนี้สินหมุนเวียน (Current Liabilities)

เลขที่	ชื่อบัญชี
201	เงินเบิกเกินบัญชีธนาคาร (Bank Overdraft)
202	เจ้าหนี้ (Accounts Payable)
203	ตั๋วเงินจ่าย (Notes Payable)

ตารางที่ 3.2 ผังบัญชี หนี้สินหมุนเวียน (Current Liabilities) (ต่อ)

เลขที่	ชื่อบัญชี
204	ค่าใช้จ่ายค้างจ่าย (Accrual Expense)
205	เงินกู้ระยะยาว (Long-Tern Loan)

ตารางที่ 3.3 ผังบัญชี ส่วนของเจ้าของ (Owners' Equity)

เลขที่	ชื่อบัญชี
301	ทุน (Capital Account)
302	เบิกใช้ส่วนตัว (Drawing Account)
303	บัญชีกำไรขาดทุน (Profit and Loss Account)

ตารางที่ 3.4 ผังบัญชี รายได้ (Revenue)

เลขที่	ชื่อบัญชี
401	รายได้ค่าเช่า (Rent Revenue)
402	รายได้ค่าบริการ (Service Fee Earned)
403	รายได้อื่น ๆ (Other Revenue)

ตารางที่ 3.5 ผังบัญชี ค่าใช้จ่าย (Expenses)

เลขที่	ชื่อบัญชี
501	เงินเดือนและค่าจ้าง (Wage and Salary)
502	ค่าเช่า (Rent Expense)
503	ค่าโฆษณา (Advertising Expense)
504	ค่าซ่อมแซมและบำรุงรักษา (Repair and Maintenance)
505	ค่าสาธารณูปโภค (Utilitties Expense)
506	ค่าเบี้ยประกันภัย (Insurance Expense)
507	ค่ารับรอง (Entertainment Expense)
508	ค่าใช้จ่ายเบ็ดเตล็ด (Miscellaneous Expense)

3.4.3 นำข้อมูลรายการทางการเงินที่เกิดขึ้นในแต่ละวัน

เรียกว่า รายการทางบัญชี (Account Transaction) มาบันทึกลงสมุดรายวัน (General Journal) โดยต้องบันทึกตามหลักของบัญชีคู่ คือ ด้านเดบิตและเครดิต ด้วยจำนวนเงินที่เท่ากัน แต่จำนวนบัญชีไม่จำเป็นต้องเท่ากัน กล่าวคือ กิจการอาจบันทึกรายการทางด้านเดบิตมากกว่าหนึ่งบัญชี และบันทึกรายการทางด้านเครดิต เพียงหนึ่งหรือสองบัญชีเท่านั้น

ตารางที่ 3.6 รูปแบบสมุดรายวันทั่วไป

วันที่ (Date)	ชื่อบัญชี และคำอธิบายรายการ (Account Title and Explanation)	อ้างถึง (Post Ref)	เดบิต (Debit)		เครดิต (Credit)	

3.4.4 นำข้อมูลที่ได้จากการบันทึกลงสมุดรายวัน ไปบันทึกลงในสมุดบัญชีแยกประเภท (General Ledger) หมายถึง กลุ่มของบัญชี (Group of Account) ที่มีการแบ่งตามประเภทของ รายการบัญชีตามกลุ่ม คือ ในรายการบัญชีหนึ่ง อาจมีทั้งส่วนที่เดบิตและเครดิต บางรายการอาจมีเพียงด้านเดียว เมื่อรวบรวมแต่ละข้อมูลแล้ว จะนำเอาข้อมูลจากบัญชีแยกประเภทไปคำนวณงบต่อไป

3.4.5 ออกงบการเงิน โดยงบแรกที่สามารถออกได้จากข้อมูลทางบัญชีที่มีคือ งบทดลอง (Trial Balance) คืองบที่กิจการจัดทำขึ้นเพื่อสรุปยอดคงเหลือของบัญชีแยกประเภททุกบัญชี ที่กิจการมีอยู่ ซึ่งได้บันทึกรายการตามหลักบัญชีคู่และมีข้อมูลทางด้านบัญชีเพียงพอ ที่จะนำมาคำนวณงบการเงิน

ตารางที่ 3.7 ตัวอย่างงบทดลอง

อู่ซ่อมรถ(ตัวอย่าง) งบทดลอง วันที่ 31 มกราคม 25XX						
ลำดับที่	ชื่อบัญชี	เลขที่บัญชี	เดบิต		เครดิต	
1	เงินสด	101	71,000	-		
2	เงินฝากธนาคาร	102	346,000	-		

ตารางที่ 3.7 ตัวอย่างงบทดลอง (ต่อ)

อู่ซ่อมรถ(ตัวอย่าง) งบทดลอง วันที่ 31 มกราคม 25XX						
ลำดับที่	ชื่อบัญชี	เลขที่บัญชี	เดบิต		เครดิต	
3	ลูกหนี้	103	60,000			
4	สินค้าคงเหลือ	104	120,000	-		
5	ตั๋วเงินรับ	105	427,000	-		
6	วัสดุสำนักงาน	106	480,000	-		
7	เงินเบิกเกินบัญชีธนาคาร	201			200,000	-
8	ตั๋วเงินจ่าย	203			500,000	-
9	ทุน	301			647,000	-
10	ถอนใช้ส่วนตัว	302	70,000	-		
11	รายได้ค่าเช่า	401			320,000	-
12	เงินเดือนและค่าจ้าง	501	9,000	-		
13	ค่าเช่า	502	36,000	-		
14	ค่าโฆษณา	503	43,000	-		
15	ค่าซ่อมแซมและบำรุงรักษา	504	5,000	-		
			1,667,000		1,667,000	

ข้อผิดพลาดที่ทำให้งบทดลองไม่ลงตัว

งบทดลองของกิจการจากตัวอย่างมีเพียง 15 บัญชี ยอดคงเหลือแต่ละบัญชีมีไม่มาก งบทดลองจึงลงตัวได้ง่าย ในทางปฏิบัติบัญชีในงบทดลองอาจมีเป็น 30 - 50 บัญชี ยอดคงเหลือของแต่ละบัญชีบางบัญชีอาจมีหลายร้อยล้านบาท โอกาสที่งบทดลองจะไม่ลงตัวมีมาก บางครั้งงบทดลองลงตัวก็ไม่ได้หมายความว่าบัญชีที่เราได้บันทึกไว้ถูกต้องแล้ว ทั้งนี้เพราะบางครั้งลงบัญชีผิดทั้งทางด้านเดบิต และเครดิต งบทดลองก็ลงตัวเช่นกัน

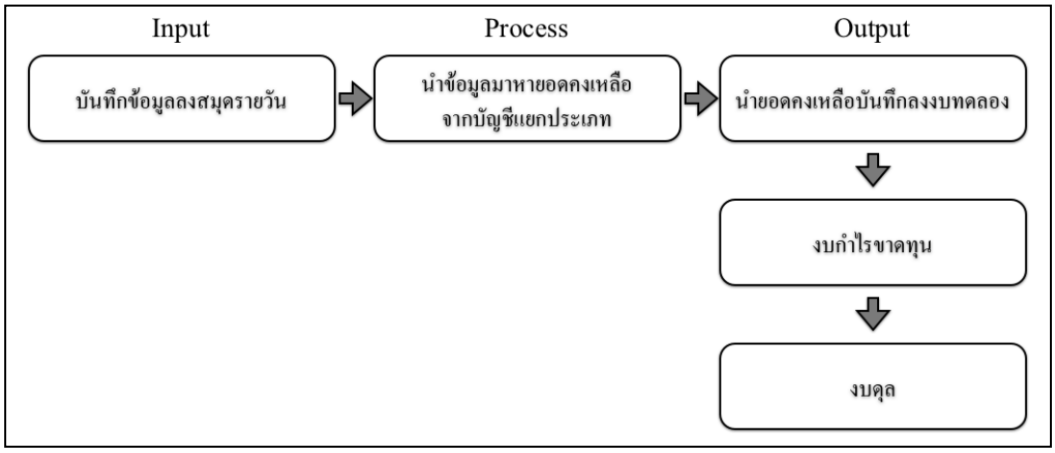
ข้อผิดพลาดเกิดจากการจัดทำงบทดลองไม่ลงตัว เกิดขึ้นได้ดังต่อไปนี้

1. เกิดจากการบวกยอดรวมทั้งสองข้างผิด
2. ยกยอดคงเหลือจากบัญชีแยกประเภทมาลงในงบทดลองผิด

3. นำยอดคงเหลือในบัญชีแยกประเภทมาลงผิดข้าง
4. ลืมยกยอดคงเหลือบางบัญชีมาใส่ในงบทดลอง
5. หายอดคงเหลือบางบัญชีผิด
6. ถ้าหาไม่พบ ต้องตรวจการบัญชีใหม่หมดทุกบัญชีและหายอดคงเหลือใหม่หมดเช่นกัน

เมื่อตรวจสอบแล้วว่า ไม่มีข้อผิดพลาดจึงสามารถนำเอาข้อมูลจากงบทดลองมาคำนวณ หางบกำไรขาดทุน (Income Statement) และงบดุล (Balance Sheet) ตามลำดับ

จากกระบวนการทำงานดังกล่าวผู้วิจัยได้ทำการแบ่งส่วนของการทำงานออกเป็น 3 ส่วน ได้แก่ (1) Input (2) Process (3) Output ดังรูป



รูปที่ 3.1 สรุปกระบวนการทำงานทางบัญชี

ผู้วิจัยจึงได้นำกระบวนการของการบัญชี มาปรับเป็น Web Service ได้นำส่วนที่เป็น Process มาเป็นส่วนของ Service และแบ่งส่วนอื่นออกเป็นส่วนแสดงผลดังนี้

1) ส่วนของหน้าจอหลัก
   - ส่วนของการบันทึกสมุดรายวัน
   - ส่วนของปฏิทิน
   - ส่วนของการแสดงผลของ สมุดรายวัน
2) ส่วนของบริการ
   - คำนวณบัญชีแยกประเภท
   - คำนวณงบการเงิน

3) ส่วนแสดงผล

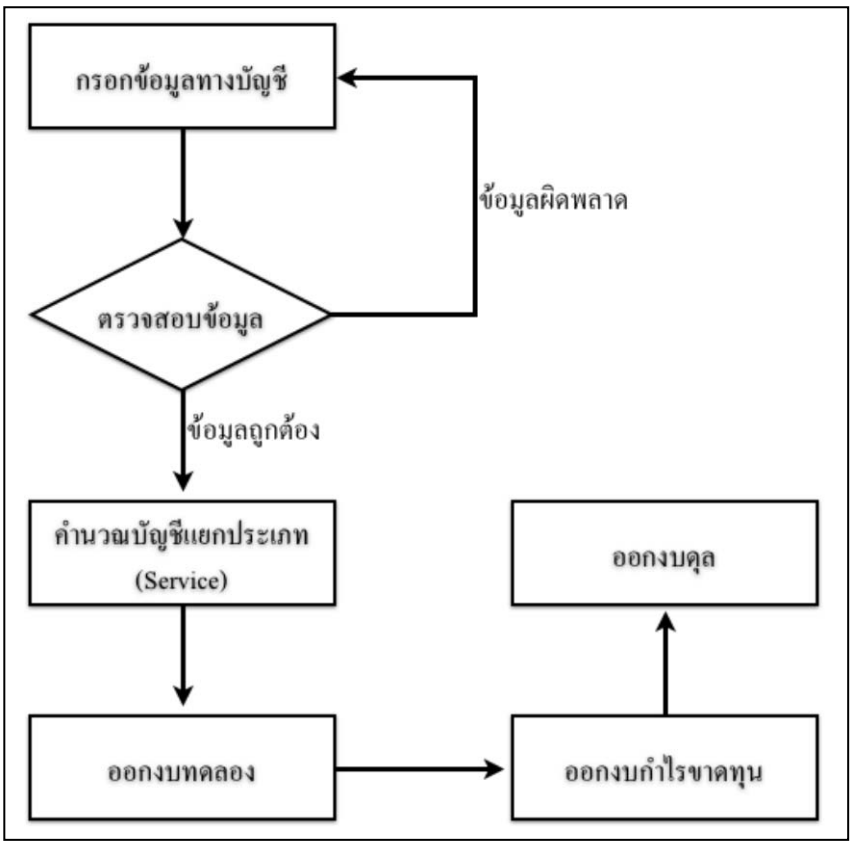
- แสดงงบทดลอง
- แสดงงบกำไรขาดทุน
- แสดงงบดุล

3.5 การออกแบบระบบ (System Design)

นำผลการวิเคราะห์ความต้องการและรูปแบบการทำงานมาทำการออกแบบตามหลักของ Software Engineering ซึ่งสามารถออกแบบในส่วนต่าง ๆ ได้ดังนี้

3.5.1 การออกแบบผังระบบ (System Flowchart)

จะเป็นการออกแบบภาพรวมของระบบ ซึ่งโครงสร้างการทำงานของการให้บริการทางเว็บสำหรับระบบบัญชี จะมีผังการทำงานดังนี้

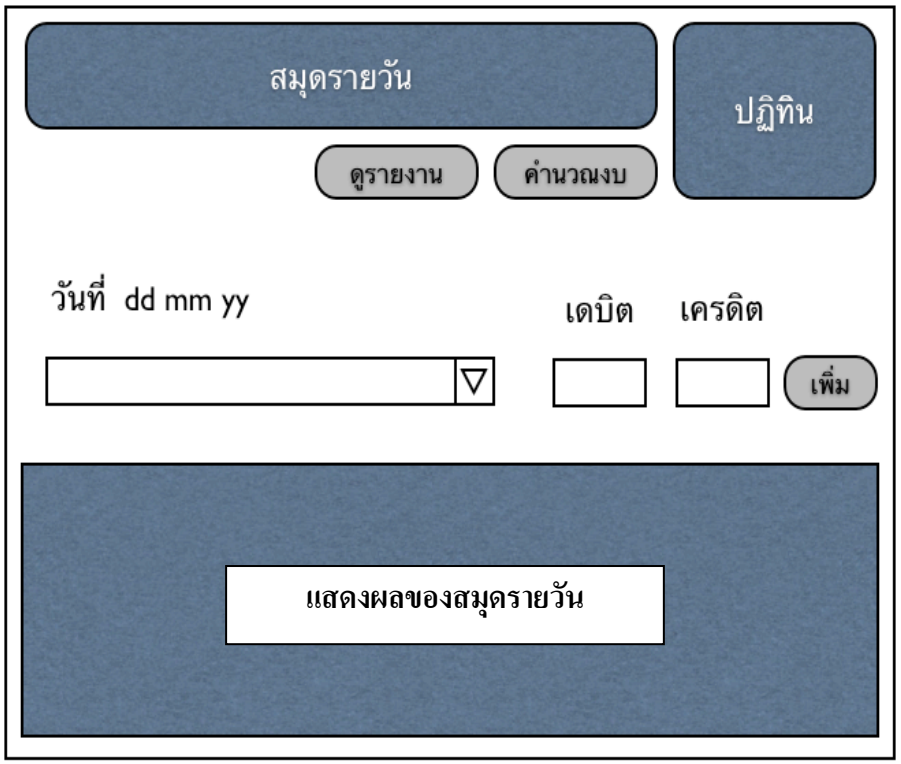


รูปที่ 3.2 แสดงโครงสร้างการทำงานของระบบ Web Service for Accounting System

จากรูปที่ 3.2 อธิบายการทำงานของระบบได้คือ เริ่มต้นที่ผู้ใช้ต้องกรอกข้อมูลทางบัญชีลงสมุดรายวันในหน้าแรกก่อน หากมีการกรอกผิดพลาด ไม่ครบถ้วน ระบบจะมีคำเตือนแสดงขึ้นมา เมื่อกรอกเสร็จแล้ว และข้อมูลถูกต้อง ระบบจะส่งข้อมูลของสมุดรายวัน ไปคำนวณในบัญชีแยกประเภท โดยในส่วนนี้จะเป็นการเรียกใช้งาน Service แล้ว ผู้ใช้จะไม่เห็น รูปแบบการคำนวณ จากนั้นระบบจะส่งผลลัพธ์กลับมาเป็นงบการเงินทั้ง 3 งบ

3.5.2 การออกแบบหน้าจอ (Screen Design)

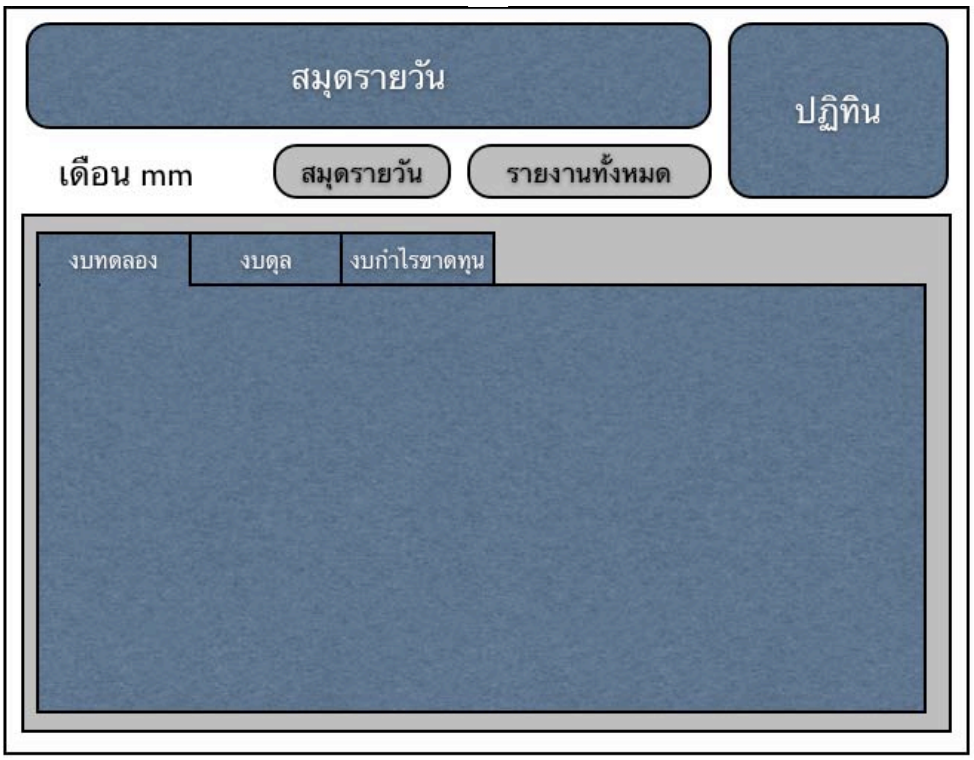
จากข้อมูลข้างต้น พบว่ามีส่วนแสดงผลอยู่ 3 ส่วน คือ (1) ส่วนของการกรอกสมุดรายวัน (2) ส่วนของการรายงานผลการกรอกสมุดรายวัน (3) ส่วนของการแสดงงบการเงิน ผู้วิจัยได้ออกแบบหน้าจอของระบบดังนี้



รูปที่ 3.3 แสดงส่วนของสมุดรายวัน

ส่วนที่เป็น DropDownList เป็นการดึงข้อมูลออกมาจาก ผังบัญชี เพื่อประหยัดเวลาในการบันทึกข้อมูล และลดอัตราการผิดพลาดจากการกรอกข้อมูลเองอีกด้วย โดยข้อมูลของผังบัญชีจะเก็บไว้ในรูปของ array ส่วนของการแสดงผลนั้น จะแสดงตามสมุดรายวันแล้วสามารถ แก้ไข - ลบ ข้อมูลได้ตลอดเวลา หลังจากบันทึกไปแล้ว ปุ่ม “ดูรายงาน” เป็นการดูข้อมูลของสมุดรายวันที่ได้

บันทึกเอาไว้ตลอดทั้งเดือนขึ้นมาแสดง ปุ่ม “คำนวณงบ” เป็นการส่งข้อมูลสมุดรายวันไปทำงานคำนวณ แล้วส่งผลมารายงานเป็นงบการเงินต่าง ๆ ดังรูป

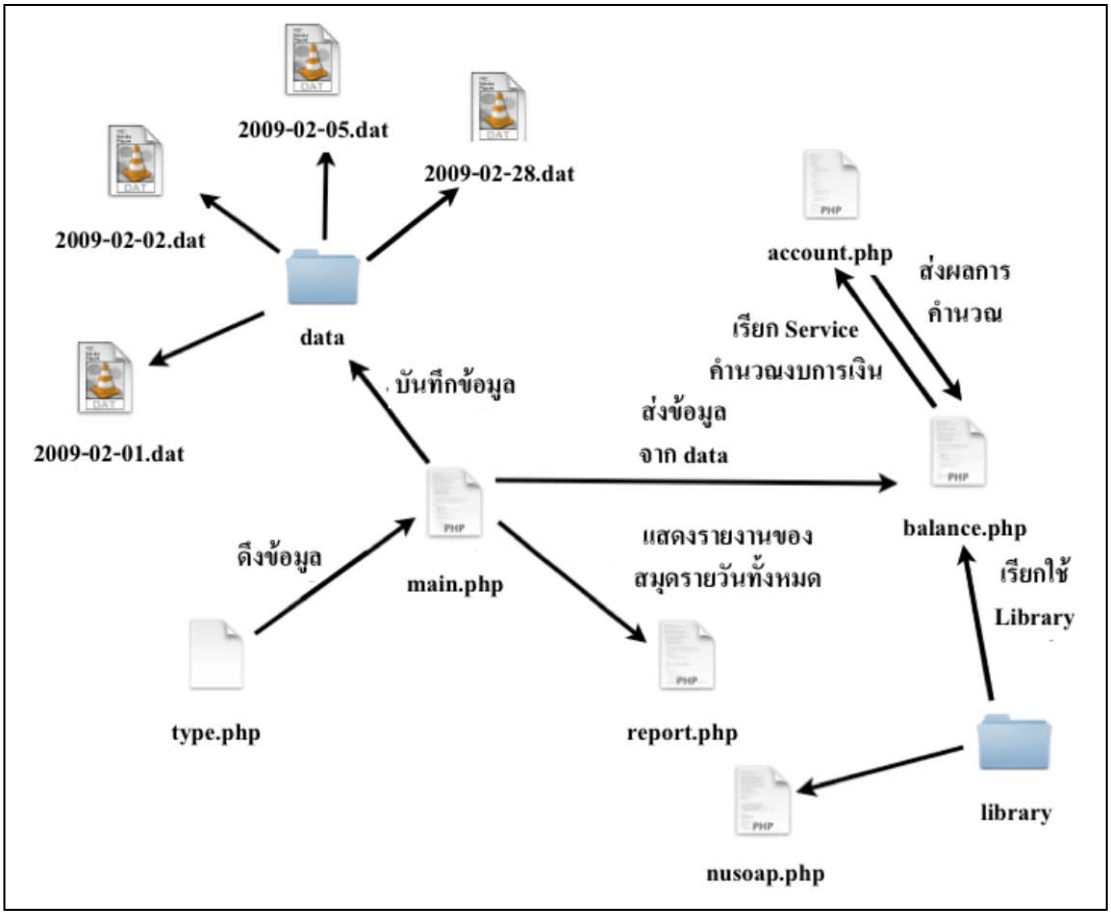


รูปที่ 3.4 แสดงหน้าจอส่วนของการรายงานงบการเงิน

ส่วนของการแสดงผลนี้ จะรายงานออกมาเป็นผลลัพธ์จากการคำนวณการเงินผ่านทาง Web Service เรียบร้อยแล้ว ปุ่ม “สมุดรายวัน” จะกลับไปยังหน้าก่อนนี้ เพื่อบันทึกเพิ่มเติม หรือกลับไปแก้ไข ส่วนปุ่ม “รายงานทั้งหมด” ก็คือปุ่มที่แสดงผลสมุดรายวันของทั้งเดือน งบต่าง ๆ ทางผู้วิจัยได้ทำเป็น tab เพื่อช่วยให้การดูข้อมูลง่าย และประหยัดในส่วนของการแสดงผลไปได้มาก

3.5.3 ออกแบบโครงสร้างของโปรแกรม

หลังจากได้ออกแบบส่วนของการทำงานและหน้าจอไปแล้ว ผู้วิจัยจึงได้ออกแบบตัวของโปรแกรม โดยมีความสัมพันธ์ของแต่ละไฟล์ ดังนี้



รูปที่ 3.5 แสดงความสัมพันธ์ ระหว่าง class

โดยแต่ละส่วนสามารถอธิบายโดยละเอียดดังนี้

การเก็บข้อมูลของผังบัญชีจะจัดเก็บลงที่ type.php โดยมีการจัดเก็บลงไปภายในซอร์สโค้ด โดยได้จัดเก็บเป็น array ดังนี้

```
<?
        $type_array = array(
                101 => array('group' => 1, 'name' => 'เงินสด(Cash)'),
                102 => array('group' => 1, 'name' => 'ธนาคาร/เงินฝากธนาคาร(Bank)'),
                103 => array('group' => 1, 'name' => 'ลูกหนี้(Account Receivable)')
                );
?>
```

รูปที่ 3.6 แสดงการเก็บข้อมูลของผังบัญชี

เมื่อมีการเก็บข้อมูลเป็น array ทำให้ง่ายต่อการติดตั้ง เมื่อไปติดตั้งเครื่องอื่นก็สามารถใช้งานได้เลยดังรูป โดยไม่จำเป็นต้องติดตั้งระบบฐานข้อมูลก่อน โดยการเก็บเริ่มจากการแบ่งหมวดหมู่ตามผังบัญชี เช่น 101 102 เป็นต้น จากนั้นระบุไปให้ครบรายการที่กิจการนั้น ๆ จำเป็นต้องใช้ และครอบคลุมงานทางบัญชีส่วนใหญ่ได้

ในส่วนของหน้าหลัก main.php มีการเรียกข้อมูลดังกล่าวมาใช้งาน โดยจะนำข้อมูลของผังบัญชีมาแสดงผลในรูปของ DropDownList เพื่อให้ง่ายต่อการบันทึกรายการทางบัญชี หลังจากที่บันทึกรายการไปแล้ว ระบบจะทำการบันทึกข้อมูลลงใน *.dat โดยข้อมูลที่ทำการส่งไปบันทึกมีรูปแบบดังนี้

type = 101	string=0	cr= 400,000	dr= 0

type = 503	string=1	cr= 0	dr= 400,000

รูปที่ 3.7 แสดงการส่งข้อมูลของผังบัญชี

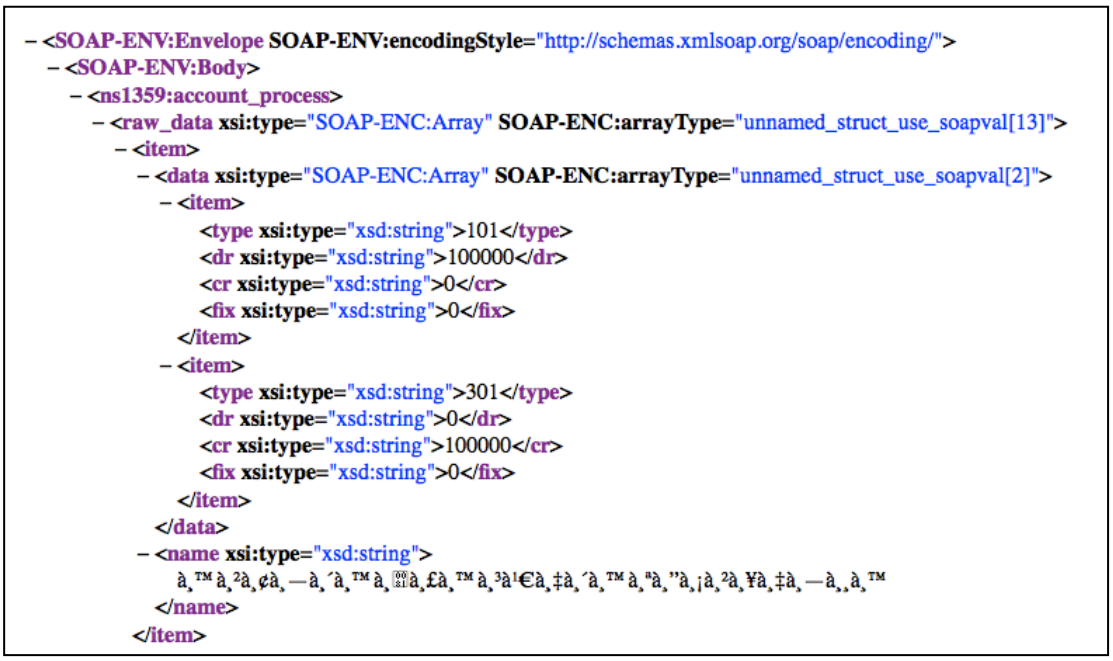
3.5.4 แสดงข้อมูลที่ถูกส่งไปบันทึกสมุดบัญชีรายวัน

ส่วนที่นำมาคำนวณคืออินเด็กซ์ที่ชื่อว่า [type] ซึ่งเป็นส่วนที่ใช้บอกถึงผังบัญชีที่ได้ทำการบันทึกลงไป [dr] คือการเดบิต (Debit) และ [cr] คือการเครดิต (Credit) ดังรูปหมายถึง ได้ทำการเดบิตหมวด 503 จำนวน 40,000 บาท และได้ทำการเครดิตหมวด 101 จำนวน 40,000 บาท เป็นต้น เมื่อบันทึกบัญชีแล้ว ระบบจะทำการเข้ารหัสเก็บเอาไว้ เพื่อให้ง่ายต่อการส่งข้อมูลไปยังเว็บเซอร์วิส เนื่องจากผู้วิจัยพบว่าการส่งข้อมูลเป็นชุดของ array ไปนั้น จะทำให้การนำข้อมูลภายในอินเด็กซ์ของ array ออกมาคำนวณได้ยาก และโปรแกรมจะมีการเขียนที่ซับซ้อนมาก ยากต่อการแก้ไข แต่การเข้ารหัสจะทำให้ระบบมองเป็นชุดของ String และเมื่อถอดรหัสที่ส่วนของเว็บเซอร์วิส ทำให้ดึงเอามาเฉพาะส่วนที่ต้องการคำนวณได้

การส่งข้อมูลของ Web Service ของระบบนี้ สามารถเขียนเป็น Process การทำงานได้ดังรูป

เมื่อทำการบันทึกสมุดรายวันจนครบทุกรายการทางบัญชีแล้ว หากต้องการดูข้อมูลทั้งหมดสามารถทำได้ผ่านทาง report.php โดยไฟล์นี้จะแสดงผลของสมุดรายวันที่ได้ทำการ บันทึกแล้วออกมาทั้งหมด ทำให้ง่ายต่อการดูภาพรวมของการบันทึกข้อมูล

การส่งข้อมูลไปเพื่อทำการคำนวณนั้นจะส่งข้อมูลไปเก็บที่ balance.php ซึ่งเป็นหน้าส่วน ของการแสดงผล และเรียกใช้เว็บเซอร์วิสจาก account.php ทำการคำนวณ โดยข้อมูลที่ส่งไปยัง Service นั้นมีรูปแบบดังนี้



รูปที่ 3.9 แสดงการส่งข้อมูลไปยัง เว็บเซอร์วิส

จากรูป จะเห็นได้ว่าการส่งข้อมูลนั้นจะมีการปรับให้เป็น xml เพื่อให้เกิดเป็นมาตรฐานของการส่งข้อมูลไปกระทำการต่าง ๆ ที่ได้กำหนดไว้ในแต่ละ Service นั่นเอง

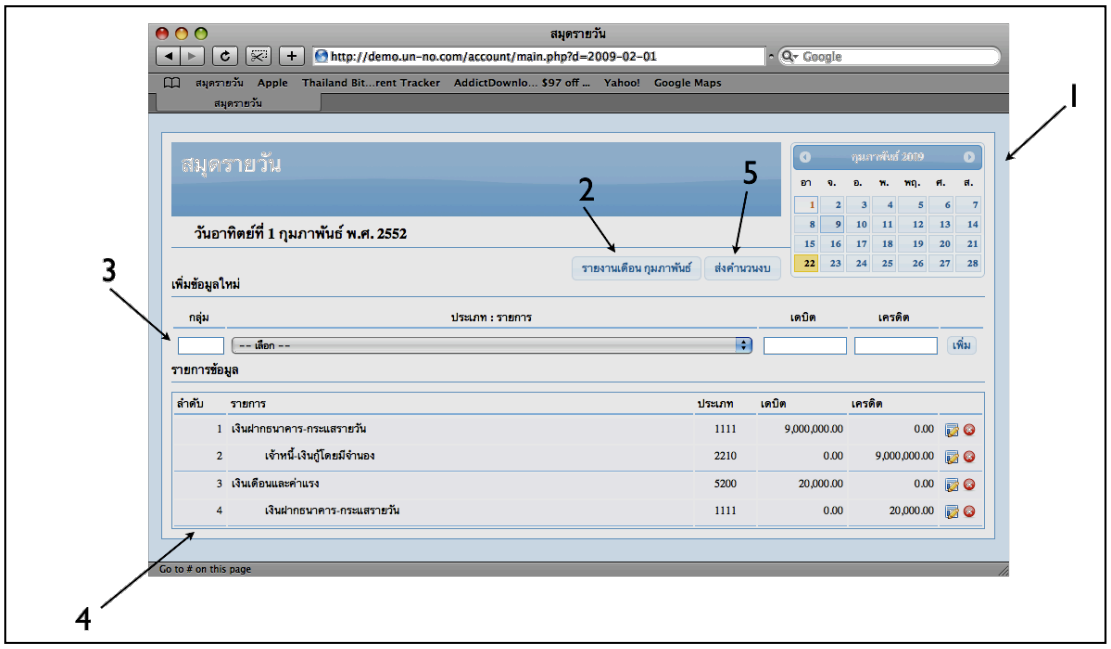
เมื่อเว็บเซอร์วิสทำการคำนวณแล้ว จะส่งผลลัพธ์กลับมายังหน้าจอแสดงผล เพื่อทำการจัดรูปแบบการแสดงผลให้ออกมาเป็นงบการเงินที่ต้องการ

3.6 การพัฒนาระบบ (Implementation)

สำหรับขั้นตอนนี้การพัฒนาระบบนั้น ได้นำเอาข้อมูลต่าง ๆ ไม่ว่าจะจากการวิเคราะห์ระบบและการออกแบบต่าง ๆ มาใช้เป็นต้นแบบในการพัฒนา ซึ่งมีส่วนหลัก ๆ 3 ส่วนคือ (1) หน้าจอหลัก (2) ส่วนของบริการ (3) ส่วนของการแสดงผล

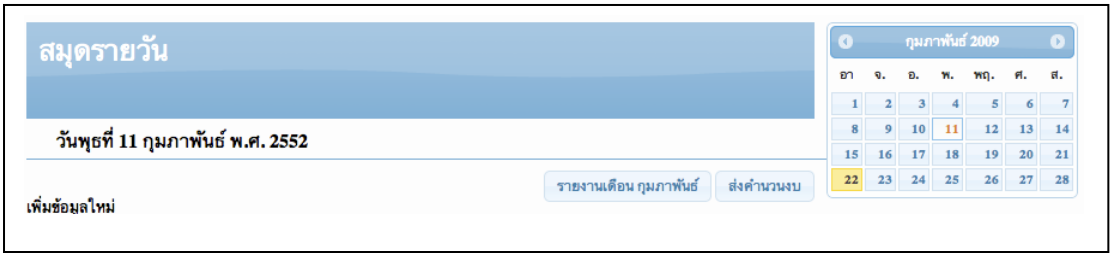
3.6.1 หน้าจอหลัก

เป็นการพัฒนาระบบการทำงานส่วนในการบันทึกข้อมูลลงในสมุดรายวัน ซึ่งส่วนนี้จะทำหน้าที่เก็บข้อมูลด้วย โดยข้อมูลที่ใช้ในการบันทึก จะนำมาจาก type.php หน้าจอหลักจะประกอบไปด้วย 5 ส่วนดังนี้



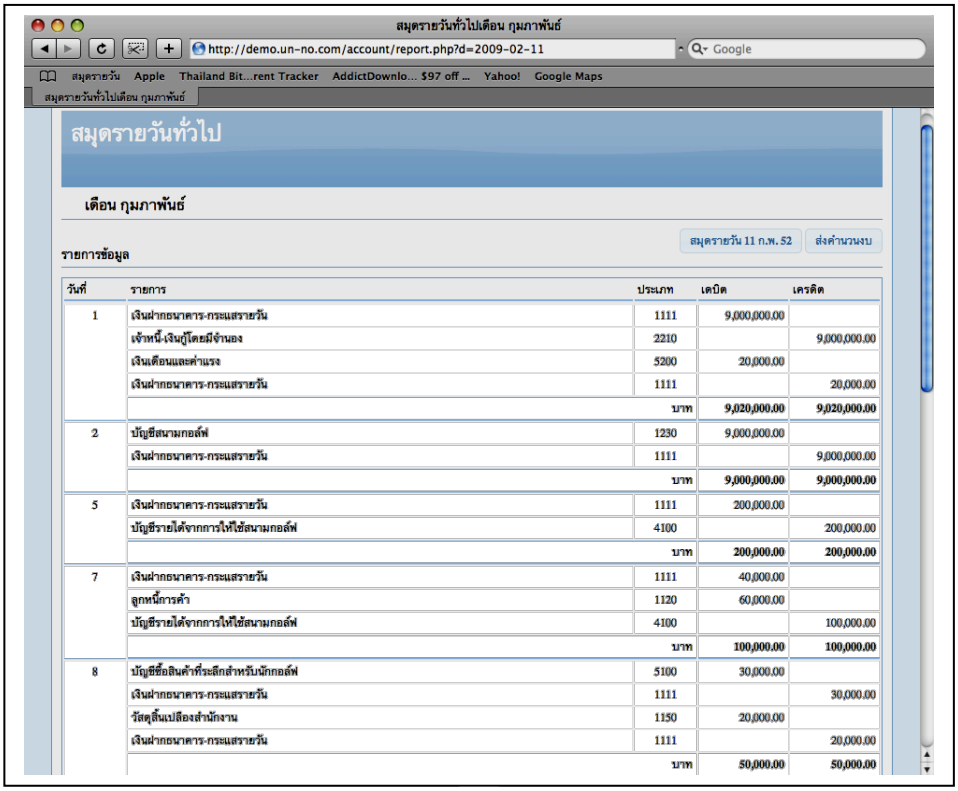
รูปที่ 3.10 แสดงหน้าจอหลักของระบบ

ส่วนที่ 1 ปฏิทิน โดยในส่วนนี้จะใช้แสดงวันที่ทำการกรอกข้อมูล โดยจะสอดคล้องกับวันที่ทางซ้ายมือ เมื่อต้องการเปลี่ยนวันที่ทำการกรอกข้อมูล ก็สามารถคลิกที่วันที่ หรือเดือนที่ต้องการได้ทันที



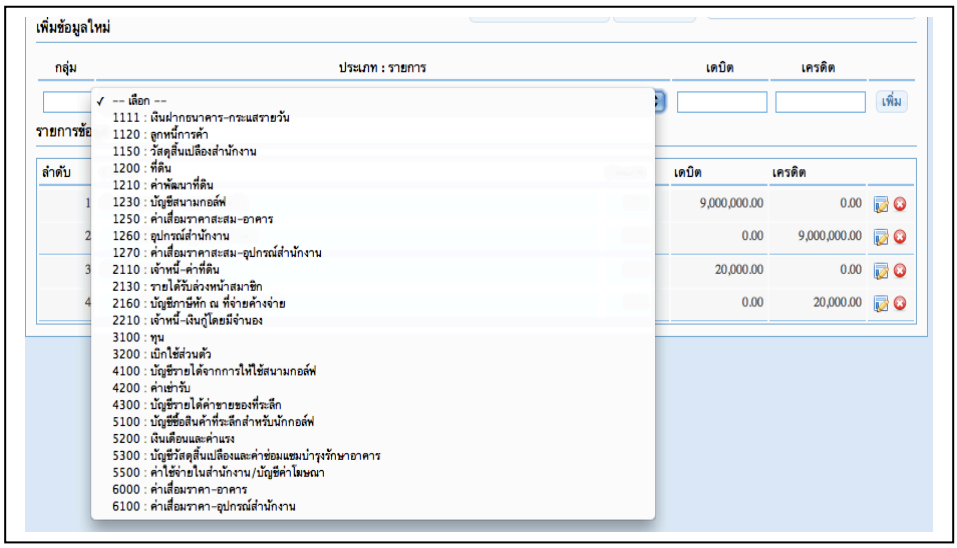
รูปที่ 3.11 แสดงการเปลี่ยนวันที่ของปฏิทิน

ส่วนที่ 2 เป็นปุ่มที่ทำหน้าที่เรียกดูสมุดรายวันทั้งหมดที่ได้ทำการบันทึกเอาไว้ แล้วโดยหลังจากคลิก ระบบจะทำการเรียก report.php เพื่อแสดงรายงานของสมุดรายวันในเดือนนั้นทั้งหมด



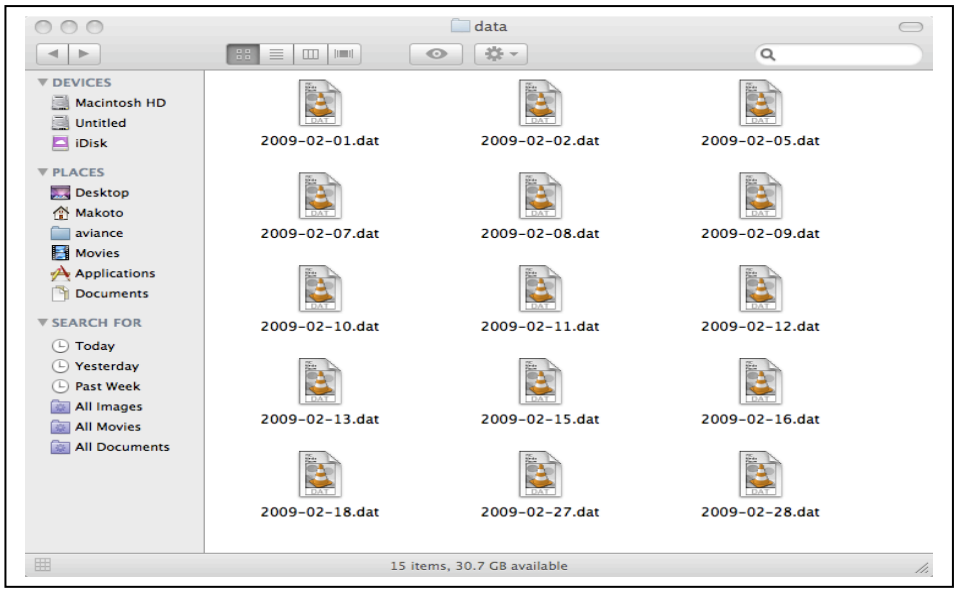
รูปที่ 3.12 แสดงรายงานสมุดบัญชีรายวันทั้งหมด

ส่วนที่ 3 ส่วนบันทึกข้อมูล โดยจะทำการเลือกข้อมูลที่จะทำการบันทึกจากข้อมูลที่อยู่ใน type.php ซึ่งเป็นข้อมูลผังบัญชี



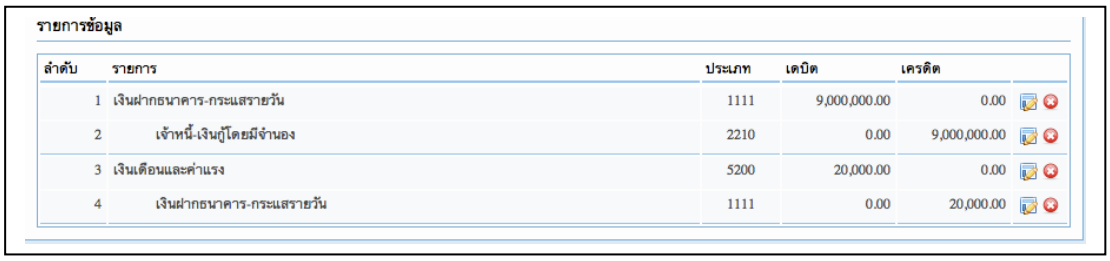
รูปที่ 3.13 แสดงส่วนของการบันทึกข้อมูล

เมื่อบันทึกจะทำการสร้าง .dat file ขึ้นมาเก็บข้อมูลเอาไว้ เป็นรายวัน



รูปที่ 3.14 แสดง file ที่บันทึกจากสมุดรายวัน

ส่วนที่ 4 เป็นส่วนที่แสดงผลจากข้อมูลที่บันทึกไปแล้วโดยสามารถแก้ไขและลบได้

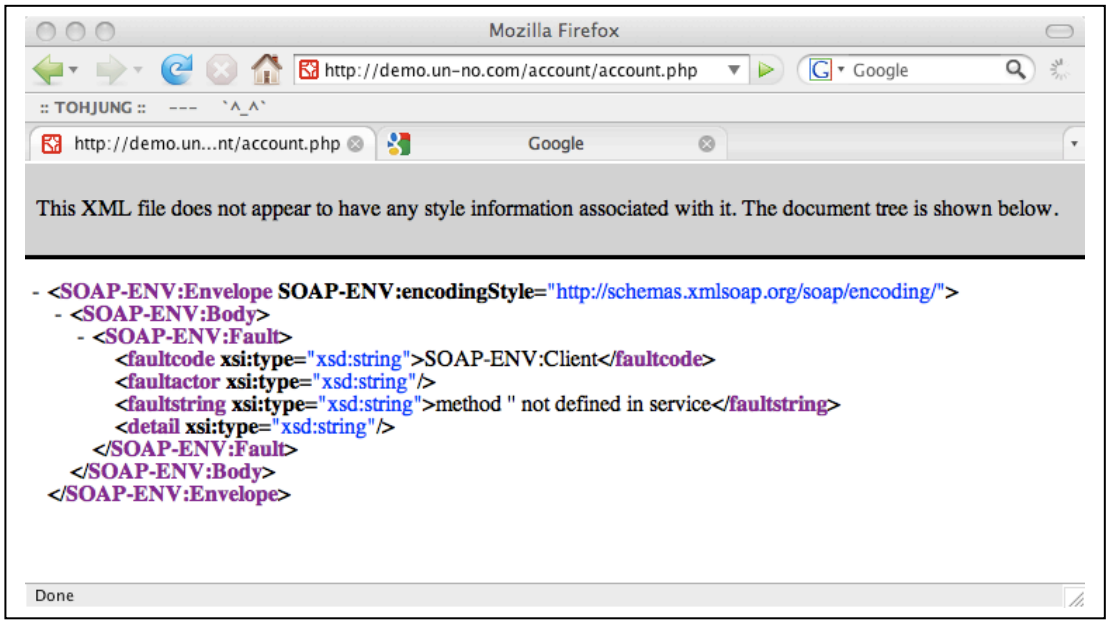


รูปที่ 3.15 แสดงสมุดรายวันที่บันทึกแล้ว

ส่วนที่ 5 ปุ่มที่ใช้ในการส่งข้อมูลไปคำนวณงบการเงินต่าง ๆ ปุ่ม “ส่งคำนวณงบ” จะทำหน้าที่ส่งข้อมูลจากสมุดรายวันไปคำนวณงบ โดยผ่านทาง Web Service ซึ่งในระบบนี้ก็คือ account.php

3.6.2 ส่วนของบริการ

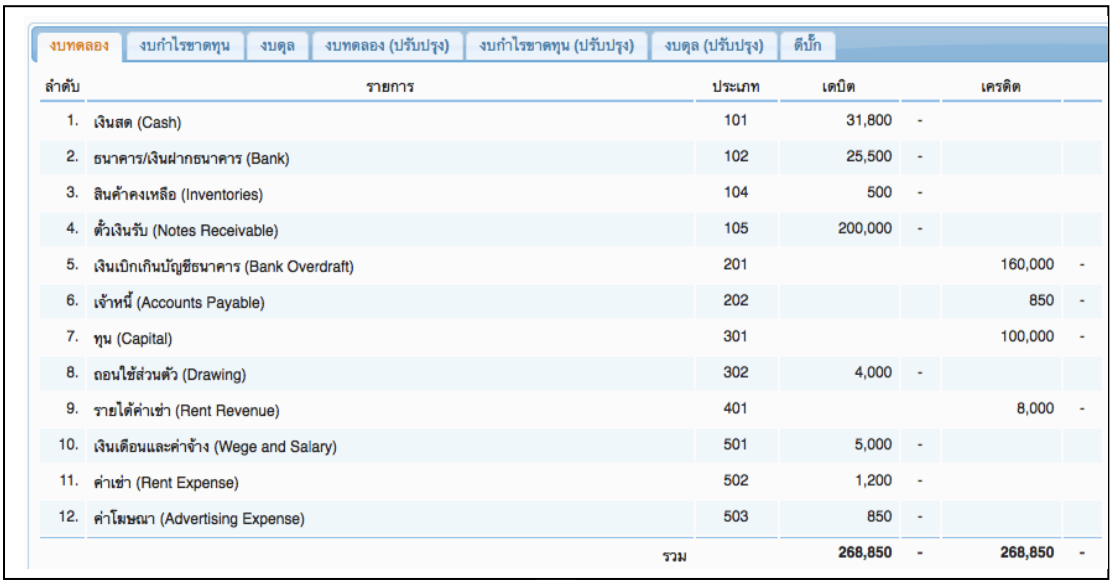
ส่วนนี้เป็นส่วนที่สืบเนื่องจากส่วนที่ 1 หรือส่วนของหน้าจอหลัก โดยส่วนนี้ จะไม่มีหน้าจอให้ผู้ใช้เห็น แต่สามารถเรียกดูการทำงานได้ โดยสามารถเรียกดู SOAP Message ได้โดยตรง



รูปที่ 3.16 แสดง SOAP Message ของ account.php

3.6.3 ส่วนของการแสดงผล

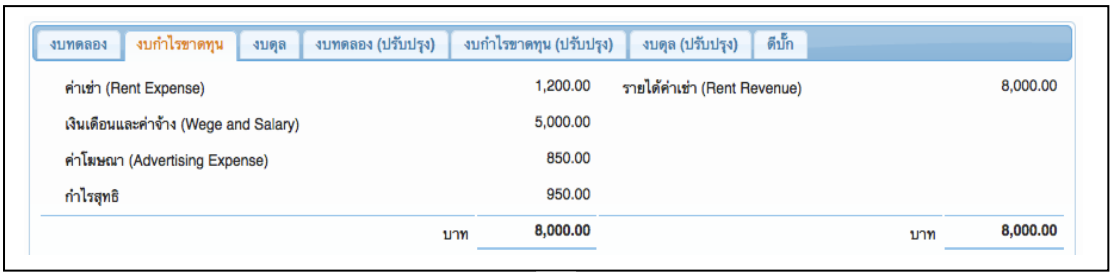
ส่วนนี้เป็นส่วนที่หลังจากข้อมูลได้รับการคำนวณจาก Web Service เรียบร้อยแล้ว ระบบจะนำเอาข้อมูลมาแสดงผลใน balance.php เป็นในลักษณะของงบการเงินทั้ง 3 งบ



งบทดลอง | งบกำไรขาดทุน | งบดุล | งบทดลอง (ปรับปรุง) | งบกำไรขาดทุน (ปรับปรุง) | งบดุล (ปรับปรุง) | ดีบั๊ก

ลำดับ	รายการ	ประเภท	เดบิต		เครดิต	
1.	เงินสด (Cash)	101	31,800	-		
2.	ธนาคาร/เงินฝากธนาคาร (Bank)	102	25,500	-		
3.	สินค้าคงเหลือ (Inventories)	104	500	-		
4.	ตั๋วเงินรับ (Notes Receivable)	105	200,000	-		
5.	เงินเบิกเกินบัญชีธนาคาร (Bank Overdraft)	201			160,000	-
6.	เจ้าหนี้ (Accounts Payable)	202			850	-
7.	ทุน (Capital)	301			100,000	-
8.	ถอนใช้ส่วนตัว (Drawing)	302	4,000	-		
9.	รายได้ค่าเช่า (Rent Revenue)	401			8,000	-
10.	เงินเดือนและค่าจ้าง (Wege and Salary)	501	5,000	-		
11.	ค่าเช่า (Rent Expense)	502	1,200	-		
12.	ค่าโฆษณา (Advertising Expense)	503	850	-		
		รวม	**268,850**	-	**268,850**	-

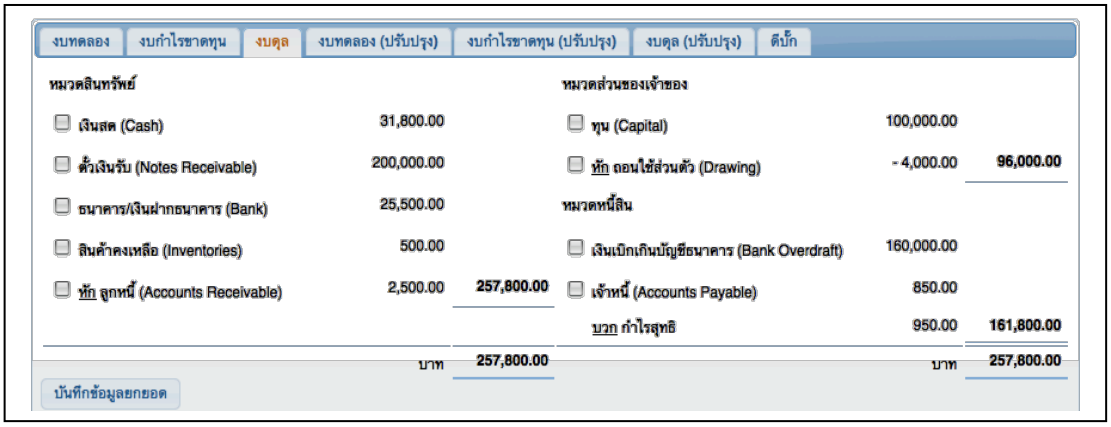
รูปที่ 3.17 แสดงหน้าจอของการรายงานงบทดลอง



งบทดลอง | งบกำไรขาดทุน | งบดุล | งบทดลอง (ปรับปรุง) | งบกำไรขาดทุน (ปรับปรุง) | งบดุล (ปรับปรุง) | ดีบั๊ก

ค่าเช่า (Rent Expense)	1,200.00	รายได้ค่าเช่า (Rent Revenue)	8,000.00
เงินเดือนและค่าจ้าง (Wege and Salary)	5,000.00		
ค่าโฆษณา (Advertising Expense)	850.00		
กำไรสุทธิ	950.00		
บาท	**8,000.00**	บาท	**8,000.00**

รูปที่ 3.18 แสดงหน้าจอของการรายงานงบกำไรขาดทุน



รูปที่ 3.19 แสดงหน้าจอของการรายงานงบดุล

และในตัวของโปรแกรมยังสามารถดูงบการเงินแบบปรับปรุงได้ หากมีการปรับปรุงแล้ว ดังรูป

รูปที่ 3.20 แสดงหน้าจอรายงานของงบการเงิน

บทที่ 4
การทดสอบ และสรุปผล

ส่วนของการทดสอบระบบนี้จะเป็นการทดสอบโดยใช้ข้อมูลที่เกิดขึ้นเป็นเวลา 1 เดือน โดยจะจำลองข้อมูลขึ้นมาจากตัวอย่างรายการบัญชีจากกิจการสัญญากอล์ฟ ซึ่งเป็นตัวอย่างของบทที่ 3 จากนั้นจะทำการสรุปผลการทดสอบว่าเป็นอย่างไร

4.1 สภาพแวดล้อมที่ใช้ในการทดสอบ

ผู้วิจัยได้ใช้เครื่องคอมพิวเตอร์โน๊ตบุคของตนเองเป็นเครื่องทดสอบ โดยเครื่องที่ใช้ในการพัฒนาและทดสอบมีรายละเอียดดังนี้

4.1.1 ฮาร์ดแวร์

- คอมพิวเตอร์แบบโน๊ตบุค 2.4 GHz Core 2 DUO
- หน่วยความจำ 2 กิ๊กกะไบต์
- ฮาร์ดดิส 200 กิ๊กกะไบต์

4.1.2 ซอฟต์แวร์

- ระบบปฏิบัติการ Mac OS X Leopard Version 10.5.6
- เว็บเซิร์ฟเวอร์ Apache ผ่านโปรแกรม MAMP (MAC, Apache, MySQL, PHP)
- ภาษาสคริปต์ PHP Web Browser Safari และ Firefox

4.2 ข้อมูลที่ใช้ในการทดสอบ

ข้อมูลนี้เป็นข้อมูลของกิจการ รถเช่าของนายทินกร ที่เริ่มจัดทำบัญชีตั้งแต่วันที่ 1 ถึง 31 มีนาคม และได้ออกรายการทางบัญชีเอาไว้เรียบร้อยโดยมีรายการค้าต่าง ๆ ดังนี้

มี.ค. 1 นายทินกรได้เปิดร้านให้เช่าจักรยานยนต์ ชื่อ “ทินกร-รถเช่า” โดยนำเงินสดมาลงทุน 100,00 บาท

2 ร้านขายรถจักยานยนต์ออกตั๋วเงินรับค่ารถให้นายทินกร เป็นเงินเชื่อ 5 คัน คันละ 40,000 บาท

5 นำเงินฝากธนาคาร 60,000 บาท

7 จ่ายค่าเช่าร้าน 1,200 บาท และซื้อของสิ้นเปลือง 500 บาท

10 รับรายได้ค่าเช่ารถ 1,200 บาท และนำฝากธนาคารทันที

11 จ่ายชำระหนี้ค่าตั๋วเงิน จำนวน 40,000 บาท ด้วยเช็ค

17 มีรายได้ค่าเช่ารถจำนวน 5,000 บาท แต่ลูกค้าชำระเป็นเงินสดวันนี้ 2,500 บาท ที่เหลือขอค้างไว้ก่อน

19 จ่ายเงินเดือนลูกจ้าง 5,000 บาท

23 นายทินกรนำเงินสดไปใช้ส่วนตัว 4,000 บาท

26 รับชำระหนี้จากลูกหนี้ 2,500 บาท ด้วยเช็คฝากธนาคารเรียบร้อยแล้ว

30 รับบิลโฆษณาร้านเช่ารถ 850 บาท ยังไม่ได้ไปชำระ

การบันทึกรายการทำได้ดังนี้

ตารางที่ 4.1 สมุดรายวันทั่วไป

วันที่	รายการ	อ้างถึง	เดบิต	เครดิต
มี.ค. 1	เงินสด	101	100,000	
	ทุน-นายทินกร	301		100,000
	นายทินกรนำเงินมาลงทุน			
2	ตั๋วเงินรับ	105	200,000	
	เจ้าหนี้การค้า	201		200,000
	ซื้อรถจากร้านจักรยานยนต์เป็นตั๋วเงิน แบบเงินเชื่อ 5 คัน			
5	เงินฝากธนาคาร	102	60,000	
	เงินสด	101		60,000
	นำเงินสดฝากธนาคาร			

ตารางที่ 4.1 สมุดรายวันทั่วไป (ต่อ)

วันที่	รายการ	อ้างถึง	เดบิต	เครดิต
7	ค่าเช่าร้าน	502	1,200	
	วัสดุสิ้นเปลือง	104	500	
	เงินสด	101		1,700
	จ่ายค่าเช่าร้านและซื้อวัสดุสิ้นเปลืองเป็นเงินสด			
10	เงินฝากธนาคาร	102	3,000	
	รายได้ค่าเช่า	401		3,000
	รับรายได้ค่าเช่ารถ ฝากธนาคารทันที			
11	เงินเบิกเกินบัญชีธนาคาร	201	40,000	
	เงินฝากธนาคาร	102		400,000
	จ่ายชำระหนี้ร้านรถจักยานยนต์ด้วยเช็ค			
17	เงินสด	101	2,500	
	ลูกหนี้	103	2,500	
	รายได้ค่าเช่า	401		5,000
	รับรายได้ค่าเช่ารถเป็นเงินสดบางส่วนที่เหลือลูกค้าขอค้างไว้ก่อน			
19	เงินเดือนและค่าจ้าง	501	5,000	
	เงินสด	101		5,000
	จ่ายเงินดือนลูกจ้างในร้าน			
23	ถอนใช้ส่วนตัว	302	4,000	
	เงินสด	101		4,000
	นายทินกรถอนเงินไปใช้ส่วนตัว			
26	เงินฝากธนาคาร	102	2,500	
	ลูกหนี้	103		2,500
	รับชำระหนี้เป็นเงินฝากธนาคาร			

ตารางที่ 4.1 สมุดรายวันทั่วไป (ต่อ)

วันที่	รายการ	อ้างถึง	เดบิต	เครดิต
30	ค่าโฆษณา	503	850	
	เจ้าหนี้การค้า	202		850
	ได้รับบิลค่าโฆษณา ยังไม่ได้ชำระ			

จากข้อมูล นี่คือการบันทึกสมุดรายวัน แล้วนำผลลัพธ์ไปคำนวณในรูปแบบของบัญชี แยกประเภท ซึ่งเป็นการคำนวณ เบื้องหลัง โดยในระบบนี้จะนำเอาข้อมูลชุดนี้กรอกเข้าไป ทางหน้าจอหลัก หากผลลัพธ์ออกมาเป็น ปิดงบลงตัว แสดงว่า โปรแกรมมีการคำนวณที่ถูกต้อง

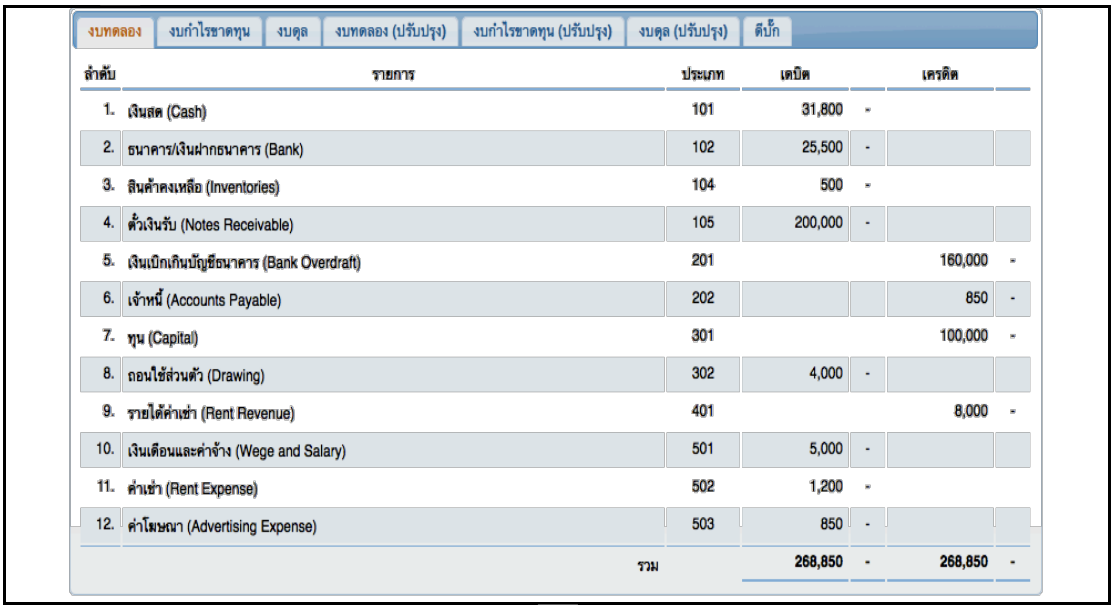
4.3 ขั้นตอนในการทดสอบ

ขั้นตอนนี้จะนำเอาข้อมูลดิบมากรอกข้อมูลลงในโปรแกรมระบบการบริการทางเว็บสำหรับระบบบัญชีจนครบทุกรายการ จากนั้นจะทำการแสดงผลออกมา เพื่อดูว่าสามารถปิดงบได้หรือไม่ โดยเมื่อบันทึกไปแล้ว โปรแกรมก็จะทำการเขียน dat file ขึ้นมาตามวันที่มีข้อมูล ดังรูปที่ 3.13 และเมื่อต้องการตรวจสอบทีเดียวทั้งหมด ให้เข้าไปที่ report.php ผ่านทางปุ่ม “รายงานเดือน xxx” ก็จะสามารถดูข้อมูลของสมุดรายวันทั้งเดือนได้

วันที่	รายการ	ประเภท	เดบิต	เครดิต
1	เงินสด (Cash)	101	100,000.00	
	ทุน (Capital)	301		100,000.00
	นายทินกรนำเงินสดมาลงทุน			
		บาท	100,000.00	100,000.00
2	ตั๋วเงินรับ (Notes Receivable)	105	200,000.00	
	เงินเบิกเกินบัญชีธนาคาร (Bank Overdraft)	201		200,000.00
	ซื้อตั๋วเงินรับ 5 ใบเป็นเงินเชื่อ			
		บาท	200,000.00	200,000.00
5	ธนาคาร/เงินฝากธนาคาร (Bank)	102	60,000.00	
	เงินสด (Cash)	101		60,000.00
	นำเงินสดฝากธนาคาร			
		บาท	60,000.00	60,000.00
7	ค่าเช่า (Rent Expense)	502	1,200.00	
	สินค้าคงเหลือ (Inventories)	104	500.00	
	เงินสด (Cash)	101		1,700.00
	จ่ายค่าเช่าร้านและซื้อวัสดุเป็นเงินสด			
		บาท	1,700.00	1,700.00
10	ธนาคาร/เงินฝากธนาคาร (Bank)	102	3,000.00	
	รายได้ค่าเช่า (Rent Revenue)	401		3,000.00
	รับรายได้ค่าเช่ารถ ฝากธนาคาร			
		บาท	3,000.00	3,000.00

รูปที่ 4.1 ข้อมูลสมุดรายวันของเดือนมีนาคม

จากนั้นเมื่อตรวจสอบข้อมูลอย่างละเอียดแล้ว ก็ทำการคลิกปุ่ม “ส่ง” ไปคำนวณงบการเงิน โดยผลลัพธ์ของการคำนวณ เป็นดังต่อไปนี้

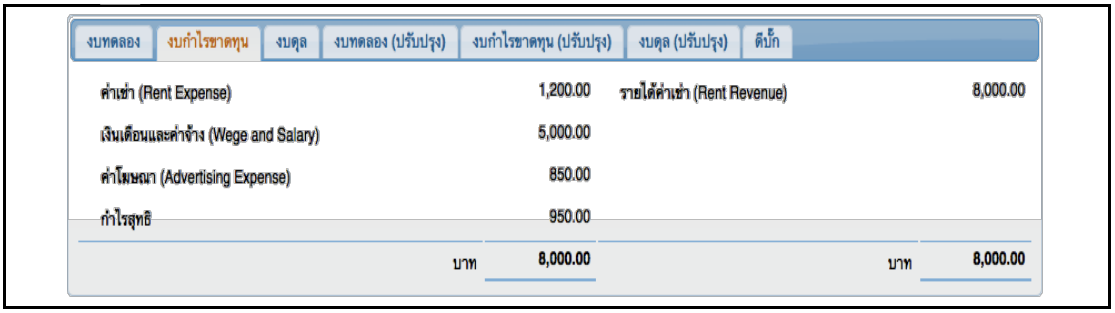


งบทดลอง | งบกำไรขาดทุน | งบดุล | งบทดลอง (ปรับปรุง) | งบกำไรขาดทุน (ปรับปรุง) | งบดุล (ปรับปรุง) | ดีบั๊ก

ลำดับ	รายการ	ประเภท	เดบิต		เครดิต	
1.	เงินสด (Cash)	101	31,800	-		
2.	ธนาคาร/เงินฝากธนาคาร (Bank)	102	25,500	-		
3.	สินค้าคงเหลือ (Inventories)	104	500	-		
4.	ตั๋วเงินรับ (Notes Receivable)	105	200,000	-		
5.	เงินเบิกเกินบัญชีธนาคาร (Bank Overdraft)	201			160,000	-
6.	เจ้าหนี้ (Accounts Payable)	202			850	-
7.	ทุน (Capital)	301			100,000	-
8.	ถอนใช้ส่วนตัว (Drawing)	302	4,000	-		
9.	รายได้ค่าเช่า (Rent Revenue)	401			8,000	-
10.	เงินเดือนและค่าจ้าง (Wege and Salary)	501	5,000	-		
11.	ค่าเช่า (Rent Expense)	502	1,200	-		
12.	ค่าโฆษณา (Advertising Expense)	503	850	-		
		รวม	268,850	-	268,850	-

รูปที่ 4.2 แสดงรายงานงบทดลอง

จากภาพจะเห็นได้ว่า ด้านเดบิตเท่ากับด้านเครดิต นั่นหมายถึง งบทดลองลงตัว ส่งผลให้การนำข้อมูลจากงบทดลองไปคำนวณหางบกำไรขาดทุน และงบดุล สามารถปิดงบได้ลงตัวตามไปด้วยดังรูป



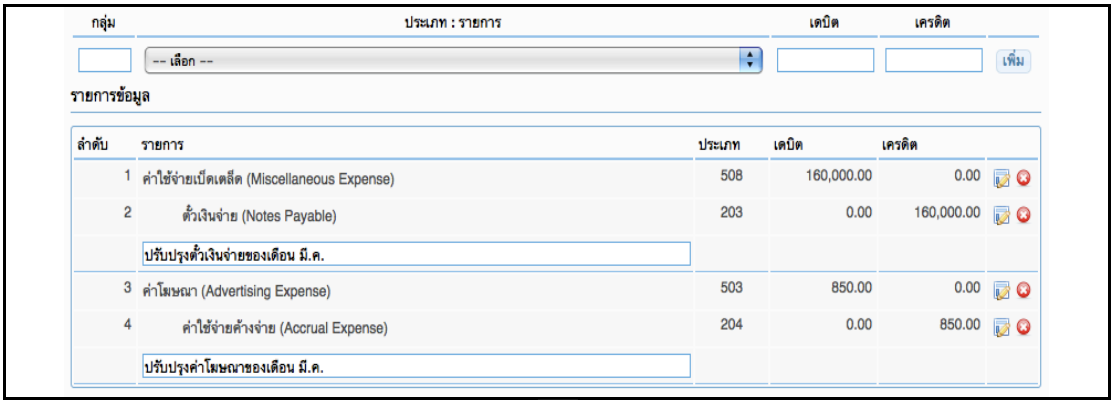
งบทดลอง | งบกำไรขาดทุน | งบดุล | งบทดลอง (ปรับปรุง) | งบกำไรขาดทุน (ปรับปรุง) | งบดุล (ปรับปรุง) | ดีบั๊ก

ค่าเช่า (Rent Expense)	1,200.00	รายได้ค่าเช่า (Rent Revenue)	8,000.00
เงินเดือนและค่าจ้าง (Wege and Salary)	5,000.00		
ค่าโฆษณา (Advertising Expense)	850.00		
กำไรสุทธิ	950.00		
บาท	8,000.00	บาท	8,000.00

รูปที่ 4.3 แสดงรายงานงบกำไรขาดทุน

รูปที่ 4.4 แสดงรายงานงบดุล

จากการทดสอบพบว่าการคำนวณงบการเงินทั้ง 3 งบ สามารถคำนวณได้อย่างถูกต้อง เมื่อมีการบันทึกรายบัญชีเพื่อปรับปรุงระบบก็สามารถบันทึก และคำนวณออกมาเป็นงบหลังปรับปรุงได้



รูปที่ 4.5 แสดงการบันทึกการปรับปรุงงบการเงิน

เมื่อได้ปรับปรุงงบการเงินแล้ว ระบบจะทำการคำนวณงบการเงินหลังปรับปรุงออกมาให้ทั้ง 3 งบ ดังนี้

งบทดลอง | งบกำไรขาดทุน | งบดุล | งบทดลอง (ปรับปรุง) | งบกำไรขาดทุน (ปรับปรุง) | งบดุล (ปรับปรุง) | ดีบั๊ก

ลำดับ	รายการ	ประเภท	เดบิต		เครดิต	
1.	เงินสด (Cash)	101	31,800	-		
2.	ธนาคาร/เงินฝากธนาคาร (Bank)	102	25,500	-		
3.	สินค้าคงเหลือ (Inventories)	104	500	-		
4.	ตั๋วเงินรับ (Notes Receivable)	105	200,000	-		
5.	เงินเบิกเกินบัญชีธนาคาร (Bank Overdraft)	201			160,000	-
6.	เจ้าหนี้ (Accounts Payable)	202			850	-
7.	ตั๋วเงินจ่าย (Notes Payable)	203			160,000	-
8.	ค่าใช้จ่ายค้างจ่าย (Accrual Expense)	204			850	-
9.	ทุน (Capital)	301			100,000	-
10.	ถอนใช้ส่วนตัว (Drawing)	302	4,000	-		
11.	รายได้ค่าเช่า (Rent Revenue)	401			8,000	-
12.	เงินเดือนและค่าจ้าง (Wege and Salary)	501	5,000	-		
13.	ค่าเช่า (Rent Expense)	502	1,200	-		
14.	ค่าโฆษณา (Advertising Expense)	503	1,700	-		
15.	ค่าใช้จ่ายเบ็ดเตล็ด (Miscellaneous Expense)	508	160,000	-		
	รวม		**429,700**	-	**429,700**	-

รูปที่ 4.6 แสดงงบทดลองหลังปรับปรุง

งบทดลอง | งบกำไรขาดทุน | งบดุล | งบทดลอง (ปรับปรุง) | งบกำไรขาดทุน (ปรับปรุง) | งบดุล (ปรับปรุง) | ดีบั๊ก

ค่าเช่า (Rent Expense)	1,200.00	รายได้ค่าเช่า (Rent Revenue)	8,000.00
เงินเดือนและค่าจ้าง (Wege and Salary)	5,000.00	ขาดทุนสุทธิ	159,900.00
ค่าโฆษณา (Advertising Expense)	1,700.00		
ค่าใช้จ่ายเบ็ดเตล็ด (Miscellaneous Expense)	160,000.00		
บาท	**167,900.00**	บาท	**167,900.00**

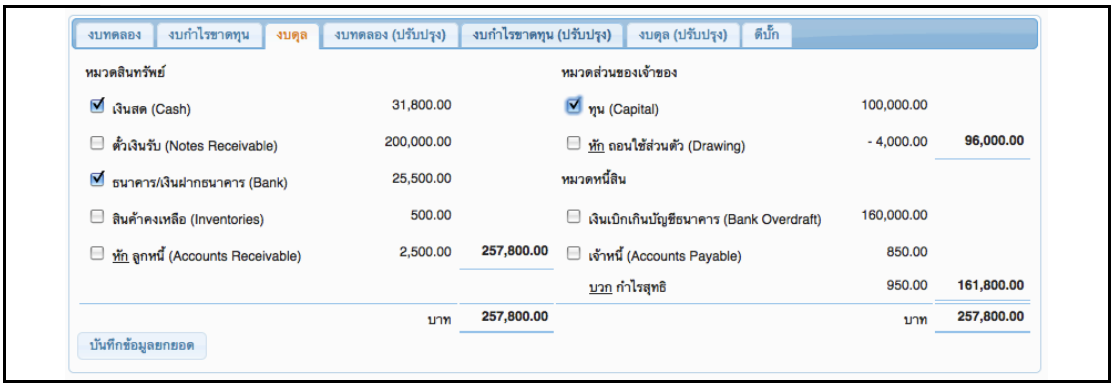
รูปที่ 4.7 แสดงงบกำไรขาดทุนหลังปรับปรุง

งบทดลอง | งบกำไรขาดทุน | งบดุล | งบทดลอง (ปรับปรุง) | งบกำไรขาดทุน (ปรับปรุง) | งบดุล (ปรับปรุง) | ดีบั๊ก

หมวดสินทรัพย์			หมวดส่วนของเจ้าของ		
เงินสด (Cash)	31,800.00		ทุน (Capital)	100,000.00	
ตั๋วเงินรับ (Notes Receivable)	200,000.00		หัก ถอนใช้ส่วนตัว (Drawing)	- 4,000.00	**96,000.00**
ธนาคาร/เงินฝากธนาคาร (Bank)	25,500.00		หมวดหนี้สิน		
สินค้าคงเหลือ (Inventories)	500.00		เงินเบิกเกินบัญชีธนาคาร (Bank Overdraft)	160,000.00	
หัก ลูกหนี้ (Accounts Receivable)	2,500.00	**257,800.00**	เจ้าหนี้ (Accounts Payable)	850.00	
			ตั๋วเงินจ่าย (Notes Payable)	160,000.00	
			ค่าใช้จ่ายค้างจ่าย (Accrual Expense)	850.00	
			หัก ขาดทุนสุทธิ	- 159,900.00	**161,800.00**
	บาท	**257,800.00**		บาท	**257,800.00**

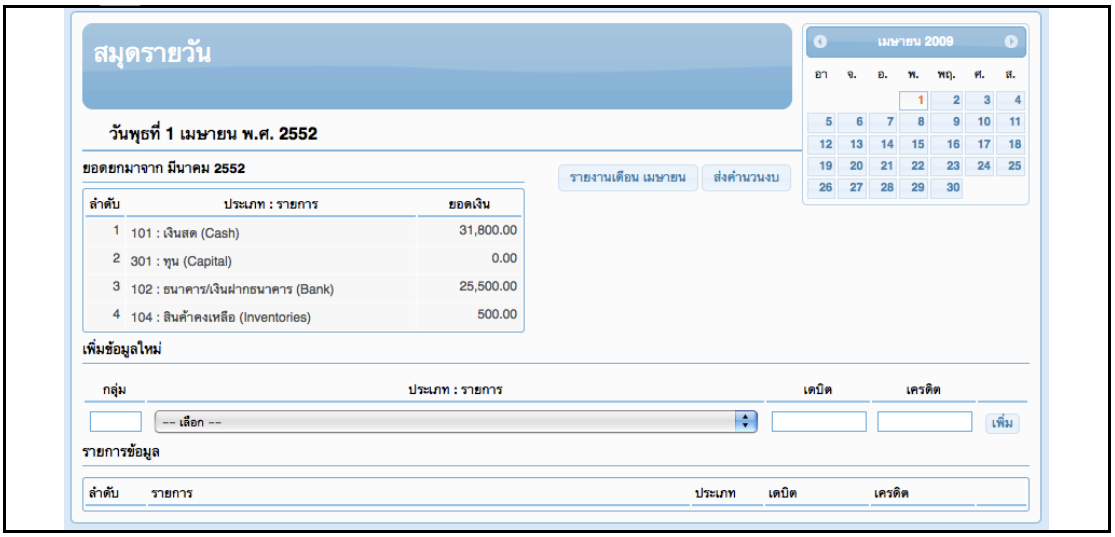
รูปที่ 4.8 แสดงงบดุลหลังปรับปรุง

หลังจากจบการทำบัญชีแล้ว สามารถเลือกรายการบัญชีบางรายการให้สามารถยกยอดไปแสดงผลในเดือนหน้าได้ โดยการเลือกจากงบดุลดังรูป



รูปที่ 4.9 แสดงการเลือกรายการยอดยกไป

เมื่อเลือกได้แล้ว ให้ผู้ใช้กดปุ่มเพื่อทำการยกยอดที่ได้ทำการเลือกเอาไว้ไปแสดงผลในเดือนถัดไปดังรูป



รูปที่ 4.10 แสดงการเลือกรายงานการยกยอด

จากนั้นผู้วิจัยได้นำเอาส่วนที่เป็นบริการ อัพโหลดไปไว้ที่เซิร์ฟเวอร์ที่การสื่อสารแห่งประเทศไทย เพื่อทดสอบการใช้งานจริง ผลคือเครื่องที่ติด Firewall ไม่สามารถเรียกได้ จำเป็นต้องมีการ Config ระบบปฏิบัติการเสียก่อน ส่วนเครื่องที่ไม่ติดอะไร สามารถเรียกใช้งานได้ปกติ และหากมีการเรียกบริการของระบบบัญชีเลย โดยไม่ได้เขียนโปรแกรม เพื่อส่งพารามิเตอร์ ก็ไม่สามารถใช้งานได้เช่นกัน เมื่อมีการเปลี่ยนบราวเซอร์ในการเรียกใช้ พบว่าสามารถเรียกใช้งานได้ทุกบราวเซอร์ต่างกันเพียงความเร็วในการใช้งานเท่านั้น

เมื่อได้ทำการทดสอบ และสรุปผลการใช้งานแล้ว ผู้พัฒนาได้นำเอารูปแบบการทำงานไปให้กับโปรแกรมเมอร์ท่านอื่นอีก 3 ท่าน ทดสอบโปรแกรมในส่วนของ Web Services เพื่อดูว่าต้องพัฒนาตรงส่วนไหนเพิ่มเติม โดยหลังจากอธิบายหลักการทำงาน และนำเอาส่วนของซอร์สโค้ดโปรแกรมไปศึกษาวิธีการทำงาน ก็ได้ให้โปรแกรมเมอร์ทั้ง 3 ท่าน ได้วิเคราะห์โปรแกรมออกมา โดยการวิเคราะห์โปรแกรมจะอยู่ในรูปของแบบสอบถาม โดยมีรูปแบบดังนี้

แบบสอบถามการทดสอบโปรแกรม

คำถาม			
1. เข้าใจรูปแบบการทำงานของฟังก์ชั่น ที่ใช้เป็น Web Services	เข้าใจยาก	พอใช้	เข้าใจง่าย
2. เข้าใจกระบวนการเขียนฟังก์ชั่น รับค่าผลการคำนวณจาก Web Services	เข้าใจยาก	พอใช้	เข้าใจง่าย
3. ต้องมีความรู้ ความเข้าใจระบบบัญชีก่อนการเรียกใช้ Web Services	มาก	ปานกลาง	น้อย
4. ความยาก/ง่าย ต่อการนำไปประยุกต์ใช้งาน	ยาก	ปานกลาง	ง่าย
5. แนวคิดของโปรแกรมกับการพัฒนาต่อไปยังโมดุล หรือประเภทงานอื่น ๆ			
6. ข้อเสนอแนะ			

เมื่อไปมอบให้กับโปรแกรมเมอร์ทั้ง 3 คนใช้งาน จึงสามารถสรุปผลการทดลองได้ดังนี้

โปรแกรมเมอร์คนที่ 1

คำถาม			
1. เข้าใจรูปแบบการทำงานของฟังก์ชั่น ที่ใช้เป็น Web Services	เข้าใจยาก	/	เข้าใจง่าย
2. เข้าใจกระบวนการเขียนฟังก์ชั่น รับค่าผลการคำนวณจาก Web Services	/	พอใช้	เข้าใจง่าย
3. ต้องมีความรู้ ความเข้าใจระบบบัญชีก่อนการเรียกใช้ Web Services	มาก	/	น้อย
4. ความยาก/ง่าย ต่อการนำไปประยุกต์ใช้งาน	ยาก	/	ง่าย
5. แนวคิดของโปรแกรม กับการพัฒนาต่อไปยังโมดุล หรือประเภทงานอื่น ๆ จากการทดสอบการทำงานของ Web Service ของระบบบัญชีพบว่า มีความจำเป็นต้องมีความรู้ ทางบัญชีเท่ากับการพัฒนาโปรแกรมแบบปกติ แต่ในการนำไปใช้กับระบบงานอื่น พบว่าเป็น แนวทางการพัฒนาที่ดี เพราะหากทุก ๆ โปรแกรมมีรูปแบบการทำงานบนส่วนกลางของระบบ การบำรุงรักษาจะทำได้ง่าย และการพัฒนาเพิ่มเติม ก็สามารถทำได้ เป็นอย่างดี			
6. ข้อเสนอแนะ อาจเป็นเพราะเป็นการนำมาประยุกต์ใช้เป็นครั้งแรก ทำให้การทำงานโดยรวมเข้าใจยาก ผู้พัฒนาจำเป็นต้องมีความรู้ความชำนาญในภาษาที่จะนำมาใช้เป็นอย่างดี			

โปรแกรมเมอร์คนที่ 2

คำถาม			
1. เข้าใจรูปแบบการทำงานของฟังก์ชั่นที่ใช้เป็น Web Services	เข้าใจยาก	/	เข้าใจง่าย
2. เข้าใจกระบวนการเขียนฟังก์ชั่น รับค่าผลการคำนวณจาก Web Services	เข้าใจยาก	/	เข้าใจง่าย
3. ต้องมีความรู้ ความเข้าใจระบบบัญชีก่อนการเรียกใช้ Web Services	มาก	/	น้อย
4. ความยาก/ง่าย ต่อการนำไปประยุกต์ใช้งาน	ยาก	/	ง่าย
5. แนวคิดของโปรแกรม กับการพัฒนาต่อไปยังโมดุล หรือประเภทงานอื่น ๆ เป็นแนวคิดการทำงานที่มีโอกาสเติบโต เพราะหลายองค์กรก็นำ Web Service มาประยุกต์ใช้อยู่ แนวคิดของโปรแกรมนี้อาจทำให้นักพัฒนาต้องปรับตัวในการพัฒนาบ้าง แต่ไม่มากเท่าไร			
6. ข้อเสนอแนะ หากจะนำไปใช้จริง อาจต้องปรับรูปแบบการทำงานให้สอดคล้องกับความเป็นจริงมากขึ้น โดยยึดถือผู้พัฒนาระดับทั่วไปเป็นหลัก โดยการทำให้ง่ายต่อความเข้าใจมากขึ้น			

โปรแกรมเมอร์คนที่ 3

คำถาม			
1. เข้าใจรูปแบบการทำงานของฟังก์ชั่น ที่ใช้เป็น Web Services	เข้าใจยาก	/	เข้าใจง่าย
2. เข้าใจกระบวนการเขียนฟังก์ชั่น รับค่าผลการคำนวณจาก Web Services	/	พอใช้	เข้าใจง่าย
3. ต้องมีความรู้ ความเข้าใจระบบบัญชีก่อนการเรียกใช้ Web Services	มาก	ปานกลาง	/
4. ความยาก/ง่าย ต่อการนำไปประยุกต์ใช้งาน	ยาก	ปานกลาง	/
5. แนวคิดของโปรแกรม กับการพัฒนาต่อไปยังโมดุล หรือประเภทงานอื่น ๆ การนำเอาการทำงานทั้งหมดไปรวมกัน แล้วเรียกโดยใช้ Web Service จะเป็นประโยชน์ต่อการพัฒนาโปรแกรมในระดับ Enterprise มากขึ้น เพราะโครงสร้างของการทำงาน ง่ายต่อการจัดระเบียบ พัฒนาและปรับปรุงเพิ่มเติม			
6. ข้อเสนอแนะ Web Services สำหรับระบบบัญชีนั้น ยังไม่ยืดหยุ่นต่อการทำงานมากพอ อาจต้องมีการปรับปรุงให้ใช้งานได้ง่ายขึ้น และออกแบบการทำงานให้ง่ายต่อความเข้าใจ			

วิธีการที่โปรแกรมเมอร์แต่ละท่านได้นำไปประยุกต์ หรือทดลองใช้มีความแตกต่างกัน โดยได้มีการสัมภาษณ์ถึงวิธีการของแต่ละท่านไว้ดังนี้

โปรแกรมเมอร์ท่านที่ 1 ได้นำเอาโปรแกรมทั้งหมด ไปเขียนเลียนแบบโปรแกรมของผู้พัฒนา โดยใช้ภาษา PHP แต่โปรแกรมเมอร์ท่านที่ 1 ไม่ได้มีความชำนาญในตัวภาษามาก ทำให้เกิดความลำบากในการศึกษาส่วนของโค้ดโปรแกรมระบบบัญชี ในตัวของโปรแกรมของโปรแกรมเมอร์ ได้ทำการทดสอบเบื้องต้นพบว่ามีความจำเป็นต้องศึกษาระบบการทำงานของบัญชีเพิ่มเติม ทำให้มีการทดสอบนานกว่าที่ได้ตกลงเวลากันไว้ และได้สรุปว่าโครงสร้างการทำงานของ Web Service for Accounting System นั้น เป็นแนวคิดที่สร้างมาเพื่อการต่อยอดโปรแกรม โดยระบุได้ตรงกับความต้องการของผู้พัฒนา

โปรแกรมเมอร์ท่านที่ 2 เป็นผู้ที่มีความเชี่ยวชาญภาษา PHP ได้ทำเอาโปรแกรมของผู้พัฒนาไปทดสอบ โดยการส่งพารามิเตอร์ไป และดูผลลัพธ์จากการรายงานในลักษณะของ Text

Mode แสดงผลผ่าน Web Browser จากนั้นพบว่าการทำงานทำงานได้ดี แต่ไม่ยืดหยุ่น เพราะการทำงานมีเพียง 1 ฟังก์ชั่น และนำเอามาแยกเองทีหลัง แต่เป็นแนวคิดที่สามารถ นำไปต่อยอดได้

โปรแกรมเมอร์ท่านที่ 3 เป็นผู้ที่ได้นำเอาโปรแกรมระบบบัญชีไปใช้จริงกับเว็บของบริษัทของตนเอง และได้มีการปรับปรุงบางส่วนของการทำงานให้สอดคล้องกับการทำงานมากขึ้น และพบว่าขั้นตอนการพัฒนานั้น ยากกว่าปกติอยู่บ้าง เพราะเป็นเทคโนโลยีและรูปแบบการเขียนโปรแกรมที่ไม่คุ้นเคย แต่ก็สามารถนำเอาไปใช้งานจริงได้

ข้อสรุปจากโปรแกรมเมอร์ทั้ง 3 คนสามารถสรุปผลโดยรวมได้ว่าระบบการให้บริการทางเว็บสำหรับระบบบัญชีนั้น ยังคงใช้งานได้ไม่คล่องตัวเท่าที่ควร เนื่องด้วยมีองค์ประกอบอยู่หลายประการที่นำเอามาใช้ในการทดสอบด้วย โดยประกอบเป็นความรู้เรื่องของการเข้า รหัสโปรแกรมการพัฒนาภาษา PHP หรือการศึกษาการออกงบการเงิน ทั้งหมดล้วนเป็นองค์ประกอบที่ใช้ในการพัฒนาโปรแกรมทั้งสิ้น โดยผู้พัฒนาได้บอกถึงข้อจำกัดดังกล่าวไว้ก่อน ที่จะนำเอาโปรแกรมมาให้โปรแกรมเมอร์ได้ใช้งาน แต่การทดสอบก็ผ่านไปไปได้ด้วยดี

มองจากมุมของนักพัฒนาพบว่า เป็นการพัฒนาโปรแกรมที่แปลกใหม่ เนื่องจากในครั้งแรกของการใช้งาน ไม่รู้สึกถึงความต่างของการเรียกใช้ Web Service กับการเขียนโดยใช้ระบบเรียกข้อมูลธรรมดา แต่ในการเขียนโปรแกรมขึ้นมาเพิ่มเติม เพื่อเรียกใช้ Web Service นั้นสามารถเรียกได้เลย และไม่จำเป็นต้องรื้อโค้ดออกมาดูด้วย เนื่องจาก Web Service ใช้ภาษา XML ในการติดต่ออยู่แล้ว จึงไม่ใช่ปัญหาของการเรียกใช้งาน ทำให้เกิดความสะดวกสบาย และข้อจำกัดน้อยลง ทำให้สามารถเขียนโปรแกรมได้ตามที่ต้องการมากขึ้น และใช้เวลาน้อยลง

4.4 อภิปรายผล

จากการทดสอบใช้งานระบบ การให้บริการทางเว็บสำหรับระบบบัญชี ในหลาย ๆ รูปแบบ พบว่าสามารถใช้งานได้อย่างมีประสิทธิภาพ พร้อมทั้งลดปัญหาความผิดพลาดจากตัวบุคคล (Human Error) ได้เป็นอย่างดี จากการที่ตัวโปรแกรมเป็นเว็บไซต์ ทำให้ไม่มีข้อจำกัดในเรื่องของระบบปฏิบัติการ สามารถเรียกใช้จากสิ่งแวดล้อมที่ต่างกันได้อย่างไม่มีปัญหา จากจุดเริ่มต้นนี้ระบบการให้บริการทางเว็บสำหรับระบบบัญชีนี้จะเอื้อต่อการนำไปศึกษา หาแนวทางการพัฒนาเพื่อเอา SOA เข้ามาประยุกต์ใช้จริงในองค์กร เพราะหากมีการนำแนวคิดนี้ไปศึกษาในระบบระบบหนึ่ง อาจไม่จำเป็นต้องเป็นโปรแกรมใหญ่ ๆ โปรแกรมเดียว แล้วแตกเป็นโมดุลเล็ก ในการพัฒนาแต่อาจจะสามารถแยกกันพัฒนาได้ตามความถนัดของนักพัฒนา อีกทั้งยังทำงานร่วมกันได้อย่างมีประสิทธิภาพ โดยไม่ต้องปรับตัวเพื่อเรียนรู้เทคโนโลยีของบริษัทหุ้นส่วนเลย

บทที่ 5
บทสรุป

ระบบการจัดการทางด้านการเงิน เป็นระบบที่องค์กรเกือบทุกแห่งมีความจำเป็นต้องให้ความสำคัญ เพราะระบบนี้ในส่วนใหญ่ไม่ได้ให้มีการดูแล หรือพัฒนาระบบรองรับ ได้เท่าที่ควร เนื่องจากเป็นส่วนที่หลาย ๆ องค์กรจำเป็นอยู่แล้ว และไม่ได้มีการเปลี่ยนแปลงรูปแบบบ่อยๆ หรืออาจจะยกให้กับสำนักงานบัญชีเป็นคนจัดการไปเลย เนื่องจากในปัจจุบัน ได้มีการเอาเทคโนโลยีสารสนเทศเข้ามาใช้ในองค์กรกันอย่างกว้างขวาง จากการนำเอาเทคโนโลยีสารสนเทศมาประยุกต์ใช้กับระบบงาน โดยเฉพาะงานเบื้องหลังขององค์กรเช่น งานทางการเงิน งานทางบัญชี เป็นต้น ทำให้การทำงานในด้านดังกล่าวมีความสะดวกสูงขึ้นมาก แล้วยังรวมไปถึงการลดกระดาษ และลดค่าความผิดพลาดอันเกิดจากมือของมนุษย์ (Human Error) จากงานทางการเงินที่สำคัญทั้ง 2 งาน คือ งานการเงิน และงานทางบัญชี ระบบการทำงานมีความแตกต่างกันอยู่พอสมควร กล่าวคือ งานการเงิน เป็นการจัดสรรการเงินของแต่ละองค์กร รวมไปถึงการแจ้งขอเพิ่ม โอนเพิ่ม โอนลด จัดการกระบวนการ (Process) ของการเงิน ส่งผลให้มีการตัดสินใจของบุคคลากรเข้ามา ทำให้บางกรณีระบบสารสนเทศ หรือโปรแกรมคอมพิวเตอร์ไม่สามารถควบคุมได้ แต่ส่วนงานทางบัญชีนั้น มีมาตรฐานการทำงานเหมือนกันทั่วโลก เพราะหัวใจของการบัญชี คือการจัดรายงานของระบบเงินขององค์กร ทำให้สามารถรู้ความเป็นไปขององค์กรได้ โดยดูจากรายงานทางบัญชีนั่นเอง

เนื่องจากระบบงานทางบัญชีได้เป็นมาตรฐานเดียวกัน ทำให้มีโปรแกรมทางงานบัญชีออกมามากมาย ทั้งในรูปแบบของโปรแกรมสำเร็จรูป และการเขียนโปรแกรมขึ้นมาใหม่ ทั้ง 2 แบบมีความแตกต่างในเชิงการใช้งานอยู่คือ โปรแกรมสำเร็จรูปหาซื้อได้ง่าย มีการบำรุงรักษาจากผู้พัฒนาต่อเนื่อง แต่ไม่สามารถแก้ไขโปแกรมได้ ส่วนการพัฒนาขึ้นมา ต้องใช้เวลาในการพัฒนาโดยระยะเวลาขึ้นอยู่กับความซับซ้อนของระบบ และขึ้นกับนักพัฒนาด้วย ข้อดีคือสามารถเขียนโปรแกรมให้เข้ากับพฤติกรรมการใช้งานขององค์กรนั้น ๆ ได้ เนื่องด้วยความเข้ากันได้ ทำให้ในปัจจุบันได้มีการจ้างเขียนโปรแกรมทางงานบัญชีเยอะขึ้น ทำให้ Software House หรือ โปรแกรมเมอร์ที่รับงานอยู่เป็นประจำต้องเขียนโปรแกรมซ้ำเดิมตลอดเวลา เวลาสร้างฟังก์ชั่น ขึ้นมาส่วนใหญ่โปรแกรมเมอร์จะใช้เป็นเพียงผู้เดียว ไม่สามารถต่อยอดได้ หรืออาจต้องใช้เวลาในการศึกษาโค้ด

โดยงานวิจัยชิ้นนี้ มีจุดมุ่งหมายที่จะลดระยะเวลาการพัฒนาโปรแกรมระบบบัญชี ซึ่งจะนำเอาแนวคิดของ SOA (Service-Oriented Architecture) มาประยุกต์ใช้กับระบบงาน โดยการเขียนโปรแกรมทางบัญชี สามารถเขียนได้หลายแบบ แต่ในงานวิจัยนี้ได้นำเอาเทคโนโลยีเว็บเซอร์วิสมาใช้ในการพัฒนา เพราะมีการติดต่อกันระหว่าง Server-Client ด้วยภาษา xml ทำให้สามารถพัฒนาด้วยภาษาอะไรก็ได้ แล้วยังสามารถนำกลับมาใช้ซ้ำได้อีก โดยไม่จำเป็นต้องเขียนฟังก์ชั่นขึ้นมาใหม่อีกทั้งด้วยความสามารถของเว็บเซอร์วิส และแนวคิดของ SOA ทำให้สามารถพัฒนาระบบสารสนเทศขององค์กรได้ตั้งแต่องค์กรเล็กๆ ไปจนถึงองค์กรขนาดใหญ่ได้

ขั้นตอนการพัฒนางานวิจัยชิ้นนี้ เริ่มต้นจากการศึกษางานทางบัญชีก่อน โดยศึกษาจากการคำนวณด้วยมือก่อน เพื่อสร้างความเข้าใจในระบบงาน จากนั้นได้ไปศึกษากับสำนักงานบัญชีเพื่อดูว่าใช้งานจริงเป็นแบบไหน มีขั้นตอนอย่างไร และได้ทำการศึกษาโปรแกรมสำเร็จรูปทางบัญชี เพื่อดูการทำงานว่าจากการทำด้วยมือมาสู่รูปแบบของโปรแกรม ต้องปรับตัวอย่างไรบ้าง จากการศึกษาได้เข้าใจถึงกระบวนการทำงานแต่ละ Process ว่ามีการทำงานอย่างไร และขอบเขตถึงไหน จากนั้นได้ทำการเลือกเทคโนโลยีที่จะนำมาใช้ในการพัฒนา โดยผู้วิจัยได้ตัดสินใจใช้ภาษา PHP ในการพัฒนาเนื่องจากมีความยืดหยุ่นในการพัฒนาสูง และยังสามารถพัฒนาบนระบบปฏิบัติการที่หลากหลายได้ โดยไม่มีปัญหาเรื่องความเข้ากันได้ และผู้ใช้ก็มีเพียง Web Browser เช่น Internet Explorer (IE) Firefox Safari ก็สามารถใช้งานได้ และผู้วิจัยได้เลือก NuSoap มาใช้ในการพัฒนาเว็บเซอร์วิสสำหรับ PHP

ขั้นตอนการทดสอบระบบ คือได้เอาข้อมูลทางบัญชีตัวมาทดลองใช้งานดู เป็นระยะเวลา 1 เดือน ซึ่งเพียงพอต่อการออกงบการเงินพื้นฐานได้ ได้แก่ งบทดลอง งบกำไรขาดทุน และงบดุล และได้ทดลองในระบบ โดยการจำลอง Server ขึ้นมาก่อนภายในเครื่องตัวเดียวกัน จากนั้นได้นำเอาส่วนที่ให้บริการไปไว้ที่ Server จริง โดยตัวของ Server ตั้งอยู่ที่ กสท. (การสื่อสารแห่งประเทศไทย) และทำการเรียกใช้งานดูผลการทำงานคือ สามารถเรียกใช้งานได้ทั้ง 2 กรณี และมีผลการคำนวณงบที่ถูกต้อง จากนั้นเมื่อมาดูส่วนในฝั่ง Client ได้มีการพัฒนาเพียงส่วนที่กรอกข้อมูล และส่วนในการแสดงผลเท่านั้น ทำให้หากมีการพัฒนาโปรแกรมขึ้นมาใหม่ ก็สามารถเรียกใช้บริการทางบัญชีที่เก็บไว้ทาง Server ได้ทันที

5.1 สรุปผลการวิจัย

จากการวิจัยสามารถสรุปได้ดังนี้

1. ในการใช้โปรแกรมสำเร็จรูปในการทำงานบัญชีกับโปรแกรมทางบัญชีที่ทางผู้วิจัยได้พัฒนาขึ้นมา พบว่ามีผลลัพธ์การทำงานที่ถูกต้อง 100 เปอร์เซ็นต์ทั้ง 2 โปรแกรม แต่ในมุมของ

ผู้พัฒนาได้มีรูปแบบการเขียนโปรแกรมที่แตกต่างกัน ในกรณีของการเขียนโปรแกรมทางบัญชีขึ้นมาใช้งานแบบดั้งเดิมนั้น ข้อมูลและการคำนวณทั้งหมดจะเก็บรวมอยู่ในที่ ๆ เดียวเป็นส่วนใหญ่ ทำให้เวลาพัฒนาโปรแกรมต้องพัฒนาขึ้นมาทั้งหมด และหากมีการพัฒนาเพิ่มเติมต้องลงไปแก้ไขที่ Source Code เป็นหลัก แต่ในกรณีพัฒนาโดยยึดหลักของ SOA นั้น ได้แบ่งการพัฒนาเป็น 2 ส่วนชัดเจน คือส่วนอินพุตและแสดงผล กับส่วนของบริการ เมื่อติดปัญหาที่ส่วนใดส่วนหนึ่ง สามารถดึงมาแก้ไขเป็นส่วน ๆ ได้ ทำให้ลดเวลาการทำงานลงเป็นอย่างมาก

2. จากการพัฒนาทำให้ทราบถึงประสิทธิภาพของการส่งข้อมูล ทางผู้พัฒนาได้ใช้ระบบปฏิบัติการในการพัฒนาถึง 3 ระบบได้แก่ (1) Windows XP (2) Windows Vista Ultimate Edition (3) Macintosh OS X Leopard 10.5.6 เมื่อมีการใช้ทดลองโปรแกรม พบว่าไม่มี Browser ใดที่แสดงผล ผิดเพี้ยน แต่ในการคำนวณและส่งกลับมาแสดงผล สามารถเรียงตามความรวดเร็วในการแสดงผล ได้ดังนี้ Safari (OSX) -> FireFox 3.0 (OSX) -> FireFox 3.0 (Vista) -> Internet Exploror 6.0 (XP) -> Internet Exploror 7.0 (Vista) ทำให้สรุปได้ว่าประสิทธิภาพในการส่งข้อมูลขึ้นอยู่กับ Browser โดยหากมีการพัฒนาต่อ ทางผู้วิจัยแนะนำให้ใช้ Firefox ในการทดสอบ วิเคราะห์ และแสดงผล เนื่องด้วยความเสถียรของตัว Browser และแสดงผลได้ถูกหลักของเว็บไซต์มากที่สุด

3. จากการทดสอบภายในเครื่องเดียวกันทั้งฝั่ง Client และ Server ก็พบว่ามีการใช้งานได้ดี และเมื่อนำเอาส่วน Service ไปไว้ที่ Server จริง และทดสอบเรียกใช้งาน ก็พบว่าสามารถใช้งานได้ และความเร็วของการทำงานก็ขึ้นอยู่กับความเร็วของอินเตอร์เน็ตของผู้ใช้ และส่งผลให้ไม่ว่าผู้ใช้งานจะอยู่ที่ไหนก็ตาม ก็สามารถทำงานทางบัญชีได้ ขอเพียงมีอินเตอร์เน็ต

4. จากการใช้งานในรูปแบบของบัญชี พบว่าไม่แตกต่างจากการพัฒนาตามปกติที่มีการคำนวณในเครื่องเดียวกันแต่อย่างใด เพียงแต่การรับส่งข้อมูลต่างกันเท่านั้น ทำให้ผู้ใช้ไม่มีความจำเป็นต้องปรับตัวเพื่อเข้าหาระบบการทำงานที่ใช้ Web Service เลย

5.2 การประยุกต์งานวิจัย

งานวิจัยชิ้นนี้เป็นการนำเอาแนวคิดที่ยังเป็นสิ่งที่ใหม่สำหรับนักพัฒนามาประยุกต์ใช้งานกับการทำงานแบบพื้นฐาน ทำให้เกิดการนำเอาแนวคิดของ SOA มาพัฒนาให้เกิดรูปธรรม โดยไม่จำเป็นต้องเอาแนวคิดนี้ ไปใช้ในโปรเจคใหญ่ ๆ เพียงอย่างเดียว และจากแนวคิดการนำเอา SOA มาใช้อย่างจริงจัง จะทำให้การใช้งานเครือข่ายคอมพิวเตอร์มีประสิทธิภาพสูงมากขึ้น ไม่ว่าจะเป็นองค์กร หรือห้างร้านขนาดไหนก็ตาม โดยแนวทางการวิจัยสำหรับนักวิจัยในอนาคต สามารถนำเอาแนวคิด SOA มาประยุกต์ใช้งานกับระบบที่มีความซับซ้อนมากกว่า หรือประยุกต์ การทำแนวคิด

ของ SOA ให้มีประสิทธิภาพสูงขึ้นกว่าเดิม และเมื่อมีการพัฒนาใช้งานจนเกิดความชำนาญ ทำให้การพัฒนาโปรแกรมในระดับเอ็นเตอร์ไพรส์ ที่สามารถเชื่อมต่อหลาย ๆ องค์กรเข้าด้วยกันได้

5.3 แนวทางการพัฒนาต่อ

เนื่องจากระบบการให้บริการงานบัญชีนั้น จะประกอบไปด้วยส่วนของการอินพุต ส่วนของการให้บริการการคำนวณงบ และส่วนของการแสดงผล เมื่อต้องการนำไปพัฒนาต่อจะแยกหัวข้อเพื่อเป็นแนวทางในการพัฒนาดังนี้

ส่วนที่ติดต่อกับผู้ใช้ (Client)

1. ยังไม่สามารถทำงานให้เป็นระบบอัตโนมัติในบางระบบได้ เนื่องจากขึ้นอยู่กับผู้ใช้เป็นหลัก

2. ระบบการแก้ไขข้อมูล ยังต้องไปแก้ไขที่ช่อง Textbox อยู่ หากพัฒนาต่อควรสามารถแก้ไขได้ที่บรรทัดของข้อมูลได้เลย

ส่วนของการให้บริการ (Service)

1. ระบบยังสามารถคำนวณได้เฉพาะงบการเงินพื้นฐานเท่านั้น หากนำไปพัฒนาต่อควรสามารถคำนวณได้ทุกงบ สามารถคำนวณค่าเสื่อมราคาได้ คำนวณภาษีทั้งกลางปีและปลายปีได้

2. ตอนนี้ยังใช้ข้อมูลที่บันทึกเป็น text file อยู่เนื่องด้วยเหตุผลในการนำเสนอ และง่ายต่อการติดตั้ง ถ้าต้องใช้งานจริงควรเปลี่ยนเป็นการใช้ฐานข้อมูลจะมีประสิทธิภาพสูงกว่า

ส่วนของการแสดงผล (Report)

1. ยังไม่สามารถออก Report แบบมาตรฐานได้

จากงานวิจัยนี้ ได้สร้างขึ้นเพื่อรองรับงานทางบัญชีเบื้องต้นอย่างเดียว ถ้านำไปใช้ ควรทำการเพิ่มงานส่วนอื่นเข้าไป เรียกใช้กลุ่มของบริการเช่น งานทางการเงิน ทรัพยากรบุคคล เป็นต้น จะทำให้ระบบดูเป็นรูปเป็นร่างออกมาได้ หรือพัฒนาส่วนของบริการทางการบัญชี ให้สามารถคลอบคลุมชนิดของข้อมูลที่ส่งไปให้ได้มากกว่านี้ เพื่อรองรับการพัฒนาไปเป็นการ ให้บริการบัญชีแบบพานิชย์อิเล็คทรอนิคส์ (E-Commerce) ได้ในอนาคต

รายการอ้างอิง

เชาวลีย์ พงศ์ผาติโรจน์ และวรศักดิ์ ทุมมานนท์. (2004). **Accounting Principle 1**. กรุงเทพฯ : สำนักพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย.

ปาริชาติ มณีมัย. (2551). **หลักการบัญชีขั้นต้น.** กรุงเทพฯ : สำนักพิมพ์โอเดียนสโตร์.

สุธี พงศาสสกุลชัย. (2550). **การพัฒนาระบบด้วยสถาปัตยกรรมเชิงบริการยนเทคโนโลยีของ Web Service**. สำนักพิมพ์เคทีพี.

วิชา ศิริธรรมจักร์. (2549). **Web Programming ด้วย AJAX และ PHP**. สำนักพิมพ์เคทีพี.

อมรรัตน์ โกมลหิรัญ และกรด เจิมจรุง. (2547). **คัมภีร์การโปรแกรมเชิงวัตถุด้วย PHP (PHP-OOP).** สำนักพิมพ์เคทีพี.

Eyhab Al-Masri, Qusay H. Mahmoud. (2008). **Investigating Web Services on the World Wide Web**. Proceedings of the 2008 ACM symposium on Web Services Technology.

Judith Hurwitz, Robin Bloor, Carol Baroudi, Macia Kaufman. (2007). **Service Oriented Architecture for Dummies**. Wiley Publishing Inc.

Keith Ballinger. (2003). **Net Web Services Architecture and Implementation**. Addison Wesley.

Scott Nichol. (2004). **Introduction to NuSoap** . [On-line]. Available : http://www.scottnichol.com/nusoapintro.htm

Thomas Erl. (2006). **Service-Oriented Architecture Concepts, Technology and Design**. Prentice Hall.

Timothy C. Lethbridge and Robert Laganiere. (2005). **Object-Oriented Software Engineering Practical Software Development using UML and JAVA**. McGRAW-HILL International.

Youcef Baghdadi. (2005). **A Web Services-Based Business Interactions Manager to Support Electronic Commerce Applications**. Proceedings of the 2005 ACM symposium on Web Services.

ภาคผนวก ก

ผลงานวิจัยที่นำเสนอในการประชุมทางวิชาการ

การให้บริการทางเว็บสำหรับระบบบัญชี
WEB SERVICE FOR ACCOUNTING SYSTEM

วศิน ตรีสินธุรส
สาขาวิชาวิศวกรรมคอมพิวเตอร์ สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี
ทำการวิจัยเสร็จเรียบร้อย พ.ศ. 2551

บทคัดย่อ

การทำงานในบริษัท หรือองค์กรต่างๆ ในปัจจุบันนั้นล้วนมีความเกี่ยวข้องกับการเงินทั้งสิ้น และการที่จะต้องดูแลสภาวะทางการเงินขององค์กรนั้น จำเป็นต้องมีผู้เชี่ยวชาญเฉพาะทางมาดูแลเป็นพิเศษ โดยผู้ดูแลนั้นเป็นเพียงส่วนหนึ่งของระบบการดูแลทั้งหมด แต่ส่วนที่มีผลต่อการดูแลมากคือตัวโปรแกรมบัญชีขององค์กรนั้น ๆ ผู้วิจัย จึงมีความตั้งใจที่จะสร้างระบบบริการทางเว็บสำหรับช่วยงานผู้เขียนโปรแกรมเพื่อช่วยในการเขียนโปรแกรมสำหรับงานทางบัญชีทำได้สะดวกมากยิ่งขึ้น

ผู้วิจัยได้ทำการพัฒนาระบบบัญชีโดยใช้เทคโนโลยี Web Service โดยทำการสร้างโมดุล ซึ่งประกอบด้วยการคำนวณงบต่างๆ เช่น งบทดลอง งบดุล งบกำไรขาดทุน เป็นต้น จะทำงานโดยการรับค่าเพื่อนำมาประมวลผลของงบการเงินและการส่งผลลัพท์การคำนวนกลับคืนมายังโปรแกรม เตรียมเอาไว้ให้ผู้พัฒนาเอารูปแบบไปพัฒนาระบบบัญชีของตนเองรูปแบบการทำงาน คือ ให้ผู้ใช้ทำการระบุข้อมูลทางการเงินผ่านทางโปรแกรมของผู้ใช้เอง และทำการเรียกใช้การให้บริการทางเว็บสำหรับระบบบัญชี ซึ่งอยู่ที่ฝั่งเซิฟเวอร์ ระบบจะทำการประมวลผล และส่งค่ากลับมาเป็นผลลัพธ์ทางบัญชีที่ต้องการ หรืองบการเงิน ในรูปแบบของ XML เพื่อที่ผู้พัฒนาระบบบัญชี ไม่จำเป็นต้องเขียนระบบการทำงานทั้งหมด แต่เรียกใช้การบริการจากระบบที่ผู้วิจัยพัฒนาขึ้นให้ทำงานร่วมกับโปรแกรมของนักพัฒนาระบบได้ตลอดเวลา เป็นการประหยัดเวลาและเป็นมาตรฐานร่วมกัน

Abstract

Nowadays, working in companies or organizations are entirely related to financial and it is necessary to use some specialist to manage financial conditions of organization. Such specialist is only a part of whole management system but the most affective part of system is accounting application software of those organizations. Therefore, the researcher paid an attention on creating a web service system to help the programmers for more convenience on accounting system programming.

The researcher developed accounting system with Web Service technology by creating a module which consisted of the calculated statements such as trial balances, balance sheets, profit and loss statements etc. This system was operated by receiving all values for processing financial statements and sent the calculated results back to the program which programmers could develop their own accounting systems. Operation format was to let users specify financial data via their own programs and request servers for Web Service. This system could process and sent back financial results or statements as needed in format of XML which the developers did not need to create the entire system. But only request services from this system that developed by the research and operate together with their own developed system for saving times and being co-standard.

ความสำคัญและความเป็นมาของปัญหา

จากรูปแบบการทำงานในองค์กรในปัจจุบัน ไม่ว่าจะเป็นองค์กรขนาดใหญ่ หรือองค์กรขนาดเล็กก็ล้วนมีรูปแบบการบริหารที่คล้ายกัน นั่นคือมีการแบ่งเป็นส่วนงานต่างๆตามหน้าที่ ที่ส่วนงานนนั้นได้รับมา ในแต่ละส่วนงานนั้นย่อมต้องมีผู้ที่เชี่ยวชาญเฉพาะทางเข้ามาดูแล โดยการทำงานแบบนี้จะช่วยให้เจ้าขององค์กร สามารถดูแลบุคลากรภายในองค์กร-ให้มีประสิทธิภาพมากขึ้น ในทุกๆองค์กรจำเป็นต้องมีส่วนงานหนึ่งที่คอยช่วยเรื่องของการจัดการทางการเงิน นั่นคือส่วนงานการเงินและการบัญชี ซึ่งมีรูปแบบการทำงานเหมือนกันทุกๆองค์กร ในการทำบัญชีนั้นมีการคำนวนที่ค่อนข้างซับซ้อนเพื่อให้เกิดความผิดพลาดให้น้อยที่สุด จึงจำเป็นต้องใช้บุคลากรที่จบมาทางงานบัญชีโดยตรงมาควบคุมการทำงาน เพราะผลกระทบของงานบัญชีนั้น มีความสำคัญเป็นอันดับแรกๆ ขององค์กร เพราะเมื่อเกิดความผิดพลาดขึ้นจะเกิดผลกระทบไปยังส่วนอื่นๆขององค์กรเกือบทั้งหมด แต่ในปัจจุบันหลายๆองค์กรกลับให้ความสำคัญน้อยเกินความจำเป็น อาจเป็นเพราะหลายๆสาเหตุดังนี้

1. เพราะตัวงานไม่ได้เกี่ยวกับรูปแบบการทำงานขององค์กรโดยตรง แต่คอยดูแลอยู่เบื้องหลัง
2. เพราะมีรูปแบบการทำงานตายตัว
3. อาจเนื่องมาจากความไว้วางใจของเจ้าของโครงการเองและอาจมีอีกหลายสาเหตุซึ่งขึ้นอยู่กับองค์กรนั้นๆ แต่เนื่องด้วยสาเหตุดังกล่าว อันเป็นเหตุทำให้ส่วนงานบัญชีไม่ได้รับ

การดูแลเท่าที่ควร

ในการทำงานเกี่ยวกับบัญชีนั้น สามารถแบ่งการทำงานได้เป็น การทำงานแบบเดิมโดยใช้บุคลากรทำงานบนสมุดบัญชี โดยการทำงานทั้งหมดทำบนสมุดบันทึกงานของเจ้าหน้าที่ทางบัญชี และจะมีการออกรายงานผ่านทางคอมพิวเตอร์ ข้อดีก็คือ เป็นการทำงานที่เจ้าหน้าที่ไม่จำเป็นต้องปรับตัวมากนัก ทำให้สามารถทำงานได้อย่างมีประสิทธิภาพ แต่ก็ขึ้นอยู่กับบุคลากรท่านนั้นด้วย อีกรูปแบบหนึ่งก็คือ การทำงานโดยใช้โปรแกรมสำเร็จรูปที่มีจำหน่ายอยู่ ข้อดีคือสามารถลดจำนวนกระดาษได้เป็นปริมาณมาก อีกทั้งยังสามารถคำนวนได้อย่างมีประสิทธิภาพและสามารถลดความผิดพลาดที่เกิดจากบุคคลได้ แต่มีข้อจำกัดคือโปรแกรมได้ออกแบบมาไว้ก่อนแล้วซึ่งส่วนใหญ่มีรูปแบบการทำงานที่เป็นกลาง แต่ไม่มีลักษณะเฉพาะ (Feature) ของโปรแกรมซึ่งอาจจะไม่เข้ากับรูปแบบการทำงานขององค์กรนั้น หรืออาจต้องทำงานหลายๆรอบ ส่วนรูปแบบสุดท้ายคือ การเขียนโปรแกรมทางการบัญชีขึ้นมาใหม่โดยผ่านขั้นตอนของวิศวกรรมซอฟแวร์ (Software Engineering) เพื่อให้การออกแบบโปรแกรมทางการบัญชีนั้น สามารถเข้ากับระบบงานขององค์กรนั้นๆให้มากที่สุด วิธีการนี้ เริ่มเป็นที่นิยมในปัจจุบัน เนื่องจากองค์กรใหม่ๆเกิดขึ้นมากมาย รวมไปถึงเทคโนโลยีการเขียนโปรแกรม และอินเตอร์เน็ต มีประสิทธิภาพมากขึ้น ข้อดีคือพนักงานสามารถออกแบบระบบงานเองได้ โดยแจ้งกับทีมงานฝ่ายวิเคราะห์ระบบ (System Analysis) และโปรแกรมมักจะมีความเข้ากันได้กับระบบอื่นๆในองค์กรสูงกว่าวิธีอื่นๆ ข้อจำกัดคือ ต้องใช้เวลาในการออกแบบระบบ ศึกษาระบบงานโดยเฉพาะโปรแกรมเมอร์ หรือนักวิเคราะห์ระบบซึ่งส่วนใหญ่มักไม่มีความรู้เรื่องของงานทางการบัญชีมากเท่าที่ควร ทำให้ต้องใช้เวลาในการศึกษาเพิ่มขึ้นอีก รวมไปถึงเจ้าหน้าที่ทางการบัญชีต้องรอจนกระทั่งกระบวนการการเขียนโปรแกรมทั้งหมดเสร็จสมบูรณ์ โดยขั้นตอนการเขียนระบบงานหนึ่งๆขึ้นมานั้น ประกอบไปด้วย การเก็บข้อมูลการใช้งานเบื้องต้น (Requirement) , การวิเคราะห์และออกแบบระบบ (System Analysis), การเขียนโปรแกรม (Programming), การทดสอบระบบ (Testing) แล้วถึงจะส่งมอบระบบไปยังองค์กรนั้นๆที่ได้ว่าจ้างในการเขียนโปรแกรมหรือระบบงาน ดังรูปที่ 1

รูปที่ 1 ขั้นตอนการอกแบบระบบในรูปแบบของวิศวกรรมซอฟแวร์ (Software Engineering)

จากรูปแบบการทำงานจะพบว่า ถ้าระบบงานที่ได้รับมาจากเจ้าหน้าที่ขององค์กรนั้น เป็นระบบที่มีความซับซ้อนจะทำให้การวิเคราะห์ระบบ รวมไปถึงการเขียนโปรแกรมเป็นไปด้วยความลำบาก เพราะต้องศึกษาศาสตร์อื่นเพิ่มเติม

เมื่อมีการเขียนโปรแกรมหรือระบบงานหนึ่งๆเสร็จแล้ว พบว่าโปรแกรมเมอร์มักไม่ได้เก็บงานที่ทำเอาไว้ หรืออาจจะเก็บไว้เป็น Module ของตัวเองเผื่องานครั้งต่อไป แต่จากการกระทำดังกล่าวจะพบถึงช่องว่างของการเขียนโปรแกรมของโปรแกรมเมอร์ โดยเฉพาะ เมื่อมีการเขียนโปรแกรมซ้ำจะพบว่าข้อจำกัดของการเขียนซ้ำคือ

- ต้องใช้เทคโนโลยีในการเขียนโปรแกรม เป็นเทคโนโลยีเดิม
- ประหยัดเวลาในการเขียนใหม่ แต่ไม่สามารถส่งต่อ Module นั้นๆให้กับคนอื่นได้ หรือไม่ก็สื่อสารได้ลำบาก เพราะส่วนใหญ่ผู้เขียนมักจะเข้าใจเพียงคนเดียว
- ส่วนใหญ่แล้ว มักจะเขียนไปเป็นระเบียบเท่าไหร่ เพราะไม่ได้เป็นส่วนที่ติดต่อกับผู้ใช้ แต่เป็นส่วนที่คนเขียนโปรแกรมใช้เองคนเดียว เลยไม่ได้ให้ความสำคัญเท่าที่ควร

งานทางบัญชีนั้นเป็นงานที่มีมาตรฐานเดียวกันทั่วโลก อาจจะแตกต่างกันเพียงภาษาที่ใช้ในท้องที่เท่านั้น แต่รูปแบบการทำงานโดยรวมเหมือนกันทุกประการ ไม่ว่าจะเป็นการออกงบต่างๆ, คำนวนงบดุล, งบกำไรขาดทุนเป็นต้น แต่เมื่อมีองค์กรใหม่ๆ ก็จำเป็นต้องมีระบบงานบัญชีตามมาอยู่แล้วเป็นเงาตามตัว แต่ต้องมีการเขียนซ้ำใหม่ทุกครั้ง และจำเป็นต้องเป็นเทคโนโลยีเดิมๆ หรือภาษาเดิมๆตลอด ทำให้ไม่เกิดความยืดหยุ่น (Flexibility) ในการเขียนโปรแกรม อีกทั้งเมื่อมีการรับงานใหม่ๆ ที่ทางองค์กรผู้ว่าจ้างได้ต้องการเทคโนโลยีที่ต่างไปจาก Module ที่โปรแกรมเมอร์ได้มีเก็บเอาไว้ ก็มีความจำเป็นต้องเขียนขึ้นมาใหม่อยู่ดี

ดังนั้นจากรูปแบบดังกล่าวพบว่า การเขียนโปรแกรมงานทางการบัญชีในปัจจุบันเป็นที่นิยม แต่ยังคงมีข้อจำกัดอยู่ ทั้งๆที่งานทางการบัญชีเป็นงานที่มีมาตรฐานเหมือนกันทั่วโลก แต่ด้วยข้อจำกัดดังกล่าวทำให้ต้องมีการเขียนโปรแกรมซ้ำๆอยู่หลายครั้ง อันไม่ก่อให้เกิดการเขียนซ้ำ (Reusable) ได้อย่างมีประสิทธิภาพ จึงได้นำเอาเทคโนโลยีเว็บเซอร์วิส (Web Service) มาใช้งานในส่วนที่เป็นระบบบัญชี โดยรูปแบบการทำงาน จะเขียนโปรแกรมส่วนที่เป็นระบบงานหลักของการบัญชีเอาไว้เป็นบริการ (Service) เอาไว้ที่ Server จากนั้นโปรแกรมเมอร์ก็สามารถเรียกเอาบริการงานทางบัญชีมาใช้ได้โดยไม่จำเป็นต้องเขียนรูปแบบงานซ้ำแต่อย่างใด โดยเขียนเฉพาะส่วนที่ใช้ในการรับข้อมูลเท่านั้น โดยประโยชน์โดยตรงของเว็บเซอร์วิสคือ มีการเชื่อมต่อกันด้วยภาษาเอ็กซ์เอ็มแอล (XML) ทำให้เกิดเป็นมาตรฐานเพื่อรองรับการเขียนโปรแกรมในเทคโนโลยีที่หลากหลาย เพราะมาตรฐาน XML เป็นที่ยอมรับไปทั่วโลก และทุกๆภาษาสามารถอ่านแท็ก XML ได้ อีกทั้งในส่วนของเว็บเซอร์วิส ยังสามารถเขียนขึ้นได้จากภาษาหลายภาษาด้วย ทำให้สะดวกแก่การพัฒนาต่อ อีกทั้งยังทำให้เกิดการเขียนซ้ำอย่างมีประสิทธิภาพอีกด้วย

วัตถุประสงค์ของการวิจัย

1. เพื่อให้เกิดมาตรฐานการเขียนโปรแกรมในงานทางบัญชี
2. เพื่อรองรับการพัฒนาต่อของโปรแกรมในรูปแบบเดิม

แต่มีการใช้เทคโนโลยีที่หลากหลาย

3. เพื่อลดระยะเวลาการทำงานของโปรแกรมเมอร์

4. เพื่อใช้ในการประยุกต์ไปสู่การเขียนโปรแกรมในรูปแบบของ การเขียนโปรแกรมเชิงบริการ(Service-Oriented Architecture)

5. เพื่อรองรับการพัฒนาต่อ และเป็นจุดเริ่มต้นของการนำเอาเทคโนโลยีเว็บเซอร์วิสมาประยุกต์ใช้กับรูปแบบการทำงานในปัจจุบัน

ประโยชน์ของการวิจัย

1. สามารถนำเอาเทคโนโลยีเว็บเซอร์วิสมาประยุกต์ใช้ให้เกิดความสะดวกแก่โปรแกรมเมอร์ได้

2. สามารถทำให้เกิดแนวทางการพัฒนาไปสู่รูปแบบโปรแกรมที่หลากหลายมากขึ้น

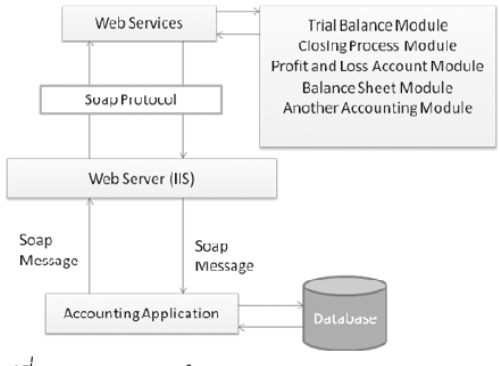
3. สามารถนำเอาไปใช้ในระบบบัญชีได้ทุกๆ ที่ ที่มีการเชื่อมต่ออินเตอร์เน็ตทำให้เกิดความสะดวกในการใช้งานมากขึ้น เพราะการให้บริการอินเตอร์เน็ตในปัจจุบัน ได้มีประสิทธิภาพที่สูงกว่าแต่ก่อนมาก

4. สามารถนำเอาไปวิจัยต่อ เพื่อการเรียนรู้ หรือเพื่อออกแบบระบบใช้งานได้

วิธีการดำเนินวิจัย

ในขั้นตอนแรกได้ทำการ ศึกษาค้นคว้าข้อมูลทางการบัญชีเพื่อให้ทราบว่ามีการทำงานอย่างไร มีรูปแบบการคำนวนงบอย่างไร จากนั้น ได้ทดลองใช้ภาษาต่างๆในการพัฒนาเว็บเซอร์วิส ทำให้ได้พบว่าในปัจจุบัน เทคโนโลยีต่างๆ สามารถเขียนเว็บเซอร์วิส ได้เกือบทุกภาษา แต่พบว่าการเขียนบนภาษา C# ในสิ่งแวดล้อมของ Microsoft Visual Studio นั้นสามารถพัฒนาได้รวดเร็ว และมีประสิทธิภาพการใช้งานสูง แต่ยังมีข้อจำกัดอยู่ที่ เว็บเซอร์เวอร์ จำเป็นต้องใช้ IIS เป็นเว็บเซอร์เวอร์เพียงอย่างเดียว

การทำงานคือจะเขียนโมดุลการคำนวนทางบัญชี เอาไว้ที่เว็บเซอร์วิส โดยจะนำมาซึ่งส่วนที่เป็นมาตรฐาน และส่วนที่ใช้ในการคำนวนเท่านั้น เพื่อให้เกิดการเรียกใช้ได้ ทุกๆ โปรแกรมโดยไม่มีข้อจำกัดในเรื่องของฐานข้อมูล โดยมีรูปแบบการทำงานโดยรวมดังภาพที่ 2



รูปที่ 2 ภาพรวมการทำงานของระบบ

จากการวิจัยจะมีการออกแบบโปรแกรมการทำงานออกเป็น 2 ส่วนคือ

ส่วนที่ 1 ส่วนที่เป็นส่วนที่ให้บริการเว็บเซอร์วิส เขียนโดยภาษา C#

ส่วนที่ 2 ส่วนที่เป็นโปรแกรมทดสอบการเรียกใช้งาน ซึ่งสามารถเขียนโดยภาษาอะไรก็ได้ แต่ในงานวิจัยเขียนโดยใช้ C# เช่นกัน

โดยโปรแกรมที่เขียนขึ้นมานั้น โปรแกรมเมอร์จะเขียนเฉพาะในส่วนที่เป็นส่วนกรอกข้อมูลนั่นคือ สมุดรายวัน (General Journal) จากนั้นจะทำการเรียกเอาบริการต่างๆ จากเว็บเซอร์วิส มาใช้ในการคำนวน โดยบริการนั้นจะขึ้นอยู่กับว่าจังหวะนั้น ต้องการจะทำอะไร ในงบการเงิน โดยจะสื่อสารกันระหว่าง โปรแกรมทดลอง และเว็บเซอร์วิส ด้วยภาษา XML

ผลการวิจัย

การเรียกใช้บริการทางการบัญชีผ่านทางเว็บเซอร์วิส เมื่อสามารถเรียกใช้ได้ จะพบว่าปริมาณการเขียนโปรแกรมในส่วนของโปรแกรมทดลองนั้นลดลงไปพอสมควร เนื่องจากโปรแกรมส่วนที่สำคัญสำหรับงานทางบัญชีนั้น ได้ไปอยู่ในรูปแบบของเว็บเซอร์วิสแล้ว และจะมีผลทำให้เวลาที่เขียนโปรแกรมทางการบัญชีขึ้นมาด้วยเทคโนโลยีอื่น ก็สามารถเรียกใช้เว็บเซอร์วิสทางการบัญชีได้ทันที โดยไม่ต้องเขียนใหม่ และยังช่วยให้การออกแบบระบบมีความยืดหยุ่นมากขึ้น ยกตัวอย่างเช่น ถ้าองค์กรหนึ่งมีการออกแบบส่วนงานในโมดุลต่างๆในองค์กร เป็นคนละแบบกัน เช่น บางส่วนเป็น Windows Application และบางส่วนเป็น Web Application ก็ยังสามารถทำงานเชื่อมต่อกันได้ โดยการใช้เทคโนโลยีเว็บเซอร์วิส หรืออย่างกรณีที่เป็นองค์กรใหญ่ มีองค์กรย่อยมากมาย แต่ละองค์กรก็มีระบบเป็นของตนเอง ก็สามารถประยุกต์งานวิจัย นำไปใช้กับองค์กร ทั้งใหญ่และเล็ก เพื่อให้องค์กรใหญ่สามารถเรียกดูความเคลื่อนไหวขององค์กรเล็กได้ ตลอดเวลา หรือองค์กรย่อยก็สามารถเรียกใช้บริการขององค์กรใหญ่ได้อีกด้วย

บทสรุปและข้อเสนอแนะ

สามารถสรุปในประเด็นสำคัญได้ดังนี้

– การนำเอาเทคโนโลยีเว็บเซอร์วิสมาใช้กับงานในชีวิตประจำวันนั้น สามารถนำเอามาใช้ได้ โดยในอดีตอาจมองว่าเป็นเรื่องที่ไม่จำเป็น แต่ถ้าสละเวลาศึกษา และประยุกต์ใช้กับตัวงาน โดยเฉพาะงานที่มีโครงสร้างพื้นฐานเหมือนกัน เช่น งานบัญชี จะพบว่าสามารถเขียนโปรแกรมขึ้นมารองรับได้หลายๆแพลตฟอร์ม (Platform) โดยไม่จำเป็นต้องเขียนซ้ำ ทำให้ประหยัดเวลาการทำงานลงไปได้มาก

– การนำไปพัฒนาต่อ ให้ไปอยู่ในรูปแบบของ SOA เป็นสิ่งที่โลกในปัจจุบันให้ความสนใจ อีกทั้งผู้เชี่ยวชาญทาง SOA หรือ เว็บเซอร์วิสในปัจจุบันยังคงมีไม่มาก ด้วยศักยภาพของเทคโนโลยี ทำให้มีคุณค่าแก่การเรียนรู้ เพื่อใช้ในการทำงานในอนาคต

บรรณานุกรม

Youcef Baghdadi. (2005). A Web Services-Based Business Interactions Manager to Support Electronic Commerce Applications. Xi'an, China : ICEC'05, August 15-17, 2005.

Eyhab Al-Masri and Qusay H. Mahmoud. (2008) Investigating Web Services on the World Wide Web. Beijing, China : The International World Wide Web Conference Committee (IW3C2) WWW 2008, April 21-25, 2008.

Keith Ballinger. (2003). .Net Web Services Architecture and Implementation. United Stage of America : Addison Wesley.

Thomas Erl. (2006). Service-Oriented Architecture Concepts, Technology and Design. United Stage of America : PRENTICE.

Judith Hurwitz, Robin Bloor, Carol Baroudi and Macia Kaufman. (2007). Service Oriented Architecture for Dummies. Canada : Wiley Publishing, Inc.

Timothy C. Lethbridge and Robert Laganiere. (2005). Object-Oriented Software Engineering Practical Software Development using UML and JAVA. Singapore : Mc Graw-Hill.

เชาวลีย์ พงศ์ผาติโรจน์ และ ดร. วรศักดิ์ ทุมมานนท์. (2004). **Accounting Principle 1.** โรงพิมพ์ จุฬาลงกรณ์มหาวิทยาลัย : สำนักพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย.

ประวัติผู้เขียน

นายวศิน ตรีสินธุรส เกิดเมื่อวันที่ 2 ธันวาคม พ.ศ. 2523 ที่อำเภอเมือง จังหวัดนครราชสีมา เริ่มเข้าศึกษาระดับชั้นอนุบาล 1-ชั้นประถมศึกษาปีที่ 6 ที่โรงเรียนอนุบาลนครราชสีมา อำเภอเมือง จังหวัดนครราชสีมา ระดับมัธยมศึกษาตอนต้น-ตอนปลาย ที่โรงเรียนราชสีมาวิทยาลัย อำเภอเมือง จังหวัดนครราชสีมา ในปีการศึกษา 2546 ได้เข้าศึกษาต่อในระดับปริญญาตรี ที่มหาวิทยาลัยราชภัฏนครราชสีมา สาขาวิทยาการคอมพิวเตอร์ และสำเร็จการศึกษาเมื่อปี พ.ศ. 2548 ภายหลังสำเร็จการศึกษาในระดับปริญญาตรี ได้เข้าศึกษาต่อในระดับปริญญาโท สาขาวิชาวิศวกรรมคอมพิวเตอร์ สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี ในปีการศึกษา 2549

ในระหว่างการศึกษาได้รับความอนุเคราะห์อย่างดียิ่งจากคณาจารย์ในสาขาวิชา และได้รับความไว้วางใจให้เป็นผู้ช่วยสอนปฏิบัติการรายวิชา Computer Programming