การศึกษาสถาปัตยกรรมเชิงบริการ และการนำไปประยุกต์ใช้งานด้วยสถาปัตยกรรมซอฟต์แวร์แบบ REST

STUDY OF SERVICE-ORIENTED ARCHITECTURE AND ITS IMPLEMENTATION BY USING REST ARCHITECTURE

พชรวร บุญชู

PACHARAWON BOONCHOO

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมสารสนเทศ

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2553

KMITL-2010-EN-M-230-155

STUDY OF SERVICE-ORIENTED ARCHITECTURE AND ITS IMPLEMENTATION BY USING REST ARCHITECTURE

PACHARAWON BOONCHOO

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

MASTER OF ENGINEERING IN INFORMATION ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2010

KMITL-2010-EN-M-230-155

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองวิทยานิพนธ์

หัวข้อวิทยานิพนธ์ การศึกษาสถาปัตยกรรมเชิงบริการ และการนำไปประยุกต์ใช้งานด้วยสถาปัตยกรรมซอฟต์แวร์แบบ REST

Thesis Title Study of Service – Oriented Architecture and its Implementation by Using REST Architecture

นักศึกษา นายพชรวร บุญชู

รหัสประจำตัว 48061054

ปริญญา วิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชา วิศวกรรมสารสนเทศ

อาจารย์ที่ปรึกษาวิทยานิพนธ์ ผศ.ดร.พิทักษ์ ธรรมวาริน

หมายเลขวิทยานิพนธ์ KMITL-2010-EN-M-230-155

คณะกรรมการสอบวิทยานิพนธ์		ลายมือชื่อ
รศ.อรลาภ	แสงอรุณ	
รศ.ดร.ชวลิต	เบญจางคประเสริฐ	
รศ.ดร.อรรถสิทธิ์	หล่าสกุล	
ดร.สัญญา	คูณขาว	
ผศ.ดร.พิทักษ์	ธรรมวาริน	

วัน / เดือน/ ปี ที่สอบ วันพุธที่ 20 ตุลาคม พ.ศ. 2553 เวลา 13.00 – 15.00 น.

สถานที่สอบ ณ อาคาร A ชั้น 3 ห้องประชุม 4

คณะวิศวกรรมศาสตร์ รับรองแล้ว

(รองศาสตราจารย์ ดร.สุชัชวีร์ สุวรรณสวัสดิ์)

คณบดี คณะวิศวกรรมศาสตร์

วันที่ 20 ตุลาคม พ.ศ. 2553

กิตติกรรมประกาศ

วิทยานิพนธ์เล่มนี้สำเร็จลุล่วงได้ด้วยความกรุณาจาก ผศ. มยุรี เลิศเวชกุล และ ผศ.ดร. พิทักษ์ ธรรมวาริน อาจารย์ที่ปรึกษาวิทยานิพนธ์ทั้งสองท่าน ผู้ซึ่งให้คำแนะนำ และคำปรึกษาในการทำวิทยานิพนธ์เล่มนี้มาโดยตลอด อีกทั้งยังช่วยตรวจทานแก้ไข รวมถึงให้การช่วยเหลือสนับสนุนในทุกๆด้าน เพื่อให้วิทยานิพนธ์เล่มนี้เสร็จสมบูรณ์ ข้าพเจ้ามีความซาบซึ้งในความอนุเคราะห์ของอาจารย์ทั้งสองท่าน และขอกราบขอบพระคุณเป็นอย่างสูงมา ณ โอกาสนี้ด้วย

ขอขอบพระคุณคณาจารย์ สาขาวิชาวิศวกรรมสารสนเทศ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทุกท่านที่ได้ประสิทธิประสาทวิชาความรู้ และให้คำแนะนำอันเป็นประโยชน์ต่อการทำวิทยานิพนธ์

ขอขอบคุณพี่ๆ เพื่อนๆ และน้องๆ ทั้งที่ได้ร่วมเรียนและร่วมงานกัน ที่คอยให้ความช่วยเหลือ และเป็นกำลังใจเสมอมา

ขอขอบคุณบัณฑิตวิทยาลัย และบัณฑิตศึกษา คณะวิศวกรรมศาสตร์ สำหรับคำแนะนำและความช่วยเหลือในทุกๆ เรื่อง

สุดท้ายนี้ข้าพเจ้าขอกราบขอบพระคุณ บิดา มารดา ผู้ให้กำเนิด ให้โอกาสทางการศึกษา เป็นกำลังใจ และคอยช่วยเหลือสนับสนุนมาโดยตลอด สำหรับคุณงามความดีทั้งหลายอันใดที่เกิดจากวิทยานิพนธ์เล่มนี้ ข้าพเจ้าขอมอบให้กับทุกๆ ท่านที่ได้กล่าวมาแล้วข้างต้น

พชรวร บุญชู

หัวข้อวิทยานิพนธ์	การศึกษาสถาปัตยกรรมเชิงบริการ และการนำไปประยุกต์ใช้งานด้วยสถาปัตยกรรมซอฟต์แวร์แบบ REST
นักศึกษา	นาย พชรวร บุญชู
รหัสนักศึกษา	48061054
ปริญญา	วิศวกรรมศาสตร์มหาบัณฑิต
สาขา	วิศวกรรมสารสนเทศ
พ.ศ.	2553
อาจารย์ที่ปรึกษาวิทยานิพนธ์	ผศ.ดร.พิทักษ์ ธรรมวาริน

บทคัดย่อ

โครงสร้างของระบบภายในองค์กรที่มีการให้บริการที่หลากหลายอย่างผู้ให้บริการโทรศัพท์เคลื่อนที่มักจะประกอบไปด้วยระบบที่มีความแตกต่างทั้งในด้านของระบบปฏิบัติการและซอฟต์แวร์ประยุกต์ของระบบที่ออกแบบมาเพื่อใช้งานเฉพาะเท่านั้น ดังนั้นเมื่อมีความจำเป็นต้องเปลี่ยนแปลงกระบวนการทางธุรกิจก็มักจะเกิดความล่าช้าและความยุ่งยากในการเปลี่ยนแปลงและพัฒนาโดยใช้โครงสร้างระบบเดิม การนำสถาปัตยกรรมเชิงบริการมาใช้ปรับปรุงและพัฒนาระบบนั้นเป็นวิธีการหนึ่งสำหรับการแก้ไขปัญหาดังกล่าวได้ โดยที่แนวความคิดของสถาปัตยกรรมเชิงบริการทำให้การพัฒนาเซอร์วิสแอพพลิเคชันนั้นง่ายขึ้น และทำให้แอพพลิเคชันดังกล่าวสามารถนำมาใช้งานใหม่ได้ด้วยการจัดระบบที่มีอยู่ออกเป็นลำดับชั้นต่างๆ โดยมีชั้นของเซอร์วิสเป็นส่วนสำคัญ เทคโนโลยีที่ใช้ในการพัฒนาเว็บเซอร์วิสก็เป็นอีกปัจจัยที่ต้องคำนึงถึงโดยเฉพาะอย่างยิ่งเมื่อมีการใช้งานเว็บเซอร์วิสผ่านอินเตอร์เน็ตที่มีความเร็วต่ำและมีค่าใช้จ่ายสูงอย่างการเชื่อมต่ออินเตอร์เน็ตผ่านโทรศัพท์เคลื่อนที่ด้วยเทคโนโลยีของ EDGE หรือ GPRS การพัฒนาเว็บเซอร์วิสด้วยสถาปัตยกรรมแบบ REST นั้นสามารถช่วยลดขนาดของข้อมูลในการร้องขอและตอบกลับของบริการในแต่ละทรานแซคชัน อีกทั้งยังช่วยลดระยะเวลาในการตอบสนองของการเรียกใช้บริการเมื่อเปรียบเทียบกับเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP งานวิจัยนี้จึงได้นำเสนอแนวทางการออกแบบระบบให้บริการข้อมูลบนเครือข่ายโทรศัพท์เคลื่อนที่ที่ใช้แนวคิดของสถาปัตยกรรมเชิงบริการภายใต้การให้บริการเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST และนำเสนอกลไกในการรองรับการทำงานร่วมกับแอพพลิเคชันเดิมที่ไม่สามารถพัฒนาให้เป็นเว็บเซอร์วิสได้ด้วยการใช้เซอร์วิสบัสเป็นตัวกลางในการเชื่อมต่อระบบให้บริการต่างๆ เข้าด้วยกัน

Thesis Title	Study of Service-Oriented Architecture and its Implementation by using REST Architecture
Student	Mr. Pacharawon Boonchoo
Student ID.	48061054
Degree	Master of Engineering
Program	Information Engineering
Year	2010
Thesis Advisor	Asst.Prof.Dr.Pitak Thumwarin

ABSTRACT

Normally, a system of mobile network operator, that provides various services, is mostly composed of heterogeneous systems those are different in both of operating system platforms and its proprietary applications. Thus, changing of the business processes of the existing system could be very slow and difficult. Service-Oriented Architecture (SOA) could be a solution for the mentioned problem in which SOA design concept is to simplify the service application development and hence makes the system components to be reusable by classifying the system components into layers. Moreover, service application technologies must be concerned especially for provision of mobile web services which are to be accessed through low speed, high cost GPRS / EDGE internet connection. Developing web services under REST architecture can reduce the transaction message length and improve the latency for service response, comparing to SOAP architecture. The research proposes a proper way to design a mobile information service system by using SOA concept under REST web service architecture. And Service Bus concept is introduced to solve the interoperation problem with proprietary service applications those are not able to change to be web services.

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญตาราง (ต่อ)

ตารางที่ หน้า

สารบัญรูป

รูปที่ หน้า

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

รูปที่ หน้า

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

สำหรับการพัฒนาระบบเพื่อตอบสนองความต้องการใช้งานภายในองค์กร หรือการพัฒนาระบบเพื่อให้บริการแก่ลูกค้าองค์กรก็ดี มีแนวโน้มไปในทิศทางการขยายตัวอย่างต่อเนื่อง ทั้งนี้เมื่อทำการศึกษาและวิเคราะห์ถึงปัญหาในเรื่องของการพัฒนาระบบโดยอ้างอิงถึงระบบในธุรกิจโทรคมนาคมสำหรับผู้ให้บริการโทรศัพท์เคลื่อนที่ (Mobile Network Operator, MNO) [1] จะเห็นได้ว่าในปัจจุบันการให้บริการเสริม (Value Added Service , VAS) [2] ได้เข้ามามีบทบาทสำคัญสำหรับผู้ให้บริการเครือข่ายโทรศัพท์เคลื่อนที่ ไม่ว่าจะเป็นบริการข้อมูลข่าวสาร บริการบันเทิง หรือบริการด้านสิทธิประโยชน์ และรายการส่งเสริมการขายต่างๆ สำหรับลูกค้า เป็นต้น โดยการที่จะจัดเตรียมเพื่อให้บริการทั้งหลายเหล่านี้จำเป็นต้องมีการพัฒนาระบบเพิ่มเติมเพื่อรองรับการให้บริการแบบเฉพาะเจาะจงตามความต้องการนั้นๆ ในขณะที่เทคโนโลยีและแพลทฟอร์มของระบบคอมพิวเตอร์มีการเปลี่ยนแปลงไปอย่างรวดเร็ว จึงมักจะมีการพัฒนาแอพพลิเคชันและระบบให้บริการใหม่ๆ บนแพลทฟอร์มที่เปลี่ยนแปลงไปแต่ยังจำเป็นต้องมีการเชื่อมต่อหรือใช้งานร่วมกับระบบให้บริการเดิมที่มีอยู่ เนื่องจากสาเหตุที่กล่าวมานี้จึงทำให้มักจะเกิดปัญหาเกี่ยวกับความล่าช้าในการพัฒนาระบบให้บริการใหม่ๆ และการเปิดให้บริการแก่ลูกค้า อีกทั้งยังอาจจะเกิดปัญหาความไม่เข้ากันของระบบเดิมที่ใช้อยู่กับระบบใหม่เมื่อมีความจำเป็นต้องนำมาใช้งานร่วมกัน ทั้งนี้นอกจากสาเหตุข้างต้นแล้วยังอาจจะเป็นผลสืบเนื่องมาจากการที่มีผู้เกี่ยวข้องจากหลายหน่วยงาน การเปลี่ยนผู้ออกแบบ และหรือผู้ดูแลระบบ รวมถึงความไม่มีมาตรฐานของการออกแบบระบบอย่างเหมาะสม

ปัญหาดังกล่าวส่งผลให้ผู้ประกอบการถูกลดโอกาสในการแข่งขัน ลูกค้าที่ขอใช้บริการเกิดความไม่พอใจในความล่าช้าของการให้บริการ มีค่าใช้จ่ายในการลงทุนสูงสำหรับการพัฒนาระบบใหม่

หรือค่าใช้จ่ายในการแก้ไขระบบเดิมเพิ่มมากขึ้น และบางครั้งอาจทำให้การใช้ทรัพยากรของระบบที่มีอยู่ไม่คุ้มค่า และไม่เกิดประโยชน์เท่าที่ควรจะเป็น

ดังนั้นหากเรามีการกำหนดมาตรฐานของการออกแบบโดยมุ่งเน้นให้เกิดประสิทธิภาพสูงสุด และเป็นไปในทิศทางเดียวกันทุกระบบ ด้วยการคำนึงถึงข้อจำกัดต่างๆ ของการให้บริการ และความซับซ้อนของการให้บริการแต่ละประเภท รวมถึงเทคโนโลยีที่เหมาะสมและแก้ปัญหาได้ในระยะยาว ก็จะเป็นการช่วยให้ปัญหาที่กล่าวมานั้นลดลงได้

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

ในการพัฒนาระบบทั่วไปนั้นจะต้องคำนึงถึงปัจจัยที่สำคัญสองประการคือ ค่าใช้จ่ายในการลงทุน และเวลาที่ใช้ในการพัฒนาระบบ ซึ่งปัจจัยทั้งสองอาจจะเป็นข้อจำกัดที่ทำให้การตัดสินใจในการออกแบบและการพัฒนาระบบเป็นไปอย่างยากลำบาก และมักจะเป็นผลให้ระบบที่พัฒนาขึ้นไม่มีประสิทธิภาพเท่าที่ควร โดยอาจจะมีแนวทางในการพัฒนาที่แตกต่างกันไป ซึ่งอาจจะทำให้เกิดผลกระทบต่างๆ ตามมาอีกมากมาย ทั้งในเรื่องของความล่าช้าในการให้บริการ การใช้ทรัพยากรที่มีอย่างไม่ถูกต้องเหมาะสม และการขยายขนาดของระบบในอนาคตเป็นไปได้ยาก เป็นต้น

งานวิจัยนี้ต้องการศึกษา และนำทฤษฎีที่เหมาะสมมาประยุกต์ใช้เพื่อกำหนดมาตรฐานของการออกแบบและพัฒนาระบบ โดยมุ่งเน้นไปที่การพัฒนาระบบให้บริการข้อมูลที่เหมาะสมสำหรับผู้ให้บริการโทรศัพท์เคลื่อนที่ ด้วยการศึกษาถึงทฤษฎี และสถาปัตยกรรมของการพัฒนาซอฟต์แวร์ที่มีประสิทธิภาพ และมีความเหมาะสมที่จะนำมาใช้ในการให้บริการ และนำแนวทางดังกล่าวไปพัฒนาเป็นระบบตัวอย่างเพื่อทดสอบประสิทธิภาพเปรียบเทียบกับสถาปัตยกรรมที่มีอยู่เดิม เพื่อแสดงให้เห็นว่าการพัฒนาระบบตามแนวทางใหม่นั้นสามารถทำงานหรือให้บริการทดแทนระบบเดิมที่มีอยู่ได้ ทั้งนี้เพื่อให้การพัฒนาระบบให้บริการของที่จำเป็นจะต้องมีการเพิ่มเติมบริการใหม่ๆ อยู่เสมอในขณะที่ยังคงต้องรักษาบริการเดิมที่มีอยู่ ให้สามารถรองรับความต้องการของลูกค้าได้ในเวลาที่รวดเร็วขึ้น และเป็นการทำให้ระบบถูกใช้งานอย่างมีประสิทธิภาพมากขึ้น โดยมีการใช้ทรัพยากรที่เหมาะสมอย่างคุ้มค่าเพื่อให้เกิดประโยชน์สูงสุด

1.3 สมมุติฐานของการศึกษา

การเปลี่ยนแปลงสถาปัตยกรรมของระบบเดิมที่มีอยู่ให้กลายเป็นระบบตามแนวคิดของสถาปัตยกรรมเชิงบริการภายใต้การให้บริการเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST ที่กำหนดขึ้นนั้นจะช่วยทำให้มีรูปแบบในการปรับปรุงแก้ไขระบบเดิม หรือพัฒนาระบบใหม่ๆ ที่ชัดเจน สามารถตอบสนองความต้องของการกระบวนการทางธุรกิจ โดยมีต้นทุนการพัฒนาระบบต่ำลง และใช้เวลาในการพัฒนาที่รวดเร็วขึ้น โดยเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST ที่พัฒนาขึ้นมานั้นจะช่วยลดขนาดของข้อมูลในการร้องขอบริการ และขนาดของข้อมูลในการตอบกลับบริการ ทำให้การใช้แบนด์-วิดท์ (Bandwidth) ในเน็ตเวิร์คลดน้อยลง อีกทั้งยังเป็นการเพิ่มประสิทธิภาพของการให้บริการด้วยเวลาในการตอบสนองของบริการที่เร็วขึ้น เมื่อเทียบประสิทธิภาพกับเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP และ การนำเซอร์วิสบัสมาใช้เป็นตัวกลางจะช่วยทำให้แอพพลิเคชันเดิมที่ไม่สามารถพัฒนาให้เป็นเว็บเซอร์วิสได้ อย่างเช่น แม่ข่าย LDAP นั้นสามารถเชื่อมต่อกับระบบให้บริการอื่นๆ หรือทำให้สามารถเรียกใช้งานด้วยเว็บเซอร์วิสได้

1.4 ทฤษฎีหรือแนวความคิดที่ใช้ในการวิจัย

งานวิจัยนี้ได้แสดงให้เห็นถึงแนวทางของการออกแบบและพัฒนาระบบโดยอ้างอิงระบบให้บริการข้อมูลของผู้ให้บริการโทรศัพท์เคลื่อนที่ เนื่องจากผู้ให้บริการโทรศัพท์เคลื่อนที่มีแนวโน้มสูงในการเปลี่ยนแปลงการพัฒนาระบบ ภายใต้ข้อจำกัดต่างๆ ในด้านการให้บริการแก่ลูกค้าทั้งในเรื่องของราคา และความรวดเร็วในการให้บริการ ซึ่งงานวิจัยฉบับนี้ได้ทำการศึกษา และหยิบยกทฤษฎีสถาปัตยกรรมเชิงบริการ (Service-Oriented Architecture, SOA) [3, 4] มาใช้เป็นแนวทางการออกแบบ และพัฒนาระบบ โดยมุ่งเน้นให้ระบบที่แตกต่างกันสามารถทำงานร่วมกันได้โดยที่ไม่ขึ้นอยู่กับแพลตฟอร์มของระบบ ภาษาคอมพิวเตอร์ และเทคโนโลยีที่นำมาใช้ และประยุกต์ใช้การพัฒนาเว็บเซอร์วิสโดยอาศัยรูปแบบแนวความคิดของสถาปัตยกรรม REST (Representational State Transfer) [5, 6, 7] เป็นแอพพลิเคชันในการเชื่อมต่อแลกเปลี่ยนข้อมูลการให้บริการระหว่างระบบต่างๆ และนำเสนอแนวคิดของช่องทางกลางในการแลกเปลี่ยนข้อมูลบริการ หรือ เซอร์วิสบัส (Service Bus) เพื่อทำให้

ระบบให้บริการใหม่สามารถให้บริการร่วมกับระบบที่มีอยู่เดิมหรือระบบที่ไม่สามารถปรับเปลี่ยนแพลทฟอร์มได้อย่างมีประสิทธิภาพ

1.5 ขอบเขตการวิจัย

ขอบเขตของการวิจัยในวิทยานิพนธ์ฉบับนี้คือทำการศึกษา วิเคราะห์ และนำแนวคิดของสถาปัตยกรรมเชิงบริการภายใต้การให้บริการด้วยเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST โดยการนำแนวคิดมาของสถาปัตยกรรมเชิงบริการมาประยุกต์กับระบบบริการข้อมูลของผู้ให้บริการเครือข่ายโทรศัพท์เคลื่อนที่ เพื่อให้สามารถบริการลูกค้าได้อย่างมีประสิทธิภาพบนข้อจำกัดเกี่ยวในเรื่องของความเร็วในการเชื่อมต่ออินเตอร์เน็ตผ่านโทรศัพท์เคลื่อนที่ โดยมุ่งเน้นไปที่การพัฒนาเว็บเซอร์วิสอันเป็นแนวทางที่สามารถนำไปใช้ในการพัฒนาระบบได้จริง โดยทำการพัฒนาเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST เพื่อจำลองการทดสอบเปรียบเทียบประสิทธิภาพกับเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP พร้อมทั้งนำเสนอกลไกสำหรับการรองรับแอพพลิเคชันเดิมในระบบที่ไม่สามารถพัฒนาปรับปรุงให้เป็นเว็บเซอร์วิส ให้สามารถเชื่อมต่อกับระบบอื่นๆ และเรียกใช้งานแบบเว็บเซอร์วิสได้โดยใช้เซอร์วิสบัสเป็นตัวกลาง

1.6 ขั้นตอนของการวิจัย

1. ศึกษาและค้นคว้าทฤษฎีที่เกี่ยวข้องเพื่อนำมาประยุกต์ใช้ในการกำหนดมาตรฐานของการออกแบบเพื่อพัฒนาระบบอย่างเหมาะสมและมีประสิทธิภาพ
2. ทำการกำหนดมาตรฐานของการออกแบบเพื่อพัฒนาระบบ
3. พัฒนาระบบในบางส่วนเพื่อแสดงให้เห็นถึงความสามารถในการนำไปใช้งานจริง
4. เก็บรวบรวมข้อมูลผลการทดสอบระบบ
5. วิเคราะห์และสรุปผลการทดสอบ

บทที่ 2

ทฤษฎีที่เกี่ยวข้อง

2.1 ผู้ให้บริการเครือข่ายโทรศัพท์เคลื่อนที่

ผู้ให้บริการเครือข่ายโทรศัพท์เคลื่อนที่ (Mobile Network Operator, MNO) [1] หรือ ผู้ให้บริการโทรศัพท์เคลื่อนที่ (Mobile Operator) คือ บริษัทที่ทำธุรกิจเกี่ยวกับด้านการสื่อสารโทรคมนาคม โดยทำหน้าที่เป็นผู้ให้บริการการใช้งานโทรศัพท์แบบเคลื่อนที่แก่สมาชิก หรือลูกค้าของผู้ให้บริการรายนั้นๆ สำหรับผู้ให้บริการโทรศัพท์เคลื่อนที่ในประเทศไทย จำเป็นมีต้องได้รับใบอนุญาต หรือ สัญญาร่วมการงานให้ใช้คลื่นความถี่ในช่วงสเปกตรัม (Spectrum) โดยมีหน่วยงานรัฐบาลที่เกี่ยวข้อง หรือเป็นเจ้าของคลื่นความถี่ในช่วงสเปกตรัมดังกล่าวเป็นผู้ออกให้ หรืออนุญาตให้ใช้ ด้วยเงื่อนไขหรือวิธีการที่ต่างกันไปตามความเหมาะสม ซึ่งผู้ให้บริการเครือข่ายโทรศัพท์เคลื่อนที่ต้องวางโครงสร้างพื้นฐานที่จำเป็นต่อการให้บริการโทรศัพท์เคลื่อนที่ทั้งหมด จึงสามารถให้บริการโทรศัพท์เคลื่อนที่ภายในประเทศได้ ซึ่งคลื่นความถี่ในช่วงสเปกตรัมที่ได้รับการจัดสรรมานั้นจะขึ้นอยู่กับประเภทของเทคโนโลยีที่ผู้ให้บริการเลือกใช้ให้บริการแก่สมาชิก หรือลูกค้าของผู้ให้บริการรายนั้นๆ ตัวอย่างเช่น ผู้ให้บริการโทรศัพท์เคลื่อนที่ระบบจีเอสเอ็ม (Global System for Mobile Communications, GSM) [9] ก็จะต้องขอการจัดสรรคลื่นความถี่สเปกตรัมในช่วงที่เป็นคลื่นความถี่ของเทคโนโลยีจีเอสเอ็ม (GSM frequency ranges) [10] เป็นต้น

2.2 ผู้ให้บริการเครือข่ายโทรศัพท์เคลื่อนที่เสมือน

ผู้ให้บริการเครือข่ายโทรศัพท์เคลื่อนที่เสมือน (Mobile Virtual Network Operator, MVNO) [8] คือบริษัทที่ให้บริการโทรศัพท์เคลื่อนที่โดยไม่จำเป็นต้องมีใบอนุญาตประกอบกิจการสื่อสารในคลื่นความถี่สเปกตรัมนั้นๆ และไม่มีโครงสร้างพื้นฐานที่จำเป็นต่อการให้บริการโทรศัพท์เคลื่อนที่ด้วย

เช่นกัน แต่สามารถให้บริการได้โดยใช้วิธีการแบ่งเช่าโครงข่ายและความถี่จากผู้ให้บริการโทรศัพท์เคลื่อนที่ที่มีใบอนุญาตประกอบกิจการสื่อสารพร้อมโครงสร้างพื้นฐานนั่นเอง

2.3 บริการเสริม

บริการเสริม (Value Added Service, VAS) [2] คือ ส่วนหนึ่งของการให้บริการแก่สมาชิกสำหรับผู้ให้บริการโทรศัพท์เคลื่อนที่ โดยบริการเสริมจะครอบคลุมบริการทั้งหลายที่ไม่ใช่การติดต่อสื่อสารด้วยเสียงที่เป็นบริการหลักในการให้บริการของผู้ให้บริการโทรศัพท์เคลื่อนที่นั่นเอง แนวความคิดหลักของบริการเสริมก็คือการนำเอาการบริการตามมาตรฐานมาพัฒนาให้เกิดมูลค่าเพิ่มมากขึ้น เช่น การนำเสนอข้อมูลข่าวสาร,ความบันเทิงต่างๆตามความต้องการของลูกค้าลูกค้าและเก็บค่าบริการตามช่วงเวลา หรือจำนวนครั้งที่ใช้บริการอย่างเช่น บริการข้อความสั้น(Short Message Service, SMS) [11] บริการข้อความมัลติมีเดีย (Multimedia Messaging Service, MMS) [12] หรือบริการเชื่อมต่ออินเตอร์เน็ตผ่านโทรศัพท์เคลื่อนที่ (General Packet Radio Service, GPRS or Enhanced Data rates for GSM Evolution, EDGE) [13, 14, 15] รวมไปถึงบริการต่างๆเพื่อให้สิทธิประโยชน์แก่ลูกค้าเป็นพิเศษ เป็นต้น

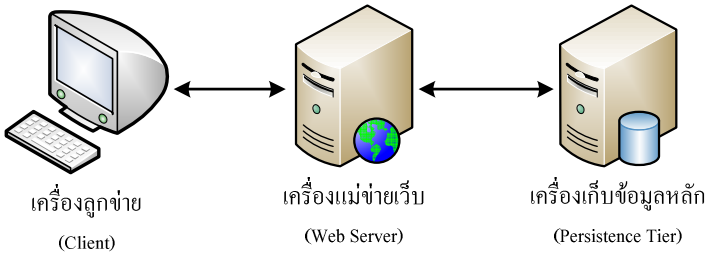
บริการเสริมสามารถพัฒนาบนระบบการติดต่อสื่อสารของผู้ให้บริการเครือข่ายโทรศัพท์เคลื่อนที่เอง หรือ พัฒนาบนระบบของผู้ให้บริการอื่น (Third-Party Value Added Services Provider, VASP) ซึ่งเป็นที่รู้จักกันดีในชื่อผู้ให้บริการข้อมูล (Content Provider , CP) โดยให้บริการผ่านทางเกตเวย์ (Gateway) ต่างๆที่ผู้ให้บริการเครือข่ายโทรศัพท์เคลื่อนที่ได้จัดเตรียมไว้ให้

2.4 สถาปัตยกรรมเชิงบริการ

2.4.1 พัฒนาการของสถาปัตยกรรมเชิงบริการ

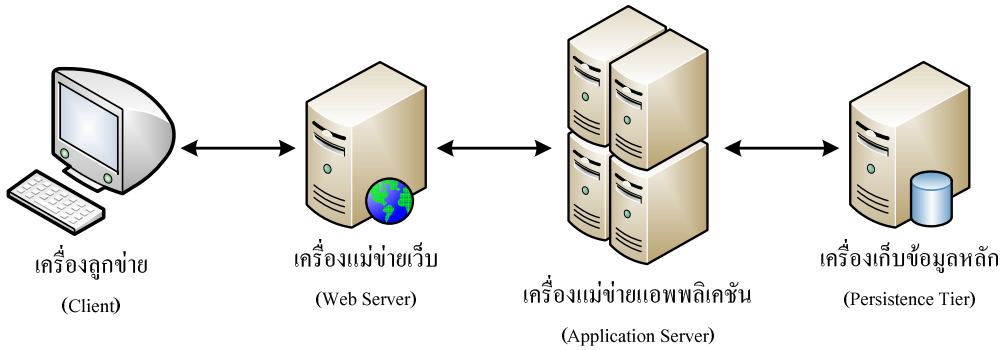
สถาปัตยกรรมเชิงบริการ (Service-Oriented Architecture, SOA) [3, 4, 16] มีพัฒนาการมาจากระบบคอมพิวเตอร์แบบกระจายตัว (Distributed Computing) โดยเริ่มตั้งแต่ที่เป็นแบบชั้นเดียว (Single Tier) ไปสู่แบบลำดับชั้นของเว็บ (Web Tier) ดังแสดงให้เห็นในรูปที่ 2.1 ในปัจจุบันมีการพัฒนาเว็บ

แอพ-พลิเคชัน (Web Application) ขึ้นมาใช้งาน ซึ่งอาจจะพัฒนาโดยใช้ภาษา Java, .NET หรือ PHP ทั้งในส่วนของการแสดงผล (Presentation) และส่วนของการประมวลผลทางธุรกิจ (Business Logic) บนเครื่องแม่ข่ายเว็บ (Web Server) และเชื่อมโยงต่อไปยังชั้นของระบบหลัก (Persistence Tier) อันได้แก่ เครื่องเก็บข้อมูลหลัก เป็นต้น ทั้งนี้ผู้พัฒนายังต้องพัฒนาระบบในส่วนที่เป็นบริการของระบบ (System Service) เช่น การจัดสภาวะโหลดให้สมดุล (Load Balancing) การจัดการทรานแซคชัน (Transaction Management) และการรักษาความปลอดภัย (Security) ด้วยเหตุนี้เองทำให้บางครั้งการพัฒนาเว็บแอพ-พลิเคชันแบบลำดับชั้นของเว็บสำหรับระบบขนาดใหญ่มักจะทำได้ยาก



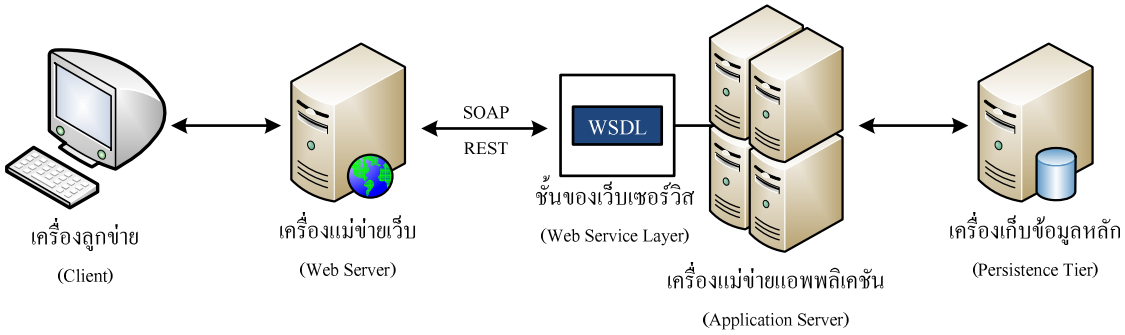
รูปที่ 2.1 สถาปัตยกรรมแบบลำดับชั้นของเว็บ (Web-Tier Architecture)

ต่อมาได้มีแนวทางการพัฒนาระบบแบบหลายชั้น (N-Tier) โดยมีการปรับปรุงโครงสร้างจากระบบแบบลำดับชั้นของเว็บ โดยนำเอาเครื่องแม่ข่ายแอพพลิเคชัน (Application Server) มาเป็นมิดเดิล-แวร์ (Middleware) อยู่ระหว่างเครื่องแม่ข่ายเว็บ กับ เครื่องเก็บข้อมูลหลัก เพื่อมาใช้ในการจัดการส่วนที่เป็นบริการของระบบ และจัดการเรื่องที่เกี่ยวข้องกับทรัพยากรต่างๆ ของระบบ ทำให้สามารถที่จะเน้นการพัฒนาไปที่ส่วนการแสดงผล และประมวลผลเพื่อให้บริการได้ตามที่ต้องการ ดังแสดงให้เห็นในรูปที่ 2.2



รูปที่ 2.2 สถาปัตยกรรมแบบหลายชั้น (N-Tier Architecture)

หลังจากนั้นมีการนำเอาเทคโนโลยีที่เรียกว่าเว็บเซอร์วิสเข้ามาใช้งาน ซึ่งเป็นการเปลี่ยนโปรโตคอลเฉพาะ (Proprietary Protocol) ที่ใช้งานอยู่สำหรับส่วนติดต่อแลกเปลี่ยนข้อมูลกัน ให้กลายเป็นมาตรฐานตามรูปแบบของเว็บเซอร์วิส เช่น มาตรฐานเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP (Simple Object Access Protocol) [17] หรือเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST (Representational State Transfer) [5, 6] โดยมีการนำเอามาตรฐาน WSDL (Web Services Description Language) [18, 19, 20] มาใช้ในการอธิบายบริการที่มีอยู่ในเครื่องแม่ข่ายแอพพลิเคชัน โดยที่ชั้นของเว็บเซอร์วิส (Web Service Layer) ถูกเพิ่มเข้ามาเป็นส่วนติดต่อและระหว่างเครื่องแม่ข่ายเว็บ กับ เครื่องแม่ข่ายแอพพลิเคชัน ดังตัวอย่างในรูปที่ 2.3 ซึ่งทำให้เกิดการพัฒนาและใช้งานอย่างเป็นรูปแบบมาตรฐานมากขึ้น



รูปที่ 2.3 สถาปัตยกรรมแบบหลายชั้นเมื่อมีการนำชั้นของเว็บเซอร์วิส (Web Service Layer) มาใช้งาน

การนำเว็บเซอร์วิสมาใช้งานนั้น จะทำให้สามารถพัฒนาส่วนติดต่อของส่วนประมวลผลและส่วนการแสดงผลได้อย่างหลากหลายมากขึ้น อันเนื่องมาจากการพัฒนาส่วนการแสดงผลไม่จำเป็นต้องผูกติดอยู่กับเทคโนโลยีของส่วนประมวลผลอีกต่อไป ตัวอย่างเช่น ไม่จำเป็นต้องใช้ JSP/JSF ในการเรียกใช้งานเว็บเซอร์วิสที่พัฒนาโดยใช้ Java แต่สามารถใช้ PHP หรือ .NET หรือภาษาอื่นๆ เรียกใช้งานได้ ดังนั้นในการพัฒนาระบบคอมพิวเตอร์แบบกระจายที่มีชั้นของเว็บเซอร์วิส จึงสามารถเชื่อมโยงระบบที่หลากหลาย (Heterogeneous System) ให้สามารถนำมาใช้งานร่วมกันได้อย่างมีประสิทธิภาพ

2.4.2 ความหมายของสถาปัตยกรรมเชิงบริการ

สถาปัตยกรรมเชิงบริการ (Service-Oriented Architecture, SOA) เป็นแนวคิดในการออกแบบระบบให้อยู่ในรูปของระบบเชิงบริการ (Service Oriented) กล่าวคือ เป็นการจัดแบ่งระบบออกเป็นชั้นๆ โดยสามารถทำให้บริการที่สามารถนำกลับมาใช้ใหม่ได้ ซึ่งทำให้ต่อการเชื่อมต่อระบบที่หลากหลายเข้าด้วยกัน ทั้งนี้ยังช่วยลดค่าใช้จ่ายในการพัฒนา และ ปรับเปลี่ยนระบบ ทำให้การพัฒนาระบบใหม่ๆ เป็นได้อย่างรวดเร็วยิ่งขึ้น

รูปที่ 2.4 การแบ่งชั้นของสถาปัตยกรรมเชิงบริการ (SOA Layers)

แนวคิดของสถาปัตยกรรมเชิงบริการเป็นการจัดแบ่งระบบออกเป็น 4 ลำดับชั้น [16] คือ ชั้นทรัพยากร (Resource Layer) ชั้นเซอร์วิส (Service Layer) ชั้นกระบวนการ (Process Layer) และ ชั้นการเรียกใช้งาน (Access Layer) ดังแสดงความสัมพันธ์ให้เห็นในรูปที่ 2.4 ซึ่งแต่ละลำดับชั้นมีรายละเอียดดังนี้

ชั้นทรัพยากร (Resource Layer) จะเป็นชั้นที่รวมระบบภายในองค์กรที่ทำให้หน้าที่ในการประมวลผล เก็บข้อมูล หรือ ให้บริการ เช่น ระบบฐานข้อมูล ระบบบริการเสริมหลักต่างๆ ระบบเครื่องแม่ข่าย LDAP เป็นต้น

ชั้นเซอร์วิส (Service Layer) เป็นชั้นของเซอร์วิสที่เปิดให้เรียกใช้บริการได้ทั้งหมด โดยที่แต่ละเซอร์วิสนั้นสามารถนำกลับมาเรียกใช้ใหม่ได้ โดยที่เซอร์วิสเหล่านี้พัฒนามาจากฟังก์ชัน (Function) การคำนวณ การประมวลผล หรือ โมดูล (Module) ต่างๆ ที่ทำงานอยู่บนชั้นทรัพยากร (Resource Layer)

ชั้นกระบวนการ (Process Layer) เป็นชั้นของการนำเอาเซอร์วิสมาพัฒนา จัดกระบวนการลำดับขั้นตอน ให้เป็นบริการที่สมบูรณ์ และพร้อมที่จะเรียกใช้งาน ซึ่งเรียกอีกอย่างได้ว่าเป็นกระบวนการทางธุรกิจ (Business Process)

ชั้นการเรียกใช้งาน (Access Layer) เป็นชั้นของการเรียกใช้กระบวนการทางธุรกิจที่พัฒนาขึ้นอยู่ในชั้นกระบวนการ โดยเรียกใช้ง่ายผ่านระบบช่องทางต่างๆ เช่น การเรียกใช้งานผ่านโทรศัพท์เคลื่อนที่ (Mobile Phone) เป็นต้น

2.4.3 ประโยชน์ของการพัฒนาสถาปัตยกรรมเชิงบริการ [16]

การพัฒนาระบบภายในองค์กรด้วยสถาปัตยกรรมเชิงบริการจะเกิดประโยชน์ เช่น สามารถเชื่อมโยงระบบ หรือธุรกิจต่างๆ ได้ง่าย สามารถปรับเปลี่ยนระบบที่ใช้งานอยู่ได้ง่าย ลดค่าใช้จ่ายในการลงทุนและค่าใช้จ่ายในการบำรุงรักษา และทำให้การทำงานของฝ่ายธุรกิจกับฝ่ายพัฒนาระบบสอดคล้องกันมากขึ้น โดยสามารถอธิบายได้ดังนี้

ทำให้ระบบต่างๆ สามารถเชื่อมโยงกันได้ง่าย การพัฒนาระบบแบบสถาปัตยกรรมเชิงบริการทำให้สามารถเชื่อมโยงระบบต่างๆ ภายในองค์กร และภายนอกองค์กร ที่อาจใช้เทคโนโลยีที่แตกต่างกัน จึงทำให้สามารถให้บริการกับลูกค้า คู่ค้า และบุคลากรในองค์กรได้สะดวกมากยิ่งขึ้น

การปรับเปลี่ยนระบบเดิมที่ใช้งานอยู่ หรือการพัฒนาระบบใหม่ๆ สามารถทำได้ง่ายขึ้น การพัฒนาระบบแบบสถาปัตยกรรมเชิงบริการทำให้สามารถเซอร์วิสเดิมที่มีอยู่แล้วกลับมาใช้ใหม่ได้ หรือเมื่อมีการเปลี่ยนแปลงความต้องการบางอย่างก็ไม่จำเป็นต้องมีการพัฒนาใหม่ทั้งหมด ดังนั้นการปรับเปลี่ยนกระบวนการทางธุรกิจจึงเป็นไปได้อย่างรวดเร็ว และทำให้สามารถแข่งขันในตลาดธุรกิจได้

ลดค่าใช้จ่ายในลงทุน การพัฒนาระบบแบบสถาปัตยกรรมเชิงบริการนั้นทำให้องค์กรสามารถที่จะใช้เทคโนโลยีที่หลากหลาย จึงทำให้สามารถเลือกใช้เทคโนโลยีต่างๆ ได้โดยไม่ต้องผูกติดกับเทคโนโลยีใดเทคโนโลยีหนึ่ง ทำให้สามารถควบคุมค่าใช้จ่ายในด้านการลงทุนและพัฒนาระบบลดลง

สามารถพัฒนากระบวนการทางธุรกิจได้ง่าย การพัฒนากระบวนการทางธุรกิจของฝ่ายพัฒนาระบบจะมีขั้นตอนที่ชัดเจนมากขึ้น สามารถแสดงบนแอพพลิเคชันในเชิงกราฟฟิคที่เข้าใจได้ง่ายขึ้น ทำให้หน่วยงานทางธุรกิจที่สามารถที่จะเข้ามาร่วมทำการพัฒนาระบบร่วมกันได้อย่างมีประสิทธิภาพมากขึ้น

2.5 เว็บเซอร์วิส

เว็บเซอร์วิส (Web Service) [4, 21, 22] คือ แอพพลิเคชันที่พัฒนาขึ้นเพื่อใช้ในการแลกเปลี่ยนข้อมูลกัน ระหว่างเครื่องคอมพิวเตอร์ผ่านระบบเครือข่าย เว็บเซอร์วิสจะใช้หลักการสถาปัตยกรรมเชิงบริการโดยจะใช้โปรโตคอลที่มีมาตรฐานกลาง (Standard Protocol) ในการติดต่อสื่อสารกัน และเนื้อหาที่ใช้ส่งและรับส่วนใหญ่จะอยู่ในรูปแบบ XML (eXtensible Markup Language) โดยที่สถาบันวิจัยการ์ทเนอร์ (Gartner Research) ได้ให้คำนิยามของเว็บเซอร์วิสไว้ดังนี้ *"เว็บเซอร์วิสคือ ซอฟต์แวร์คอมโพเนนท์ที่มีการเชื่อมต่อกันแบบไม่เจาะจง (Loosely Coupled) ที่ส่งบริการผ่านเทคโนโลยีอินเทอร์เน็ตที่มีมาตรฐาน"* [23]

ความหมายของเว็บเซอร์วิสในแง่ของการพัฒนาซอฟต์แวร์ มีคุณลักษณะเช่นเดียวกับเซอร์วิสทั่วๆ ไป หรือเรียกได้ว่าเป็นซอฟต์แวร์คอมโพเนนท์ (Software Component) ที่อาจเป็นฟังก์ชันหรือโมดูลที่มีกระบวนการทำงานภายในที่สามารถรับอินพุตเข้ามาเพื่อประมวลผล และส่งผลลัพธ์กลับออกไป เซอร์วิสเหล่านี้อาจจะถูกกำหนดเป็นกระบวนการทางธุรกิจ (Business Process) กล่าวคือจะเป็นฟังก์ชันที่มีหน้าที่เฉพาะการประมวลผลซึ่งจะไม่เกี่ยวข้องกับส่วนแสดงผล (Presentation Logic) นอกจากนี้ด้วยเทคโนโลยีระบบคอมพิวเตอร์แบบกระจาย ทำให้สามารถที่จะพัฒนาซอฟต์แวร์เซอร์วิสเพื่อเรียกใช้จากระยะไกล (Remote) ผ่านระบบเครือข่ายอินเทอร์เน็ต (Internet) ได้

2.5.1 คุณลักษณะพื้นฐานของเว็บเซอร์วิส

ลักษณะพื้นฐานที่สำคัญของเว็บเซอร์วิสมีดังต่อไปนี้

- เว็บเซอร์วิสสามารถเชื่อมโยงโปรแกรมประยุกต์ที่อยู่บนต่างแพลตฟอร์มกันโดยผ่านเครือข่ายอินเทอร์เน็ต
- เว็บเซอร์วิสสามารถใช้โปรแกรมภาษาคอมพิวเตอร์ได้หลากหลายภาษา อาทิเช่น Java, PHP หรือ .NET แล้วทำการแปลงซอฟต์แวร์คอมโพเนนท์ที่มีอยู่ให้กลายเป็นเว็บเซอร์วิส โดยที่ใช้การเชื่อมต่อกันแบบไม่เจาะจง (Loosely Couple) ดังนั้นแต่ละคอมโพเนนท์จะเป็นอิสระต่อกันและมีฟังก์ชันที่สมบูรณ์ในตัว
- เว็บเซอร์วิส ไม่ใช่เว็บเพจซึ่งจะไม่รวมถึงการจัดการส่วนแสดงผลเหมือนการสร้างเว็บทั่วไป

- การเรียกใช้งานเว็บเซอร์วิสทำได้โดยเรียกผ่าน URI (Uniform Resource Identifier) [24] ของเว็บเซอร์วิสที่ต้องการ
- เว็บเซอร์วิสใช้เอกสารในรูปแบบภาษามาตรฐานอย่างเช่น XML ในการส่งข้อมูลระหว่างผู้ให้บริการเซอร์วิสและผู้ใช้บริการเซอร์วิส
- สามารถเรียกใช้งานเว็บเซอร์วิสจากซอฟต์แวร์ประยุกต์อื่นๆ ผ่านโปรโตคอลสื่อสารบนอินเตอร์เน็ต
- สามารถค้นหาเซอร์วิสที่มีให้บริการจากทะเบียนเซอร์วิส (Service Registry) โดยใช้มาตรฐานกลาง เช่น UDDI (Universal Description Discovery and Integration) [25] หรือ WSDL (Web Services Description Language) [18, 19, 20] เป็นต้น
- เว็บเซอร์วิสสามารถที่จะเรียกใช้โดยเครื่องลูกข่ายที่แตกต่างกันได้หลายชนิด เช่น คอมพิวเตอร์ หรือ โทรศัพท์เคลื่อนที่

2.5.2 โมเดลการทำงานของเว็บเซอร์วิส

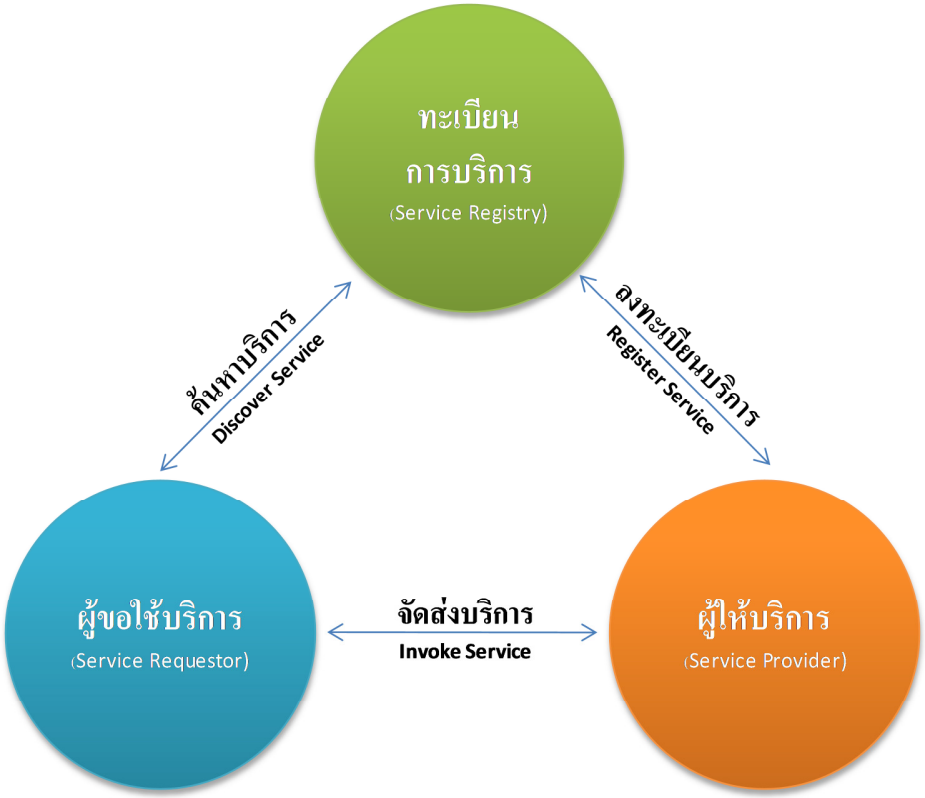
กระบวนการการทำงานของเว็บเซอร์วิสจะมีขั้นตอนการทำงานเช่นเดียวกับซอฟต์แวร์ที่ใช้สถาปัตยกรรมเชิงบริการ ซึ่งเราสามารถที่จะแบ่งบทบาทองค์ประกอบของเว็บเซอร์วิสได้เป็นสามส่วน คือ ผู้ให้เซอร์วิส ผู้ขอใช้เซอร์วิส และ ทะเบียนเซอร์วิส โดยทั้งสามองค์ประกอบมีความสัมพันธ์ดังแสดงในรูปที่ 2.5 และสามารถอธิบายได้ดังนี้

ผู้ให้เซอร์วิส (Service Provider) ผู้ให้เซอร์วิสจะมีหน้าที่ในการพัฒนาและติดตั้งเว็บเซอร์วิส และเป็นผู้ที่นิยามความหมายของเซอร์วิสและทำการลงทะเบียนเซอร์วิสนั้นๆ กับทะเบียนเซอร์วิส (Service Registry)

ผู้ขอใช้เซอร์วิส (Service Requestor) ผู้ขอใช้เซอร์วิสจะเป็นผู้เรียกใช้เว็บเซอร์วิส โดยอาจทำการค้นหาบริการจากทะเบียนเซอร์วิส เพื่อให้ได้เซอร์วิสที่ต้องการ จากนั้นทำการเรียกใช้เซอร์วิสจากผู้ให้เซอร์วิส

ทะเบียนเซอร์วิส (Service Registry) หรืออาจเรียกว่าตัวแทนเซอร์วิส (Service Broker) มีหน้าที่ในการรับลงทะเบียนและช่วยในการค้นหาเซอร์วิส โดยที่ทะเบียนเซอร์วิสจะเก็บรายละเอียดของ

เซอร์วิสต่างๆเช่น นิยามของเซอร์วิส และตำแหน่งของเว็บเซอร์วิส ซึ่งทำหน้าที่คล้ายกับสมุดโทรศัพท์เพื่อช่วยให้ผู้ขอใช้เซอร์วิสสามารถค้นหาเซอร์วิสที่ต้องการได้



รูปที่ 2.5 โมเดลการทำงานของเว็บเซอร์วิส

2.5.3 มาตรฐานที่ใช้ในการพัฒนาเว็บเซอร์วิส

มาตรฐานที่ใช้ในการพัฒนาเว็บเซอร์วิสที่เกี่ยวข้องกับงานวิจัยประกอบไปด้วยมาตรฐาน XML, WSDL, SOAP และ REST ซึ่งรายละเอียดของแต่ละมาตรฐานมีดังนี้

2.5.3.1 XML (eXtensible Markup Language) [26, 27]

XML (eXtensible Markup Language) เป็นมาตรฐานที่ทาง W3C (World Wide Web Consortium) ประกาศให้เป็นมาตรฐานของข้อมูล โดย XML จะอยู่ในรูปของไฟล์ข้อความที่ใช้ยูนิโค้ด

(Unicode) และสามารถที่สร้างรูปแบบในการที่จะแสดงข้อมูลที่ซับซ้อนในรูปแบบของข้อความที่สามารถอ่านได้ง่าย ในปัจจุบัน XML ได้กลายเป็นมาตรฐานสำคัญสำหรับการกำหนดโครงสร้างข้อมูลเนื้อหา และรูปแบบของข้อมูลของเอกสารอิเล็กทรอนิกส์ และยังมีการพัฒนาเพื่อให้สามารถแลกเปลี่ยนข้อมูลระหว่างระบบต่างๆ โปรแกรมประยุกต์ หรืออุปกรณ์ต่างๆ ผ่านทางอินเทอร์เน็ตได้อีกด้วย

XML จัดเป็นภาษารูปแบบหนึ่งที่ใช้เน้นส่วนที่เป็นข้อมูล หรือเรียกว่ามาร์กอัพ (Markup) โดยสามารถกำหนดชื่อส่วนประกอบ (Element) และชื่อแอททริบิวต์ (Attribute) ได้ตามความต้องการ เอกสาร XML จึงเป็นแค่ไฟล์ข้อมูลชนิดหนึ่ง ที่มีส่วนประกอบเปิด และสวนประกอบปิดครอบข้อมูลไว้ตรงกลางเท่านั้น เนื่องจากความง่ายในการสร้างเอกสาร ทำให้ XML ถูกใช้ในการติดต่อกับระบบที่ต่างกันอย่างแพร่หลาย ทั้งนี้รูปแบบของภาษา XML ยังเป็นภาษาพื้นฐานให้กับภาษาอื่นๆ อีกด้วย ยกตัวอย่างเช่น Geography Markup Language (GML), RSS, XHTML, WSDL เป็นต้น

2.5.3.2 WSDL (Web Services Description Language) [8, 19, 20]

WSDL (Web Services Description Language) คือข้อมูลที่อยู่ในรูปแบบของ XML ที่ใช้สำหรับอธิบายเซอร์วิสที่เครื่องแม่ข่ายเว็บเซอร์วิสให้บริการอยู่ โดยที่สามารถใช้ WSDL เพื่อสร้างไฟล์ที่ระบุถึงเซอร์วิสที่แม่ข่ายเว็บเซอร์วิสให้บริการ พร้อมรายละเอียดภายในของเซอร์วิสแต่ละตัวซึ่งเครื่องแม่ข่ายเว็บเซอร์วิสสนับสนุนกระบวนการภายในเซอร์วิสนั้นๆ เอกสาร WSDL ยังทำหน้าที่อธิบายรูปแบบข้อมูลที่เครื่องลูกจ่ายต้องปฏิบัติตามในการร้องขอทำกระบวนการต่างๆ

เนื่องจากเอกสาร WSDL ทำหน้าที่กำหนดความต้องการทั้งเครื่องแม่ข่ายและฝั่งเครื่องลูกข่าย โดยที่แม่ข่ายเว็บเซอร์วิสตกลงที่จะให้เซอร์วิสที่กำหนดไว้เมื่อเครื่องลูกข่ายส่งคำร้องขอในรูปแบบที่ถูกต้องและเหมาะสมเท่านั้น โดยที่เอกสาร WSDL ที่ใช้อธิบายเซอร์วิสของเว็บเซอร์วิสจะประกอบไปด้วยส่วนประกอบ (Element) หลักดังต่อไปนี้

<types>…</types> เป็นการนิยามชนิดข้อมูลที่ใช้ในเว็บเซอร์วิส โดยใช้รูปแบบของภาษา XML เป็นตัวกำหนดชนิดข้อมูล

<message>…</message> เป็นตัวกำหนดข้อความ (Message) ที่ใช้กับเว็บเซอร์วิส เปรียบเสมือนอินพุตที่เว็บเซอร์วิสต้องการ และเอาท์พุตที่จะตอบกลับค่ากลับมา ส่วนที่กำหนดข้อความ

ที่ใช้ในการติดต่อระหว่างลูกข่ายกับแม่ข่ายเว็บเซอร์วิส ในแต่ละข้อความสามารถมี <part> ได้มากกว่าหนึ่ง และให้มองเป็นพารามิเตอร์ของฟังก์ชันที่ได้เรียกไป ดังแสดงตัวอย่างในรูปที่ 2.6 นี้ มี 2 ข้อความ คือ multiplyRequest และ multiplyResponse โดยที่

- multiplyRequest เป็นข้อความที่ประกอบด้วยข้อมูลชนิด float 2 ตัวคือ num1 และ num2
- multiplyResponse เป็นข้อความที่ประกอบด้วยข้อมูลชนิด float 1 ตัวคือ Result

```
<message name=' multiplyRequest'>
        <part name='num1' type='xsd:float'/>
        <part name='num2' type='xsd:float'/>
</message>
<message name='multiplyResponse'>
        <part name='Result' type='xsd:float'/>
</message>
```

รูปที่ 2.6 เอกสาร WSDL ในส่วนประกอบ <message>

<portType>…</portType> เป็นส่วนสำคัญที่สุดสำหรับเอกสาร WSDL ซึ่งประกอบไปด้วยกระบวนการ (Operations) หรือ วิธีการ (Method) หรือฟังก์ชัน (Function) ที่เว็บเซอร์วิสนั้นมีให้เรียกใช้งาน ซึ่งส่วนประกอบนี้จะเป็นตัวบอกว่าเซอร์วิสทั้งหมดของเว็บเซอร์วิสดังกล่าวมีกระบวนการอะไรอยู่บ้างนั่นเอง ในตัวอย่างแสดงให้เห็นในรูปที่ 2.7 คือการประกาศกระบวนการคูณ (Multiply) ตัวเลขสองตัวโดยมี multiplyRequest เป็นอินพุต และ multiplyResponse เป็นเอาต์พุต

```
<portType name='plusPortType'>
        <operation name='multiply'>
                <input message='tns: multiplyRequest'/>
                <output message='tns: multiplyResponse'/>
        </operation>
</portType>
```

รูปที่ 2.7 เอกสาร WSDL ในส่วนประกอบ <portType>

<binding>…</binding> เป็นการกำหนดโปรโตคอลที่ใช้ในการติดต่อกับกับเว็บเซอร์วิส และกำหนดว่าข้อความจะถูกส่งและเข้ารหัสอย่างไร จากตัวอย่างในดังแสดงในรูป 2.8 กำหนดให้ส่งข้อความในรูปแบบของ RPC (Remote Procedure Call) โดยใช้ SOAP บนโปรโตคอล HTTP นอกจากนี้ยังมีการกำหนด namespace และ SOAPAction header สำหรับกระบวนการ multiply() อีกด้วย

```
<binding name='multiplyBinding' type='tns: multiplyPortType'>
        <soap:binding style='rpc' transport='http://schemas.xmlsoap.org/soap/http'/>
        <operation name='multiply'>
                <soap:operation soapAction='urn:xmethods-delayed-quotes#multiply'/>
                <input>
                        <soap:body use='encoded' namespace='urn:xmethods-delayed-
quotes'  encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'/>
                </input>
                <output>
                        <soap:body use='encoded' namespace='urn:xmethods-delayed-
quotes' encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'/>
                </output>
        </operation>
</binding>
```

รูปที่ 2.8 เอกสาร WSDL ในส่วนประกอบ <binding>

<service>…</service> เป็นส่วนที่กำหนด URI ของเว็บเซอร์วิสที่เครื่องลูกข่ายต้องเรียกใช้งานดังแสดงตัวอย่างในรูปที่ 2.9

```
<service name='multiplyService'>
        <port name='multiplyPort' binding='multiplyBinding'>
                <soap:address location='http://wsserver/multiply_server.php'/>
        </port>
</service>
```

รูปที่ 2.9 เอกสาร WSDL ในส่วนประกอบ <service>

2.5.3.2 SOAP (Simple Object Access Protocol) [17, 28]

SOAP (Simple Object Access Protocol) คือโปรโตคอล หรือวิธีการมาตรฐานที่ใช้ในการสื่อสารกันระหว่างเว็บเซอร์วิสโดยอาศัยการส่งข้อมูลผ่านโปรโตคอลสื่อสารอย่างเช่น HTTP สำหรับการร้องขอบริการและตอบกลับผลลัพธ์ของบริการระหว่างเครื่องลูกข่ายเว็บเซอร์วิส กับเครื่องแม่ข่ายเว็บเซอร์วิส

การพัฒนาเว็บเซอร์วิสด้วยสถาปัตยกรรมแบบ SOAP นั้น ส่วนใหญ่จะใช้งานร่วมกับเอกสาร WSDL เพื่อเรียกดูรูปแบบของอินพุต และ เอาท์พุตของเซอร์วิสที่ต้องการเรียกใช้บริการ จากนั้นเครื่องลูกข่ายเว็บเซอร์วิสที่จะร้องขอบริการเครื่องแม่ข่ายเว็บเซอร์วิสแบบ SOAP ก็จะทำการสร้างข้อความ SOAP (SOAP Message) ขึ้นเพื่อติดต่อกับเครื่องแม่ข่ายเว็บเซอร์วิส โดยที่ส่วนของ SOAP Envelope ที่บรรจุข้อมูลไว้ภายในซึ่งแบ่งออกได้เป็น 2 ส่วน ดังตัวอย่างในรูปที่ 2.10 คือ ส่วนหัวของ SOAP (SOAP Header) อันประกอบด้วยคำอธิบายหรือรายละเอียดต่างๆ และส่วนที่สอง คือ ส่วนเนื้อหาของ SOAP (SOAP Body) ซึ่งประกอบด้วยเนื้อหาของการร้องขอหรือการตอบกลับของเว็บเซอร์วิสในรูปแบบของภาษามาตรฐาน XML

รูปที่ 2.10 โครงสร้างของข้อความ SOAP

การส่งข้อความ SOAP มีสองรูปแบบคือ SOAP-RPC และ ข้อความ SOAP (SOAP Message) โดยที่ SOAP-RPC ใช้ในการส่งข้อความเพื่อเรียกใช้เซอร์วิสซึ่งโดยส่วนมากแล้วจะการร้องขอในรูปแบบซิงโครนัส (Synchronous) ส่วนข้อความ SOAP ใช้ในการส่งข่าวสารหรือข้อมูลในรูปแบบ XML ระหว่างผู้ให้บริการ และผู้ขอใช้บริการ โดยสามารถส่งได้ทั้งแบบซิงโครนัส (Synchronous) และอะซิงโครนัส(Asynchronous)

2.5.4 REST (Representational State Transfer) [5, 6]

REST (Representational State Transfer) เป็นรูปแบบทฤษฎีของสถาปัตยกรรมทางซอฟต์แวร์สำหรับการนำเสนอสื่อต่างๆ เช่น เอกสารเอชทีเอ็มแอล (HTML) โดยที่ REST เองนั้นไม่ได้ถูกจำกัดให้ใช้งานกับโปรโตคอลใดเป็นพิเศษ แต่ส่วนใหญ่ก็จะใช้งานร่วมกับ HTTP ที่มีลักษณะการทำงานแบบการร้องขอและตอบรับ (Request-Response) และตัวเนื้อหาของสื่อที่จะส่งไปนั้นสามารถเป็นข้อมูลชนิดใดก็ได้ตามที่ต้องการ เช่น Plain Text, HTML, XML หรือ JSON เป็นต้น

เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST (ในบางครั้งเรียกว่า RESTful) จะเน้นไปที่ผลลัพธ์ของการให้เซอร์วิส โดยจะต้องสามารถกำหนดที่อยู่ของทรัพยากรในระบบให้สามารถเข้าถึงได้ด้วยการกำหนดคีย์หลักที่ไม่ซ้ำกันให้กับทรัพยากรทั้งหมด แล้วระบุที่อยู่ผ่านทาง URI [24] เท่านั้น ดังนั้นทุกครั้งที่ต้องการร้องขอใช้เซอร์วิสหรืออ้างอิงผลลัพธ์ของการให้เซอร์วิส ก็เพียงแค่ระบุ URI ร่วมกับกระบวนการทำงานของ HTTP (HTTP Method) [7] เท่านั้น

พื้นฐานของการออกแบบสถาปัตยกรรม REST นั้นเน้นที่การใช้งานกระบวนการทำงานของ HTTP (POST, GET, PUT และ DELETE) ให้ถูกต้องและเหมาะสม การไม่จัดเก็บสถานะของการทำงาน (Stateless) การใช้รูปแบบของ URL ที่มีลักษณะคล้ายกับโครงสร้างของไดเรคทอรี และการนำสามารถเลือกใช้ข้อมูลมาตรฐานชนิดในก็ได้ในการแสดงเนื้อหาของการร้องขอเซอร์วิสและการตอบกลับ ซึ่งมีรายละเอียดดังต่อไปนี้

การกำหนดรูปแบบของ URL ที่คล้ายกับโครงสร้างไดเรคทอรี การใช้ URL เป็นตัวชี้ไปที่วัตถุ ทรัพยากร หรือ เซอร์วิสต่างๆ โดยสามารถกำหนดกระบวนการของ HTTP ไปพร้อมกับการร้องขอเพื่อเป็นการแสดงว่าต้องการให้แม่ข่ายเว็บเซอร์วิสดำเนินการอย่างไรกับทรัพยากรดังกล่าว

การไม่จัดเก็บสถานะการทำงาน (Stateless) เครื่องแม่ข่ายจะไม่เก็บข้อมูลเซสชันของการใช้เซอร์วิสของผู้ใช้ไว้ โดยเครื่องแม่ข่ายจะจัดการเพียงสถานะของทรัพยากรที่ดูแลอยู่เท่านั้น ซึ่งในกรณีที่แอพพลิเคชันมีความต้องการที่จะเก็บข้อมูลเซสชันของการใช้เซอร์วิสของผู้ใช้นั้น สามารถทำได้ด้วยการเก็บไว้ที่เครื่องลูกข่ายเอง และส่งมาที่เครื่องแม่ข่ายพร้อมกับการร้องขอเซอร์วิสเมื่อต้องการใช้งาน ซึ่งลักษณะการทำงานเช่นนี้จะช่วยทำให้ระบบสามารถขยายการให้เซอร์วิสในอนาคตได้ง่าย (Scalable)

กำหนดวิธีการใช้งานกระบวนการทำงานของ HTTP อย่างชัดเจน การใช้งานกระบวนการต่างๆของ HTTP นั้นจะต้องเป็นไปตามรายละเอียดที่แสดงในตารางที่ 2.1

ตารางที่ 2.1 มาตรฐานการใช้งานกระบวนการของ HTTP

กระบวนการ	รายละเอียด
POST	เป็นกระบวนการที่ใช้สั่งให้เครื่องแม่ข่ายทำการเก็บข้อมูลซึ่งสามารถเลือกได้ว่าจะส่งหรือไม่ส่งข้อมูลไปพร้อมกับการร้องขอก็ได้
GET	เป็นกระบวนการทำงานเพื่อเรียกดูข้อมูลที่ต้องการจากเครื่องแม่ข่าย
PUT	เป็นกระบวนการที่ใช้สั่งให้เครื่องแม่ข่ายทำการสร้างหรือแก้ไขข้อมูลด้วยข้อมูลที่ส่งมาพร้อมกับการร้องขอ
DELETE	เป็นกระบวนการที่ใช้สำหรับสั่งให้เครื่องแม่ข่ายทำการลบข้อมูล

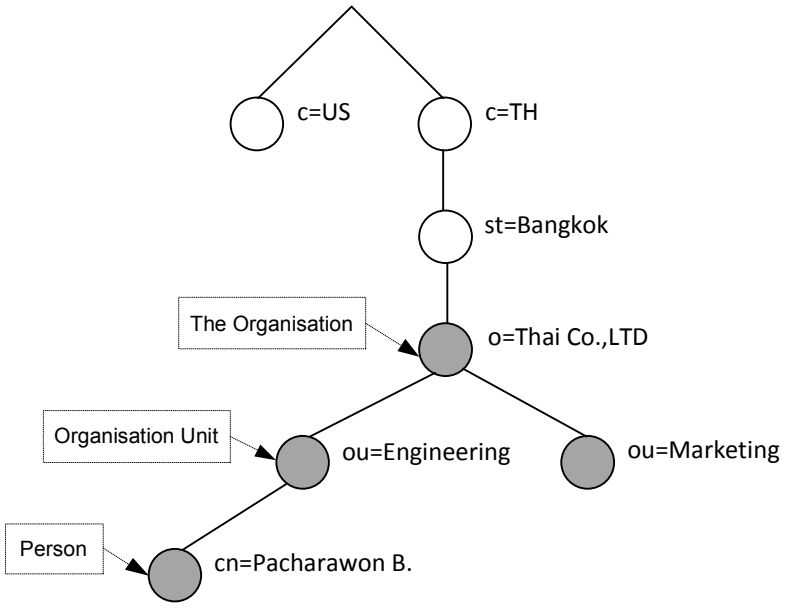
โดยกระบวนการ GET, PUT, DETETE จะต้องอ้างอิงตำแหน่งของทรัพยากรด้วยค่าของคีย์หลัก

สามารถใช้กับรูปแบบของข้อมูลมาตรฐานชนิดใดก็ได้ ส่วนแสดงเนื้อหาของการร้องขอและการตอบกลับนั้นสามารถเลือกใช้ข้อมูลชนิดใดก็ได้ที่สามารถอ้างอิง MIME-type และ Content-Type ได้เช่น HTML, Plain Text หรือ JSON เป็นต้น

2.6 โปรโตคอล LDAP

LDAP (Lightweight Directory Access Protocol) [29] เป็นโปรโตคอลที่พัฒนามาจากโปรโตคอล X.500 [30] ซึ่งใช้ในการเข้าถึงและแก้ไขไดเรคทอรี ซึ่งไดเรคทอรีนั้นอาจเรียกได้ว่าเป็นฐานข้อมูล (Database) แบบพิเศษ หรือเป็นโกดังข้อมูล (Data repository) ที่บรรจุรายละเอียดของทรัพยากร ต่างๆ เช่น ชื่อผู้ใช้งาน แอพพลิเคชัน เอกสารข้อมูล เครื่องพิมพ์ เป็นต้น โดยกระบวนการทำงานของ LDAP จะเน้นที่การเข้าถึงข้อมูลหรือ อ่านข้อมูล มากกว่าการแก้ไข หรือ เขียนข้อมูล และ LDAP จะไม่สนับสนุนการสอบถามข้อมูลด้วย SQL (Structured Query Language)

อย่างไรก็ตามถึงแม้การเก็บข้อมูลแบบไดเรคทอรีจะมีคุณสมบัติด้อยกว่าฐานข้อมูลทั่วไปหลายประการ แต่เนื่องจากโปรโตโคล LDAP มีความเร็วในการเข้าถึงข้อมูลสูง จึงทำให้เป็นที่ยอมรับ และนำมาใช้งานทั่วไป นอกจากประโยชน์ในการค้นหาข้อมูลได้อย่างรวดเร็ว ไดเรคทอรีของ LDAP มีการจัดโครงสร้างแบบลำดับชั้น (Hierarchical) โดยข้อมูลจะถูกบรรจุอยู่ในเอนทรี (Entries) ซึ่งแต่ละเอนทรีจะประกอบด้วยแอททริบิวต์ (Attribute) ในรูปของเครื่องหมายเท่ากับ (=) โดยประเภท (type) จะถูกกำหนดไว้ด้วยตัวระบุวัตถุ (Object Identifier, OID) ส่วนค่าของแอททริบิวต์ (value) ก็จะมีรูปแบบที่ระบุไว้อย่างชัดเจน



รูปที่ 2.11 ตัวอย่างโครงสร้างไดเรคทอรีของ LDAP

เอนทรีจะถูกจัดไว้เป็นลำดับชั้นด้วย DN (Distinguished Name)โดยเอนทรีใดๆ ที่อยู่ใต้เอนทรีอื่นจะมี DN ของเอนทรีอื่นเป็นข้อความที่ตามหลัง (Suffix) ของเอนทรีนั้น รูปแบบของไดเรคทอรีจะระบุ DN และระบุว่า แต่ละเอนทรีจะประกอบไปด้วยแอททริบิวต์ใดบ้าง ดังแสดงให้เห็นในรูปที่ 2.11

แอททริบิวต์ส่วนใหญ่จะมีการใช้ตัวอักษรย่อระบุประเภทซึ่งได้แก่ UID = User ID, CN = Common Name, SN = Surname, L = Location, OU = Organizational Unit, O = Organization, DC = Domain Component, ST = State, C = Country เป็นต้น โดยข้อมูลเพิ่มเติมในส่วนนี้สามารถหาได้จาก RFC2256 [31]

2.7 งานวิจัยที่เกี่ยวข้อง

การนำสถาปัตยกรรมเชิงบริการไปใช้งานนั้นสามารถทำได้จริงหากเพียงแต่ต้องเข้าใจในกระบวนการที่มีอยู่เพื่อประยุกต์ใช้ตามแนวความคิดของสถาปัตยกรรมเชิงบริการ ซึ่งอาจจะเป็นการนำไปใช้ทั้งระบบก็ดี หรือการเริ่มนำไปใช้ในการเปลี่ยนแปลงเพียงบางส่วนก็ดี ดังตัวอย่างเช่น ในกรณีศึกษาของการปรับเปลี่ยนสถาปัตยกรรมเดิมที่ใช้อยู่และประยุกต์ใช้สถาปัตยกรรมเชิงบริการของบริษัทเอบีซี (A Case Study in SOA and Re-Architecture at Company ABC) [32] ที่นำเรื่องของ System E ที่มีลักษณะของระบบขนาดใหญ่เพื่อใช้เป็นระบบคลังข้อมูล (Data Warehouse) เพื่อมาจัดการบริหารความเสี่ยงของบริษัทซึ่งระบบมีลักษณะเป็นนิชเทคโนโลยี (Niche Technology) ที่มีลักษะเฉพาะบนระบบตัวเอง กล่าวคือยังมีข้อจำกัดทั้งเรื่องการใช้งาน และ ข้อจำกัดในด้านการขยายระบบอีกด้วย หลังจากนั้นมีการเสนอให้ทำการปรับปรุง และพัฒนา System E ให้เป็นไปตามรูปแบบของสถาปัตยกรรมเชิงบริการ โดยงานวิจัยดังกล่าวแสดงให้เห็นถึงการเปรียบเทียบระบบในปัจจุบันกับระบบใหม่ที่เป็นสถาปัตยกรรมเชิงบริการ และมีการจัดทำระบบนำร่อง (Pilot system) ซึ่งแนวทางการปรับปรุงดังกล่าวได้รับความเห็นชอบจากผู้บริหารระดับสูงเป็นอย่างดี

หรือตัวอย่างงานวิจัยเกี่ยวกับการนำสถาปัตยกรรมเชิงบริการไปประยุกต์ใช้สำหรับธุรกิจขนาดเล็ก (Scaling Down SOA to Small Businesses) [33] ซึ่งเป็นการนำเสนอทางเลือกใหม่ของการปรับขนาด (Scaling) ของสถาปัตยกรรมเชิงบริการให้เล็กลงเพื่อให้เหมาะกับธุรกิจขนาดเล็ก โดยเรียกแนวความคิดดังกล่าวว่า “Outside-In” ซึ่งแนวความคิดดังกล่าวจะทำให้ธุรกิจขนาดเล็กสามารถนำ

สถาปัตยกรรมเชิงบริการไปประยุกต์ใช้งานได้ โดยที่งานวิจัยดังกล่าวยังชี้ให้เห็นว่าการนำสถาปัตยกรรมเชิงบริการมาใช้นั้นเกิดประโยชน์ทั้งในเรื่องของการลดต้นทุนและเพิ่มความคล่องตัวสำหรับธุรกิจอีกด้วย

บทที่ 3

แนวทางสำหรับการพัฒนาระบบ

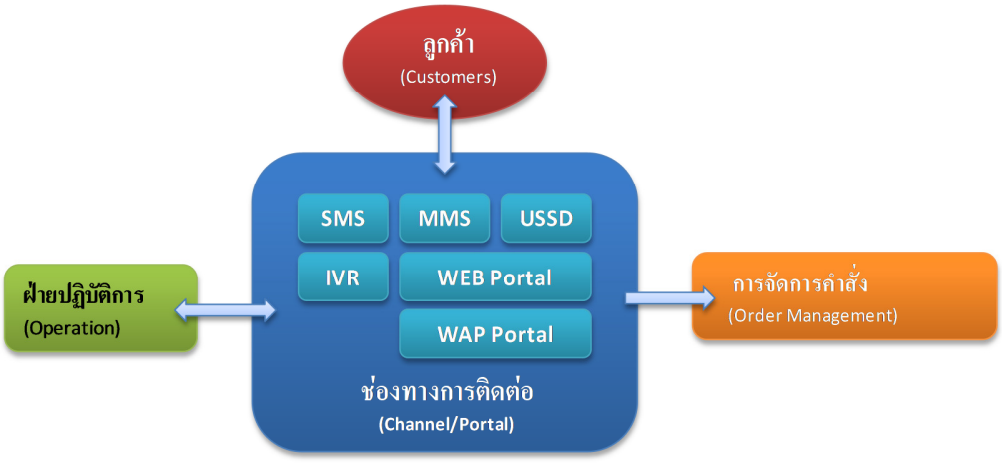
3.1 การออกแบบโครงสร้างระบบ

ในการพัฒนาระบบให้ไปเป็นตามสถาปัตยกรรมเชิงบริการนั้น จำเป็นต้องมีการทำความเข้าใจถึงโครงสร้างระบบที่มีอยู่ในปัจจุบันเพื่อแบ่งประเภทการใช้งานและบริการออกเป็นกลุ่มๆ ตัวอย่างสำหรับระบบสำหรับผู้ให้บริการเครือข่ายโทรศัพท์เคลื่อนที่แล้วนั้น สามารถจัดแบ่งระบบต่างๆ ที่สำคัญในการจัดการ และ การให้บริการลูกค้าออกเป็น กลุ่มของระบบช่องทางในการติดต่อ (Channel/Portal) กลุ่มระบบจัดการคำสั่ง (Order Management) กลุ่มระบบการจัดเตรียมบริการ (Provisioning Management) กลุ่มระบบการจัดการข้อมูลลูกค้า (Profile Management) กลุ่มระบบการจัดการเรื่องค่าใช้จ่าย (Charging Management) และ กลุ่มของระบบที่ทำหน้าที่ให้บริการแก่ลูกค้า (Content/Service) โดยที่แต่ละกลุ่มอาจจะประกอบไปด้วยระบบย่อยๆ ที่ทำหน้าที่ต่างกันไป ตามรายละเอียดดังต่อไปนี้

กลุ่มของระบบช่องทางในการติดต่อ (Channel/Portal) มีหน้าที่ในการเป็นส่วนติดต่อ (Interface) กับลูกค้าโดยตรง ซึ่งเมื่อลูกค้าต้องการขอใช้บริการก็จะทำการร้องขอบริการผ่านช่องทางต่างๆ อย่างเช่น SMS (Short Message Service), IVR (Interactive voice response), USSD (Unstructured Supplementary Service Data), MMS (Multimedia Messaging Service), WEB Portal และ WAP Portal โดยผ่านโทรศัพท์เคลื่อนที่ หรือคอมพิวเตอร์ที่เชื่อมต่ออินเทอร์เน็ต หลังจากระบบช่องทางการติดต่อได้รับการร้องขอบริการก็จะทำการส่งคำสั่งต่อให้กับระบบที่มีหน้าที่จัดการคำสั่งเพื่อดำเนินการต่อไป ดังแสดงตัวอย่างความสัมพันธ์ในรูปที่ 3.1 ส่วนฝ่ายปฏิบัติการจะมีหน้าที่ดูแล และแก้ปัญหาของระบบให้สามารถใช้งานได้เป็นปกติ

กลุ่มของระบบจัดการคำสั่ง (Order Management) มีหน้าที่ในการดูแลจัดการตรวจสอบความถูกต้องของคำสั่งที่มีการส่งเข้ามาซึ่งอาจจะมีการตรวจสอบข้อมูลลูกค้าผ่านทางระบบจัดการข้อมูล

ลูกค้า (Profile Management) แล้วทำการจัดคิวของการคำสั่งเพื่อรอขั้นตอนถัดไปโดยที่เมื่อถึงคิวของคำสั่งดังกล่าวก็จะมีการหักค่าใช้บริการ และส่งต่อไปยังระบบที่ทำการจัดเตรียมบริการ (Provisioning Management) ในกรณีที่ไม่สามารถจัดเตรียมบริการให้ลูกค้าได้นั้น ระบบจัดการคำสั่งก็จะทำหน้าที่คืนเงินค่าบริการที่หักไปแล้วให้กับลูกค้าเช่นเดียวกัน ความสัมพันธ์ดังแสดงในรูปที่ 3.2



รูปที่ 3.1 ความสัมพันธ์ของระบบในกลุ่มช่องทางการติดต่อ

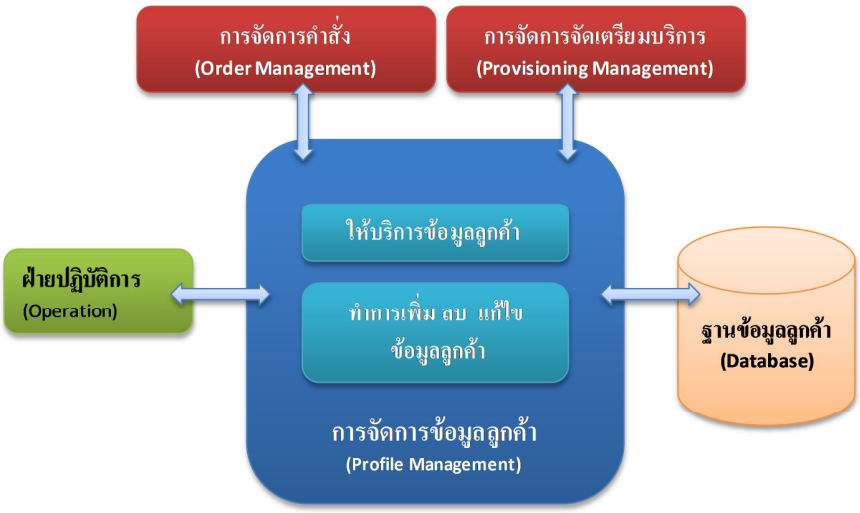
รูปที่ 3.2 ความสัมพันธ์ของระบบในกลุ่มการจัดการคำสั่ง

กลุ่มระบบจัดการเพื่อการจัดเตรียมบริการ (Provisioning Management) มีหน้าที่ตรวจสอบความถูกต้องของคำสั่งที่ได้รับมาจากระบบจัดการคำสั่ง หลังจากนั้นจะทำการจัดคิวสำหรับคำสั่งต่างๆ และทำการส่งคำสั่งไปจัดการบริการให้แก่ลูกค้าผ่านระบบให้บริการลูกค้า (Content/Service) และทำการแก้ไขข้อมูลลูกค้าให้ถูกต้อง ดังแสดงความสัมพันธ์ในรูปที่ 3.3

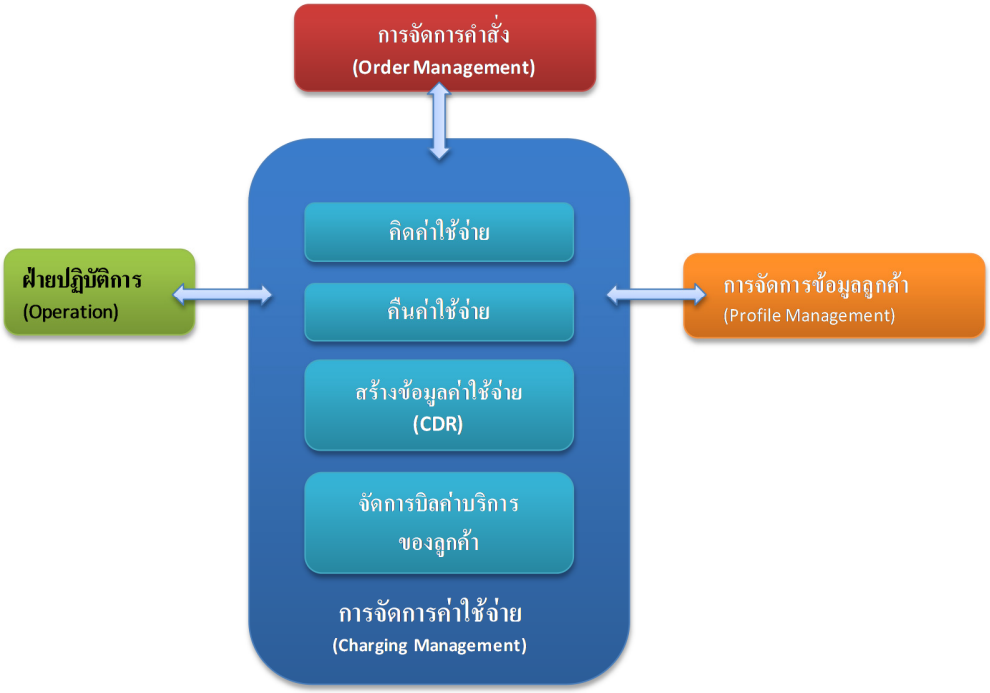
รูปที่ 3.3 ความสัมพันธ์ของระบบในกลุ่มการจัดเตรียมบริการ

กลุ่มระบบการจัดการข้อมูลลูกค้า (Profile Management) มีหน้าที่ให้บริการข้อมูลลูกค้า หรือทำการเพิ่ม ลบ แก้ไข ข้อมูลลูกค้า โดยจะรับคำสั่งจากระบบการจัดการคำสั่ง และ ระบบการจัดการจัดเตรียมบริการ ดังแสดงในรูปที่ 3.4

กลุ่มจัดการค่าใช้จ่าย (Charging Management) มีหน้าที่คิดคำนวณและหักค่าใช้จ่าย หรือคืนค่าใช้จ่ายให้ลูกค้า ทำการสร้างข้อมูลค่าใช้จ่าย (CDR) และ จัดการเรื่องบิลค่าใช้จ่ายของลูกค้า โดยมีการเชื่อมต่อกับระบบการจัดการข้อมูลลูกค้าเพื่อขอดูข้อมูลลูกค้า ดังแสดงในรูปที่ 3.5

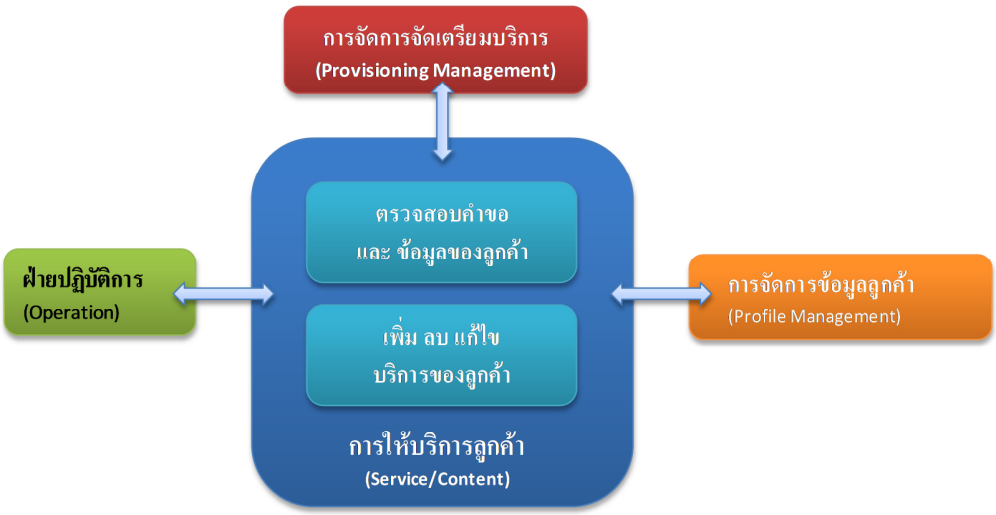


รูปที่ 3.4 ความสัมพันธ์ของระบบในกลุ่มการจัดการข้อมูลลูกค้า



รูปที่ 3.5 ความสัมพันธ์ของระบบในกลุ่มการจัดการค่าใช้จ่าย

กลุ่มระบบให้บริการ (Service/Content) มีหน้าที่ตรวจสอบการให้บริการแก่ลูกค้า และ ทำการเพิ่ม ลบ แก้ไข บริการตามคำสั่งของบริการที่รับมาจากระบบการจัดการจัดเตรียมบริการ โดยมีการเชื่อมต่อกับระบบจัดการข้อมูลลูกค้าเพื่อทำการขอข้อมูลมาตรวจสอบ ดังแสดงในรูปที่ 3.6



รูปที่ 3.6 ความสัมพันธ์ของระบบในกลุ่มการให้บริการลูกค้า

จากความสัมพันธ์ทั้งหมดของระบบที่มีอยู่จะเห็นได้ว่าการให้บริการแกลูกค้านั้นมีความยุ่งยากซับซ้อนและต้องผ่านระบบหรืออุปกรณ์หลายประเภท ในกรณีของโครงสร้างระบบโดยรวมนั้น เราจึงควรที่จะปรับปรุงสถาปัตยกรรมโครงสร้างการเชื่อมต่อของระบบให้ยืดหยุ่นมากขึ้น เพื่อรองรับการขยายระบบในอนาคต ซึ่งจะส่งผลต่อการลดค่าใช้จ่ายในระยะยาวอีกด้วย

จากแนวความคิดของสถาปัตยกรรมเชิงบริการที่มีการแบ่งชั้นของระบบออกเป็น 4 ชั้น อันได้แก่ชั้นทรัพยากร (Resource Layer) ชั้นเซอร์วิส (Service Layer) ชั้นกระบวนการ (Process Layer) และ ชั้นการเรียกใช้งาน (Access Layer) ดังนั้นเมื่อพิจารณาจากหน้าของแต่ละระบบที่กล่าวมาข้างต้น เราสามารถจัดแบ่งระบบต่างๆใหม่ให้เป็นไปในแนวทางของสถาปัตยกรรมเชิงบริการดังต่อไปนี้

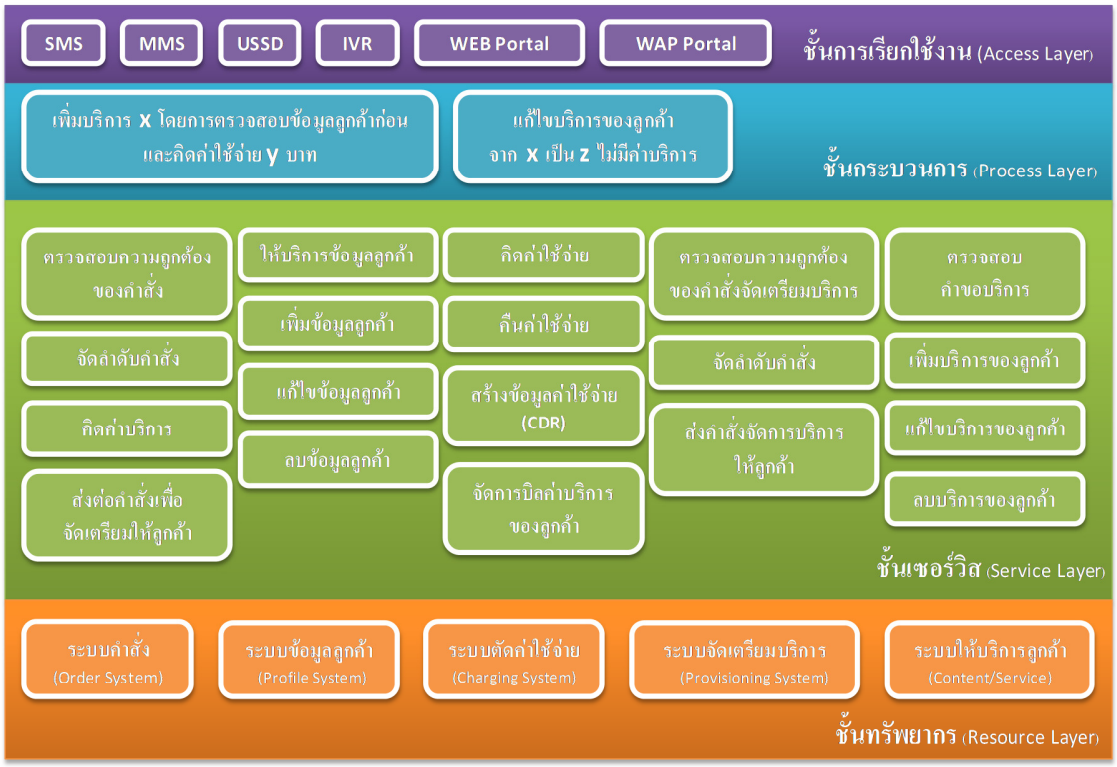
ชั้นทรัพยากร (Resource Layer) ประกอบด้วยระบบทั้งหมดมีที่อยู่รวมถึง ฐานข้อมูลต่างๆ ด้วย

ชั้นเซอร์วิส (Service Layer) เป็นการแยกเอากระบวนการทำงานจากระบบต่างๆในชั้นทรัพยากรที่สามารถนำกลับมาใช้ใหม่ได้ มาสร้างเป็นเซอร์วิสเพื่อให้ง่าย และ ยืดหยุ่น ต่อการเรียกใช้

ชั้นกระบวนการ (Process Layer) เป็นการนำเซอร์วิสหลายๆอย่างมาประกอบรวมกันเป็นกระบวนการทางธุรกิจเพื่อให้เรียกใช้งานผ่านชั้นการเรียกใช้งานได้ง่ายขึ้นและตรงตามความต้องการ

ชั้นการเรียกใช้งาน (Access Layer) เป็นการแยกระบบที่เป็นช่องทางการติดต่อมารวมกันไว้ในชั้นนี้เพื่อติดต่อกับชั้นกระบวนการเพื่อให้ได้บริการที่ต้องการ

ในรูปที่ 3.7 เป็นตัวอย่างที่แสดงให้เห็นถึงผลการวิเคราะห์ระบบตามแนวทางของสถาปัตยกรรมเชิงบริการ ซึ่งสามารถแบ่งระบบ และ กระบวนการออกเป็น 4 ลำดับชั้น โดยจะเห็นว่าผู้ให้บริการสามารถกำหนดช่องทางในการให้บริการสำหรับการซื้อบริการ x ซึ่งมีค่าบริการ y บาทผ่านช่องทางใดก็ได้ที่มีอยู่ในชั้นการเรียกใช้งาน และจากนั้นทำการสร้างกระบวนการทางธุรกิจ (Business Process) โดยการเรียกใช้เลือกเซอร์วิสที่ต้องทำจากชั้นเซอร์วิส เพื่อให้ได้ “เพิ่มช่องทางในการใช้บริการ x สำหรับลูกค้าที่ร้องขอการใช้บริการ โดยการตรวจสอบข้อมูลเครดิตของลูกค้า และคิดค่าใช้จ่าย y บาทจากบัญชีของลูกค้ารายนั้น” ที่อยู่ในชั้นกระบวนการ ซึ่งเซอร์วิสต่างๆที่อยู่ในชั้นเซอร์วิสนั้นจะทำเป็นเซอร์วิสที่ขึ้นอยู่กับระบบในชั้นทรัพยากร เป็นต้น



รูปที่ 3.7 การออกแบบระบบโดยการแบ่งกระบวนการออกเป็น 4 ลำดับชั้นตามแนวความคิดของสถาปัตยกรรมเชิงบริการ

จะเห็นได้ว่าเมื่อมีการออกแบบสถาปัตยกรรมเชิงบริการแล้วนั้น ทำให้เราสามารถที่จะสร้างกระบวนการทางธุรกิจขึ้นมาได้รวดเร็วและใช้งบประมาณที่ไม่มากในกรณีที่มีบริการในชั้นบริการครอบคลุมอยู่แล้ว ซึ่งในกรณีที่บริการเป็นบริการใหม่และไม่มีระบบในชั้นทรัพยากรสามารถรองรับได้นั้น ก็เพียงออกแบบระบบใหม่ในรูปแบบสถาปัตยกรรมเชิงบริการโดยเอากระบวนการต่างๆที่ระบบใหม่จะทำได้มาจัดทำอยู่ในชั้นบริการ จากนั้นก็สามารถสร้างกระบวนการทางธุรกิจใหม่ตามที่ต้องการ

3.2 การออกแบบบริการ

สำหรับการออกแบบบริการตามสถาปัตยกรรมเชิงบริการนั้นนิยมใช้การพัฒนาซอฟต์แวร์คอมโพเนนท์ (Software Component) ในรูปแบบที่เรียกว่าเว็บเซอร์วิส ซึ่งมีการทำงานคือสามารถรับอินพุตเข้ามาเพื่อประมวลผล และส่งผลลัพธ์กลับออกไป ซึ่งบริการเหล่านี้จะเน้นที่การประมวลผลเพื่อให้ได้ผลลัพธ์ตามที่ต้องการเพียงอย่างเดียว ด้วยมาตรฐานการพัฒนาเว็บเซอร์วิสนั้นทำให้เว็บเซอร์วิสเองสามารถใช้งานได้โดยไม่ยึดติดกับภาษาคอมพิวเตอร์ที่นำมาใช้ในการพัฒนาซอฟต์แวร์ เพียงแต่ผู้พัฒนาต้องทำซอฟต์แวร์ดังกล่าวให้เป็นเว็บเซอร์วิสเท่านั้น โดยที่ระบบดังกล่าวจะถูกเรียกว่า “แม่ข่ายเว็บเซอร์วิส (Web Service Server)” และผู้ที่ทำการเรียกใช้งานบริการดังกล่าวจะถูกเรียกว่า “เครื่องลูกข่าย (Client)” โดยที่ขั้นตอนของการพัฒนาบริการในรูปแบบของเว็บเซอร์วิสมีดังต่อไปนี้

3.2.1 การแจกแจงบริการทั้งหมดที่ระบบสามารถทำได้

หลักจากการจัดโครงสร้างระบบตามสถาปัตยกรรมเชิงบริการแล้วนั้นให้แจกแจงกระบวนการหรือ ฟังก์ชันงาน ให้ได้กระบวนการที่ย่อยเล็กที่สุดโดยพิจารณาจากผลลัพธ์ของกระบวนการดังกล่าวที่แต่ละระบบในชั้นทรัพยากรนั้นทำได้ ซึ่งในที่นี้จะยกตัวอย่างระบบที่ให้บริการเสียงเพลงรอสาย (Ring Back Tone, RBT) มาประกอบการอธิบายขั้นตอนของการออกแบบเว็บเซอร์วิส โดยที่ฟังก์ชันงานที่ระบบสามารถแจกแจงออกมาได้มีดังนี้

- เปิดบริการเสียงเพลงรอสาย (Subscription)
- ปิดบริการเสียงเพลงรอสาย (Un-subscription)
- แก้ไขข้อมูลสมาชิกที่เปิดบริการเสียงเพลงรอสายไว้

- ดาวน์โหลด (Download) เพลงเข้ากล่องเพลง (Inbox)
- ลบเพลงออกจากกล่องเพลง
- เลือกดูรายการเพลงทั้งหมดที่อยู่ในกล่องเพลง
- ตั้งค่าเพลงให้เป็นเสียงรอสาย
- ตั้งค่าเพลงเพื่อยกเลิกการเป็นเสียงรอสาย
- เลือกดูรายการเพลงที่ตั้งค่าไว้เป็นเสียงรอสาย

3.2.2 การกำหนดรายละเอียดของบริการ

หลังจากที่แจกแจงบริการแต่ละประเภทออกจากระบบได้แล้วนั้น ต้องมีการการกำหนดรายละเอียดของบริการโดยที่ต้องทราบอินพุตทั้งหมดที่บริการนั้นๆต้องการเพื่อทำการประมวลผลให้ได้ผลลัพธ์ออกมาเป็นเอาท์พุตของแต่ละบริการ โดยที่เราเลือกใช้เอกสาร WSDL เวอร์ชัน 2.0 [12,13] เป็นภาษาในการใช้อธิบายรายละเอียดของบริการเนื่องจากสนับสนุนการการอธิบายบริการทั้งเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP และ เว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST ตัวอย่างเช่น ฟังก์ชั่นงานเปิดบริการเสียงเพลงรอสาย (Subscription) ต้องการค่าอินพุตทั้งหมด 9 ค่าซึ่งประกอบไปด้วย หมายเลขโทรศัพท์เคลื่อนที่ เลขประจำตัวประชาชน ชื่อ นามสกุล ประเทศ รหัสไปรษณีย์ อีเมล์ และวันที่ในการทำรายการ โดยการใช้หมายเลขโทรศัพท์เป็นคีย์หลัก (Primary Key, PK) ของการอ้างอิง ซึ่งมีเอาท์พุตเป็นผลลัพธ์ของการเปิดบริการเสียงเพลงรอสาย

3.2.3 พัฒนาเว็บเซอร์วิสตามรายละเอียดของบริการ

พัฒนาเว็บเซอร์วิสให้รับอินพุตตามรายละเอียดของบริการที่มีการกำหนดไว้ในเอกสาร WSDL จากนั้นนำอินพุตไปประมวลผลตามจุดประสงค์ของบริการนั้นๆ และทำการส่งค่าผลลัพธ์กลับไปด้วยค่าเอาท์พุตที่กำหนดไว้

3.3 หลักการพัฒนาเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST

เนื่องจาก REST เป็นแนวความคิดด้านสถาปัตยกรรมทางซอฟต์แวร์ ไม่ใช่โปรโตคอลมาตรฐานสำหรับเว็บเซอร์วิสที่ชัดเจนเหมือนอย่างมาตรฐานการพัฒนาเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP ดังนั้นในวิทยานิพนธ์เล่มนี้จึงได้มีการกำหนดหลักการออกแบบเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST ให้เหมาะสำหรับการนำไปใช้ออกแบบบริการสำหรับผู้ให้บริการโทรศัพท์เคลื่อนโดยคำนึงถึงความเร็วในการให้บริการที่ค่อนข้างจำกัดและ โครงสร้างของบริการทั้งหมดที่มีหมายเลขโทรศัพท์เคลื่อนเป็นคีย์ที่ระบุถึงสมาชิกในระบบ

พื้นฐานของการออกแบบด้วยแนวทางของสถาปัตยกรรม REST นั้นเน้นมุ่งเน้นที่การพัฒนาบริการที่เครื่องแม่ข่ายเว็บเซอร์วิสจะไม่มีการจัดเก็บสถานะการทำงาน (Stateless) กำหนดวิธีการใช้งานกระบวนการทำงานของ HTTP (POST, GET, PUT และ DELETE) อย่างชัดเจนให้ถูกต้องและเหมาะสมกับประเภทของบริการ การใช้รูปแบบของ URL ที่มีลักษณะคล้ายกับโครงสร้างของไดเรคทอรี่เพื่อเข้าถึงที่อยู่ของบริการ และ การเลือกใช้ข้อมูลมาตรฐานชนิดใดก็ได้ในการแสดงเนื้อหาของการร้องขอบริการและการตอบกลับ ซึ่งมีรายละเอียดดังต่อไปนี้

3.3.1 ไม่มีการจัดเก็บสถานะการทำงาน (Stateless)

การทำงานของบริการที่สอดคล้องกับสถาปัตยกรรม REST คือ เครื่องแม่ข่ายเว็บเซอร์วิสจะจัดเก็บเฉพาะข้อมูลตามที่เครื่องลูกข่ายส่งมา โดยที่จะไม่มีการเก็บสถานะการทำงานใดๆของการใช้บริการของผู้ใช้ไว้ ซึ่งในกรณีที่บริการดังกล่าวจำเป็นต้องมีการเก็บข้อมูลเซสชัน หรือข้อมูลที่มีการเปลี่ยนแปลง ณ ปัจจุบัน ของการใช้บริการของผู้ใช้นั้น สามารถทำได้ด้วยการเก็บไว้ หรือ สร้างข้อมูลที่เครื่องลูกข่ายเอง เช่น วันเวลา ลำดับหน้าที่ผู้ใช้งานเปิดอยู่ สถานการณ์เข้า-ออกระบบ เป็นต้น แล้วส่งมายังเครื่องแม่ข่ายพร้อมกับการร้องขอบริการเมื่อต้องการใช้งาน ซึ่งลักษณะการทำงานเช่นนี้จะช่วยทำให้ระบบสามารถขยายการให้บริการในอนาคตได้ง่าย (Scalable)

3.3.2 กำหนดวิธีการใช้งานกระบวนการทำงานของ HTTP อย่างชัดเจน

จากตัวอย่างที่มีการแจกแจงฟังก์ชันงานของระบบเสียงเพลงรอสายทั้งหมด 9 ฟังก์ชัน สามารถกำหนดกระบวนการทำงานของ HTTP (POST, GET, PUT และ DELETE) ตามฟังก์ชันงานได้ ดังที่แสดงให้เห็นในตารางที่ 3.1

ตารางที่ 3.1 ผลการวิเคราะห์เพื่อจับคู่เว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST กับกระบวนการ HTTP

เซอร์วิส	กระบวนการ HTTP
เปิดบริการเสียงเพลงรอสาย (Subscription)	POST
ปิดบริการเสียงเพลงรอสาย (Un-subscription)	DELETE
แก้ไขข้อมูลสมาชิกที่เปิดบริการเสียงเพลงรอสายไว้	PUT
ดาวน์โหลด (Download) เพลงเข้ากล่องเพลง (Inbox)	POST
ลบเพลงออกจากกล่องเพลง	DELETE
เลือกดูรายการเพลงทั้งหมดที่อยู่ในกล่องเพลง	GET
ตั้งค่าเพลงให้เป็นเสียงรอสาย	POST
ตั้งค่าเพลงยกเลิกการเป็นเสียงรอสาย	DELETE
เลือกดูรายการเพลงที่ตั้งค่าไว้เป็นเสียงรอสาย	GET

3.3.3 การกำหนดรูปแบบ URL ที่คล้ายกับโครงสร้างไดเรคทอรี่

วิธีการเรียกใช้งานเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST นั้นจะเหมือนกับการร้องขอแบบ HTTP ทั่วไป ดังนั้นการอ้างอิงถึงเว็บเซอร์วิสใดๆก็ตามสามารถทำได้ด้วยการเรียกไปที่ URL ของเซอร์วิสที่ต้องการควบคู่กับการจับคู่กระบวนการ HTTP ซึ่งจะทำให้ทราบว่าต้องการทำอะไรกับเซอร์วิสหรือทรัพยากรของเซอร์วิสนั้นๆ โดยพื้นฐานการกำหนดรูปแบบ URL จะใช้รูปแบบที่คล้ายกับโครงสร้างของไดเรคทอรี่ (Directory) เพื่อให้เป็นระเบียบเรียบร้อย และง่ายต่อการอ้างถึงทรัพยากรที่ต้องการ จากการวิเคราะห์ถึงรูปแบบของ URL ที่เหมาะกับการให้บริการสำหรับผู้ให้บริการ

โทรศัพท์เคลื่อนที่โดยใช้หมายเลขโทรศัพท์เป็นคีย์ในการอ้างอิงนั้นดังแสดงให้เห็นในรูปที่ 3.8 โดยมีรายละเอียดดังต่อไปนี้

IP คือหมายเลขไอพีของเครื่องแม่ข่ายเว็บเซอร์วิส หรืออ้างอิงเป็นโดเมน (Domain) ของเครื่องแม่ข่ายเว็บเซอร์วิสก็ได้เช่นกัน

PORT ใช้สำหรับอ้างอิงหมายเลขพอร์ตบนเครื่องแม่ข่ายที่เซอร์วิสดังกล่าวทำงานอยู่

หมายเลขโทรศัพท์ ใช้อ้างอิงข้อมูลลูกค้าซึ่งจะเป็นหมายเลขทั้งหมดที่ไม่ซ้ำกันเลยในระบบ

บริการ ใช้ระบุบริการหลัก ซึ่งในที่นี้เราใช้บริการ RBT (บริการเสียงเพลงรอสาย)

บริการย่อย เป็นเหมือนส่วนประกอบย่อยๆ ในบริการหลักนั้น จากตัวอย่างบริการเสียงเพลงรอสายนั้นบริการย่อยจะมี INBOX (กล่องเพลง) และ SETTING (การตั้งค่าเสียงเพลงรอสาย)

ทรัพยากรของบริการย่อย เป็นทรัพยากรที่อยู่ในบริการย่อย จากตัวอย่างในบริการย่อย INBOX และ SETTING จะมีทรัพยากรของบริการย่อยเป็นรหัสของเพลง (Tonecode) เช่น 10002345 เป็นต้น

HTTP : / / IP:PORT / หมายเลขโทรศัพท์ / บริการ */ {บริการย่อย} / {ทรัพยากรของบริการย่อย}*

รูปที่ 3.8 รูปแบบ URL ที่คล้ายกับโครงสร้างไดเรคทอรี่

ตัวอย่าง URL ที่ถูกกำหนดตามรูปแบบที่กล่าวข้างต้นแสดงในเห็นตามลำดับต่อไปนี้

เปิดบริการเสียงเพลงรอสาย (Subscription) ใช้กระบวนการ POST ไปที่

HTTP : / / Server : 80 / 0810987654 / RBT

ปิดบริการเสียงเพลงรอสาย (Un-subscription) ใช้กระบวนการ DELETE ไปที่

HTTP : / / Server : 80 / 0810987654 / RBT

แก้ไขข้อมูลสมาชิกที่เปิดบริการเสียงเพลงรอสายไว้ ใช้กระบวนการ PUT ไปที่

HTTP : / / Server : 80 / 0810987654 / RBT

ดาวน์โหลด (Download) เพลงเข้ากล่องเพลง (Inbox) ใช้กระบวนการ POST ไปที่

HTTP : / / Server : 80 / 0810987654 / RBT / INBOX / 10002345

ลบเพลงออกจากกล่องเพลง ใช้กระบวนการ DELETE ไปที่

HTTP : / / Server : 80 / 0810987654 / RBT / INBOX / 10002345

เลือกดูรายการเพลงทั้งหมดที่อยู่ในกล่องเพลง ใช้กระบวนการ GET ไปที่

HTTP : / / Server : 80 / 0810987654 / RBT / INBOX

ตั้งค่าเพลงให้เป็นเสียงรอสาย ใช้กระบวนการ POST ไปที่

HTTP : / / Server : 80 / 0810987654 / RBT / SETTING / 10002345

ตั้งค่าเพลงเพื่อยกเลิกการเป็นเสียงรอสาย ใช้กระบวนการ DELETE ไปที่

HTTP : / / Server : 80 / 0810987654 / RBT / SETTING / 10002345

เลือกดูรายการเพลงที่ตั้งค่าไว้เป็นเสียงรอสาย ใช้กระบวนการ GET ไปที่

HTTP : / / Server : 80 / 0810987654 / RBT / SETTING

3.3.4 การกำหนดมาตรฐานของข้อมูลที่ใช้ในการร้องรอและตอบกลับ

การกำหนดมาตรฐานของข้อมูลที่ใช้ในการร้องรอและตอบกลับสามารถเลือกใช้รูปแบบของภาษามาตรฐานใดๆก็ได้มาใช้ ซึ่งในที่นี้เราเลือกใช้รูปแบบของภาษามาตรฐาน XML ในการแสดงส่วนเนื้อหาของการร้องขอและการตอบกลับ โดยอ้างอิงจากเนื้อหาของการร้องขอและการตอบกลับจาก เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP เพื่อให้ได้ค่าพารามิเตอร์ที่ครบถ้วนใกล้เคียงกับบริการส่วนใหญ่ที่มีอยู่ในระบบ เพื่อให้ได้ผลลัพธ์จากการร้องขอบริการของเว็บเซอร์วิสที่เหมือนกัน

3.4 การพัฒนาเซอร์วิสบัสสำหรับระบบเดิมที่มีอยู่

จากศึกษาและพิจารณาเพื่อเปลี่ยนแปลงสถาปัตยกรรมเดิมให้กลายเป็นสถาปัตยกรรมเชิงบริการโดยการพัฒนาระบบต่างๆที่เกิดขึ้นตามหลักการเชิงบริการและใช้วิธีการพัฒนาเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST ตามแนวการพัฒนาในหัวข้อ 3.3 แล้วนั้น แต่ยังมีปัญหาอีกส่วนหนึ่งคือระบบเดิมบางส่วนที่การแก้ไขให้เป็นเว็บเซอร์วิสนั้นเป็นไปได้ยาก ใช้ต้นทุนสูงในการพัฒนาเปลี่ยนแปลง หรืออาจจะไม่สามารถทำการแก้ไขได้เลย เพราะฉะนั้นส่วนสำคัญอีกส่วนหนึ่งคือในการพัฒนาระบบเดิมให้เป็นไปตามแนวทางของสถาปัตยกรรมเชิงบริการ คือการพัฒนาเซอร์วิสบัส (Service Bus) สำหรับระบบเดิมที่มีอยู่ โดยเซอร์วิสบัสที่พัฒนาขึ้นมาจะใช้ในการติดต่อระหว่างระบบในชั้นทรัพยากรและทำตัวเป็นผู้ให้บริการเซอร์วิสเพื่อให้ชั้นกระบวนการมาทำการเรียกใช้งานต่อไป

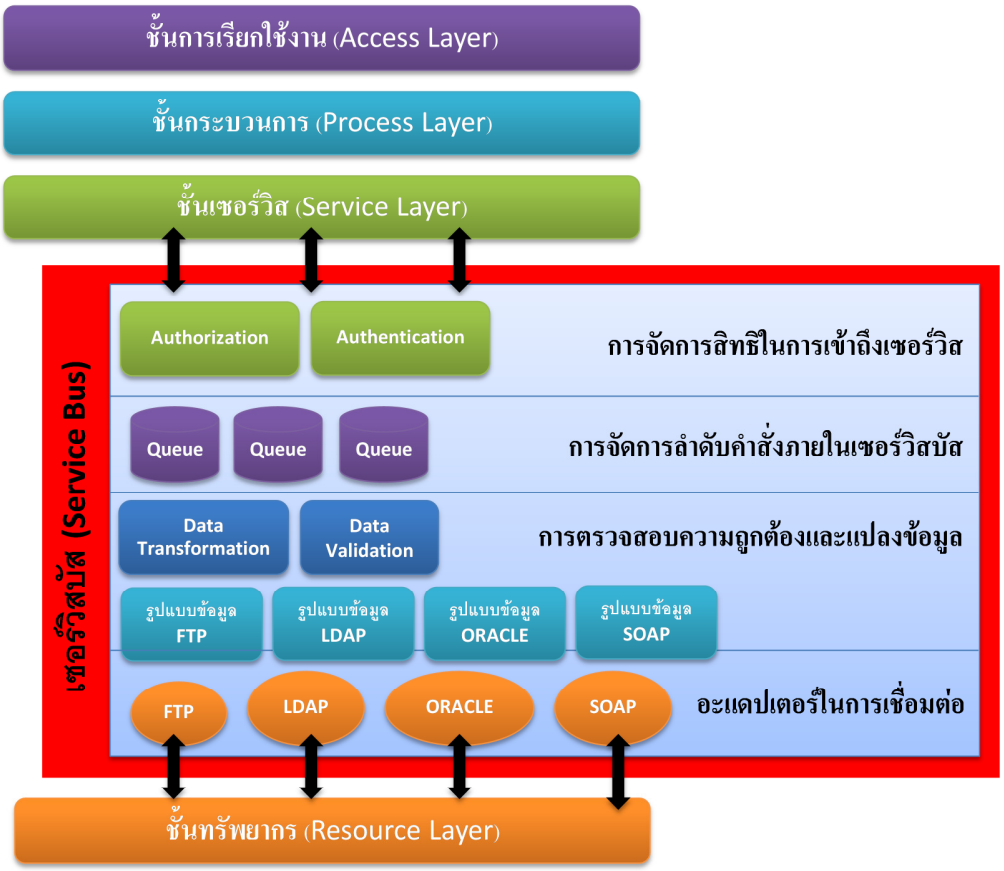
เซอร์วิสบัสประกอบไปด้วยอะแดปเตอร์ในการเชื่อมต่อ (Adapter) การตรวจสอบความถูกต้องและแปลงข้อมูล (Data Validator and Data Transformer) การจัดการลำดับคำสั่งภายในเซอร์วิสบัส (Queue Management) และ การจัดการสิทธิในการเข้าถึงเซอร์วิส (Authentication and Authorization Management) ดังแสดงให้เห็นในรูปที่ 3.9 โดยที่ส่วนประกอบต่อละตัวมีรายละเอียดในการพัฒนาดังต่อไปนี้

อะแดปเตอร์ในการเชื่อมต่อ (Adapter) เป็นตัวกลางที่ทำให้เซอร์วิสบัสสามารถเชื่อมต่อและสื่อสารกับระบบปลายทางได้ โดยที่อะแดปเตอร์อาจจะเป็นอะแดปเตอร์ในโปรโตคอลมาตรฐาน เช่น FTP, LDAP, SOAP, ORACLE เป็นต้น หรืออาจจะเป็นแดปเตอร์ที่ถูกพัฒนาขึ้นมาใช้เพื่อเชื่อมต่อกับระบบที่มีความเฉพาะตัวสำหรับองค์กรหนึ่งๆเท่านั้น เช่น อะแดปเตอร์สำหรับเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST เป็นต้น

การตรวจสอบความถูกต้องและแปลงข้อมูล (Data Validator and Data Transformer) เป็นกระบวนการที่ใช้ในการตรวจสอบข้อมูลที่จะส่งไปยังระบบปลายทางให้อยู่ในรูปแบบที่ถูกต้องตามที่ปลายทางกำหนดไว้ หรือ เป็นการแปลงข้อมูลจากรูปแบบหนึ่งไปยังรูปแบบที่ระบบปลายทางต้องการ ทั้งนี้ต้องมีการกำหนดรูปแบบของข้อมูลที่ระบบต้องการควบคู่ไปกับอะแดปเตอร์ในการเชื่อมต่อ

การจัดการลำดับคำสั่งภายในเซอร์วิสบัส (Queue Management) มีหน้าที่จัดการลำดับของคำสั่งที่เข้ามายังเซอร์วิสบัสแล้วส่งต่อไปยังกระบวนการตรวจสอบความถูกต้องและแปลงข้อมูลซึ่งเป็นกระบวนการที่จะรับรองว่าคำสั่งขอใช้เซอร์วิสจะได้รับบริการอย่างแน่นอน และ จะถูกกระจายไปยังระบบปลายทางได้อย่างเหมาะสมเพื่อเป็นการจัดการกระบวนการไม่ให้ส่งผลกระทบต่อระบบต่างๆ หรือกระทบกับเซอร์วิสบัสเองจนทำให้เกิดปัญหาเรื่องของประสิทธิภาพในการให้บริการ

การจัดการสิทธิในการเข้าถึงเซอร์วิส (Authentication and Authorization Management) เป็นกระบวนที่ใช้พิจารณาและให้สิทธิในการเข้าถึงเซอร์วิสของผู้ขอใช้บริการ เพื่อป้องกันเซอร์วิสบางอย่างที่เป็นระบบปิดที่ต้องการการยืนยันตัวตนก่อนการใช้งาน หรือบางเซอร์วิสที่เปิดให้ใช้เป็นสาธารณะก็จะส่งต่อไปยังกระบวนการจัดลำดับทันทีโดยไม่ต้องทำการตรวจสอบสิทธิในการเข้าถึงเซอร์วิส



รูปที่ 3.9 ส่วนประกอบของเซอร์วิสบัสและการนำไปใช้กับสถาปัตยกรรมเชิงบริการ

บทที่ 4

การทดสอบประสิทธิภาพของเว็บเซอร์วิส

4.1 หลักการทดสอบ

จากการกำหนดรูปแบบสำหรับการพัฒนาระบบดังที่ได้กล่าวมาแล้วนั้น เมื่อมีการปรับปรุงโครงสร้างของระบบให้เป็นไปตามแนวทางของสถาปัตยกรรมเชิงบริการ ซึ่งเน้นในเรื่องของความยืดหยุ่นในการให้บริการที่สามารถเพิ่มเติมแก้ไขบริการได้โดยมีค่าใช้จ่ายที่น้อยลง ซึ่งในส่วนของชั้นเซอร์วิสในสถาปัตยกรรมเชิงบริการได้กำหนดใช้เว็บเซอร์วิสภายใต้สถาปัตยกรรม REST ดังที่กล่าวมาแล้ว

ทั้งนี้การทดสอบ [34] ได้ถูกทดลองขึ้นเพื่อจะแสดงให้เห็นว่าเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST นั้นช่วยลดปริมาณข้อมูลที่ใช้ในการรับส่ง และยังเพิ่มประสิทธิภาพในการทำงานของเซอร์วิสให้ตอบสนองได้เร็วขึ้น ซึ่งจะเหมาะกับการให้บริการบนเครือข่ายที่เชื่อมต่อโดยโทรศัพท์เคลื่อนที่ที่มีข้อจำกัดในเรื่องของความเร็วและปริมาณการรับส่งข้อมูล

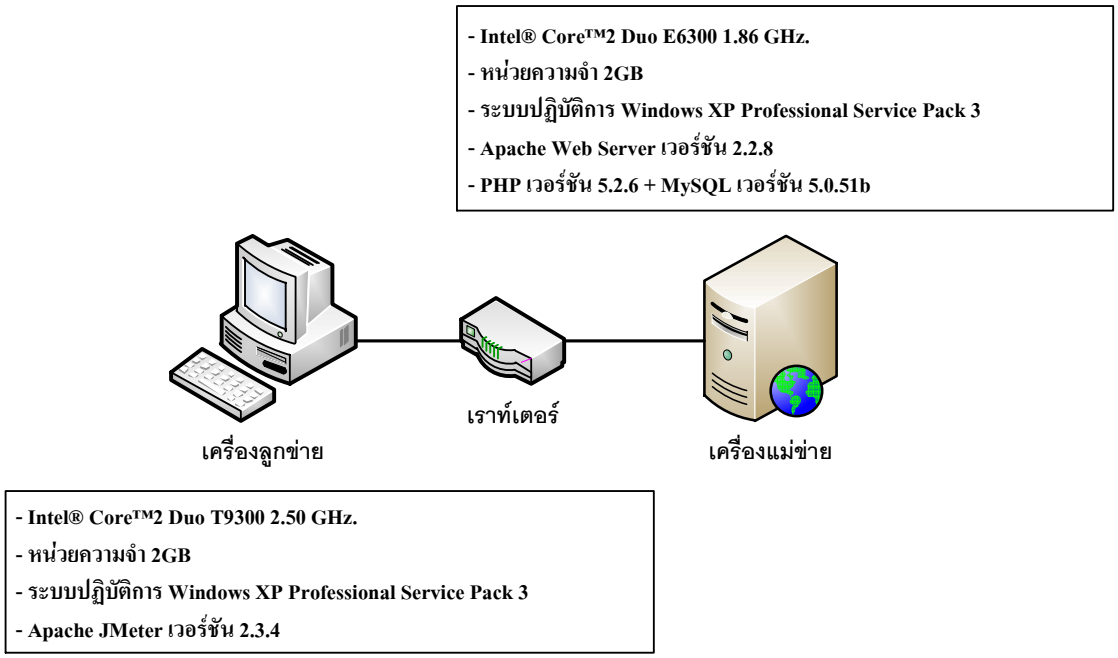
การทดสอบทำโดยการพัฒนาเว็บเซอร์วิสตัวอย่างขึ้นบนระบบทดสอบเพื่อเปรียบเทียบระหว่างประสิทธิภาพของการให้บริการเว็บเซอร์วิสด้วยรูปแบบตามมาตรฐานของ SOAP และรูปแบบตามแนวทางทฤษฎีของ REST ที่ได้กล่าวไว้ในบทที่ 3 โดยจะแสดงให้เห็นถึงวิธีการทดสอบและผลการทดสอบที่ได้จากการทดสอบ

4.2 ระบบที่ใช้ในการทดสอบ

ผู้วิจัยได้ทำการทดสอบประสิทธิภาพของเว็บเซอร์วิสที่ใช้สถาปัยกรรม REST โดยเปรียบเทียบกับเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP โดยทดสอบให้เครื่องลูกข่ายเชื่อมต่อกับเครื่องแม่ข่ายที่ให้บริการเว็บเซอร์วิสตามสถาปัตยกรรมทั้งสองแบบผ่านเราท์เตอร์ดังแสดงให้เห็นในรูปที่ 4.1

โดยเครื่องแม่ข่ายใช้คอมพิวเตอร์ที่มีหน่วยประมวลผลกลางเป็น Intel® Core™2 Duo E6300 1.86 GHz. หน่วยความจำ 2GB ทำงานบนระบบปฏิบัติการ Windows XP Professional Service Pack 3 และใช้ซอฟต์แวร์ Apache Web Server เวอร์ชัน 2.2.8 ที่รองรับภาษา PHP เวอร์ชัน 5.2.6 เป็นซอฟต์แวร์ระบบแม่ข่ายเว็บ โดยใช้ระบบจัดการฐานข้อมูล MySQL เวอร์ชัน 5.0.51b

และเครื่องลูกข่ายใช้คอมพิวเตอร์ที่มีหน่วยประมวลผลกลางเป็น Intel® Core™2 Duo T9300 2.50 GHz. หน่วยความจำ 2GB ทำงานบนระบบปฏิบัติการ Windows XP Professional Service Pack 3 และใช้ซอฟต์แวร์ Apache JMeter [35] เวอร์ชัน 2.3.4 ในการทดสอบและบันทึกผลการทดสอบ



รูปที่ 4.1 การเชื่อมต่อกันระหว่างเครื่องลูกข่ายและเครื่องแม่ข่ายที่ใช้ในการทดสอบ

4.3 แม่ข่ายเว็บเซอร์วิส

การทำงานของเว็บเซอร์วิสแบ่งออกเป็น 4 บริการ คือ CREATE, RETRIEVE, UPDATE และ DELETE ในที่นี้ใช้การจำลองการสร้างข้อมูลและเก็บข้อมูลสมาชิกซึ่งประกอบไปด้วย หมายเลขโทรศัพท์ เลขประจำตัวประชาชน ชื่อ นามสกุล ชื่อประเทศ รหัสไปรษณีย์ อีเมล์ และวันที่ในการทำรายการ โดยการใช้หมายเลขโทรศัพท์เป็นคีย์หลัก (Primary Key :PK) ของตารางดังกล่าว

บริการ CREATE ใช้ในการสร้างข้อมูลสมาชิก บริการ UPDATE ใช้แก้ไขข้อมูลสมาชิกที่มีอยู่แล้ว บริการ DELETE ใช้ในการลบข้อมูลสมาชิก โดยที่บริการทั้ง 3 นี้จะส่งผลลัพธ์ของการสร้าง การแก้ไข และการลบ กลับไปยังเครื่องลูกข่าย ส่วนบริการ RETRIEVE เป็นบริการในการเรียกดูข้อมูลสมาชิก ซึ่งในกรณีที่ไม่พบข้อมูลสมาชิกจะส่งรหัสที่บ่งบอกว่าไม่พบข้อมูลสมาชิกกลับไปยังเครื่องลูกข่ายเป็นผลลัพธ์

เว็บเซอร์วิสที่ใช้ทำการทดสอบนั้นพัฒนาด้วยภาษา PHP เวอร์ชัน 5 โดยใช้ฟังก์ชั่นมาตรฐานของภาษา PHP เองทั้งหมด โดยได้ทำการพัฒนาเว็บเซอร์วิสออกเป็นสองชุด ซึ่งเป็นไปตามสถาปัตยกรรม SOAP และสถาปัตยกรรม REST สำหรับเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP จะเรียกใช้งานคลาส SoapServer พร้อมอ้างอิงไฟล์ WSDL ที่ระบุถึงบริการจากบริการ 4 ชนิด (CREATE, RETRIEVE, UPDATE และ DELETE) ส่วนเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST นั้นใช้การร้องขอตามมาตรฐานของ HTTP ในภาษา PHP และใช้คลาส SimpleXMLElement ในการอ่านเนื้อหาของการร้องขอ โดยมีบริการ 4 บริการเช่นเดียวกันกับเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP

4.4 แม่ข่ายเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST

เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST ที่ใช้ทดสอบนั้นพัฒนาขึ้นด้วยการอ้างอิงตามพื้นฐาน 4 ข้อของทฤษฎีสถาปัตยกรรม REST ดังที่ได้กล่าวมาแล้วข้างต้น โดยการจับคู่บริการของเว็บเซอร์วิสตามสถาปัตยกรรม REST กับกระบวนการทำงานของ HTTP เป็นไปดังที่แสดงในตารางที่ 4.1

ตารางที่ 4.1 การจับคู่บริการเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST กับกระบวนการ HTTP

เซอร์วิส	กระบวนการ HTTP	รายละเอียดการทำงาน
CREATE	POST	สร้างข้อมูลสมาชิกในระบบ
RETRIEVE	GET	เรียกดูข้อมูลสมาชิกในระบบ
UPDATE	PUT	แก้ไขข้อมูลสมาชิกในระบบ
DELETE	DELETE	ลบข้อมูลสมาชิกในระบบ

โดยทั้งนี้การออกแบบเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST ในการทดสอบนั้น สถานะการทำงาน ค่าข้อมูลควบคุมการทำงานเซสชันของผู้ใช้งาน และพารามิเตอร์ทุกตัวจะถูกจัดการและส่งมาจากเครื่องลูกข่ายเท่านั้นโดยการอ้างอิงทรัพยากรและบริการของเว็บเซอร์วิสจะเป็นไปตามโครงสร้างของ URL ที่กำหนดดังในรูปที่ 4.1 ซึ่งมีรูปแบบที่คล้ายกับโครงสร้างไดเรคทอรี่ ซึ่งรูปที่ 4.2 เป็นการแสดงตัวอย่างการเข้าถึงเครื่องแม่ข่ายที่ชื่อ “TESTSERV” โดยเรียกบริการชื่อ “FREECONTENT” ที่ทำงานที่พอร์ตหมายเลข “88” เพื่ออ้างอิงถึงข้อมูลของหมายเลขโทรศัพท์ “66810987654”

HTTP : // IP : PORT / หมายเลขโทรศัพท์ / บริการ

รูปที่ 4.2 การกำหนดโครงสร้างของ URL ของเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST

HTTP : // TESTSERV : 88 / 66810987654 / FREECONTENT

รูปที่ 4.3 ตัวอย่างการกำหนด URL ตามโครงสร้าง URL ที่กำหนด

และสุดท้ายเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST ที่พัฒนาขึ้นมาเพื่อทดสอบนั้น ผู้วิจัยได้เลือกใช้ภาษามาตรฐาน XML ในการแสดงส่วนเนื้อหาของการร้องขอและการตอบกลับ โดยอ้างอิงจากเนื้อหาของการร้องขอและการตอบกลับจากเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP ให้ได้ค่าพารามิเตอร์ที่ครบถ้วนเช่นเดียวกัน เพื่อให้ได้ผลลัพธ์จากการร้องขอบริการของเว็บเซอร์วิสทั้งสองสถาปัตยกรรมที่เหมือนกัน ดังตัวอย่างของการร้องขอและการตอบกลับของเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP ในรูปที่ 4.3 และ รูปที่ 4.4 และตัวอย่างของการร้องขอและตอบกลับของเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST ที่แสดงในรูปที่ 4.5 และ รูปที่ 4.6

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:ns1="urn:xmethods-delayed-quotes"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
      <ns1:create>
         <msisdn xsi:type="xsd:string">66810987654</msisdn>
         <id xsi:type="xsd:string">3100001000990</id>
         <title xsi:type="xsd:string">mr</title>
         <firstname xsi:type="xsd:string">SOAP</firstname>
         <lastname xsi:type="xsd:string">TEST</lastname>
         <country xsi:type="xsd:string">Thailand</country>
         <postcode xsi:type="xsd:string">10220</postcode>
         <email xsi:type="xsd:string">66810987654@testsoap.com</email>
         <active_date xsi:type="xsd:datetime">2010-07-01T08:30:00</active_date>
      </ns1:create>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

รูปที่ 4.4 ตัวอย่างเนื้อหาการร้องขอบริการ CREATE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:ns1="urn:xmethods-delayed-quotes"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
      <ns1:createResponse>
         <result xsi:type="xsd:string">0</result>
      </ns1:createResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

รูปที่ 4.5 ตัวอย่างเนื้อหาการตอบกลับบริการ CREATE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP

```
<?xml version="1.0" encoding="UTF-8"?>
<subscriber>
    <msisdn>66810987654</msisdn>
    <id>3100001000990</id>
    <title>mr</title>
    <firstname>REST</firstname>
    <lastname>TEST</lastname>
    <country> Thailand </country>
    <postcode>10220</postcode>
    <email>66810987654@testrest.com</email>
    <active_date>2010/07/01 08:30:00</active_date>
</subscriber>
```

รูปที่ 4.6 ตัวอย่างเนื้อหาการร้องขอบริการ CREATE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST

```
<?xml version="1.0" encoding="UTF-8"?>
<subscriber>
    <result>0</result>
</subscriber>
```

รูปที่ 4.7 ตัวอย่างเนื้อหาการตอบกลับบริการ CREATE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST

4.5 วิธีการทดสอบ

ผู้วิจัยได้ทำการทดสอบโดยใช้ซอฟต์แวร์ Apache JMeter เพื่อสร้างการร้องขอบริการไปยังแม่ข่ายเว็บเซอร์วิสทั้ง 2 สถาปัตยกรรม แล้วเก็บผลลัพธ์ที่แม่ข่ายเว็บเซอร์วิสตอบกลับมาไว้ในเครื่องลูกข่าย โดยทำการแยกทดสอบทีละสถาปัตยกรรม และทดสอบรอบละหนึ่งบริการเท่านั้นด้วยการร้องขอบริการจำนวน 25,000 ครั้งต่อการทดสอบ 1 รอบ และทำการทดสอบซ้ำทั้งหมด 3 รอบสำหรับหนึ่งบริการของเว็บเซอร์วิสหนึ่งสถาปัตยกรรม

ขนาดของข้อมูลที่ใช้ในการทดสอบในบริการ CREATE และ บริการ UPDATE กำหนดให้เป็นข้อมูลที่มีขนาดเท่ากัน โดยเป็นข้อมูลที่สุ่มขึ้นให้แตกต่างกัน แต่จะมีความยาวของชุดอักขระที่เท่ากัน ทั้งนี้เพื่อหลีกเลี่ยงผลการทดสอบที่คลาดเคลื่อนเนื่องจากการสุ่มข้อมูล

รูปแบบของการร้องขอทั้งหมดนั้นจะใช้การเชื่อมต่อ HTTP แบบทรานแซคชัน โดยไม่มีการเก็บแคชของ HTTP และมีส่วนหัวของการเชื่อมต่อ HTTP (HTTP header) ขนาด 160ไบต์ โดยประมาณใกล้เคียงกันทั้งสำหรับการร้องขอไปยังแม่ข่าย เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP และแม่ข่ายเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST ซึ่งส่วนหัวของการเชื่อมต่อ HTTP นี้จะไม่ถูกนำมาใช้ในการวิเคราะห์ผลการทดสอบ โดยการทดสอบจะพิจารณาเฉพาะขนาดของเนื้อหาที่อยู่ภายในเท่านั้น

4.6 ผลการทดสอบ

4.6.1 ผลการทดสอบเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP

ถ้าพิจารณาจากเรื่องขนาดของเนื้อหา (Content-Length) ที่ส่งการร้องขอบริการออกไปนั้นสามารถแบ่งได้เป็นสองกลุ่มคือกลุ่มของบริการ CREATE และบริการ UPDATE ซึ่งสองบริการนี้จะมีเนื้อหาที่ส่งไปขนาดประมาณ 950 ไบต์เนื่องจากต้องส่งข้อมูลสมาชิกไปด้วย เพื่อทำการสร้างหรือแก้ไข โดยที่การตอบกลับจากแม่ข่ายเว็บเซอร์วิสจะส่งผลลัพธ์ของการสร้างหรือแก้ไขกลับมาด้วยเนื้อหาขนาดประมาณ 515 ไบต์ ส่วนบริการ RETRIEVE และบริการ DELETE จะเป็นบริการที่มีการส่งการร้องขอที่มีเนื้อหาในขนาดที่เล็กกว่าเพราะมีการส่งเพียงแค่คีย์หลักเท่านั้น ส่วนการตอบกลับจากแม่ข่ายเว็บเซอร์วิสนั้นถ้าเป็นบริการ DELETE จะส่งผลลัพธ์ของการลบกลับมาด้วยเนื้อหาขนาดประมาณ 515

ไบต์ และบริการ RETRIEVE จะส่งข้อมูลสมาชิกกลับมาเป็นผลลัพธ์จึงทำให้มีขนาดเนื้อหาตอบกลับที่ค่อนข้างใหญ่ประมาณ 1,460 ไบต์

เวลาที่ใช้ในการทำรายการ (Latency) แต่ละรายการเฉลี่ยโดยรวมจากการทดสอบทั้งสามรอบนั้นพบว่า เวลาในการตอบสนองบริการแต่ละครั้งนั้นไม่แตกต่างกันมากนัก โดยจะใช้เวลาทั้งสิ้น 21-24 มิลลิวินาที ส่วนปริมาณงานที่ทำในช่วงเวลาหนึ่งวินาที (Transactions per second) นั้นอยู่ที่ประมาณ 40-45 ทรานแซคชันต่อวินาที ดังจะแสดงค่าเฉลี่ยจากการทดสอบทั้งหมดได้ดังตารางที่ 4.2 ตารางที่ 4.3 และตารางที่ 4.4

ตารางที่ 4.2 ผลการทดสอบ เว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP รอบที่ 1

SOAP	ขนาดเนื้อหา		การทดสอบรอบที่ 1	
	การร้องขอ (ไบต์)	การตอบกลับ (ไบต์)	เวลาเฉลี่ย (มิลลิวินาที)	ปริมาณงานที่ทำได้ (จำนวนรายการต่อวินาที)
CREATE	953	515	24.76	40.39
RETRIEVE	536	1460	24.03	41.61
UPDATE	951	515	22.8	43.85
DELETE	532	515	22.63	44.19

ตารางที่ 4.3 ผลการทดสอบ เว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP รอบที่ 2

SOAP	ขนาดเนื้อหา		การทดสอบรอบที่ 2	
	การร้องขอ (ไบต์)	การตอบกลับ (ไบต์)	เวลาเฉลี่ย (มิลลิวินาที)	ปริมาณงานที่ทำได้ (จำนวนรายการต่อวินาที)
CREATE	953	515	24.84	40.25
RETRIEVE	536	1460	23.62	42.34
UPDATE	951	515	24.03	41.61
DELETE	532	515	21.78	45.91

ตารางที่ 4.4 ผลการทดสอบ เว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP รอบที่ 3

SOAP	ขนาดเนื้อหา		การทดสอบรอบที่ 3	
	การร้องขอ (ไบต์)	การตอบกลับ (ไบต์)	เวลาเฉลี่ย (มิลลิวินาที)	ปริมาณงานที่ทำได้ (จำนวนรายการต่อวินาที)
CREATE	953	515	23.69	42.2
RETRIEVE	536	1460	23.92	41.81
UPDATE	951	515	23.65	42.28
DELETE	532	515	22.86	43.74

4.6.2 ผลการทดสอบเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST

จากผลการทดสอบของเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST นั้นแสดงให้เห็นได้ชัดเจนว่าขนาดของเนื้อหา (Content-Length) ที่ส่งการร้องขอและที่แม่ข่ายเว็บเซอร์วิสตอบกลับมาจะมีขนาดที่เล็กกว่าของ เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP อย่างเห็นได้ชัดโดยเฉพาะในกรณีการร้องขอบริการ RETRIEVE และบริการ DELETE นั้นไม่ต้องส่งเนื้อหาใดๆ ไปด้วย เนื่องจากทั้งสองบริการนี้ใช้เพียงการระบุค่าของคีย์หลักซึ่งในที่นี้คือหมายเลขโทรศัพท์ไว้ใน URL เท่านั้น ส่วนขนาดเนื้อหาที่ตอบกลับจากแม่ข่ายเว็บเซอร์วิสในกรณีการใช้บริการ DELETE จะเป็นรหัสแสดงผลลัพธ์ของการลบขนาดประมาณ 67 ไบต์ เนื้อหาของการตอบรับจากบริการ RETRIEVE นั้นจะมีขนาดใหญ่ที่สุดเมื่อเทียบจากทั้ง 4 บริการ เนื่องจากเป็นการตอบกลับข้อมูลสมาชิกด้วยข้อความขนาดประมาณ 316 ไบต์ และกรณีบริการ CREATE และบริการ UPDATE จะมีการส่งการร้องขอที่มีเนื้อหาที่มีข้อมูลสมาชิกขนาดประมาณ 319 ไบต์ และแม่ข่ายเว็บเซอร์วิสจะตอบกลับด้วยผลลัพธ์ของการสร้างและแก้ไขกลับมาด้วยเนื้อหาขนาดประมาณ 67 ไบต์

เวลาที่ใช้ในการทำรายการ (Latency) แต่ละรายการเฉลี่ยโดยรวมจากการทดสอบทั้งสามรอบนั้นพบว่าเวลาในการตอบสนองบริการแต่ละครั้งนั้นใช้เวลาอยู่ในช่วง 15-17 มิลลิวินาที และค่าปริมาณงานที่ทำในช่วงเวลาหนึ่งวินาที (Transactions per second) นั้นอยู่ที่ประมาณ 56-62 ทรานแซคชันต่อ

วินาที โดยบริการ RETRIEVE จะมีค่าปริมาณงานที่ทำในช่วงเวลาหนึ่งวินาทีน้อยที่สุด ดังจะเห็นได้จากตารางที่ 4.5 ตารางที่ 4.6 และ ตารางที่ 4.7

ตารางที่ 4.5 ผลการทดสอบเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST รอบที่ 1

REST	ขนาดเนื้อหา		การทดสอบรอบที่ 1	
	การร้องขอ (ไบต์)	การตอบกลับ (ไบต์)	เวลาเฉลี่ย (มิลลิวินาที)	ปริมาณงานที่ทำได้ (จำนวนรายการต่อวินาที)
CREATE	319	67	16.64	60.1
RETRIEVE	0	316	17.01	58.81
UPDATE	317	67	16.03	62.37
DELETE	0	67	15.84	63.14

ตารางที่ 4.6 ผลการทดสอบเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST รอบที่ 2

REST	ขนาดเนื้อหา		การทดสอบรอบที่ 2	
	การร้องขอ (ไบต์)	การตอบกลับ (ไบต์)	เวลาเฉลี่ย (มิลลิวินาที)	ปริมาณงานที่ทำได้ (จำนวนรายการต่อวินาที)
CREATE	319	67	15.94	62.72
RETRIEVE	0	316	17.73	56.4
UPDATE	317	67	16.9	62.16
DELETE	0	67	15.88	62.95

ตารางที่ 4.7 ผลการทดสอบเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST รอบที่ 3

REST	ขนาดเนื้อหา		การทดสอบรอบที่ 3	
	การร้องขอ (ไบต์)	การตอบกลับ (ไบต์)	เวลาเฉลี่ย (มิลลิวินาที)	ปริมาณงานที่ทำได้ (จำนวนรายการต่อวินาที)
CREATE	319	67	16.07	62.22
RETRIEVE	0	316	17.32	57.71
UPDATE	317	67	16.04	62.35
DELETE	0	67	16.02	62.4

4.6.3 ผลการทดสอบเว็บเซอร์วิสทั้งสองสถาปัตยกรรมโดยการเพิ่มจำนวนการร้องขอ

ผลการทดสอบในส่วนนี้เป็นการทดสอบเพิ่มเติมโดยการเพิ่มจำนวนการร้องขอบริการจากจำนวน 25,000 ครั้งต่อการทดสอบ 1 รอบเป็น 50,000 ครั้ง และ 75,000 ครั้งต่อการทดสอบ 1 รอบ ทั้งนี้เพื่อเป็นการวิเคราะห์แนวโน้มของการใช้งานเว็บเซอร์วิสที่มีการร้องขอสูงมากขึ้น โดยผลการทดสอบแสดงจะเห็นว่าเมื่อมีจำนวนการร้องขอเพิ่มมากขึ้นเวลาที่ใช้ในการทำรายเพิ่มสูงขึ้น และค่าปริมาณงานที่ทำในช่วงเวลาหนึ่งวินาทีนั้นลดลงทั้งนี้ด้วยข้อจำกัดเกี่ยวกับประสิทธิภาพของเครื่องแม่ข่ายเว็บเซอร์วิส เมื่อพิจารณาความแตกต่างเมื่อเปรียบเทียบเว็บเซอร์วิสทั้งสองสถาปัตยกรรมนั้นพบว่าเวลาในการตอบสนองบริการ และค่าปริมาณงานที่ได้จากเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST ยังคงให้ผลการทดสอบที่แสดงให้เห็นถึงประสิทธิภาพที่ดีเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP ดังแสดงให้เห็นในตารางที่ 4.8 ตารางที่ 4.9 และตารางที่ 4.10 ตามลำดับ

ตารางที่ 4.8 ผลการทดสอบเปรียบเทียบเว็บเซอร์วิสที่จำนวน 25,000 ทรานแซคชัน

25,000 ทรานแซคชัน	SOAP		REST	
	เวลาเฉลี่ย (มิลลิวินาที)	ปริมาณงานที่ทำได้ (ทรานแซคชันต่อวินาที)	เวลาเฉลี่ย (มิลลิวินาที)	ปริมาณงานที่ทำได้ (ทรานแซคชันต่อวินาที)
CREATE	24.43	40.95	16.22	61.68
RETRIEVE	23.86	41.92	17.35	57.64
UPDATE	23.49	42.58	16.32	62.29
DELETE	22.42	44.61	15.91	62.83

ตารางที่ 4.9 ผลการทดสอบเปรียบเทียบเว็บเซอร์วิสที่จำนวน 50,000 ทรานแซคชัน

50,000 ทรานแซคชัน	SOAP		REST	
	เวลาเฉลี่ย (มิลลิวินาที)	ปริมาณงานที่ทำได้ (ทรานแซคชันต่อวินาที)	เวลาเฉลี่ย (มิลลิวินาที)	ปริมาณงานที่ทำได้ (ทรานแซคชันต่อวินาที)
CREATE	28.21	35.45	18.07	55.35
RETRIEVE	29.04	34.43	19.90	50.25
UPDATE	28.65	34.91	19.29	51.84
DELETE	27.43	36.46	18.35	54.48

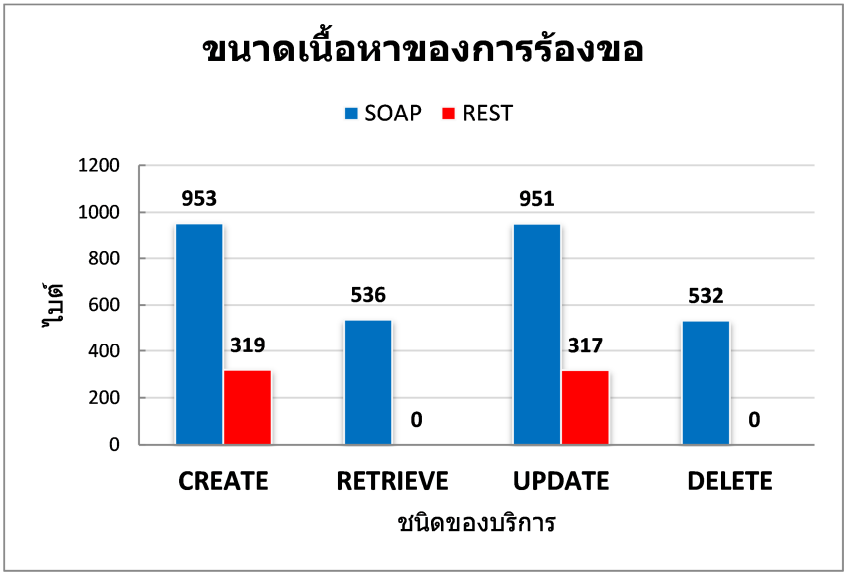
ตารางที่ 4.10 ผลการทดสอบเปรียบเทียบเว็บเซอร์วิสที่จำนวน 75,000 ทรานแซคชัน

75,000 ทรานแซคชัน	SOAP		REST	
	เวลาเฉลี่ย (มิลลิวินาที)	ปริมาณงานที่ทำได้ (ทรานแซคชันต่อวินาที)	เวลาเฉลี่ย (มิลลิวินาที)	ปริมาณงานที่ทำได้ (ทรานแซคชันต่อวินาที)
CREATE	33.24	30.09	21.19	47.18
RETRIEVE	33.14	30.17	22.62	44.21
UPDATE	32.14	31.11	21.81	45.86
DELETE	33.06	30.25	21.56	46.38

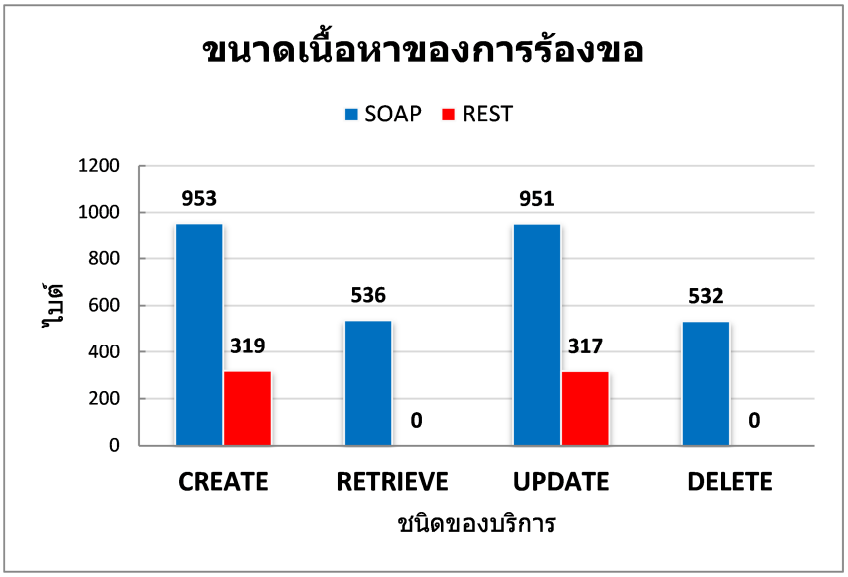
4.7 การอภิปรายผลการทดสอบ

จากผลการทดสอบสามารถทำการเปรียบเทียบขนาดของเนื้อหาในการร้องขอบริการจากเว็บเซอร์วิสและขนาดของเนื้อหาในการตอบกลับจากแม่ข่ายเว็บเซอร์วิสตามรูปที่ 4.7 และ รูปที่ 4.8 ซึ่งเห็นได้ว่าในการร้องขอบริการ CREATE และบริการ UPDATE จากแม่ข่ายเว็บเซอร์วิสนั้นสามารถลดขนาดของข้อมูลที่ต้องส่งไปได้ประมาณ 66% และการร้องขอบริการ RETRIEVE และบริการ DELETE นั้นสามารถลดขนาดของข้อมูลที่ต้องส่งไปได้ทั้งหมด 100% เพราะด้วยรูปแบบของสถาปัตยกรรมเว็บเซอร์วิสแบบ REST นั้นมีการอ้างอิงคีย์หลักอยู่ใน URL แล้ว จึงทำให้รูปแบบของบริการดังกล่าวไม่ต้องส่งเนื้อหาข้อมูลไปด้วย

และเนื้อหาที่แม่ข่ายเว็บเซอร์วิสตอบกลับนั้นในกรณีที่เป็นการร้องขอข้อมูลด้วยบริการ RETRIEVE นั้นสามารถลดขนาดของเนื้อหาได้ ประมาณ 78% และกรณีที่เป็นการร้องขออื่นๆ ที่มีการตอบกลับด้วยรหัสแสดงผลลัพธ์ของการทำรายการต่างๆอย่าง CREATE, UPDATE หรือ DELETE นั้น ถ้าใช้เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST จะสามารถลดขนาดของเนื้อหาตอบกลับได้ประมาณ 87%



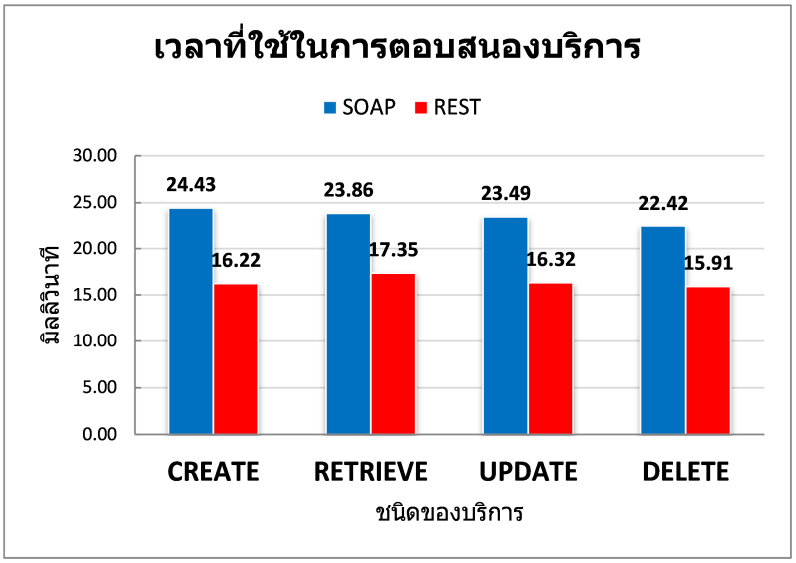
รูปที่ 4.8 เปรียบเทียบขนาดของเนื้อหาในการร้องขอ



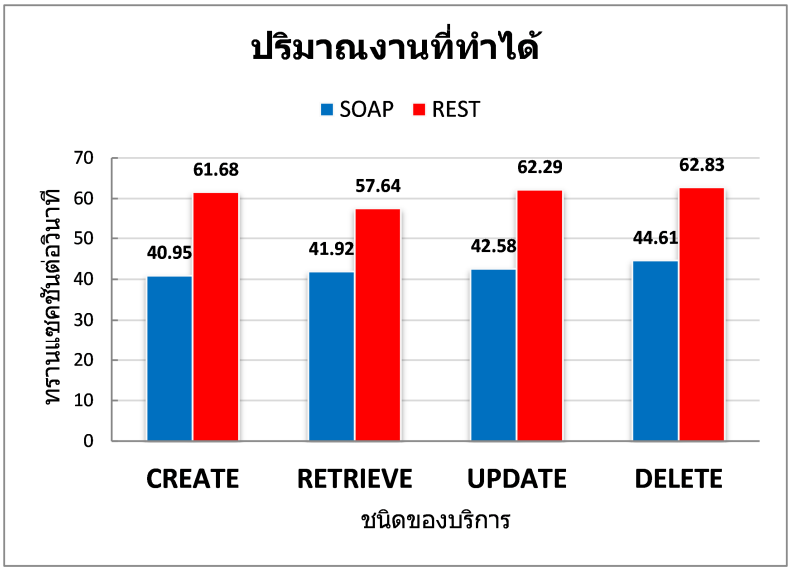
รูปที่ 4.9 เปรียบเทียบขนาดของเนื้อหาในการตอบกลับ

ในแง่ของเวลาตอบสนองในการใช้บริการ ซึ่งเป็นเวลารวมทั้งหมดที่ใช้ในการส่งการร้องขอบริการไปยังแม่ข่ายเว็บเซอร์วิส เวลาในการประมวลผลของแม่ข่ายเว็บเซอร์วิส และเวลาในการส่งข้อมูลตอบกลับมายังเครื่องลูกข่าย จากข้อมูลเปรียบเทียบเวลาตอบสนองในการใช้บริการในรูปที่ 4.9 จะเห็นว่าระยะเวลาที่ใช้ในการร้องขอบริการจากเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST จะน้อยกว่าเวลาตอบสนองของการร้องขอบริการจากเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP ประมาณ 30% โดยเฉลี่ยต่อหนึ่งทรานแซคชัน

เมื่อเวลาตอบสนองในการร้องขอใช้บริการต่อหนึ่งทรานแซคชันลดลงแล้วนั้นย่อมส่งผลถึงปริมาณงานที่สามารถทำได้ในช่วงเวลาหนึ่งๆ ย่อมเพิ่มขึ้น ดังแสดงให้เห็นได้จากรูปที่ 4.10 ซึ่งจะเห็นว่าเมื่อร้องขอใช้บริการเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST นั้นมีค่าปริมาณงานที่ทำได้ในช่วงเวลาหนึ่งวินาทีเพิ่มขึ้นประมาณ 30% เมื่อเทียบกับเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP



รูปที่ 4.10 เปรียบเทียบเวลาที่ใช้ในการทำรายการ

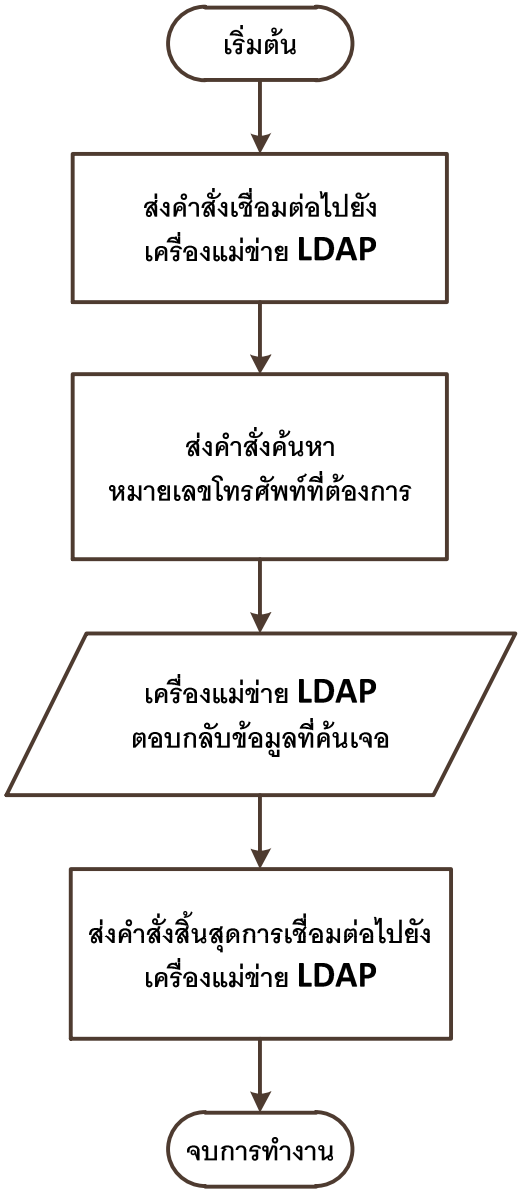


รูปที่ 4.11 เปรียบเทียบค่าปริมาณงานที่ทำได้ในช่วงวินาที

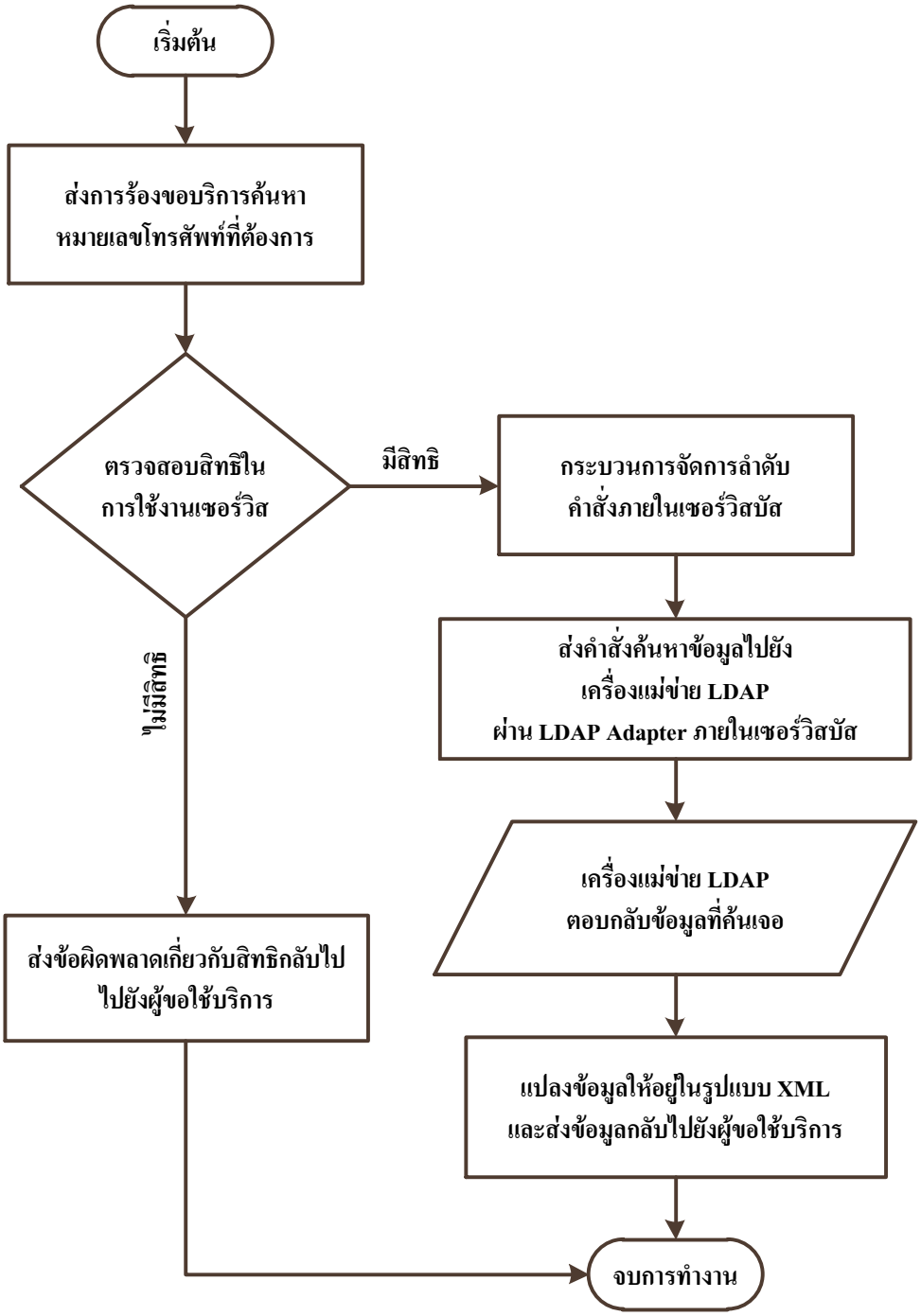
4.8 การทดสอบเซอร์วิสบัส

สำหรับการทดสอบอีกส่วนหนึ่งคือการทดสอบเพื่อแสดงให้เห็นถึงการจำลองการใช้งานแนวความคิดเกี่ยวกับเซอร์วิสบัส ด้วยการร้องขอไปที่ระบบเดิมเมื่อเปรียบเทียบกับการร้องขอแบบเว็บเซอร์วิสผ่านทางเซอร์วิสบัสไปยังระบบเดิม โดยเป็นการจำลองการร้องขอไปยังเครื่องแม่ข่าย LDAP

(Lightweight Directory Access Protocol) โดยกระบวนการทำงานของการร้องขอการค้นหาข้อมูลบนเครื่องแม่ข่าย LDAP ด้วยคำสั่ง LDAP ดังแสดงให้เห็นในรูป 4.11 และ กระบวนการทำงานของการร้องขอการค้นหาด้วยเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST ผ่านเซอร์วิสบัสในรูปที่ 4.12



รูปที่ 4.12 กระบวนการทำงานของการค้นหาข้อมูลบนเครื่องแม่ข่าย LDAP ด้วยคำสั่ง LDAP



รูปที่ 4.13 กระบวนการทำงานของการร้องขอการค้นหาด้วยเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST ผ่านเซอร์วิสบัส

ตารางที่ 4.11 ผลการทดสอบเซอร์วิสบัสที่เชื่อมต่อกับเครื่องแม่ข่าย LDAP

LDAP	การทดสอบ 25000 ทรานแซคชัน	
	เวลาเฉลี่ย (มิลลิวินาที)	ปริมาณงานที่ทำได้ (ทรานแซคชันต่อวินาที)
LDAP	**22.52**	**44.41**
SERVICE BUS	**40.90**	**24.45**

จากการทดสอบพบผลการทดสอบเซอร์วิสบัสที่เชื่อมต่อกับเครื่องแม่ข่าย LDAP นั้นพบว่าใช้เวลาในการตอบสนองการให้บริการมากกว่าการเรียกใช้คำสั่ง LDAP ผ่านเครื่องแม่ข่าย LDAP เอง ซึ่งทำให้ปริมาณงานที่ทำได้ในช่วงหนึ่งวินาทีลดลงด้วยเช่นกัน แต่ทั้งนี้การทดสอบเป็นเพียงการจำลองการค้นหาข้อมูลบนเครื่องแม่ข่าย LDAP ที่มีปริมาณข้อมูลน้อย โดยในการกรณีตัวอย่าง ถ้าหากอ้างอิงจากการทดลองด้วยเครื่องแม่ข่าย LDAP ที่ใช้งานจริงที่มีปริมาณข้อมูลลูกค้าจำนวน 20 ล้านรายการ ระยะเวลาที่ทำการค้นหาโดยเฉลี่ยจะมีค่าประมาณ 100 มิลลิวินาทีต่อทรานแซคชัน ดังนั้นหากทำการทดเพิ่มเวลาที่ใช้ในการค้นหาข้อมูลจากฐานข้อมูลที่มีขนาดใหญ่มากขึ้น ก็จะเห็นได้ว่าค่าของเวลาเฉลี่ยของการตอบสนองต่อทรานแซคชันที่ใช้จะไม่แตกต่างกันมากนัก

บทที่ 5

สรุปผลการวิจัยและข้อเสนอแนะ

สำหรับงานวิจัยฉบับนี้ ต้องการพิสูจน์ให้เห็นถึงข้อดีของการออกแบบพัฒนาระบบตามแนวทางของสถาปัตยกรรมเชิงบริการ โดยอ้างอิงด้วยการพัฒนาระบบการให้บริการเสริมแก่ลูกค้าที่ใช้งานโทรศัพท์เคลื่อนที่ภายใต้แนวทางของสถาปัตยกรรมเชิงบริการ (Service-Oriented Architecture, SOA) โดยจะเห็นได้ว่า เมื่อมีการปรับปรุงสถาปัตยกรรมเดิมที่ใช้อยู่แล้วให้กลายเป็นสถาปัตยกรรมเชิงบริการแล้วนั้น เมื่อมีบริการใหม่ๆที่ต้องการให้บริการแก่ลูกค้านั้นทำให้เราสามารถนำเซอร์วิสที่มีอยู่แล้วกลับมาใช้งานได้ง่ายทำให้ลดค่าใช้จ่าย และระยะเวลาในการพัฒนาที่จะเกิดขึ้นในการพัฒนาบริการใหม่ๆได้

หลังจากมีการปรับปรุงพัฒนาสถาปัตยกรรมที่มีอยู่ในเป็นไปตามแนวทางของสถาปัตยกรรมเชิงบริการแล้วนั้น จะเห็นได้ว่าส่วนหัวใจสำคัญของสถาปัตยกรรมเชิงบริการคือการออกแบบและพัฒนาเซอร์วิสในลักษณะของเว็บเซอร์วิส ซึ่งช่วยทำให้สามารถพัฒนาเซอร์วิสให้สอดคล้องกับแนวทางดังกล่าว โดยงานวิจัยนี้ได้มีการกำหนดมาตรฐานและรูปแบบของการพัฒนาเว็บเซอร์วิสภายใต้สถาปัตยกรรม REST ซึ่งเป็นการชี้ให้เห็นถึงความสำคัญในการเลือกใช้เทคโนโลยีเว็บเซอร์วิสให้เหมาะสมกับความต้องการและลักษณะการใช้งานเพื่อให้เกิดประโยชน์สูงสุดบนทรัพยากรที่มีอยู่อย่างจำกัด ในส่วนแรกของงานวิจัยได้กล่าวถึงความรู้เบื้องเกี่ยวกับเว็บเซอร์วิส, SOAP และแนวความคิดเกี่ยวกับการนำทฤษฎีสถาปัตยกรรมแบบ REST มาพัฒนาเป็นเว็บเซอร์วิส ในส่วนที่สองนั้นได้กล่าวถึงวิธีการเรียกใช้บริการ และความแตกต่างของการเรียกใช้บริการ เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP กับ เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST และในส่วนสุดท้ายจะเป็นการทดสอบเปรียบเทียบให้เห็นถึงความแตกต่างของการเรียกใช้บริการ เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP กับ เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST ในการให้บริการแบบเดียวกันในเชิงของขนาดเนื้อหาที่ใช้ในการร้องขอบริการและการตอบรับ รวมทั้งเวลาในการตอบสนองทั้งหมดของการร้องขอบริการนั้นๆเพื่อเป็นแนวทางในการตัดสินใจเลือกใช้รูปแบบเทคโนโลยีเว็บเซอร์วิสได้อย่างเหมาะสม

จากข้อสรุปผลการทดลองจะเห็นได้ว่าการเรียกใช้งาน เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST นั้นสามารถลดปริมาณข้อมูลการร้องขอบริการและการตอบกลับระหว่างเครื่องลูกข่ายและเครื่องแม่ข่ายได้ 80% ในกระบวนการสร้างหรือแก้ไขและลดปริมาณการรับส่งข้อมูลได้มากถึง 90% ในกรณีที่เป็นการเรียกดูข้อมูลหรือลบข้อมูลในตำแหน่งทรัพยากรที่ต้องการ ทั้งยังช่วยทำให้เครื่องแม่ข่ายสามารถรองรับการให้บริการเพิ่มมากขึ้นได้โดยที่การร้องขอบริการจากเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST นั้นจะช่วยลดระยะเวลาในการตอบสนองการขอบริการ และเพิ่มปริมาณงานที่ทำได้ของแม่ข่ายเว็บเซอร์วิสได้ถึงประมาณ 30% และหากผู้ให้บริการเว็บเซอร์วิสใช้การพัฒนา เว็บเซอร์วิสภายใต้สถาปัตยกรรมแบบ REST ก็จะทำให้ผู้ขอใช้บริการนั้นได้รับการตอบสนองที่เร็วขึ้น และยังเป็นการช่วยทำให้มีการใช้งานแบนด์วิธอย่างคุ้มค่าอีกด้วย

ทั้งนี้เนื่องจากการพัฒนาเว็บเซอร์วิสที่ใช้สถาปัตกรรม REST อาจจะมีความยุ่งยากมากกว่าการพัฒนาเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP แต่ด้วยข้อดีสถาปัตยกรรม REST ดังที่เห็นในการทดสอบนั้นจึงควรพิจารณาถึงผู้ใช้บริการที่เป็นกลุ่มเป้าหมายด้วย เพราะว่าผู้ใช้บริการเป็นจำนวนมากยังมีการใช้งานอินเทอร์เน็ตบนเครือข่ายอินเทอร์เน็ตที่มีแบนด์วิธไม่สูงมากนัก อย่างเช่นการใช้งาน GPRS/EDGE บนโทรศัพท์มือถือ หากเลือกพัฒนาเว็บเซอร์วิสโดนใช้สถาปัตยกรรม REST นั้นก็จะทำให้ผู้ขอใช้บริการนั้นใช้งานได้เร็วขึ้น แถมยังเป็นการช่วยทำให้มีการใช้งานแบนด์วิดท์อย่างคุ้มค่าอีกด้วย

เอกสารอ้างอิง

[1] Wikipedia, "**Mobile network operator**", Published on the Internet, http://en.wikipedia.org/wiki/Mobile_network_operator

[2] Wikipedia, "**Value Added Service**", Published on the Internet, http://en.wikipedia.org/wiki/Value_added_service

[3] Thomas Erl, "**Service-Oriented Architecture Concepts Technology and Design**", Prentice Hall, 2005

[4] Eric Newcomer and Greg Lomow, "**Understanding SOA with Web Services**", Addison-Wesley, 2005

[5] Roy Thomas Fielding, "**Architectural Styles and the Design of Network-based Software Architectures**", Chapter 5, PhD thesis, University of California, Irvine, 2000

[6] Alex Rodriguez, "**RESTful Web Services: The basics**", IBM, November 2008

[7] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L.Masinter, P.Leach and T.Berners-Lee, "**Hypertext transfer protocol 1.1**", Internet, June 1999

[8] Wikipedia, "**Mobile Virtual Network Operator**", Published on the Internet, http://en.wikipedia.org/wiki/Mobile_Virtual_Network_Operator

[9] Wikipedia, "**Global System for Mobile Communications**", Published on the Internet, http://en.wikipedia.org/wiki/Global_System_for_Mobile_Communications

[10] Wikipedia, "**GSM Frequency Ranges**", Published on the Internet, http://en.wikipedia.org/wiki/GSM_frequency_ranges

[11] Wikipedia, "**Short Message Service**", Published on the Internet, http://en.wikipedia.org/wiki/Short_Message_Service

[12] Wikipedia, "**Multimedia Messaging Service**", Published on the Internet, http://en.wikipedia.org/wiki/Multimedia_Messaging_Service

[13] Wikipedia, "**General Packet Radio Service**", Published on the Internet, http://en.wikipedia.org/wiki/General_Packet_Radio_Service

[14] Wikipedia, "**Enhanced Data Rates for GSM Evolution**", Published on the Internet, http://en.wikipedia.org/wiki/Enhanced_data_rates_for_gsm_evolution

[15] J. Schiller, "**Mobile Communications**", Addison-Wesley, 2000

[16] Thanachart Numnonda and Thanisa Kruawaisayawan, "**SOA: สถาปัตยกรรมเชิงบริการ**"

[17] Nilo Mitra, Yves Lafon, "**SOAP Version 1.2 Part 0: Primer (Second Edition)**", Published on the Internet, W3C Recommendation, April 2007

[18] Roberto Chinnici, Jean-Jacques Moreau, Arthur Ryman, Sanjiva Weerawarana, "**Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language**", Published on the Internet, W3C Candidate Recommendation, March 2006

[19] Roberto Chinnici, Hugo Haas, Amelia A. Lewis, Jean-Jacques Moreau, David Orchard, Sanjiva Weerawarana. "**Web Services Description Language (WSDL) Version 2.0 Part 2: Adjuncts**", Published on the Internet, W3C Candidate Recommendation, June 2007

[20] Lawrence Mandel, **"Describe REST Web services with WSDL 2.0"**, Published on the Internet, Rational Software Developer, IBM 2008

[21] Wikipedia, "**Web Service**", Published on the Internet, http://en.wikipedia.org/wiki/Web_service

[22] Martin Gudgin, Marc Hadley and Tony Rogers, "**Web Services Addressing 1.0 – Core**", Published on the Internet, W3C Recommendation, May 2006

[23] D. Smith, J. Correia, B. Pring, D. Plummer., "**Gartner's Internet Strategies Research Note M-13-3593**", 9 April 2001

[24] T. Berners-Lee, L.Masinter and M. McCahill, "**RFC 1738 : Uniform Resource Locators (URL)**", Published: www.rfc-editor.org, December 1994

[25] Wikipedia, "**Universal Description Discovery and Integration**", Published on the Internet, http://en.wikipedia.org/wiki/UDDI

[26] W3C, "**Extensible Markup Language (XML)**", Published on the Internet, http://www.w3.org/XML/

[27] Wikipedia, "**Extensible Markup Language (XML)**", Published on the Internet, http://en.wikipedia.org/wiki/Xml

[28] Wikipedia, "**Simple Object Access Protocol (SOAP)**", Published on the Internet, http://en.wikipedia.org/wiki/Simple_Object_Access_Protocol

[29] Vassiliki Koutsonikola, Athena Vakali, "**LDAP: Framework, Practices, and Trends**", IEEE Internet Computing, vol. 8, no. 5, pp. 66-72, Sep./Oct. 2004

[30] D W Chadwick, "**Understanding X.500 - The Directory**", Published on the Internet, 1994

[31] Mark Wahl, "**RFC2256 : A Summary of the X.500(96) User Schema for use with LDAPv3**", IETF Standards Track, December 1997

[32] Wong-Bushby I, Egan R.,Isaacson C.. "**A Case Study in SOA and Re-Architecture at Company ABC**", Proceedings of the 39th Hawaii International Conference on System Sciences'06**,** 2006

[33] Enrique Castro-Leon, Jackson He, Mark Chang, "**Scaling Down SOA to Small Businesses**", IEEE International Conference on Service-Oriented Computing and Applications (SOCA'07), 2007

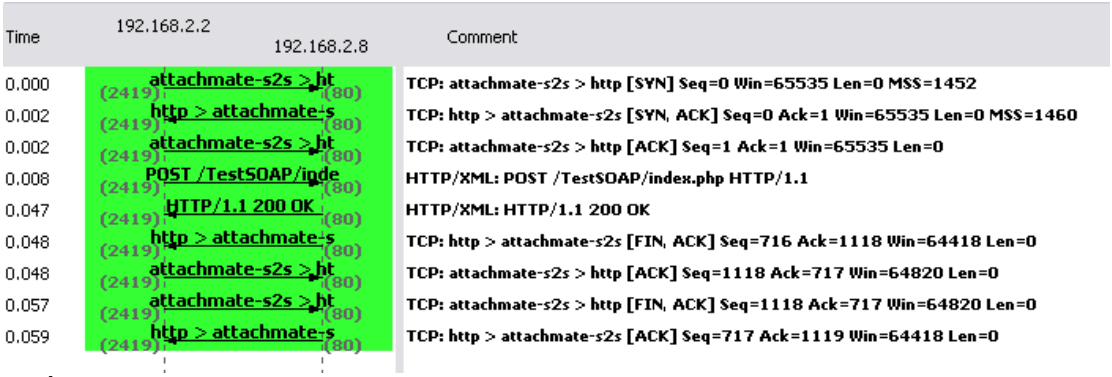
[34] Pacharawon Boonchoo, "**Consideration the performance of Web Service using REST architecture**", The 3rd National Conference on Information Technology (NCIT), 2010

[35] Naga, Sravanthi (119500) Performance Testing CoE, "**Open Source Performance Testing Using Apache JMeter**", Cognizant Technology Solutions, 2008

ภาคผนวก ก.

ตัวอย่างการร้องขอและตอบกลับในการทดสอบ

ตัวอย่างการเรียกใช้บริการ CREATE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP



รูปที่ ก.1 โฟลว์ของ TCP แพคเกจในการเรียกใช้บริการ CREATE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP

```
POST /TestSOAP/index.php HTTP/1.1
Content-Type: text/xml
Connection: close
User-Agent: Jakarta Commons-HttpClient/3.1
Host: 192.168.2.8
Content-Length: 953

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ns1="urn:xmethods-delayed-quotes"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
<ns1:create>
<msisdn xsi:type="xsd:string">66823125287</msisdn>
<id xsi:type="xsd:string">3000283125287</id>
<title xsi:type="xsd:string">mr</title>
<firstname xsi:type="xsd:string">SOAP</firstname>
<lastname xsi:type="xsd:string">Thread-1</lastname>
<country xsi:type="xsd:string">Thailand</country>
<postcode xsi:type="xsd:string">10220</postcode>
<email xsi:type="xsd:string">66823125287@testsoap.com</email>
<active_date xsi:type="xsd:datetime">2553-10-22T22:12:10</active_date>
</ns1:create>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

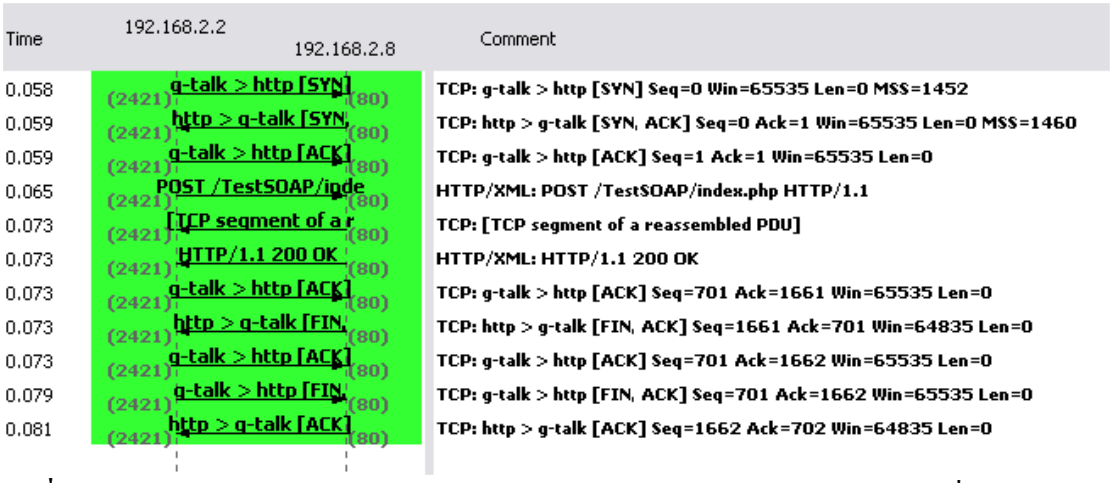
รูปที่ ก.2 การร้องขอในการเรียกใช้บริการ CREATE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP

```
HTTP/1.1 200 OK
Date: Fri, 22 Oct 2010 15:14:14 GMT
Server: Apache/2.2.8 (Win32) PHP/5.2.6
X-Powered-By: PHP/5.2.6
Content-Length: 515
Connection: close
Content-Type: text/xml; charset=utf-8

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="urn:xmethods-delayed-quotes" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
<ns1:createResponse>
<result xsi:type="xsd:string">0</result
></ns1:createResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

รูปที่ ก.3 การตอบกลับในการเรียกใช้บริการ CREATE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP

ตัวอย่างการเรียกใช้บริการ RETRIVE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP



รูปที่ ก.4 โฟลว์ของ TCP แพคเกจในการเรียกใช้บริการ RETRIVE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP

```
POST /TestSOAP/index.php HTTP/1.1
Content-Type: text/xml
Connection: close
User-Agent: Jakarta Commons-HttpClient/3.1
Host: 192.168.2.8
Content-Length: 536

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ns1="urn:xmethods-delayed-quotes"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
<ns1:retrieve><msisdn xsi:type="xsd:string">66810000237</msisdn></ns1:retrieve>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

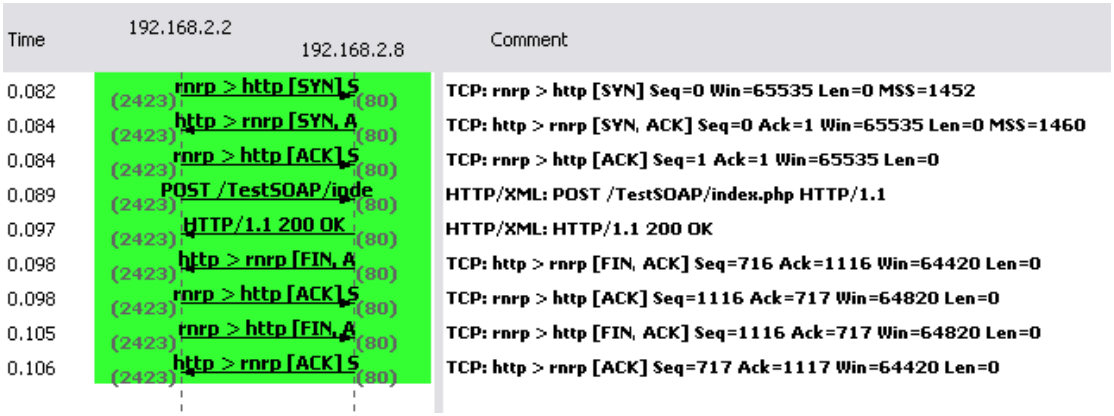
รูปที่ ก.5 การร้องขอในการเรียกใช้บริการ RETRIVE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP

```
HTTP/1.1 200 OK
Date: Fri, 22 Oct 2010 15:14:14 GMT
Server: Apache/2.2.8 (Win32) PHP/5.2.6
X-Powered-By: PHP/5.2.6
Content-Length: 1459
Connection: close
Content-Type: text/xml; charset=utf-8

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="urn:xmethods-delayed-quotes" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:ns2="http://xml.apache.org/xml-soap"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"><SOAP-
ENV:Body><ns1:retrieveResponse><result xsi:type="ns2:Map"><item><key
xsi:type="xsd:string">msisdn</key><value
xsi:type="xsd:string">66810000237</value></item><item><key xsi:type="xsd:string">id</key><value
xsi:type="xsd:string">3001763617653</value></item><item><key
xsi:type="xsd:string">title</key><value xsi:type="xsd:string">dr</value></item><item><key
xsi:type="xsd:string">firstname</key><value xsi:type="xsd:string">xSOAP</value></item><item><key
xsi:type="xsd:string">lastname</key><value xsi:type="xsd:string">xThread-
11</value></item><item><key xsi:type="xsd:string">country</key><value
xsi:type="xsd:string">SIAM</value></item><item><key xsi:type="xsd:string">postcode</key><value
xsi:type="xsd:string">12345</value></item><item><key xsi:type="xsd:string">email</key><value
xsi:type="xsd:string">66810000237@testsoap.com</value></item><item><key
xsi:type="xsd:string">active_date</key><value xsi:type="xsd:string">2553-10-18
08:42:03</value></item></result></ns1:retrieveResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```


รูปที่ ก.6 การตอบรับในการเรียกใช้บริการ RETRIVE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP

ตัวอย่างการเรียกใช้บริการ UPDATE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP



รูปที่ ก.7 โฟลว์ของ TCP แพคเกจในการเรียกใช้บริการ UPDATE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP

```
POST /TestSOAP/index.php HTTP/1.1
Content-Type: text/xml
Connection: close
User-Agent: Jakarta Commons-HttpClient/3.1
Host: 192.168.2.8
Content-Length: 951

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ns1="urn:xmethods-delayed-quotes"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
<ns1:update>
<msisdn xsi:type="xsd:string">66823125287</msisdn>
<id xsi:type="xsd:string">3000283125287</id>
<title xsi:type="xsd:string">dr</title>
<firstname xsi:type="xsd:string">xSOAP</firstname>
<lastname xsi:type="xsd:string">xThread-1</lastname>
<country xsi:type="xsd:string">SIAM</country>
<postcode xsi:type="xsd:string">12345</postcode>
<email xsi:type="xsd:string">66823125287@testsoap.com</email>
<active_date xsi:type="xsd:datetime">2553-10-22T22:12:10</active_date>
</ns1:update>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

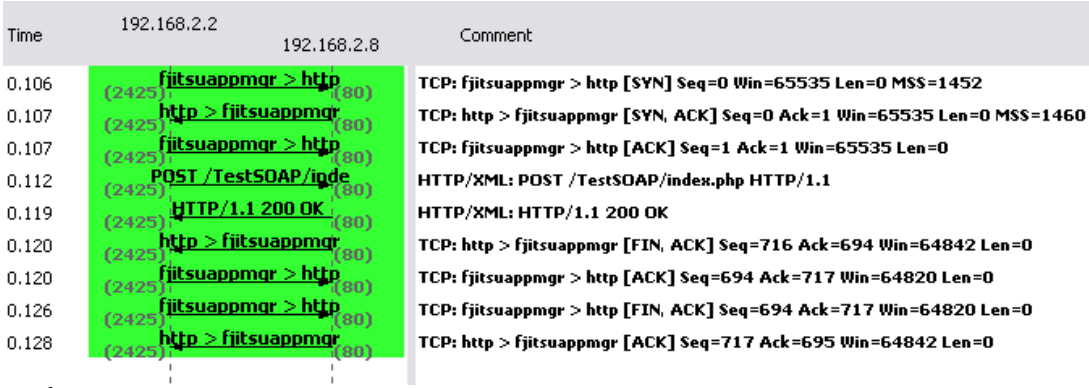
รูปที่ ก.8 การร้องขอในการเรียกใช้บริการ UPDATE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP

```
HTTP/1.1 200 OK
Date: Fri, 22 Oct 2010 15:14:14 GMT
Server: Apache/2.2.8 (Win32) PHP/5.2.6
X-Powered-By: PHP/5.2.6
Content-Length: 515
Connection: close
Content-Type: text/xml; charset=utf-8

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="urn:xmethods-delayed-quotes" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
<ns1:updateResponse><result xsi:type="xsd:string">0</result></ns1:updateResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

รูปที่ ก.9 การตอบรับในการเรียกใช้บริการ UPDATE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP

ตัวอย่างการเรียกใช้บริการ DELETE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP



รูปที่ ก.10 โฟลว์ของ TCP แพคเกจในการเรียกใช้บริการ DELETE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP

```
POST /TestSOAP/index.php HTTP/1.1
Content-Type: text/xml
Connection: close
User-Agent: Jakarta Commons-HttpClient/3.1
Host: 192.168.2.8
Content-Length: 529

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ns1="urn:xmethods-delayed-quotes"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
<ns1:delete>
<msisdn xsi:type="xsd:string">23125287</msisdn>
</ns1:delete>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

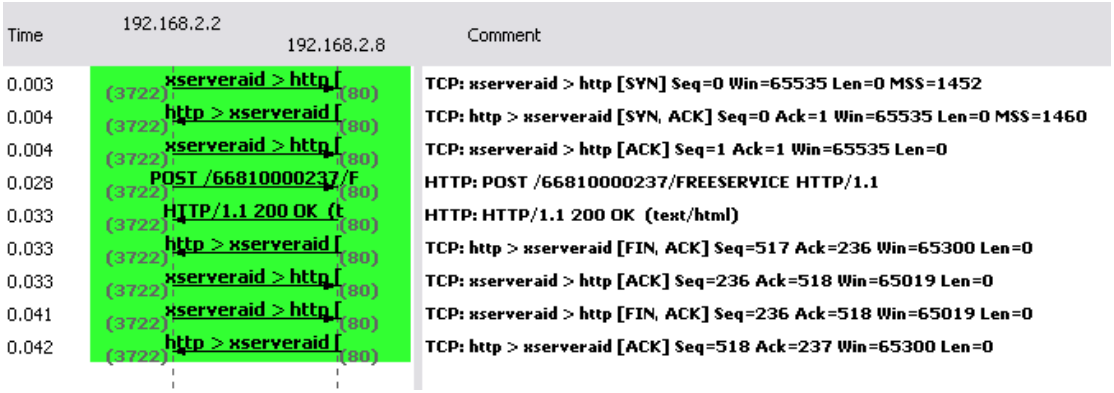
รูปที่ ก.11 การร้องขอในการเรียกใช้บริการ DELETE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP

```
HTTP/1.1 200 OK
Date: Fri, 22 Oct 2010 15:14:14 GMT
Server: Apache/2.2.8 (Win32) PHP/5.2.6
X-Powered-By: PHP/5.2.6
Content-Length: 515
Connection: close
Content-Type: text/xml; charset=utf-8

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="urn:xmethods-delayed-quotes" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
<ns1:deleteResponse><result xsi:type="xsd:string">0</result></ns1:deleteResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

รูปที่ ก.12 การตอบกลับในการเรียกใช้บริการ UPDATE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP

ตัวอย่างการเรียกใช้บริการ CREATE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST



รูปที่ ก.13 โฟลว์ของ TCP แพคเกจในการเรียกใช้บริการ CREATE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST

```
POST /66810000237/FREESERVICE HTTP/1.1
Content-Type: text/xml
Connection: close
User-Agent: Jakarta Commons-HttpClient/3.1
Host: 192.168.2.8
Content-Length: 319

<?xml version="1.0"?>
<subscriber>
<msisdn>66810000237</msisdn>
<id>3001242519358</id>
<title>mr</title>
<firstname>REST</firstname>
<lastname>Thread-1</lastname>
<country>Thailand</country>
<postcode>10220</postcode>
<email>66810000237@testrest.com</email>
<active_date>2553/10/22 23:04:06</active_date>
</subscriber>
```

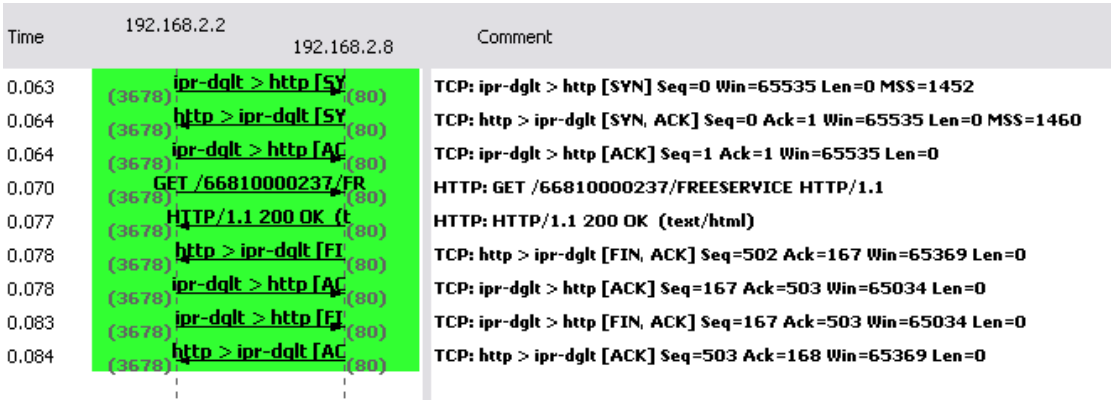
รูปที่ ก.14 การร้องขอในการเรียกใช้บริการ CREATE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST

```
HTTP/1.1 200 OK
Date: Fri, 22 Oct 2010 16:06:16 GMT
Server: Apache/2.2.8 (Win32) PHP/5.2.6
X-Powered-By: PHP/5.2.6
Content-Length: 67
Connection: close
Content-Type: text/html

<?xml version="1.0"?>
<subscriber>
<result>0</result>
</subscriber>
```

รูปที่ ก.15 การตอบกลับในการเรียกใช้บริการ CREATE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST

ตัวอย่างการเรียกใช้บริการ RETRIVE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST



รูปที่ ก.16 โฟลว์ของ TCP แพคเกจในการเรียกใช้บริการ RETRIVE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST

```
GET /66810000237/FREESERVICE HTTP/1.1
Connection: close
User-Agent: Java/1.6.0_20
Host: 192.168.2.8
```

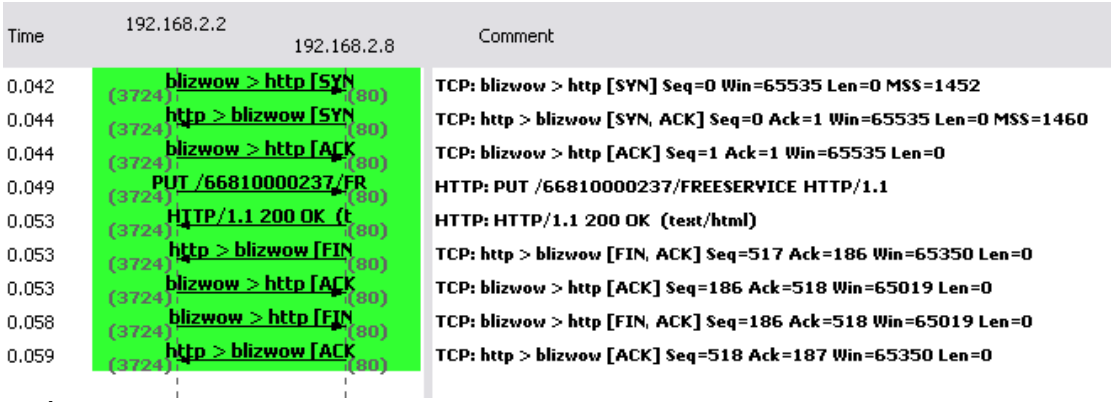
รูปที่ ก.17 การร้องขอในการเรียกใช้บริการ RETRIVE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST

```
HTTP/1.1 200 OK
Date: Fri, 22 Oct 2010 16:38:15 GMT
Server: Apache/2.2.8 (Win32) PHP/5.2.6
X-Powered-By: PHP/5.2.6
Content-Length: 315
Connection: close
Content-Type: text/html

<?xml version="1.0"?><subscriber>
<msisdn>66810000237</msisdn>
<id>3001242519358</id>
<title>mr</title>
<firstname>REST</firstname>
<lastname>Thread-1</lastname>
<country>Thailand</country>
<postcode>10220</postcode>
<email>66810000237@testrest.com</email>
<active_date>2553/10/22 23:04:06</active_date></subscriber>
```

รูปที่ ก.18 การตอบกลับในการเรียกใช้บริการ RETRIVE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST

ตัวอย่างการเรียกใช้บริการ UPDATE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST



รูปที่ ก.19 โฟลว์ของ TCP แพคเกจในการเรียกใช้บริการ UPDATE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST

```
PUT /66810000237/FREESERVICE HTTP/1.1
Content-Type: text/xml
Connection: close
User-Agent: Jakarta Commons-HttpClient/3.1
Host: 192.168.2.8
Content-Length: 317

<?xml version="1.0"?>
<subscriber>
<msisdn>66877898562</msisdn>
<id>3000247898562</id>
<title>dr</title>
<firstname>xREST</firstname>
<lastname>xThread-1</lastname>
<country>SIAM</country>
<postcode>12345</postcode>
<email>66877898562@testrest.com</email>
<active_date>2553/10/22 23:36:02</active_date>
</subscriber>
```

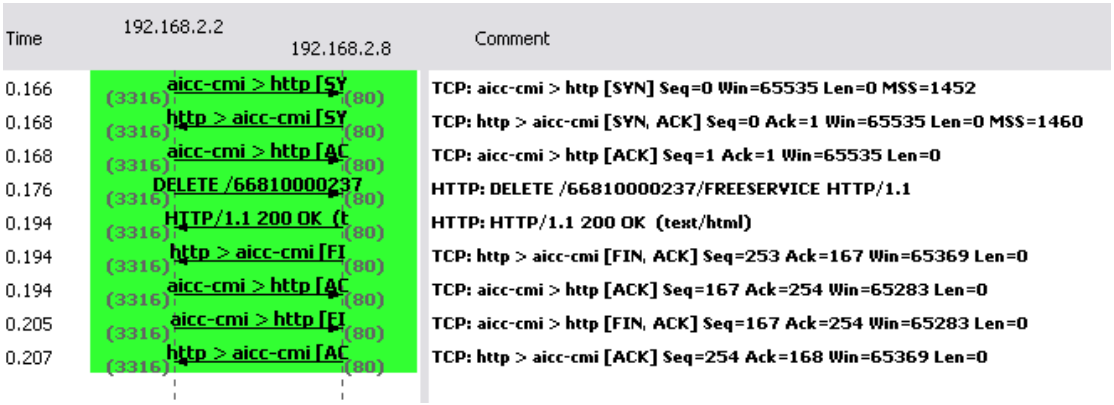
รูปที่ ก.20 การร้องขอในการเรียกใช้บริการ UPDATE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST

```
HTTP/1.1 200 OK
Date: Fri, 22 Oct 2010 16:06:16 GMT
Server: Apache/2.2.8 (Win32) PHP/5.2.6
X-Powered-By: PHP/5.2.6
Content-Length: 67
Connection: close
Content-Type: text/html

<?xml version="1.0"?>
<subscriber>
<result>0</result>
</subscriber>
```

รูปที่ ก.21 การตอบกลับในการเรียกใช้บริการ UPDATE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST

ตัวอย่างการเรียกใช้บริการ DELETE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST



รูปที่ ก.22 โฟลว์ของ TCP แพคเกจในการเรียกใช้บริการ DELETE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST

```
DELETE /66810000237/FREESERVICE  HTTP/1.1
Connection: close
User-Agent: Java/1.6.0_20
Host: 192.168.2.8
```

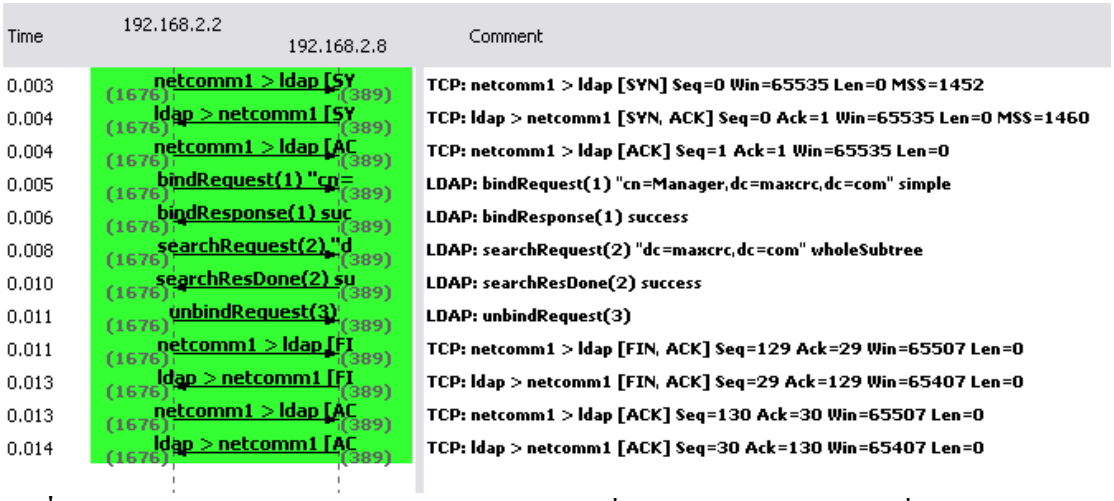
รูปที่ ก.23 การร้องขอในการเรียกใช้บริการ DELETE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST

```
HTTP/1.1 200 OK
Date: Fri, 22 Oct 2010 16:06:16 GMT
Server: Apache/2.2.8 (Win32) PHP/5.2.6
X-Powered-By: PHP/5.2.6
Content-Length: 67
Connection: close
Content-Type: text/html

<?xml version="1.0"?>
<subscriber>
<result>0</result>
</subscriber>
```

รูปที่ ก.24 การตอบกลับในการเรียกใช้บริการ DELETE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST

ตัวอย่างการค้นหาข้อมูลบนเครื่องแม่ข่าย LDAP ผ่านคำสั่ง LDAP

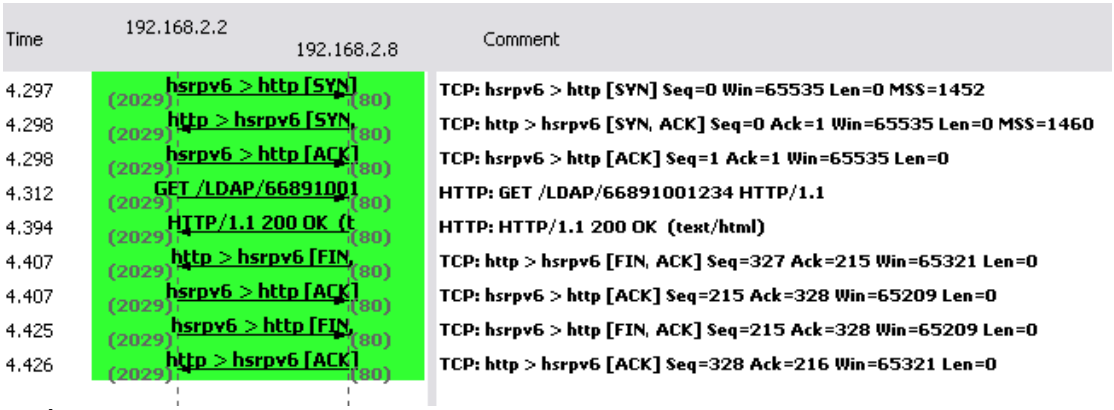


รูปที่ ก.25 โฟลว์ของ TCP ของการค้นหาข้อมูลบนเครื่องแม่ข่าย LDAP ผ่านคำสั่ง LDAP

Protocol	Info
TCP	netcomm1 > ldap [SYN] Seq=0 Win=65535 Len=0 MSS=1452
TCP	ldap > netcomm1 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
TCP	netcomm1 > ldap [ACK] Seq=1 Ack=1 Win=65535 Len=0
LDAP	bindRequest(1) "cn=Manager,dc=maxcrc,dc=com" simple
LDAP	bindResponse(1) success
LDAP	searchRequest(2) "dc=maxcrc,dc=com" wholeSubtree
LDAP	searchResDone(2) success [0 results]
LDAP	unbindRequest(3)
TCP	netcomm1 > ldap [FIN, ACK] Seq=129 Ack=29 Win=65507 Len=0
TCP	ldap > netcomm1 [FIN, ACK] Seq=29 Ack=129 Win=65407 Len=0
TCP	netcomm1 > ldap [ACK] Seq=130 Ack=30 Win=65507 Len=0
TCP	ldap > netcomm1 [ACK] Seq=30 Ack=130 Win=65407 Len=0

รูปที่ ก.26 ตัวอย่าง TCP แพคเกจของการค้นหาข้อมูลบนเครื่องแม่ข่าย LDAP ผ่านคำสั่ง LDAP

ตัวอย่างการค้นหาข้อมูลบนเครื่องแม่ข่าย LDAP ผ่านเว็บเซอร์วิสที่ใช้เซอร์วิสบัส



รูปที่ ก.27 โฟลว์ของ TCP ของในการค้นหาข้อมูลบนเครื่องแม่ข่าย LDAP ผ่านเว็บเซอร์วิสที่ใช้เซอร์วิสบัส

```
GET /LDAP/66891001234 HTTP/1.1
Connection: close
password: a94a8fe5ccb19ba61c4c0873d391e987982fbbd3
User-Agent: Java/1.6.0_20
Host: 192.168.2.8
```

รูปที่ ก.28 การร้องขอในการค้นหาข้อมูลบนเครื่องแม่ข่าย LDAP ผ่านเว็บเซอร์วิสที่ใช้เซอร์วิสบัส

```
HTTP/1.1 200 OK
Date: Fri, 22 Oct 2010 14:28:17 GMT
Server: Apache/2.2.8 (Win32) PHP/5.2.6
X-Powered-By: PHP/5.2.6
Content-Length: 140
Connection: close
Content-Type: text/html
<searchresult>
<cn>Chai, Somchai Kamdee</cn>
<objectClass>person</objectClass>
<sn>Kamdee</sn>
<telephoneNumber>66891001234</telephoneNumber>
<searchresult>
```

รูปที่ ก.29 การตอบกลับในการค้นหาข้อมูลบนเครื่องแม่ข่าย LDAP ผ่านเว็บเซอร์วิสที่ใช้เซอร์วิส

ภาคผนวก ข.

ผลการทดสอบเว็บเซอร์วิส

ตัวอย่างผลการทดสอบบริการ CREATE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP

ตารางที่ ข.1 ตัวอย่างผลการทดสอบบริการ CREATE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP ในช่วงเวลาที่มีการทำงานสูงสุด

บันทึกเวลา	บริการ	เวลาตอบสนอง (มิลลิวินาที)	HTTP Response	ขนาดข้อมูล (ไบต์)
1279387576621	SOAP-Create	16	200 OK	515
1279387576624	SOAP-Create	18	200 OK	515
1279387576627	SOAP-Create	20	200 OK	515
1279387576632	SOAP-Create	18	200 OK	515
1279387576636	SOAP-Create	17	200 OK	515
1279387576639	SOAP-Create	16	200 OK	515
1279387576643	SOAP-Create	16	200 OK	515
1279387576648	SOAP-Create	13	200 OK	515
1279387576651	SOAP-Create	14	200 OK	515
1279387576654	SOAP-Create	15	200 OK	515
1279387576656	SOAP-Create	17	200 OK	515
1279387576660	SOAP-Create	15	200 OK	515
1279387576662	SOAP-Create	21	200 OK	515
1279387576666	SOAP-Create	25	200 OK	515
1279387576671	SOAP-Create	23	200 OK	515
1279387576674	SOAP-Create	23	200 OK	515
1279387576676	SOAP-Create	22	200 OK	515
1279387576684	SOAP-Create	18	200 OK	515
1279387576692	SOAP-Create	13	200 OK	515

บันทึกเวลา	บริการ	เวลาตอบสนอง (มิลลิวินาที)	HTTP Response	ขนาดข้อมูล (ไบต์)
1279387576695	SOAP-Create	16	200 OK	515
1279387576698	SOAP-Create	17	200 OK	515
1279387576703	SOAP-Create	21	200 OK	515
1279387576700	SOAP-Create	19	200 OK	515
1279387576706	SOAP-Create	22	200 OK	515
1279387576712	SOAP-Create	20	200 OK	515
1279387576716	SOAP-Create	19	200 OK	515
1279387576721	SOAP-Create	17	200 OK	515
1279387576725	SOAP-Create	15	200 OK	515
1279387576730	SOAP-Create	14	200 OK	515
1279387576733	SOAP-Create	14	200 OK	515
1279387576736	SOAP-Create	18	200 OK	515
1279387576739	SOAP-Create	20	200 OK	515
1279387576742	SOAP-Create	20	200 OK	515
1279387576746	SOAP-Create	19	200 OK	515
1279387576749	SOAP-Create	19	200 OK	515
1279387576755	SOAP-Create	16	200 OK	515
1279387576760	SOAP-Create	15	200 OK	515
1279387576763	SOAP-Create	14	200 OK	515
1279387576767	SOAP-Create	17	200 OK	515
1279387576769	SOAP-Create	17	200 OK	515
1279387576773	SOAP-Create	20	200 OK	515
1279387576779	SOAP-Create	17	200 OK	515
1279387576776	SOAP-Create	24	200 OK	515
1279387576785	SOAP-Create	18	200 OK	515
1279387576787	SOAP-Create	18	200 OK	515
1279387576795	SOAP-Create	13	200 OK	515
1279387576798	SOAP-Create	15	200 OK	515
1279387576801	SOAP-Create	18	200 OK	515
1279387576804	SOAP-Create	20	200 OK	515

บันทึกเวลา	บริการ	เวลาตอบสนอง (มิลลิวินาที)	HTTP Response	ขนาดข้อมูล (ไบต์)
1279387576807	SOAP-Create	18	200 OK	515
1279387576810	SOAP-Create	19	200 OK	515
1279387576814	SOAP-Create	19	200 OK	515
1279387576820	SOAP-Create	19	200 OK	515
1279387576825	SOAP-Create	16	200 OK	515
1279387576827	SOAP-Create	18	200 OK	515
1279387576830	SOAP-Create	17	200 OK	515
1279387576834	SOAP-Create	18	200 OK	515
1279387576840	SOAP-Create	17	200 OK	515
1279387576842	SOAP-Create	17	200 OK	515
1279387576846	SOAP-Create	17	200 OK	515
1279387576849	SOAP-Create	17	200 OK	515
1279387576853	SOAP-Create	17	200 OK	515
1279387576858	SOAP-Create	17	200 OK	515
1279387576861	SOAP-Create	16	200 OK	515
1279387576864	SOAP-Create	15	200 OK	515
1279387576867	SOAP-Create	16	200 OK	515
1279387576871	SOAP-Create	18	200 OK	515
1279387576876	SOAP-Create	18	200 OK	515
1279387576878	SOAP-Create	19	200 OK	515
1279387576885	SOAP-Create	16	200 OK	515
1279387576881	SOAP-Create	24	200 OK	515
1279387576891	SOAP-Create	17	200 OK	515
1279387576895	SOAP-Create	17	200 OK	515
1279387576899	SOAP-Create	15	200 OK	515
1279387576902	SOAP-Create	17	200 OK	515
1279387576906	SOAP-Create	15	200 OK	515
1279387576910	SOAP-Create	17	200 OK	515
1279387576913	SOAP-Create	22	200 OK	515
1279387576915	SOAP-Create	24	200 OK	515

บันทึกเวลา	บริการ	เวลาตอบสนอง (มิลลิวินาที)	HTTP Response	ขนาดข้อมูล (ไบต์)
1279387576920	SOAP-Create	21	200 OK	515
1279387576923	SOAP-Create	21	200 OK	515
1279387576928	SOAP-Create	18	200 OK	515
1279387576937	SOAP-Create	14	200 OK	515
1279387576940	SOAP-Create	15	200 OK	515
1279387576942	SOAP-Create	16	200 OK	515
1279387576946	SOAP-Create	15	200 OK	515
1279387576948	SOAP-Create	18	200 OK	515
1279387576952	SOAP-Create	20	200 OK	515
1279387576956	SOAP-Create	20	200 OK	515
1279387576959	SOAP-Create	19	200 OK	515
1279387576962	SOAP-Create	19	200 OK	515
1279387576968	SOAP-Create	14	200 OK	515
1279387576973	SOAP-Create	16	200 OK	515
1279387576978	SOAP-Create	14	200 OK	515
1279387576980	SOAP-Create	19	200 OK	515
1279387576982	SOAP-Create	22	200 OK	515
1279387576983	SOAP-Create	23	200 OK	515
1279387576990	SOAP-Create	21	200 OK	515
1279387576994	SOAP-Create	24	200 OK	515
1279387577001	SOAP-Create	21	200 OK	515

ตัวอย่างผลการทดสอบบริการ RETRIVE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP

ตารางที่ ข.2 ตัวอย่างผลการทดสอบบริการ RETRIVE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP ในช่วงเวลาที่มีการทำงานสูงสุด

บันทึกเวลา	บริการ	เวลาตอบสนอง (มิลลิวินาที)	HTTP Response	ขนาดข้อมูล (ไบต์)
1279393379891	SOAP-Retrieve	23	200 OK	1460
1279393379908	SOAP-Retrieve	30	200 OK	1460
1279393379913	SOAP-Retrieve	28	200 OK	1460
1279393379913	SOAP-Retrieve	30	200 OK	1460
1279393379908	SOAP-Retrieve	37	200 OK	1460
1279393379915	SOAP-Retrieve	33	200 OK	1460
1279393379939	SOAP-Retrieve	13	200 OK	1460
1279393379942	SOAP-Retrieve	17	200 OK	1460
1279393379944	SOAP-Retrieve	21	200 OK	1460
1279393379946	SOAP-Retrieve	25	200 OK	1460
1279393379954	SOAP-Retrieve	20	200 OK	1460
1279393379949	SOAP-Retrieve	30	200 OK	1460
1279393379960	SOAP-Retrieve	20	200 OK	1460
1279393379966	SOAP-Retrieve	17	200 OK	1460
1279393379973	SOAP-Retrieve	14	200 OK	1460
1279393379975	SOAP-Retrieve	17	200 OK	1460
1279393379981	SOAP-Retrieve	20	200 OK	1460
1279393379981	SOAP-Retrieve	32	200 OK	1460
1279393379988	SOAP-Retrieve	26	200 OK	1460
1279393379993	SOAP-Retrieve	24	200 OK	1460
1279393379984	SOAP-Retrieve	37	200 OK	1460
1279393380003	SOAP-Retrieve	19	200 OK	1460
1279393380014	SOAP-Retrieve	18	200 OK	1460
1279393380015	SOAP-Retrieve	20	200 OK	1460
1279393380019	SOAP-Retrieve	22	200 OK	1460

บันทึกเวลา	บริการ	เวลาตอบสนอง (มิลลิวินาที)	HTTP Response	ขนาดข้อมูล (ไบต์)
1279393380022	SOAP-Retrieve	25	200 OK	1460
1279393380023	SOAP-Retrieve	26	200 OK	1460
1279393380033	SOAP-Retrieve	19	200 OK	1460
1279393380036	SOAP-Retrieve	19	200 OK	1460
1279393380042	SOAP-Retrieve	16	200 OK	1460
1279393380048	SOAP-Retrieve	14	200 OK	1460
1279393380050	SOAP-Retrieve	17	200 OK	1460
1279393380053	SOAP-Retrieve	21	200 OK	1460
1279393380056	SOAP-Retrieve	23	200 OK	1460
1279393380063	SOAP-Retrieve	19	200 OK	1460
1279393380060	SOAP-Retrieve	26	200 OK	1460
1279393380068	SOAP-Retrieve	22	200 OK	1460
1279393380075	SOAP-Retrieve	16	200 OK	1460
1279393380080	SOAP-Retrieve	14	200 OK	1460
1279393380083	SOAP-Retrieve	16	200 OK	1460
1279393380087	SOAP-Retrieve	28	200 OK	1460
1279393380091	SOAP-Retrieve	30	200 OK	1460
1279393380092	SOAP-Retrieve	32	200 OK	1460
1279393380095	SOAP-Retrieve	33	200 OK	1460
1279393380100	SOAP-Retrieve	30	200 OK	1460
1279393380117	SOAP-Retrieve	16	200 OK	1460
1279393380122	SOAP-Retrieve	14	200 OK	1460
1279393380126	SOAP-Retrieve	20	200 OK	1460
1279393380130	SOAP-Retrieve	20	200 OK	1460
1279393380131	SOAP-Retrieve	23	200 OK	1460
1279393380138	SOAP-Retrieve	19	200 OK	1460
1279393380135	SOAP-Retrieve	27	200 OK	1460
1279393380147	SOAP-Retrieve	17	200 OK	1460
1279393380151	SOAP-Retrieve	17	200 OK	1460
1279393380156	SOAP-Retrieve	14	200 OK	1460

บันทึกเวลา	บริการ	เวลาตอบสนอง (มิลลิวินาที)	HTTP Response	ขนาดข้อมูล (ไบต์)
1279393380158	SOAP-Retrieve	17	200 OK	1460
1279393380163	SOAP-Retrieve	16	200 OK	1460
1279393380165	SOAP-Retrieve	21	200 OK	1460
1279393380169	SOAP-Retrieve	20	200 OK	1460
1279393380171	SOAP-Retrieve	24	200 OK	1460
1279393380176	SOAP-Retrieve	24	200 OK	1460
1279393380181	SOAP-Retrieve	22	200 OK	1460
1279393380187	SOAP-Retrieve	19	200 OK	1460
1279393380190	SOAP-Retrieve	20	200 OK	1460
1279393380197	SOAP-Retrieve	15	200 OK	1460
1279393380201	SOAP-Retrieve	17	200 OK	1460
1279393380204	SOAP-Retrieve	15	200 OK	1460
1279393380207	SOAP-Retrieve	24	200 OK	1460
1279393380213	SOAP-Retrieve	26	200 OK	1460
1279393380211	SOAP-Retrieve	33	200 OK	1460
1279393380220	SOAP-Retrieve	26	200 OK	1460
1279393380220	SOAP-Retrieve	28	200 OK	1460
1279393380232	SOAP-Retrieve	19	200 OK	1460
1279393380240	SOAP-Retrieve	13	200 OK	1460
1279393380245	SOAP-Retrieve	16	200 OK	1460
1279393380247	SOAP-Retrieve	25	200 OK	1460
1279393380249	SOAP-Retrieve	26	200 OK	1460
1279393380252	SOAP-Retrieve	27	200 OK	1460
1279393380255	SOAP-Retrieve	24	200 OK	1460
1279393380262	SOAP-Retrieve	19	200 OK	1460
1279393380274	SOAP-Retrieve	13	200 OK	1460
1279393380276	SOAP-Retrieve	18	200 OK	1460
1279393380281	SOAP-Retrieve	27	200 OK	1460
1279393380283	SOAP-Retrieve	27	200 OK	1460
1279393380280	SOAP-Retrieve	32	200 OK	1460

บันทึกเวลา	บริการ	เวลาตอบสนอง (มิลลิวินาที)	HTTP Response	ขนาดข้อมูล (ไบต์)
1279393380288	SOAP-Retrieve	27	200 OK	1460
1279393380296	SOAP-Retrieve	24	200 OK	1460
1279393380309	SOAP-Retrieve	21	200 OK	1460
1279393380311	SOAP-Retrieve	28	200 OK	1460
1279393380313	SOAP-Retrieve	31	200 OK	1460
1279393380316	SOAP-Retrieve	29	200 OK	1460
1279393380321	SOAP-Retrieve	28	200 OK	1460
1279393380331	SOAP-Retrieve	25	200 OK	1460
1279393380340	SOAP-Retrieve	21	200 OK	1460
1279393380345	SOAP-Retrieve	19	200 OK	1460
1279393380346	SOAP-Retrieve	22	200 OK	1460
1279393380351	SOAP-Retrieve	19	200 OK	1460
1279393380358	SOAP-Retrieve	16	200 OK	1460
1279393380362	SOAP-Retrieve	14	200 OK	1460
1279393380365	SOAP-Retrieve	16	200 OK	1460

ตัวอย่างผลการทดสอบบริการ UPDATE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP

ตารางที่ ข.3 ตัวอย่างผลการทดสอบบริการ UPDATE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP ในช่วงเวลาที่มีการทำงานสูงสุด

บันทึกเวลา	บริการ	เวลาตอบสนอง (มิลลิวินาที)	HTTP Response	ขนาดข้อมูล (ไบต์)
1279390482698	SOAP-Update	26	200	515
1279390482700	SOAP-Update	24	200	515
1279390482707	SOAP-Update	19	200	515
1279390482717	SOAP-Update	14	200	515
1279390482721	SOAP-Update	17	200	515
1279390482725	SOAP-Update	20	200	515
1279390482726	SOAP-Update	23	200	515

บันทึกเวลา	บริการ	เวลาตอบสนอง (มิลลิวินาที)	HTTP Response	ขนาดข้อมูล (ไบต์)
1279390482732	SOAP-Update	20	200	515
1279390482728	SOAP-Update	26	200	515
1279390482740	SOAP-Update	18	200	515
1279390482747	SOAP-Update	16	200	515
1279390482750	SOAP-Update	18	200	515
1279390482753	SOAP-Update	19	200	515
1279390482756	SOAP-Update	19	200	515
1279390482760	SOAP-Update	18	200	515
1279390482765	SOAP-Update	17	200	515
1279390482770	SOAP-Update	15	200	515
1279390482773	SOAP-Update	16	200	515
1279390482776	SOAP-Update	19	200	515
1279390482779	SOAP-Update	19	200	515
1279390482784	SOAP-Update	16	200	515
1279390482786	SOAP-Update	18	200	515
1279390482790	SOAP-Update	17	200	515
1279390482796	SOAP-Update	17	200	515
1279390482800	SOAP-Update	19	200	515
1279390482802	SOAP-Update	20	200	515
1279390482806	SOAP-Update	20	200	515
1279390482808	SOAP-Update	20	200	515
1279390482814	SOAP-Update	18	200	515
1279390482820	SOAP-Update	14	200	515
1279390482824	SOAP-Update	15	200	515
1279390482827	SOAP-Update	18	200	515
1279390482833	SOAP-Update	18	200	515
1279390482835	SOAP-Update	19	200	515
1279390482829	SOAP-Update	29	200	515
1279390482840	SOAP-Update	21	200	515
1279390482846	SOAP-Update	20	200	515

บันทึกเวลา	บริการ	เวลาตอบสนอง (มิลลิวินาที)	HTTP Response	ขนาดข้อมูล (ไบต์)
1279390482852	SOAP-Update	18	200	515
1279390482856	SOAP-Update	17	200	515
1279390482859	SOAP-Update	16	200	515
1279390482862	SOAP-Update	18	200	515
1279390482867	SOAP-Update	16	200	515
1279390482871	SOAP-Update	16	200	515
1279390482874	SOAP-Update	17	200	515
1279390482877	SOAP-Update	21	200	515
1279390482881	SOAP-Update	18	200	515
1279390482884	SOAP-Update	18	200	515
1279390482889	SOAP-Update	17	200	515
1279390482892	SOAP-Update	17	200	515
1279390482901	SOAP-Update	14	200	515
1279390482899	SOAP-Update	21	200	515
1279390482903	SOAP-Update	23	200	515
1279390482907	SOAP-Update	22	200	515
1279390482910	SOAP-Update	22	200	515
1279390482916	SOAP-Update	19	200	515
1279390482922	SOAP-Update	16	200	515
1279390482927	SOAP-Update	16	200	515
1279390482931	SOAP-Update	17	200	515
1279390482934	SOAP-Update	19	200	515
1279390482936	SOAP-Update	20	200	515
1279390482939	SOAP-Update	20	200	515
1279390482944	SOAP-Update	18	200	515
1279390482949	SOAP-Update	16	200	515
1279390482954	SOAP-Update	15	200	515
1279390482957	SOAP-Update	17	200	515
1279390482961	SOAP-Update	17	200	515
1279390482963	SOAP-Update	20	200	515

บันทึกเวลา	บริการ	เวลาตอบสนอง (มิลลิวินาที)	HTTP Response	ขนาดข้อมูล (ไบต์)
1279390482967	SOAP-Update	20	200	515
1279390482971	SOAP-Update	20	200	515
1279390482976	SOAP-Update	17	200	515
1279390482979	SOAP-Update	22	200	515
1279390482984	SOAP-Update	17	200	515
1279390482988	SOAP-Update	17	200	515
1279390482993	SOAP-Update	15	200	515
1279390482995	SOAP-Update	17	200	515
1279390483002	SOAP-Update	21	200	515
1279390483003	SOAP-Update	25	200	515
1279390483006	SOAP-Update	25	200	515
1279390483010	SOAP-Update	24	200	515
1279390483013	SOAP-Update	22	200	515
1279390483024	SOAP-Update	17	200	515
1279390483029	SOAP-Update	19	200	515
1279390483032	SOAP-Update	21	200	515
1279390483035	SOAP-Update	21	200	515
1279390483036	SOAP-Update	23	200	515
1279390483042	SOAP-Update	20	200	515
1279390483049	SOAP-Update	17	200	515
1279390483054	SOAP-Update	17	200	515
1279390483057	SOAP-Update	19	200	515
1279390483060	SOAP-Update	19	200	515
1279390483064	SOAP-Update	18	200	515
1279390483067	SOAP-Update	18	200	515
1279390483073	SOAP-Update	16	200	515
1279390483077	SOAP-Update	17	200	515
1279390483080	SOAP-Update	19	200	515
1279390483084	SOAP-Update	19	200	515
1279390483086	SOAP-Update	20	200	515

บันทึกเวลา	บริการ	เวลาตอบสนอง (มิลลิวินาที)	HTTP Response	ขนาดข้อมูล (ไบต์)
1279390483090	SOAP-Update	18	200	515
1279390483095	SOAP-Update	15	200	515
1279390483101	SOAP-Update	15	200	515

ตัวอย่างผลการทดสอบบริการ DELETE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP

ตารางที่ ข.4 ตัวอย่างผลการทดสอบบริการ DELETE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP ในช่วงเวลาที่มีการทำงานสูงสุด

บันทึกเวลา	บริการ	เวลาตอบสนอง (มิลลิวินาที)	HTTP Response	ขนาดข้อมูล (ไบต์)
1279392250085	SOAP-Delete	20	200	515
1279392250088	SOAP-Delete	20	200	515
1279392250094	SOAP-Delete	16	200	515
1279392250096	SOAP-Delete	16	200	515
1279392250103	SOAP-Delete	13	200	515
1279392250107	SOAP-Delete	13	200	515
1279392250113	SOAP-Delete	15	200	515
1279392250115	SOAP-Delete	15	200	515
1279392250110	SOAP-Delete	25	200	515
1279392250119	SOAP-Delete	19	200	515
1279392250123	SOAP-Delete	20	200	515
1279392250131	SOAP-Delete	15	200	515
1279392250133	SOAP-Delete	16	200	515
1279392250137	SOAP-Delete	15	200	515
1279392250139	SOAP-Delete	18	200	515
1279392250144	SOAP-Delete	18	200	515
1279392250147	SOAP-Delete	17	200	515
1279392250151	SOAP-Delete	16	200	515

บันทึกเวลา	บริการ	เวลาตอบสนอง (มิลลิวินาที)	HTTP Response	ขนาดข้อมูล (ไบต์)
1279392250154	SOAP-Delete	16	200	515
1279392250159	SOAP-Delete	13	200	515
1279392250163	SOAP-Delete	15	200	515
1279392250165	SOAP-Delete	20	200	515
1279392250169	SOAP-Delete	22	200	515
1279392250171	SOAP-Delete	23	200	515
1279392250174	SOAP-Delete	23	200	515
1279392250180	SOAP-Delete	19	200	515
1279392250186	SOAP-Delete	17	200	515
1279392250193	SOAP-Delete	13	200	515
1279392250196	SOAP-Delete	15	200	515
1279392250198	SOAP-Delete	19	200	515
1279392250201	SOAP-Delete	19	200	515
1279392250204	SOAP-Delete	18	200	515
1279392250208	SOAP-Delete	20	200	515
1279392250213	SOAP-Delete	16	200	515
1279392250218	SOAP-Delete	13	200	515
1279392250221	SOAP-Delete	16	200	515
1279392250224	SOAP-Delete	19	200	515
1279392250230	SOAP-Delete	21	200	515
1279392250231	SOAP-Delete	23	200	515
1279392250232	SOAP-Delete	24	200	515
1279392250239	SOAP-Delete	20	200	515
1279392250244	SOAP-Delete	18	200	515
1279392250252	SOAP-Delete	13	200	515
1279392250255	SOAP-Delete	15	200	515
1279392250257	SOAP-Delete	17	200	515
1279392250260	SOAP-Delete	19	200	515
1279392250263	SOAP-Delete	19	200	515
1279392250266	SOAP-Delete	19	200	515

บันทึกเวลา	บริการ	เวลาตอบสนอง (มิลลิวินาที)	HTTP Response	ขนาดข้อมูล (ไบต์)
1279392250271	SOAP-Delete	17	200	515
1279392250275	SOAP-Delete	15	200	515
1279392250280	SOAP-Delete	14	200	515
1279392250283	SOAP-Delete	14	200	515
1279392250286	SOAP-Delete	16	200	515
1279392250289	SOAP-Delete	16	200	515
1279392250292	SOAP-Delete	18	200	515
1279392250296	SOAP-Delete	21	200	515
1279392250298	SOAP-Delete	24	200	515
1279392250303	SOAP-Delete	24	200	515
1279392250306	SOAP-Delete	25	200	515
1279392250312	SOAP-Delete	20	200	515
1279392250318	SOAP-Delete	16	200	515
1279392250323	SOAP-Delete	14	200	515
1279392250328	SOAP-Delete	15	200	515
1279392250334	SOAP-Delete	20	200	515
1279392250332	SOAP-Delete	24	200	515
1279392250336	SOAP-Delete	23	200	515
1279392250338	SOAP-Delete	23	200	515
1279392250345	SOAP-Delete	18	200	515
1279392250355	SOAP-Delete	13	200	515
1279392250357	SOAP-Delete	15	200	515
1279392250362	SOAP-Delete	22	200	515
1279392250365	SOAP-Delete	21	200	515
1279392250361	SOAP-Delete	27	200	515
1279392250370	SOAP-Delete	20	200	515
1279392250373	SOAP-Delete	19	200	515
1279392250385	SOAP-Delete	13	200	515
1279392250387	SOAP-Delete	18	200	515
1279392250389	SOAP-Delete	23	200	515

บันทึกเวลา	บริการ	เวลาตอบสนอง (มิลลิวินาที)	HTTP Response	ขนาดข้อมูล (ไบต์)
1279392250392	SOAP-Delete	23	200	515
1279392250400	SOAP-Delete	18	200	515
1279392250393	SOAP-Delete	27	200	515
1279392250406	SOAP-Delete	18	200	515
1279392250413	SOAP-Delete	20	200	515
1279392250416	SOAP-Delete	19	200	515
1279392250419	SOAP-Delete	18	200	515
1279392250422	SOAP-Delete	17	200	515
1279392250426	SOAP-Delete	15	200	515
1279392250434	SOAP-Delete	13	200	515
1279392250436	SOAP-Delete	15	200	515
1279392250441	SOAP-Delete	17	200	515
1279392250438	SOAP-Delete	24	200	515
1279392250443	SOAP-Delete	21	200	515
1279392250448	SOAP-Delete	19	200	515
1279392250453	SOAP-Delete	17	200	515
1279392250460	SOAP-Delete	13	200	515
1279392250463	SOAP-Delete	14	200	515
1279392250466	SOAP-Delete	18	200	515
1279392250468	SOAP-Delete	22	200	515
1279392250471	SOAP-Delete	22	200	515
1279392250478	SOAP-Delete	18	200	515

ตัวอย่างผลการทดสอบบริการ CREATE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST

ตารางที่ ข.5 ตัวอย่างผลการทดสอบบริการ CREATE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST ในช่วงเวลาที่มีการทำงานสูงสุด

บันทึกเวลา	บริการ	เวลาตอบสนอง (มิลลิวินาที)	HTTP Response	ขนาดข้อมูล (ไบต์)
1279388548790	REST-Create	14	200 OK	67
1279388548793	REST-Create	15	200 OK	67
1279388548797	REST-Create	15	200 OK	67
1279388548800	REST-Create	15	200 OK	67
1279388548803	REST-Create	14	200 OK	67
1279388548806	REST-Create	16	200 OK	67
1279388548810	REST-Create	14	200 OK	67
1279388548813	REST-Create	15	200 OK	67
1279388548817	REST-Create	13	200 OK	67
1279388548818	REST-Create	16	200 OK	67
1279388548823	REST-Create	14	200 OK	67
1279388548826	REST-Create	15	200 OK	67
1279388548830	REST-Create	13	200 OK	67
1279388548832	REST-Create	15	200 OK	67
1279388548836	REST-Create	14	200 OK	67
1279388548839	REST-Create	14	200 OK	67
1279388548842	REST-Create	14	200 OK	67
1279388548845	REST-Create	15	200 OK	67
1279388548848	REST-Create	13	200 OK	67
1279388548851	REST-Create	14	200 OK	67
1279388548855	REST-Create	14	200 OK	67
1279388548858	REST-Create	16	200 OK	67
1279388548861	REST-Create	17	200 OK	67
1279388548863	REST-Create	20	200 OK	67
1279388548866	REST-Create	20	200 OK	67

บันทึกเวลา	บริการ	เวลาตอบสนอง (มิลลิวินาที)	HTTP Response	ขนาดข้อมูล (ไบต์)
1279388548871	REST-Create	18	200 OK	67
1279388548876	REST-Create	15	200 OK	67
1279388548879	REST-Create	15	200 OK	67
1279388548884	REST-Create	15	200 OK	67
1279388548888	REST-Create	14	200 OK	67
1279388548893	REST-Create	14	200 OK	67
1279388548891	REST-Create	21	200 OK	67
1279388548895	REST-Create	19	200 OK	67
1279388548900	REST-Create	18	200 OK	67
1279388548904	REST-Create	16	200 OK	67
1279388548909	REST-Create	14	200 OK	67
1279388548914	REST-Create	12	200 OK	67
1279388548916	REST-Create	13	200 OK	67
1279388548920	REST-Create	14	200 OK	67
1279388548924	REST-Create	15	200 OK	67
1279388548922	REST-Create	22	200 OK	67
1279388548931	REST-Create	16	200 OK	67
1279388548928	REST-Create	22	200 OK	67
1279388548935	REST-Create	17	200 OK	67
1279388548941	REST-Create	14	200 OK	67
1279388548946	REST-Create	17	200 OK	67
1279388548948	REST-Create	18	200 OK	67
1279388548952	REST-Create	16	200 OK	67
1279388548953	REST-Create	18	200 OK	67
1279388548956	REST-Create	18	200 OK	67
1279388548964	REST-Create	15	200 OK	67
1279388548967	REST-Create	15	200 OK	67
1279388548969	REST-Create	17	200 OK	67
1279388548973	REST-Create	16	200 OK	67
1279388548976	REST-Create	17	200 OK	67

บันทึกเวลา	บริการ	เวลาตอบสนอง (มิลลิวินาที)	HTTP Response	ขนาดข้อมูล (ไบต์)
1279388548981	REST-Create	15	200 OK	67
1279388548984	REST-Create	15	200 OK	67
1279388548988	REST-Create	14	200 OK	67
1279388548991	REST-Create	14	200 OK	67
1279388548995	REST-Create	13	200 OK	67
1279388548998	REST-Create	13	200 OK	67
1279388549001	REST-Create	15	200 OK	67
1279388549004	REST-Create	16	200 OK	67
1279388549006	REST-Create	18	200 OK	67
1279388549010	REST-Create	17	200 OK	67
1279388549013	REST-Create	17	200 OK	67
1279388549018	REST-Create	15	200 OK	67
1279388549022	REST-Create	14	200 OK	67
1279388549025	REST-Create	14	200 OK	67
1279388549028	REST-Create	13	200 OK	67
1279388549031	REST-Create	15	200 OK	67
1279388549035	REST-Create	12	200 OK	67
1279388549038	REST-Create	14	200 OK	67
1279388549041	REST-Create	16	200 OK	67
1279388549042	REST-Create	17	200 OK	67
1279388549047	REST-Create	19	200 OK	67
1279388549049	REST-Create	17	200 OK	67
1279388549054	REST-Create	14	200 OK	67
1279388549058	REST-Create	13	200 OK	67
1279388549061	REST-Create	15	200 OK	67
1279388549067	REST-Create	17	200 OK	67
1279388549069	REST-Create	19	200 OK	67
1279388549068	REST-Create	22	200 OK	67
1279388549073	REST-Create	21	200 OK	67
1279388549078	REST-Create	18	200 OK	67

บันทึกเวลา	บริการ	เวลาตอบสนอง (มิลลิวินาที)	HTTP Response	ขนาดข้อมูล (ไบต์)
1279388549086	REST-Create	14	200 OK	67
1279388549089	REST-Create	14	200 OK	67
1279388549091	REST-Create	15	200 OK	67
1279388549095	REST-Create	14	200 OK	67
1279388549098	REST-Create	15	200 OK	67
1279388549102	REST-Create	15	200 OK	67
1279388549105	REST-Create	15	200 OK	67
1279388549108	REST-Create	16	200 OK	67
1279388549111	REST-Create	16	200 OK	67
1279388549114	REST-Create	16	200 OK	67
1279388549119	REST-Create	13	200 OK	67
1279388549122	REST-Create	14	200 OK	67
1279388549125	REST-Create	14	200 OK	67
1279388549128	REST-Create	14	200 OK	67
1279388549132	REST-Create	13	200 OK	67

ตัวอย่างผลการทดสอบบริการ RETRIVE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST

ตารางที่ ข.6 ตัวอย่างผลการทดสอบบริการ RETRIVE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST ในช่วงเวลาที่มีการทำงานสูงสุด

บันทึกเวลา	บริการ	เวลาตอบสนอง (มิลลิวินาที)	HTTP Response	ขนาดข้อมูล (ไบต์)
1279392743735	REST-Retrieve	15	200 OK	316
1279392743738	REST-Retrieve	17	200 OK	316
1279392743741	REST-Retrieve	17	200 OK	316
1279392743745	REST-Retrieve	16	200 OK	316
1279392743748	REST-Retrieve	16	200 OK	316
1279392743752	REST-Retrieve	16	200 OK	316

บันทึกเวลา	บริการ	เวลาตอบสนอง (มิลลิวินาที)	HTTP Response	ขนาดข้อมูล (ไบต์)
1279392743756	REST-Retrieve	15	200 OK	316
1279392743759	REST-Retrieve	16	200 OK	316
1279392743763	REST-Retrieve	14	200 OK	316
1279392743765	REST-Retrieve	16	200 OK	316
1279392743769	REST-Retrieve	14	200 OK	316
1279392743773	REST-Retrieve	14	200 OK	316
1279392743776	REST-Retrieve	14	200 OK	316
1279392743779	REST-Retrieve	15	200 OK	316
1279392743782	REST-Retrieve	17	200 OK	316
1279392743784	REST-Retrieve	16	200 OK	316
1279392743789	REST-Retrieve	13	200 OK	316
1279392743792	REST-Retrieve	15	200 OK	316
1279392743795	REST-Retrieve	19	200 OK	316
1279392743800	REST-Retrieve	19	200 OK	316
1279392743802	REST-Retrieve	21	200 OK	316
1279392743803	REST-Retrieve	22	200 OK	316
1279392743809	REST-Retrieve	20	200 OK	316
1279392743815	REST-Retrieve	17	200 OK	316
1279392743821	REST-Retrieve	14	200 OK	316
1279392743824	REST-Retrieve	14	200 OK	316
1279392743827	REST-Retrieve	14	200 OK	316
1279392743830	REST-Retrieve	15	200 OK	316
1279392743833	REST-Retrieve	16	200 OK	316
1279392743837	REST-Retrieve	14	200 OK	316
1279392743839	REST-Retrieve	19	200 OK	316
1279392743842	REST-Retrieve	20	200 OK	316
1279392743846	REST-Retrieve	21	200 OK	316
1279392743851	REST-Retrieve	18	200 OK	316
1279392743853	REST-Retrieve	19	200 OK	316
1279392743860	REST-Retrieve	15	200 OK	316

บันทึกเวลา	บริการ	เวลาตอบสนอง (มิลลิวินาที)	HTTP Response	ขนาดข้อมูล (ไบต์)
1279392743864	REST-Retrieve	16	200 OK	316
1279392743868	REST-Retrieve	16	200 OK	316
1279392743870	REST-Retrieve	17	200 OK	316
1279392743873	REST-Retrieve	16	200 OK	316
1279392743877	REST-Retrieve	16	200 OK	316
1279392743882	REST-Retrieve	14	200 OK	316
1279392743886	REST-Retrieve	16	200 OK	316
1279392743889	REST-Retrieve	15	200 OK	316
1279392743891	REST-Retrieve	15	200 OK	316
1279392743894	REST-Retrieve	14	200 OK	316
1279392743897	REST-Retrieve	15	200 OK	316
1279392743904	REST-Retrieve	16	200 OK	316
1279392743909	REST-Retrieve	18	200 OK	316
1279392743906	REST-Retrieve	24	200 OK	316
1279392743907	REST-Retrieve	26	200 OK	316
1279392743914	REST-Retrieve	22	200 OK	316
1279392743922	REST-Retrieve	17	200 OK	316
1279392743928	REST-Retrieve	14	200 OK	316
1279392743932	REST-Retrieve	14	200 OK	316
1279392743935	REST-Retrieve	16	200 OK	316
1279392743937	REST-Retrieve	18	200 OK	316
1279392743941	REST-Retrieve	17	200 OK	316
1279392743944	REST-Retrieve	17	200 OK	316
1279392743948	REST-Retrieve	16	200 OK	316
1279392743953	REST-Retrieve	14	200 OK	316
1279392743956	REST-Retrieve	15	200 OK	316
1279392743960	REST-Retrieve	15	200 OK	316
1279392743963	REST-Retrieve	16	200 OK	316
1279392743966	REST-Retrieve	18	200 OK	316
1279392743969	REST-Retrieve	20	200 OK	316

บันทึกเวลา	บริการ	เวลาตอบสนอง (มิลลิวินาที)	HTTP Response	ขนาดข้อมูล (ไบต์)
1279392743972	REST-Retrieve	20	200 OK	316
1279392743977	REST-Retrieve	19	200 OK	316
1279392743981	REST-Retrieve	18	200 OK	316
1279392743985	REST-Retrieve	19	200 OK	316
1279392743990	REST-Retrieve	15	200 OK	316
1279392743994	REST-Retrieve	15	200 OK	316
1279392743997	REST-Retrieve	15	200 OK	316
1279392744001	REST-Retrieve	18	200 OK	316
1279392744005	REST-Retrieve	20	200 OK	316
1279392744006	REST-Retrieve	22	200 OK	316
1279392744010	REST-Retrieve	20	200 OK	316
1279392744013	REST-Retrieve	19	200 OK	316
1279392744020	REST-Retrieve	16	200 OK	316
1279392744027	REST-Retrieve	12	200 OK	316
1279392744029	REST-Retrieve	15	200 OK	316
1279392744034	REST-Retrieve	13	200 OK	316
1279392744032	REST-Retrieve	19	200 OK	316
1279392744038	REST-Retrieve	16	200 OK	316
1279392744041	REST-Retrieve	16	200 OK	316
1279392744046	REST-Retrieve	15	200 OK	316
1279392744049	REST-Retrieve	15	200 OK	316
1279392744052	REST-Retrieve	15	200 OK	316
1279392744056	REST-Retrieve	14	200 OK	316
1279392744059	REST-Retrieve	15	200 OK	316
1279392744062	REST-Retrieve	15	200 OK	316
1279392744066	REST-Retrieve	14	200 OK	316
1279392744069	REST-Retrieve	19	200 OK	316
1279392744071	REST-Retrieve	21	200 OK	316
1279392744076	REST-Retrieve	18	200 OK	316
1279392744079	REST-Retrieve	18	200 OK	316

บันทึกเวลา	บริการ	เวลาตอบสนอง (มิลลิวินาที)	HTTP Response	ขนาดข้อมูล (ไบต์)
1279392744082	REST-Retrieve	18	200 OK	316
1279392744090	REST-Retrieve	12	200 OK	316
1279392744093	REST-Retrieve	13	200 OK	316
1279392744095	REST-Retrieve	18	200 OK	316

ตัวอย่างผลการทดสอบบริการ UPDATE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST

ตารางที่ ข.7 ตัวอย่างผลการทดสอบบริการ UPDATE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST ในช่วงเวลาที่มีการทำงานสูงสุด

บันทึกเวลา	บริการ	เวลาตอบสนอง (มิลลิวินาที)	HTTP Response	ขนาดข้อมูล (ไบต์)
1279390972354	REST-Update	19	200	67
1279390972358	REST-Update	16	200	67
1279390972362	REST-Update	14	200	67
1279390972367	REST-Update	15	200	67
1279390972370	REST-Update	19	200	67
1279390972375	REST-Update	19	200	67
1279390972376	REST-Update	22	200	67
1279390972378	REST-Update	22	200	67
1279390972383	REST-Update	20	200	67
1279390972391	REST-Update	15	200	67
1279390972396	REST-Update	14	200	67
1279390972399	REST-Update	15	200	67
1279390972402	REST-Update	15	200	67
1279390972404	REST-Update	17	200	67
1279390972408	REST-Update	16	200	67
1279390972411	REST-Update	16	200	67
1279390972416	REST-Update	14	200	67

บันทึกเวลา	บริการ	เวลาตอบสนอง (มิลลิวินาที)	HTTP Response	ขนาดข้อมูล (ไบต์)
1279390972419	REST-Update	14	200	67
1279390972422	REST-Update	15	200	67
1279390972425	REST-Update	18	200	67
1279390972429	REST-Update	17	200	67
1279390972432	REST-Update	16	200	67
1279390972435	REST-Update	16	200	67
1279390972438	REST-Update	16	200	67
1279390972445	REST-Update	11	200	67
1279390972448	REST-Update	14	200	67
1279390972450	REST-Update	15	200	67
1279390972452	REST-Update	16	200	67
1279390972456	REST-Update	19	200	67
1279390972458	REST-Update	17	200	67
1279390972464	REST-Update	14	200	67
1279390972467	REST-Update	16	200	67
1279390972469	REST-Update	17	200	67
1279390972477	REST-Update	17	200	67
1279390972477	REST-Update	22	200	67
1279390972480	REST-Update	21	200	67
1279390972485	REST-Update	19	200	67
1279390972487	REST-Update	20	200	67
1279390972496	REST-Update	14	200	67
1279390972500	REST-Update	13	200	67
1279390972502	REST-Update	15	200	67
1279390972506	REST-Update	13	200	67
1279390972508	REST-Update	15	200	67
1279390972512	REST-Update	14	200	67
1279390972515	REST-Update	15	200	67
1279390972518	REST-Update	15	200	67
1279390972521	REST-Update	15	200	67

บันทึกเวลา	บริการ	เวลาตอบสนอง (มิลลิวินาที)	HTTP Response	ขนาดข้อมูล (ไบต์)
1279390972525	REST-Update	14	200	67
1279390972528	REST-Update	14	200	67
1279390972531	REST-Update	32	200	67
1279390972538	REST-Update	67	200	67
1279390972534	REST-Update	71	200	67
1279390972543	REST-Update	62	200	67
1279390972541	REST-Update	64	200	67
1279390972565	REST-Update	58	200	67
1279390972607	REST-Update	19	200	67
1279390972606	REST-Update	23	200	67
1279390972608	REST-Update	22	200	67
1279390972609	REST-Update	24	200	67
1279390972625	REST-Update	10	200	67
1279390972627	REST-Update	12	200	67
1279390972631	REST-Update	15	200	67
1279390972632	REST-Update	18	200	67
1279390972635	REST-Update	18	200	67
1279390972636	REST-Update	21	200	67
1279390972641	REST-Update	18	200	67
1279390972648	REST-Update	15	200	67
1279390972651	REST-Update	14	200	67
1279390972655	REST-Update	14	200	67
1279390972658	REST-Update	14	200	67
1279390972660	REST-Update	16	200	67
1279390972664	REST-Update	15	200	67
1279390972667	REST-Update	14	200	67
1279390972670	REST-Update	18	200	67
1279390972673	REST-Update	16	200	67
1279390972677	REST-Update	18	200	67
1279390972681	REST-Update	19	200	67

บันทึกเวลา	บริการ	เวลาตอบสนอง (มิลลิวินาที)	HTTP Response	ขนาดข้อมูล (ไบต์)
1279390972683	REST-Update	22	200	67
1279390972691	REST-Update	16	200	67
1279390972690	REST-Update	20	200	67
1279390972696	REST-Update	18	200	67
1279390972701	REST-Update	16	200	67
1279390972706	REST-Update	14	200	67
1279390972709	REST-Update	14	200	67
1279390972712	REST-Update	14	200	67
1279390972716	REST-Update	14	200	67
1279390972719	REST-Update	14	200	67
1279390972721	REST-Update	15	200	67
1279390972725	REST-Update	14	200	67
1279390972728	REST-Update	14	200	67
1279390972731	REST-Update	15	200	67
1279390972735	REST-Update	13	200	67
1279390972737	REST-Update	15	200	67
1279390972741	REST-Update	14	200	67
1279390972743	REST-Update	16	200	67
1279390972747	REST-Update	14	200	67
1279390972750	REST-Update	15	200	67
1279390972753	REST-Update	15	200	67
1279390972757	REST-Update	15	200	67
1279390972760	REST-Update	14	200	67

ตัวอย่างผลการทดสอบบริการ DELETE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST

ตารางที่ ข.8 ตัวอย่างผลการทดสอบบริการ DELETE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST ในช่วงเวลาที่มีการทำงานสูงสุด

บันทึกเวลา	บริการ	เวลาตอบสนอง (มิลลิวินาที)	HTTP Response	ขนาดข้อมูล (ไบต์)
1279391550658	REST-Delete	15	200	67
1279391550660	REST-Delete	16	200	67
1279391550664	REST-Delete	15	200	67
1279391550667	REST-Delete	15	200	67
1279391550670	REST-Delete	15	200	67
1279391550674	REST-Delete	16	200	67
1279391550677	REST-Delete	15	200	67
1279391550681	REST-Delete	14	200	67
1279391550683	REST-Delete	14	200	67
1279391550686	REST-Delete	15	200	67
1279391550691	REST-Delete	13	200	67
1279391550693	REST-Delete	14	200	67
1279391550697	REST-Delete	13	200	67
1279391550698	REST-Delete	17	200	67
1279391550703	REST-Delete	15	200	67
1279391550705	REST-Delete	18	200	67
1279391550711	REST-Delete	15	200	67
1279391550708	REST-Delete	22	200	67
1279391550716	REST-Delete	15	200	67
1279391550719	REST-Delete	15	200	67
1279391550724	REST-Delete	14	200	67
1279391550727	REST-Delete	17	200	67
1279391550731	REST-Delete	17	200	67
1279391550735	REST-Delete	16	200	67
1279391550732	REST-Delete	23	200	67
1279391550739	REST-Delete	18	200	67

บันทึกเวลา	บริการ	เวลาตอบสนอง (มิลลิวินาที)	HTTP Response	ขนาดข้อมูล (ไบต์)
1279391550745	REST-Delete	16	200	67
1279391550750	REST-Delete	14	200	67
1279391550752	REST-Delete	15	200	67
1279391550756	REST-Delete	14	200	67
1279391550758	REST-Delete	15	200	67
1279391550762	REST-Delete	16	200	67
1279391550765	REST-Delete	15	200	67
1279391550768	REST-Delete	16	200	67
1279391550771	REST-Delete	15	200	67
1279391550775	REST-Delete	14	200	67
1279391550779	REST-Delete	13	200	67
1279391550781	REST-Delete	13	200	67
1279391550785	REST-Delete	14	200	67
1279391550787	REST-Delete	15	200	67
1279391550790	REST-Delete	16	200	67
1279391550794	REST-Delete	16	200	67
1279391550796	REST-Delete	17	200	67
1279391550800	REST-Delete	16	200	67
1279391550803	REST-Delete	16	200	67
1279391550807	REST-Delete	16	200	67
1279391550811	REST-Delete	15	200	67
1279391550815	REST-Delete	14	200	67
1279391550817	REST-Delete	16	200	67
1279391550821	REST-Delete	12	200	67
1279391550824	REST-Delete	13	200	67
1279391550827	REST-Delete	14	200	67
1279391550830	REST-Delete	16	200	67
1279391550835	REST-Delete	16	200	67
1279391550835	REST-Delete	19	200	67
1279391550838	REST-Delete	21	200	67

บันทึกเวลา	บริการ	เวลาตอบสนอง (มิลลิวินาที)	HTTP Response	ขนาดข้อมูล (ไบต์)
1279391550842	REST-Delete	20	200	67
1279391550847	REST-Delete	18	200	67
1279391550852	REST-Delete	15	200	67
1279391550856	REST-Delete	15	200	67
1279391550860	REST-Delete	15	200	67
1279391550863	REST-Delete	14	200	67
1279391550866	REST-Delete	14	200	67
1279391550869	REST-Delete	15	200	67
1279391550872	REST-Delete	15	200	67
1279391550876	REST-Delete	14	200	67
1279391550878	REST-Delete	16	200	67
1279391550881	REST-Delete	16	200	67
1279391550886	REST-Delete	14	200	67
1279391550889	REST-Delete	14	200	67
1279391550892	REST-Delete	13	200	67
1279391550895	REST-Delete	14	200	67
1279391550898	REST-Delete	14	200	67
1279391550901	REST-Delete	14	200	67
1279391550904	REST-Delete	14	200	67
1279391550907	REST-Delete	14	200	67
1279391550910	REST-Delete	15	200	67
1279391550913	REST-Delete	14	200	67
1279391550916	REST-Delete	15	200	67
1279391550920	REST-Delete	15	200	67
1279391550923	REST-Delete	15	200	67
1279391550926	REST-Delete	14	200	67
1279391550928	REST-Delete	14	200	67
1279391550932	REST-Delete	16	200	67
1279391550936	REST-Delete	17	200	67
1279391550939	REST-Delete	19	200	67

บันทึกเวลา	บริการ	เวลาตอบสนอง (มิลลิวินาที)	HTTP Response	ขนาดข้อมูล (ไบต์)
1279391550941	REST-Delete	20	200	67
1279391550944	REST-Delete	20	200	67
1279391550949	REST-Delete	18	200	67
1279391550954	REST-Delete	16	200	67
1279391550959	REST-Delete	14	200	67
1279391550962	REST-Delete	15	200	67
1279391550965	REST-Delete	15	200	67
1279391550968	REST-Delete	16	200	67
1279391550971	REST-Delete	16	200	67
1279391550975	REST-Delete	14	200	67
1279391550978	REST-Delete	15	200	67
1279391550982	REST-Delete	13	200	67
1279391550986	REST-Delete	13	200	67
1279391550988	REST-Delete	15	200	67

ภาคผนวก ค.

ผลงานทางวิชาการที่ได้รับการตีพิมพ์

ผลงานทางวิชาการ

เรื่อง การพิจารณาประสิทธิภาพการทำงานของเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST
(Consideration the performance of Web Service using REST architecture)

ได้รับการตีพิมพ์ในการประชุม

การประชุมวิชาการระดับประเทศด้านเทคโนโลยีสารสนเทศ ครั้งที่ 3
(The 3rd National Conference on Information Technology, NCIT2010)

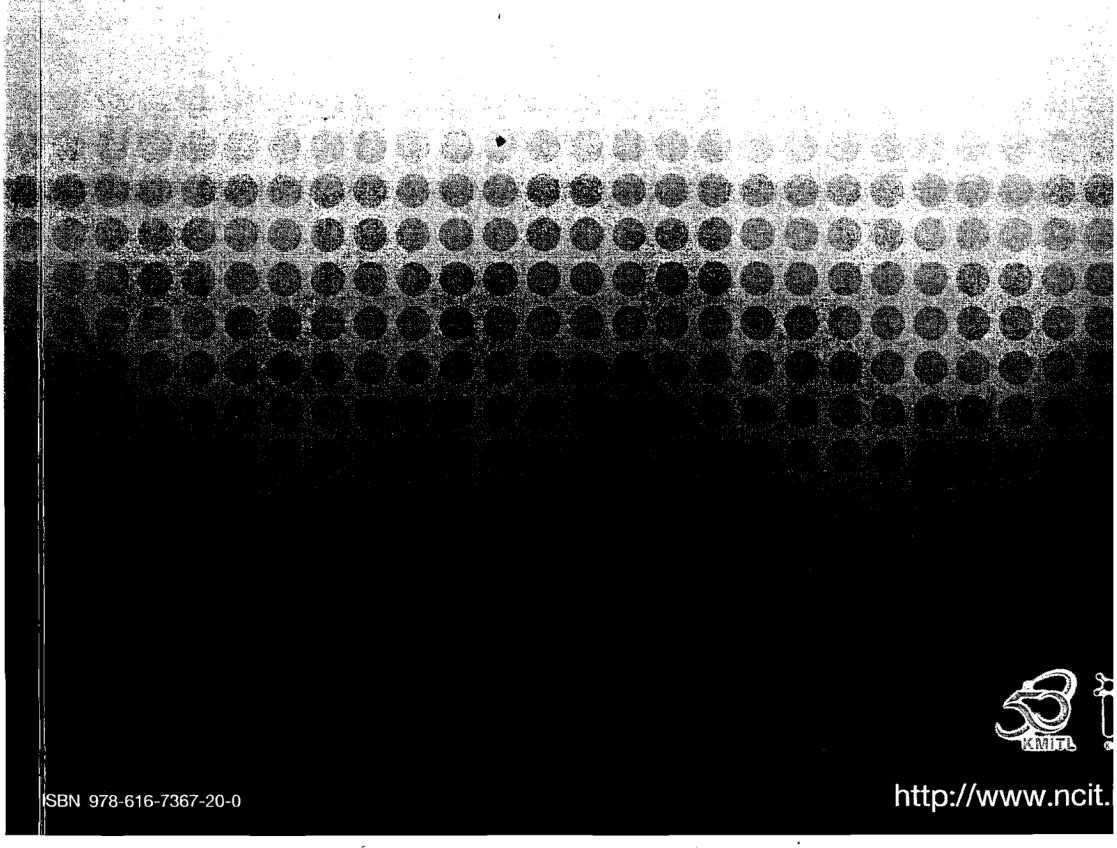
วันที่ 28-29 ตุลาคม พ.ศ. 2553
โรงแรม ดิเอมเมอรัลด์ กรุงเทพมหานคร ประเทศไทย

NCIT

The 3rd National Conference on Information Technology: NCIT 2010

"IT Innovation for Global Awareness"

28-29 October 2010, Bangkok, Thailand



ISBN 978-616-7367-20-0

http://www.ncit.

Sponsored by

CNET – 2: Networking

เวลา 10:15-11:55 น.

ห้องประชุม: เพชรชมพู 2

INSYS & MM – 2: Soft Eng & HID

เวลา 10:15-11:55 น.

ห้องประชุม: เพชรชมพู 3

การพิจารณาประสิทธิภาพการทำงานของเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST

พชรวร บุญชู แตะ มยุรี เลิศเวชกุล

สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง กรุงเทพฯ

เว็บเซอร์วิสเป็นเทคโนโลยีที่กำลังได้รับความนิยมสำหรับการพัฒนาเว็บแอพพลิเคชันเพื่อให้บริการแลกเปลี่ยนข้อมูลผ่านระบบเครือข่ายอินเทอร์เน็ต แต่เนื่องด้วยในปัจจุบันยังคงมีการใช้งานอินเทอร์เน็ตบนระบบเครือข่ายที่มีความเร็วจำกัดอยู่เป็นจำนวนมาก โดยเฉพาะอย่างยิ่งการใช้ GPRS/EDGE เพื่อเชื่อมต่ออินเทอร์เน็ตผ่านโทรศัพท์มือถือที่มีคุณภาพการเชื่อมต่อที่ต่ำแต่มีค่าใช้จ่ายที่ค่อนข้างสูง ดังนั้นการพิจารณารูปแบบของการให้บริการเว็บเซอร์วิสที่เหมาะสมจึงเป็นสิ่งสำคัญ ผู้วิจัยจึงได้ศึกษาถึงข้อดีของการนำทฤษฎีสถาปัตยกรรม REST มาใช้ในการพัฒนาเว็บเซอร์วิส และทำการทดสอบเปรียบเทียบประสิทธิภาพของการร้องขอบริการจากเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST และสถาปัตยกรรมแบบ SOAP โดยผลการทดสอบแสดงให้เห็นว่าการให้บริการข้อมูลด้วย เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST นั้นสามารถช่วยลดระยะเวลาที่ใช้ในการร้องขอบริการจากเว็บเซอร์วิสและเพิ่มประสิทธิภาพในการให้บริการของแม่ข่ายเว็บเซอร์วิสได้ถึงประมาณ 30 เปอร์เซนต์ อีกทั้งยังเป็นการลดปริมาณข้อมูลที่ต้องรับส่งระหว่างเครือข่ายได้มากถึงประมาณ 90 เปอร์เซนต์ เมื่อเทียบกับเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP

การพิจารณาประสิทธิภาพการทำงานของเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST

พชรวร บุญชู และ ผศ.มยุรี เลิศเวชกุล

สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง กรุงเทพฯ
Emails: s8061054@kmitl.ac.th, klmayure@kmitl.ac.th

บทคัดย่อ

เว็บเซอร์วิสเป็นเทคโนโลยีที่กำลังได้รับความนิยมสำหรับการพัฒนาเว็บแอพพลิเคชันเพื่อให้บริการแลกเปลี่ยนข้อมูลผ่านระบบเครือข่ายอินเทอร์เน็ต แต่เนื่องด้วยในปัจจุบันยังคงมีการใช้งานอินเทอร์เน็ตบนระบนเครือข่ายที่มีความเร็วจำกัดอยู่เป็นจำนวนมาก โดยเฉพาะอย่างยิ่งการใช้ GPRS/EDGE เพื่อเชื่อมต่ออินเทอร์เน็ตผ่านโทรศัพท์มือถือที่มีคุณภาพการเชื่อมต่อที่ต่ำแต่มีค่าใช้จ่ายที่ค่อนข้างสูง ดังนั้นการพิจารณารูปแบบของการให้บริการเว็บเซอร์วิสที่เหมาะสมจึงเป็นสิ่งสำคัญ ผู้วิจัยจึงได้ศึกษาถึงข้อดีของการนำทฤษฎีสถาปัตยกรรม REST มาใช้ในการพัฒนาเว็บเซอร์วิส และทำการทดสอบเปรียบเทียบประสิทธิภาพของการร้องขอบริการจากเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST และสถาปัตยกรรมแบบ SOAP โดยผลการทดสอบแสดงให้เห็นว่าการให้บริการข้อมูลด้วยเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST นั้นสามารถช่วยลดระยะเวลาที่ใช้ในการร้องขอบริการจากเว็บเซอร์วิสและเพิ่มประสิทธิภาพในการให้บริการของแม่ข่ายเว็บเซอร์วิสได้ถึงประมาณ 30 เปอร์เซนต์ อีกทั้งยังเป็นการลดปริมาณข้อมูลที่ต้องรับส่งระหว่างเครือข่ายได้มากถึงประมาณ 90 เปอร์เซนต์ เมื่อเทียบกับเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP

คำสำคัญ– REST; SOAP; เว็บเซอร์วิส; ประสิทธิภาพ

1. บทนำ

แนวโน้มสำหรับการให้บริการเว็บ (World Wide Web) [1] ในปัจจุบันได้มีการให้บริการเนื้อหาของเว็บที่เป็นแบบไดนามิคเพิ่มมากขึ้นอย่างต่อเนื่อง และเทคโนโลยีที่เป็นที่นิยมสำหรับการพัฒนาเว็บแอพพลิเคชันในลักษณะดังกล่าวเรียกว่า เว็บเซอร์วิส (Web service) โดยที่เว็บเซอร์วิสนั้นคือเว็บแอพพลิเคชันที่เป็นแม่ข่ายในการให้บริการเนื้อหาที่ผู้ใช้งานสามารถเข้าถึงได้โดยโปรโตคอลสื่อสารอย่าง เช่น Hypertext transfer protocol (HTTP) [2] เป็นต้น

เนื่องจากในปัจจุบันยังคงมีการใช้งานอินเทอร์เน็ตที่มีข้อจำกัดในเรื่องความเร็วในการใช้บริการและปริมาณการรับส่งข้อมูลที่จำกัดอยู่เป็นจำนวนมาก โดยเฉพาะอย่างยิ่งการใช้ GPRS หรือ EDGE เพื่อเชื่อมต่ออินเทอร์เน็ตผ่านโทรศัพท์มือถือที่มีคุณภาพในการเชื่อมต่อที่ต่ำแต่มีค่าใช้จ่ายที่ค่อนข้างสูง ด้วยเหตุนี้เองทำให้เกิดการใช้งานเว็บเซอร์วิสจึงถูกใช้งานอยู่บนการเชื่อมต่อที่มีข้อจำกัด ทำให้การร้องขอใช้บริการเว็บเซอร์วิสอาจจะใช้เวลานานเกินไป หรือ อาจจะไม่ได้รับการให้บริการจากแม่ข่ายเว็บเซอร์วิสเลยในบางครั้ง

ส่วนสำคัญในงานวิจัยฉบับนี้คือการนำทฤษฎีสถาปัตยกรรม REST มาใช้ในการพัฒนาเว็บเซอร์วิสเพื่อเป็นแนวทางในแก้ไขปัญหาการใช้งานเว็บเซอร์วิสบนการเชื่อมต่อที่มีคุณภาพต่ำ และทำการทดสอบประสิทธิภาพการทำงานของ เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST ดังกล่าวเมื่อเปรียบเทียบกับการร้องขอบริการผ่าน เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP

2. เว็บเซอร์วิส

เว็บเซอร์วิสเป็นซอฟต์แวร์ที่ถูกพัฒนาขึ้นเพื่อให้บริการแลกเปลี่ยนข้อมูลระหว่างเครื่องคอมพิวเตอร์ผ่านระบบเครือข่ายอินเทอร์เน็ตด้วยการร้องขอบริการเว็บเซอร์วิสในรูปแบบของ Simple Object Access Protocol (SOAP) [3] หรือ ส่งการร้องขอบริการเว็บเซอร์วิสตามทฤษฎีของสถาปัตยกรรม Representational State Transfer (REST) [1] เป็นต้น ทั้งนี้จะอาศัยรูปแบบภาษามาตรฐานที่เรียกว่า Extensible Markup Language (XML) ในการส่งเนื้อหาของการร้องขอบริการนั้น ทำให้เครื่องคอมพิวเตอร์ที่เรียกใช้งานเว็บเซอร์วิสสามารถสื่อสารกันได้ถึงแม้ว่าจะมีแพลตฟอร์มที่แตกต่าง หรือถูกพัฒนาด้วยภาษาโปรแกรมที่ต่างกันก็ตาม

2.1 สถาปัตยกรรม SOAP

Simple Object Access Protocol (SOAP) คือโปรโตคอล หรือวิธีการมาตรฐานที่ใช้ในการสื่อสารกันระหว่างเว็บเซอร์วิสโดยอาศัยการส่งข้อมูลผ่านโปรโตคอลสื่อสารอย่างเช่น HTTP หรือ SMTP เป็นต้น แต่โดยทั่วไปแล้วจะใช้การส่งบน HTTP เนื่องจากเป็นโปรโตคอลที่แพร่หลายและมีการใช้งานอยู่แล้วบนอินเทอร์เน็ต โดยเมื่อพัฒนาโปรแกรมด้วยรูปแบบของ SOAP ในภาษาโปรแกรมที่ใช้ และใช้งานร่วมกับ Web Services Description Language (WSDL) [7] จากนั้น SOAP ก็จะสร้างข้อความ SOAP (SOAP Message) [6] เพื่อติดต่อกับแอพพลิเคชันปลายทางให้โดยอัตโนมัติ

ข้อความ SOAP (SOAP Envelope) ที่บรรจุข้อมูลไว้ภายในแบ่งออกได้เป็น 2 ส่วน คือ ส่วนหัว SOAP (SOAP Header) อันประกอบด้วยคำอธิบาย หรือ รายละเอียดต่างๆ และส่วนที่สอง คือ เนื้อหา SOAP (SOAP Body) ซึ่งประกอบด้วยเนื้อหาของการร้องขอหรือการตอบกลับของเว็บเซอร์วิสในรูปแบบของภาษามาตรฐาน XML

2.2 สถาปัตยกรรม REST

Representational State Transfer (REST) เป็นรูปแบบทฤษฎีของสถาปัตยกรรมทางซอฟต์แวร์สำหรับการนำเสนอสื่อต่างๆ เช่น เอกสาร เอชทีเอ็มแอล โดยที่ REST เองนั้นไม่ได้ถูกจำกัดให้ใช้งานกับโปรโตคอลใดเป็นพิเศษ แต่ส่วนใหญ่ก็จะใช้งานร่วมกับ HTTP ที่มีลักษณะการทำงานแบบการร้องขอและตอบรับ (Request-Response) และตัวเนื้อหาของสื่อที่จะส่งไปนั้นสามารถเป็นข้อมูลชนิดใดก็ได้ตามที่ต้องการ เช่น HTML, Plain Text, XML หรือ JSON เป็นต้น

เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST [4] (ในบางครั้งเรียกว่า RESTful) จะเน้นไปที่ผลลัพธ์ของการให้บริการ โดยจะต้องสามารถกำหนดที่อยู่ของทรัพยากรในระบบให้สามารถเข้าถึงได้ด้วยการกำหนดคีย์หลักที่ไม่ซ้ำกันให้กับทรัพยากรทั้งหมด แล้วระบุที่อยู่ผ่านทาง URL [5] เท่านั้น ดังนั้นทุกครั้งที่ต้องการร้องขอใช้บริการหรืออ้างอิงผลลัพธ์ของการให้บริการ ก็เพียงแค่ระบุ URL ร่วมกับกระบวนการทำงานของ HTTP (HTTP Method) เท่านั้น

2.3 ทฤษฎีสถาปัตยกรรม REST

พื้นฐานของการออกแบบสถาปัตยกรรม REST นั้นเน้นที่การใช้งานกระบวนการทำงานของ HTTP (POST, GET, PUT และ DELETE) ให้ถูกต้องและเหมาะสม, การไม่จัดเก็บสถานะการทำงาน (Stateless), การใช้รูปแบบของ URL ที่มีลักษณะคล้ายกับโครงสร้างของไดเรคทอรี่ และ การนำสามารถเลือกใช้ข้อมูลมาตรฐานชนิดในก็ได้ในการแสดงเนื้อหาของการร้องขอบริการและการตอบกลับ ซึ่งมีรายละเอียดดังต่อไปนี้

2.3.1 การกำหนดรูปแบบของ URL ที่คล้ายกับโครงสร้างไดเรคทอรี่

การใช้ URL เป็นตัวชี้ไปที่วัตถุ ทรัพยากร หรือ บริการต่างๆ โดยสามารถกำหนดกระบวนการของ HTTP ไปพร้อมกับการร้องขอเพื่อเป็นการแสดงว่าต้องการให้แม่ข่ายเว็บเซอร์วิสดำเนินการอย่างไรกับทรัพยากรดังกล่าว

2.3.2 การไม่จัดเก็บสถานะการทำงาน (Stateless)

เครื่องแม่ข่ายจะไม่เก็บข้อมูลเซสชันของการใช้บริการของผู้ใช้ไว้ โดยเครื่องแม่ข่ายจะจัดการเพียงสถานะของทรัพยากรที่ดูแลอยู่เท่านั้น ซึ่งในกรณีที่แอพพลิเคชันมีความต้องการที่จะเก็บข้อมูลเซสชันของการใช้บริการของผู้ใช้นั้น สามารถทำได้ด้วยการเก็บไว้ที่เครื่องลูกข่ายเอง และส่งมาที่เครื่องแม่ข่ายพร้อมกับการร้องขอบริการเมื่อต้องการใช้งาน ซึ่งลักษณะการทำงานเช่นนี้จะช่วยทำให้ระบบสามารถขยายการให้บริการในอนาคตได้ง่าย (Scalable)

2.3.3 กำหนดวิธีการใช้งานกระบวนการทำงานของ HTTP อย่างชัดเจน

การใช้งานกระบวนการต่างๆของ HTTP นั้นจะต้องเป็นไปตามรายละเอียดที่แสดงในตาราง 1.

ตาราง 1. มาตรฐานการใช้งานกระบวนการของ HTTP

กระบวนการ	รายละเอียด
POST	เป็นกระบวนการที่ใช้สั่งให้เครื่องแม่ข่ายทำการเก็บข้อมูล ซึ่งสามารถเลือกได้ว่าจะส่งหรือไม่ส่งข้อมูลไปพร้อมกับการร้องขอก็ได้
GET	เป็นกระบวนการทำงานเพื่อเรียกดูข้อมูลที่ต้องการจากเครื่องแม่ข่าย
PUT	เป็นกระบวนการที่ใช้สั่งให้เครื่องแม่ข่ายทำการสร้างหรือแก้ไขข้อมูลด้วยข้อมูลที่ส่งมาพร้อมกับการร้องขอ
DELETE	เป็นกระบวนการที่ใช้สำหรับสั่งให้เครื่องแม่ข่ายทำการลบข้อมูล

โดยกระบวนการ GET, PUT และ DETETE จะต้องอ้างอิงตำแหน่งของทรัพยากรด้วยค่าของคีย์หลัก

2.3.4 สามารถใช้กับรูปแบบของข้อมูลมาตรฐานชนิดใดก็ได้

ส่วนแสดงเนื้อหาของการร้องขอและการตอบกลับนั้นสามารถเลือกใช้ข้อมูลชนิดใดก็ได้ที่สามารถอ้างอิง MIME-type และ Content-Type ได้ เช่น HTML, Plain Text หรือ JSON เป็นต้น

3. การทดสอบประสิทธิภาพของการให้บริการเว็บเซอร์วิสด้วยรูปแบบของ SOAP และ REST

สำหรับในข้อหัวนี้จะกล่าวถึงการทดสอบเปรียบเทียบระหว่างประสิทธิภาพของการให้บริการเว็บเซอร์วิสด้วยรูปแบบตามมาตรฐานของ SOAP และรูปแบบตามแนวทางทฤษฎีของ REST ที่ได้กล่าวไว้ในหัวข้อที่ 2. โดยจะแสดงให้เห็นถึงวิธีการทดสอบและผลการทดสอบที่ได้จากการทดสอบ

3.1 ระบบที่ใช้ในการทดสอบ

ผู้วิจัยได้ทำการทดสอบประสิทธิภาพของเว็บเซอร์วิสที่ใช้สถาปัยกรรม REST โดยเปรียบเทียบกับเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP โดยทดสอบให้เครื่องลูกข่ายเชื่อมต่อกับเครื่องแม่ข่ายที่ให้บริการเว็บเซอร์วิสตามสถาปัตยกรรมทั้งสองแบบผ่านเราท์เตอร์

โดยเครื่องแม่ข่ายใช้คอมพิวเตอร์ที่มีหน่วยประมวลผลกลางเป็น Intel® Core™2 Duo E6300 1.86 GHz. หน่วยความจำ 2GB ทำงานบนระบบปฏิบัติการ Windows XP Professional Service Pack 3 และใช้ซอฟต์แวร์ Apache Web Server เวอร์ชัน 2.2.8 ที่รองรับภาษา PHP เวอร์ชัน 5.2.6 เป็นซอฟต์แวร์ระบบแม่ข่ายเว็บ โดยใช้ระบบจัดการฐานข้อมูล MySQL เวอร์ชัน 5.0.51b

เครื่องลูกข่ายใช้คอมพิวเตอร์ที่มีหน่วยประมวลผลกลางเป็น Intel® Core™2 Duo T9300 2.50 GHz. หน่วยความจำ 2GB ทำงานบนระบบปฏิบัติการ Windows XP Professional Service Pack 3 และใช้ซอฟต์แวร์ Apache JMeter [8] เวอร์ชัน 2.3.4 ในการทดสอบและบันทึกผลการทดสอบ

3.2 แม่ข่ายเว็บเซอร์วิส

การทำงานของเว็บเซอร์วิสแบ่งออกเป็น 4 บริการ คือ CREATE, RETRIEVE, UPDATE และ DELETE ในที่นี้ใช้การจำลองการสร้างและเก็บข้อมูลสมาชิกซึ่งประกอบไปด้วย หมายเลขโทรศัพท์, เลขประจำตัวประชาชน, ชื่อ, นามสกุล, ชื่อประเทศ, รหัสไปรษณีย์, อีเมล์ และวันที่ในการทำรายการ โดยการใช้หมายเลขโทรศัพท์เป็นคีย์หลัก (Primary Key :PK) ของตารางดังกล่าว

บริการ CREATE ใช้ในการสร้างข้อมูลสมาชิก บริการ UPDATE ใช้แก้ไขข้อมูลสมาชิกที่มีอยู่แล้ว บริการ DELETE ใช้ในการลบข้อมูลสมาชิก โดยที่บริการทั้ง 3 นี้จะส่งผลลัพธ์ของการสร้าง การแก้ไข และการลบกลับไปยังเครื่องลูกข่าย ส่วนบริการ RETRIEVE เป็นบริการในการเรียกดูข้อมูลสมาชิก ซึ่งในกรณีที่ไม่พบข้อมูลสมาชิกจะส่งรหัสที่บ่งบอกว่าไม่พบข้อมูลสมาชิกกลับไปยังเครื่องลูกข่ายเป็นผลลัพธ์

เว็บเซอร์วิสที่ใช้ทำการทดสอบนั้นพัฒนาด้วยภาษา PHP เวอร์ชัน 5 โดยใช้ฟังก์ชั่นมาตรฐานของภาษา PHP เองทั้งหมด โดยได้ทำการพัฒนาเว็บเซอร์วิสออกเป็นสองชุด ซึ่งเป็นไปตามสถาปัตยกรรม SOAP และสถาปัตยกรรม REST สำหรับเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP จะเรียกใช้งานคลาส SoapServer พร้อมอ้างอิงไฟล์ WSDL ที่ระบุถึงบริการจากบริการ 4 ชนิด (CREATE, RETRIEVE, UPDATE และ DELETE) ส่วนเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST นั้นใช้การร้องขอตามมาตรฐานของ HTTP ในภาษา PHP และใช้คลาส SimpleXMLElement ในการอ่านเนื้อหาของการร้องขอ โดยมีบริการ 4 บริการเช่นเดียวกันกับเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP

3.3 แม่ข่ายเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST

เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST ที่ใช้ทดสอบนั้น ผู้วิจัยได้ทำการพัฒนาขึ้นด้วยการอ้างอิงตามพื้นฐาน 4 ข้อของทฤษฎีสถาปัตยกรรม REST ดังที่ได้กล่าวมาแล้วข้างต้น โดยการจับคู่บริการของเว็บเซอร์วิสตามสถาปัตยกรรม REST กับกระบวนการทำงานของ HTTP เป็นไปดังที่แสดงในตาราง 2.

ตาราง 2. การจับคู่บริการเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST กับกระบวนการ HTTP

บริการ	กระบวนการ HTTP	รายละเอียดการทำงาน
CREATE	POST	สร้างข้อมูลสมาชิกในระบบ
RETRIEVE	GET	เรียกดูข้อมูลสมาชิกในระบบ
UPDATE	PUT	แก้ไขข้อมูลสมาชิกในระบบ
DELETE	DELETE	ลบข้อมูลสมาชิกในระบบ

โดยทั้งนี้การออกแบบเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST ในการทดสอบนั้น สถานะการทำงาน ค่าข้อมูลควบคุมการทำงานเซสชันของผู้ใช้งาน และพารามิเตอร์ทุกตัวจะถูกจัดการและส่งมาจากเครื่องลูกข่ายเท่านั้นโดยการอ้างอิงทรัพยากรและบริการของเว็บเซอร์วิสจะเป็นไปตามโครงสร้างของ URL ที่กำหนดดังในรูปที่ 1. ซึ่งมีรูปแบบที่คล้ายกับโครงสร้างไดเรคทอรี่ ซึ่งรูปที่ 2. เป็นการแสดงตัวอย่างการเข้าถึงเครื่องแม่ข่ายที่ชื่อ “TESTSERV” โดยเรียกบริการชื่อ “FREECONTENT” ที่ทำงานที่พอร์ตหมายเลข “88” เพื่ออ้างอิงถึงข้อมูลของหมายเลขโทรศัพท์ “66810987654”

```
HTTP : // IP : PORT / SERVICE / RESOURCE
```

รูปที่ 1. การกำหนดโครงสร้างของ URL ของเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST

```
HTTP : // TESTSERV : 88 / FREECONTENT / 66810987654
```

รูปที่ 2. ตัวอย่างการกำหนด URL ตามโครงสร้าง URL ที่กำหนด

และสุดท้ายเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST ที่พัฒนาขึ้นมาเพื่อทดสอบนั้น ผู้วิจัยได้เลือกใช้ภาษามาตรฐาน XML ในการแสดงส่วนเนื้อหาของการร้องขอและการตอบกลับ โดยอ้างอิงจากเนื้อหาของการร้องขอและการตอบกลับจาก เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP ให้ได้ค่าพารามิเตอร์ที่ครบถ้วนเช่นเดียวกัน เพื่อให้ได้ผลลัพธ์จากการร้องขอบริการของเว็บเซอร์วิสทั้งสองสถาปัตยกรรมที่เหมือนกัน ดังตัวอย่างของการร้องขอและการตอบกลับของเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST ในรูปที่ 3. และ 4. และตัวอย่างของการร้องขอและตอบกลับของเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP ที่แสดงในรูป 5. และ 6.

```
<?xml version="1.0" encoding="UTF-8"?>
<subscriber>
   <msisdn>66810987654</msisdn>
   <id>3100001000990</id>
   <title>mr</title>
   <firstname>REST</firstname>
   <lastname>TEST</lastname>
   <country> Thailand </country>
   <postcode>10220</postcode>
   <email>66810987654@rest.com</email>
   <active_date>2010/07/01 08:30:00</active_date>
</subscriber>
```

รูปที่ 3. ตัวอย่างเนื้อหาการร้องขอบริการ CREATE
ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST

```
<?xml version="1.0" encoding="UTF-8"?>
<subscriber>
   <result>0</result>
</subscriber>
```

รูปที่ 4. ตัวอย่างเนื้อหาการตอบกลับบริการ CREATE
ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:ns1="urn:xmethods-delayed-quotes"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
      <ns1:create>
         <msisdn xsi:type="xsd:string">66810987654</msisdn>
         <id xsi:type="xsd:string">3100001000990</id>
         <title xsi:type="xsd:string">mr</title>
         <firstname xsi:type="xsd:string">SOAP</firstname>
         <lastname xsi:type="xsd:string">TEST</lastname>
         <country xsi:type="xsd:string">Thailand</country>
         <postcode xsi:type="xsd:string">10220</postcode>
         <email xsi:type="xsd:string">66810987654@soap.com</email>
         <active_date xsi:type="xsd:datetime">2010-07-
01T08:30:00</active_date>
      </ns1:create>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

รูปที่ 5. ตัวอย่างเนื้อหาการร้องขอบริการ CREATE
ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:ns1="urn:xmethods-delayed-quotes"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
      <ns1:createResponse>
         <result xsi:type="xsd:string">0</result>
      </ns1:createResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

รูปที่ 6. ตัวอย่างเนื้อหาการตอบกลับบริการ CREATE
ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP

3.4 วิธีการทดสอบ

ผู้วิจัยได้ทำการทดสอบโดยใช้ซอฟต์แวร์ Apache JMeter เพื่อสร้างการร้องขอบริการไปยังแม่ข่ายเว็บเซอร์วิสทั้ง 2 สถาปัตยกรรม แล้วเก็บผลลัพธ์ที่แม่ข่ายเว็บเซอร์วิสตอบกลับมาไว้ในเครื่องลูกข่าย โดยทำการแยกทดสอบทีละสถาปัตยกรรม และทดสอบรอบละหนึ่งบริการเท่านั้น ด้วยการร้องขอบริการจำนวน 25,000 ครั้งต่อการทดสอบ 1 รอบ และทำการทดสอบซ้ำทั้งหมด 3 รอบสำหรับหนึ่งบริการของเว็บเซอร์วิสหนึ่งสถาปัตยกรรม

รูปแบบของการร้องขอทั้งหมดนั้นจะใช้การเชื่อมต่อ HTTP แบบทรานแซคชัน โดยไม่มีการเก็บแคชของ HTTP และมีส่วนหัวของการเชื่อมต่อ HTTP (HTTP header) ขนาด 160ไบต์ โดยประมาณใกล้เคียงกันทั้งสำหรับการร้องขอไปยังแม่ข่าย เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP และแม่ข่าย เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST ซึ่งส่วนหัวนี้จะไม่ถูกนำมาใช้ในการวิเคราะห์ผลการทดสอบ โดยการทดสอบจะพิจารณาเฉพาะขนาดของเนื้อหาที่อยู่ภายในเท่านั้น

4. ผลการทดสอบ

4.1 ผลการทดสอบเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP

ถ้าพิจารณาจากเรื่องขนาดของเนื้อหา (Content-Length) ที่ส่งการร้องขอบริการออกไปนั้นสามารถแบ่งได้เป็นสองกลุ่มคือกลุ่มของบริการ CREATE และบริการ UPDATE ซึ่งสองบริการนี้จะมีเนื้อหาที่ส่งไปขนาดประมาณ 950 ไบต์เนื่องจากต้องส่งข้อมูลสมาชิกไปด้วย เพื่อทำการสร้างหรือแก้ไข โดยที่การตอบกลับจากแม่ข่ายเว็บเซอร์วิสจะส่งผลลัพธ์ของการสร้างหรือแก้ไขกลับมาด้วยเนื้อหาขนาดประมาณ 515 ไบต์ ส่วนบริการ RETRIEVE และบริการ DELETE จะเป็นบริการที่มีการส่งการร้องขอที่มีเนื้อหาในขนาดที่เล็กกว่าเพราะมีการส่งเพียงแค่คีย์หลักเท่านั้น ส่วนการตอบกลับจากแม่ข่ายเว็บเซอร์วิสนั้นถ้าเป็นบริการ DELETE จะส่งผลลัพธ์ของการลบกลับมาด้วยเนื้อหาขนาดประมาณ 515 ไบต์ และบริการ RETRIEVE จะส่งข้อมูลสมาชิกกลับมาเป็นผลลัพธ์จึงทำให้มีขนาดเนื้อหาตอบกลับที่ค่อนข้างใหญ่ประมาณ 1,460 ไบต์

เวลาที่ใช้ในการทำรายการ (Latency) แต่ละรายการเฉลี่ยโดยรวมจากการทดสอบทั้งสามรอบนั้นพบว่า เวลาในการตอบสนองบริการแต่ละครั้งนั้นไม่แตกต่างกันมากนัก โดยจะใช้เวลาทั้งสิ้น 21-24 มิลลิวินาที ส่วนปริมาณงานที่ทำในช่วงเวลาหนึ่งวินาที (Transactions per second) นั้นอยู่ที่ประมาณ 40-45 ทรานแซคชันต่อวินาที ดังจะแสดงค่าเฉลี่ยจากการทดสอบทั้งหมดได้ดังตาราง 3.

ตาราง 3. แสดงค่าเฉลี่ยจากการทดสอบเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP

SOAP	ขนาดเนื้อหา		เฉลี่ยจากการทดสอบทั้ง 3 รอบ	
	การร้องขอ (ไบต์)	การตอบกลับ (ไบต์)	เวลาเฉลี่ย (มิลลิวินาที)	ปริมาณงานที่ทำได้ (ทรานแซคชันต่อวินาที)
CREATE	953	515	24.43	40.95
RETRIEVE	536	1460	23.86	41.92
UPDATE	951	515	23.49	42.58
DELETE	532	515	22.42	44.61

4.2 ผลการทดสอบเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST

จากผลการทดสอบของเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST นั้นแสดงให้เห็นได้ชัดเจนว่าขนาดของเนื้อหา (Content-Length) ที่ส่งการร้องขอและที่แม่ข่ายเว็บเซอร์วิสตอบกลับมาจะมีขนาดที่เล็กกว่าของ เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP อย่างเห็นได้ชัดโดยเฉพาะในกรณีการร้องขอบริการ RETRIEVE และบริการ DELETE นั้นไม่ต้องส่งเนื้อหาใดๆ ไปด้วย เนื่องจากทั้งสองบริการนี้ใช้เพียงการระบุค่าของคีย์หลักซึ่งในที่นี้คือหมายเลขโทรศัพท์ไว้ใน URL เท่านั้น ส่วนขนาดเนื้อหาที่ตอบกลับจากแม่ข่ายเว็บเซอร์วิสในกรณีการใช้บริการ DELETE จะเป็นรหัสแสดงผลลัพธ์ของการลบขนาดประมาณ 67 ไบต์ เนื้อหาของการตอบรับจากบริการ RETRIEVE นั้นจะมีขนาดใหญ่ที่สุดเมื่อเทียบจากทั้ง 4

บริการ เนื่องจากเป็นการตอบกลับข้อมูลสมาชิกด้วยข้อความขนาดประมาณ 316 ไบต์ และกรณีบริการ CREATE และบริการ UPDATE จะมีการส่งการร้องขอที่มีเนื้อหาที่มีข้อมูลสมาชิกขนาดประมาณ 319 ไบต์ และแม่ข่ายเว็บเซอร์วิสจะตอบกลับด้วยผลลัพธ์ของการสร้างและแก้ไขกลับมาด้วยเนื้อหาขนาดประมาณ 67 ไบต์

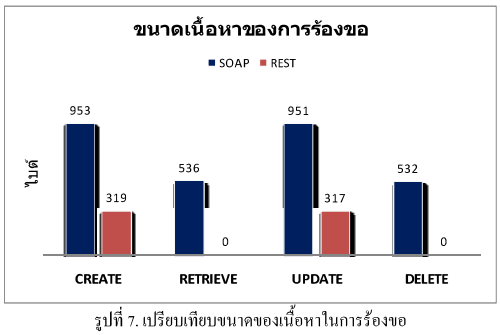
เวลาที่ใช้ในการทำรายการ (Latency) แต่ละรายการเฉลี่ยโดยรวมจากการทดสอบทั้งสามรอบนั้นพบว่าเวลาในการตอบสนองบริการแต่ละครั้งนั้นใช้เวลาอยู่ในช่วง 15-17 มิลลิวินาที และค่าปริมาณงานที่ทำในช่วงเวลาหนึ่งวินาที (Transactions per second) นั้นอยู่ที่ประมาณ 56-62 ทรานแซคชันต่อวินาที โดยบริการ RETRIEVE จะมีค่าปริมาณงานที่ทำในช่วงเวลาหนึ่งวินาทีน้อยที่สุด ดังจะเห็นได้จากตาราง 4.

ตาราง 4. แสดงค่าเฉลี่ยจากการทดสอบเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST

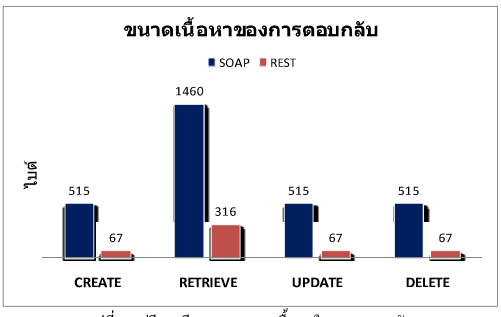
REST	ขนาดเนื้อหา		เฉลี่ยจากการทดสอบทั้ง 3 รอบ	
	การร้องขอ (ไบต์)	การตอบกลับ (ไบต์)	เวลาเฉลี่ย (มิลลิวินาที)	ปริมาณงานที่ทำได้ (ทรานแซคชันต่อวินาที)
CREATE	319	67	16.22	61.68
RETRIEVE	0	316	17.35	57.64
UPDATE	317	67	16.32	62.29
DELETE	0	67	15.91	62.83

จากผลการทดสอบสามารถทำการเปรียบเทียบขนาดของเนื้อหาในการร้องขอบริการจากเว็บเซอร์วิสและขนาดของเนื้อหาในการตอบกลับจากแม่ข่ายเว็บเซอร์วิสตามรูปที่ 7. และ 8. ซึ่งเห็นได้ว่าในการร้องขอบริการ CREATE และบริการ UPDATE จากแม่ข่ายเว็บเซอร์วิสนั้นสามารถลดขนาดของข้อมูลที่ต้องส่งไปได้ประมาณ 66% และการร้องขอบริการ RETRIEVE และบริการ DELETE นั้นสามารถลดขนาดของข้อมูลที่ต้องส่งไปได้ทั้งหมด 100% เพราะด้วยรูปแบบของสถาปัตยกรรมเว็บเซอร์วิสแบบ REST นั้นมีการอ้างอิงคีย์หลักอยู่ใน URL แล้ว จึงทำให้รูปแบบของบริการดังกล่าวไม่ต้องส่งเนื้อหาข้อมูลไปด้วย

และเนื้อหาที่แม่ข่ายเว็บเซอร์วิสตอบกลับนั้นในกรณีที่เป็นการร้องขอข้อมูลด้วยบริการ RETRIEVE นั้นสามารถลดขนาดของเนื้อหาได้ประมาณ 78% และกรณีที่เป็นการร้องขออื่นๆ ที่มีการตอบกลับด้วยรหัสแสดงผลลัพธ์ของการทำรายการต่างๆอย่าง CREATE, UPDATE หรือ DELETE นั้น ถ้าใช้เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST จะสามารถลดขนาดของเนื้อหาตอบกลับได้ประมาณ 87%

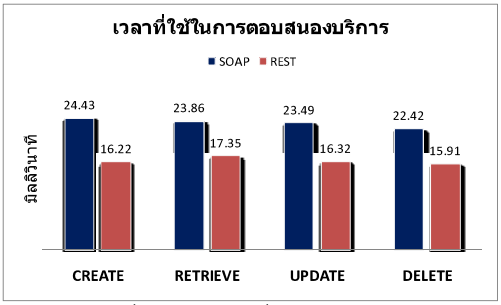


รูปที่ 7. เปรียบเทียบขนาดของเนื้อหาในการร้องขอ



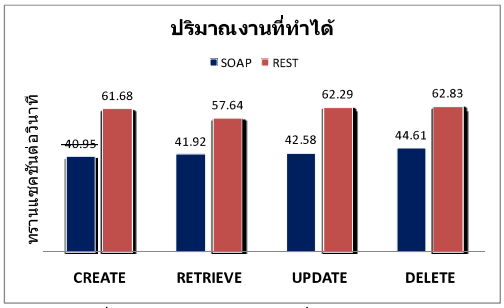
รูปที่ 8. เปรียบเทียบขนาดของเนื้อหาในการตอบกลับ

ในแง่ของเวลาตอบสนองในการใช้บริการ ซึ่งเป็นเวลารวมทั้งหมดที่ใช้ในการส่งการร้องขอบริการไปยังแม่ข่ายเว็บเซอร์วิส เวลาในการประมวลผลของแม่ข่ายเว็บเซอร์วิส และเวลาในการส่งข้อมูลตอบกลับมายังเครื่องลูกข่าย จากข้อมูลเปรียบเทียบเวลาตอบสนองในการใช้บริการในรูปที่ 9. จะเห็นว่าระยะเวลาที่ใช้ในการร้องขอบริการจากเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST จะน้อยกว่าเวลาตอบสนองของการร้องขอบริการจากเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP ประมาณ 30% โดยเฉลี่ยต่อหนึ่งทรานแซคชัน



รูปที่ 9. เปรียบเทียบเวลาที่ใช้ในการทำรายการ

เมื่อเวลาตอบสนองในการร้องขอใช้บริการต่อหนึ่งทรานแซคชันลดลงแล้วนั้นย่อมส่งผลถึงปริมาณงานที่สามารถทำได้ในช่วงเวลาหนึ่งๆ ย่อมเพิ่มขึ้น ดังแสดงให้เห็นการเปรียบเทียบได้จากรูปที่ 10. ซึ่งจะเห็นว่าเมื่อร้องขอใช้บริการเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST นั้นมีค่าปริมาณงานที่ทำได้ในช่วงเวลาหนึ่งวินาทีเพิ่มขึ้นประมาณ 30%



รูปที่ 10. เปรียบเทียบค่าปริมาณงานที่ทำได้ในช่วงวินาที

5. บทสรุปงานวิจัย

สำหรับงานวิจัยฉบับนี้ ต้องการชี้ให้เห็นถึงความสำคัญในการเลือกใช้เทคโนโลยีเว็บเซอร์วิสให้เหมาะสมกับความต้องการและลักษณะการใช้งานเพื่อให้เกิดประโยชน์สูงสุดบนทรัพยากรที่มีอยู่จำกัด ในส่วนแรกของงานวิจัยได้กล่าวถึงความรู้เบื้องเกี่ยวกับเว็บเซอร์วิส, SOAP และแนวความคิดเกี่ยวกับการนำทฤษฎีสถาปัตยกรรมแบบ REST มาพัฒนาเป็นเว็บเซอร์วิส ในส่วนที่สองนั้นได้กล่าวถึงวิธีการเรียกใช้บริการ และความแตกต่างของการเรียกใช้บริการ เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP กับ เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST และในส่วนสุดท้ายจะเป็นการทดสอบเปรียบเทียบให้เห็นถึงความแตกต่างของการเรียกใช้บริการ เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP กับ เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST ในการให้บริการแบบเดียวกันในเชิงของขนาดเนื้อหาที่ใช้ในการร้องขอบริการและการตอบรับ รวมทั้งเวลาในการตอบสนองทั้งหมดของการร้องขอบริการนั้นๆเพื่อเป็นแนวทางในการตัดสินใจเลือกใช้รูปแบบเทคโนโลยีเว็บเซอร์วิสได้อย่างเหมาะสม

จากข้อสรุปผลการทดลองจะเห็นได้ว่าการเรียกใช้งาน เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST นั้นสามารถลดปริมาณข้อมูลการร้องขอบริการและการตอบกลับระหว่างเครื่องลูกข่ายและเครื่องแม่ข่ายได้ 80% โดยประมาณ ในกระบวนการสร้างหรือแก้ไขและลดปริมาณการรับส่งข้อมูลได้มากถึง 90% โดยประมาณ ในกรณีที่เป็นการเรียกดูข้อมูลหรือลบข้อมูลในตำแหน่งทรัพยากรที่ต้องการ ทั้งยังช่วยทำให้เครื่องแม่ข่ายสามารถรองรับการให้บริการเพิ่มมากขึ้นได้โดยที่การร้องขอบริการจากเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST นั้นจะช่วยลดระยะเวลาในการตอบสนองการขอบริการ และเพิ่มปริมาณงานที่ทำได้ของแม่ข่ายเว็บเซอร์วิสได้ถึงประมาณ 30% และหากผู้ให้บริการเว็บเซอร์วิสใช้การพัฒนาเว็บเซอร์วิสภายใต้สถาปัตยกรรมแบบ REST ก็จะทำให้ผู้ขอใช้บริการนั้นได้รับการตอบสนองที่เร็วขึ้น และยังเป็นการช่วยทำให้มีการใช้งานแบนด์วิธอย่างคุ้มค่าอีกด้วย

ทั้งนี้การพัฒนาเว็บเซอร์วิสที่ใช้สถาปัตกรรม REST อาจจะมีความยุ่งยากมากกว่าการพัฒนาเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP จึงทำให้เว็บเซอร์วิสที่ใช้สถาปัตกรรม REST ไม่เป็นที่นิยมมากนัก แต่ด้วยข้อดีของการให้บริการด้วยเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST ดังที่แสดงในผลการทดสอบนั้น ชี้ให้เห็นว่าเว็บเซอร์วิสที่ใช้สถาปัตกรรม REST นั้นเหมาะที่จะให้บริการผู้ใช้บริการที่เชื่อมต่ออินเทอร์เน็ตเครือข่ายที่มีแบนด์วิธไม่สูงมากนัก อย่างเช่นการใช้งาน GPRS/EDGE [9] บนโทรศัพท์มือถือ ซึ่งจะทำให้ผู้ใช้บริการนั้นสามารถใช้บริการได้เร็วขึ้นและยังเป็นการใช้แบนด์วิธอย่างคุ้มค่าอีกด้วย

สำหรับการศึกษารูปแบบการนำทฤษฎีสถาปัตยกรรมแบบ REST มาใช้ให้เต็มประสิทธิภาพและเป็นแนวทางที่ชัดเจนขึ้น ทั้งเรื่องของการกำหนดโครงสร้างต่างๆหรือแม้กระทั่งการประยุกต์ใช้ทฤษฎีสถาปัตยกรรมแบบ REST ไปช่วยในการปรับปรุงระบบที่ใช้งานปัจจุบันอยู่นั้นกำลังอยู่ในช่วงการวิจัยศึกษาค้นคว้า และจัดเตรียมเพื่อจะเผยแพร่ต่อไป

เอกสารอ้างอิง

[1] Roy Thomas Fielding, “Architectural Styles and the Design of Network-based Software Architectures”, Chapter 5, PhD thesis, University of California, Irvine, 2000

[2] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L.Masinter, P.Leach and T.Berners-Lee, “Hypertext transfer protocol 1.1”, Internet, June 1999

[3] Nilo Mitra, Yves Lafon, “SOAP Version 1.2 Part 0: Primer (Second Edition)”, Published on the Internet, W3C Recommendation, April 2007

[4] Alex Rodriguez, “RESTful Web Services: The basics”, IBM, November 2008

[5] T. Berners-Lee, L.Masinter and M. McCahill, “RFC 1738 : Uniform Resource Locators (URL)”, Published: www.rfc-editor.org, December 1994

[6] Martin Gudgin, Marc Hadley and Tony Rogers, “Web Services Addressing 1.0 – Core”, Published on the Internet, W3C Recommendation, May 2006

[7] Roberto Chinnici, Jean-Jacques Moreau, Arthur Ryman, Sanjiva Weerawarana, “Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language”, Published on the Internet, W3C Candidate Recommendation, March 2006

[8] Naga, Sravanthi (119500) Performance Testing CoE, “Open Source Performance Testing Using Apache JMeter”, Cognizant Technology Solutions, 2008

[9] J. Schiller, “Mobile Communications”, Addison-Wesley, 2000

ประวัติผู้เขียน

ชื่อ – นามสกุล	นาย พชรวร บุญชู
วัน เดือน ปีเกิด	8 กุมภาพันธ์ พ.ศ. 2526 ที่จังหวัดกรุงเทพมหานคร
ที่อยู่	199/1 ซอยกรุงเทพกรีฑา 27 ถนนกรุงเทพกรีฑา
	สะพานสูง กรุงเทพมหานคร
ประวัติการศึกษา	พ.ศ. 2548 วิศวกรรมศาสตร์บัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์
	มหาวิทยาลัยมหิดล
ประวัติการทำงาน	
พ.ศ. 2548 - 2549	นักพัฒนาเว็บแอพพลิเคชัน
	บริษัท เอเชียซอฟท์ คอร์ปอเรชั่น จำกัด (มหาชน)
พ.ศ. 2549 - 2551	วิศวกร บริษัท แอดวานซ์ อินโฟร์ เซอร์วิส จำกัด (มหาชน)
ปัจจุบัน	วิศวกรอาวุโส บริษัท ทรูมูฟ จำกัด