การพัฒนาเว็บเซอร์วิสบนระบบฝังตัว

EMBEDDED WEB SERVICE DEVELOPMENT

นายธเนศ ชูวิเศษวณิชย์

นายธีรยุทธ สุทธิสุวรรณ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

การพัฒนาระบบเว็บเซอร์วิสบนอุปกรณ์ฝังตัว

EMBEDDED WEB SERVICE DEVELOPMENT

โดย

นายธเนศ ชูวิเศษวณิชย์

นายธีรยุทธ สุทธิสุวรรณ

อาจารย์ที่ปรึกษา

ผศ. อภิเนตร อูนากูล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

ปริญญานิพนธ์ปีการศึกษา 2546

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาเว็บเซอร์วิสบนระบบฝังตัว

Embedded Web Service Development

ผู้จัดทำ

1. นายธเนศ ชูวิเศษวณิชย์ รหัสประจำตัว 43010175
2. นายธีรยุทร สุทธิสุวรรณ รหัสประจำตัว 43010189

อาจารย์ที่ปรึกษา

(ผศ. อภิเนตร อูนากูล)

การพัฒนาเว็บเซอร์วิสบนระบบฝังตัว

นายธเนศ ชูวิเศษวณิชย์ 43010175
นายธีรยุทธ สุทธิสุวรรณ 43010189
ผศ.อภิเนตร อูนากูล อาจารย์ที่ปรึกษา
ปีการศึกษา 2546

บทคัดย่อ

ในปัจจุบันการสื่อสารระหว่างแอพพลิเคชั่นผ่านออบเจ็กต์ระยะไกล (Remote Procedure Call) ได้รับการพัฒนาขึ้นด้วยเทคโนโลยีเว็บเซอร์วิส (Web Service) ที่นำเสนอแนวคิดในการกำหนดมาตรฐานการเชื่อมต่อระหว่างแอพพลิเคชั่น เพื่อให้ระบบที่มีความแตกต่างกันสามารถติดต่อสื่อสารถึงกันได้ โดยอาศัยข้อดีของโปรโตคอลเอชทีทีพี (HTTP Protocol) ที่มีความนิยมใช้กันอย่างแพร่หลาย และกลไกในการแลกเปลี่ยนข้อมูลผ่านทาง SOAP (Simple Object Access Protocol) ที่ใช้โครงสร้างของ XML Language ที่ใช้ในการนิยามความหมายของข้อมูล

การนำเอาเว็บเซอร์วิสมาพัฒนาบนระบบฝังตัวจะช่วยทำให้อุปกรณ์มีความสามารถในการเชื่อมต่อกับอุปกรณ์ต่างๆ และความสามารถในการเชื่อมโยงข้อมูลที่ใช้ในการควบคุมอุปกรณ์ได้อย่างมีประสิทธิภาพสูงสุด จึงได้เกิดโครงงานในปีการศึกษา 2545 ที่นำเสนอซอฟต์แวร์บนระบบฝังตัวที่ช่วยในการจัดการเอกสารที่มีรูปแบบตามมาตรฐานซิมเปิ้ล ออบเจ็กต์โปรโตคอล และเป็นเซิร์ฟเวอร์ที่ให้บริการเซอร์วิสกับอุปกรณ์ต่างๆได้ และในโครงงานนี้เป็นการพัฒนาซอฟต์แวร์ดังกล่าวต่อเนื่องจากเดิม เพื่อเพิ่มประสิทธิภาพและความสามารถในการทำงานให้เหมาะกับการนำไปใช้งานจริงต่อไป

ปริญญานิพนธ์ฉบับนี้จัดทำขึ้นเพื่อวัตถุประสงค์ในการนำเสนอแนวความคิด ทฤษฎีและหลักการในการพัฒนาเว็บเซอร์วิสบนระบบฝังตัวที่สามารถเป็นเซิร์ฟเวอร์ที่ให้บริการเซอร์วิส

EMBEDDED WEB SERVICE DEVELOPMENT

Mr. Thanet Choowisetwanich
Mr. Teerayut Sutthisuwan
Mr. Apinetr Unakul Advisor
Academic Year 2003

ABSTRACT

Presently, application communicated via Remote Procedure Call is developed with Web Service Technology, the concept is to define standard communication among applications, that have different technologies are accessible. Web Service uses the benefit of ubiquitous HTTP Protocol that is simple and lightweight mechanism for exchanging structured and type information among peers in a decentralized, distributed environment using SOAP, based on XML

Embedded Web Service Development will enhance connection with equipment and link data for distributed environment. The previous project in academic year 2002 presented the software that manage SOAP Message and responds the request SOAP Message as Server , and this project presents the continuous software development to improve performance and increase abilities for using in the industrial environment

The objective of this project is to present the idea, the mechanism, and the concept of developing "Embedded Web Service Development" which is software system that provide service to equipment of the network

กิตติกรรมประกาศ

การทำปริญญานิพนธ์ฉบับนี้ไม่อาจเสร็จได้ด้วยดี หากไม่ได้รับความช่วยเหลือและร่วมมือจากหลายฝ่ายด้วยกัน บุคคลแรกที่ต้องกล่าวถึงเพราะเป็นส่วนสำคัญที่ทำให้วิทยานิพนธ์ฉบับนี้เสร็จลงได้ก็คือ อาจารย์อภิเนตร อูนากูล ซึ่งเป็นอาจารย์ที่ปรึกษาวิทยานิพนธ์ที่ให้ความเอาใจใส่ แนะนำวิธีการคิดที่เป็นระบบ และช่วยเหลือในทุกๆด้านเสมอมา จึงขอขอบพระคุณเป็นอย่างสูง

ขอขอบคุณอาจารย์ในภาควิชาวิศวกรรมคอมพิวเตอร์ทุกๆท่าน ที่ให้ความรู้ คำปรึกษา ตลอดจนให้การฝึกอบรมความรู้ความสามารถตลอดระยะเวลา 3 ปี

ขอขอบคุณห้องปฏิบัติการ Embedded System Lab ที่เอื้อเฟื้อสถานที่ที่สะดวกสบายและเต็มไปด้วยสิ่งที่เกื้อหนุนต่อการเรียนรู้และดำเนินโครงงาน

ขอขอบคุณสำนักวิจัยและบริการคอมพิวเตอร์ของสถาบัน ที่เอื้อเฟื้อสถานที่ในดำเนินโครงงานและพำนักอาศัยตลอดระยะเวลาที่ผ่านมา รวมถึงขอขอบคุณพี่กิตติ พี่อู๋ และพี่ๆทุกคนที่มีความเมตตาเสมอมา

ขอขอบคุณบริษัทเน็ตแกจเกจ ที่เอื้อเฟื้อชุดพัฒนาคอม86 สำหรับใช้ในการดำเนินโครงงานนี้ โดยเฉพาะพี่โอ ที่ให้คำปรึกษาและคำแนะนำที่มีค่าอย่างยิ่งในการดำเนินโครงงานให้สำเร็จได้ และขอขอบคุณพี่ขนิษฐา ที่ช่วยประสานงานในการประกวดโครงงาน และส่งมอบชุดพัฒนาดังกล่าวด้วย

ขอขอบคุณพี่ฮิต ที่พัฒนาโครงงานที่ดีและสมบูรณ์ที่สามารถนำมาศึกษาและพัฒนาต่อได้โดยสะดวก รวมถึงยังให้คำแนะนำในแนวทางในการดำเนินโครงงานที่มีคุณค่าอย่างมาก

ขอขอบคุณพี่ๆและเพื่อนๆ ทุกคนที่คอยให้คำปรึกษา ช่วยเหลือและให้กำลังใจตลอดมา ซึ่งทำให้ชีวิตในสถาบันเต็มไปด้วยความอบอุ่น และประสบการณ์ที่ล้ำค่าในชีวิต

สุดท้ายนี้ต้องขอขอบพระคุณบุคคลสำคัญที่สุดที่ทำให้ข้าพเจ้ามีวันนี้ก็คือ บิดา มารดา อันเป็นที่เคารพรักยิ่ง ซึ่งได้เลี้ยงดูข้าพเจ้ามาเป็นอย่างดี พร้อมทั้งให้โอกาสในการศึกษาอย่างเต็มที่ และยังให้กำลังใจ เอาใจใส่เสมอมา ในทุกๆด้านอันหาที่เปรียบมิได้ ข้าพเจ้าขอระลึกในพระคุณอันสุดประมาณและขอกราบขอบพระคุณมา ณ ที่นี้

ธเนศ ชูวิเศษวณิชย์

ธีรยุทธ สุทธิสุวรรณ

สารบัญ

หน้า

หน้า

หน้า

สารบัญภาพ

หน้า

หน้า

หน้า

สารบัญตาราง

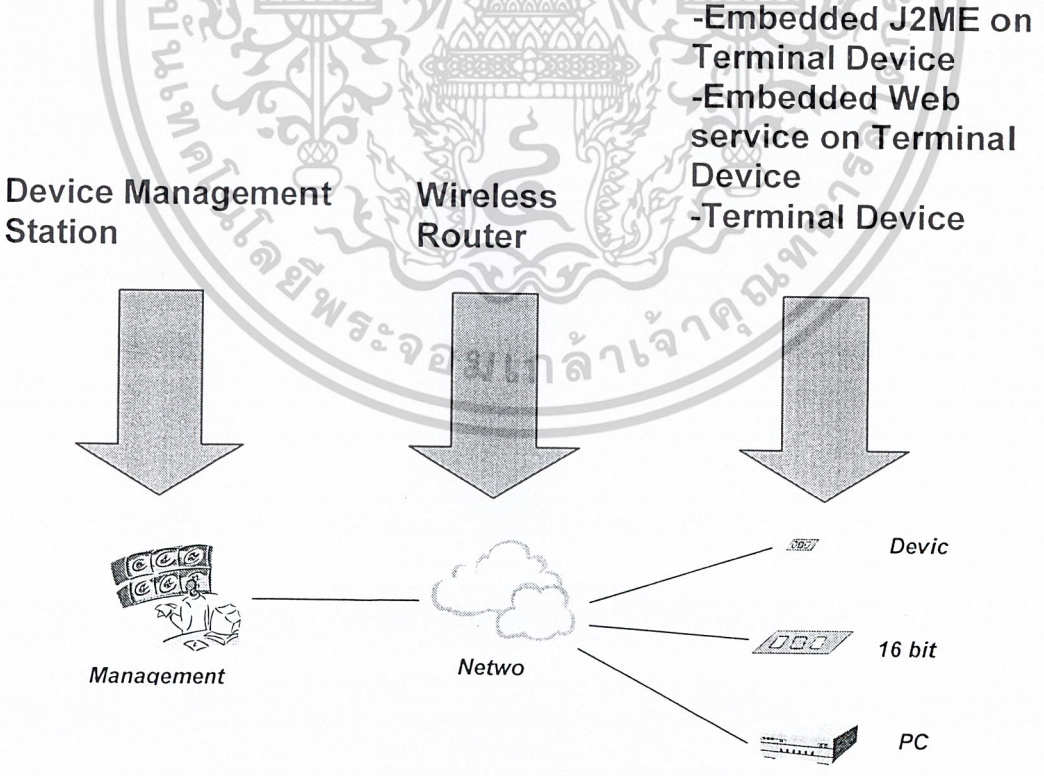
หน้า

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

ในปัจจุบันการควบคุมการทำงานของแอพพลิเคชั่น (Application) บนอุปกรณ์ต่างๆนั้น มักจะใช้คน เพื่อคอยควบคุมการทำงาน ซึ่งทำเกิดปัญหาต่างๆที่คนเป็นสาเหตุตามมา ยกตัวอย่าง เช่น เกิดจากความสะเพร่า ความประมาท รวมถึงความล่าช้าในการทำงาน เป็นต้น ดังนั้นจึงได้มีแนวคิดในการที่จะปรับปรุงการทำงาน กล่าวคือ เปลี่ยนจากการใช้คนในการควบคุมมาเป็นการควบคุมด้วยแอพพลิเคชั่นแทน

ในการนำเครื่องคอมพิวเตอร์ที่มีราคาแพง มาใช้ในการควบคุมการทำงานระหว่างแอพพลิเคชั่น อาจจะไม่คุ้มค่าต่อการลงทุน อีกทั้งยังมีปัญหาสภาวะแวดล้อมของสถานที่ในการทำงานที่ไม่เหมาะกับการทำงานของเครื่องคอมพิวเตอร์ ซึ่งจะมีผลต่ออายุการใช้งานเครื่องคอมพิวเตอร์ดังกล่าว ทำให้มีค่าใช้จ่ายในส่วนของการบำรุงรักษามากขึ้น จากปัญหาเหล่านี้ จึงได้เกิดแนวคิดในการนำอุปกรณ์บนระบบฝังตัวที่มีความทนทานต่อสภาวะแวดล้อมมากกว่าเครื่องคอมพิวเตอร์ ราคาถูกกว่า อีกทั้งยังมีความสะดวกในด้านการเชื่อมต่อกับอุปกรณ์ภายนอกที่ดีสร้างการเชื่อมต่อเข้ากับเครือข่ายเพื่อให้สามารถเรียกใช้งานได้จากทุกหนทุกแห่ง



รูปภาพ 1-1 การเชื่อมต่ออุปกรณ์บนระบบฝังตัวผ่านเครือข่ายเพื่อให้สามารถเรียกใช้งานได้จากทุกแห่ง

การสื่อสารระหว่างแอพพลิเคชั่นจะช่วยให้การเชื่อมโยงข้อมูลผ่านทางอินเทอร์เน็ตโดยใช้เทคโนโลยีเว็บเซอร์วิส ที่มีการติดต่อสื่อสารโดยอาศัยซิมเปิ้ลออบเจ็กต์แอคเซสโปรโตคอล ที่ทำงานอยู่บนโปรโตคอลเอชทีทีพี ที่เป็นที่นิยมใช้งานกันอย่างแพร่หลาย เราจึงไม่ต้องทำการเปิดพอร์ตการสื่อสารขึ้นมาใหม่ และไม่เกิดปัญหาขึ้นในส่วนของไฟร์วอล (Firewall) และพร็อกซี่เซิร์ฟเวอร์ (Proxy Server) อีกทั้งซิมเปิ้ลออบเจ็กต์โปรโตคอลนั้น ยังเป็นมาตรฐานเปิดทำให้สามารถติดต่อสื่อสารกับเครื่องคอมพิวเตอร์ที่มีความแตกต่างกันทั้งระบบปฏิบัติการ เทคโนโลยี และภาษาในการพัฒนาซึ่งจะช่วยแก้ปัญหาของการเชื่อมโยงข้อมูลระหว่างเครื่องในแต่ละส่วนของขั้นตอนการทำงานที่แตกต่างกัน

จากเหตุผลข้างต้นนี้ จึงนำไปสู่แนวคิดในการนำเทคโนโลยีการติดต่อสื่อสารแบบเว็บเซอร์วิส ลงไปใช้บนอุปกรณ์ฝังตัว ดังนั้นจึงได้เกิดโครงงานของนักศึกษาระดับปริญญาตรี ภาควิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง ในปีการศึกษา 2545 ที่ผ่านมา ได้จัดทำโครงงานเกี่ยวกับ การพัฒนาเว็บเซอร์วิสบนระบบฝังตัว ซึ่งผลจากการดำเนินการโครงงานดังกล่าว ได้แก่ ซอฟต์แวร์ไลบรารีที่สามารถพัฒนาให้เป็นโปรแกรมที่สามารถทำการเชื่อมต่อผ่านระบบเว็บเซอร์วิสได้ โดยในโครงงานดังกล่าวได้ทำการทดลองนำเอาซอฟต์แวร์ไปทดลองสร้างการให้บริการเว็บเซอร์วิสบนระบบฝังตัวซึ่งทำการเลือกบอร์ดคอม 86 เป็นแพล็ตฟอร์มตัวอย่างในการสร้าง

แต่ในสภาพแวดล้อมในการทำงานของระบบซอฟต์แวร์บนอุปกรณ์ฝังตัวนั้น จะมีการติดต่อและควบคุมระบบฮาร์ดแวร์หรือเครื่องจักรที่ทำงานอยู่ตลอดเวลา หรืออาจเกิดเหตุการณ์ใดๆที่ต้องการการทำงานที่ทันเวลา (Real-time processing) ซึ่งจำเป็นอย่างยิ่งที่ต้องมีระบบที่สามารถรองรับการทำงานดังกล่าวเพื่อสามารถนำไปใช้ในสภาพแวดล้อมจริงได้ อีกทั้งยังไม่สามารถรองรับวิธีการติดต่อด้วยโปรโตคอลอื่นๆ ได้ในเวลาเดียวกัน โครงงานนี้จึงทำการพัฒนาระบบเว็บเซอร์วิส ต่อจากโครงงานรุ่นที่ผ่านมา โดยเพิ่มความสามารถในการรองรับการทำงานแบบมัลติทาสก์กิ้ง และการทำงานแบบเรียลไทม์ได้ รวมถึงออกแบบโครงสร้างใหม่ เพื่อให้สามารถรองรับการพัฒนาโปรโตคอลอื่นได้บนระบบเดียวกัน แต่ยังสามารถเรียกใช้บริการที่มีอยู่ได้เพื่อเพิ่มความหลากหลายในการติดต่อใช้งาน อีกทั้งยังพัฒนาวิธีการสร้างและพัฒนาบริการต่างๆบนระบบที่พัฒนาขึ้นด้วย

1.2 วัตถุประสงค์ของงานวิจัย

ปรับปรุงและเพิ่มประสิทธิภาพในการทำงานของโซบเซิร์ฟเวอร์ในด้าน

1. การประมวลผลแบบมัลติทาสก์กิ้ง
2. รองรับการขยายเพื่อนำไปใช้งานร่วมกับโปรโตคอลอื่นๆ ได้
3. ง่ายต่อการนำไปใช้และพัฒนาสู่การใช้งานจริง

1.3 ขอบเขตของงานวิจัย

งานวิจัยนี้ต้องการพัฒนาเว็บเซอร์วิสบนอุปกรณ์ระบบฝังตัว โดยนำผลงานจากโครงงานวิจัยปริญญาตรีของนักศึกษา ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ปีการศึกษา 2545 มาทำการศึกษาโครงสร้างและแนวคิดในการออกแบบโดยการทำวิศวกรรมย้อนกลับ (Reverse Engineer) เพื่อทำความเข้าใจระบบโครงสร้างและสถาปัตยกรรมของระบบเว็บเซอร์วิสบนระบบฝังตัวโดยรวม และปรับปรุงข้อบกพร่องต่างๆ แล้วทำการศึกษาการทำงานแบบมัลติทาสก์กิ้ง ที่สามารถรองรับการทำงานหลายอย่างพร้อมๆกัน เพื่อทำการออกแบบระบบเว็บเซอร์วิสที่สามารถทำงานแบบมัลติทาสก์กิ้งได้ แล้วทำการออกแบบสถาปัตยกรรมของระบบใหม่ ให้สามารถรองรับการพัฒนารูปแบบการสื่อสารด้วยโปรโตคอลอื่นร่วมด้วย โดยยังสามารถใช้งานบริการต่างๆ ที่มีอยู่ร่วมกับระบบเว็บเซอร์วิสที่มีอยู่แล้วได้ในเวลาเดียวกัน สุดท้ายยังรวมถึงพัฒนารูปแบบวิธีการและสิ่งอำนวยความสะดวกในการพัฒนาบริการบนระบบที่เว็บเซอร์วิสที่ได้สร้างขึ้น เพื่อช่วยให้นักพัฒนาท่านอื่นที่สนใจและเกี่ยวข้องกับงานในด้านของอุปกรณ์ฝังตัว สามารถนำไปใช้เพื่อพัฒนาบริการบนระบบนี้ต่อไป

อนึ่งงานวิจัยนี้มีการนำชุดพัฒนาบอร์ดคอม86ของบริษัทเน็ตเก็ดเจ็ต (Netgadgets) มาสำหรับใช้เป็นตัวอย่างแพลตฟอร์มในการวิจัยและพัฒนาตัวอย่างการให้บริการเว็บเซอร์วิสในงานวิจัยนี้

1.4 เป้าหมายของโครงงาน

1. ซอฟต์แวร์ไลบรารี่ของเว็บเซอร์วิสบนระบบฝังตัว
2. เฟรมเวิร์ค (Framework) ที่ใช้เป็นแนวทางในการพัฒนาบริการในระบบ

1.5 วิธีการดำเนินงาน

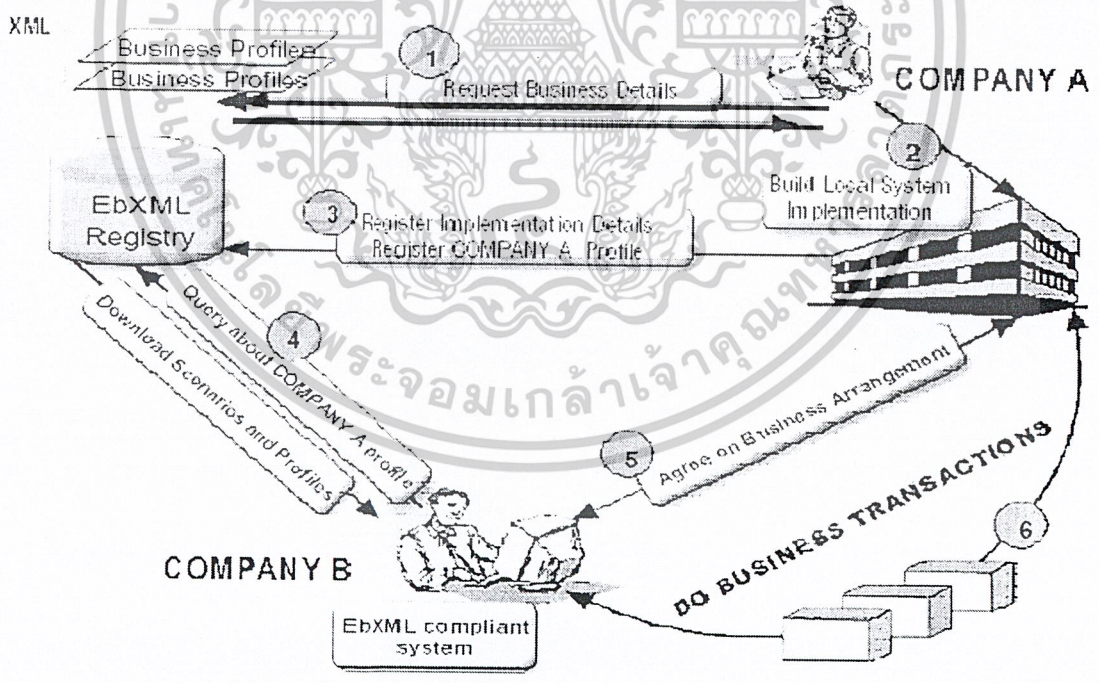
1. ศึกษาทฤษฎีพื้นฐานต่างๆ ที่เกี่ยวข้องกับงานวิจัย ซึ่งได้แก่ ระบบเว็บเซอร์วิส ภาษาเอ็กซ์เอ็มแอล มาตรฐานซิมเปิ้ลออบเจ็กต์แอคเซสโพรโตคอล มาตราฐานดับบิวเอสดีแอลโพรโตคอล ยูดีดีไอ โพรโตคอล ระบบฝังตัว และบอร์ดคอม86
2. ทำการศึกษาสถาปัตยกรรมการทำงานของระบบเว็บเซอร์วิสบนระบบฝังตัวที่ได้นำมาพัฒนาต่อ แล้วแก้ไขปรับปรุงข้อบกพร่องต่างๆ ที่พบในซอฟต์แวร์ไลบรารี
3. ศึกษาทฤษฎีพื้นฐาน เกี่ยวกับระบบการทำงานแบบมัลติทาสก์กิ้งและแบบเรียลไทม์ และศึกษาแนวคิดในการพัฒนาโปรแกรมบนระบบดังกล่าว
4. ศึกษาและทดลองใช้งานโปรแกรมที่ทำหน้าที่เป็นระบบปฏิบัติการที่ทำงานแบบมัลติทาสก์กิ้งและแบบเรียลไทม์ ซึ่งได้แก่ ไมโครซี โอเอสทู (μC/OSii)
5. สร้างสภาพแวดล้อมในการพัฒนาระบบเว็บเซอร์วิสบนไมโครซี โอเอสทู แล้วทดลองพัฒนาโปรแกรมที่ทำงานแบบมัลติทาสก์กิ้ง บนระบบที่เตรียมไว้
6. ออกแบบสถาปัตยกรรมการทำงานของระบบเว็บเซอร์วิสที่งานแบบมัลติทาสก์กิ้ง รวมถึงออกแบบโครงสร้างของระบบซอฟต์แวร์ที่จะพัฒนา
7. ออกแบบวิธีการในการพัฒนาหรือสร้างบริการบนระบบเว็บเซอร์วิสที่ได้ออกแบบไว้
8. ทำการพัฒนาซอฟต์แวร์ตามที่ได้ออกแบบไว้ โดยพัฒนาแยกเป็นส่วน หรือพัฒนาเป็น module แล้วทำการทดสอบส่วนย่อยๆนั้น แล้วทำการแก้ไขให้สมบูรณ์
9. หลังจากพัฒนาจบครบองค์ประกอบของระบบซอฟต์แวร์แล้ว ก็นำมาประกอบกัน แล้วทดสอบการทำงานโดยรวมของระบบ แล้วแก้ไขจนสามารถทำงานได้อย่างถูกต้อง
10. สรุปผลการทำงานซอฟต์แวร์ ผลที่ได้รับจากงานวิจัยชิ้นนี้ และแนวทางในการพัฒนางานวิจัยนี้เพิ่มเติมและแนวทางในการนำไปประยุกต์ใช้

บทที่ 2
ทฤษฎีและหลักการ

2.1 เว็บเซอร์วิส (Web Service)

2.1.1 เทคโนโลยีเว็บเซอร์วิส

การพัฒนาทางเทคโนโลยีใดๆ ล้วนแล้วแต่มีการพัฒนาเป็นยุคสมัย แต่ละยุคสมัยก็จะมีการใช้เทคโนโลยีที่มีจุดสำคัญของการพัฒนาที่เด่นชัด การพัฒนาเทคโนโลยีประยุกต์ใช้งานบนเว็บก็เช่นเดียวกัน เริ่มจากยุคแรกที่เป็นแบบสแตติกเว็บเพจที่เป็นเพียงแค่ไฟล์เอชทีเอมแอลไว้ให้ผู้ใช้ดูข้อมูลเฉยๆ จนสู่ยุคที่ 2 เป็นแบบไดนามิกเว็บเพจ (Dynamic Web Page) ที่สามารถแปรเปลี่ยนตามคำร้องขอของผู้ใช้งานได้ ซึ่งเป็นการเพิ่มประสิทธิภาพของเว็บเพจ จนกระทั่งในปัจจุบันมีการพัฒนาเว็บเพจเข้ามาสู่ยุคที่ 3 หรือแบบการบริการผ่านเว็บเซอร์วิส โดยจุดประสงค์เพื่อรองรับความต้องการการใช้งานที่สูงขึ้น และเป็นมาตรฐานในการทำงานร่วมกัน



รูปภาพ 2-1 ลักษณะของเว็บเซอร์วิส

เว็บเซอร์วิส คือ ซอฟต์แวร์คอมโพแนนท์ (Software Component) ที่ผู้ให้บริการนำมาสร้างเป็นแอพพลิเคชั่นสำหรับให้บริการการทำงานหนึ่งๆ ให้กับผู้ขอบริการทางอินเทอร์เน็ต หรือผู้ขอบริการสามารถที่จะขอบริการจากหลายๆที่มาประกอบกันได้ โดยที่แต่ละระบบนั้นมีความเป็นอิสระจากกัน (Loosely Coupled) ซึ่งปัจจัยพื้นฐานของเว็บเซอร์วิสที่ควรมี ซึ่งมีความเกี่ยงข้องกันในทางเทคนิคและทางธุรกิจคือ

- การรวมซอฟต์แวร์ต่างระบบกัน จะต้องอนุญาตให้แต่ละระบบมีความเป็นอิสระจากกัน
- อินเทอร์เฟสทางด้านการบริการของซอฟต์แวร์ควรจะเผยแพร่สู่สาธารณชนและสามารถเข้าถึงได้ง่าย
- แมสเซจที่ใช้ติดต่อกันของการทำงานแบบแอพพลิเคชั่นกับแอพพลิเคชั่นต้องสอดคล้องกับมาตรฐานเปิดบนอินเทอร์เน็ต
- แอพพลิเคชั่นสามารถสร้างได้จากการใช้ซอฟต์แวร์คอมโพเนนท์ทั้งจากภายในและภายนอกองค์กร โดยสร้างตามการดำเนินธุรกิจหลักขององค์กร
- มีแหล่งซอฟต์แวร์คอมโพเนนท์ที่หาได้ง่าย ซึ่งช่วยเพิ่มความยืดหยุ่นในการสร้างแอพพลิเคชั่นที่มีคุณลักษณะเฉพาะตามกระบวนการดำเนินการทางธุรกิจ
- การนำซอฟต์แวร์จากภายนอกมาใช้ใหม่ช่วยให้เกิดการลดต้นทุนและช่วยปรับปรุงบริการให้แก่ลูกค้า

หลักการพื้นฐานของเทคโนโลยีต่างๆที่พัฒนามาสู่เว็บเซอร์วิส

- การพัฒนาโปรแกรมแบบซอฟต์แวร์คอมโพเนนท์ ตามแนวคิดของการออกแบบและพัฒนาโปรแกรมเชิงวัตถุ (Object-Oriented Concept)
- การออกแบบระบบแบบกระจายจากศูนย์กลาง (Distributed Computing) ซึ่งเป็นเป้าหมายสำคัญของการพัฒนาระบบตามสถาปัตยกรรมแบบไคลเอนต์เซิร์ฟเวอร์ (Client-Server)
- การทำอีดีไอ หรือ อิเล็กทรอนิกส์ดาต้าอินเทอร์เช้นต์ (Electronic Data Interchange) ซึ่งสร้างขึ้นโดยกำหนดรูปแบบและมาตรฐานของข้อมูลสำหรับการทำธุรกิจ
- การบูรณาการของซอฟต์แวร์ต่างระบบ หรือ เอนเทอร์ไพรส แอพพลิเคชั่น อินทิเกรชั่น : อีเอไอ (Enterprise Application Integration : EAI) ที่อยู่บนพื้นฐานของความต้องการใช้ข้อมูลร่วมกัน รวมทั้งการแลกเปลี่ยนข้อมูลระหว่างแอพพลิเคชั่นให้สามารถทำงานได้อย่างถูกต้องเหมาะสม
- รูปแบบการให้บริการซอฟต์แวร์แบบเอเอสพี (ASP) หรือ แอพพลิเคชั่นเซอร์วิสโพรไวเดอร์ (Application Service Provider)

2.1.2 ลักษณะของเว็บเซอร์วิส

การพัฒนาเว็บเซอร์วิสใช้สถาปัตยกรรมบริการในลักษณะที่เรียกว่า สถาปัตยกรรมของแนวคิดทางด้านบริการ หรือ เอสโอเอ (Service-Oriented Architecture : SOA) เป็นแนวคิดเบื้องต้น แอพพลิเคชั่น

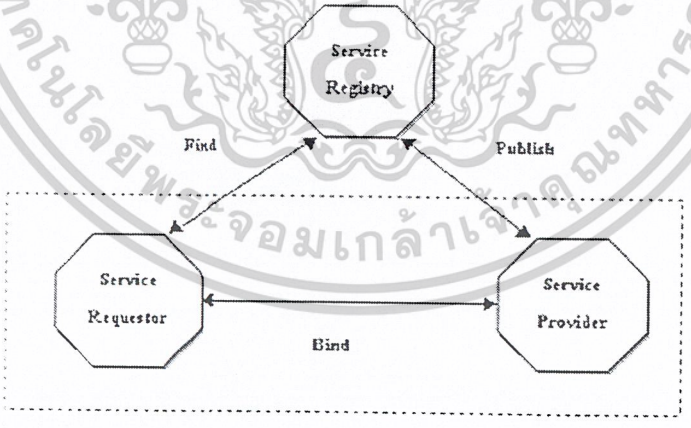
ส่วนใหญ่ในโลกของธุรกิจที่ใช้งานในปัจจุบัน เป็นแอพพลิเคชั่นและระบบย่อยที่ถูกสร้างขึ้น มีการทำงานที่ต้องสัมพันธ์กันอย่างไม่เป็นอิสระจากกัน การเปลี่ยนแปลงการทำงานในระบบย่อยหรือแอพพลิเคชั่นหนึ่ง อาจจะมีผลกระทบกับอีกแอพพลิเคชั่นหนึ่ง หรือบางครั้งอาจจะกระทบทั้งระบบ ทำให้การบำรุงรักษานั้นทำได้ยาก และ มีต้นทุนที่สูงขึ้น รวมทั้งยังเป็นข้อจำกัดในการเชื่อมต่อกับระบบของคู่ค้าอื่นๆ

เอสโอเอมีส่วนประกอบหหหลัก 3 ส่วนคือ

1. ผู้ให้บริการ หรือที่เรียกว่า เซฮร์วิสโพรไวเดอร์ (Service Provider)
2. ผู้ขอใช้บริการ หรือที่เรียกว่า เซอร์วิสรีเควสเตอร์ (Service Requester)
3. ตัวแทนของผู้ให้บริการ หรือที่เรียกว่า เซอร์วิสโบรกเคอร์ (Service Broker)

โดยส่วนประกอบหลักทั้ง 3 ส่วนนี้ สามารถติดต่อถึงกันได้โดยใช้ฟังก์ชันพื้นฐาน (Primary Function) คือการประกาศ (Publish) การค้นหา (Find) และการเรียกใช้ (Bind) ซึ่งฟังก์ชันทั้ง 3 มีการทำงานดังนี้

1. ผู้ให้บริการทำการประกาศบริการที่ตนเองให้บริการ ไปยังตัวแทนของผู้ให้บริการ ซึ่งตัวแทนของผู้ให้บริการจะทำการบันทึกเก็บไว้ในไดเรกทอรีของบริการ (Directory Service)
2. ผู้ขอใช้บริการจะทำการค้นหาบริการจากตัวแทนของผู้ให้บริการ
3. เมื่อพบบริการที่ต้องการแล้ว ผู้ให้บริการและผู้ขอใช้บริการจะทำการติดต่อกัน โดยผู้ขอใช้บริการจะทำการเรียกใช้บริการไปยังผู้ให้บริการนั้น



รูปภาพ 2-2 แสดงความสัมพันธ์ของผู้ให้บริการ ผู้ขอใช้บริการ และตัวแทนผู้ให้บริการ

2.1.3 เทคโนโลยีพื้นฐานของเว็บเซอร์วิส

จากแนวคิดของเอสโอเอ ถูกนำมาใช้เป็นหลักการพื้นฐานของการพัฒนาเทคโนโลยีด้านเว็บเซอร์วิส ซึ่งเทคโนโลยีที่ถูกพัฒนาขึ้นมานี้เป็นมาตรฐานเปิดบนอินเทอร์เน็ตที่เกิดจากการทำงานร่วมกันของนักวิจัย และที่ปรึกษาจากบริษัทซอฟต์แวร์ต่างๆ

- เอ็กซ์เอ็มแอล หรือ เอ็กซ์เทนซิเบิ้ลมาร์กอัพแลงเกวด เป็นภาษามาร์กอัพที่เป็น เท็กซ์เบส (Text-Based) ที่ใช้เป็นมาตรฐานในการแลกเปลี่ยนข้อมูลบนอินเตอร์เน็ตในปัจจุบัน ผู้ที่ทำหน้าที่รับผิดชอบ และกำหนดมาตรฐานของ เอ็กซ์เอ็มแอล คือ ดับบริวทรีซี หรือ เวิร์ดไวด์เว็บคอนซอร์เดเที่ยม (W3C-World Wide Web Consortium)
- โซบ หรือ ซิมเปิ้ลออปเจ็กต์แอคเซสโปรโตคอล เป็นโปรโตคอลที่สร้างบนพื้นฐานของเอ็กซ์เอ็มแอล และยังเป็นแมสเซจจิ้งโปรโตคอล (Messaging Protocol) สำหรับใช้ในการแลกเปลี่ยนข้อมูลในสภาวะแวดล้อมแบบกระจายศูนย์ (Decentralized , Distributed Environment) โดยโซบ ได้กำหนดแมสเซจจิ้งโปรโตคอลระหว่างผู้ขอบริการกับผู้ให้บริการในการติดต่อสื่อสารกัน เช่น กำหนดให้ผู้ขอใช้บริการต้องส่งข้อมูล เช่น ข้อมูลที่ระบุฟังค์ชันและค่าพารามิเตอร์ต่างๆ ที่ต้องใช้ในแอพพลิเคชั่นที่ร้องขอ ส่งไปให้กับผู้ให้บริการ
- ดับบิวเอสดีแอล หรือ เว็บเซอร์วิสเดสสคริปต์ชั่นแลงเกวด (WSDL – Web Service Description Language) เป็นภาษาที่ใช้อธิบายคุณลักษณะการให้บริการของเว็บเซอร์วิส และวิธีการติดต่อขอรับบริการจากเว็บเซอร์วิส ดับบิวเอสดีแอลสร้างขึ้นโดยใช้ภาษาเอ็กซ์เอ็มแอล

2.2 เอ็กซ์เอ็มแอล (XML)

มาร์กอัพแลงเกวด (Markup language) คือ ภาษาที่ใช้อธิบายความหมายของเอกสารหรือข้อมูล ในรูปของแท็ก (tag) เป็นส่วนมาก มาร์กอัพแลงเกวดมีข้อดีคือ สามารถใช้ได้กับทุกแพลตฟอร์ม

2.2.1 ลักษณะที่สำคัญของเอ็กซ์เอ็มแอล

เอ็กซ์เอ็มแอล เป็นภาษามาร์กอัพที่เป็นเท็กซ์เบสที่ใช้เป็นมาตรฐานในการแลกเปลี่ยนข้อมูลบนอินเตอร์เน็ตในปัจจุบัน โดยมีผู้ที่ทำหน้าที่รับผิดชอบและกำหนดมาตรฐานของเอ็กซ์เอ็มแอล คือ องค์การดับบิวทรีซี

- จุดประสงค์ในการออกแบบ เพื่อกำหนดเป็นมาตรฐานในการทำเอกสารบนเครือข่ายอินเทอร์เน็ตเพื่อการแลกเปลี่ยนข้อมูลที่เป็นลำดับขั้น ดังนั้นจึงต้องมีรูปแบบที่สามารถใช้ได้ทั่วไปบนอินเทอร์เน็ต
- มีลักษณะที่ไม่ขึ้นกับแพลตฟอร์ม (Platform Independence) สามารถนำไปใช้กับคอมพิวเตอร์ระบบใด แพลตฟอร์มไหนก็ได้ เนื่องจากเอกสารเอ็กซ์เอ็มแอล เป็นเท็กซ์ไฟล์ (text file) ธรรมดา
- เป็นภาษาที่ได้รับการออกแบบมาเพื่อให้สามารถนิยามความหมายของข้อมูลได้ จึงมีการจัดโครงสร้างข้อมูล แบ่งข้อมูลออกเป็นหมวดหมู่และส่วนประกอบย่อย
- ไม่มีแท็กที่ถูกนิยามไว้ก่อน อนุญาตให้ผู้ใช้สร้างแท็กขึ้นมาเองเพื่อใช้อธิบายข้อมูลได้ โดยที่ผู้ใช้กำหนดแท็ก และใช้งานได้ทันที เนื่องจากแท็กเป็นอะไรก็ได้ที่ผู้ใช้กำหนดจึงทำให้เอ็กซ์เอ็มแอล ขยายขีดความสามารถต่อไปได้ (Extensible) และข้อมูลสามารถอธิบายความหมายข้อมูลของตัวมันเองได้ (Self Describe)
- ส่วนข้อมูลและส่วนการแสดงผลของเอกสารเอ็กซ์เอ็มแอล ถูกแยกออกจากกันอย่างชัดเจน ในเอกสารเอ็กซ์เอ็มแอล นั้น จะมีแต่ตัวเนื้อข้อมูล ส่วนการแสดงผลนั้น เราสามารถใช้สไตล์ชีท (style sheet) ได้หลายประเภท เช่น ซีเอสเอส (CSS), เอ็กซ์เอสแอล (XSL) เป็นต้น
- เนื่องจากเอกสารเอ็กซ์เอ็มแอล เป็นเพียงข้อความง่ายๆที่ประกอบด้วยแท็กบางอย่างเท่านั้น จึงสามารถสร้างเอกสารเอ็กซ์เอ็มแอล ด้วยเท็กซ์อิดิเตอร์ (text editor) ทั่วๆไปได้
- การอ่านและแปรความหมายของเอกสาร เอ็กซ์เอ็มแอล สามารถทำได้โดยใช้โปรแกรมที่เรียกว่า เอ็กซ์เอ็มแอลพาร์เซอร์ (XML Parser)ได้ ตัวอย่างของโปรแกรมประเภท เอ็กซ์เอ็มแอลพาร์เซอร์ เช่น เอ็มเอสเอ็กซ์เอ็มแอล (MSXML) ซึ่งอยู่ในอินเทอร์เน็ตเอ็กซ์พลอเรอร์ (Internet Explorer) ของไมโครซอฟต์ และ เจเอเอ็กซ์พี (JAXP) ของบริษัทซัน (Sun) เป็นต้น
- มีวิธีกำหนดโครงสร้างเอกสาร 2 วิธีคือ ดีทีดี และ เอ็กซ์เอ็มแอลสครีม่า (XML Schema) ซึ่งไม่ได้บังคับว่าจำเป็นต้องมีไฟล์กำหนดโครงสร้างเอกสาร แต่ถ้ามีและเอกสารเอ็กซ์เอ็มแอล

มีรูปแบบถูกต้องตาม ดีทีดี หรือ เอ็กซ์เอ็มแอลสครีม่า จะถือว่าเอกสารเอ็กซ์เอ็มแอลนั้นมีคุณสมบัติถูกต้อง (valid)

- มีความกะทัดรัด เข้าใจง่ายและใช้ประโยชน์ได้กว้างขวาง
- สามารถใช้ได้หลายภาษาผสมกัน เนื่องจากเอ็กซ์เอ็มแอล สนับสนุน ยูนิโค้ด(UNICODE)
- ได้รับการสนับสนุนในโปรแกรมระบบฐานข้อมูลหลายๆค่าย สามารถดึงข้อมูลจากฐานข้อมูล (Database) มาอยู่ในรูปของเอ็กซ์เอ็มแอลได้

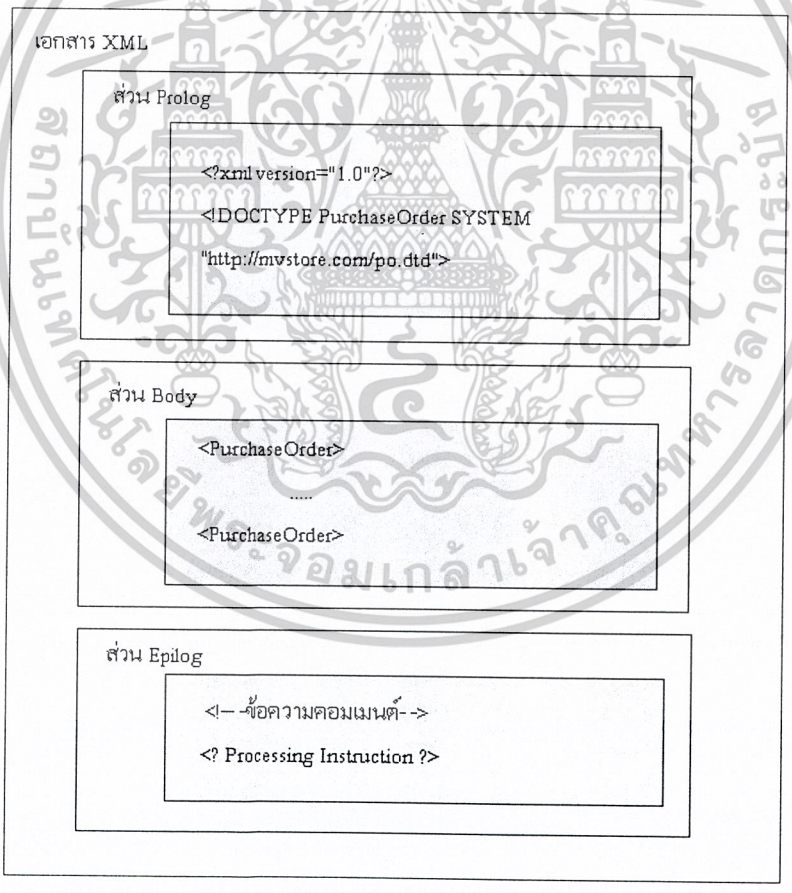
2.2.2 ประโยชน์ของเอ็กซ์เอ็มแอล

- ใช้สำหรับสร้างข้อมูลที่สามารถอธิบายความหมายของตัวมันเองได้ (Self-Describe Data) จากการที่เราสามารถกำหนดแท็ก มาอธิบายข้อมูลที่อยู่ภายในแท็กเองได้ ทำให้ข้อมูลมีความหมายในตัวมันเอง และสามารถเขียนโปรแกรมมาดึงข้อมูลไปใช้งานง่าย และคนทั่วไปก็อ่านได้เข้าใจ
- ใช้สำหรับการแลกเปลี่ยนข้อมูล (Data Exchange) ด้วยความที่เป็นไฟล์ข้อความธรรมดา ทำให้เอกสารเอ็กซ์เอ็มแอล เป็นภาษากลางที่สามารถใช้ได้ทุกแพลตฟอร์ม จึงไม่แปลกที่เราจะสามารถแลกเปลี่ยนข้อมูลข้ามแพลตฟอร์มได้
- เป็นรูปแบบข้อความในการสื่อสาร (Messaging Format) ระหว่างแอพพลิเคชั่นหรือโปรแกรม เอ็กซ์เอ็มแอลเป็นรูปแบบการสื่อสารระหว่างองค์ประกอบต่างๆ ตามแนวคิดของเว็บเซอร์วิส นอกจากเอ็กซ์เอ็มแอลแล้ว ยังมีมาตรฐานต่างๆที่เป็นส่วนสำคัญในการทำงานของเว็บเซอร์วิส เช่น โซบ หรือ ซิมเปิ้ลออปเจ็กต์แอคเซสโปรโตคอล และยูดีดีไอ หรือ ยูนิเวอร์ซัลเดสสคริปต์ชั่นดิสคัฟเวอรี่แอนด์อินทิเกรชั่น ซึ่งทั้ง โซบ และ ยูดีดีไอ ล้วนมีพื้นฐานมาจาก เอ็กซ์เอ็มแอล เช่นกัน
- ประโยชน์ในเชิงเทคโนโลยีอินเทอร์เน็ตและการพัฒนาเว็บ
- เป็นรากฐานของภาษาใหม่ๆในการพัฒนาเว็บ ภาษาใหม่ๆนี้ได้แก่ ภาษา เอ็กซ์เอชทีเอ็มแอล (XHTML), แม็ทเอ็มแอล (MathML คือ กลุ่มของแท็กเพื่อใช้นิยามเครื่องหมายในทางคณิตศาสตร์ชั้นสูง), วีเอ็มแอล (VML คือภาษาที่ใช้วาดรูปกราฟริกเพื่อแสดงผ่านเว็บบราวเซอร์) และ ดับบิวเอ็มแอล (WML คือ ภาษาที่ใช้ในการสร้าง แวพไซต์ (Wap Site)) เป็นต้น
- ใช้ในแวดวงธุรกิจแบบบีทูบี (B2B = Business to Business) กรณีนี้จะต้องใช้ภาษาเฉพาะอย่างเช่น ซีเอ็มเอ็กซ์แอล (cMXL = Commerce XML), เอ็กซ์ซีบีแอล (xCBL = XML Common Business Language) เป็นต้น โดยมีแท็กที่ใช้สนับสนุนการจัดการเกี่ยวกับแคตตาล็อกสินค้า และธุรกรรมเกี่ยวกับอีคอมเมิร์ส

2.2.3 โครงสร้างของภาษาเอ็กซ์เอ็มแอล

โครงสร้างของเอกสารเอ็กซ์เอ็มแอล ประกอบด้วย 3 ส่วน

- โปรล็อก (Prolog) แบ่งเป็น ดีแคร์เรชั่น (Declaration) และ ด็อกคิวเมนท์ไทพ์ดีแคร์เรชั่น (Document Type Declaration)
- บอดี้ (Body) เป็นส่วนของเนื้อเอกสารจริงๆ ได้แก่ข้อความหรือข้อมูลในเอกสาร และ แท็กที่นิยามข้อความหรือข้อมูลเหล่านั้น
- อีพิล็อก (Epilog) คือส่วนที่เป็นข้อความจำพวกคอมเมนท์ (Comment) และ พีไอ (PI) หรือโปรเซสซิ่งอินสตรักชั่น (Processing Instruction) จริงๆแล้วส่วนนี้ไม่จำเป็นต้องอยู่บริเวณล่างสุดของเอกสารดังในภาพ แต่คอมเมนท์ และ พีไอ จะแทรกอยู่ในส่วนบอดี้ของเอกสาร แต่ในภาพนี้แยกไว้ด้านล่างเพื่อให้เห็นว่าในเอกสาร เอ็กซ์เอ็มแอล หนึ่งๆ มีองค์ประกอบ 3 ส่วน



รูปภาพ 2-3 ส่วนประกอบของเอกสารเอ็กซ์เอ็มแอล

2.2.4 การแปลความหมายของเอ็กซ์เอ็มแอลด้วยเอ็กซ์เอ็มแอลพาร์เซอร์

ในการพัฒนาแอพพลิเคชั่นโดยมีการนำเอ็กซ์เอ็มแอล ไปใช้งานนั้น สิ่งหนึ่งที่ผู้พัฒนาจำเป็นต้องทำคือการพาส (Parse) เอกสารเอ็กซ์เอ็มแอล เนื่องจากโปรแกรมมองเห็นเอกสารเอ็กซ์เอ็มแอล เป็นเพียงอักขระธรรมดาเท่านั้น เราจึงต้องมีการเขียนโปรแกรมเพื่อให้ แอพพลิเคชั่น สามารถเข้าใจได้ว่าเอกสารเอ็กซ์เอ็มแอล ของเรามีการจัดโครงสร้างเป็นอย่างไร โดยเราเรียกโปรแกรมนั้นว่าพาร์เซอร์

การจำแนกชนิดของเอ็กซ์เอ็มแอลพาร์เซอร์ แบ่งได้เป็นหลายวิธี แต่หลักเกณฑ์ที่นิยมกันมี 2 วิธี คือ

1. เอ็กซ์เอ็มแอลพาร์เซอร์ แบบแวลิดเดติ้ง (Validating) และ แบบนอนแวลิดเดติ้ง (Non-Validating) ซึ่งหมายถึง พาร์เซอร์ที่มีการตรวจสอบคุณสมบัติเวลฟอร์มเนส และคุณสมบัติแวลิดของเอกสาร ส่วนเอ็กซ์เอ็มแอลพาร์เซอร์ แบบนอนแวลิดเดติ้ง หมายถึง เอ็กซ์เอ็มแอลพาร์เซอร์ที่ไม่มีการตรวจสอบคุณสมบัติแวลิด
2. เอ็กซ์เอ็มแอลพาร์เซอร์ แบที่รองรับ ด็อคคิวเมนท์ออบเจ็กต์โมเดลลิ่ง (DOM – Document Object Model) และแบบที่รองรับที่เรียกว่า แซก หรือ ซิมเปิ้ลเอพีไอฟอร์เอ็กซ์เอ็มแอล (SAX – Simple API for XML)

การแปลเอกสารโดยซิมเปิ้ลเอพีไอฟอร์เอ็กซ์เอ็มแอล

ซิมเปิ้ลเอพีไอฟอร์เอ็กซ์เอ็มแอลไม่ได้เป็นมาตรฐานของหน่วยงานใด แต่เป็นวิธีที่นักพัฒนากลุ่มหนึ่งคิดค้นขึ้น และนำไปใช้กันมากจนได้รับการยอมรับทั่วไป ปัจจุบันเอ็กซ์เอ็มแอลพาร์เซอร์ จำนวนมากล้วนแต่รองรับซิมเปิ้ลเอพีไอฟอร์เอ็กซ์เอ็มแอล กันแล้ว

ซิมเปิ้ลเอพีไอฟอร์เอ็กซ์เอ็มแอลจะไม่โหลดเอกสารเอ็กซ์เอ็มแอล เข้ามาในหน่วยความจำจึงไม่มีการสร้างภาพรวมของเอกสารไว้ก่อนว่าเอกสารนั้นมีโครงสร้างเป็นอย่างไร วิธีการของมันคือเมื่อแอพพลิเคชั่นมีการร้องขอมา มันจึงจะอ่านเอกสารเอ็กซ์เอ็มแอลไล่ตั้งแต่ต้นเอกสารและตัวพาร์เซอร์จะสร้างอีเวนต์ (Event) ให้กับจุดต่างๆที่สำคัญของเอกสารทุกๆจุด เช่นทุกๆ เอลเลอเมนท์เปิด เอลเลอเมนท์ปิด ทุกๆแอททริบิว เป็นต้น ลักษณะการทำอย่างเช่นนี้เรียกว่า อีเวนต์ไดรเวอร์พาร์เซอร์ (Event-Driven Parser)

ในทางปฏิบัติแล้วซิมเปิ้ลเอพีไอฟอร์เอ็กซ์เอ็มแอล มักไม่ได้มีการแปลเอกสารทั้งเอกสาร เมื่อเราได้ข้อมูลที่ต้องการแล้วเราก็จะหยุด วิธีการแปลแบบซิมเปิ้ลเอพีไอฟอร์เอ็กซ์เอ็มแอล จึงทำงานได้รวดเร็วและเขียนโปรแกรมได้ง่าย

ข้อดีของซิมเปิ้ลเอพีไอฟอร์เอ็กซ์เอ็มแอล

- ไม่เปลืองหน่วยความจำ เพราะไม่ได้โหลดทั้งเอกสารมาเก็บไว้เหมือน ด็อคคิวเมนท์ออบเจ็กต์โมเดลลิ่ง จึงเหมาะกับกรณีที่เอกสารเอ็กซ์เอ็มแอล มีขนาดใหญ่ และเครื่องเครื่องคอมพิวเตอร์มีหน่วยความจำไม่มาก

- ซิมเปิ้ลเอพีไอฟอร์เอ็กซ์เอ็มแอลเข้าถึงข้อมูลได้ง่ายกว่า ด็อคคิวเมนท์ออบเจ็กต์โมเดลลิ่ง เพราะการทำงานของ แซก จะมองเพียงจุดเริ่มต้นและจุดสิ้นสุดของเอกสารและของ เอลเลอเมนท์ และจะสนใจเฉพาะสิ่งที่ต้องการเท่านั้น จึงเหมาะกับกรณีต้องการใช้ข้อมูลจากเอลเลอเมนท์เพียงไม่กี่ตัวในเอกสาร ซึ่งถ้าใช้วิธี ด็อคคิวเมนท์ออบเจ็กต์โมเดลลิ่ง จะเป็นงานที่กินแรงเกินความจำเป็น
- สามารถเขียนโปรแกรมได้ง่ายกว่า เพราะมองข้อมูลในลักษณะเส้นตรง

ข้อเสียของซิมเปิ้ลเอพีไอฟอร์เอ็กซ์เอ็มแอล

- ไม่สามารถบอกความสัมพันธ์ของข้อมูลได้
- ไม่สามารถใช้ข้อมูลแบบสุ่มได้ เพราะไม่มีข้อมูลใดเหลืออยู่ในหน่วยความจำหลังจากที่ซิมเปิ้ลเอพีไอฟอร์เอ็กซ์เอ็มแอล ได้ทำการประมวลผลและสร้างอีเวนต์ที่มีความสัมพันธ์กันเสร็จแล้ว
- ยากที่จะสร้างฟังก์ชั่น ในการค้นหาข้อมูล
- อ่านได้เพียงอย่างเดียว ไม่สามารถเขียนหรืออัพเดตข้อมูลได้
- เป็นวิธีที่ไม่มีประสิทธิภาพถ้าต้องมีการเข้าถึงข้อมูลเดิมบ่อยครั้ง

2.3 ซิมเปิ้ลออบเจ็คต์แอคเซสโปรโตคอล (Simple Object Access Protocol)

เนื่องจากจุดประสงค์หลักของการใช้งานเว็บเซอร์วิส เราต้องการให้แอพพลิเคชั่น มีการทำงานกับแอพพลิเคชั่นที่ทำงานอยู่บนเครื่องอื่นโดยผ่านทางเครือข่าย ซึ่งเทคโนโลยีที่มีอยู่ปัจจุบันที่ใช้มีการสื่อสารระหว่างออบเจ็กต์ในระยะไกล หรือ อาร์พีซี (Remote Procedure Calls : RPC) เช่น ดีคอม (DCOM) , อีเจบี (EJB) หรือคอบร้า (COBRA) นั้นไม่ได้ถูกออกแบบมาใช้สำหรับโปรโตคอลเอชทีทีพี

เทคนิคอาร์พีซีของเทคโนโลยีที่กล่าวข้างต้นต่างก็มีปัญหาในด้านการนำมาใช้งานในแง่ของความเข้ากันได้ของการเรียกใช้งานข้ามเทคโนโลยี เนื่องจากเป็นเทคโนโลยีเฉพาะของแต่ละค่าย (ยกเว้น คอบร้า) ผู้พัฒนาระบบจะต้องพัฒนาโปรแกรมที่มีความซับซ้อน และยังมีปัญหาในส่วนของไฟร์วอล และพร็อกซี่เซิร์ฟเวอร์ ด้วยเนื่องจากโดยปกติเซิร์ฟเวอร์จะปิดการสื่อสารที่ไม่ใช่โปรโตคอลเอชทีทีพีออกไปเพื่อความปลอดภัยของระบบที่มีการติดต่อสื่อสารกับภายนอก

ดังนั้นทางเลือกของการสื่อสารที่จะนำมาใช้ในการทำบริการเว็บเซอร์วิส คือ ให้ทำงานอยู่บนโปรโตคอลเอชทีทีพี ซึ่งโซบ นอกจากจะทำงานบนโปรโตคอลเอชทีทีพีแล้ว ยังเป็นมาตรฐานเปิดที่จะทำให้สามารถติดต่อสื่อสารกับคอมพิวเตอร์ที่มีความแตกต่างทางระบบปฏิบัติการณ์, เทคโนโลยีรวมไปถึงภาษาที่ใช้ในการพัฒนา

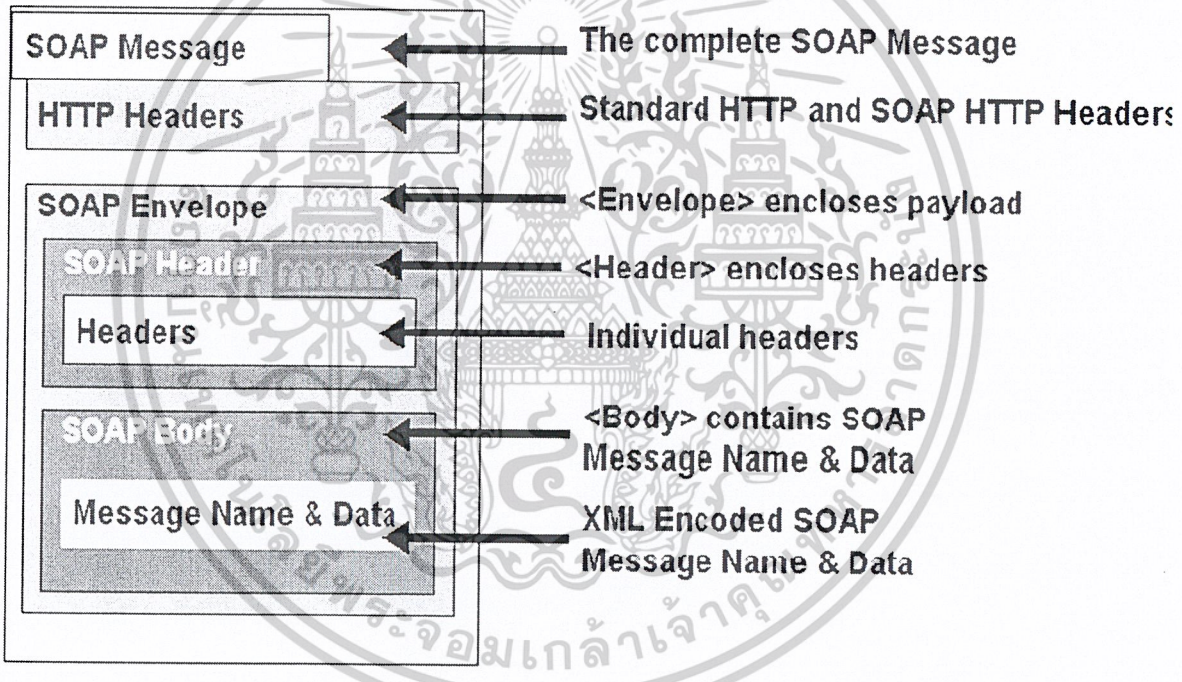
โซบ หรือ ซิมเปิ้ลออบเจ็กต์แอคเซสโปรโตคอล (SOAP = Simple Object Access Protocol) เป็นโปรโตคอลที่ใช้ เอ็กซ์เอ็มแอล เป็นพื้นฐาน เพื่อให้ซอฟต์แวร์คอมโพแนนท์ และแอพพลิเคชันสามารถติดต่อกันผ่านเอชทีทีพี ซึ่งเป็นมาตรฐานอินเทอร์เน็ตโปรโตคอลได้ ซึ่งข้อดีของโซบก็คือโซบไม่ขึ้นกับคอมโพแนนท์เทคโนโลยี และภาษาการเขียนโปรแกรมใดๆ สามารถเขียนได้ง่ายและขยายเพิ่มเติมได้

2.3.1 ส่วนประกอบของโซบ

โซบแมสเซจ ใช้ไวยากรณ์ของเอ็กซ์เอ็มแอล ในการสร้างประกอบด้วย 3 เอลเลอเมนท์มาตรฐาน คือ โซบเอนวิล๊อพ, โซบเฮดเดอร์, โซบบอดี้

โซบนั้นประกอบด้วยกัน 3 ส่วนคือ

1. โซบเอนวิล๊อพ (SOAP Envelop) จะเป็นส่วนสำหรับใช้ในการระบุสิ่งที่อยู่ในเอกสาร ว่าจะต้องจัดการอย่างไร และบอกถึงความจำเป็นในการใช้งาน
2. โซบเอนโค้ดดิ้งรูล (SOAP Encoding Rule) จะเป็นส่วนสำหรับกำหนดกลไกที่ใช้ในการแลกเปลี่ยนข้อมูล ว่ามีข้อตกลงยังไงในการแลกเปลี่ยนข้อมูล
3. โซบอาร์พีซีรีพรีเซนเทชั่น (SOAP RPC Representation) จะเป็นส่วนสำหรับนิยามรูปแบบรีโมท โพรซีเยอร์คอล (Remote Procedure Call) และ การตอบสนอง (Respone)



รูปภาพ 2-4 โครงสร้างของโซบ

โซบแมสเซจ จะต้องเป็นไปตามกฎนี้

- เอนวิล๊อพเป็นเอลเลอเมนท์ที่อยู่บนสุด ต้องมีเอลเลอเมนท์นี้เสมอ
- เฮดเดอร์อาจจะมีหรือไม่มีก็ได้ แต่ถ้ามีต้องเป็นชายเอลเลอเมนท์ (Child Element) แรกของเอนวิล๊อพ
- บอดี้ต้องเป็นชายเอลเลอเมนท์แรกของเอนวิล๊อพ หรือ เฮดเดอร์

ตัวอย่างของ โซบแมสเซจ ตัวอย่างนี้คือ GetLastTradePrice โซบรีเควส (SOAP Request) ที่ส่งไปยัง StockQuote เซอร์วิสประกอบด้วยสตริงพารามิเตอร์ ซิมโบ (Symbol) และ คืนค่าโฟลต (Float) กลับมากับ โซบเรสสปอน (SOAP Response) โซบเอนวิล๊อพเอลเลอเมนท์อยู่ที่จุดบนสุดของเอกสารเอ๊กซ์เอ็มแอล เอ๊กซ์เอ็มแอลเนมสเปซใช้เพื่อป้องกันการสับสนของโซบไอเด็นทิฟาย (SOAP Identifier)โซบรีเควสจะเป็นดังนี้

```
POST /StockQuote HTTP/1.1
    HOST:www. Stockquoteserver.com
    Content-Type: text/xml; charset="utf-8"
    Content-Length: nnnn
    SOAPAction: "Some-URI"

    <SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
    <m:GetLastTradePrice xmlns:m="Some-URI">
    <symbol>DIS</symbol>
    </m:GetLastTradePrice>
    </SOAP-ENV:Body>
    </SOAP-ENV:Envelope>
```

รูปภาพ 2-5 ตัวอย่างของโซบรีเควส

โซบเรสสปอนจะเป็นดังนี้

```
HTTP/1.1 200 OK
    Content-Type: text/xml; charset="utf-8"
    Content-Length: nnnn

    <SOAP-ENV:Envelope

    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
```

```
SOAP-encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
<m:GetLastTradePriceREsponse xmlns:m="Some-URI">
<Price>34.5</Price>
</m:GetLastTradePriceREsponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

รูปภาพ 2-6 ตัวอย่างของโซบเรสสปอน

นอกจากนั้นในโซบแมสเซจจะมีการใช้ เอ็กซ์เอ็มแอลเนมสเปซ ทุกๆ เอลเลอเมนท์ในเอกสารจะขึ้นต้นด้วยเนมสเปซซึ่งเนมสเปซจะถูกกำหนดโดยใช้เอ็กซ์เอ็มแอลเอนเอสแอททริบิว (xmlns attribute) ดังนี้

```
SOAP-Envelope
        xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
```

รูปภาพ 2-7 เอ็กซ์เอ็มแอลเอนเอสแอททริบิว

แต่หากไม่ต้องการให้ขึ้นต้นด้วยเนมสเปซทำได้ดังนี้

```
<Envelope
        xmlns="http://schemas.xmlsoap.org/soap/envelope/">
```

รูปภาพ 2-8 เอ็กซ์เอ็มแอลเอนเอสแอททริบิวที่ไม่ขึ้นด้วยเนมสเปซ

2.3.1.1 เอนวิล๊อพ (Envelope)

เอนวิล๊อพเอลเลอเมนท์เป็นเอลเลอเมนท์บนสุดของเอกสารเอ็กซ์เอ็มแอลที่ใช้แสดงแมสเซจ อาจประกอบด้วยแอททริบิว คือ เอนวิล๊อพเนมสเปซ และเอนโค้ดดิ้งสไตล์ (EncodingStyle)

- เอนวิล๊อพเนมสเปซ (Envelope Namespace) โซบแมสเซจจะแสดงเวอร์ชั่น โดยใช้เนมสเปซของเอนวิล๊อพเอลเลอเมนท์เนมสเปซ (Element Namespace) จะถูกนำไปใช้ขึ้นต้น เอนวิล๊อพ ,เฮดเดอร์ และบอดี้เอลเลอเมนท์ (Body Element)
- เอนโค้ดดิ้งสไตล์แอททริบิว (EncodingStyle Attribute) ใช้แสดงวิธีซีรีไรท์เซชั่น (Serialization) ของโซบแมสเซจ แอททริบิวนี้สามารถมีได้ในทุกเอลเลอเมนท์ และจะมีผลกับคอนเทน (Content) ของเอลเลอเมนท์นั้นและชายด์เอลเลอเมนท์ทั้งหมดของที่ได้ประกาศแอททริบิว

2.3.1.2 บอดี้ (Body)

เป็นส่วนของข้อมูลที่ใช้ในการแลกเปลี่ยน โดยทั่วไปข้อมูลจะเป็นการมาร์แชลอาร์พีซีคอล (Marshall RPC Call) และเออเรอรีพอร์ตชายด์เอลเลอเมนท์ (Error Report Child Element) ของบอดี้เอลเลอเมนท์จะถูกเรียกว่า บอดี้เอนทรี่ (Body Entry)

บอดี้เอนทรี่จะต้องเป็นไปตามกฎนี้

- บอดี้เอนทรี่จะต้องแสดงด้วยชื่อเต็มที่ประกอบด้วยเนมสเปซยูอาร์ไอ (Namespace URI) และ ชื่อของมัน (Local Name)
- โซบเอนโค้ดดิ้งสไตล์แอททริบิว (SOAP EncodingStyle Attribute) อาจจะมีได้ เพื่อแสดงถึงวิธี เอนโค้ด (Encode) ของบอดี้เอนทรี่นั้น

2.3.1.3 เฮดเดอร์ (Header)

เป็นส่วนเพิ่มเติมของแมสเซจสามารถประกอบด้วยอินฟอร์เมชั่นที่ระบุไปยังแอพพลิเคชั่น ส่วนเพิ่มเติมนี้อาจนำไปใช้อิมพลีเมนท์เป็นออเทนทิเคชั่น (Authentication), ทรานแซคชั่นแมนเนทเมนท์ (Transaction Management) เป็นต้น ทุก ๆ ชายเอลเลอเมนท์ของเฮดเดอร์เอลเลอเมนท์ จะถูกเรียกว่า เฮดเดอร์เอนทรี่ (Header Entry)

เฮดเดอร์เอนทรี่จะต้องเป็นไปตามกฎดังนี้

- เฮดเดอร์เอนทรี่จะต้องแสดงด้วยชื่อเต็มที่ประกอบด้วยเนมสเปซยูอาร์ไอ และ ชื่อของมัน
- อาจมีโซบเอนโค้ดดิ้งสไตล์แอททริบิวที่ใช้สำหรับเฮดเดอร์เอนทรี่
- โซบมัสอันเดอร์สแตนแอททริบิว (SOAP mustUnderStand Attribute) และ โซบแอคเทอร์
- แอททริบิว (SOAP actor attribute) จะมีหรือไม่ก็ได้ เพื่อใช้บอกว่าจะจัดการกับเอนทรี่ นั้นอย่างไรและโดยใคร

```
<soap:Header>
    <m:local xmlns:m="http://www.w3schools.com/local/"
                soap:actor="http://www.w3schools.com/appml/">
        <m:language>en</m:language>
        <m:currency>USD</m:currency>
    </m:local>
<soap:Header>
```

รูปภาพ 2-9 ตัวอย่างของโซบเฮดเดอร์

2.3.1.3.1 โซบแอคเตอร์แอททริบิว

โซบแมสเซจสามารถถูกส่งไปยังปลายทางผ่านตัวกลางของโซบ (SOAP Intermediary) ตัวกลางของโซบ คือ แอพพลิเคชั่นที่มีความสามารถทั้งรับและส่งต่อโซบแมสเซจ ในการส่งต่อตัวกลางจะต้องไม่ส่งต่อเฮดเดอร์เอลเลอเมนท์ไปให้กับแอพพลิเคชั่น โดยตัวกลางนี้จะถูกกำหนดเป็นยูอาร์ไอ ถ้าไม่มีแอททริบิวนี้ หมายถึง ผู้รับโซบแมสเซจเป็นปลายทางสุดท้าย

2.3.1.3.2 โซบอันเดอร์สแตนแอททริบิว

ใช้แสดงว่า เฮดเดอร์เอนทรี่นี้จำเป็นหรือเป็นเพียงออปชั่นสำหรับผู้รับที่จะนำไปประมวลผล ค่าของมันคือ “1” หรือ “0” ถ้าไม่มีการระบุจะมีค่าเท่ากับ “0” โดย “0” หมายถึงเป็นออปชั่นและ “1” หมายถึงจำเป็น โดยจะต้องประมวลผลให้ถูกต้องตามซีเมนติก (Semantic) หรือ ต้องไม่เฟว (Fail) เช่น แมสเซจเป็นส่วนหนึ่งของทรานแซคชั่น ถ้าปลายทางไม่รองรับทรานแซคชั่น จะต้องไม่ประมวลผลและส่งผลผิดพลาดกลับไปในกรณีที่มัสอันเดอร์สแตนแอททริบิว เป็น “1” แต่ถ้าเป็น “0” จะยังคงประมวลผลต่อไป

2.3.2 โซบฟลอเอลเลอเมนท์ (SOAP Fault Element)

ข้อความแสดงความผิดพลาดจากแอพพลิเคชั่นของโซบจะเก็บอยู่ในฟอลเอลเลอเมนท์ ซึ่งถ้ามีจะต้องปรากฏในบอดี้เอลเลอเมนท์เพียงครั้งเดียวในโซบแมสเซจ ตัวอย่างอาจเป็นดังนี้

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
        soap:encodingSytle="http://schemas.xmlsoap.org/soap/encoding/">
    <soap:Body>
        <soap:Fault>
            <faultcode>soap:MustUnderstand</faultcode>
            <faultstring>Mandatory Header error.</faultstring>
            <faultactor>http://www.wrox.com/heroes/endpoint.asp</faultactor>
            <detail>
                <w:source xmlns:w=" http://www.wrox.com/">
                    <module>endpoint.asp</module>
                    <line>203</line>
                </w:source>
            </detail>
        </soap:Fault>
            </soap:Body>
```

รูปภาพ 2-10 ตัวอย่างของโซบแมสเซจที่มีความผิดพลาด

โซบฟอลเอลเลอเมนท์มีเอลเลอเมนท์ย่อย ๆ ดังตารางนี้

Sub Element	Description
<faultcode>	โค้ดที่ระบุถึงการ error
<faultstring>	ข้อความการ error
<faultactor>	ใครเป็นสาเหตุของการ error
<detail>	ระบุอินฟอร์เมชั่นของการ error

ตาราง 2-1 ตัวอย่างของเอลเลอเมนท์ย่อยๆ

ค่าของฟอลโค้ด (Fault Code) สามารถมีค่าได้ดังตารางนี้

Error	Description
VersionMismatch	namespace ภายใน SOAP Envelope element ไม่ถูกต้อง
MustUnderstand	child element ของ Header element กับ mustUnderstand attribute ที่มีค่า “1” ผู้รับไม่รับรอง
Client	แมสเซจที่รูปแบบไม่ถูกต้อง หรือมีอินฟอร์เมชั่นไม่ถูกต้อง
Server	เกิดปัญหากับเซิร์ฟเวอร์ ไม่สามารถประมวลผลได้

ตาราง 2-2 ตัวอย่างของค่าฟอลโค้ด

2.3.3 โซบเอตโค้ดดิ้ง (SOAP Encoding)

โซบเอนโค้ดดิ้ง มีวิธีการแมพ (Map) จากไทพ์ของโปรแกรมมิ่งไปเป็นเอ็กซ์เอ็มแอล 2 วิธี คือ จากภายนอก โดยใช้ดับบิวเอสดีแอล ที่บอกถึงไทพ์ของข้อมูลที่รับหรือส่ง หรือใช้เอ็กซ์เอสไอไทพ์แอททริบิว (xsi:type Attribute) ในกรณีที่ภาษาที่ใช้ไม่รองรับ ดับบิวเอสดีแอล โดยทั้ง 2 วิธีจะใช้เอ็กซ์เอ็มแอลสครีม่าในการระบุไทพ์ ตัวอย่างของการใช้เอ็กซ์เอสไอจะเป็นดังนี้

```
<soap:Envelope xmlns:soap=" http://schemas.xmlsoap.org/soap/envelope/"
        soap:encodingSytle="http://schemas.xmlsoap.org/soap/encoding/"
        xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/1999/XMLSchema">
    <soap:Body>
      <m:MixedMessage xmlns:m="http://www.wrox.com/mix/">
        <param1 xsi:type="xsd:string">OU812</param1>
        <param2 xsi:type="xsd:integer">2001</param2>
        <param3 xsi:type="xsd:double">3.14159</param3>
      </m:MixedMessage>
    </soap:Body>
        </soap:Envelope>
```

รูปภาพ 2-11 ตัวอย่างของการใช้เอ็กซ์เอสแอล

2.3.4 โซบในเอชทีทีพี

ในการส่งโซบผ่านทางเอชทีทีพีจะต้องใช้คอนเทนไทพ์ (Content-Type) เป็นเท็กเอ็กซ์เอมแอล (Text/xml) แต่ในโซบรีเควสจะต้องมี เฮดเดอร์โซบแอคชั่น (Header SOAPAction) ภายในเอชทีทีพีเฮดเดอร์ (HTTP Header) ตัวโซบแอคชั่น (SOAPAction) จะเป็นตัวบอกให้เซิร์ฟเวอร์รู้ว่าเอชทีทีพีโพส (HTTP Post) นั้นเป็นโซบแมสเซจ และค่าของเฮดเดอร์ คือ ยูอาร์ไอที่แสดงถึงจุดหมายของโซบแมสเซจส่วนโซบเรสสปอนจะต้องมีสเตตัทโค้ด (Status Code) ตามมาตรฐานของเอชทีทีพี โดย 200 - 299 แสดงว่าสำเร็จ แต่ถ้าเรสสปอนแมสเซจเป็นการฟอล (Fault) แล้วสเตตัทโค้ดจะต้องเป็น 500 ซึ่งแสดงถึง อินเทอร์นัลเซิร์ฟเวอร์เออเรอ (Internal Server Error) ตัวอย่าง ของเรซสปอนที่มีสเตตัดโค้ดเป็น 500 อาจเป็นดังนี้

```
HTTP/1.1 500 Internal Server Error
Content-Type: text/xml
Content-Length: ###
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 soap:encodingSytle="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/1999/XMLSchema">
 <soap:Body>
      <soap:Fault>
          <faultcode>soap:VersionMismatch</faultcode>
          <faultstring>The SOAP namespace is incorrect.</faultstring>
          <faultactor>http://www.wrox.com/endpoint.asp</faultactor>
          <detail>
              <w:errorinfo xmlns:w=" http://www.wrox.com/">
                  <desc>The SOAP namespace was blank.</desc>
              </w:errorinfo>
          </detail>
      </soap:Fault>
 </soap:Body>
      </soap:Envelope>
```

รูปภาพ 2-12 ตัวอย่างของเรสสปอนที่มีสเตตัทโค้ดเป็น 500

2.3.5 โซบทูลคิต

โซบทูลคิต คือ เครื่องมือที่จะทำหน้าที่ในการประมวลผลโซบรีเควส และส่งเรสสปอนกลับไปให้ไคลเอนต์โดยโซบทูลคิต แต่ละตัวใช้ได้บนแพลตฟอร์มที่ต่างกัน รองรับเทคโนโลยีเว็บเซอร์วิสที่ต่างกัน บางตัวอาจจะรองรับดับบิวเอสดีแอล หรือ ยูดีดีไอ ในขณะที่บางตัวรองรับอย่างใดอย่างหนึ่งหรือไม่รองรับทั้งสองอย่าง และถึงแม้ว่าโซบจะเป็นอิสระต่อแพลตฟอร์ม แต่ในโซบทูลคิตบางตัวยังไม่สามารถทำงานข้ามผลิตภัณฑ์ (Interoperability) ได้ เนื่องจากรองรับเทคโนโลยีของเว็บเซอร์วิสไม่เหมือนกัน เช่น เอ็กซ์เอสไอในบางผลิตภัณฑ์จะบังคับให้โซบแมสเซจจะต้องระบุไทพ์ด้วยเอ็กซ์เอสไอ หากไม่มีก็จะเป็นฟอล (Fault) ซึ่งในปัจจุบันแต่ละผลิตภัณฑ์ก็พยายามพัฒนาให้สามารถทำงานด้วยกันได้

ชื่อ	แพลตฟอร์ม	แหล่งที่มา
4s4c	COM	http://www.4s4c.com
A SOAP for RPC NT Service	COM	http://www.whitemesa.com
Apache Axis	Java	http://xml.apache.org/axis
Apache SOAP	Java	http://xml.apache.org/soap
CapeConnect	Java	http://www.capeclear.com
Glue	Java	http://www.themindelectric.com
IBM Web Service Toolkit	Java	http://www.alphaworks.ibm.com
IdooXoap for Java and C++	Java, C++	http://www.zvon.org
Microsoft SOAP Toolkit	COM	http://www.microsoft.com
PacketSOAP	COM	http://www.packetsoap.com
SOAP::Lite	Perl	http://www.soaplite.com
Visual Studio .NET	COM	http://www.microsoft.com

ตาราง 2-3 ตัวอย่างของโซบทูลคิต

2.3.6 โครงสร้างภายในของโซบทูลคิตของไมโครซอฟต์

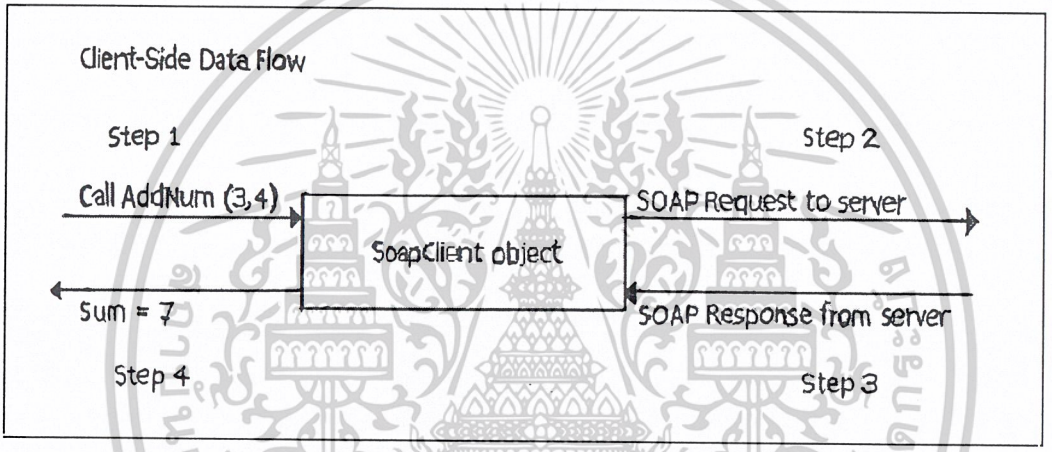
เว็บเซอร์วิสเดสสคริปต์ชั่นแลงเกวด หรือ ดับบิวเอสดีแอลเป็นรูปแบบหนึ่งของเอ็กซ์เอ็มแอลเพื่อใช้อธิบายว่าเซิร์ฟเวอร์นั้นมีเซอร์วิส และออปเปอร์เรชั่น (Operation) ภายในเซอร์วิสอะไรให้ใช้บ้าง ซึ่งออปเปอร์เรชั่นภายในเซอร์วิสนั้นจะอธิบายรูปแบบที่ไคลเอนต์ต้องใช้ในการเรียกออปเปอร์เรชั่น

เว็บเซอร์วิสเมต้าแลงเกวด หรือ ดับบิวเอสเอ็มแอลไฟล์ (WSML File) เป็นส่วนที่เพิ่มขึ้นมาจากไฟล์ดับบิวเอสดีแอลเพื่อใช้ในการเชื่อมโยงแต่ละออปเปอร์เรชั่นในเซอร์วิสกับคอมออบเจ็คต์ (COM Object) โดยดับบิวเอสเอ็มแอลไฟล์จะพิจารณาว่าคอมออบเจ็กต์ตัวใดที่จะต้องถูกโหลดขึ้นมาเมื่อมีการเรียกใช้ ออปเปอร์เรชั่นในเซอร์วิส

ทั้งไฟล์ดับบิวเอสดีแอล และดับบิวเอสเอ็มแอลจะต้องอยู่ที่เซิร์ฟเวอร์ โดยไคลเอนต์ที่ต้องการที่จะส่งโซบรีเควสไปที่เซิร์ฟเวอร์จะต้องได้รับดับบิวเอสดีแอลจากเซิร์ฟเวอร์ก่อน เพื่อที่จะรู้รูปแบบของโซบรีเควสในการติดต่อ

การสร้างโซบไคลเอนต์ออบเจคต์ (SOAP Client Object)

ทางฝั่งไคลเอนต์เมื่อแอพพลิเคชั่นส่งแมสเซจไปที่โซบไคลเอนต์ออบเจ็กต์ เพื่อเรียกใช้ ออปเปอร์เรชั่นหนึ่งๆนั้น (ตามรูปนี้ คือการบวกเลข 2 ตัว) โซบไคลเอนต์ออบเจ็กต์ จะทำการสร้างโซบรีเควสส่งไปที่เซิร์ฟเวอร์ เมื่อเซิร์ฟเวอร์ทำการประมวลผลเสร็จเรียบร้อยแล้ว ก็จะส่งผลลัพธ์ในรูปแบบของโซบเรซสปอนกลับมาให้ไคลเอนต์ หลังจากนั้นโซบไคลเอนต์ออบเจ็กต์ จึงจะทำการแปลและส่งแมสเซจที่เก็บผลลัพธ์ของออปเปอร์เรชั่น นั้นกลับไปให้แอพพลิเคชั่นตามไดอะแกรมด้านล่าง

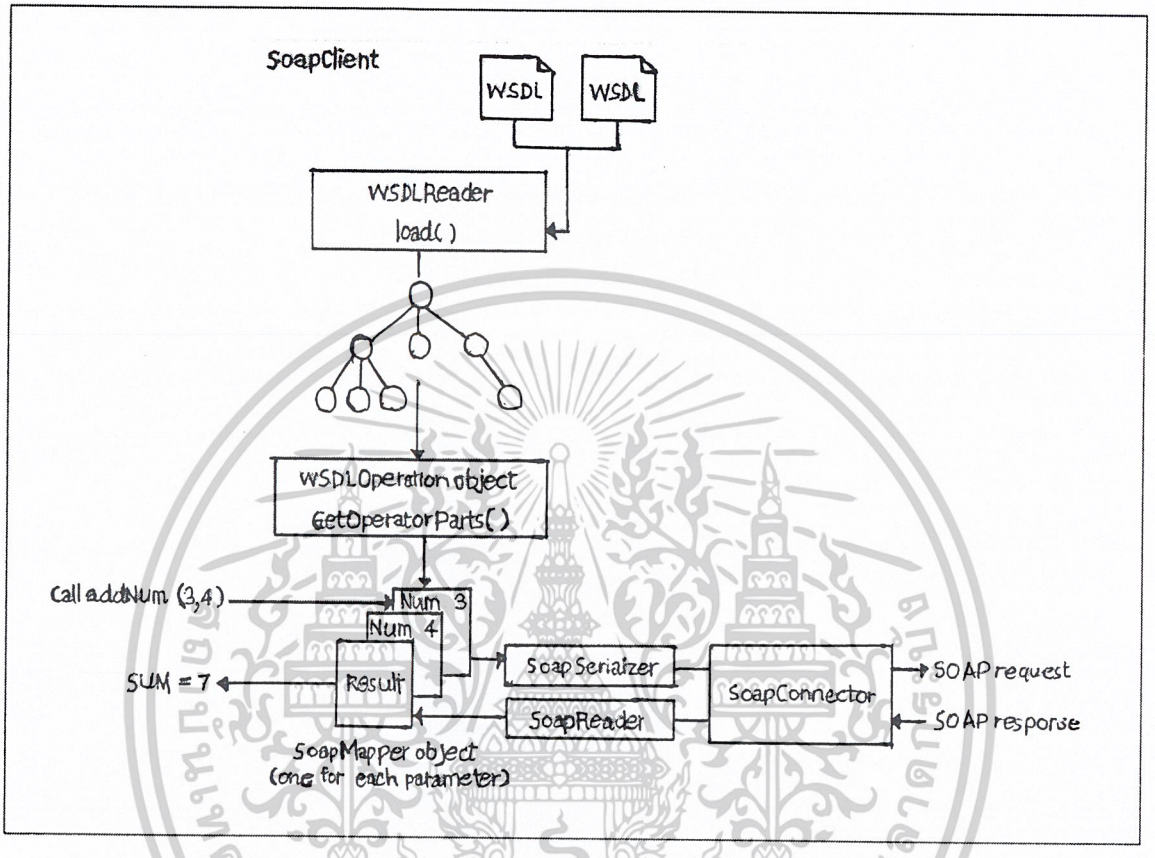


รูปภาพ 2-13 ตัวอย่างไดอะแกรมของการสร้างโซบไคลเอนต์ออบเจ็กต์

แอพพลิเคชั่นต้องอินสแทนทิเอตโซบออบเจ็กต์ (Instantiate Soap Object) ก่อนที่จะทำการเรียกใช้ออปเปอร์เรชั่นใดๆ (โซบไคลเอนต์ออบเจ็กต์ นี้ถูกสร้างเก็บไว้ในไฟล์ MSSOAP1.dll) โดยใช้เมธอด SoapClient.mssoapinit (ชื่อไฟล์ ดับบิวเอสดีแอล, ชื่อเซอร์วิส, ข้อมูลพอร์ต) โดยข้อมูลพอร์ตจะใช้ในการแบ่งแยกช่องทางที่ใช้ในการส่งโซบแมสเซจ

การประมวลผลภายในโซบไคลเอนต์ออบเจ็กต์

มีการทำงานดังรูป



รูปภาพ 2-14 ตัวอย่างของการทำงานการประมวลผลภายในโซบไคลเอนต์ออบเจ็กต์

ดับเอสดีแอลรีดเดอร์ออบเจ็กต์ (WSDL Reader Oject) จะทำการโหลดไฟล์ดับบิวเอสดีแอล และไฟล์ ดับบิวเอสเอ็มแอลเข้าไปไว้ในดอม (DOM) แล้วทำการวิเคราะห์โดยดับบิวเอสดีแอลรีดเดอร์ออบเจ็กต์ (WSDLReader Object) จะสร้างดับบิวเอสแอลโอเปอร์เรชั่นออบเจ็กต์ (WSDLOperation object) เพื่อทำการเรียกใช้งานออปเปอร์เรชั่น (ตามรูปนี้คือ AddNum(3, 4)) หลังจากนั้นดับบิวเอสแอลโอเปอร์เรชั่น จะเรียกเมธอด GetOperationParts เพื่อดึงรูปแบบแมสเซจของอินพุทและเอาท์พุทของออปเปอร์เรชั่นออกมาแล้วโซบไคล์เอนต์ (SoapClient) จะสร้างโซบแมพเปอร์ออบเจ็กต์ (Soap Mapper Objects) ให้กับแต่ละส่วนที่ได้จากเมธอด GetOperationParts และโหลดค่าที่ได้มาจากการเรียกใช้ออปเปอร์เรชั่น เข้าไว้ในออบเจ็กต์เหล่านี้ พอถึงขั้นตอนนี้ โซบซีรีไรส์ออบเจ็กต์(SoapSerializer Objects) จะสร้างโซบรีเควสแมสเซจ (SOAP Request Message) จาก โซบแมพเปอร์ออบเจ็กต์ (SoapMapper Objects) ที่เหมาะสมและส่งไปที่เซิร์ฟเวอร์

เมื่อเซิร์ฟเวอร์ประมวลผลเสร็จแล้วก็จะส่งโซบเรซสปอนกลับไปที่ไคลเอนต์ โซบไคลเอนต์ออบเจ็กต์จะทำการโหลดผลลัพธ์ของออปเปอร์เรชั่นไปเก็บในโซบแมพเปอร์ออบเจ็กต์ที่เหมาะสมแล้ว

ส่งผลลัพธ์กลับไปให้แอพพลิเคชั่นของผู้ใช้

การสร้างโซบเซอร์เวอร์ออบเจคต์ (SOAP Server Object)

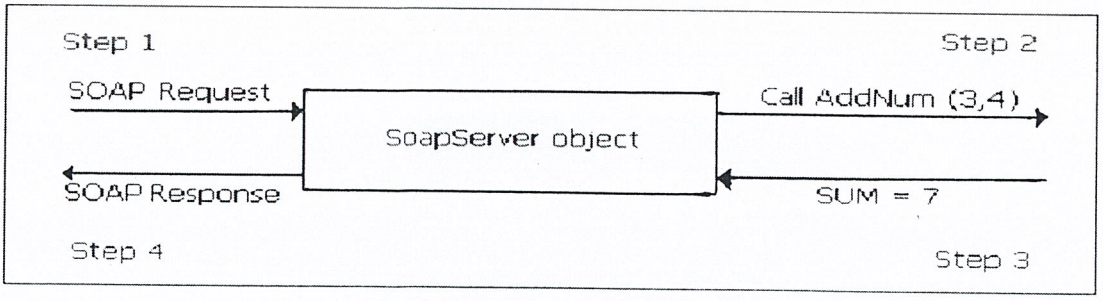
ส่วนทางฝั่งเซิร์ฟเวอร์เราสามารถกำหนดรูปแบบการรับโซบรีเควส (SOAP Listener) ได้ 2 รูปแบบ คือ อินเทอร์เน็ตเซิร์ฟเวอร์เอพีไอเซิร์ฟเวอร์ (Internet Server API Server) หรือ ไอเอสเอพีไอเซิร์ฟเวอร์ (ISAPI Server) และ แอกทีฟเซิร์ฟเวอร์เพจเซิร์ฟเวอร์ (Active Server Pages Server) หรือเอเอสพีเซิร์ฟเวอร์ (ASP Server)

```
<definitions>
 …
   <service name="DocSample1">
      <port name="DocSample1PortType" binding="tns:DocSample1Binding">
         <soap:address location=http://localhost/ DocSample1Test/ DocSample1.wsdl/>
      </port>
   </service>
…
</definitions>
```

รูปภาพ 2-15 ตัวอย่างของโซบเซิร์ฟเวอร์ออบเจ็คต์

การทำงานทางฝั่งเซิร์ฟเวอร์

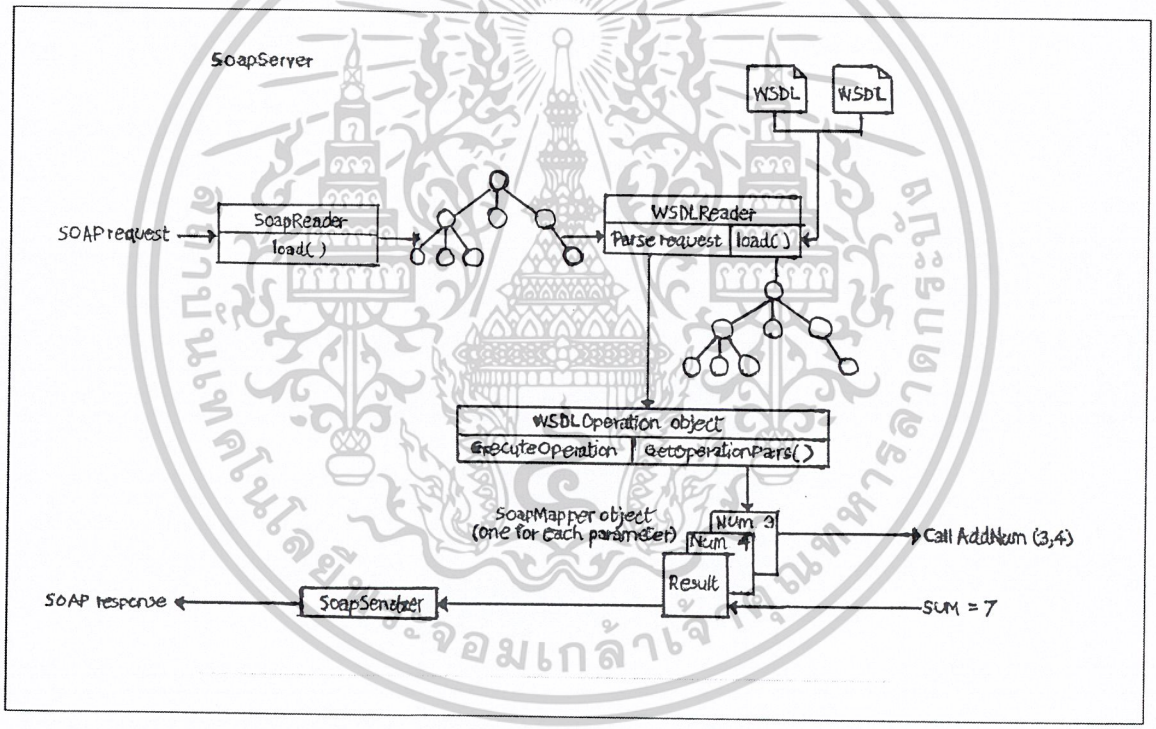
ไม่ว่าโซบรีเควสจะเรียกไปยังไอเอสเอพีไอเซิร์ฟเวอร์ หรือ เอเอสพีเซิร์ฟเวอร์ลิสเทนเนอร์ก็ตาม ข้อมูลที่รับส่งก็ยังเหมือนกัน เมื่อโซบเซิร์ฟเวอร์ออบเจ็กต์ ได้รับโซบรีเควสจากไคลเอนต์ ก็จะทำการแปลและเรียกไปยังเมธอด COM ที่ตรงกับออปเปอร์เรชั่นที่ต้องการ (ตามไดอะแกรมนี้คือเมธอด AddNum) หลังจากนั้นเมธอดที่ถูกเรียกใช้งานก็จะส่งผลลัพธ์กลับมาให้โซบเซิร์ฟเวอร์ออบเจ็กต์ซึ่งมันก็จะแปลงผลลัพธ์ให้อยู่ในรูปของ โซบเรสสปอนแล้วจึงส่งกลับไปที่ไคลเอนต์ ดังรูป (Server – Side Data Flow)



รูปภาพ 2-16 ตัวอย่างเซิร์ฟเวอร์ไซต์ดาต้าโฟล

การประมวลผลภายในโซบเซิร์ฟเวอร์ออบเจ็กต์

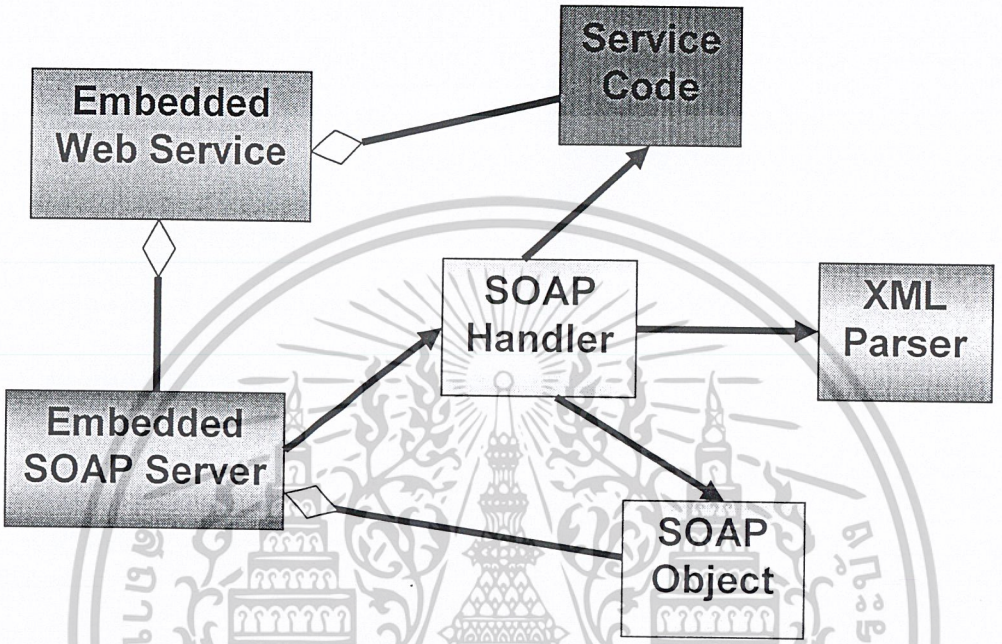
เมื่อได้รับโซบรีเควสจากไคลเอนต์แล้ว โซบรีดเดอร์ออบเจ็กต์ (SOAPReader Object) ก็จะโหลดรีเควสแมสเซจไปไว้ในดอมสตักเตอร์ (DOM Struture) หนึ่ง ในขณะที่ดับบิวเอสดีแอลรีดเดอร์ (WSDLReader) จะโหลดไฟล์ดับบิวเอสดีแอล และไฟล์ดับบิวเอสเอมแอลไปไว้ในอีกดอมสตรักเตอร์หนึ่งตัว ดับบิวเอสดีแอลรีดเดอร์จะแปลรีเควสและสร้างดับบิวเอสดีแอลออปเปอร์เรชั่นออบเจ็กต์ขึ้นเพื่อใช้ในรีเควสออปเปอร์เรชั่น หลังจากนั้น ดับบิวเอสดีแอลออปเปอร์เรชั่นออบเจ็กต์ จะเรียกใช้เมธอด GetOperationParts เพื่อดึงรูปแบบแมสเซจของอินพุทและเอาท์พุทของออปเปอร์เรชั่นออกมา แล้วโซบเซอร์เวอร์จะสร้างโซบแมพเปอร์ออบเจ็กต์ให้กับแต่ละส่วนที่ได้จากเมธอด GetOperationParts และโหลดค่าที่ได้จากการเรียกใช้ ออปเปอร์ชั่น เข้าไว้ในออบเจ็กต์เหล่านี้ พอถึงขั้นตอนนี้ โซบเซิร์ฟเวอร์ออบเจ็กต์จะทำการเรียกไปที่คอมเมธอด (COM method) ที่ตรงกับที่ร้องขอเข้ามา ดังรูป



รูปภาพ 2-17 ตัวอย่างของการทำงานการประมวลผลภายในโซบเซิร์ฟเวอร์ออบเจ็กต์

2.4 การออกแบบเว็บเซอร์วิสบนระบบฝังตัว จากโครงงานปี 2545

2.4.1 โครงสร้างของเว็บเซอร์วิสบนระบบฝังตัว

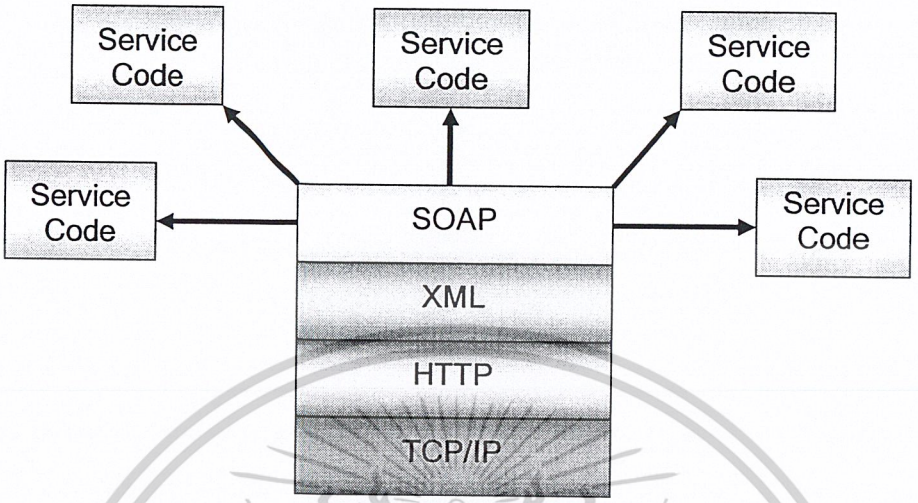


รูปภาพ 2-18 คอนเท็กซ์ไดอะแกรม (Context Diagram) ของเว็บเซอร์วิสบนระบบฝังตัว

สำหรับโครงสร้างภายในเว็บเซอร์วิสนี้มีส่วนที่สำคัญ คือ

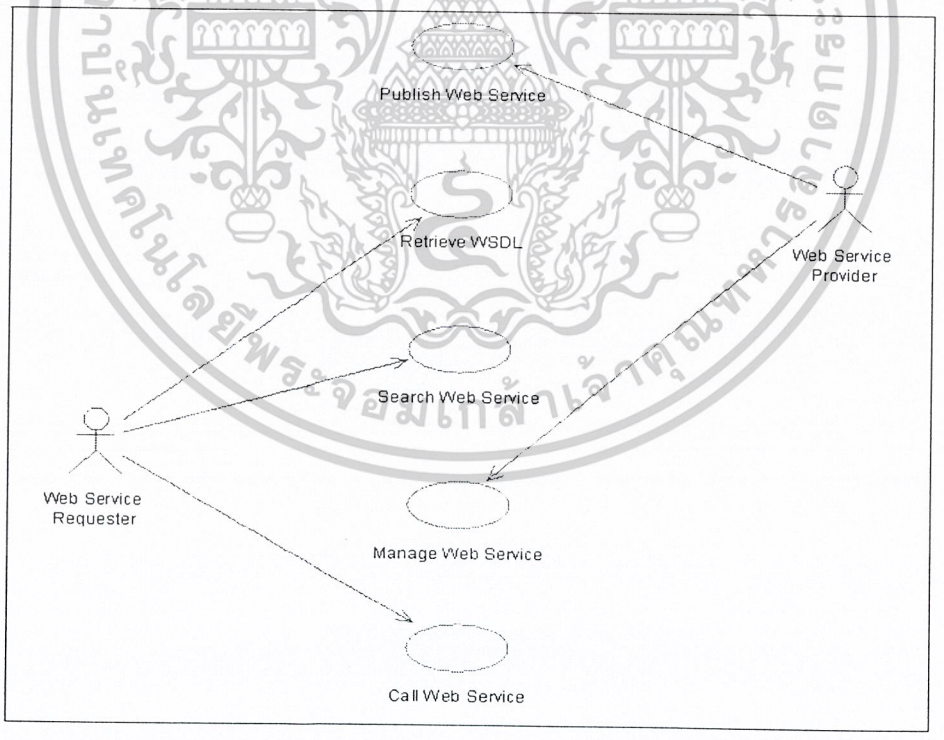
- โซบเซิร์ฟเวอร์บนระบบฝังตัว (Embedded Server) เป็นส่วนที่จัดการเฮดเดอร์เอชทีทีพีและการติดต่อผ่านทางโพรโตคอลทีซีพีไอพี
- โซบแฮนเดิลเลอร์ (SOAP Handler) เป็นส่วนที่ทำหน้าที่จัดการโซบแมสเซจ ซึ่งจะแบ่งเป็น 2 ส่วนคือ ส่วนที่ทำหน้าที่จัดการโซบแมสเซจที่ได้รับมาเมื่อมีการเรียกใช้เซอร์วิสโดยถอดโซบแมสเซจ ให้เป็นรายละเอียดและข้อมูลอินพุทในการเรียกใช้เซอร์วิส และส่วนที่ทำหน้าที่จัดการสร้างโซบแมสเซจเพื่อตอบรับการเรียกใช้เซอร์วิสโดยโดยนำเอาผลลัพธ์ที่ได้จากไฟล์เอาท์พุทที่ได้จากการเรียกเซอร์วิสมาเป็นโซบแมสเซจ
- เซอร์วิสโค้ด (Service Code) เป็นตัวแอพพลิเคชั่นที่อยู่บนตัวอุปกรณ์คอม 86ซึ่งจะถูกเรียกใช้งานโดยส่วนของโซบแฮนเดิลเลอร์
- โซบออปเจ็กต์ (SOAP Object) เป็นส่วนที่ทำหน้าที่เสมือนเป็นส่วนประกอบของโซบแมสเสจซึ่งจะถูกเรียกใช้โดยโซบแฮนเดิลเลอร์และเป็นส่วนหนึ่งของโซบเซิร์ฟเวอร์
- เอ็กซ์เอ็มแอลพาร์เซอร์ (XML Parser) จะทำหน้าที่ในการแปลเอกสารเอ็กซ์เอ็มแอล โดยที่จะถูกโซบแฮนเดอร์เลอร์ นั้นทำการเรียกใช้โซบแฮนเดอร์เลอร์ จะทำหน้าที่ในการจัดการโซบ โดยจะถูกเรียกใช้งานโดยโซบเซิร์ฟเวอร์ในระบบฝังตัว

2.4.2 โครงสร้างทางสถาปัตยกรรมของเว็บเซอร์วิสบนระบบฝังตัว



รูปภาพ 2-19 โครงสร้างทางสถาปัตยกรรมของเว็บเซอร์วิสบนระบบฝังตัว

2.4.3 ยูสเคสไดอะแกรมของระบบ (Use-Case Diagram)



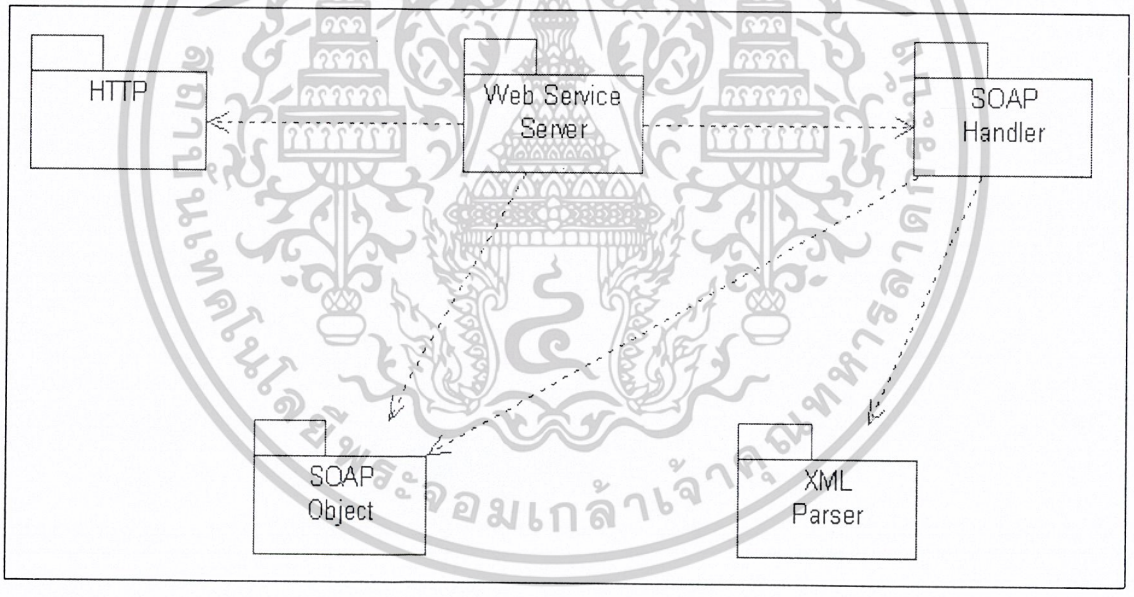
รูปภาพ 2-20 ยูสเคสไดอะแกรมของระบบ (Use-Case Diagram)

การร้องขอใช้บริการเว็บเซอร์วิส

Use Case	Call Web Service
ผู้ใช้	ผู้ให้บริการเว็บเซอร์วิส
เงื่อนไข	ผู้ใช้บริการเว็บเซอร์วิสจะต้องร้องขอตามข้อกำหนดในเอกสารดับบิวเอสดีแอลที่ได้มาจากตัวยูดีดีไอ และแอพพลิเคชั่นของผู้ใช้จะต้องสนับสนุนการทำงานของของโปรโตคอลโซบ และ เอชทีทีพี
ข้อมูลที่เกี่ยวข้อง	เมื่อได้รับเอชทีทีพีรีเควส มา จากนั้นแอพพลิเคชั่นโซพพาร์เซอร์จะทำการแปลงให้อยู่ในรูปโซบแมสเซจแล้วค่อยทำการดึงรายละเอียดในการเรียกใช้บริการ เพื่อนำมาใช้ในการติดต่อกับตัวบริการนั้นๆ

ตาราง 2-4 รายละเอียดของการร้องขอใช้บริการเว็บเซอร์วิส

2.4.4 คอมโพเนนท์ไดอะแกรมของระบบ (Component Diagram)



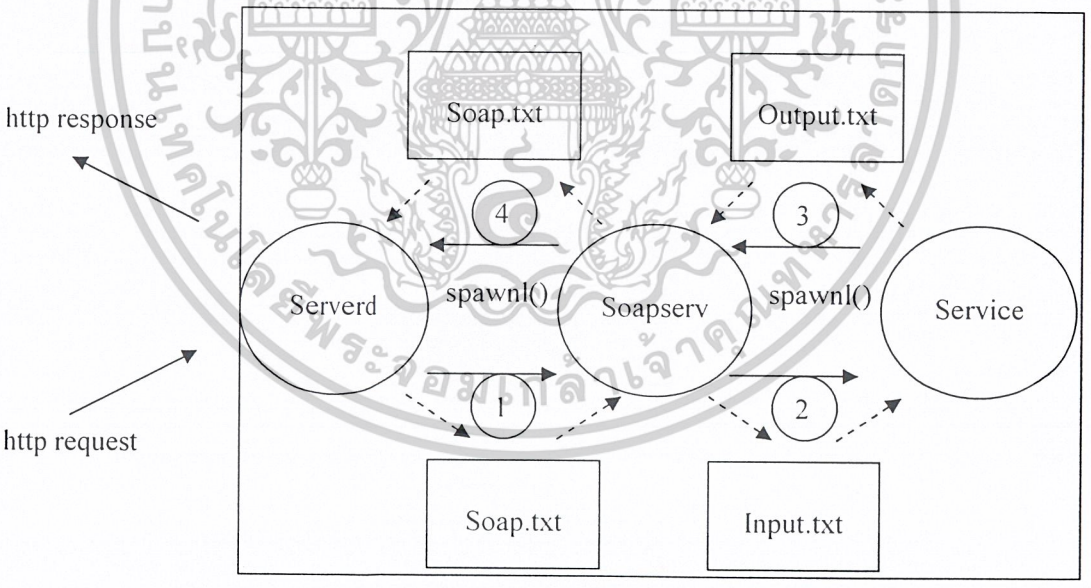
รูปภาพ 2-21 แสดงคอมโพเนนท์ไดอะแกรม

2.4.5 กลไกการทำงานของระบบ

การทำงานเว็บเซอร์วิสจะประกอบขึ้นจาก 3 ส่วน ได้แก่

1. **Serverd.exe** ทำหน้าที่เป็นเซิร์ฟเวอร์โพรเซส ที่คอยรับรีเควส จากการขอใช้บริการผ่านพอร์ต 80 แล้วทำการสื่อสารข้อมูลตามโปรโตคอลเอชทีทีพี แล้วได้เอาท์พุตเป็นเอกสารโซบ จากนั้นก็รอผลลัพธ์จาก Soapserv.exe เพื่อส่งผลลัพธ์กลับไปยังผู้ที่ร้องขอต่อไปในขั้นตอนสุดท้าย
2. **Soapserv.exe** ทำหน้าที่รับเอกสารโซบ ที่ได้จาก Serverd.exe แล้วทำการดีซีเรียลไรท์ เอกสารโซบ ให้ได้เป็นข้อมูลของเซอร์วิสที่เรียกใช้ และพารามิเตอร์สำหรับการทำเซอร์วิสนั้นๆ จากนั้นก็รอผลลัพธ์จากเซอร์วิสแอพพลิเคชั่น เพื่อนำส่งต่อไปให้ Serverd.exe ทำการส่งต่อไป
3. **Service application** เป็นแอพพลิเคชั่นที่มีฟังก์ชันการทำงานที่ผู้ร้องขอนั้น ต้องการเข้ามาขอใช้บริการ โดยแอพพลิเคชั่นนี้จะรับค่าพารามิเตอร์ในรูปแบบของการทำรีโมทโพรซีเยอร์คอล (Remote Procedure Call) มาทำการประมวล จากนั้นผลลัพธ์ที่ได้ก็จะถูกส่งไปยัง Soapserv.exe เพื่อทำการห่อหุ้มเอกสารกลับให้เป็นรูปแบบของเอกสารโซบ เพื่อส่งผลลัพธ์กลับไป

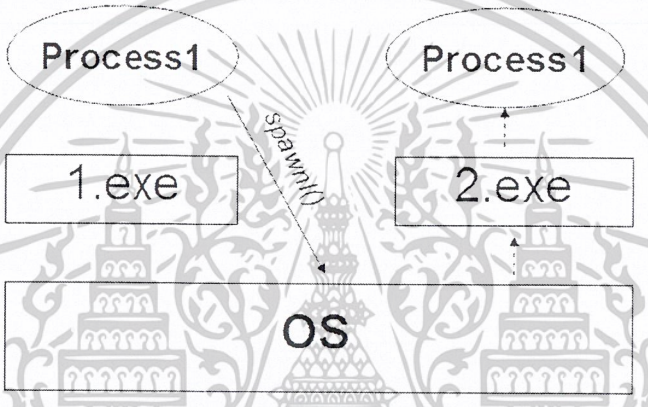
ขั้นตอนการทำงานของระบบเว็บเซอร์วิสแบบเดิม มีดังรูปต่อไปนี้



รูปภาพ 2-22 ขั้นตอนการทำงานของระบบเว็บเซอร์วิสบนระบบฝังตัวของโครงงานรุ่นที่ผ่านมา

การเรียกให้แอพพลิเคชั่นอื่นทำงานต่อ ด้วยฟังก์ชัน spawnl()

เป็นฟังค์ชันที่ใช้ในการเรียกแอพพลิเคชั่นอื่นขึ้นมาทำงานเป็น child process โดยเรียก path ของแอพพลิเคชั่นที่ต้องการ แล้วตามด้วย argument ให้ด้วย ทำให้เหมือนการเรียกใช้ผ่าน shell หรือ dos โดยก่อนที่ระบบปฏิบัติการจะสร้างโพรเซสให้แอพพลิเคชั่นที่ถูกเรียกนั้นทำงาน ก็จะทำการบันทึกค่าตัวแปรแบบโลคอล (Local variable) และสถานะของโพรเซสที่ทำการเรียก เช่น รีจิสเตอร์ของซีพียู เก็บไว้ในสแตกของระบบ จากนั้นจึงค่อยสร้างโพรเซสของแอพพลิเคชั่นใหม่ขึ้นมาให้ทำงานเหมือนการเรียกแอพพลิเคชั่นแบบปกติตามคำสั่งที่โพรเซสที่ทำการเรียกส่งมาให้ และหลังจากที่โพรเซสของแอพพลิเคชั่นที่ถูกเรียกขึ้นมานั้นทำงานจนจบแล้ว ระบบปฏิบัติการก็จะนำค่าของตัวแปรและสถานะต่างๆของโพรเซสก่อนหน้า มาอยู่ในระบบเพื่อทำงานต่อจากจุดที่เรียกคำสั่ง spawnl() ต่อไป



รูปภาพ 2-23 การเรียกแอพพลิเคชั่นอื่นโดยใช้คำสั่ง spawnl()

2.4.6 ข้อเสียของลักษณะการทำงานแบบเดิม

- สามารถทำงานได้ทีละ 1 รีเควส ในการทำงานในช่วงเวลาหนึ่งๆ เท่านั้น
- รีเควสอื่นๆ จะเข้ามาทำงานได้หลังจาก request แรกได้รับการประมวลผลเรียบร้อยแล้วเท่านั้น
- หากโพรเซสใด มีการรอหรือทำการประมวลผลนานๆ ทั้งกระบวนการก็จะต้องหยุดรอที่จุดนั้น ทำให้ไม่สามารถรับรีเควสอื่นเข้ามาทำงานได้
- หากโพรเซสใด เกิดความผิดพลาดในการทำงาน (corrupt) ขึ้นแล้ว โพรเซสอื่นทั้งหมดจะไม่สามารถทำงานต่อได้เลย
- ไม่สามารถลำดับความสำคัญ (priority) ของโพรเซสที่จะทำงานได้ ไม่ว่าเป็นงานที่สำคัญมากแค่ไหนก็ต้องรอให้งานที่ทำอยู่ก่อนหน้าทำเสร็จก่อน (non-preemptive)
- ไม่มีการส่งข้อมูลข้ามโพรเซสกัน โดยถ้าจะติดต่อกันระหว่างโพรเซสนั้น ต้องทำการบันทึกข้อมูลที่จะส่งให้กันลงในไฟล์ ทำให้เกิดโอเวอร์เฮดในการทำสูง

2.5 ระบบเรียลไทม์ (Real-Time System)

2.5.1 ความหมายของเรียลไทม์

ระบบเรียลไทม์ หมายถึง ระบบที่ทำงานเพื่อให้ได้ผลลัพธ์ โดยมีเงื่อนไข เวลา (time) และการคาดคะเนเวลาที่ใช้ในการทำงานได้ (predictable manner) โดยงานแต่ละงานจะต้องถูกทำให้เสร็จภายในระยะเวลาที่กำหนดไว้ และหากมีงานหลายๆงานเกิดขึ้นพร้อมๆกันแล้ว จะต้องมีการจัดแบ่งเวลาที่ใช้ในการประมวลผลของแต่ละงาน เพื่อให้เสร็จภายในระยะเวลาที่ต้องการ

2.5.2 ลักษณะของการทำงานแบบเรียลไทม์

- โต้ตอบกับระบบภายนอกได้ตลอดเวลา
- สามารถทำงานโดยรู้เงื่อนไขของเวลา
- ทำงานบนระบบที่มีทรัพยากรจำกัด

2.5.3 ประเภทของระบบเรียลไทม์

Hard Real-time : ระบบที่ทำงานโดยต้องการผลตอบสนองตามระยะเวลาที่กำหนด หากใช้เวลาเกินจากเส้นตายที่กำหนด การทำงานนั้นก็จะถือว่าทำงานผิดพลาด หรือล้มเหลวทันที

Soft Real-time : ระบบที่ไม่เกิดความเสียหายหรือล้มเหลวทันที หากไม่เสร็จทันตามระยะเวลาที่กำหนด แต่ระบบจะทำงานให้เสร็จหรือให้ได้ผลลัพธ์โดยเร็วที่สุดเท่าที่จะเป็นไป

2.5.4 ประโยชน์ของระบบเรียลไทม์

1. ระบบสามารถทำงานหลายๆ งานได้พร้อมๆ กัน
2. ระบบสามารถทำงานที่มีความสำคัญทางด้านเวลาสูงกว่า ให้ได้ผลลัพธ์ก่อนที่จะทำงานที่มีความสำคัญทางด้านเวลาต่ำกว่า
3. หากเกิดงานที่มีความสำคัญทางด้านเวลาสูงกว่า ระบบก็จะสามารถทำงานๆนั้นได้ทันที

2.6 ระบบปฏิบัติการแบบเรียลไทม์ (Real-Time Operating System)

เป็นระบบปฏิบัติการที่มีความสามารถในการทำงานแบบเรียลไทม์ ที่รองรับคุณสมบัติของการทำงานบนเงื่อนไขของเวลา โดยจะแตกต่างจากระบบปฏิบัติการทั่วไปโดยระบบปฏิบัติการแบบเรียลไทม์นี้จะมีการรับประกันระยะเวลานานที่สุดที่ระบบจะทำงานให้ได้ (worst case latency) ตัวอย่างความแตกต่าง เช่นหากเป็นระบบปฏิบัติการทั่วไปแล้ว หากเกิด interrupt จากภายนอกขึ้น ก็จะถูกนำไปใส่ต่อไว้ใน queue แล้วจะทำงานหลังจากทำการประมวลผล interrupt ตัวอื่นใน queue ก่อนหน้านี้จนเสร็จสิ้นก่อน

แต่หากเป็นระบบปฏิบัติการแบบเรียลไทม์ หาก interrupt ที่เกิดขึ้นนั้น มีความสำคัญกว่า งานหรือ interrupt ที่ทำอยู่ในตอนนั้น ก็จะสามารถได้ทำงานก่อนทันที

2.6.1 เคอร์เนลของระบบปฏิบัติการแบบรียลไทม์ (Real-Time Kernel)

เคอร์เนล (Kernel) เป็นส่วนกลางของระบบปฏิบัติการ ที่จะถูกนำขึ้นมาทำงานบนเมโมรี่ ก่อนเป็นอันดับแรก และจะอยู่ในเมโมรี่ ตลอดเวลา โดยจะทำหน้าที่ในการจัดการโปรเซส รวมถึงการจัดการการใช้งานเมโมรี และการติดต่อดิสก์และ I/O ด้วย

สำหรับเคอร์เนลของระบบปฏิบัติการแบบเรียลไทม์นั้นจะต้องมีความสามารถดังต่อไปนี้

Service	Specification
Task Management	Preemptive Kernel Context Switching Mutual Exclusion Deadlock Embrace
Synchronization	Event Flags
Intertask Communication	Message Mailboxes Message Queues
Interrupt Management	Interrupt Latency Interrupt Response Interrupt Recovery
Time Management	Clock Tick Time Delay
Memory Management	Memory Control Block Memory Partitioning

ตาราง 2-5 ความสามารถของเคอร์เนลของระบบปฏิบัติการแบบเรียลไทม์

2.6.2 ซอฟต์แวร์ที่เลือกมาทำการศึกษาการทำงานแบบเรียลไทม์

Micro C / OSII : Real-time Kernel

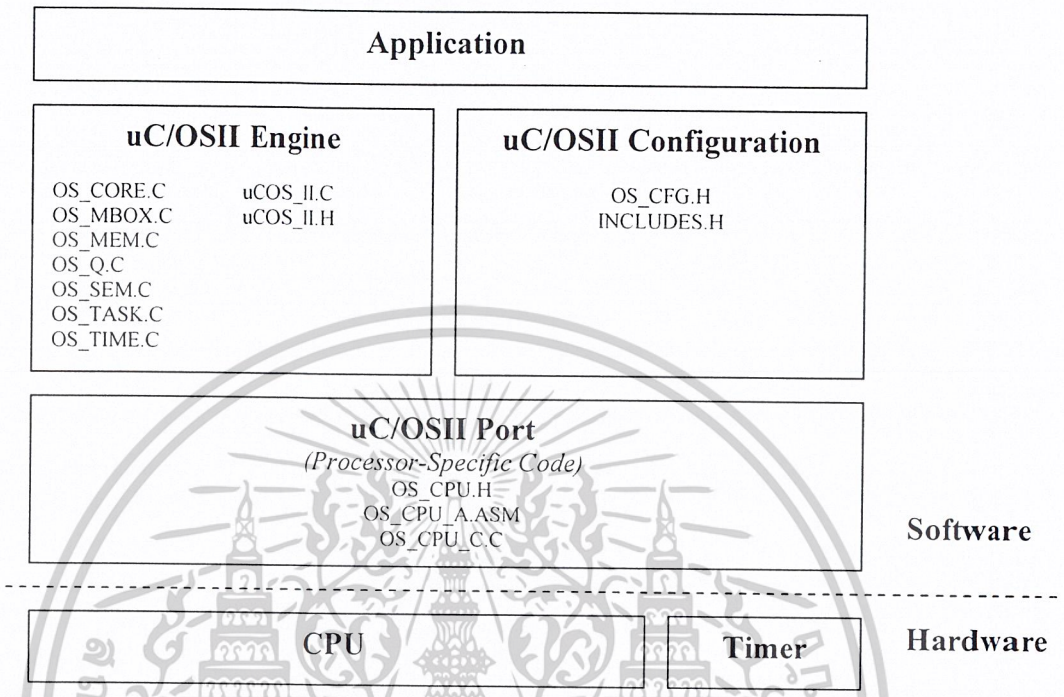
เป็นซอฟต์แวร์ประเภท opensource ซึ่งสามารถนำมาศึกษาและพัฒนาต่อได้โดยไม่ต้องเสียค่าใช้จ่าย มีลักษณะเป็นซอร์สโคดที่ประกอบไปด้วยส่วนของ engine ที่ทำหน้าที่เป็น real-time kernel และส่วนของฟังค์ชันที่สร้างขึ้นเพื่อให้นักพัฒนาสามารถใช้ในการจัดการกับงานต่างๆ ที่พัฒนาขึ้นได้เองอย่างครบถ้วน

Micro C / OSII มีความสามารถดังนี้

Source code	มีระบบการพัฒนาซอร์สโคดที่ดี มีการจัดทำเอกสารประกอบอย่างครบถ้วน และซอร์สโคดมีความง่ายต่อการศึกษา และพัฒนาต่อ
Portable	ซอร์สโคดได้ถูกพัฒนาขึ้นตามมาตรฐาน ANSI C ทำให้มีความง่ายในการนำไปพัฒนาและสร้างแอพพลิเคชั่นลงบนแพลตฟอร์มต่างๆ โดยทางผู้พัฒนา Micro C/OSII นั้นได้เตรียมเขียนส่วนของโปรแกรมภาษาแอสเซมบลี แยกเฉพาะตามไมโครโปรเซสเซอร์แต่ละแบบไว้แล้ว เพื่อนำไป port ได้ทันที
ROMable	สามารถนำพัฒนาเป็นส่วนหนึ่งของแอพพลิเคชั่นได้
Scalable	ใช้ทรัพยากรในระบบน้อย
Preemptive	เป็นแบบ fully preemptive real-time kernel ซึ่งทำให้สามารถประมวลผลงานที่มีความสำคัญสูง ให้เสร็จก่อนได้
Multitasking	สามารถรองรับงานได้ 64 task พร้อมๆกัน โดยมี 8 task เป็น task สำหรับระบบ ดังนั้นจึงสามารถมีงานที่นักพัฒนาสามารถสร้างขึ้นได้ 56 task
Deterministic	สามารถทราบถึงระยะเวลาในการประมวลผล task หนึ่งๆได้ แม้ว่าจะมีงานจำนวนมากทำงานอยู่ก็ตาม
Task Stacks	ใน stack ของแต่ละ task นั้นสามารถกำหนดเองได้ และสามารถตรวจสอบการใช้งาน stack ของแต่ละ task ได้อีกด้วย
Services	มี service ที่จำเป็นต่อการใช้งานอย่างครบถ้วน เช่น mailbox , queue , semaphore , fixed-sized memory partition เป็นต้น
Interrupt Management	สามารถเข้าแย่งการทำงานได้ หากมี priority สูงกว่า priority ของ task ที่ทำงานอยู่ในตอนนั้น แต่หากมี priority ต่ำกว่า และยังสามารถทำ nested interrupt ได้ 255 level
Robust and Reliable	มีความเชื่อถือได้ในเสถียรภาพ เนื่องจากถูกพัฒนามาเป็นระยะเวลานาน

ตาราง 2-6 ความสามารถของ Micro C/OSii

2.6.3 โครงสร้างของซอร์สโค้ดของ Micro C/OSii



รูปภาพ 2-24 โครงสร้างของซอร์สโค้ดของ Micro C/OSii

2.7 คอม86 (COM86)

ชุดพัฒนาคอม86 เป็นชุดเครื่องมือสำหรับใช้ในการพัฒนาอุปกรณ์ทางด้านระบบฝังตัวที่อยู่บนพื้นฐานการพัฒนาโดยการใช้แพลตฟอร์มที่มีไมโครโปรเซสเซอร์ตระกูล x86 ซึ่งเป็นที่คุ้นเคยกับผู้ใช้งานในปัจจุบันเป็นส่วนประกอบหลักในการทำงาน

2.7.1 จุดเด่นของคอม86

2.7.1.1 ใช้ไมโครคอนโทรลเลอร์ที่มีความสามารถสูง

ภายในชุดพัฒนาคอม86 มีบอร์ดคอม86 ซึ่งได้เลือกใช้ไมโครคอนโทรลเลอร์ Am186CC ของบริษัทเอเอ็มดี (AMD) ซึ่งเป็นไมโครคอนโทรลเลอร์ขนาด 16 บิตทำงานอยู่ที่ความเร็ว 24 เมกกะเฮิร์ต (MHz) ทำให้มีความสามารถในการประมวลผลต่างๆเพิ่มมากขึ้นเมื่อเทียบกับการพัฒนาโดยใช้ไมโครคอนโทรลเลอร์ขนาด 8 บิตที่นิยมใช้กันอยู่ในปัจจุบัน นอกจากนี้ภายในตัวไมโครคอนโทรลเลอร์เองยังได้รวมเอาอุปกรณ์พื้นฐานต่างๆที่จำเป็นในการใช้งาน เช่น ดีแรมคอมโทรลเลอร์ (DRAM controller), อินเทอร์รัพคอนโทรลเลอร์ (Interrupt controller), ยูอาร์ต (UART) เข้าไว้ภายในตัวไมโครคอนโทรลเลอร์ซึ่งทำให้ง่ายต่อการออกแบบอุปกรณ์เพื่อทำเป็นสินค้าภายหลังการพัฒนาตัวอย่าง เช่น การนำบอร์ดคอม86

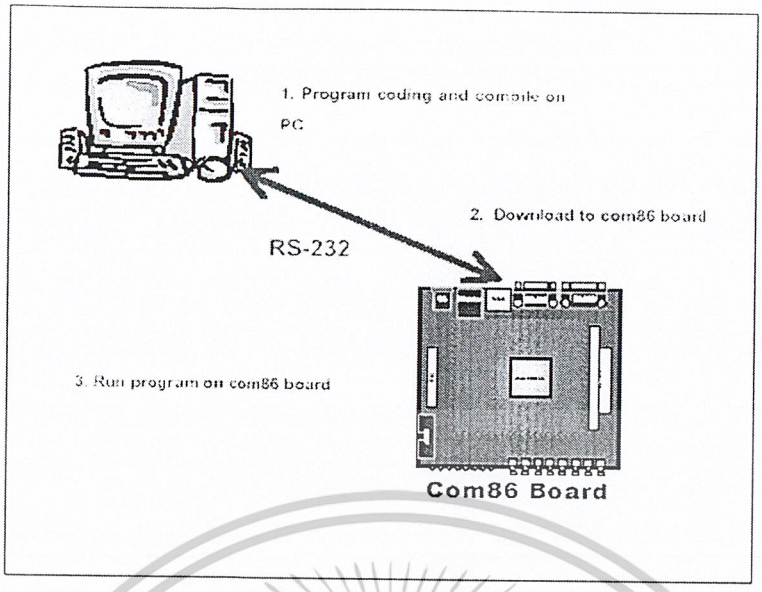
ไปพัฒนาเป็นอุปกรณ์อย่างหนึ่งที่ต้องการควบคุมการทำงานผ่านทางยูเอสบีพอร์ต (USB port) จะสามารถทำได้โดยง่ายเนื่องจากไม่ต้องการออกแบบวงจรควบคุมการทำงานในส่วนยูเอสบี นี้เพิ่มอีก เพียงแต่ออกแบบวงจรการทำงานส่วนอื่นที่ต้องการเพิ่มเติมทำให้ทุ่นเวลาในการพัฒนา เป็นต้น นอกจากนี้ภายในไมโครคอนโทรลเลอร์ยังเพิ่มส่วนโมดูล (Module) ทางด้านการสื่อสารต่างๆ เข้าไปภายในตัว ทำให้เป็นไมโครคอนโทรลเลอร์ที่มีความสามารถเด่นทางด้านการสื่อสารเหมาะกับการพัฒนาอุปกรณ์ที่ต้องใช้คุณสมบัติการสื่อสารต่างๆ ซึ่งจะสามารถทำได้ง่ายเมื่อเทียบกับการใช้ไมโครคอนโทรลเลอร์ขนาดเล็กแบบเดิมซึ่งมีข้อจำกัดต่างๆ อยู่มากมาย

2.7.1.2 มีเครื่องมือในการพัฒนาโปรแกรมมาก

เนื่องจากการพัฒนาโปรแกรมโดยใช้ชุดพัฒนาคอม86 นั้นเป็นเสมือนกับการพัฒนาโปรแกรมบนของคอมพิวเตอร์ส่วนบุคคลทั่วไป ดังนั้นจึงสามารถนำโปรแกรมเครื่องมือต่างๆที่ใช้พัฒนาโปรแกรมสำหรับคอมพิวเตอร์ส่วนบุคคลมาใช้งานได้มากมาย และโปรแกรมเครื่องมือเหล่านี้ก็เป็นที่คุ้นเคยของนักพัฒนาจึงทำให้สามารถพัฒนาอุปกรณ์ได้อย่างรวดเร็ว ลดต้นทุนในการพัฒนาต่างๆได้เป็นจำนวนมาก นอกจากนี้ภายในชุดพัฒนายังมีชุดซอฟต์แวร์เครื่องมือต่างๆที่จำเป็นสำหรับช่วยในการพัฒนาโปรแกรมด้วยอีกทางหนึ่ง

2.7.1.3 ง่ายในการเรียนรู้และพัฒนา

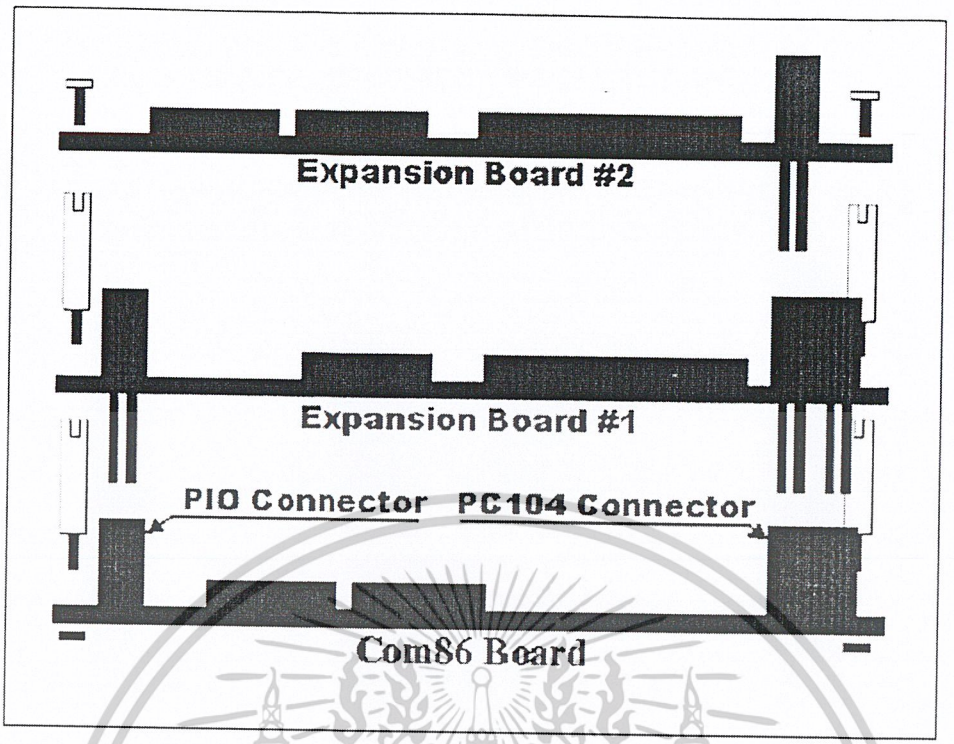
บอร์ดคอม86 ภายในชุดพัฒนาคอม86 นี้มีสถาปัตยกรรมของไมโครคอนโทรลเลอร์แบบ x86 ซึ่งเป็นสถาปัตยกรรมเดียวกับคอมพิวเตอร์ส่วนบุคคลทั่วไป โดยมีชุดคำสั่งที่เข้ากันได้กับชุดคำสั่งของ 80286 (Real mode) ทำให้ง่ายต่อการเรียนรู้ในการพัฒนาโปรแกรมต่างๆ และยังสามารถพัฒนาและทดสอบซอฟต์แวร์ต่างๆได้บนคอมพิวเตอร์ ก่อนที่จะทดสอบกับบอร์ดคอม86 ทำให้สามารถพัฒนาอุปกรณ์ทางด้านระบบฝังตัวต่างๆได้อย่างรวดเร็ว ซึ่งรูปแบบการพัฒนามีลักษณะเป็นไปดังภาพ



รูปภาพ 2-25 แสดงถึงรูปแบบการพัฒนาโปรแกรมโดยใช้บอร์ดคอม86

2.7.1.4 มีความสามารถในการเพิ่มขยายได้

ในส่วนของการเพิ่มขยายของฮาร์ดแวร์ (Hardware) สามารถทำได้โดยผ่านทางคอนเนคเตอร์ (Connector) ของบอร์ดในรูปแบบของมาตรฐาน (ซีพีหนึ่งศูนย์สี่) PC/104 ซึ่งจะมีการเชื่อมต่อแบบบัสอินเทอร์เฟส (BUS Interface) แบบไอเอสเอบัส (ISA BUS) ขนาด 8 บิต คล้ายกับไอเอสเอบัส ที่มีอยู่ในคอมพิวเตอร์ส่วนบุคคลทั่วไป ซึ่งผู้พัฒนาสามารถออกแบบฮาร์ดแวร์ให้เป็นลักษณะของบอร์ดขยายและนำมาเชื่อมต่อกับบอร์ดคอม86 และเพียงเขียนซอฟต์แวร์ไดร์เวอร์ (Software Driver) สำหรับควบคุมการทำงานของส่วนเพิ่มเข้ามาใหม่ซึ่งจะทำงานร่วมกับระบบปฏิบัติการในการรองรับการทำงานของแอพพลิเคชันต่างๆที่เพิ่มเข้ามา นอกจากนี้ตัวบอร์ดคอม86 ยังสามารถเชื่อมต่อกับอุปกรณ์ภายนอกในลักษณะของพีไอโอ (PIO) หรือ โปรแกรมเมเบิ้ลอินพุท/เอาท์พุท (Programmable Input/Output) เหมือนกันการใช้งานไมโครคอนโทลเลอร์รุ่นอื่นๆได้ด้วยเช่นกัน



รูปภาพ 2-26 แสดงการเชื่อมต่อของบอร์ดขยายกับบอร์ดคอม86

2.7.2 แอพพลิเคชั่นเป้าหมายของชุดพัฒนา (Target Application)

จากความสามารถของไมโครคอนโทรลเลอร์ของบอร์ดคอม86 ทำให้การประยุกต์ใช้งานกับแอพลิเคชั่นต่างๆนั้น สามารถทำได้มากมาย โดยตัวอย่างแอพพลิเคชั่นต่างๆที่เหมาะแก่การพัฒนาโดยบอร์ดคอม86 มีดังนี้

2.7.2.1 เกี่ยวกับอุตสาหกรรม

ตัวอย่างของอุปกรณ์ควบคุมภายในโรงงาน เช่น การควบคุมเครื่องจักรจากระยะไกล การตรวจวัดสภาวะต่างๆ ภายในโรงงานและส่งกลับไปยังศูนย์ควบคุมโดยการใช้บอร์ดคอม86 เป็นอุปกรณ์ที่ช่วยทำหน้าที่ในการสื่อสาร ควบคุมสภาวะแวดล้อม หรือเครื่องจักรต่างๆภายในโรงงานและรายงานกลับไปยังศูนย์กลางตามกำหนดเวลา เป็นต้น

2.7.2.2 อุปกรณ์ยูเอสบี (USB Peripheral)

อุปกรณ์ยูเอสบีต่างๆ สามารถพัฒนาได้โดยการใช้บอร์ดคอม86 ซึ่งภายในบอร์ดคอม86 นี้มียูเอสบีคอนโทรลเลอร์ภายในตัวไมโครคอนโทรลเลอร์ Am186CC สามารถเชื่อมต่อเข้ากับคอมพิวเตอร์ได้ที่ความเร็ว 12 เมกกะบิตเปอร์เซ็ค (Mbps) ซึ่งการพัฒนาเป็นอุปกรณ์ ยูเอสบี ต่างๆ โดยบอร์ดคอม86 นี้สามารถทำได้โดยไม่ต้องมีการต่อวงจรทางด้านยูเอสบีใดๆ เพิ่มเติมอีก

2.7.2.3 เว็บเซิร์ฟเวอร์ (Embedded web server)

เอ็มเบดเดตเว็บเซิร์ฟเวอร์ (Embedded web server) สามารถสร้างได้บนบอร์ดคอม86 โดยไม่จำเป็นต้องเชื่อมต่อวงจรอื่นๆ เพิ่มเติม ซึ่งบอร์ดคอม86 นี้สามารถสื่อสารกับภายนอกผ่านทางอีเธอร์เน็ตพอร์ตที่อยู่บนบอร์ดที่ความเร็ว 10 เมกกะบิตเปอร์เซ็ค ทำให้สามารถนำไปประยุกต์ใช้งานในกาออกแบบอุปกรณ์ควบคุมต่างๆ ผ่านเอ็มเบดเดตเว็บเซิร์ฟเวอร์ ที่ทำงานอยู่บนบอร์ดคอม86 ได้โดยง่าย

2.7.2.4 โรโบติก (Robotics)

บอร์ดคอม86 มีไมโครคอนโทรลเลอร์ที่มีประสิทธิภาพสูงในการทำงาน การพัฒนาซอฟต์แวร์สามารถพัฒนาได้ซับซ้อนมากยิ่งขึ้น การเชื่อมต่อขยายทางด้านฮาร์ดแวร์ต่างๆสามารถทำได้โดยง่าย เหมาะสำหรับการพัฒนาหุ่นยนต์ที่มีการทำงานที่ซับซ้อน

2.7.2.5 การควบคุมบ้านหรือสิ่งก่อสร้าง

อุปกรณ์สำหรับควบคุมระบบต่างๆภายในบ้าน หรืออาคารสามารถนำบอร์ดคอม86 ไปประยุกต์ใช้งานได้เช่นกัน โดยใช้เป็นส่วนในการควบคุมการทำงานของอุปกรณ์ต่างๆ หรือคอยตรวจสอบการทำงานของเซนเซอร์ (Senser) ต่างๆ ภายในบ้าน และแจ้งไปยังเจ้าของบ้านเมื่อเกิดเหตุร้ายขึ้น

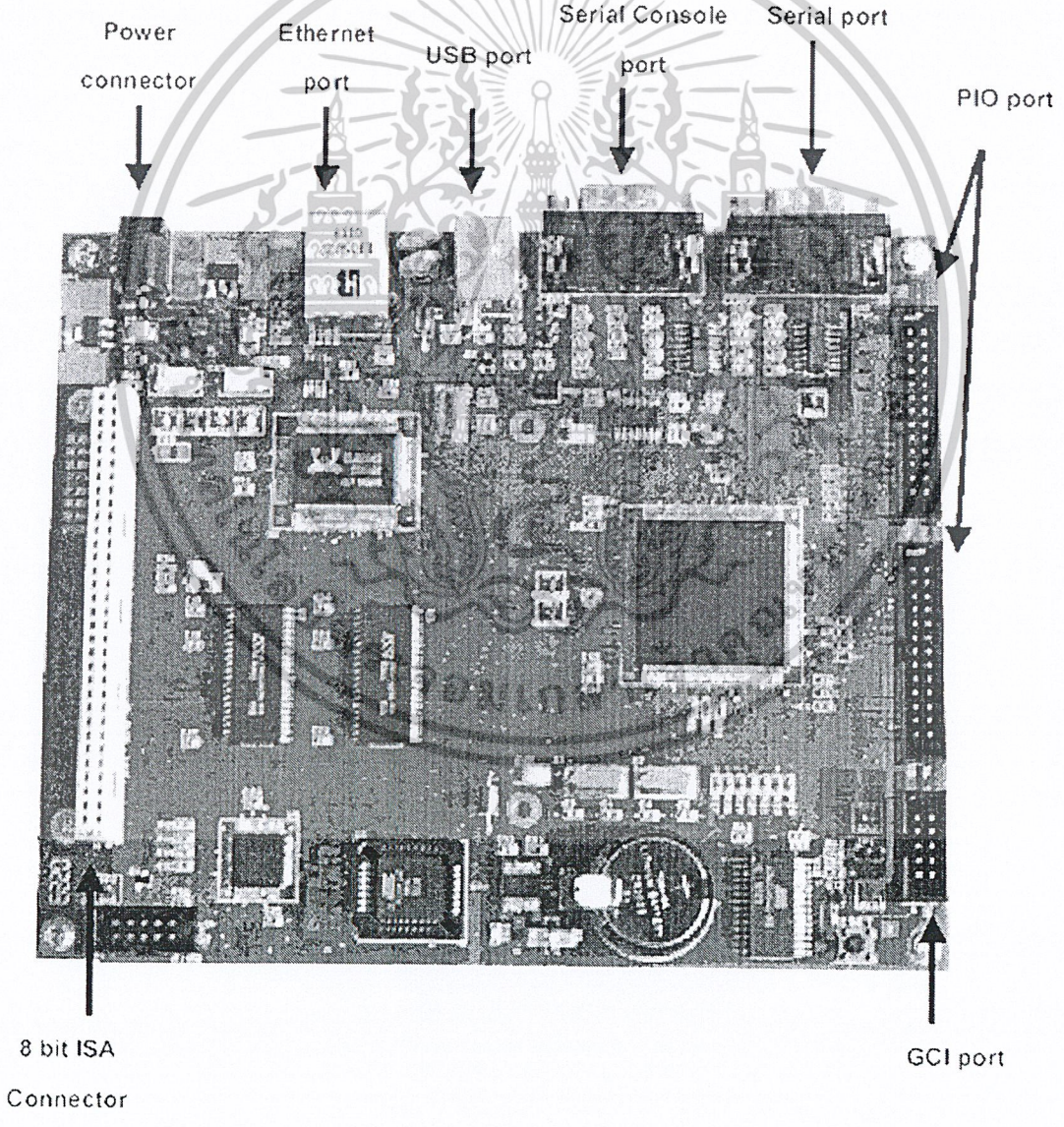
นอกจากนี้ตัวไมโครคอนโทรลเลอร์ยังรองรับการนำไปสร้างเป็นอุปกรณ์ต่างๆ ที่เกี่ยวข้องกับงานทางด้านการสื่อสารต่างๆเช่น ไอเอสดีเอ็นโมเด็มแอนด์เทอมินัลอแดบเตอร์ (ISDN Modem and Terminal Adaptor), โลเอนด์เร้าเตอร์ (Low-end Router), ไลน์การ์ดแอพพลิเคชั่น (Line card application), เอ็กซ์ดีเอสแอลแอพพลิเคชั่น (xDSL application), ดิจิตอลคอร์ดโฟน (Digital corded phone) เป็นต้น ซึ่งสามารถนำบอร์ดคอม86 ไปประยุกต์ใช้งานในการออกแบบพัฒนาได้เช่นกัน

2.7.3 คุณสมบัติของบอร์ดคอม86

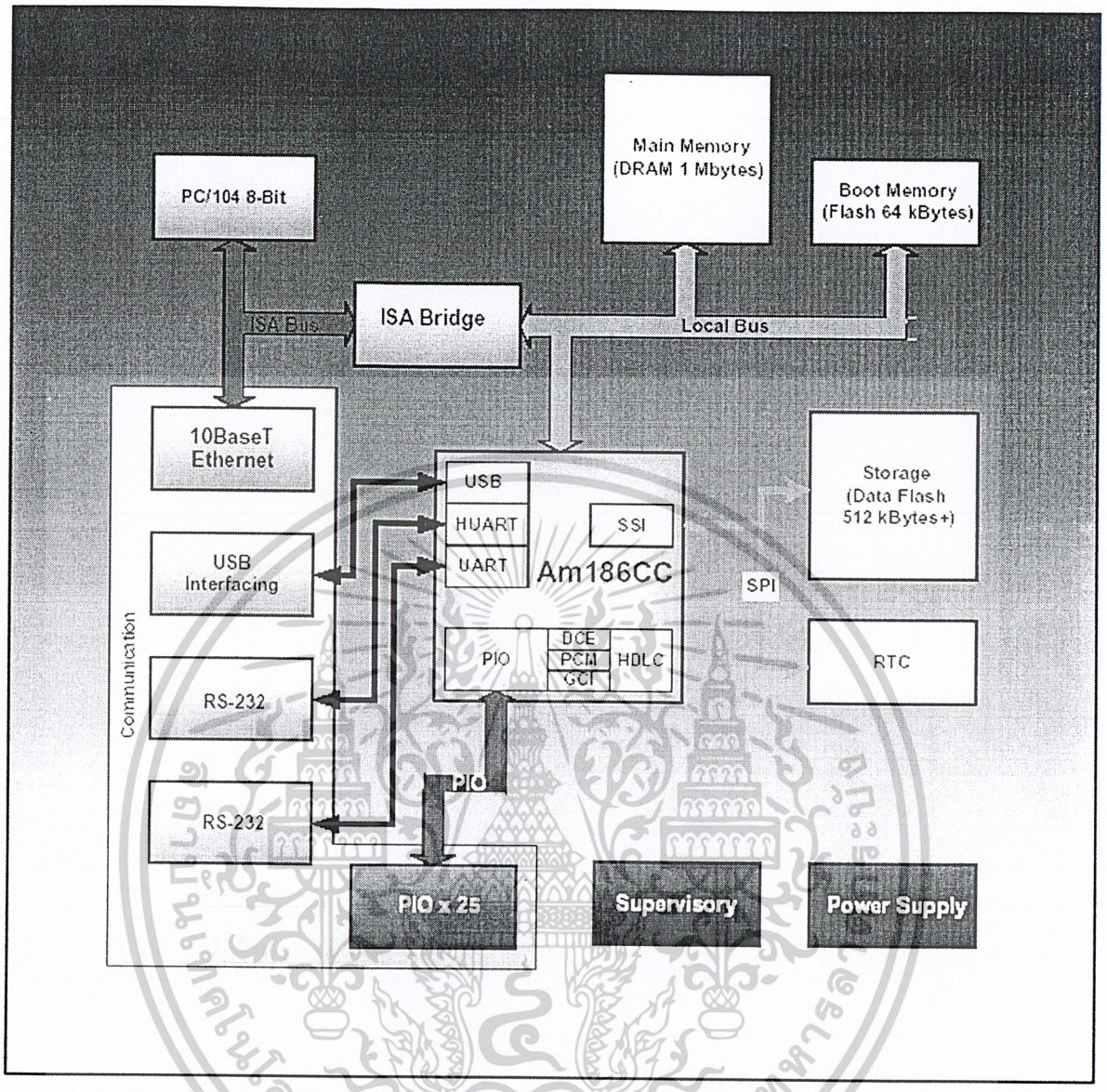
บอร์ดคอม86มีคุณสมบัติดังนี้

- ซีพียู เอซีหนึ่งแปดหกซีซี (CPU Am186CC)
- ดีแรม 1 เมกกะไบต์ (1 MBytes DRAM)
- แฟลตไบออส 64 กิโลไบต์ (64 kBytes Flash BIOS)
- 512 kBytes Serial DataFlash® storage (upgradeable)
- Full duplex IEEE 802.3 compliant on board Ethernet controller with RJ45 connector
- USB peripheral controller version 1.1 with on-board type B USB connector
- Dual RS232-serial port with on-board DB-9 connector
- Four High level Data Link Control (HDLC) channels (multiplex with PIO)
- Four independent Time slot Assigner (TSAs)

- GCI Controller with on-board GCI port for external interface
- Three programmable 16-bit timers
- Real-Time clock
- Watchdog timer and reset controller
- Interrupt Controller (10 external sources)
- 25 programmable I/O
- 8 bit ISA bus in PC/104 from factor
- 3.3 V and 5 V on-board power supply
- LED Indicators for power supply and Ethernet



รูปภาพ 2-27 ตัวอย่างของภาพชุดพัฒนาคอม86

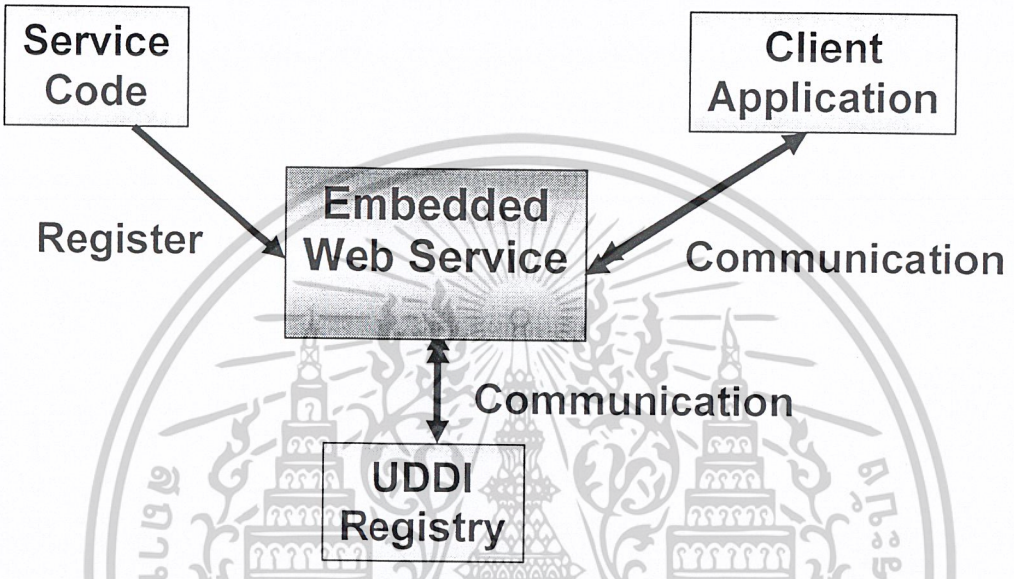


รูปภาพ 2-28 บล็อกไดอะแกรมแสดงส่วนประกอบต่างๆของคอม86

บทที่ 3

การออกแบบและพัฒนา

3.1 ขอบเขตของโครงงาน



รูปภาพ 3-1 คอนเท็กซ์ไดอะแกรม (Context Diagram) ของเว็บเซอร์วิสบนระบบฝังตัว

สำหรับการทำงานของเว็บเซอร์วิสบนระบบฝังตัวจะทำการติดต่อกับ 3 ส่วน คือ

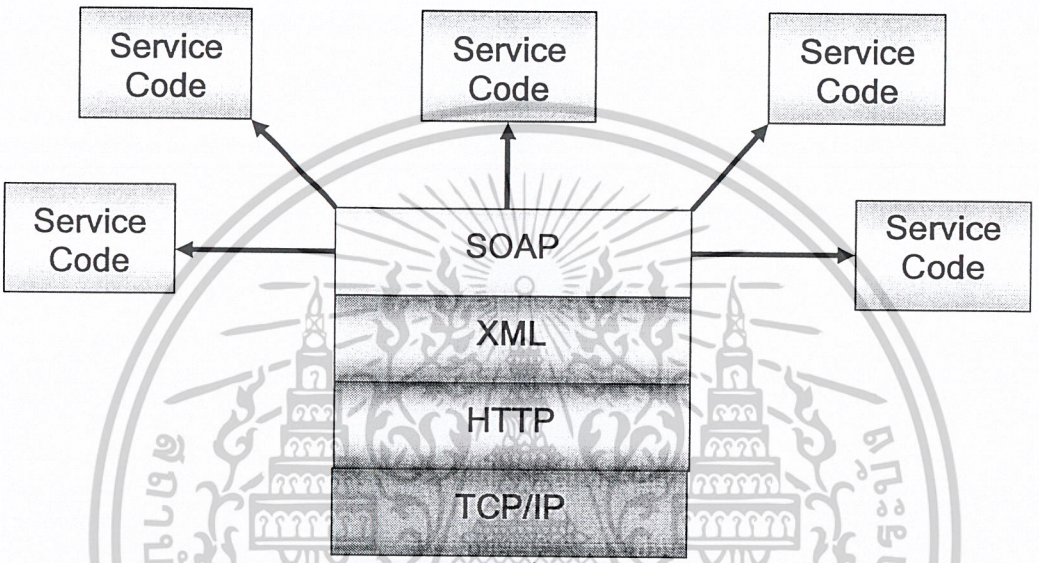
- ส่วนที่ผู้ให้บริการเว็บเซอร์วิสทำการลงทะเบียนเซอร์วิสโค้ด (Service Code) กับระบบเว็บเซอร์วิสบนระบบฝังตัวเพื่อให้ระบบสามารถให้บริการเซอร์วิสนั้นได้
- ส่วนที่ทำการติดต่อกับผู้ใช้ที่ทำการเรียกใช้บริการเว็บเซอร์วิส โดยที่ผู้ใช้บริการเว็บเซอร์วิสจะทำการเรียกใช้งานผ่านตัวแอพพลิเคชั่นทางฝั่งไคลเอนต์
- ในส่วนของเว็บเซอร์วิสนั้นจะต้องทำการประกาศบริการเว็บเซอร์วิส ไปยังยูดีดีไอรีจิสทรี่ โดยจะต้องทำการรีจิสเตอร์รายละเอียดต่างๆของตัวบริการ เพื่อให้ผู้ที่ใช้บริการนั้น สามารถเข้ามาทำการค้นหาเพื่อที่จะรู้วิธีการติดต่อกับตัวบริการนั้น แล้วจึงค่อยทำการติดต่อกับบริการ

3.2 โครงสร้างและสถาปัตยกรรม

3.2.1 สถาปัตยกรรมของเว็บเซอร์วิส

สำหรับสถาปัตยกรรมของระบบให้บริการเว็บเซอร์วิสนั้น ประกอบขึ้นจากเทคโนโลยีพื้นฐานในชั้นการติดต่อสื่อสารข้อมูลแต่ละชั้น ดังนี้

3.2.2 สถาปัตยกรรมของอาร์พีซีเซอร์วิส



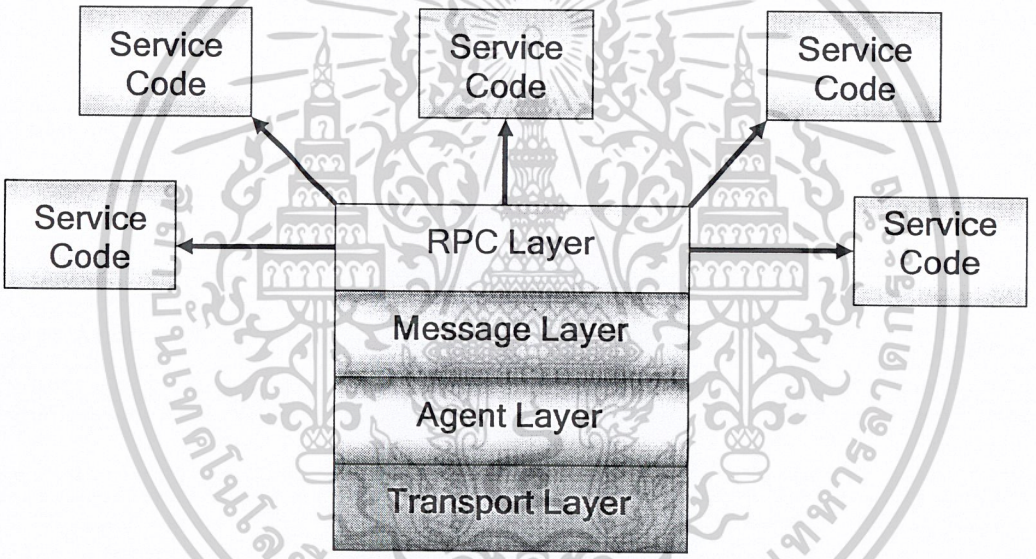
รูปภาพ 3-2 โครงสร้างของสถาปัตยกรรมของเว็บเซอร์วิส

1. **ชั้นทรานสปอร์ต (Transport Layer)** เป็นชั้นที่ทำหน้าที่เชื่อมต่อระดับเน็ตเวิร์ค ซึ่งทำการส่งข้อมูลระหว่างระบบคอมพิวเตอร์ที่ติดต่อกันผ่านเครือข่ายเน็ตเวิร์คซึ่งได้แก่ เครือข่ายอินเตอร์เน็ต โดยโปรโตคอลที่ใช้สำหรับการเชื่อมต่อในระดับนี้ ได้แก่ โปรโตคอลทีซีพี
2. **ชั้นแอพพลิเคชั่น (Application Layer)** เป็นชั้นที่ทำหน้าที่สื่อสารในระดับของการติดต่อสื่อข้อมูลเพื่อทำงานใดงานหนึ่งโดยเฉพาะ เช่น โปรโตคอลเอชทีทีพี (HTTP) ใช้ในการส่งข้อมูลในรูปแบบของเอกสารไฮเปอร์เท็กซ์ เพื่อแสดงผลเกี่ยวกับเว็บเพจ , โปรโตคอลเอฟทีพี (FTP) ใช้ในการส่งข้อมูลในรูปแบบของการรับส่งไฟล์ข้อมูล , โปรโตคอลเอสเอ็มทีพี (SMTP) ใช้ในการส่งข้อมูลในรูปแบบข้อการจัดส่งจดหมายอิเล็กทรอนิกส์ (Electronic mail) เป็นต้น
3. **ชั้นเอกสารในการแปลความหมาย (Message Layer)** เป็นชั้นที่ทำหน้าที่เป็นรูปแบบของเอกสารเพื่อกำหนดวิธีการแปลความหมายของข้อมูลที่ส่งถึงกัน เช่น เอกสารเอชทีเอ็มแอล เป็นเอกสารที่มีรูปแบบในการแปลความหมายเพื่อการแสดงผลผ่านหน้าจอเว็บบราวสเซอร์ (Web Browser) , เอกสารเอ๊กซ์เอ็มแอล เป็นเอกสารที่มีรูปแบบในการแปลความหมายที่สามารถกำหนดเองได้ ซึ่งเป็นแบบมาร์กอัพแลงเกวด (Markup Language)โดยเฉพาะ ใช้ในการสื่อสารข้อมูลระหว่างแอพพลิเคชั่น เป็นต้น

4. **ชั้นแอพพลิเคชั่นที่จัดการบริการ (Prosedure Call Layer)** เป็นชั้นที่ทำหน้าที่จัดการและเรียกใช้บริการต่างๆ ที่มีอยู่ในระบบในรูปแบบของการเรียกใช้ฟังก์ชัน ซึ่งทำให้ไม่ต้องขึ้นกับแพลตฟอร์ม ตัวอย่างเช่น ซิมเปิ้ลออบเจ็กต์แอคเซสโปรโตคอล หรือ โซบ (Simple Object Access Protocol : SOAP) ที่เป็นส่วนที่ทำหน้าที่ในการเรียกใช้บริการต่างๆ ในสถาปัตยกรรมเว็บเซอร์วิส , คอร์บา (CORBA) เป็น

3.2.3 สถาปัตยกรรมของอาร์พีซีเซอร์วิสบนระบบปฏิบัติการแบบเรียลไทม์

จากสถาปัตยกรรมเว็บเซอร์วิส ที่ประกอบไปด้วยชั้นต่างๆ ที่ทำหน้าที่เฉพาะตามแต่ละโปรโตคอล จึงสามารถสรุปให้มีความเป็นความหมายลักษณะกว้างๆ เพื่อที่จะชี้ให้เห็นถึงความสามารถของสถาปัตยกรรมการออกแบบที่สามารถครอบคลุมรูปแบบการติดต่อสื่อสารแบบอื่นๆได้



รูปภาพ 3-3 สถาปัตยกรรมเว็บเซอร์วิสในมุมมองของอาร์พีซี

จากสถาปัตยกรรมดังกล่าว จะทำให้การทำงานในแต่ละชั้นนั้น สามารถทำงานได้อย่างอิสระต่อกัน ทำให้ยืดหยุ่นต่อการนำไปพัฒนาเปลี่ยนแปลงชั้นสื่อสารแต่ละชั้น ตามแต่ละมาตรฐานหรือโปรโตคอล ทำให้ไม่จำเป็นต้องพัฒนาระบบใหม่ทั้งหมด ทำให้สามารถพัฒนาหรือเปลี่ยนแปลงไปตามแต่ละมาตรฐานได้อย่างรวดเร็ว โดยยกตัวอย่างโปรโตคอลต่างๆ ในแต่ละชั้นได้ดังนี้

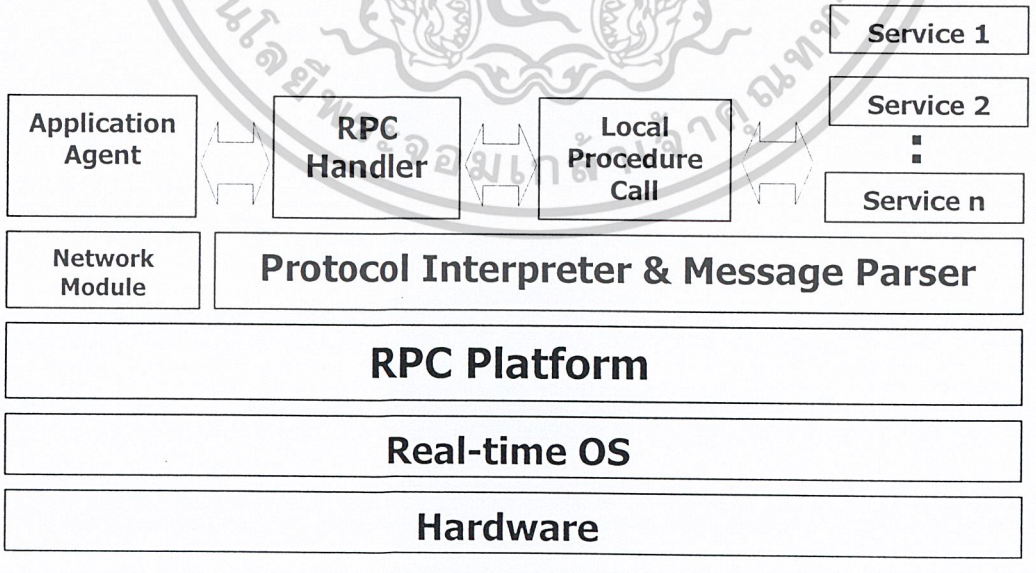
Application / Layer	SOAP Service	Web Server	SNMP Service
Transport Layer	TCP/IP	TCP/IP	UDP/IP
Agent Layer	HTTP	HTTP	SNMP
Message Layer	XML	HTML	Text format
RPC	SOAP	CGI & File Control	Network Managemant

ตาราง 3-1 โปรโตคอลในเลเยอร์ต่างๆ

การวิเคราะห์จากมุมมองดังกล่าวนี้ เพื่อเป็นแนวคิดในการออกแบบสถาปัตยกรรมเว็บเซอร์วิสที่สามารถดัดแปลงโครงสร้างให้สามารถรองรับการพัฒนาไปสู่การให้บริการอื่นๆได้ ภายในระบบเดียวกันเป็นแบบมัลติโปรโตคอล – มัลติเซอร์วิส (Multi-Protocol & Multi-Service) เพื่อเพิ่มความสามารถในการติดต่อสื่อสารบนอุปกรณ์ฝังตัว ให้ความหลากหลายคล้ายกับบนระบบให้บริการคอมพิวเตอร์ในระบบเอนเตอร์ไพร้ส์ (Enterprise Server)

3.3 กลไกการทำงานส่วนต่างๆ ของระบบอาร์พีซีเซอร์วิสบนระบบปฏิบัติการแบบเรียลไทม์

จากแนวคิดในการวิเคราะห์หน้าที่ของส่วนต่างๆในระบบเว็บเซอร์วิส ซึ่งสามารถแยกออกเป็นส่วนๆซึ่งทำงานอิสระต่อกัน รวมกับแนวคิดในการพัฒนาโปรแกรมบนระบบปฏิบัติการแบบเรียลไทม์ซึ่งสามารถรองรับการทำงานในแต่ละส่วนของระบบได้อย่างอิสระ และยังสามารถทำงานหลายๆงานได้ในเวลาเดียวกันด้วยเทคนิคของการ Scheduling โดยกลไกภายในของเคอร์เนล ทำให้สามารถนำแต่ละส่วนของสถาปัตยกรรมอาร์พีซีเซอร์วิส มาสร้างขึ้นเป็นทาสก์ย่อยๆ ที่ทำงานอยู่บนเคอร์เนลเดียวกัน ดังรูป



รูปภาพ 3-4 สถาปัตยกรรมของอาร์พีซีเซอรร์วิสบนระบบปฏิบัติการแบบเรียลไทม์

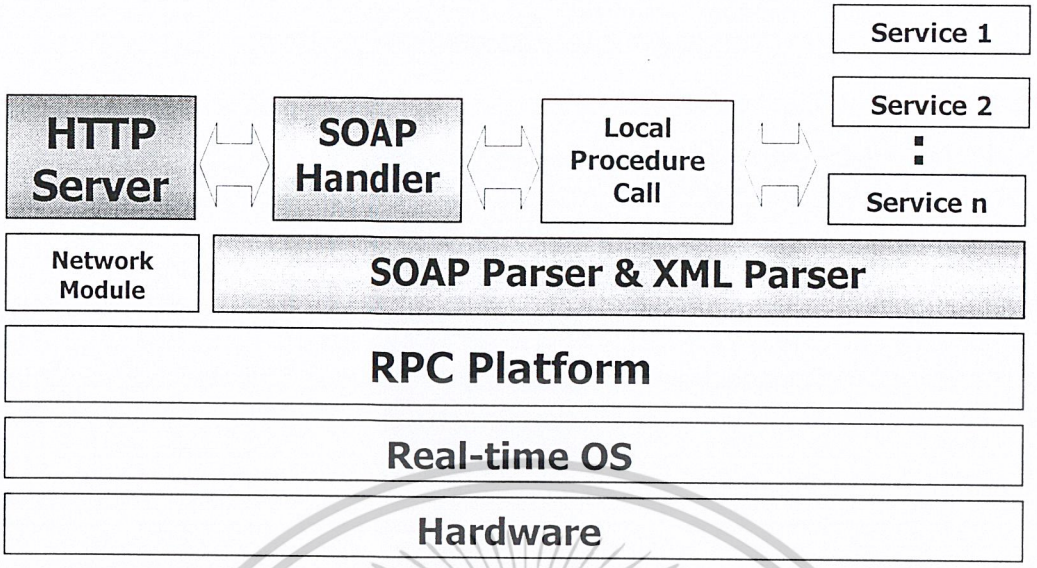
ส่วนประกอบต่างๆ สามารถอธิบายได้ดังนี้

- **ฮาร์ดแวร์** (Hardware) เป็นระบบอุปกรณ์ฝังตัว ที่นำมาใช้งาน โดยเป็นแพลตฟอร์มที่พัฒนาความสามารถในด้านต่างๆ ให้ทำงานตามที่ต้องการโดยโปรแกรมและระบบปฏิบัติการ
- **ไมโครเคอร์เนล (Micro Kernel)** เป็นส่วนของระบบปฏิบัติการที่ควบคุมอุปกรณ์ที่เป็นฮาร์ดแวร์โดยตรง และจัดการงานต่างๆ รวมถึงเตรียมระบบที่ช่วยให้โปรแกรมสามารถทำงานร่วมกันได้อย่างมีประสิทธิภาพ
- **เน็ตเวิร์คโมดูล (Network Module)** เป็นส่วนของโปรแกรมในระบบที่ทำหน้าที่ในการจัดการและควบคุมการติดต่อกับเครื่องคอมพิวเตอร์หรืออุปกรณ์ฝังตัวอื่นๆ ผ่านระบบเน็ตเวิร์ค โดยโปรโตคอลได้แก่ ทีซีพี/ไอพี , ยูดีพี/ไอพี เป็นต้น
- **ส่วนจัดการโปรโตคอลและอ่านเอกสาร (Protocol Interpreter & Message Parser)** เป็นยูทีลิตี้ที่ช่วยให้โปรแกรมบนระบบ สามารถจัดการกับเอกสาร หรือข้อมูลในรูปแบบของโปรโตคอลต่างๆ ที่เป็นมาตรฐานในการติดต่อขอใช้บริการระหว่างกัน ตัวอย่างส่วนจัดการโปรโตคอล เช่น โซบพาร์เซอร์ (SOAP Parser) , เอชทีทีพีพาร์เซอร์ (HTTP Parser) เป็นต้น และตัวอย่างยูทีลิตี้ที่ช่วยในการอ่านและจัดการเอกสาร เช่น เอ็กซ์เอ็มแอลพาร์เซอร์ (XML Parser) เป็นต้น
- **เอเจนต์ (Agent)** เป็นส่วนที่ทำหน้าติดต่อสื่อสารข้อมูลจนได้ข้อมูลที่เป็นเอกสารหรือข้อมูลในรูปของโปรโตคอลในการเรียกใช้บริการ ซึ่งทำงานบนส่วนของเน็ตเวิร์คโมดูลอีกทีหนึ่ง ตัวอย่างเช่น เอชทีทีพีเซิร์ฟเวอร์ , เอฟเอ็มทีพีเซิร์ฟเวอร์ เป็นต้น
- **อาร์พีซี (RPC)** เป็นส่วนที่ประสานงานเพื่อทำการติดต่อขอใช้บริการในระบบ หลังจากดึงข้อมูลในการเรียกใช้บริการโดยส่วนจัดการโปรโตคอล ตัวอย่างเช่น โซบเซอร์วิสทาสก์ (SOAP Service Task) เป็นต้น
- **บริการ (Service)** เป็นฟังก์ชั่นโค้ดที่เป็นบริการที่มีอยู่บนระบบ

จากสถาปัตยกรรมโครงสร้างดังกล่าว สามารถย้อนกลับไปรองรับส่วนต่างๆของระบบเว็บเซอร์วิสได้อย่างลงตัว ตามตารางดังนี้

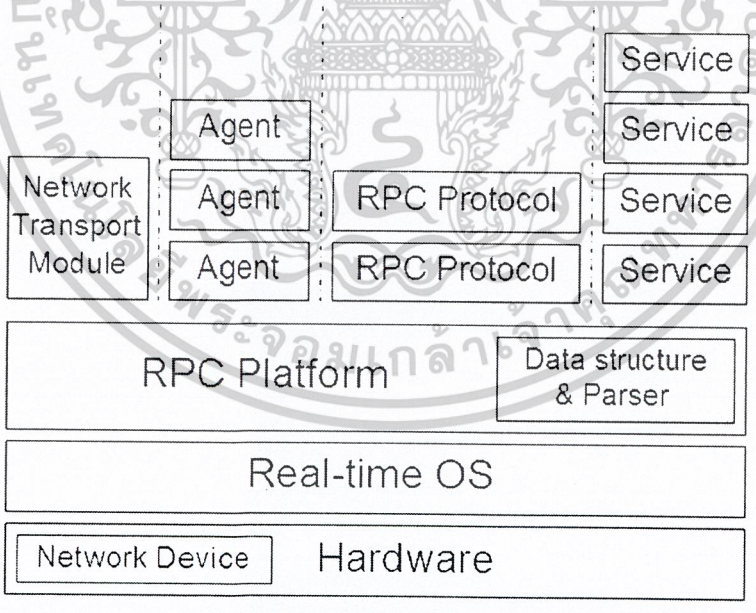
ส่วนของอาร์พีซีเซอร์วิส	ส่วนของเว็บเซอร์วิส	หน้าที่
Agent	HTTP Server	รอรับการติดต่อในรูปแบบของเอชทีทีพี
RPC	SOAP	แปลงเอกสารโซบ แล้วสร้างการเรียกใช้บริการ
Message Parser	XML Parser	อ่านและเขียนเอกสารในรูปแบบของเอ็กซ์เอ็มแอล

ตาราง 3-2 ส่วนประกอบของเว็บเซอร์วิสที่สร้างขึ้นได้ด้วยแนวคิดของอาร์พีซีเซอร์วิส



รูปภาพ 3-5 สถาปัตยกรรมเว็บเซอร์วิสบนแนวคิดของอาร์พีซีเซอร์วิส

หลังจากการพัฒนาระบบอาร์พีซีเพื่อให้บริการงานบนระบบอุปกรณ์ฝังตัวแล้ว หากต้องการเพิ่มความสามารถให้ระบบสามารถขยายการรองรับการเชื่อมต่อด้วยโปรโตคอลอื่นๆ หรือสร้างบริการใดๆ เพิ่มเติม ก็สามารถทำได้โดยทำการเพิ่มเติมโค้ดของส่วนต่างๆนั้นเพิ่มเข้าไปในระบบ

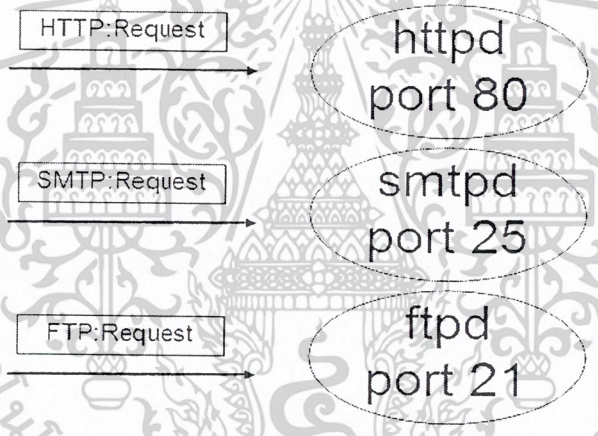


รูปภาพ 3-6 มุมมองในการพัฒนาส่วนต่างเพิ่มเติมขึ้นในระบบ

3.3.1 โพรเซสของผู้ให้บริการ (Server Daemon)

ส่วนของโพรเซสของผู้ให้บริการนี้ เป็นทาสก์ที่ทำงานอยู่ตลอดเวลา หรือที่เรียกว่า แดมอนโพรเซส (Daemon Process) ซึ่งเป็นทำหน้าที่เป็นโพรเซสที่คอยรับและทำหน้าที่เชื่อมการติดต่อในระดับเซสชั่นจากผู้ขอใช้บริการ โดยเมื่อผู้ขอใช้บริการส่งคำสั่งขอเชื่อมต่อ โพรเซสขอผู้ให้บริการนี้ก็จะสร้างช่องทางการติดต่อหรือซอคเก็ต (Socket) ขึ้นมาใหม่ แล้วบันทึกค่าต่างๆที่จำเป็นต่อการทำการติดต่อ แล้วเรียกให้ส่วนที่จัดการทาสก์ ทำการสร้างทาสก์ขึ้นมาใหม่ แล้วส่งต่อซอคเก็ตของการเชื่อมต่อครั้งนั้นๆ ไปเก็บไว้เป็นส่วนหนึ่งของทาสก์ จากนั้น โพรเซสนี้ ก็จะกลับไปรอรับฟังการเชื่อมต่อในเซสชั่นอื่น จากผู้ขอใช้บริการอื่นต่อไป

ในสถาปัตยกรรมของอาร์พีซีเซอร์วิสที่ได้ออกแบบมานั้น สามารถรองรับการทำงานของเซิร์ฟเวอร์แดมอนได้หลายๆ โพรเซสพร้อมกัน ซึ่งแต่ละโพรเซสนั้น จะรอรับการเชื่อมต่อจากพอร์ตการสื่อสารที่ต่างๆกันไป ตามหมายเลขพอร์ตที่รองรับโปรโตคอลต่างๆกันไป ทำให้สามารถรองรับการเชื่อมต่อได้พร้อมๆกัน ที่ละหลายๆโปรโตคอล



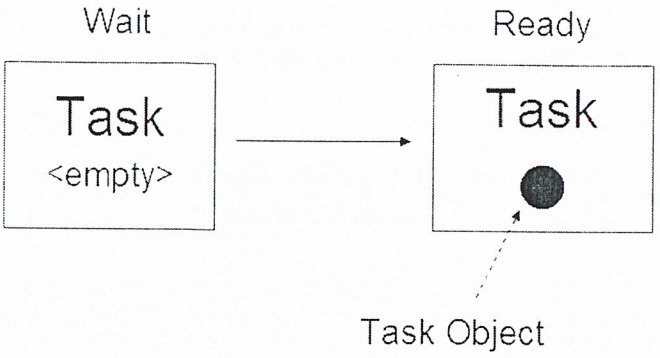
รูปภาพ 3-7 เซิร์ฟเวอร์แดมอนของโปรโตคอลต่างๆ

3.3.2 ส่วนจัดการโปรโตคอล (Protocol Handler)

ส่วนจัดการโปรโตคอลนี้ เป็นส่วนที่แปลงเอกสารที่ส่งมาจากผู้ขอใช้บริการ ให้อยู่ในรูปแบบของข้อมูลที่ใช้ในการขอใช้บริการต่างๆ โดยมีรูปแบบตามแต่ละโปรโตคอลซึ่งเป็นมาตรฐานในการติดต่อ เช่น โซบแฮนเดิ้ลเลอร์ ทำหน้าที่แปลงเอกสารโซบ แล้วนำข้อมูลไปขอเรียกใช้บริการต่อไป

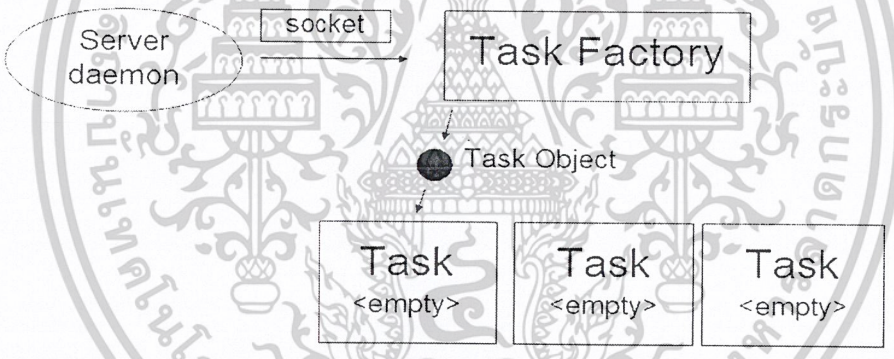
3.3.3 ทาสก์และส่วนปฏิบัติการทาสก์ (Task & Task Manager)

- **ทาสก์ (Task)** ที่เป็นรับผิดชอบในการดำเนินงานของรีเควส 1 รีเควส แล้วทำการดำเนินขั้นตอนตามลำดับที่กำหนดไว้ในแต่ละทาสก์ออบเจ็กต์ ตั้งแต่เริ่มต้น จนเสร็จกระบวนการการทำงานตามที่ผู้ขอใช้บริการทำการร้องขอ จากนั้นทาสก์นั้นก็จะว่าง แล้วกลับไปรอรับการสร้างเซอร์วิสต่อไป



รูปภาพ 3-8 สถานะของทาสก์ที่ว่าง กลายไปเป็นพร้อมทำงานเมื่อมีทาสก์ออบเจ็กต์

- **ส่วนสร้างทาสก์ หรือ ทาสก์แฟคตอรี่ (Task Factory)** ทำหน้าที่ในการสร้างทาสก์ออบเจกต์เมื่อมีการร้องขอจากผู้ขอใช้บริการเข้ามา จากนั้นก็ทำการเลือกหาว่ามีทาสก์ใดที่อยู่ในสถานะที่ว่างและพร้อมรับการทำงานอยู่ จากนั้นก็จะทำการส่งทาสก์ออบเจ็กต์ที่สร้างไว้แล้ว ไปให้กับทาสก์ที่ถูกเลือก เพื่อให้ทาสก์ๆนั้น รับผิดชอบในการดำเนินการตามขั้นตอนที่กำหนดอยู่ในทาสก์ออบเจ็กต์ที่ส่งให้ต่อไป

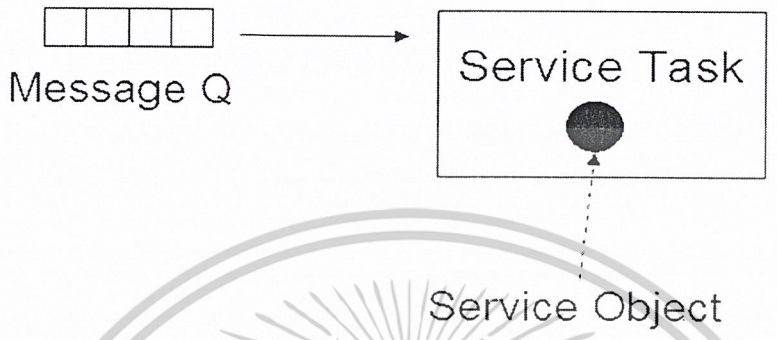


รูปภาพ 3-9 การทำงานของทาสก์แฟคตอรี่

3.3.4 บริการและส่วนจัดการบริการ (Service & Service Manger)

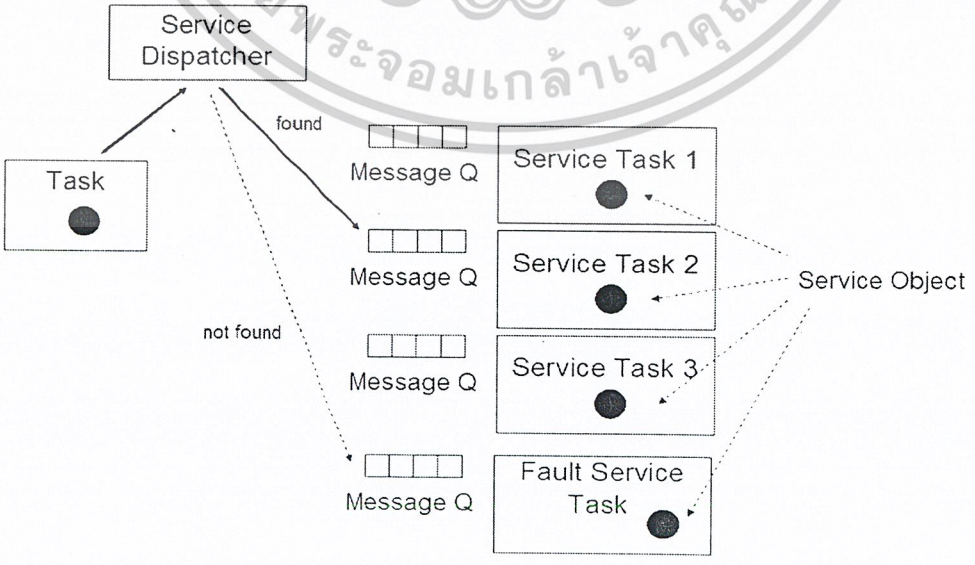
- **เซอร์วิสทาสก์ (Service Task)** เป็นทาสก์ที่ทำหน้าที่ดำเนินการกระบวนการของเซอร์วิสต่างๆที่มีในระบบ ให้รอรับการร้องขอจากผู้ขอใช้บริการ ซึ่งเซอร์วิสทาสก์นี้ จะถูกเรียกใช้โดยทาสก์ออบเจ็กต์ ที่ทำงานอยู่ในทาสก์ที่รองรับรีเควสนั้นๆอยู่ แต่เซอร์วิสทาสก์จะถูกเรียกโดยผ่านเซอร์วิสดิสแพตเชอร์ อีกที แต่ในการส่งผลลัพธ์นั้น จะทำการส่งไปให้ยังทาสก์ที่ร้องขอโดยตรง ภายในเซอร์วิสทาสก์นั้นมีเซอร์วิสออบเจ็กต์อยู่ภายใน โดยเซอร์วิสออบเจ็กต์นี้เป็นตัวที่เก็บฟังก์ชั่นที่ใช้สำหรับการประมวลผล โดยเซอร์วิสหนึ่งๆนั้น จะถูกสร้างโดยเซอร์วิสเมเนเจอร์ในตอนเริ่มแรกของการทำงานของระบบ จากนั้นเซอร์วิสออบเจ็กต์ก็จะถูกส่งเข้ามายังเซอร์วิสทาสก์ซึ่งมีการเตรียมความพร้อมด้วยการสร้าง แมสเสจคิว (Message Queue) ที่จำเป็นในการรอรับการร้องขอจากเซอร์วิสดิสแพตเชอร์ แล้วทำการเริ่มต้นกระบวนการ จนไปตลอดเวลาขณะระบบทำงาน โดยขณะที่เซอร์วิสทาสก์ถูกเรียกใช้ผ่านการส่งข้อมูลมาที่แมสเซจคิว ของเซอร์วิส

ทาสก์นั้น แล้วเซอร์วิสทาสก์กำลังประมวลผลตามฟังก์ชั่นที่มีอยู่ หากมีทาสก์ใด ที่ต้องการขอบริการมาที่เซอร์วิสทาสก์เดียวกันนี้ ข้อมูลในการร้องขอบริการจะถูกนำเข้าในแมสเซจคิวเอาไว้จนกว่าเซอร์วิสทาสก์นั้นจะทำงานเสร็จสิ้นงานที่ถูกร้องขอก่อนหน้า แล้วจึงไปนำข้อมูลการร้องขอบริการจากแมสเซจคิวออกมาทำการประมวลเพื่อให้บริการต่อไป



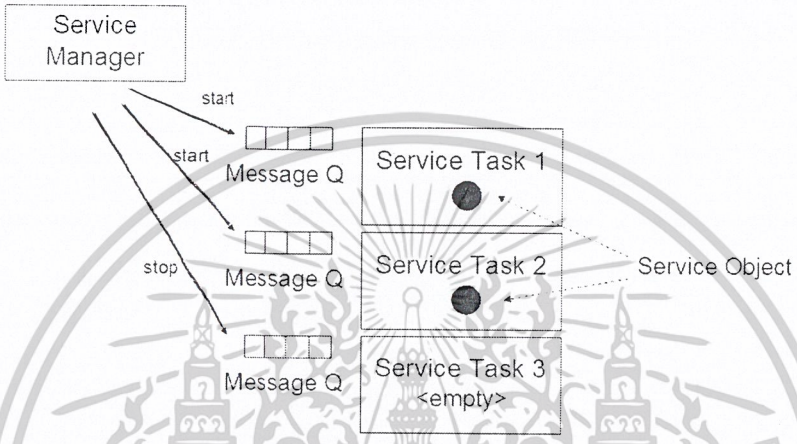
รูปภาพ 3-10 เซอร์วิสทาสก์ ที่มีเซอร์วิสออบเจ็กตทำงานอยู่ภายใน โดยรับข้อมูลจากแมสเซจคิว

- เซอร์วิสดิสแพตเชอร์ (**Service Dispatcher**) เป็นส่วนที่ทำหน้าที่แจกจ่ายการร้องขอใช้บริการไปยังเซอร์วิสทาสก์แต่ละอัน โดยจะตรวจหาว่าเซอร์วิสที่ทาสก์เรียกเข้ามานั้น มีอยู่ในระบบหรือไม่ ถ้ามี ก็จะไปค้นหาตำแหน่งของแมสเซจคิวของเซอร์วิสนั้นๆ เพื่อส่งข้อมูลการร้องของบริการจากทาสก์ที่ร้องขอนั้นต่อไปยังเซอร์วิสทาสก์อีกที แต่หากเซอร์วิสที่ทาสก์ร้องขอนั้น ไม่มีอยู่ในรายการเซอร์วิสที่มีอยู่ในระบบแล้ว เซอร์วิสดิสแพตเชอร์ก็จะทำการส่งข้อมูลร้องขอเหล่านั้น ไปยังบริการที่รองรับตอบบริการที่ผิดพลาด หรือ ฟอล์ตเซอร์วิส (Fault Service) ซึ่งเป็นเซอร์วิสทาสก์หนึ่งในระบบที่คอยตอบความผิดพลาดกลับไปยังทาสก์ที่ร้องขอ เพื่อส่งข้อมูลความผิดพลาดในการเรียกใช้บริการกลับไปยังผู้ขอใช้บริการต่อไป



รูปภาพ 3-11 การเรียกเซอร์วิสทาสก์โดยเซอร์วิสดิสแพตเชอร์

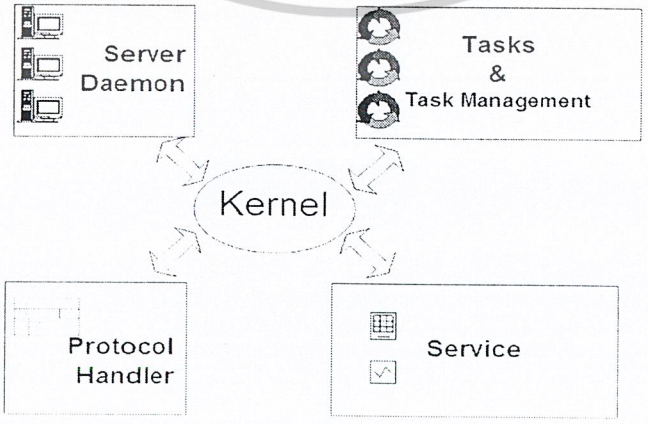
- **ผู้จัดการบริการ หรือ เซอร์วิสเมเนเจอร์ (Service Manager)** เป็นส่วนที่ทำหน้าที่ในการสร้างเซอร์วิสออบเจ็กต์ที่มีอยู่ในระบบ แล้วส่งไปให้เซอร์วิสทาสก์ทำงานตามขั้นตอนที่มีอยู่ในออบเจ็กต์ หากเซอร์วิสใดมีสถานะให้หยุดบริการ เซอร์วิสเมเนเจอร์ก็จะไม่สร้างเซอร์วิสออบเจ็กต์นั้นให้เซอร์วิสทาสก์ทำงาน แต่หากผู้ดูแลระบบทำการเปิดให้บริการนั้น เซอร์วิสทาสก์ก็จะสร้างเซอร์วิสออบเจ็กต์ขึ้นมา แล้วทำตามกระบวนการโดยส่งไปยังเซอร์วิสทาสก์ที่ว่างอยู่ เพื่อให้บริการนั้นเปิดใช้งานขึ้นมากอีกครั้ง



รูปภาพ 3-12 การควบคุมเซอร์วิสทาสก์โดยเซอร์วิสเมเนเจอร์

3.3.5 เคอร์เนลและระบบปฏิบัติการแบบเรียลไทม์

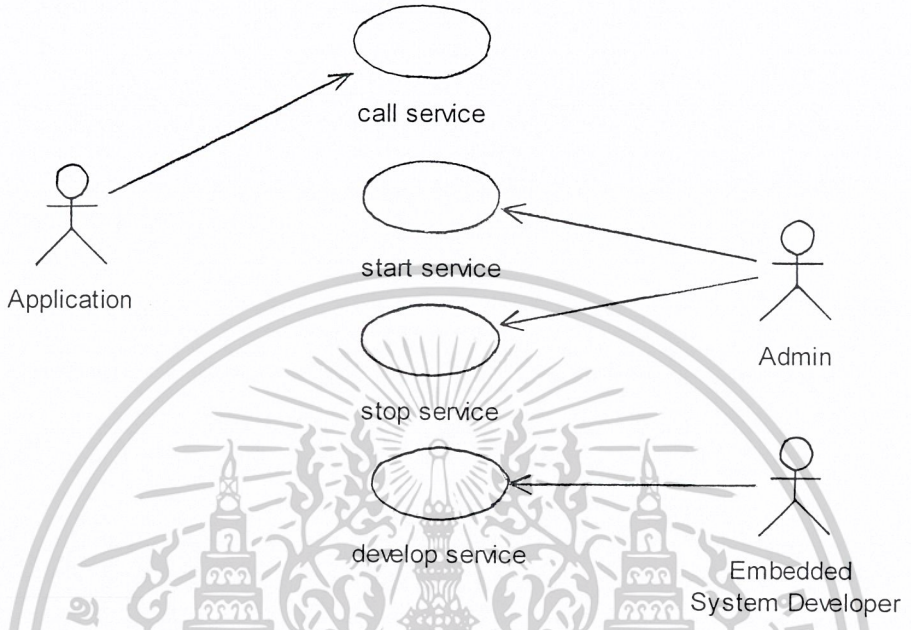
เป็นส่วนที่ทำหน้าที่ควบคุมและจัดการการทำงานของทาสก์ในแต่ละส่วนของระบบที่ได้กล่าวมาแล้วให้ทำงานร่วมกันอย่างถูกต้อง รวมถึงจัดการทรัพยากรต่างของระบบให้กับทาสก์ต่างๆที่ต้องการใช้อย่างถูกต้อง และไม่ก่อกวนการทำงานของกันและกัน รวมถึงจัดสรรช่วงเวลาในการทำงานตามระดับความสำคัญ (Priority) เพื่อช่วยให้งานที่ถูกกำหนดความสำคัญระดับสูง ได้ถูกประมวลผลเสร็จก่อนตามหลักการของเรียลไทม์ (Real-time) โดยซอฟต์แวร์ไลบรารีของระบบปฏิบัติการแบบเรียลไทม์ที่นำมาพัฒนาระบบ ได้แก่ ไมโครซี/โอเอสทู (µC/OSii) ที่เป็นซอฟต์แวร์ไลบรารีที่เป็นชนิดโอเพ่นซอร์ส



รูปภาพ 3-13 ภาพรวมของโครงสร้างภายในของอาร์พีซีเซอร์วิสบนระบบปฏิบัติการแบบเรียลไทม์

3.4 รายละเอียดของซอฟต์แวร์อาร์พีซีเซอร์วิส (Software Configuration)

3.4.1 ยูสเคสไดอะแกรม (Use-Case Diagram)



รูปภาพ 3-14 ยูสเคสไดอะแกรม (Use-Case Diagram) ของระบบ

การร้องขอใช้บริการ

Use Case	Call Service
ผู้ใช้	ผู้ให้บริการเว็บเซอร์วิส
เงื่อนไข	ผู้ใช้บริการต้องส่งการขอใช้บริการผ่านทางช่องทางการติดต่อที่มีขึ้นในระบบขณะนั้น แล้วส่งข้อมูลในการขอใช้บริการให้ถูกต้องตามวิธีการในการขอใช้บริการที่เป็นมาตรฐานที่ระบบรองรับ เช่น โซบโปรโตคอล หรือ เอฟทีพีโปรโตคอล เป็นต้น
ข้อมูลที่เกี่ยวข้อง	เมื่อได้รับรีเควส มา จากนั้นแอพพลิเคชั่นพาร์เซอร์จะทำการแปลงให้อยู่ในรูปของแมสเซจแล้วค่อยทำการดึงรายละเอียดในการเรียกใช้บริการ เพื่อนำมาใช้ในการติดต่อกับตัวบริการนั้นๆ

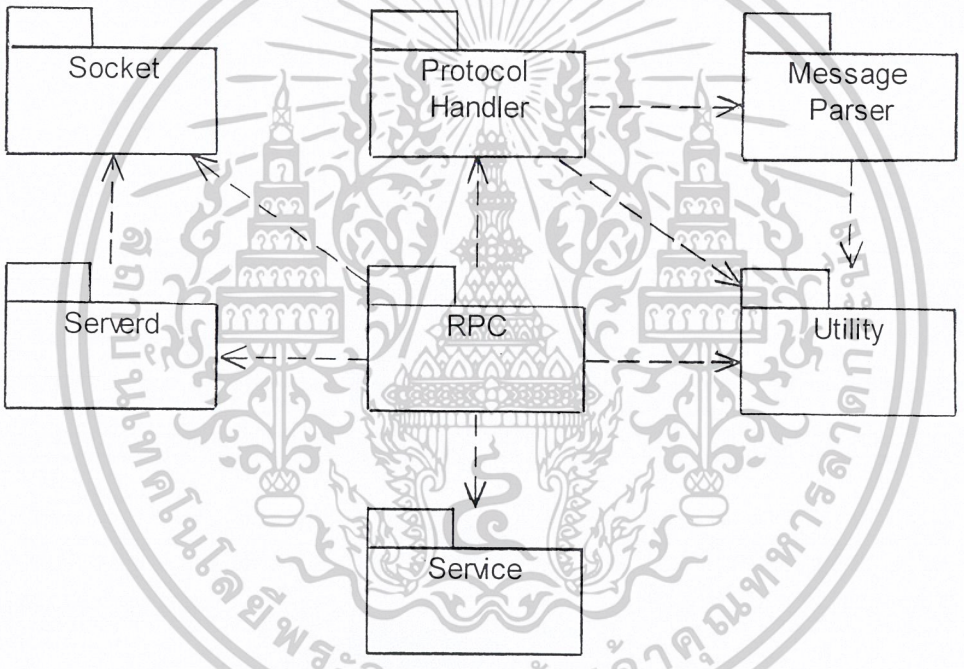
ตาราง 3-3 การร้อขอใช้บริการเว็บเซอร์วิส

การควบคุมบริการบนระบบ

Use Case	Start , Stop Service
ผู้ใช้	ผู้ดูแลระบบให้บริการ
เงื่อนไข	ผู้ดูแลอาจพัฒนาโปรแกรมส่วนของการติดต่อเพื่อควบคุมบริการ ซึ่งมีบริการในการควบคุมบริการอื่นบนระบบอีกที

ตาราง 3-4 การควบคุมบริการบนระบบ

3.4.2 คอมโพเนนท์ไดอะแกรม (Component Diagram)



รูปภาพ 3-15 คอมโพเนนท์ไดอะแกรม (Component Diagram) ของระบบ

ส่วนประกอบต่างมีรายละเอียดดังนี้

- **อาร์พีซี (RPC)** เป็นส่วนกลางของระบบที่จัดการและควบคุมการทำงานทั้งหมดของระบบ โดยจะเชื่อมโยงส่วนต่างๆ เข้ามาทำงานร่วมกัน และยังเป็นส่วนที่จัดการทาสก์ของรีเควสต่างๆ จึงถือเป็นหัวใจหลักของระบบ
- **เซิร์ฟเวอร์แดมอน (Server Daemon)** เป็นส่วนที่ทำหน้าที่เป็นเซิร์ฟเวอร์ที่คอยรับการร้องขอในการเชื่อมต่อเพื่อขอใช้บริการในแต่ละครั้ง โดยจะรวมเซิร์ฟเวอร์แดมอน ของแต่ละโปรโตคอลเข้าไว้ในส่วนนี้

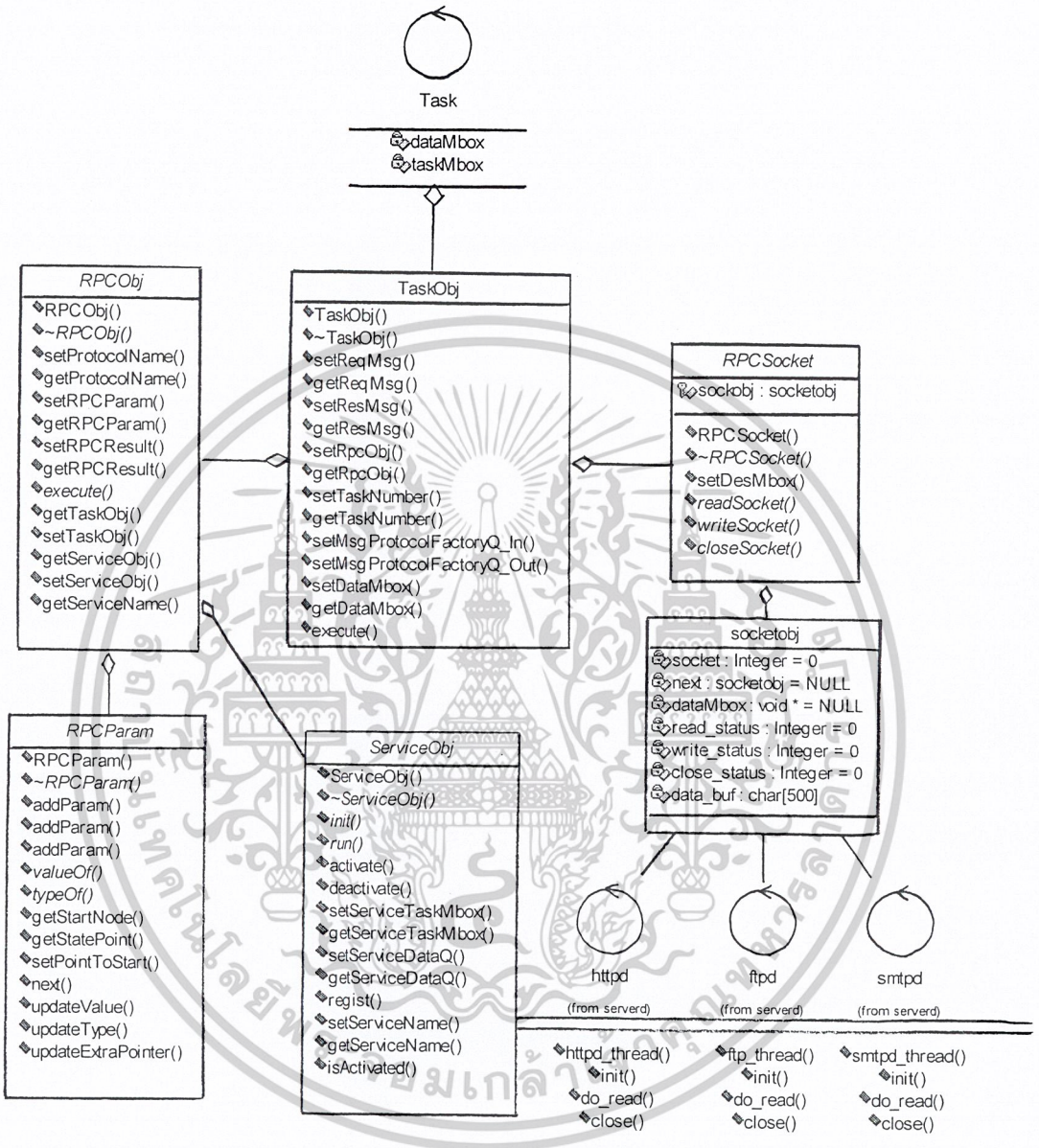
- **ซ็อคเก็ต (Socket)** เป็นกลุ่มของส่วนที่ทำหน้าที่ในการเชื่อมต่อเซสชั่นในแต่ละโปรโตคอล ที่จะเก็บสถานะในการเซสชั่นการเชื่อมต่อนั้นๆ จนกว่าการร้องขอบริการจะเสร็จสิ้น แล้วทำการจบเซสชั่นการติดต่อไป
- **โปรโตคอลแฮนเดิ้ลเลอร์ (Protocol Handler)** เป็นส่วนที่ทำหน้าที่อ่านและแปลงเอกสารที่อยู่ในรูปแบบของมาตรฐานโปรโตคอลต่างๆ ที่สร้างขึ้นในระบบ เพื่อให้ได้ข้อมูลในการร้องขอบริการในแต่ละครั้ง อีกทั้งยังทำหน้าที่ในการประกอบและสร้างเอกสารกลับไปสู่รูปแบบมาตรฐานตามโปรคอลนั้น เพื่อส่งกลับคืนไปยังผู้ร้องขอต่อไป
- **แมสเซจพาร์เซอร์ (Message Parser)** เป็นส่วนที่ทำหน้าที่ในการอ่านและบันทึกเอกสารในรูปแบบต่างๆ เช่น XML โดยส่วนนี้ จะอยู่ในลักษณะของซอฟต์แวร์ไลบรารี่ที่ถูกเรียกใช้โดยส่วนของโปรโตคอลแฮนเดิ้ลเลอร์ ที่จำเป็นต้องอ่านเอกสารในรูปแบบต่าง
- **บริการ (Service)** เป็นส่วนที่รวบรวมซอฟต์แวร์ฟังก์ชั่นที่เปิดเป็นบริการของระบบ ให้ผู้ร้องขอเข้ามาเรียกใช้ ซึ่งอาจเป็นซอฟต์แวร์ซอร์สโค้ดต่างๆ ซึ่งต้องทำการลงทะเบียนก่อนใช้ และอิมพลิเมนท์ตามคลาสแม่แบบ หรือ เทมเพลตคลาส (Template Class) เพื่อให้สามารถเข้าทำงานเป็นบริการส่วนหนึ่งของระบบได้อย่างถูกต้อง
- **ยูทิลิตี้ (Utility)** เป็นส่วนที่รวบรวมฟังก์ชั่นพิเศษที่ระบบจำเป็นต้องใช้ ทั้งเพื่อความสะดวกในการพัฒนา หรืออาจเป็นโครงสร้างข้อมูล (Data Structure) ที่สร้างขึ้นเพื่อใช้กับระบบโดยเฉพาะ เช่น โครงสร้างข้อมูลรีจิสทรี(Registry) โครงสร้างข้อมูลแบบลิงค์ลิสต์(Link list) เป็นต้น

3.4.3 คลาสไดอะแกรม (Class Diagram)

คลาสไดอะแกรมของระบบ มีดังนี้

- คลาสไดอะแกรมของอาร์พีซีเซอร์วิส
- คลาสไดอะแกรมของเซอร์วิสในระบบ
- คลาสไดอะแกรมของซ็อคเก็ตการเชื่อมต่อ
- คลาสไดอะแกรมของทาสก์แฟคตอรี่
- คลาสไดอะแกรมของเพอร์ซิสแทนต์ออบเจ็กต์
- คลาสไดอะแกรมของโซบพาร์เซอร์และเอนโค้ดเดอร์
- คลาสไดอะแกรมของโซบเอนเวลล็อฟและซีเรียลไรซ์

3.4.3.1 คลาสไดอะแกรมของอาร์พีซีเซอร์วิส



รูปภาพ 3-16 คลาสไดอะแกรมของอาร์พีซีเซอร์วิส

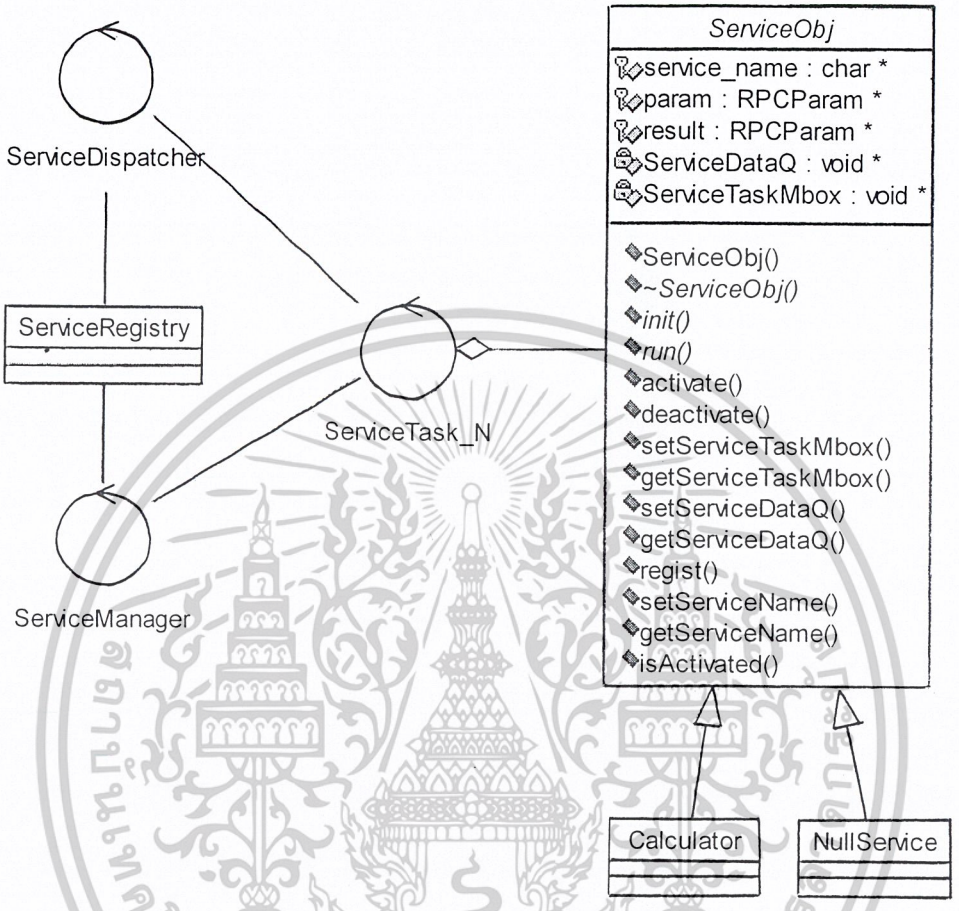
อาร์พีซีเซอร์วิส

- **ทาสก์ออบเจ็คต์ (Task Object)** เป็นคลาสที่เป็นเก็บรวบรวมตัวแปรต่างๆ ที่ใช้ในการควบคุมการทำงานของทาสก์ และใช้ในการติดต่อสื่อสารระหว่างการทำงานในทาสก์ต่างๆ คล้ายกับเป็นส่วนของทาสก์คอนโทรลบล็อคก์ (Task Control Block) ในระบบปฏิบัติการ โดยทาสก์ออบเจ็คต์นี้ จะเกิดขึ้นโดยการสร้างของทาสก์แฟคตอรี่ จากนั้นก็ทำหน้าที่เป็นตัวแทนในการติดต่อกับ

ทาสก์ต่างๆระหว่างการประสานงานกับทาสก์อื่นๆ จนกระทั่งจบกระบวนการทำงาน จึงถูกทำลายออบเจ็คต์นี้ไปจากระบบ

- **อาร์พีซีออบเจ็คต์ (RPC Object)** เป็นคลาสที่ทำหน้าที่จัดการเกี่ยวกับการขอเรียกใช้งานบริการในระบบ คล้ายกับการขอบริการแบบรีโมทโพรซีเยอร์คอล หรือ อาร์พีซี (Remote Procedure Call) ที่ทำหน้าที่ประสานงานกับส่วนของโปรโตคอลแฮนเดิ้ลเลอร์ (Protocol Handler) ที่ทำหน้าที่ในการจัดการแปลงรูปแบบของเอกสารในแมสเซจโปรโตคอลที่ใช้ในการทำอาร์พีซี ให้กลายเป็นรูปแบบของออบเจ็คต์ที่พร้อมในการร้องขอบริการจากส่วนบริการในระบบต่อไป รวมถึงนำผลลัพธ์ที่ได้จากการเรียกใช้บริการ โดยประสารงานกับส่วนของโปรโตคอลแฮนเดิ้ลเลอร์อีกครั้ง เพื่อทำให้ผลลัพธ์ที่ได้นั้นถูกบรรจุอยู่ในรูปแบบเอกสารตามโปรโตคอลที่ใช้ หลังจากได้เอกสารที่ในรูปแบบเดิมแล้ว ก็จะทำการส่งเอกสารนั้น กลับไปยังผู้รับโดยประสารงานกับส่วนของซ็อคเก็ต
- **อาพีซีพารามิเตอร์ (RPC Parameter)** เป็นคลาสที่เป็นโครงสร้างของข้อมูลในลักษณะของออบเจ็คต์พารามิเตอร์ (Object Parameter) ที่เก็บค่าตัวแปรที่เป็นพารามิเตอร์ที่ได้จากดึงข้อมูลตามแต่ละโปรโตคอล เพื่อข้อมูลเหล่านี้จะได้นำไปใช้ในการส่งค่าพารามิเตอร์ให้กับส่วนของบริการในระบบต่อไป อีกทั้งยังรับผลลัพธ์ที่ได้จากการทำงานของบริการในระบบ เพื่อประมวลผลเอกสารเพื่อการส่งกลับต่อไป ทั้งนี้เพื่อเป็นออบเจ็คต์มาตรฐานในระบบที่สามารถประสานรูปแบบโครงสร้างของการส่งข้อมูลระหว่างส่วนต่างๆ เพื่อให้มีความยืดหยุ่น โดยใช้หลักการของอินเตอร์เฟส ในแนวคิดแบบออบเจ็คต์โอเรียนเต็ด (Object-Oriented Paradigm) ที่สามารถรองรับการพัฒนาหรืออิมพลิเมนต์ส่วนของโปรโตคอลแฮนเดิ้ลเลอร์ รวมถึงบริการต่างๆ ที่จะเข้ามาเพิ่มความสามารถในระบบ ให้ทำงานกับระบบเดิมได้อย่างไม่มีปัญหา
- **อาร์พีซีโหนด (RPC Node)** เป็นโครงสร้างข้อมูลในระดับย่อยภายในอาร์พีซีพารามิเตอร์ ซึ่งมีโครงสร้างข้อมูลในลักษณะของลิงก์ลิสต์ (Linklist) ที่ง่ายต่อการจัดการ และสามารถขยายหน่วยข้อมูลได้เรื่อยๆ เหมาะกับการนำมาใช้รองรับค่าพารามิเตอร์ที่ใช้ในการเรียกใช้บริการ ที่มีความไม่แน่นอนของจำนวนพารามิเตอร์ที่ใช้ อีกทั้งยังกำหนดประเภทของข้อมูลพื้นฐานให้เป็นแบบสตริง (String) เพื่อที่จะรองรับความหลากหลายของชนิดของข้อมูล ซึ่งสตริงมีความสามารถในการแปลงข้อมูลไปสู่ข้อมูลแบบอื่นๆ ได้ง่าย ผ่านฟังก์ชั่นของการแปลงชนิดข้อมูลที่มีอยู่ในคอมไพเลอร์

3.4.3.2 คลาสไดอะแกรมของเซอร์วิสในระบบ



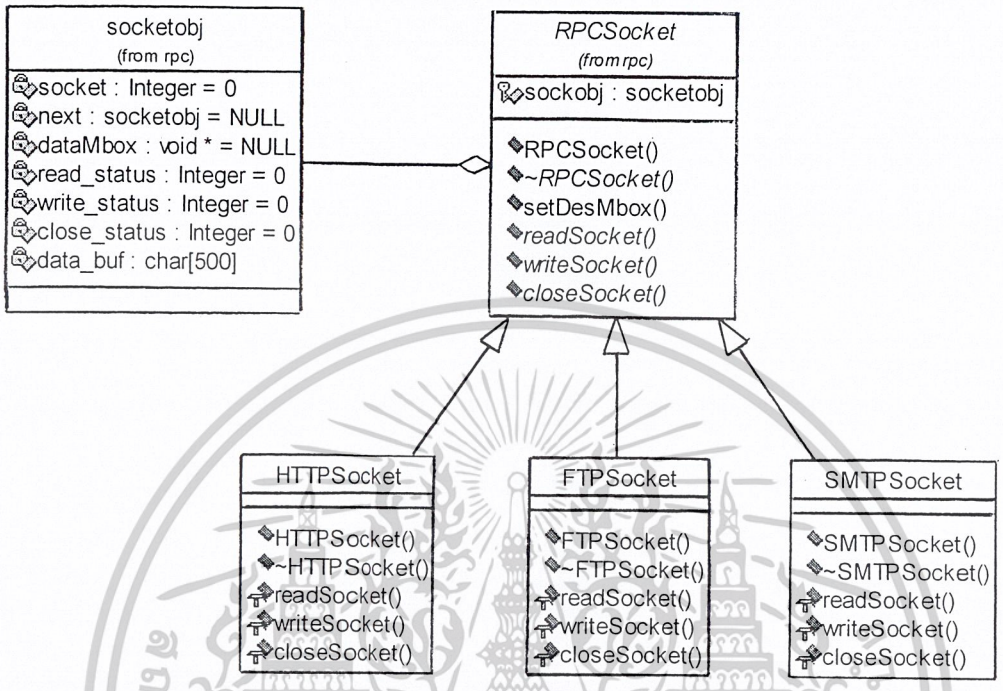
รูปภาพ 3-17 คลาสไดอะแกรมของเซอร์วิสในระบบ

เซอร์วิสในระบบ

- **เซอร์วิสออบเจ็กต์ (Service Object)** เป็นคลาสที่จัดการการทำงานของเซอร์วิสให้เป็นไปตามรูปแบบของการเรียกใช้บริการแบบอาร์พีซี เพื่อเป็นรูปแบบที่กำหนดขึ้นเพื่อสามารถทำงานร่วมกับระบบได้อย่างถูกต้อง โดยมีการรับพารามิเตอร์มาในรูปแบบของอาร์พีซีพารามิเตอร์ แล้วทำการแปลงค่าพารามิเตอร์จากรูปแบบของสตริง ให้เป็นข้อมูลที่มีชนิดที่เซอร์วิสฟังก์ชั่นสามารถนำไปประมวลภายในฟังก์ชั่นได้ รวมถึงการแปลงผลลัพธ์ที่ได้กลับไปเป็นข้อมูลชนิดสตริงเพื่อส่งผลลัพธ์กลับไป โดยในเซอร์วิสออบเจ็กต์นี้ ต้องมีส่วนของเซอร์วิสฟังก์ชั่นที่เป็นหัวใจหลักในการประมวลผลตามแต่ละบริการ โดยคลาสนี้ออกแบบให้เป็นแอพสแตร็กคลาส (Abstract Class) เพื่อความสะดวกในการพัฒนาเซอร์วิสนั้น และเป็นลักษณะบังคับให้มีลักษณะของโค้ดที่ตรงกับรูปแบบที่สามารถทำงานร่วมกับระบบได้

- **เซอร์วิสรีจิสทรี้ (Service Registry)** เป็นโครงสร้างข้อมูลที่ทำขึ้นจากคลาสของรีจิสทรี้ที่มีอยู่ในยูทิลิตี้แพคเกจ (Utility Package) เพื่อเป็นข้อมูลที่เก็บข้อมูลเบื้องต้นของเซอร์วิสที่มีอยู่ในระบบ จนถึงมีข้อมูลที่ใช้ในการควบคุมการเปิดให้ใช้บริการต่างๆในระบบโดยเซอร์วิสดิสแพตเชอร์ทาสก์ที่ทำหน้าที่ในการควบคุมและเรียกหาบริการ

3.4.3.3 คลาสไดอะแกรมของซ็อคเก็ตการเชื่อมต่อ

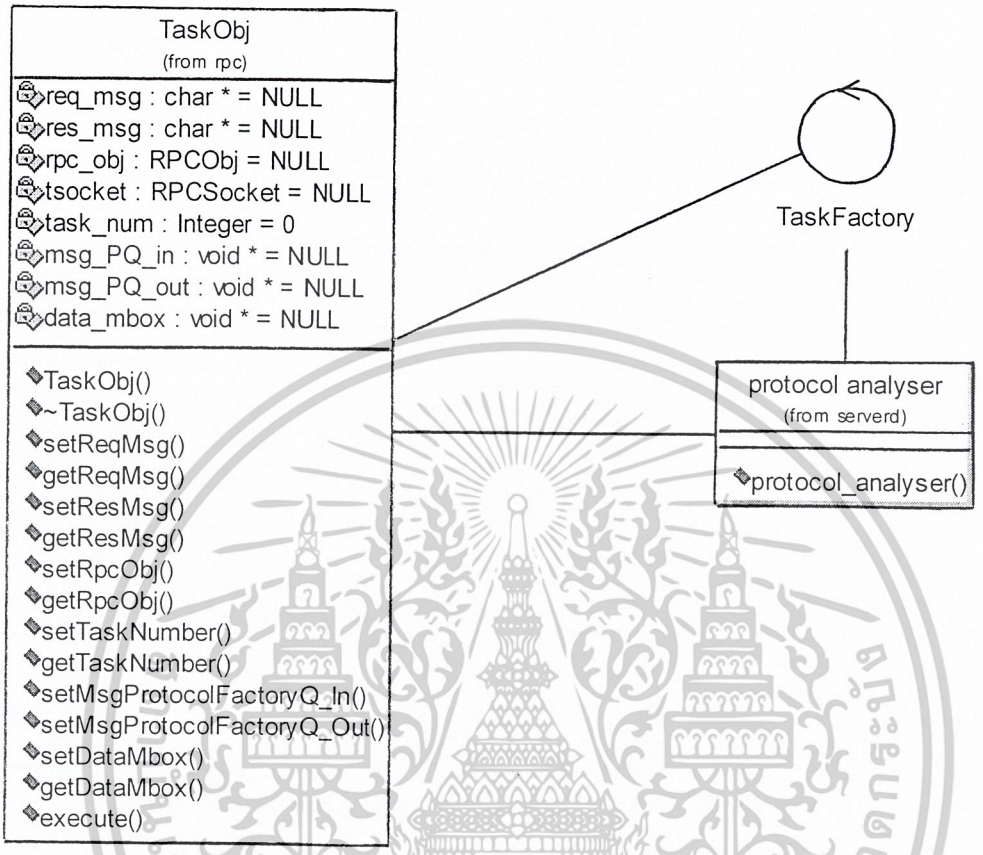


รูปภาพ 3-18 คลาสไดอะแกรมของซ็อคเก็ตการเชื่อมต่อ

ซ็อคเก็ตการเชื่อมต่อ

- อาร์พีซีซ็อคเก็ต (RPC Socket) เป็นคลาสที่ทำหน้าที่เป็นตัวแทนของทาสก์ในการเชื่อมต่อสื่อสารข้อมูลกับโมดูลที่จัดการการสื่อสารในชั้นของทรานสปอร์ต (Transport Layer) ด้วยโปรโตคอลทีซีพี(TCP) หรือยูดีพี (UDP) โดยจะเก็บรวบรวมข้อมูลและตัวแปรที่จำเป็นในการเชื่อมต่อเซสชั่นกับผู้เรียกใช้บริการของระบบ เพื่อระบบจะสามารถรองรับการเชื่อมต่อได้หลายๆ เซสชั่นในขณะเดียวกัน
- ซ็อคเก็ตออบเจ็คต์ (Socketobj) เป็นโครงสร้างข้อมูลที่อยู่ในอาร์พีซีซ็อคเก็ต ที่ทำหน้าที่สื่อสารข้อมูลกับส่วนของเซิร์ฟเวอร์ หรือส่งข้อมูลผ่านซ็อคเก็ต

3.4.3.4 คลาสไดอะแกรมของทาสก์แฟคตอรี่

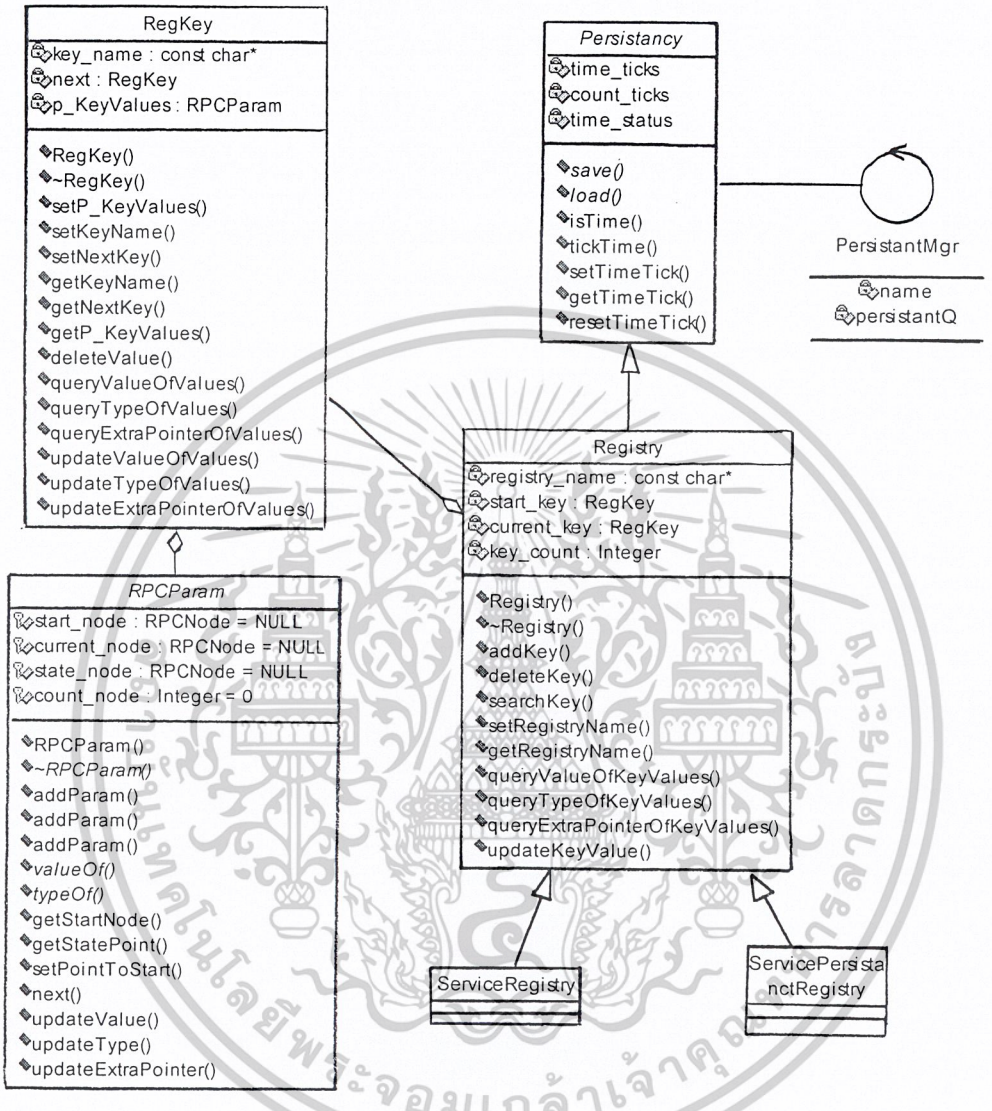


รูปภาพ 3-19 คลาสไดอะแกรมของทาสก์แฟคตอรี่

ทาสก์แฟคตอรี่

- **โปรโตคอลอนาไลเซอร์ (Protocol Analyser)** เป็นฟังก์ชั่นในการวิเคราะและกำหนดรูปแบบของการเรียกขอใช้บริการ เพื่อสร้างออบเจ็กต์ของอาร์พีซีให้ตรงกับรูปแบบดังกล่าว (ที่มีอยู่ในระบบ) โดยเป็นเหมือนฟังก์ชั่นในการดิสแพตช์ โปรโตคอลไปตามโปรโตคอลแฮนเดิ้ลเลอร์ที่ตรงกับรูปแบบตามช่องทางการเชื่อมต่อ เช่น หากเป็นโปรโตคอลเอชทีทีพี ก็จะสร้างอาร์พีซีออบเจ็กต์แบบโซบขึ้นมารองรับและทำหน้าดำเนินการในระบบสำหรับรีเควสนั้นๆ ต่อไป

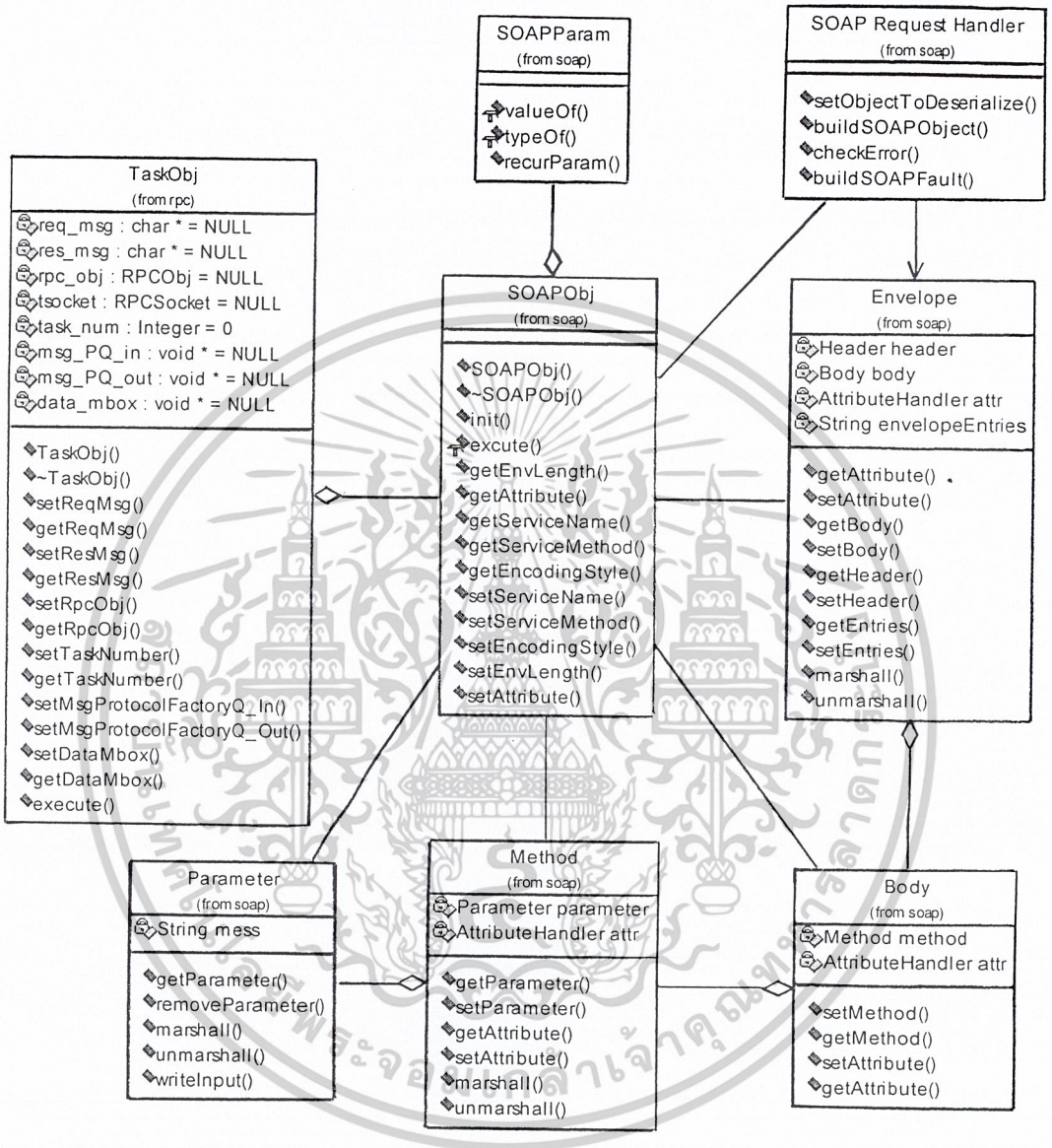
3.4.3.5 คลาสไดอะแกรมของเพอร์ซิสแทนต์ออบเจ็กต์



รูปภาพ 3-20 คลาสไดอะแกรมของเพอร์ซิสแทนต์ออบเจ็กต์

- **เพอร์ซิสแทนต์ออบเจ็กต์ (Persistance Object)** เป็นคลาสที่ทำหน้าที่ในการบันทึกข้อมูลที่เป็นชนิดรีจิสทรี้ลงในเซคคั่นนารี่สตอเรจ (Secondary Storage) เพื่อคงสถานะของเซอร์วิสทาสก์ที่ต้องการเก็บค่าที่ไม่สูญหายหลังจากการหยุดทำงานของระบบ โดยทำงานร่วมกับเพอร์ซิสแทนต์ทาสก์ที่คอยบันทึกข้อมูลในรีจิสทรี้เหล่านั้นไว้ตามระยะเวลาที่ถูกกำหนดไว้ในเพอร์ซิสแทนต์ออบเจ็กต์ โดยในการสร้างออบเจ็คต์ที่เป็นแบบเพอร์ซิสแทนต์ออบเจ็กต์นั้น จะต้องเอ็กซ์เทนต่อมาจากคลาสเพอร์ซิสแทนต์ออบเจ็กต์ ซึ่งเป็นคลาสแบบแอพสแตร็กต์คลาส (Abstract Class)

3.4.3.6 คลาสไดอะแกรมของโซบพาร์เซอร์และเอนโค้ดเดอร์



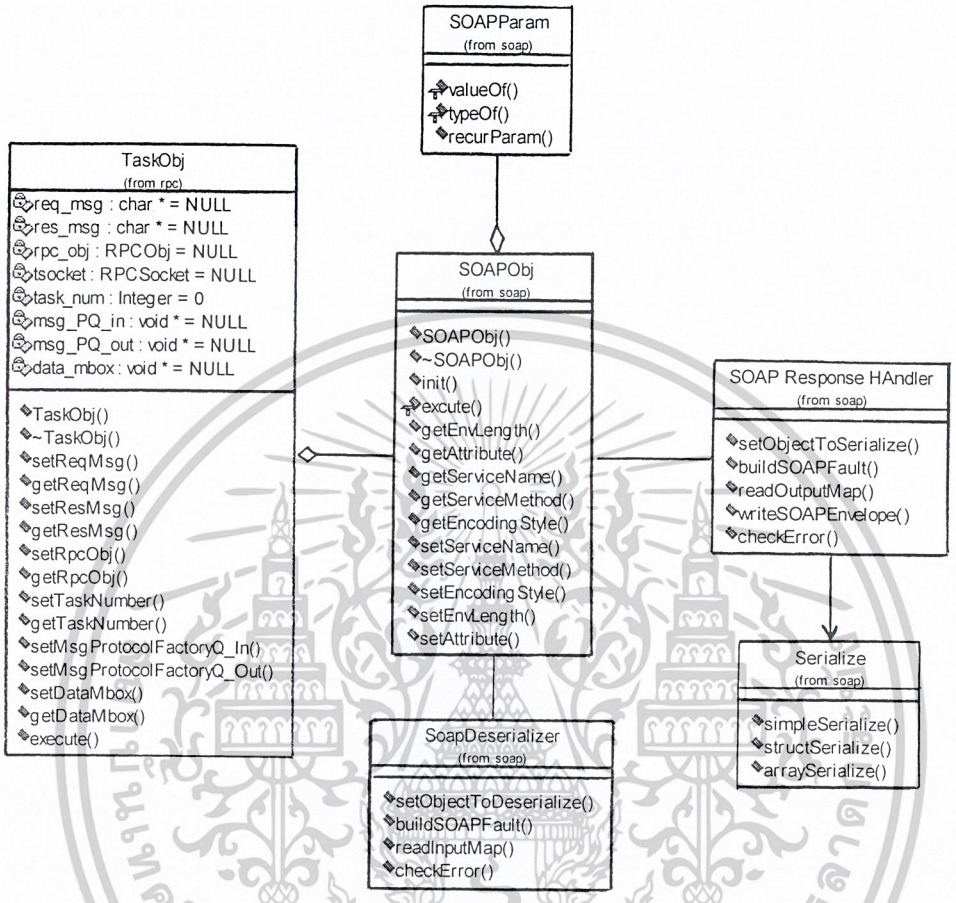
รูปภาพ 3-21 คลาสไดอะแกรมของโซบพาร์เซอร์และเอนโค้ดเดอร์

- **โซบออบเจ็คต์ (SOAP Object)** เป็นคลาสที่อิมพลิเมนต์ต่อจากอาร์พีซีออบเจ็คต์ เพื่อเป็นคลาสที่รองรับและจัดการการเรียกใช้บริการอาร์พีซีในแบบของเว็บเซอร์วิส โดยคลาสนี้จะทำการควบคุมกระบวนการทำงานตั้งแต่ส่งเอกสารที่ได้รับจากผู้ขอใช้บริการ แปลงข้อมูลแล้วเก็บในอีดับบิว

เอสพารามิเตอร์ (EWS Parameter) ซึ่งอิมพลิเมนต์ต่อมาจากอาร์พีซีพารามิเตอร์เพื่อทำหน้าที่เฉพาะเกี่ยวกับการแปลงเอกสารโซบเพิ่มเติมขึ้นเองจากส่วนของอาร์พีซีพารามิเตอร์ที่มีอยู่แล้ว

- **คลาสเอลวิล๊อพ (Envelope Class)** เป็นคลาสที่ใช้แทนส่วนของเอลวิล๊อพ ภายในโซบแมสเซจซึ่งจะมีแอททริบิว คือ คลาสเฮดเดอร์ , คลาสบอดี้ และ คลาสแอททริบิวแฮนเดิ้ลเลอร์ (AttributeHandler Class) ซึ่งจะมีฟังก์ชันในการกำหนดค่า และเรียกค่าคืนกลับจากคลาสที่เป็นสมาชิกส่วนตัวในคลาสฟังก์ชันในการสร้างโซบแมสเซจ และฟังก์ชันในการนำโซบแมสเซจมาสร้างเป็นคลาส
- **คลาสเฮดเดอร์ (Header Class)** เป็นคลาสที่ใช้แทนส่วนหัวของโซบ (SOAP Header) ภายในโซบแมสเซจซึ่งจะมีแอททริบิว คือ คลาสพารามิเตอร์เฮดเดอร์เอนติตี้ (HeaderEntries Class) และคลาสแอททริบิวแฮนเดอร์เลอร์ ซึ่งจะมีฟังก์ชันในการกำหนดค่า และเรียกค่าคืนกลับจากคลาสที่เป็นสมาชิกส่วนตัวในคลาส ฟังก์ชันในการสร้าง โซบแมสเซจ และฟังก์ชันในการนำโซบแมสเซจ มาสร้างเป็นคลาส
- **คลาสบอดี้ (Body Class)** เป็นคลาสที่ใช้แทนส่วนเนื้อหาของโซบ (SOAP Body) ภายในโซบแมสเซจ ซึ่งจะมีแอททริบิว คือ คลาส Method และ คลาสแอททริบิวแฮนเดอร์เลอร์ ซึ่งจะมีฟังก์ชันในการกำหนดค่า และเรียกค่าคืนกลับจากคลาสที่เป็นสมาชิกส่วนตัวในคลาส ฟังก์ชันในการสร้างโซบแมสเซจ และฟังก์ชันในการนำโซบแมสเซจ มาสร้างเป็นคลาส
- **คลาสเมทธอด (Method Class)** เป็นคลาสที่ใช้แทนส่วนเมธอดของส่วนเนื้อหาของโซบ ภายในโซบแมสเซจ ซึ่งจะมีแอททริบิว คือ คลาสเมธอด และ คลาสแอททริบิวแฮนเดอร์เลอร์ ซึ่งจะมีฟังก์ชันในการกำหนดค่า และเรียกค่าคืนกลับจากคลาสที่เป็นสมาชิกส่วนตัวในคลาส ฟังก์ชันในการสร้างโซบแมสเซจ และฟังก์ชันในการนำโซบแมสเซจ มาสร้างเป็นคลาส
- **คลาสพารามิเตอร์ (Parameter Class)** เป็นคลาสที่ใช้แทนส่วนพารามิเตอร์ภายในคลาสเมธอด ซึ่งจะมีแอททริบิว คือ คลาสพารามิเตอร์เอนติตี้ (ParameterEntries Class) และ คลาสแอททริบิวแฮนเดอร์เลอร์ ซึ่งจะมีฟังก์ชันในการกำหนดค่า และเรียกค่าคืนกลับจากคลาสที่เป็นสมาชิกส่วนตัวในคลาส ฟังก์ชันในการสร้างโซบแมสเซจ และฟังก์ชันในการนำโซบแมสเซจ มาสร้างเป็นคลาส
- **คลาสโซบรีเควสแฮนเดอร์เลอร์ (SOAPRequestHandler)** เป็นคลาสที่ใช้ในการจัดการโซบแมสเซจ ที่เข้ามาเพื่อเรียกใช้บริการเว็บเซอร์วิส โดยจะมีฟังก์ชันที่ใช้ในการตรวจสอบรายชื่อ และเมธทอดของบริการนั้นว่ามีภายในระบบหรือไม่ ฟังก์ชันที่ใช้ในการกำหนดส่วนกำหนดข้อผิดพลาดของโซบ (SOAP Fault) ฟังก์ชันที่เรียกใช้คลาสดีซีรีไรซ์เซอร์ เพื่อทำการแปลงค่าที่ถูกกำหนดรูปแบบภายในโซบแมสเซจตามข้อกำหนดในส่วนของการเข้ารหัสของโซบให้เป็นชนิดของข้อมูล และฟังก์ชันที่ใช้ในการเขียนข้อมูลนำเข้าไปยังไฟล์ และฟังก์ชันที่ใช้ในการตรวจสอบไวยากรณ์ของเอกสารว่าตรงกับมาตราฐานโซบ ฟังก์ชันที่ใช้ในการตรวจสอบรายการของพารามิเตอร์นำเข้าว่าตรงกับรายละเอียดของบริการหรือไม่

3.4.3.7 คลาสไดอะแกรมของโซบเอนเวลล๊อฟและซีเรียลไรซ์

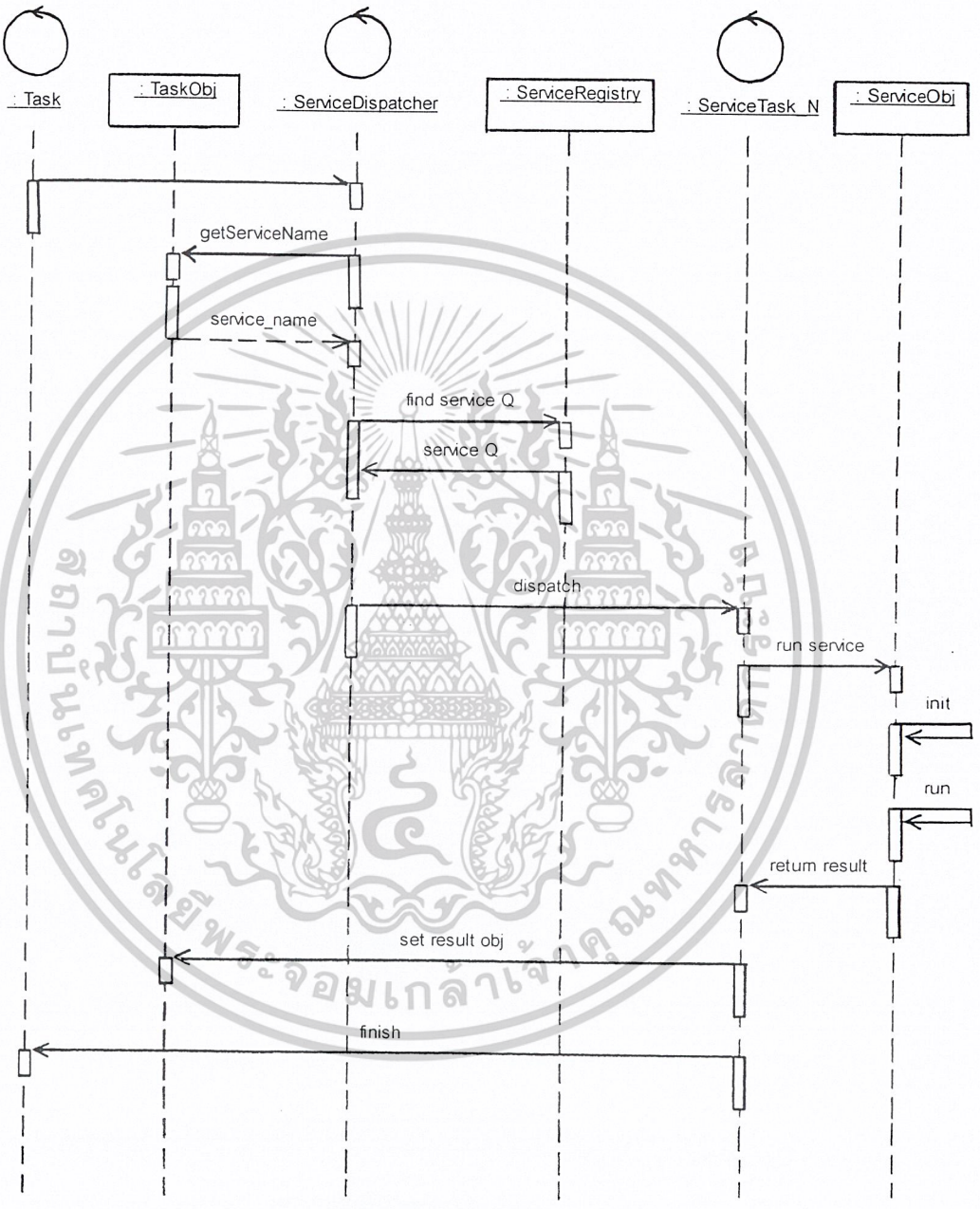


รูปภาพ 3-22 คลาสไดอะแกรมของโซบเอนเวลล๊อฟและซีเรียลไรซ์

- **คลาสซีเรียลไรซ์เซอร์ (Serializer Class)** เป็นคลาสที่ใช้ในการแปลงชนิดของข้อมูลให้เป็นค่าที่ถูกกำหนดรูปแบบภายในโซบแมสเซจ ตามข้อกำหนดในส่วนของการเข้ารหัสของโซบ ซึ่งจะมีฟังก์ชันที่ใช้ในการแปลงชนิดจากชนิดของข้อมูลพื้นฐาน ,ชนิดข้อมูลแบบอะเรย์ และชนิดข้อมูลแบบคลาสหรือโครงสร้าง ให้เป็นรูปแบบของการเข้ารหัสของโซบ
- **คลาสโซบเรสสปอนแฮนเดอร์เออร์ (SOAPReponseHandler Class)** เป็นคลาสที่ใช้ในการสร้างโซบแมสเซจ เพื่อตอบรับการเรียกใช้บริการเว็บเซอร์วิส โดยจะมีฟังก์ชันที่ใช้ในการสร้างโซบแมสเซจของบริการนั้นโดยดูจากรายละเอียดของบริการนั้น ฟังก์ชันที่เรียกใช้คลาสซีรีไรซ์เซอร์เพื่อทำการแปลงชนิดของข้อมูลให้เป็นค่าที่ถูกกำหนดรูปแบบภายในโซบแมสเซจ ตามข้อกำหนดในส่วนของการเข้ารหัสของโซบ

3.4.4 ซีเควนซ์ไดอะแกรม (Sequence Diagram)

3.4.4.1 การเรียกใช้บริการ

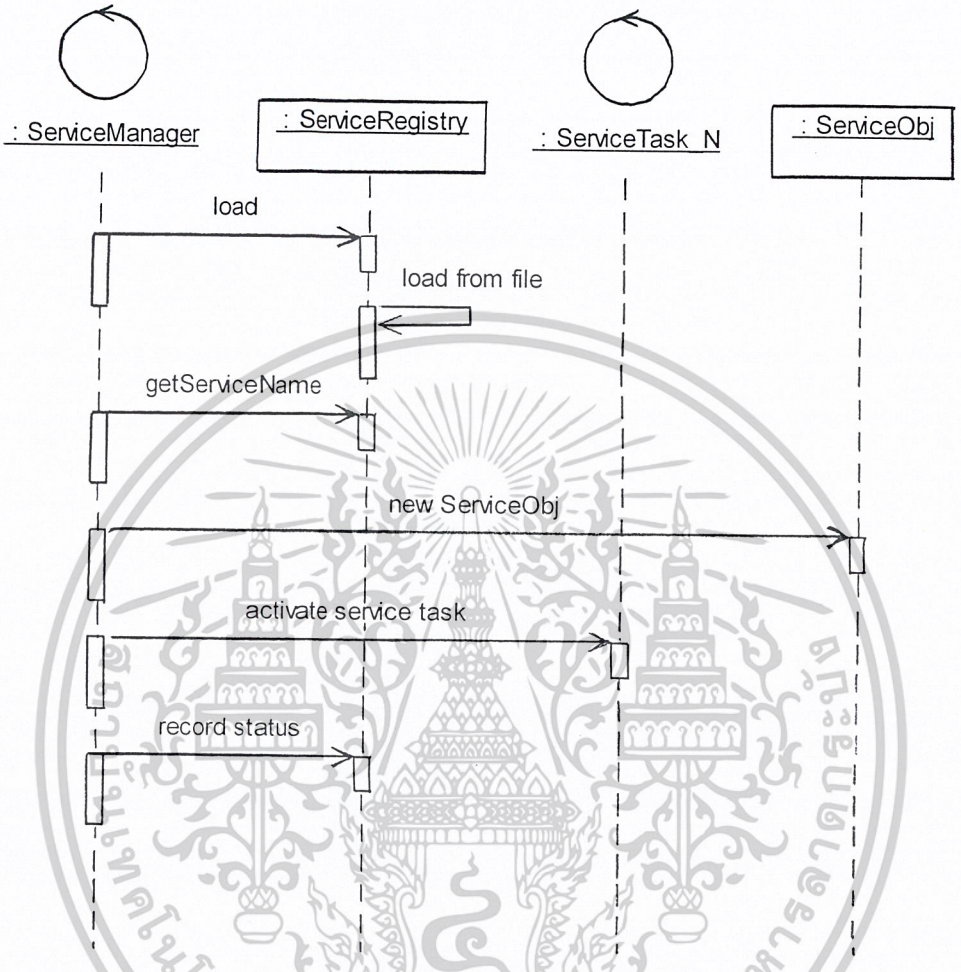


รูปภาพ 3-23 การเรียกใช้บริการ

อธิบายการทำงาน

หลังจากที่ทาสก์ได้รับการสร้างขึ้นสำหรับรีเควสหนึ่งแล้ว กระบวนการต่างๆ จะถูกกระทำโดยทาสก์ โดยภายในทาสก์ มีทาสก์ออบเจ็คต์ที่ทำหน้าที่ควบคุมการทำงานของทาสก์ ส่วนงานในด้านการติดต่อเพื่อดำเนินการในการเรียกขอใช้บริการนั้น จะมอบให้อาร์พีซีออบเจ็คต์ที่อยู่ในทาสก์ออบเจ็คต์อีกทีหนึ่ง แต่เมื่อมองดูจากภายนอกแล้ว จะเป็นการทำงานโดยทาสก์เป็นตัวดำเนินงาน เมื่อทาสก์ถึงขั้นตอนที่จะทำการเรียกใช้บริการแล้ว จะส่งแมจเซจไปยังทาส์กที่ทำหน้าที่เป็นเซอร์วิสดิสแพตเชอร์ เพื่อที่จะให้เซอร์วิสดิสแพตเชอร์ทำการค้นหาบริการจากรายการในเซอร์วิสรีจิสทรี้ แล้วดูว่าอยู่ในสถานะที่เปิดให้ใช้บริการอยู่หรือไม่ ถ้าเปิดให้ใช้บริการ ก็จะค้นหาถึงตำแหน่งของแมจเซจคิวของเซอร์วิสนั้น เปรียบเสมือนเลขที่บ้าน โดยถ้าเซอร์วิสนั้นไม่มีการทำงานให้กับรีเควสอื่นอยู่ก่อนแล้ว เซอร์วิสนั้นก็จะบริการให้รีเควสนั้นทันที แต่ถ้ายังมีการทำงานให้กับรีเควสอื่นๆอยู่ แมจเซจนั้นก็ยังคงอยู่ในคิวของเซอร์วิสนั้นต่อไปจนกว่าเซอร์วิสนั้นจะทำงานลุล่วง แล้วจึงนำรีเควสที่เก็บอยู่ในคิวมาให้บริการต่อไป จากนั้น ผลลัพธ์จะถูกส่งตรงไปยังทาสก์ที่เป็นผู้เรียก โดยใส่ไว้ยังเมล์บอซ์ของทาสก์ แล้วทำการดำเนินการเพื่อจัดส่งผลลัพธ์กลับไป

3.4.4.2 การเปิดบริการ

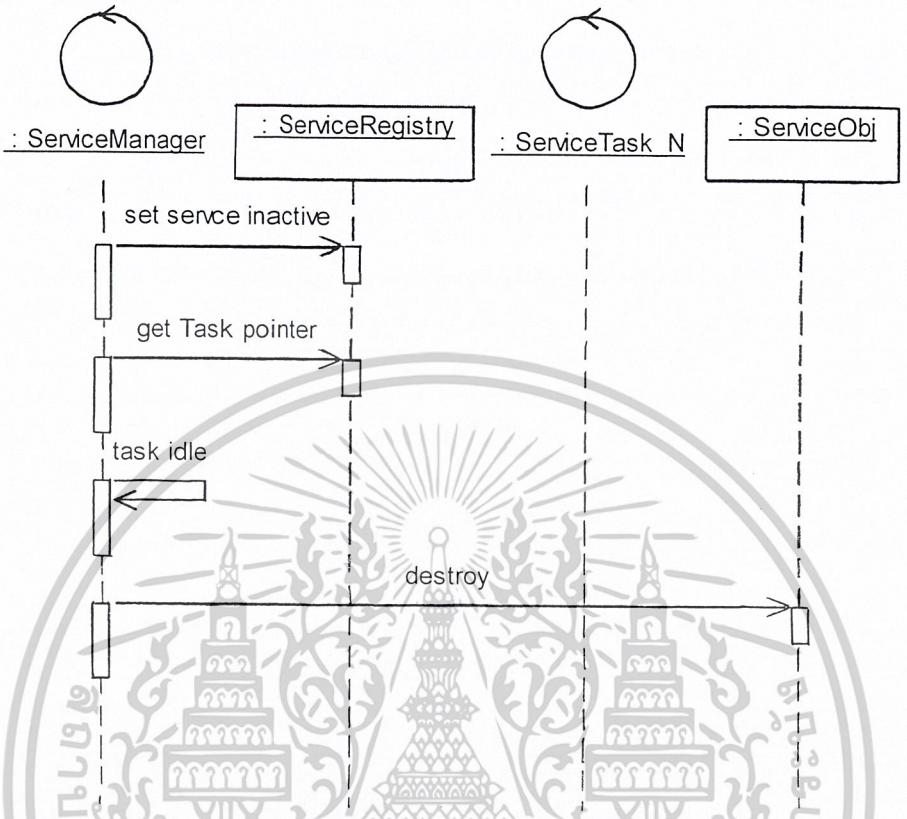


รูปภาพ 3-24 การเปิดบริการ

อธิบายการทำงาน

ผู้ดูแลระบบสามารถทำการควบคุมเปิดหรือปิดบริการที่มีอยู่ในระบบ ผ่านส่วนที่ทำหน้าที่เป็นเซอร์วิสเมเนเจอร์ ซึ่งเซอร์สิสเมเนเจอร์นี้ เป็นทาสก์ๆหนึ่งที่เป็นในลักษณะของเซอร์วิสทาสก์อันหนึ่งที่คอยควบคุมการทำงานของเซอร์วิสทาสก์อื่นๆอีกที การสั่งการไปยังเซอร์วิสเมเนเจอร์นั้นทำได้โดยส่งค่าที่มีรูปแบบในการควบคุมเซอร์วิสตามที่กำหนดขึ้น ไปยังคิวของเมเนเจอร์เซอร์วิส แล้วเมเนเจอร์เซอร์วิสก็จะนำชื่อของเซอร์วิสที่ได้มานั้น ไปหารายชื่อเซอร์วิสในเซอร์วิสรีจิสทรี้ของระบบ แล้วทำการเปลี่ยนแปลงสถานะของเซอร์วิสนั้นให้อยู่ในสถานะที่พร้อมให้บริการ จากนั้นเซอร์วิสรีจิสทรี้ก็จะสร้างเซอร์วิสออบเจ็คต์ขึ้นมาแล้วส่งทำงานที่เซอร์วิสทาสก์ที่ว่างอยู่ เพื่อให้เซอร์วิสทาสก์นั้น เริ่มรอการให้บริการต่อไป

3.4.4.3 การปิดบริการ



รูปภาพ 3-25 การปิดบริการ

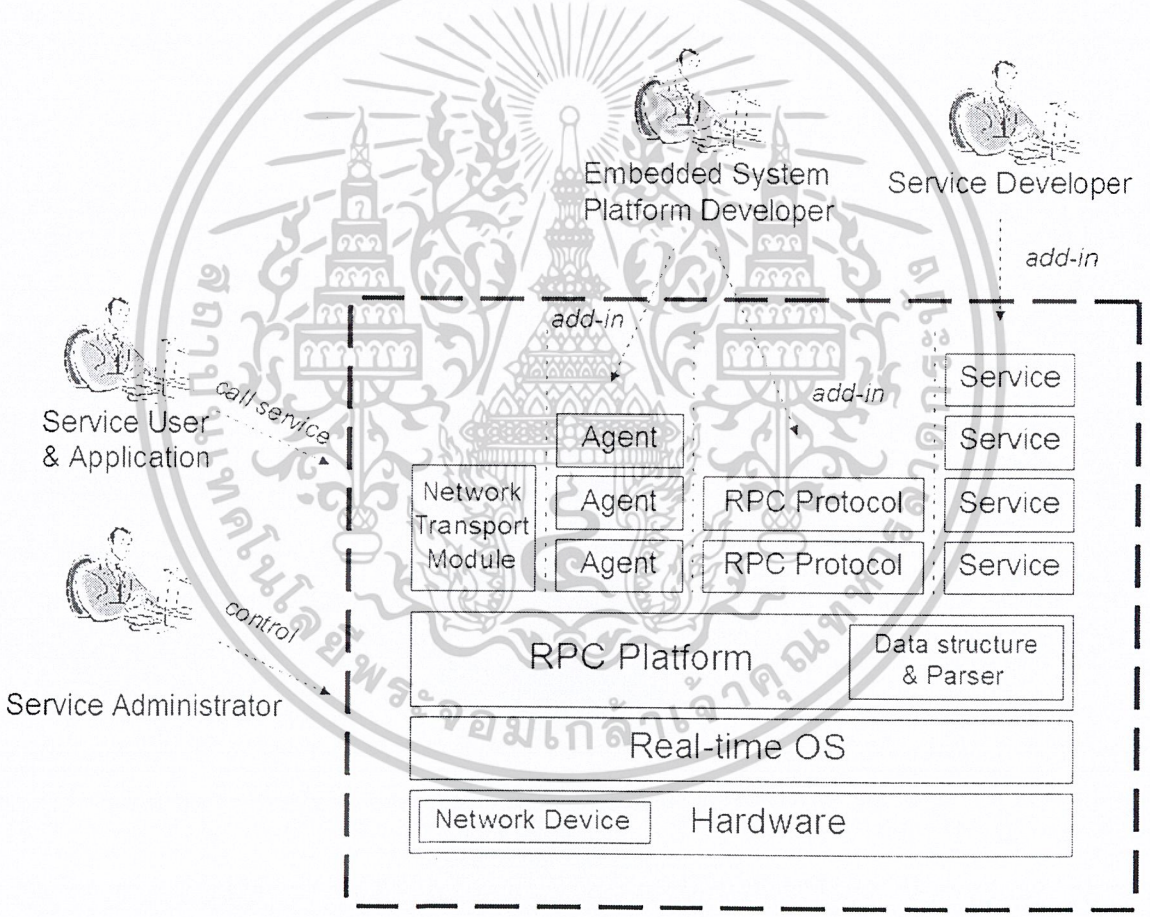
อธิบายการทำงาน

เมื่อเซอร์วิสเมเนเจอร์ได้รับคำสั่งในการปิดบริการเซอร์วิสที่ทำงานอยู่ในระบบ ทำได้โดยการเปลี่ยนแปลงสถานะของรายการเซอร์วิสที่อยู่ในเซอร์วิสเมเนเจอร์ ให้อยู่ในสถานะปิด จากนั้นก็ดึงเอาข้อมูลที่เป็นพอย์เตอร์ของเซอร์วิสทาสก์ของเซอร์วิสนั้น แล้วทำการจบการทำงานของเซอร์วิสทาสก์นั้นให้อยู่ในสถานะที่ว่าง แล้วนำเซอร์วิสออบเจ็คต์มาลบออกจากระบบ

บทที่ 4
ลักษณะการใช้งานเว็บเซอร์วิสบนอุปกรณ์ฝังตัว

ลักษณะการใช้งานเว็บเซอร์วิสบนระบบฝังตัวนั้นสามารถแบ่งได้ออกเป็น 4 ลักษณะ คือ

1. นักพัฒนา
2. นักพัฒนาบริการบนอุปกรณ์ฝังตัว
3. ผู้ที่เรียกใช้บริการของระบบ
4. ผู้ดูแลระบบการให้บริการของระบบ



รูปภาพ 4-1 บทบาทของผู้ที่เกี่ยวข้องกับระบบ

4.1 นักพัฒนา

เป็นผู้ที่นำโค้ดของระบบ ไปพัฒนาในส่วนต่างๆต่อเพื่อนำไปใช้ในงานเฉพาะอย่างที่ต้องการ โดยสามารถเพิ่มส่วนต่างๆ ดังนี้

- **โปรโตคอลแฮนเดิ้ลเลอร์ (Protocol Handler)**

เป็นส่วนที่รองรับการติดต่อในรูปแบบของโปรโตคอลในมาตรฐานต่างๆ โดยส่วนนี้ ได้ถูกออกแบบให้สามารถขยายและเพิ่มเติมโปรโตคอลแฮนเดิ้ลเลอร์ได้โดยนักพัฒนา โดยมีรูปแบบดังนี้

```
Protocol Handler In Task()
{
RPCParam *param;
MessageQ *mq;
        for(;;){
          //wait for request
          //handle protocol to parameterobject
          //save in parameter object
          //save session infomation in RPCobject
          //post return to requested task
        }
}
```

รูปภาพ 4-2 โครงสร้างการพัฒนาโปรโตคอลแฮนเดิ้ลเลอร์ในส่วนของการดึงข้อมูลมาสู่อาร์พีซีพารามิเตอร์

จากรูปโครงสร้างของโปรแกรมในส่วนของโปรโตคอลแฮนเดิ้ลเลอร์นี้ เป็นส่วนที่ทำหน้าที่อ่านข้อมูลจากมาตรฐานโปรโตคอล แล้วสร้างเก็บอาร์พีซีพารามิเตอร์เพื่อเก็บข้อมูลที่อ่านได้ แล้วเก็บบันทึกไว้ในพารามิเตอร์ออบเจ็คต์นั้น แล้วทำการเก็บบันทึกค่าที่จำเป็นต้องใช้ในการติดต่อในเซสชั่นของโปรโตคอลนั้นเอาไว้ในอาร์พีซีออบเจคต์ เมื่อทำงานเสร็จแล้ว ก็ทำการส่งแมจเซจไปยังทาสก์ที่เรียกใช้เพื่อให้ทำงานในลำดับอื่นๆต่อไป

```
Protocol Handler Out Task()
{
RPCParam *result;
MessageQ *mq;
        for(;;){
          //wait for request
          //encode and serial data to message protocol
          //post return to requested task
        }
}
```

รูปภาพ 4-3 โครงสร้างของโปรโตคอลแฮนเดิ้ลเลอร์สำหรับแปลงข้อมูลกลับไปเป็นเอกสารของโปรโตคอล

จากรูป เป็นโครงสร้างของโปรโตคอลแฮนเดิ้ลเลอร์ที่ทำหน้าที่แปลงข้อมูลที่เป็นผลลัพธ์ของการเรียกใช้บริการ ให้อยู่ในรูปของเอกสารตามมาตรฐานของแต่ละโปรโตคอล แล้วส่งผลกลับไปยังทาสก์ที่ร้องขอต่อไป

เนื่องจากระบบอาร์พีซีนี้ มีรากฐานมาจากแนวความคิดของสถาปัตยกรรมการทำงานแบบเว็บเซอร์วิส และเป็นโครงงานที่พัฒนาต่อ จึงมีการดัดแปลงโซบโปรโตคอลแฮนเดิ้ลเลอร์ มาเป็นโปรโตคอลแฮนเดิ้ลเลอร์ที่เตรียมไว้ให้ในระบบแล้ว เพื่อสามารถนำไปใช้ทำการสื่อสารแบบเว็บเซอร์วิสได้

- **เซิร์ฟเวอร์แดมอนทาสก์ (Server Daemon Task)**

เป็นส่วนที่คอยรอรับการขอทำการเชื่อมต่อกับผู้ร้องขอบริการ ซึ่งต้องรูปแบบในการติดต่อในข้อมูลในระดับของเซสชั่น (Session Layer) โดยรองรับโปรโตคอลที่ทำหน้าที่เป็นผู้รับส่งข้อมูล (Agent) ซึ่งได้แก่ โปรโตคอลเอชทีทีพี โปรโตคอลเอฟทีพี หรือ โปรโตคอลเอสเอ็มทีพี เป็นต้น ในการพัฒนาเซิร์ฟเวอร์แดมอนนั้น จะควรจะพัฒนาตามรูปแบบของโครงสร้างที่ได้ออกแบบไว้ตามหน้าที่การทำงานของเซิร์ฟเวอร์แดมอน ดังนี้

```
<serverd>_init();
<serverd>_thread();
<serverd>_close();
do_read();
```

รูปภาพ 4-4 โครงสร้างฟังก์ชั่นภายใน ในการอิมพลิเมนต์เซิร์ฟเวอร์แดมอน

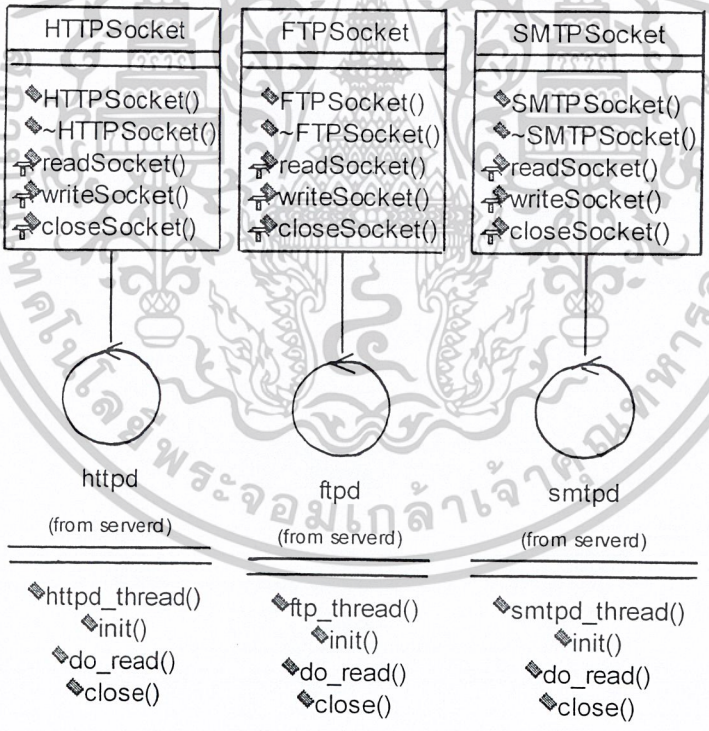
(*หมายเหตุ : <serverd> คือ ชื่อของโปรโตคอลใดๆ ที่จะถูกอิมพลิเมนต์เข้ามาใช้ในระบบ)

<serverd>_init()	เป็นฟังก์ชั่นที่เริ่มการทำงานของระบบ โดยจะเรียกฟังก์ชั่นของระบบที่ทำหน้าที่สร้างเทรด (thread) ของเซิร์ฟเวอร์แคมอน เพื่อรอรับการร้องขอเชื่อมต่อในโปรโตคอลที่เซิร์ฟเวอร์แคมอนนั้นๆ รองรับ
<serverd>_thread()	เป็นฟังก์ชั่นที่เป็นส่วนที่ประมวลผลของเทรดของเซิร์ฟเวอร์แคมอน มีลักษณะคล้ายทาสก์ ที่วนรอบการทำงานตลอดเวลา
<serverd>_close()	เป็นฟังก์ชั่นที่ปิดการเชื่อมต่อในเซสชั่นนั้นๆ
do_read()	เป็นฟังก์ชั่นในการอ่านค่าข้อมูลเข้ามาในเซิร์ฟเวอร์แคมอนแล้วเก็บลงในโครงสร้างข้อมูลที่เตรียมไว้

ตาราง 4-1 อธิบายฟังก์ชั่นในการอิมพลิเมนต์เซิร์ฟเวอร์แคมอน

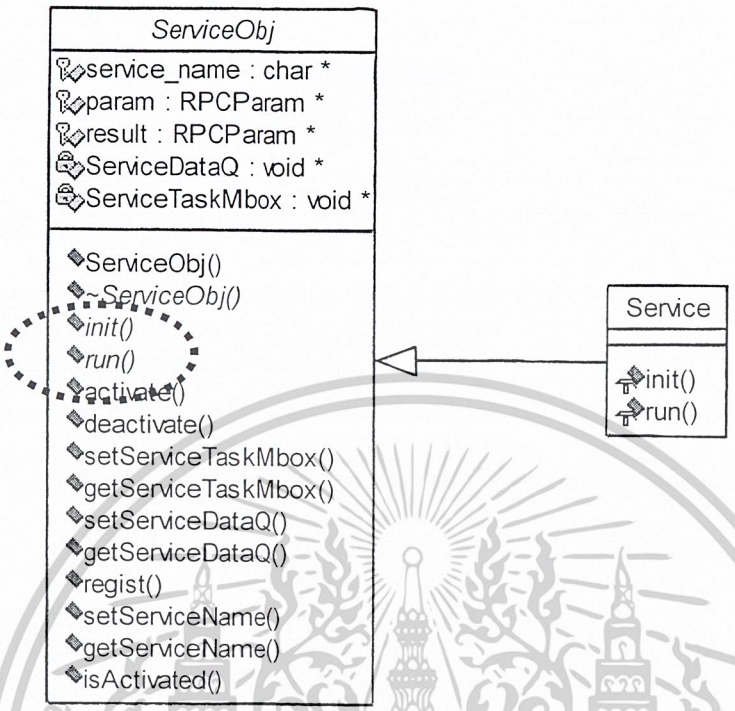
- โปรโตคอลซ็อคเก็ต **(Protocol Socket)**

เป็นคลาสที่จัดการการเชื่อมต่อของต่อละเซสชั่น โดยสร้างขึ้นมาเพื่อรองรับรูปแบบเซสชั่นที่หลากหลายตามแต่โปรโตคอล เพื่อให้ทำงานร่วมกับทาสก์และเซิร์ฟเวอร์แคมอนได้อย่างถูกต้อง



รูปภาพ 4-5 ตัวอย่างการอิมพลิเมนต์ซ็อคเก็ต เพื่อทำงานร่วมกับเซิร์ฟเวอร์แคมอนในแต่ละโปโตคอล

4.2 นักพัฒนาบริการบนระบบฝังตัว



รูปภาพ 4-6 การสร้างบริการโดยการอิมพลิเมนต์เซอร์วิสออบเจ็กต์

ในการสร้างบริการหรือเซอร์โค้ด เข้ามาใช้ในระบบนั้น นักพัฒนาจะต้องทำการอิมพลิเมนต์คลาสของเซอร์วิสออบเจ็คต์ ที่มีอยู่ในระบบ เพื่อใช้อินเตอร์เฟส (Interface) ที่จะเข้าไปทำงานร่วมกับระบบได้อย่างถูกต้อง โดยส่วนที่ต้องอิมพลิเมนต์เพิ่มนั้นมีดังนี้

- **ฟังก์ชั่น init()**

เป็นฟังก์ชั่นที่ทำหน้าที่ในการแปลงข้อมูลจากพารามิเตอร์ที่ส่งเข้ามา ในรูปของสตริง (String) ไปเป็นข้อมูลในชนิดที่ระบบต้องการใช้ในการประมวลผล เพื่อเตรียมความพร้อมก่อนที่จะทำการประมวลผลต่อไป

- **ฟังก์ชั่น run()**

เป็นฟังก์ชั่นหลัก ที่เป็นส่วนประมวลผลของบริการหรือเซอร์วิสนั้นๆ ในลักษณะของการทำงานแบบฟังก์ชั่นทั่วๆไป ที่ง่ายต่อการสร้างและพัฒนา โดยไม่ต้องสนใจส่วนอื่นๆ ในระบบ เพียงแต่เมื่อทำการประมวลผลฟังก์ชั่นเสร็จแล้ว ก็ให้แปลงข้อมูลจากชนิดที่เป็นอยู่ ไปเป็นสตริง แล้วนำไปเก็บไว้ที่อาร์พีซีรีซัลต์ เพื่อรอการนำไปจัดส่งผลลัพธ์กลับไปยังผู้ร้องขอตามโปรโตคอลต่างๆ ต่อไป

```
#ifndef CALCULATOR
#define CALCULATOR
#include "servobj.h"
class Calculator : public ServiceObj{ //extend and implement ServiceObject
public:
        Calculator(const char* serv_name):ServiceObj(serv_name){};
        ~Calculator(){};
        void init(RPCParam *p_param,RPCParam *p_result){
            //converse type from string to expecting type
        }
        void run(){
            // service process code
            // converse type to string
            // save in RPC Result
        }
};
#endif
```

รูปภาพ 4-7 ตัวอย่างการโค้ดในการสร้างเซอร์วิสในระบบ

หลังจากนักพัฒนาได้ทำการพัฒนาเซอร์วิสโค้ดเสร็จแล้ว ก่อนใช้งานนั้น ต้องมีการบันทึกรายละเอียดต่างๆที่เกี่ยวข้องกับเซอร์วิสที่ได้สร้างขึ้นมานั้น ได้แก่

- ชื่อเรียกของเซอร์วิส
- ชื่อเต็มของเซอร์วิส
- ข้อความอธิบายเซอร์วิสแบบย่อ
- สถานะในการเปิดใช้งานเมื่อระบบเริ่มต้น

ข้อมูลดังกล่าว จะถูกเก็บไว้ในไฟล์เซอร์วิสรีจิสทรี้ (Service Registry) ในรูปแบบของภาษาเอ็กซ์เอ็มแอล(XML Format) ไว้สำหรับตั้งค่าต่างๆให้กับเซอร์วิสที่ลงทะเบียนเพื่อเป็นข้อมูลในการเรียกใช้เซอร์วิสต่อไป นอกเหนือจากนี้ นักพัฒนาก็สามารถเพิ่มฟิลด์ข้อมูลอื่นๆ ที่เป็นลักษณะเฉพาะหรือมีความจำเป็นต้องใช้ในการทำงานของเซอร์วิสได้อีกด้วย

```
<Service type="registry">
<cal type="key">
<Name type="string">Calculator</Name>
<Description type="string">Sample service as calculator</Description>
<Status type="string">on</Status>
</cal>
<address type="key">
<Name type="string">Address Lists</Name>
<Description type="string">Sample service as address lists</Description>
<Status type="string">off</Status>
</address>
</Service>
```

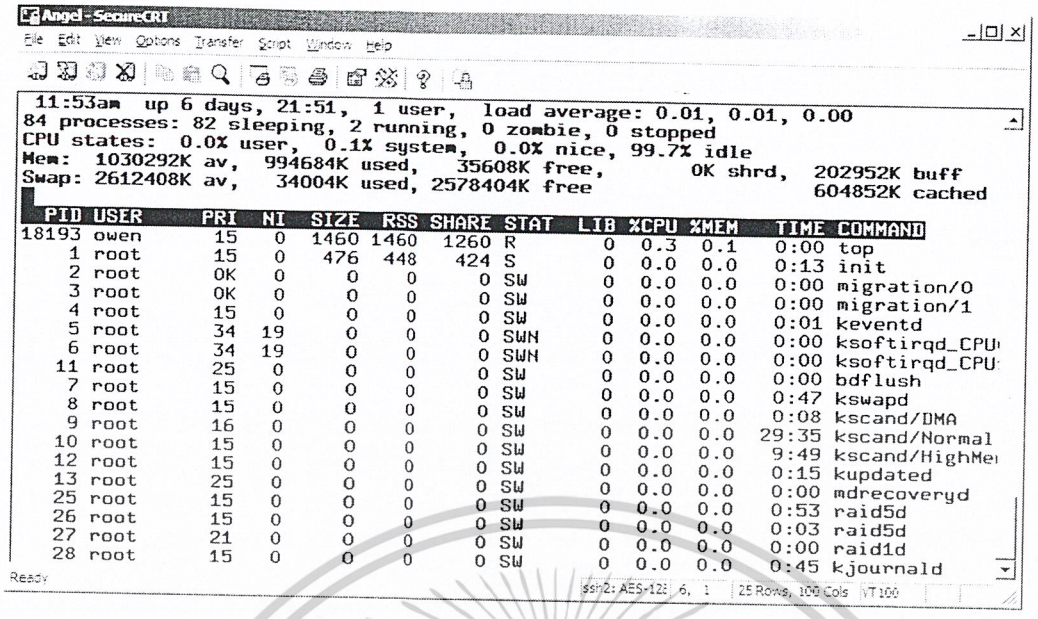
รูปภาพ 4-8 ตัวอย่างเซอร์วิสรีจิสทรี้ไฟล์ที่เก็บในรูปแบบเอกสารเอ็กซ์เอ็มแอล

4.3 ผู้ที่เรียกใช้บริการของระบบ

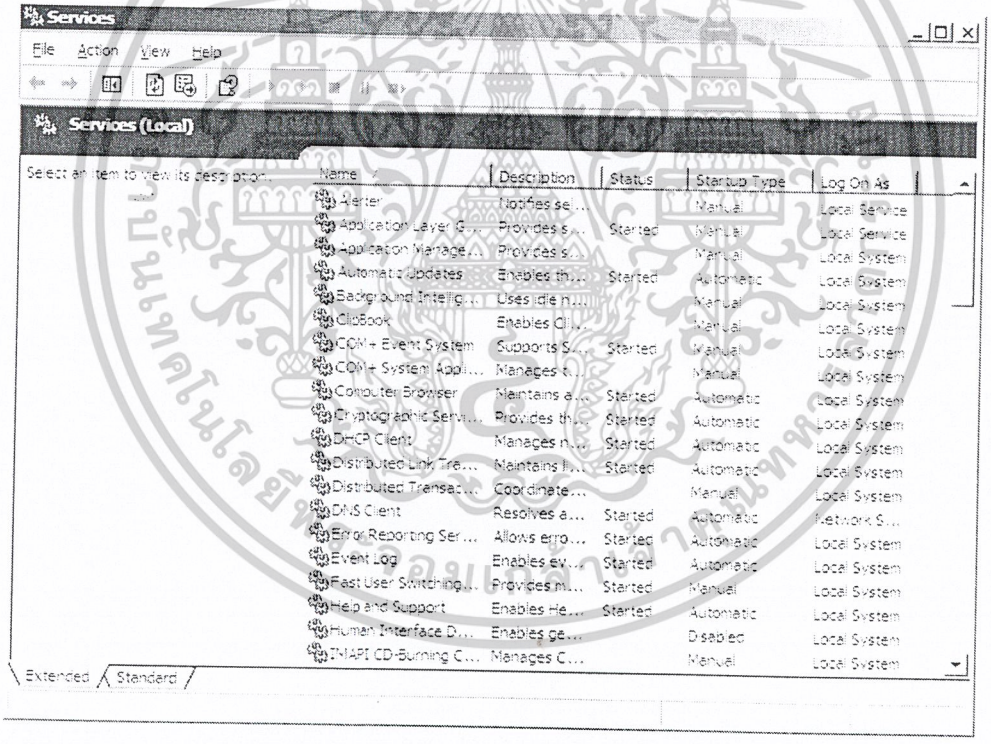
ในลักษณะการใช้งานนี้ ผู้ที่เรียกใช้บริการเซอร์วิสจากระบบฝั่งตัวจะต้องทำการร้องขอรายละเอียดของเซอร์วิสก่อนเพื่อให้ทราบถึงข้อมูลเกี่ยวกับอินเตอร์เฟซในการติดต่อกับเซอร์วิสนั้น จากนั้นสามารถที่จะทำการสร้างแอพพลิเคชั่นจากไลบรารี่ต่าง ๆ ขึ้นมาได้ตามสภาพแวดล้อมการใช้งานของผู้ใช้ได้ เช่น อาปาเช่โซบ คอทเน็ตเฟรมเวิร์ก เป็นต้น โดยผู้ใช้งานสามารถนำข้อมูลที่ได้รับกลับมาสร้างเป็นรูปแบบต่างๆ ได้ตามต้องการ อีกทั้งยังสามารถที่จะสร้างอินเตอร์เฟซได้เอง

4.4 ผู้ดูแลระบบการให้บริการของระบบ

ภายในระบบ มีส่วนที่ทำหน้าที่ในการจัดการและควบคุมเซอร์วิสในระบบ โดยผู้ดูแลระบบสามารถพัฒนาแอพพลิเคชั่นที่ใช้ติดต่อ กับเซอร์วิสเมเนเจอร์ ในลักษณะของการเรียกใช้เซอร์วิสแบบหนึ่ง ซึ่งแอพพลิเคชั่นนั้น สามารถดัดแปลงรูปแบบได้หลากหลาย อาทิเช่น การเรียกใช้เซอร์วิสเมเนเจอร์ผ่านเทลเนต (Telnet) ซึ่งสามารถให้แสดงผลในลักษณะแบบของโปรแกรมท๊อป (Top) ที่มีอยู่ในระบบปฏิบัติการลีนุกต์ (Linux) หรือยูนิกซ์ (Unix) หรืออาจเป็นแอพพลิเคชั่นที่พัฒนาบนแพลต์ฟอร์มของระบบปฏิบัติการในตระกูลวินโดวส์ (Windows OS Platform) ที่มีจุดเด่นในด้านของการแสดงผลแบบกราฟิคยูสเซอร์อินเตอร์เฟส (Graphic User Interface)



รูปภาพ 4-9 ตัวอย่างรูปแบบในการควบคุมเซอร์วิสในระบบของโปรแกรม TOP



รูปภาพ 4-10 ตัวอย่างรูปแบบในการควบคุมเซอร์วิสของเซอร์วิสเมเนเจอร์ในวินโดวส์

บทที่ 5

การทดสอบและผลการทดสอบ

เนื้อหาในบทนี้เป็นการทดสอบเพื่อประเมินผลโครงงาน ว่าสามารถบรรลุวัตถุประสงค์ของโครงงานที่ได้ตั้งไว้ได้หรือไม่ โดยได้มีการตั้งวัตถุประสงค์ของการทดลองดังนี้

1. เพื่อให้สามารถพัฒนาบริการบนระบบที่ได้สร้างขึ้นได้
2. เพื่อทดสอบการให้บริการของเว็บเซอร์วิสบนระบบฝังตัว
3. เพื่อทดสอบการทำงานของระบบแบบมัลติทาสก์กิ้งและเรียลไทม์

ในการทดสอบโครงงานนี้ จะแบ่งการทดสอบออกเป็นสองส่วนคือ การทดสอบการทำงานของระบบและการทดสอบการใช้งานของระบบ

5.1 การทดสอบการทำงานของระบบ

เป็นการทดสอบเพื่อแสดงให้เห็นว่าระบบนี้สามารถทำงานได้ตามที่ได้ออกแบบไว้หรือไม่ โดยแยกหัวข้อการทดสอบและผลการทดสอบดังตารามด้านล่าง

การทดสอบ	ผลการทดสอบ
การทำงานแบบมัลติทาสก์กิ้ง	ทาสก์ในระบบสามารถทำงานร่วมกันแบบมัลติทาสก์ได้
การสื่อสารข้อมูลผ่านระบบเว็บเซอร์วิส	สามารถติดต่อสื่อสารในระบบเว็บเซอร์วิสผ่านเครือข่ายอินเตอร์เน็ตได้อย่างถูกต้อง

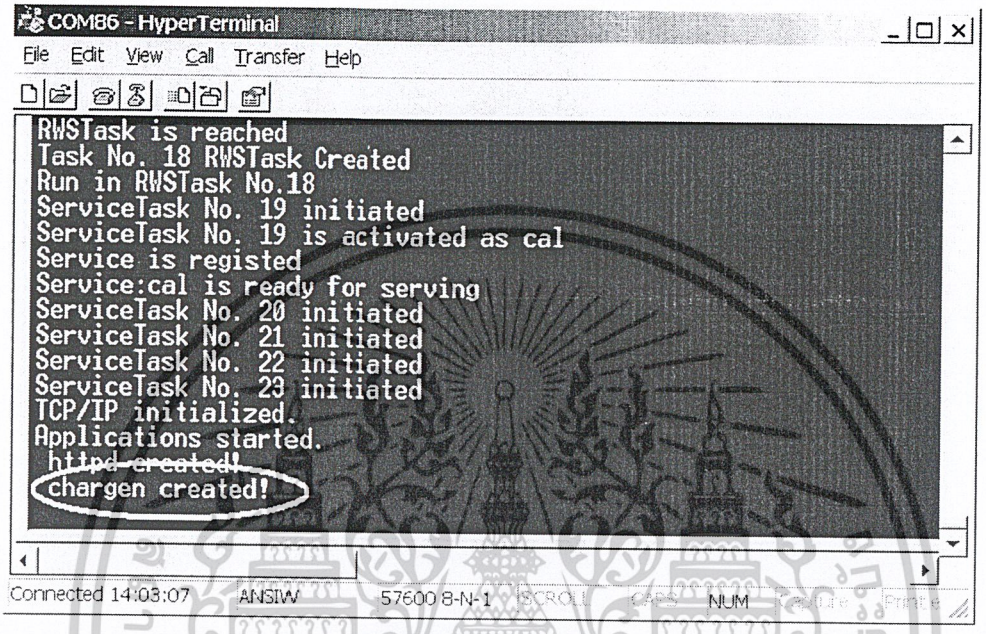
ตาราง 5-1 ผลการทดสอบการทำงานของระบบ

รายละเอียดของการทดสอบการทำงานในแต่ละหัวข้อจะได้อธิบายดังต่อไปนี้

5.1.1 ทดสอบการทำแบบมัลติทาสก์กิ้ง

การทดสอบการทำงานแบบมัลติทาสก์กิ้ง เพื่อทดสอบว่าระบบสามารถรองรับการทำงานของทาสก์บนระบบพร้อมๆกัน และมีความเป็นอิสระต่อกัน แต่เนื่องจากการทำงานแบบมัลติทาสก์กิ้งนั้นเป็นกลไกการทำงานภายใน ไม่สามารถทดสอบการทำงานโดยตรงได้ จำเป็นที่จะต้องแสดงการทำงานผ่านหน้าจอ หรือ ทดสอบการติดต่อผ่านระบบเน็ตเวิร์คที่สามารถรองรับการเชื่อมต่อได้มากกว่า 1 การเชื่อมต่อพร้อมๆกัน

ตัวอย่างที่ใช้ในการทดสอบการทำงานแบบมัลติทาสก์กิ้ง เป็นการสร้างบริการผ่านระบบเน็ตเวิร์ค ด้วยโปรโตคอลชาร์เจน (Chargen) บนพอร์ตการสื่อสารหมายเลข 19 โดยสร้างโปรแกรมที่ทำงานเป็นเซิร์ฟเวอร์ที่รองรับการเชื่อมต่อ 5 การเชื่อมต่อพร้อมกัน โดยหลังจากสั่งให้โปรแกรมทำงาน จะมีลักษณะดังรูปด้านล่าง

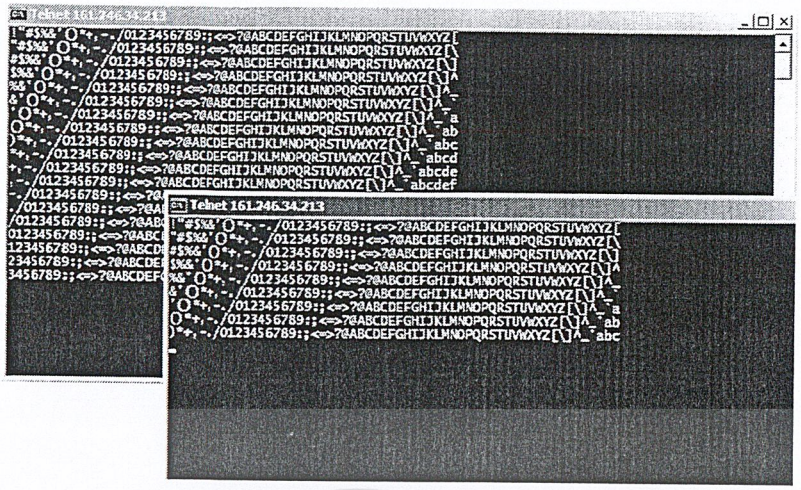


รูปภาพ 5-1 การเขียนโปรแกรมทดสอบการทำงานแบบมัลติทาสก์กิ้ง

หลังจากสั่งการให้โปรแกรมที่ทดสอบทำงาน ก็ทำการขอใช้บริการโดยทำการเชื่อมต่อด้วยโปรแกรมเทลเน็ต (telnet) ที่มีอยู่บนระบบปฏิบัติการวินโดวส์ (Windows) โดยใช้คำสั่งซึ่งมีรูปแบบดังนี้

```
telnet <ip-address> <port number>
```

โดยให้ ip-address เป็นหมายเลขไอพีของอุปกรณ์คอม86 และหมายเลขพอร์ตเป็นหมายเลข 19 โดยเปิดโปรแกรมเทลเน็ตจำนวน 2 โปรแกรม แล้วป้อนคำสั่งดังกล่าว แล้วให้ทำงานพร้อมๆกันเพื่อทดสอบผลการทำงานแบบมัลติทาสก์กิ้ง โดยผลการทดลองเป็นไปดังรูปต่อไปนี้

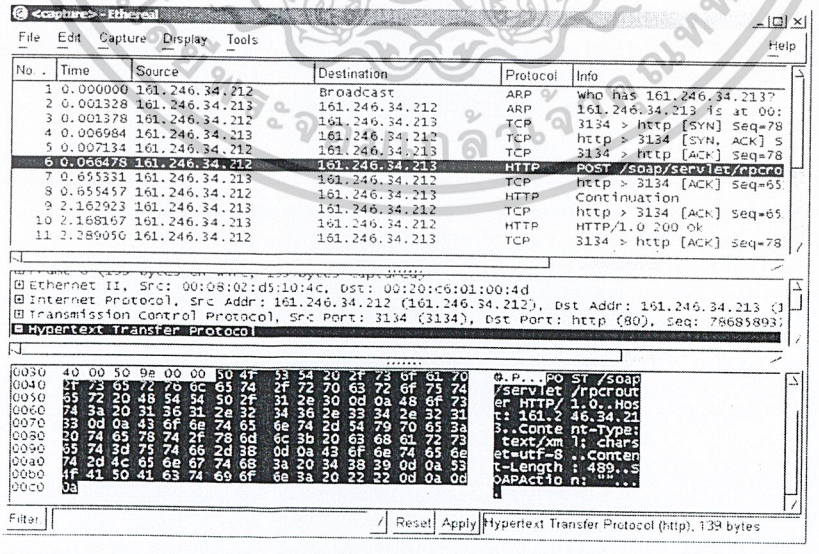


รูปภาพ 5-2 แสดงการติดต่อขอใช้บริการโปรแกรมที่ทำงานแบบมัลติทาสก์กิ้งบนอุปกรณ์คอม86

ผลการทดลอง ทำให้รู้ได้ว่า สามารถสร้างโปรแกรมที่ทำงานแบบมัลติทาสก์กิ้งบนอุปกรณ์คอม 86 ได้โดยแสดงในรูปของการรองรับการเชื่อมต่อขอใช้บริการผ่านระบบเน็ตเวิร์ค 2 การเชื่อมต่อพร้อมกัน

5.1.2 ทดสอบการสื่อสารข้อมูลผ่านระบบเว็บเซอร์วิส

การทดสอบการสื่อสารข้อมูลผ่านระบบเว็บเซอร์วิส ทำได้โดยทำการติดตามและดักจับข้อมูลที่ส่งผ่านเครือข่ายที่ถูกส่งออกมาขณะที่ระบบเว็บเซอร์วิสมีการติดต่อขอใช้บริการ โดยการติดต่อข้อมูลในระบบเว็บเซอร์วิสนั้น อยู่บนโปรโตคอลเอชทีทีพี ที่อยู่บนโปรโตคอลทีซีพี/ไอพี โดยอยู่ใช้หมายเลขพอร์ต 80 โดยผลการดักจับข้อมูลในการติดต่อสื่อสารข้อมูลขณะใช้งาน โดยใช้โปรแกรมอีเธอร์เรียล (Ethereal) มีดังรูปต่อไปนี้



รูปภาพ 5-3 ผลการตรวจสอบการรับส่งข้อมูลในระบบเว็บเซอร์วิสบนคอม86 โดยโปรแกรมอีเธอร์เรียล

5.2 การทดสอบการใช้งานของระบบ

เป็นการทดสอบผลการใช้งาน ว่าสามารถทำงานได้ตามที่ออกแบบไว้หรือไม่ โดยแยกหัวข้อการทดสอบและผลการทดสอบดังตารางด้านล่าง

การทดสอบ	ผลการทดสอบ
การพัฒนาบริการบนระบบเว็บเซอร์วิส	สามารถทำการพัฒนาบริการบนระบบเว็บเซอร์วิสได้ตามแนวทางและข้อแนะนำที่กำหนดไว้
การใช้บริการบนระบบเว็บเซอร์วิส	สามารถใช้บริการผ่านโปรโตคอลโซบในระบบเว็บเซอร์วิสได้

ตาราง 5-2 ผลการทดสอบการใช้งานของระบบ

5.2.1 ทดสอบการพัฒนาบริการบนระบบเว็บเซอร์วิส

การทดสอบการพัฒนาบริการบนระบบเว็บเซอร์วิสที่ได้พัฒนาขึ้น ทำได้โดยทดลองพัฒนาบริการขึ้นเพื่อทดลองใช้งานจริง เพื่อทดสอบแนวทางและรูปแบบที่ได้ออกแบบ ในการพัฒนาบริการในลักษณะของเซอร์วิสโค้ด ในการทดลองนั้น ขอยกตัวอย่างการพัฒนาบริการ Calculator ที่ทำการประมวลผลจำนวนจริง ผ่านระบบเว็บเซอร์วิสบนระบบอุปกรณ์ฝังตัว

ในการเขียนโปรแกรมในส่วนของบริการนั้นมีขั้นตอนดังนี้

- เขียนโปรแกรมในภาษาซีพลัสพลัส แล้วพัฒนาเซอร์วิสโค้ดโดยการทำการ inherit โค้ดแม่แบบหรือซุปเปอร์คลาส (Super Class) ที่มีอยู่แล้วในระบบ แล้วทำการพัฒนาเพิ่มในส่วนที่จำเป็นต่อการทำงานของเซอร์วิสโค้ดที่จะพัฒนานั้น ซึ่งได้แก่ ฟังก์ชั่น init และฟังก์ชั่น run

```
void Calculator::init(RPCParam *p_param,RPCParam *p_result){
        param=p_param;
        result=p_result;

        SERVICE_METHOD=param->valueOf((char *)ServiceConsts::SERVICE_METHOD);
        printf("SERVICE_METHOD = %s\n",SERVICE_METHOD);
        ARG1=atof(param->valueOf(ARGUMENT_NAME_1));
        ARG2=atof(param->valueOf(ARGUMENT_NAME_2));
        }
```

รูปภาพ 5-4 การเขียนเซอร์วิสโค้ดในส่วนของฟังก์ชั่น init

```
void Calculator::run(){
        if (strcmp(SERVICE_METHOD,PLUS)==0)          {
                ARG1+=ARG2;
        }
        else if (strcmp(SERVICE_METHOD,MINUS)==0)   {
                ARG1-=ARG2;
        }
        else if (strcmp(SERVICE_METHOD,TIMES)==0)   {
                ARG1*=ARG2;
        }
        else if (strcmp(SERVICE_METHOD,DIVIDE)==0)  {
                ARG1/=ARG2;
        }
        else
                ARG1=0.0;
        char result_st[20];
        sprintf(result_st,"%f",ARG1);
        result->addParam(RETURN_NAME,RETURN_TYPE,result_st);
        }
```

รูปภาพ 5-5 การเขียนเซอร์วิสโค้ดในส่วนของฟังก์ชั่น run

จากการทดลองเขียนโค้ดในส่วนดังกล่าว สามารถทำได้ง่ายโดยสามารถมองในลักษณะของการเขียนโปรแกรมเป็นฟังก์ชั่น ที่มีการเซ็ตอัพพารามิเตอร์ให้กับฟังก์ชั่นด้วยฟังก์ชั่น init ก่อน แล้วจากนั้นจึงสั่งให้ฟังก์ชั่นทำการประมวลผลโดยฟังก์ชั่น run เพื่อได้ผลลัพธ์ออกมา โดยไม่ต้องสนใจในส่วนที่ทำหน้าที่ติดต่อสื่อสารบนระบบเว็บเซอร์วิส ซึ่งซ้อนอยู่เบื้องหลัง

- เพิ่มเติมข้อมูลของการให้บริการบนระบบเว็บเซอร์วิส ลงในไฟล์ที่เก็บค่าข้อมูลดังกล่าว ที่มีชื่อว่า boot.xml ซึ่งจะหน้าที่ในการสร้างบริการขึ้นมาในระบบเมื่อระบบมีการเปิดใช้งานขึ้น โดยไฟล์ดังกล่าว เก็บอยู่ในรูปแบบของเอ็กซ์เอ็มแอล ซึ่งมีรูปแบบที่ง่ายต่อการเพิ่มเติมข้อมูลดังกล่าว โดยได้ทำการกำหนดรูปแบบที่ใช้ในการเพิ่มเติมข้อมูลไว้แล้ว ซึ่งตัวอย่างการเพิ่มข้อมูลบริการ เป็นดังรูปต่อไปนี้

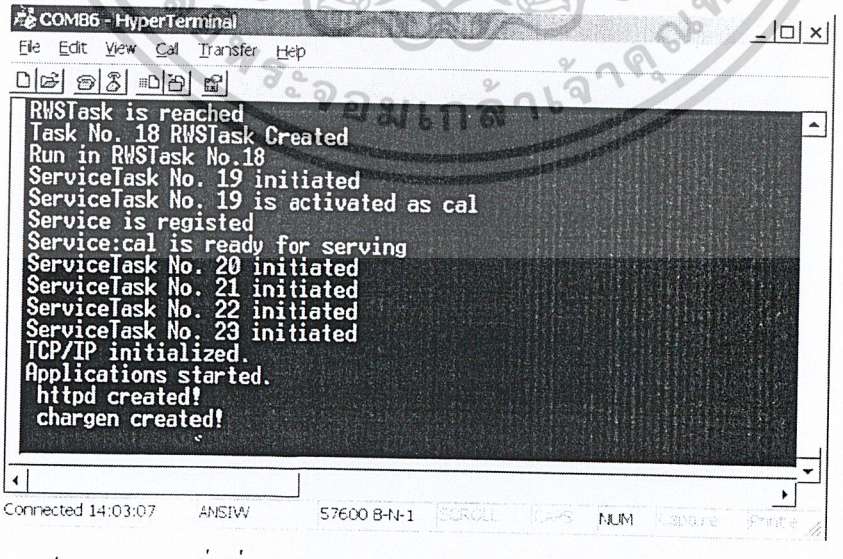
```
<cal type="key">
<Name type="string">Calculator</Name>
<Description type="string">Sample service as calculator</Description>
<Status type="string">on</Status>
<ServiceArg1 type="xsd:double">arg1</ServiceArgs1>
<ServiceArg2 type="xsd:double">arg2</ServiceArgs2>
<ServiceReturn type="xsd:double">return</ServiceReturn>
</cal>
```

รูปภาพ 5-6 แสดงการเพิ่มข้อมูลของบริการบนระบบ

จากการทดลองเพิ่มข้อมูลของบริการที่ได้ทำการพัฒนาขึ้นในระบบนั้น สามารถทำได้ง่าย เพราะไฟล์อยู่ในรูปแบบของเอ็กซ์เอ็มแอล ซึ่งเป็นรูปแบบเอกสารที่มีลักษณะเฉพาะและเป็นมาตรฐาน หากมีตัวอย่างโครงสร้างให้แล้ว ก็สามารถทำการเพิ่มข้อมูลได้อย่างง่ายดาย

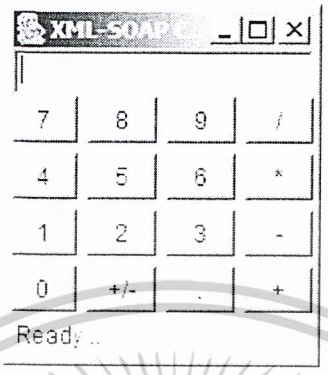
5.2.2 ทดสอบการใช้งานบริการที่พัฒนาขึ้นบนระบบ

หลังจากทำการทดลองพัฒนาบริการบนระบบแล้ว จึงสามารถทำการทดสอบการใช้งานบริการดังกล่าวบนระบบจริง โดยทำการสร้างโปรแกรมในฝั่งผู้ใช้ โดยใช้ภาษาจาวา (Java) หลังจากพัฒนาแล้วจึงทำการคอมไพล์โปรแกรมระบบเว็บเซอร์วิสบนระบบฝังตัว แล้วนำไฟล์ที่ได้ลงไปไว้บนอุปกรณ์คอม 86 แล้วทำการเริ่มการทำงานของระบบโดยการสั่งการผ่านคอนโซลซีเรียลพอร์ต



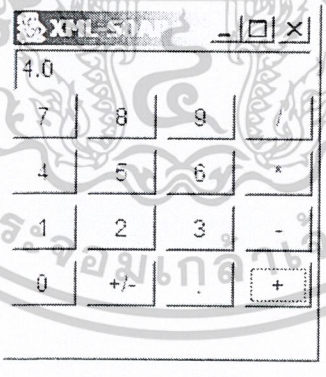
รูปภาพ 5-7 การสั่งเริ่มการทำงานของระบบเว็บเซอร์วิสบนอุปกรณ์คอม86

หลังจากให้ระบบเริ่มทำงานแล้ว ให้ทำการเรียกโปรแกรมที่ขอเรียกใช้บริการดังกล่าว ในตัวอย่างนี้ เป็นโปรแกรมที่เรียกใช้บริการ Calculator ที่ได้ทำพัฒนาในฝั่งของผู้ให้บริการจากการทดลองที่ผ่านมา โดยหลังจากเรียกใช้โปรแกรมแล้ว จะเป็นดังรูปต่อไปนี้



รูปภาพ 5-8 โปรแกรมขอเรียกใช้บริการ Calculator

แล้วทำการคลิ๊กที่ปุ่มตัวเลขที่ต้องการคำนวณ คล้ายเครื่องคิดเลข ตัวอย่าง เช่น 1 + 3 ทำได้โดยกดหมายเลข '1' ตามด้วย '+' แล้วตามด้วย '3' แล้ว '+' อีกครั้ง จากนั้น โปรแกรมจะทำการส่งข้อมูลเหล่านี้ไปขอใช้บริการ Calculator บนระบบเว็บเซอร์วิสบนอุปกรณ์คอม86 โดยหลังจากผ่านการประมวลผลแล้ว ผลที่ได้รับกลับมาจะแสดงเป็นผลลัพธ์ดังรูปต่อไปนี้



รูปภาพ 5-9 ผลลัพธ์จากการคำนวณโดยการขอใช้บริการ Calculator ผ่านเว็บเซอร์วิส

บทที่ 6
บทวิจารณ์และสรุปผล

6.1 สรุปและวิจารณ์ผลการทำงาน

ในส่วนของการสรุปและวิจารณ์ผลการทำงานสามารถแบ่งออกได้เป็นหัวข้อต่างๆ ดังนี้

6.1.1 ผลงานโดยรวมของการพัฒนาโครงงาน

สามารถออกแบบสถาปัตยกรรมของระบบเว็บเซอร์วิสบนระบบฝังตัว ที่พัฒนาจากโครงงานรุ่นที่ผ่านมา โดยออกแบบให้แต่ละส่วนของระบบ สามารถทำงานได้อย่างอิสระต่อกันในลักษณะของมัลติทาสก์กิ้ง เพื่อเพิ่มประสิทธิภาพให้กับระบบเว็บเซอร์วิสบนระบบฝังตัว อีกทั้งยังสามารถออกแบบสถาปัตยกรรมโครงสร้างและกลไกการทำงานของระบบ ที่สามารถรองรับการพัฒนาส่วนต่างๆ ให้รองรับมาตรฐานการสื่อสารข้อมูลในมาตรฐานอื่นๆ เพื่อขยายความสามารถในการติดต่อสื่อสารได้หลากหลายวิธี สุดท้ายยังได้วางแนวทางในการพัฒนาส่วนประกอบต่างๆ ในระบบเพื่อเป็นแนวทางที่นักพัฒนาท่านอื่นๆ จะสามารถพัฒนาบริการและความสามารถในการติดต่อบนระบบนี้ได้ต่อไป

6.1.2 การศึกษาและทดลอง

ในการศึกษานั้น เริ่มจากการศึกษาระบบที่เป็นผลงานของนักศึกษารุ่นที่ผ่านมา ซึ่งต้องอาศัยความสามารถในการติดตามเพื่อเรียนรู้การทำงานของระบบ รวมถึงรับทราบและแก้ไขข้อผิดพลาดที่ได้พบ ซึ่งการทำงานในส่วนนี้ ต้องใช้ความพยายามและกลวิธีในการติดตามการทำงานของระบบ ซึ่งช่วยฝึกฝนความสามารถในการพัฒนาระบบขึ้นอย่างมาก อีกทั้งยังมีการศึกษาซอฟต์แวร์ไลบรารี่ของระบบปฏิบัติการแบบเรียลไทม์ ซึ่งเป็นรูปแบบในการพัฒนาแอพพลิเคชั่นอีกแนวทางหนึ่ง ซึ่งช่วยเพิ่มเติมความรู้และแนวทางในการพัฒนาแอพพลิเคชั่นแบบเรียลไทม์และมัลติทาสก์

6.1.3 การออกแบบในส่วนของซอฟต์แวร์ของโครงงาน

หลังจากมีการคิดค้นกลไกในการทำงานของระบบแล้ว ก็ได้ทำการออกแบบโครงสร้างของซอฟต์แวร์โดยใช้โปรแกรม Rational Rose ซึ่งเป็นโปรแกรมที่ช่วยในการออกซอฟต์แวร์ด้วยแนวความคิดแบบออบเจ็คต์ (Object-Oriented Design) โดยได้ทำการสร้างไดอะแกรมต่างๆ ที่ช่วยสร้างมุมมองและขยายความเข้าใจที่มีต่อระบบ อีกทั้งยังเป็นรายละเอียดในการสร้างระบบเพื่อให้ผู้ที่สนใจนำมาศึกษาและพัฒนาต่อได้ โดยโค้ดของระบบเขียนโดยภาษาซี / ซีพลัสพลัส

6.1.4 เทคโนโลยีและแพลตฟอร์มที่เลือกมาใช้ในโครงงาน

- **แอลดับบิวไอพี (Lwip)**

เป็นซอฟต์แวร์ไลบรารี่ที่ใช้ในการพัฒนาโปรแกรมที่ต้องทำการติดต่อสื่อสารผ่านเครือข่ายอินเตอร์เน็ต โดยมีการจัดการตั้งแต่ในชั้นของไอพี (IP) รวมไปถึงทีซีพี(TCP) และยูดีพี(UDP) โดยมีอินเตอร์เฟสที่เป็นไปตามมาตรฐานบีเอสดีซ็อคเก็ต (BSD Socket) ซึ่งเป็นรูปแบบมาตรฐานในการพัฒนาโปรแกรมบนเน็ตเวิร์ค (Network Programming)

- **ไมโครซี / โอเอส ทู (Micro C/OSii)**

เป็นซอฟต์แวร์ไลบรารีที่ทำหน้าที่เป็นระบบปฏิบัติการแบบเรียลไทม์ ที่พัฒนาขึ้นด้วยภาษาซี โดยมีการพัฒนาส่วนที่เพิ่มความสามารถในพัฒนาระบบซึ่งเพียงพอต่อการใช้งานและพัฒนาระบบบนระบบปฏิบัติการแบบเรียลไทม์ เช่น แมสเซจคิว และ เซมาฟอร์ เป็นต้น ที่สำคัญยังเป็นซอฟต์แวร์ไลบรารีแบบโอเพ่นซอร์สอีกด้วย

6.2 การประเมินผลโครงงาน

- **การทำงานแบบมัลติทาสก์และเรียลไทม์**

การทำงานแบบมัลติทาสก์กิ้งและเรียลไทม์นั้น เนื่องจากแต่ละทาสก์ ต้องทำการรอผลลัพธ์จากการทำงานของอีกทาสก์หนึ่ง อาจเกิดเป็นปัญหาคอขวดของระบบได้ หากทาสก์ที่สร้างผลลัพธ์ได้รับการร้องขอเป็นจากทาสก์อื่นๆหลายทาสก์พร้อมๆกัน หรือมีกระบวนการประมวลที่ซับซ้อนและใช้เวลานาน ซึ่งทำให้ต้องเข้าคิวเพื่อรอการขอรับบริการ ทำให้การประมวลผลไม่เกิดประสิทธิภาพของการทำงานแบบมัลติทาสก์อย่างเต็มที่

- **โปรโตคอลแฮนเดิ้ลเลอร์**

ในการพัฒนาในส่วนของโปรโตคอลแฮนเดิ้ลเลอร์ หรือส่วนที่จัดการติดต่อโดยโปรโตคอลรูปแบบต่างๆนั้น ได้มีการพัฒนาเฉพาะในส่วนโซบโปรโตคอลเท่านั้น แต่ยังไม่ได้ทำการทดลองนำโปรโตคอลอื่นๆมาใช้จริง ซึ่งในทางทฤษฎีแล้วอาจทำได้ แต่ในความเป็นจริงอาจมีข้อปลีกย่อยที่ระบบไม่สามารถครอบคลุมได้ทุกๆโปรโตคอล จึงอาจเป็นแนวทางในการนำไปวิจัยและสถาปัตยกรรมที่แก้ไขปัญหาจุดนี้ต่อไป

- **การใช้ทรัพยกรในระบบ**

ในการพัฒนาโครงงานที่ผ่านมา ได้มุ่งเป้าที่การทดสอบทฤษฎีและสถาปัตยกรรมที่ได้ออกแบบขึ้น ดังนั้น จึงเป็นไปได้ที่ระบบที่พัฒนาขึ้น จะใช้ทรัพยากรหน่วยความจำอย่างมากเกินความจำเป็น ซึ่งหากมีการนำพัฒนาต่อ ก็ควรที่จะสนใจในปัญหาด้านนี้ด้วย

6.3 สรุปสิ่งที่ได้เรียนรู้ในการพัฒนาโครงงาน

1. สถาปัตยกรรมเว็บเซอร์วิสและการทำงาน
2. มาตรฐานเอ็กซ์เอ็มแอล และการนำไปใช้
3. การพัฒนาโปรแกรมบนระบบปฏิบัติการเรียลไทม์มัลติทาสก์
4. เน็ตเวิร์คโปรแกรมมิ่ง
5. แนวคิดในการพัฒนาแอพพลิเคชั่นบนระบบฝังตัว
6. ทักษะในการพัฒนาแอพพลิเคชั่นด้วยภาษาซี/ซีพลัสพลัส
7. วิธีคิดในการพัฒนาแอพพลิเคชั่นในมุมมองที่กว้างขึ้น

ภาคผนวก

การปรับแต่งคอม86 (Com86 Configuration)

1. สิ่งที่ให้มาพร้อมกับชุดพัฒนา

ในชุดพัฒนาคอม 86จะประกอบด้วยอุปกรณ์ต่างๆดังนี้

- บอร์ดคอม 86 1 บอร์ด
- อแด๊ปเตอร์สำหรับบอร์ดคอม 86 1 อัน
- สาย Serial ชนิด null modem 1 เส้น
- สาย USB ชนิด A-B 1 เส้น
- ซีดีรอมโปรแกรมและคู่มือการใช้งาน 1 แผ่น

2. ความต้องการของระบบเบื้องต้น

ในการใช้งานชุดพัฒนาคอม 86นั้นจะมีความต้องการเบื้องต้นทั้งทางด้านฮาร์ดแวร์และซอฟต์แวร์ดังนี้

- เครื่องคอมพิวเตอร์สำหรับใช้คอมไพล์และตรวจสอบการทำงานของโปรแกรม 1 เครื่อง โดยจะต้องมีพอร์ตอนุกรมสำหรับใช้งานได้อย่างน้อย 1 พอร์ต *และมีการ์ดอีเธอร์เน็ตอย่างน้อย 1 การ์ด** หากต้องการพัฒนาโปแกรมที่ใช้คุณสมบัติทางด้าน Network
- โปรแกรมเทอร์มินัล เช่น Hyperterminal เป็นต้น
- สายแลนชนิดแบบสายตรง)straight-through twisted pair cable (และ 10Base-T hub หรือสายแลนชนิดแบบครอส)twisted pair cross cable**(
- สาย serial***
- คอมไพล์เลอร์ที่สามารถคอมไพล์โปรแกรมสำหรับไมโครโพรเซสเซอร์ตระกูล 186 แบบ Real mode เช่น turbo C, Turbo assembly หรืออื่นๆ ตามที่ผู้พัฒนาต้องการใช้งาน

* สำหรับหรับใช้ เป็น Serial console ในการพัฒนาโปรแกรมต่างๆ

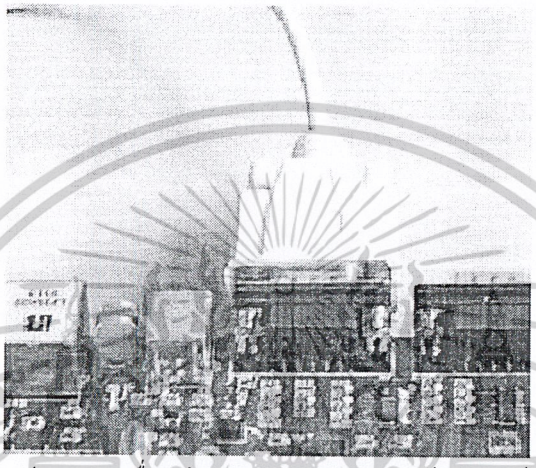
** เพิ่มเติมสำหรับใช้ในการพัฒนาที่ต้องการใช้ความสามารถทางด้าน Network

*** เพิ่มเติมสำหรับผู้ที่ต้ องการใช้งานพอร์ตอนุกรมที่เหลืออีกพอร์ตหนึ่งของบอร์ดคอม86

3. การติดตั้งใช้งาน

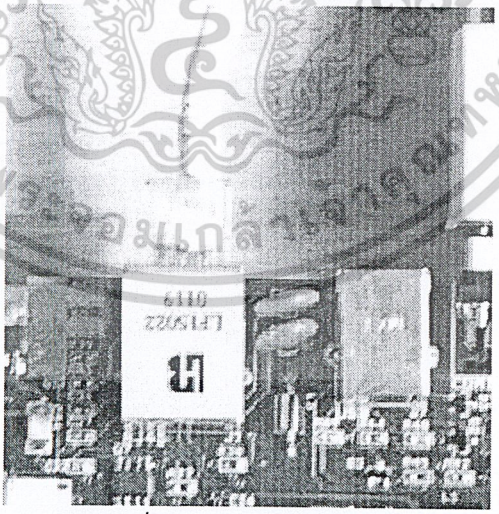
ในการติดตั้งใช้ชุดพัฒนาคอม 86ให้ปฏิบัติตามขั้นตอนต่างๆ ดังนี้

1. นำบอร์ดคอม 86ออกมาจากซองที่ใช้บรรจุเพื่อการขนส่งและตรวจสอบสภาพของบอร์ดในเบื้องต้นว่าเกิดการชำรุดเสียหายจากการขนส่งหรือไม่
2. เชื่อมต่อสายของ Serial cable ที่ให้มากับพอร์ตอนุกรมของคอมพิวเตอร์ที่ต้องการใช้งานและเชื่อมต่อสายอีกด้านเข้ากับบอร์ดคอม86



รูปภาพการเชื่อมต่อสาย serial console เข้ากับบอร์ดคอม86

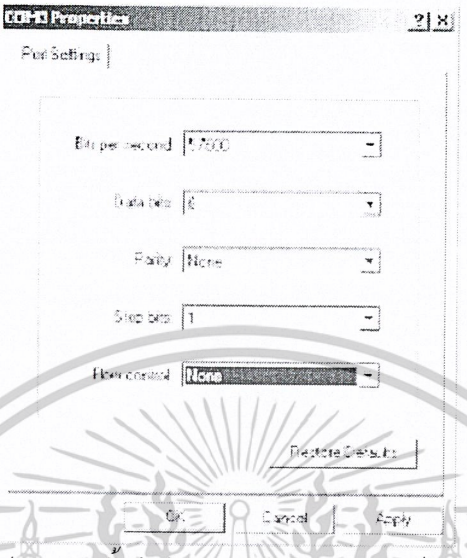
3. เชื่อมต่อสาย Ethernet เข้าบอร์ดคอม86 และเครื่องคอมพิวเตอร์หรือเน็ตเวิร์กฮับ*



รูปภาพการเชื่อมต่อสายอีเธอร์เน็ตเข้ากับบอร์ดคอม86

* สำหรับผู้ที่ต้องการใช้ความสามารถทางด้านเน็ตเวิร์ค

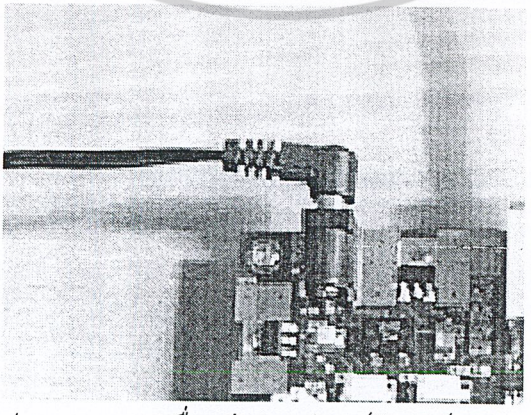
4. เปิดโปรแกรม terminal ที่ใช้งานขึ้นมาและทำการกำหนดค่าต่างๆ ดังนี้



รูปภาพการตั้งค่าต่างๆ ของโปรแกรมเทอร์มินัล

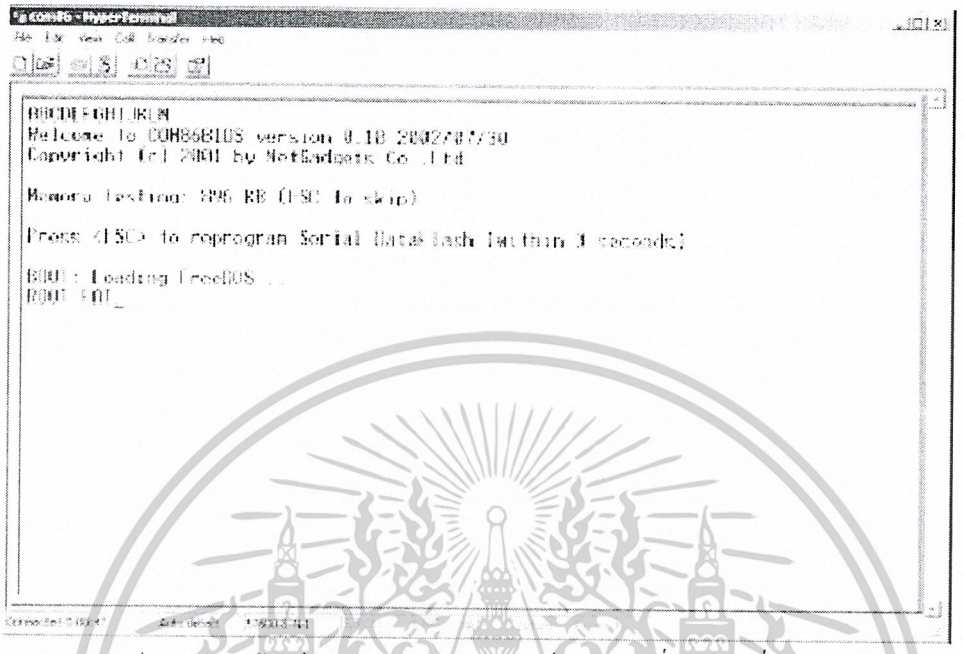
- เลือกพอร์ตการเชื่อมต่อของคอมพิวเตอร์ ตามที่ได้ทำการเชื่อมต่อในขั้นตอนที่ 2
- กำหนดอัตราการรับส่งข้อมูลอยู่ที่ 57600 บิตต่อวินาที
- ข้อมูลเป็นแบบ 8 บิต
- ไม่มีพาริตี้
- Stop บิตเท่ากับ 1
- ยกเลิกการทำโฟลว์คอนโทรลทั้งทางด้านฮาร์ดแวร์และซอฟต์แวร์ และสั่งเริ่มการเชื่อมต่อ

5. ต่อปลั๊กของอแดปเตอร์เข้ากับบอร์ดคอม 86และต่ออแดปเตอร์เข้ากับแหล่งจ่ายไฟของอาคาร



รูปภาพแสดงการเชื่อมต่ออแดปเตอร์กับบอร์ดคอม86

6. กดรีเซตสวิตช์จะเห็นโปรแกรมบนบอร์ดเริ่มทำงานโดยการแสดงผลผ่านโปรแกรมทอร์มินัลที่ใช้งาน



รูปภาพแสดงตัวอย่างการแสดงผลของบอร์ดคอม86เมื่อบอร์ดเริ่มทำงาน

4. การ transfer file ระหว่าง PC/COM86

ในการพัฒนาแอพลิเคชั่นต่างๆบนบอร์ดคอม 86นั้น นักพัฒนาย่อมเกิดความต้องการในการรับส่ง file ระหว่างบอร์ดคอม86 กับภายนอกขึ้นมาอย่างแน่นอน ซึ่งในชุดพัฒนาคอม86 ได้มีโปรแกรมยูทิลิตี้ตัวหนึ่งในการรองรับการทำงานในส่วนนี้เป็นโปรแกรมชื่อว่า transfer.exe เป็นโปรแกรมที่ช่วยในการรับ-ส่ง file ต่างๆ ผ่านทางพอร์ตอนุกรมคล้ายกับการโปรแกรม Image file ของบอร์ดคอม 86ซึ่งจะใช้โปรโตคอล X-modem ในการรับส่งไฟล์เช่นเดียวกันที่ความเร็ว 57600 bps ซึ่งรูปแบบการใช้งานของโปรแกรมมีดังนี้

Transfer [/S] [/R] [/H] [/?] filename

/S คือเลือกใช้การส่ง file ไปยัง PC

/R คือเลือกใช้การรับ file จาก PC)ค่านี้เป็นค่า default ของโปรแกรม(

/? หรือ/ H เป็นการเรียกดูรูปแบบการใช้งานของโปรแกรม

filename คือ ชื่อของไฟล์ที่ต้องการจะรับหรือส่ง

ตัวอย่างการรับ file จาก PC

การรับ file จาก PC สามารถทำได้โดยบน DOS prompt ของบอร์ดคอม 86เรียกใช้โปรแกรมดังนี้

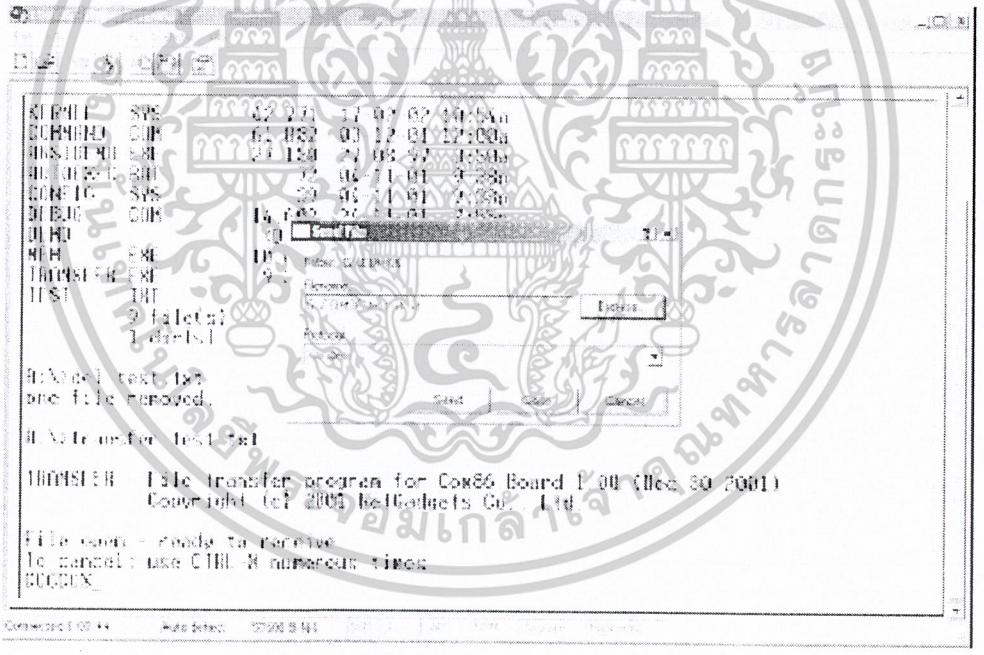
```
A:\>transfer /R  test.txt
```

เมื่อกด ปุ่ม Enter โปรแกรมแสดงข้อความต่างๆ ดังนี้

```
TRANSFER - File transfer program for Com86 Board 1.00 (Dec 30 2001)
Copyright (c) 2001 NetGadgets Co., Ltd.

File open - ready to receive
To cancel: use CTRL-X numerous times
```

หลังจากนั้นทำการส่ง file โดยการใช้โปรแกรม terminal ของทางฝั่ง PC โดยเลือกโพรโตคอลในการส่ง file เป็น X-modem ดังตัวอย่าง



รูปการส่ง file จาก PC

เมื่อทางฝั่งบอร์ดคอม 86รับ file เสร็จสิ้นก็จะกลับไปยัง DOS prompt เพื่อรอรับคำสั่งในการทำงานต่อไป

(หมายเหตุ : เนื่องจากข้อจำกัดของโพรโตคอล X-modem ที่ใช้จะทำให้ขนาดของไฟล์ต่างๆ ที่รับ-ส่งนั้นถูกปัดให้มีขนาดลงตัวที่ 128 ไบต์)

ตัวอย่างการส่ง file ไปยังเครื่องPC การส่งไฟล์ไปจากบอร์ดคอม 86ยังเครื่อง PC จะกระทำเช่นเดียวกับการรับไฟล์จากเครื่อง PC โดยการเรียกใช้งานโปรแกรม Transfer บน DOS Prompt ดังตัวอย่าง

```
A:\>transfer /s   test.txt
```

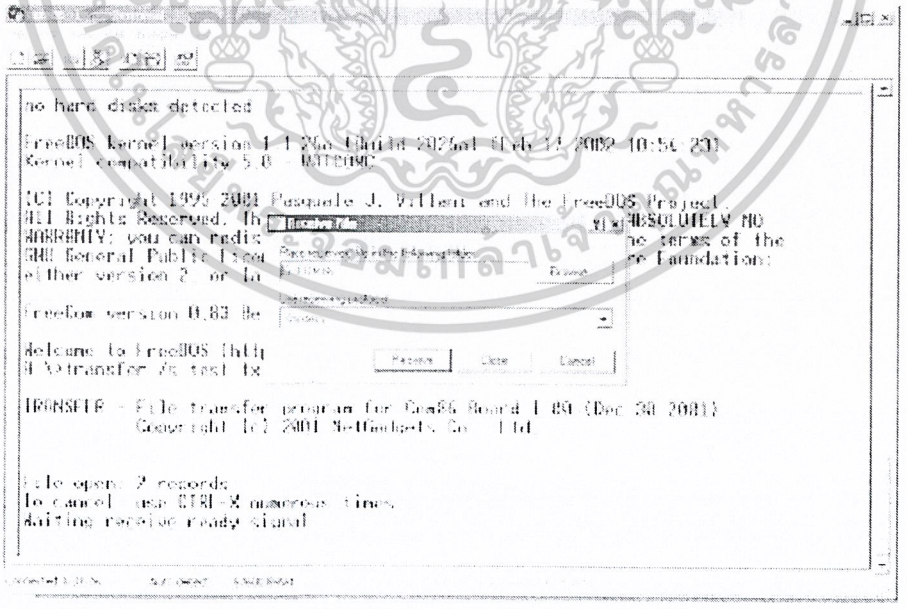
เมื่อกดคีย์ enter โปรแกรมจะแสดงข้อความดังนี้

```
A:\>transfer /s test.txt

TRANSFER - File transfer program for Com86 Board 1.00 (Dec 30 2001)
           Copyright (c) 2001 NetGadgets Co., Ltd.
File open: 2 records
```

```
To cancel: use CTRL-X numerous times
Waiting receive ready signal
```

ให้ไปที่โปรแกรมเทอร์มินัลที่ใช้งานอยู่และเลือกทำการรับ file จากเทอร์มินัลที่ใช้งานอยู่ โดยในการรับนี้ให้เลือกใช้โพรโตคอล X-modem ในการรับส่งข้อมูลเช่นเดียวกันดังตัวอย่าง



รูป การ receive file จาก com86

ซึ่งหลังจากส่งไฟล์เสร็จก็จะกลับมายัง DOS prompt เพื่อรอรับคำสั่งในการทำงานต่อไป

(หมายเหตุ เนื่องจากข้อจำกัดของโพรโตคอล X-modem ที่ใช้จะทำให้ขนาดของไฟล์ต่างๆ ที่รับ-ส่งนั้นถูกปัดให้มีขนาดลงตัวที่ 128 ไบต์)

5. การรันโปรแกรมบนบอร์ดคอม86

ในการรันโปรแกรมบนบอร์ดคอม 86นั้นจะกระทำเหมือนกับการรันโปรแกรมต่างๆ บน DOS ของเครื่อง PC ปกติโดยเพียง download file ที่ต้องการจะนำมารันเข้ามายังบอร์ดคอม 86ก่อน จากนั้นไปยังไดเรคทอรีที่เก็บไฟล์โปรแกรมที่ต้องการเรียกใช้งานเอาไว้ และทำการเรียกโปรแกรมตามปกติเหมือนกันกับบนคอมพิวเตอร์ทั่วไป ในกรณีที่ต้องการให้โปรแกรมเริ่มทำงานทุกครั้งโดยอัตโนมัติเมื่อเริ่มระบบสามารถทำได้โดยการสั่งเรียกใช้งานโปรแกรมนั้นๆ ภายใน fileAutoexec.bat ซึ่งเมื่อระบบเริ่มทำงานโปรแกรมนั้นๆ ก็จะถูกเรียกใช้งานได้โดยอัตโนมัติ

6. การทดสอบการทำงานของวงจรอีเธอเน็ต

การทดสอบวงจรอีเธอเน็ตนั้นสามารถทำได้โดยการใช้โปรแกรมสาธิตการทำงานของอีเธอเน็ตบนบอร์ดคอม 86ที่ให้มาภายในชุดพัฒนาโดยมีขั้นตอนดังนี้

- เข้าไปยังไดเรคทอรี A:\ethdemo บนบอร์ดคอม86
- เรียกโปรแกรม Init.bat ขึ้นมาทำงานซึ่งจะเป็นการติดตั้งดีไวซ์ไดร์เวอร์สำหรับวงจรอีเธอร์เนตบนบอร์ดซึ่งหลังจากเรียกใช้งานแล้วจะมีข้อความแสดงขึ้นมาดังตัวอย่าง

```
Packet driver for REALTEK 8019 Plug&Play Ethernet Card, version 1.31
(950705)
(C) Copyright 1993-95 Realtek Semiconductor Co., LTD. All RightsReserved.
Found Card 0: EtherID=00:20:18:61:81:14,IO=300,IRQ=A on 8-bit slot.
System: 186 processor, ISA bus
Packet driver software interrupt is 0x60 (96)
Interrupt (IRQ) number 0x4 (4)
I/O port 0x300 (768)
My Ethernet address is 00:20:18:61:81:14
Driver is attached to card 0
```

- เชื่อมต่อสาย UTP เข้ากับบอร์ดคอม 86ซึ่ง LED d 2จะติดสว่างขึ้น
- ทดลองใช้ program ping.exe ทดสอบการทำงานโดยการพิมพ์คำสั่งที่ DOS prompt ดังนี้

```
A:\> ping 192.168.1.x
```

โดย x เป็นหมายเลข IP address ของเครื่องที่ต้องการทดสอบการรับส่งข้อมูลซึ่งโปรแกรมจะรายงานผลการทดสอบดังตัวอย่าง

```
A:\DEMO>ping 192.168.1.1
LXPING 2.5 - HP200LX TCP/IP Suite http://lxtcp.hplx.net/
Pinging [192.168.1.1]
sent PING # 1 , PING receipt # 1 : response time 0.00 seconds
Ping Statistics
Sent         : 1
Received     : 1
Success      : 100 %
Average RTT : 0.00 seconds
```

ซึ่งในขณะที่มีการทดสอบการรับ-ส่งข้อมูลนั้น LED D 1จะมีการกระพริบเป็นจังหวะ หากวงจรอีเธอเน็ตสามารถทำงานได้ถูกต้องจะได้ผลการทดสอบดังที่ได้กล่าวมาแต่หากผลการทดสอบไม่เป็นไปตามที่กล่าวมาให้ตรวจสอบดูตามคำแนะนำในส่วนของ "การแก้ปัญหาในเบื้องต้น"

* ค่า default ของ IP address ที่ตั้งไว้จะเป็น 192. 168. 1.99 netmask 255.25.255.0 สำหรับการเปลี่ยนค่า IP address ของบอร์ดคอม86นั้นสามารถทำได้โดยการแก้ไขภายใน file tcp.cfg ในไดเรคทอรี ethdemo

7. **ข้อควรระวังในการใช้งาน**

ในการใช้งานชุดพัฒนาคอม 86มีข้อควรระวังในการใช้งานต่างๆ ดังนี้

- ควรถอดปลั๊กอแดปเตอร์ทุกครั้งในการเชื่อมต่อวงจรต่างๆเข้ากับบอร์ดคอม86
- ไม่ควรติดตั้งใช้งานในบริเวณที่เปียกชื้นหรือมีความร้อนสูง
- ไม่ควรยกเลิกการจ่ายกระแสไฟฟ้าในขณะที่กำลังทำการโปรแกรมแฟลชไบออสหรือโปรแกรมอิมเมจของดิสก์เนื่องจากอาจก่อให้เกิดความเสียหายต่อบอร์ดคอม 86ได้

บรรณานุกรม

หนังสืออ้างอิง

[1] อภิเนตร อูนากูล , "OBJECT-ORIENTED ANALYSIS AND DESIGN" , ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง , 2543

[2] Jean J. Labrosse , "MicroC/OS-II The Real-Time Kernel" , R&D Books , 1999

เว็บไซต์อ้างอิง

[1] http://webservice.bea.com/index.html

[2] http://www.perfectxml.com/Soft.asp?cat=16

[3] http://www.iturls.com/English/TechHotspot/TH_RealTimeOS.asp

[4] http://www.algonet.se/%7Estaffann/developer/realtimeintro.htm

[5] http://www.cs.arizona.edu/people/bridges/os/realtime.html

[6] http://www.tasking.com/products/C166-ST10/rtos.html

[7] http://www.eg3.com/WebID/rtos/rtos/blank/rtos/1-a-e.htm

[8] http://www.jive.nl/%7Ekamphuis/eff_c/index.html

[9] http://www.knowledgehound.com/topics/cpp.htm