บทที่ 4

การวิเคราะห์ และการออกแบบ SAFWA

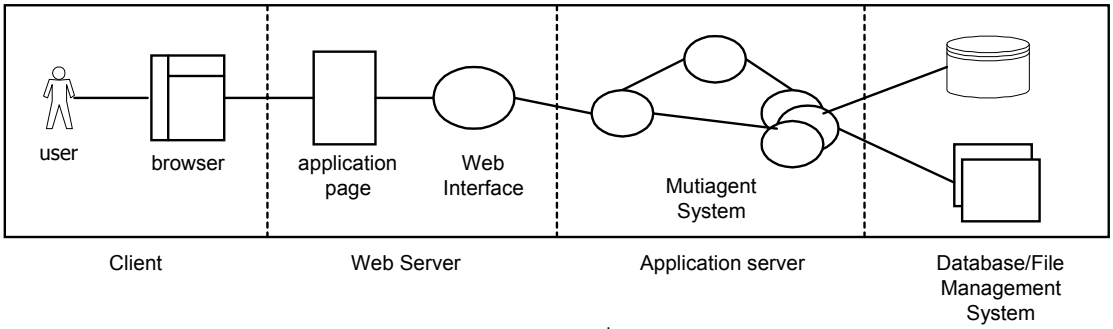
จากการศึกษาแนวคิดและการประยุกต์ใช้เอเจนท์ ในบทที่ 3 และการศึกษารูปแบบการพัฒนาโปรแกรมประยุกต์บนเว็บ ผู้วิจัยได้พบข้อสังเกตว่าเทคโนโลยีที่ใช้ในการพัฒนางานประยุกต์บนเว็บอาศัยหลักการทำงานแบบการเขียนสคริปต์ทางด้านผู้ให้บริการ (Server-sided script) โดยผู้เขียนโปรแกรมประยุกต์จะเขียนสคริปต์การทำงานต่าง ๆ ที่ต้องการ แทรกลงในเอกสารรูปแบบ HTML จากนั้นเมื่อมีการร้องขอการทำงาน เว็บเซิร์ฟเวอร์จะทำการแปลและทำงานตามสคริปต์ หลังจากนั้นก็สร้างเป็นเอกสาร HTML ไปยังผู้ร้องขอด้วยวิธีนี้ทำให้สามารถสร้างเว็บเพจ (Web page) ให้มีความยืดหยุ่นและสามารถทำงานตามต้องการได้

พิจารณาจากเทคโนโลยีข้างต้น สามารถสรุปข้อด้อยของการใช้เทคโนโลยีการเขียนสคริปต์ทางด้านผู้ให้บริการ ได้ดังนี้คือ

1. การเขียนโปรแกรมจะต้องแทรกสคริปต์การทำงานไปกับตัวไฟล์ที่ใช้ในการแสดงผล ซึ่งถ้าต้องการแก้ไขโปรแกรม หรือต้องการแก้ไขการแสดงผล ก็จะต้องทำการแก้ไขในแฟ้มเดียวกัน ไม่สามารถแยกแก้ไขได้
2. การขาดความยืดหยุ่นในการออกแบบและเขียนโปรแกรมสำหรับส่วนการแสดงผลและส่วนของโปรแกรม เมื่อต้องพัฒนาโดยอาศัยโปรแกรมเมอร์ (programmer) เพียงคนใดคนหนึ่ง ซึ่งโปรแกรมเมอร์รายนั้นอาจจะมีความถนัดเฉพาะงานเพียงอย่างใดอย่างหนึ่ง

เมื่อวิเคราะห์ถึงข้อด้อยของเทคโนโลยีที่มีใช้อยู่ทั่วไป จึงได้นำความรู้จากการศึกษาสถาปัตยกรรมเอเจนท์ และระบบมัลติเอเจนท์ มาประยุกต์และทำการออกแบบการสร้างโปรแกรมประยุกต์บนอินเตอร์เน็ตรูปแบบใหม่ โดยมีแนวคิดดังนี้

1. แยกการออกแบบและการทำงานเป็นส่วน ๆ ออกจากกัน เช่น ส่วนติดต่อผู้ใช้ ส่วนของโปรแกรมประยุกต์ และส่วนของโปรแกรมจัดการฐานข้อมูลหรือแฟ้มข้อมูล
2. นำแนวคิดของการให้บริการมาใช้โดยนำเอารูปแบบการทำงานแบบมัลติเอเจนท์มาใช้



ภาพประกอบ 4.1 สถาปัตยกรรมที่ประยุกต์ใช้สำหรับ SAFWA

จากภาพประกอบ 4.1 แสดงแผนภาพสถาปัตยกรรมที่ใช้ใน SAFWA ตามแนวคิดข้างต้น ซึ่งประกอบไปด้วยส่วนต่าง ๆ ดังนี้

Client ส่วนของ client เป็นส่วนที่ประกอบไปด้วย ผู้ใช้ และโปรแกรมแสดงผล ในลักษณะบราวเซอร์ โดยที่ผู้ใช้สามารถใช้งานโปรแกรมผ่านทางโปรแกรมเว็บบราวเซอร์ เช่น Internet Explorer, Netscape Navigator เป็นต้น

Web Server ในส่วน web server จะทำหน้าที่เก็บแฟ้มที่เป็น server page และใน server page จะมีการแทรกคำสั่งการเรียกใช้บริการ ซึ่งคำสั่งนั้นจะทำหน้าที่เรียก web interface ให้ทำงาน โดยที่ web interface ทำหน้าที่ในการร้องขอบริการและแสดงผลลัพธ์ที่ได้

Application Server ในส่วนนี้ประกอบไปด้วยระบบมัลติเอเจนท์ที่รอรับการร้องขอบริการจาก web interface และส่งคำร้องขอการทำงานไปยังเอเจนท์ที่เกี่ยวข้อง จากนั้นเอเจนท์ที่ทำงานเสร็จจะส่งผลลัพธ์กลับไปที่ Web interface

Database/File Management System ในส่วนนี้จะเป็นส่วนของระบบการจัดการแฟ้ม และฐานข้อมูลซึ่งเอเจนท์ที่ทำหน้าที่ให้บริการจะทำหน้าที่ในการเข้าถึงและ จัดการกับฐานข้อมูลเพื่อให้บริการแก่ผู้ร้องขอบริการ

จากการออกแบบข้างต้น เมื่อแยกการออกแบบและการทำงานส่วนต่าง ๆ ออกจากกันทำให้ ง่ายต่อการออกแบบ การพัฒนา และการแก้ไขซึ่งจะไม่กระทบการทำงานในส่วนอื่น

4.1 การวิเคราะห์โครงสร้างและการทำงานของ SAFWA

เนื่องจาก SAFWA ประกอบไปด้วยการทำงานหลายส่วน ซึ่งแต่ละส่วนแทนการทำงานด้วยเอเจนท์ ดังนั้นการวิเคราะห์ส่วนประกอบพื้นฐานของเอเจนท์จึงเป็นส่วนที่ต้องกระทำเป็นอันดับแรก หลังจากการวิเคราะห์เอเจนท์พื้นฐานแล้วสามารถนำเอเจนท์พื้นฐานมาวิเคราะห์ต่อเพื่อให้ได้เอเจนท์ที่ทำงานเฉพาะในส่วนต่างๆ และนำมาประกอบกันเป็น SAFWA ต่อไปได้

4.1.1 การวิเคราะห์เอเจนท์พื้นฐาน

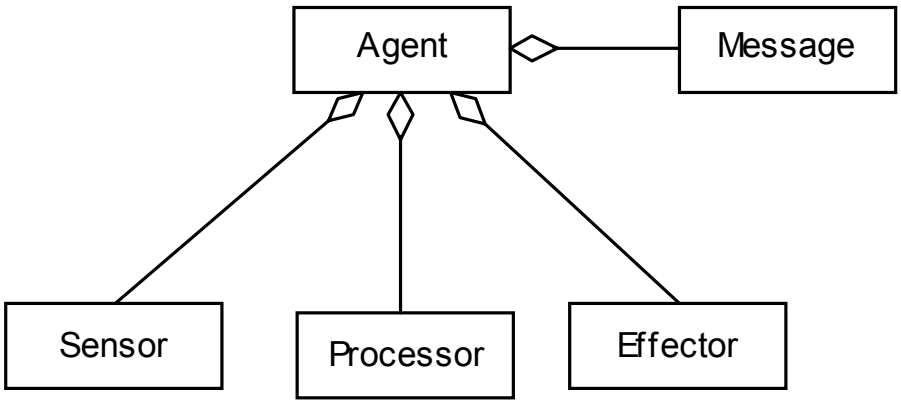
เอเจนท์พื้นฐานมีหน้าที่การทำงานสองส่วนหลักๆ ดังนี้

1. การรับส่งข้อความ และการประมวลผลข้อความ การรับส่งข้อความประกอบไปด้วยสององค์ประกอบย่อยคือ ส่วนรับ และส่วนที่ส่งข้อความ โดยที่ข้อความที่ได้รับ หรือที่จะทำการส่งอาศัยการทำงานของคลาสที่รวบรวมการทำงานเกี่ยวกับข้อความเอาไว้
2. ส่วนของการประมวลผลข้อความสำหรับเอเจนท์จะขึ้นอยู่กับหน้าที่ของเอเจนท์แต่ละตัว โดยที่เอเจนท์พื้นฐานจะไม่มีการทำงานในส่วนนี้ เนื่องจากการทำงานในส่วนนี้จะจำเพาะสำหรับเอเจนท์แต่ละตัว

4.1.1.1 ส่วนประกอบของเอเจนท์พื้นฐาน

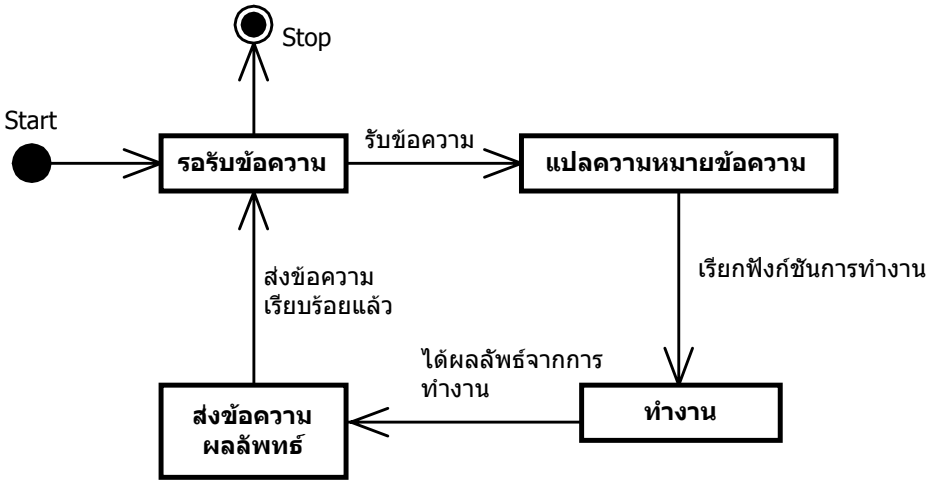
จากการวิเคราะห์ส่วนประกอบของเอเจนท์พื้นฐาน เอเจนท์พื้นฐานประกอบไปด้วยส่วนต่างๆ ดังนี้คือ ส่วนการรับข้อความ ส่วนการประมวลผลข้อความ ส่วนการส่งข้อความ และ คลาสที่ใช้ในการเก็บข้อความ และรวบรวมการทำงานเกี่ยวกับข้อความ ดังแสดงในภาพประกอบ 4.2

จากภาพประกอบ 4.2 ส่วนของ **Sensor** ทำหน้าที่ในการรอรับการร้องขอการติดต่อ และรอรับข้อความส่วน **Processor** ทำหน้าที่ในการทำงานตามหน้าที่ของตนเองซึ่งขึ้นอยู่กับว่าเอเจนท์นั้นทำหน้าที่อะไร ส่วน **Effector** ทำหน้าที่ในการตอบสนองหรือส่งข้อความไปยังเอเจนท์อื่น สุดท้ายคือ **Message** เป็นส่วนประกอบที่เอเจนท์จำเป็นต้องใช้ ทำหน้าที่ในการเก็บคำร้องขอ จัดรูปแบบคำร้องขอ และเก็บผลลัพธ์ในการทำงานเพื่อส่งต่อไปยังเอเจนท์อื่น



ภาพประกอบ 4.2 ส่วนประกอบของเอเจนท์พื้นฐาน

4.1.1.2 การทำงานภายในเอเจนท์พื้นฐาน



ภาพประกอบ 4.3 แผนภาพการทำงานภายในเอเจนท์เมื่อมีข้อความร้องขอบริการส่งเข้ามา

เอเจนท์พื้นฐานมีการทำงานดังแสดงในภาพประกอบ 4.3 ซึ่งแสดงให้เห็นถึงกระบวนการทำงานภายในที่เกิดขึ้นภายในเอเจนท์พื้นฐานเมื่อมีการร้องขอบริการ โดยมีขั้นตอนดังนี้

- เริ่มต้นรอรับข้อความจากเอเจนท์อื่น
- เมื่อรับข้อความเรียบร้อยแล้ว มีการเปลี่ยนสถานะการทำงานไปที่การแปลความหมายของข้อความเพื่อเลือกฟังก์ชันการทำงาน
- เมื่อเลือกได้แล้วทำการเรียกฟังก์ชันการทำงาน จะมีการเปลี่ยนสถานะเป็นการทำงานของฟังก์ชันที่เลือก

- หลังจากทำงานเสร็จ จะมีการเปลี่ยนสถานะการทำงานเป็นการจัดส่งผลลัพธ์
- เมื่อจัดส่งเรียบร้อยแล้ว มีการเปลี่ยนสถานะกลับไปที่ การรอรับข้อความตามเดิม

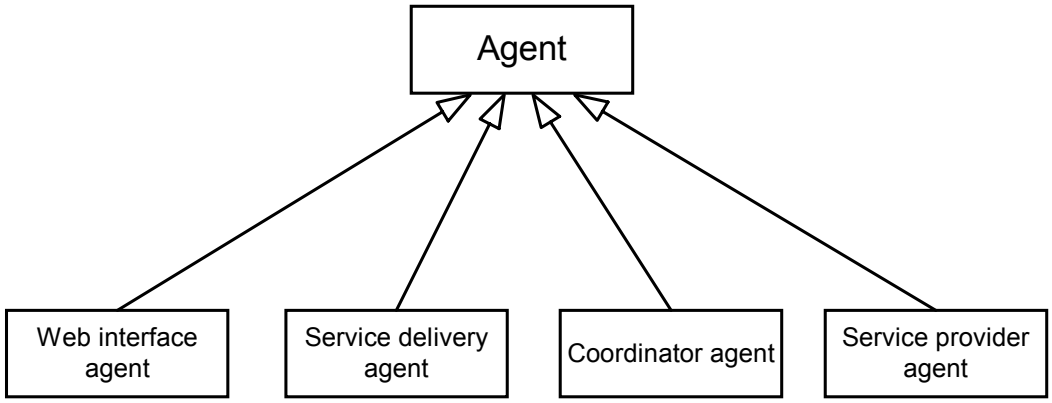
4.1.2 **การวิเคราะห์ส่วนประกอบของ** SAFWA

เมื่อเอเจนท์แต่ละตัวมีหน้าที่ต่างกัน แต่มีโครงสร้างเหมือนกัน เมื่อใช้วิธีการวิเคราะห์เชิงวัตถุก็สามารถนำเอาแนวคิดของการถ่ายทอดมาใช้ได้โดยสร้างเอเจนท์ใหม่จากเอเจนท์พื้นฐานในลักษณะการถ่ายทอดคุณสมบัติที่จำเป็นและเพิ่มความสามารถใหม่ ๆ เข้าไป โดยที่เอเจนท์แต่ละตัวมีการทำงานที่สัมพันธ์กันเพื่อให้ SAFWA สามารถทำงานตามวัตถุประสงค์ได้

4.1.2.1 **เอเจนท์ที่ทำหน้าที่ต่าง ๆ ใน** SAFWA

ในภาพประกอบ 4.4 แสดงความสัมพันธ์ระหว่างเอเจนท์ที่ทำหน้าที่ต่างๆ กับเอเจนท์พื้นฐาน โดยเอเจนท์ที่ทำหน้าที่ต่างๆ ถูกถ่ายทอดคุณสมบัติการทำงานจากเอเจนท์พื้นฐาน ได้แก่การรับส่งข้อความ การประมวลผลข้อความ และคลาสที่ใช้ในการเก็บข้อ

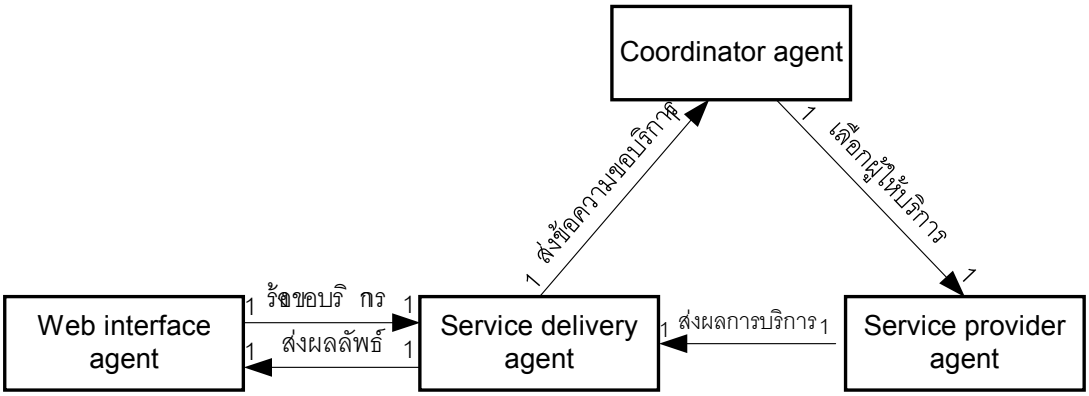
เอเจนท์ใน SAFWA ประกอบไปด้วยเอเจนท์ทีทำหน้าที่ต่างๆ กันดังนี้ Coordinator agent, Service delivery agent, Web interface agent และ Service provider agent โดยที่เอเจนท์แต่ละตัวจะมีการทำงานที่สัมพันธ์กันเพื่อให้ SAFWA สามารถทำงานได้ตามเป้าหมายที่กำหนดไว้



ภาพประกอบ 4.4 เอเจนท์ที่ทำหน้าที่ต่างๆ ใน SAFWA

4.1.2.2 **หน้าที่การทำงานของเอเจนท์ใน** SAFWA

พิจารณาเอเจนท์ที่ทำหน้าที่ต่างๆ ใน SAFWA เอเจนท์แต่ละตัวมีหน้าที่เฉพาะสำหรับทำงานอย่างใดอย่างหนึ่ง โดยที่เอเจนท์ต้องสามารถทำงานร่วมกันได้ โดยความสัมพันธ์ระหว่างเอเจนท์ต่างๆ แสดงในภาพประกอบ 4.5

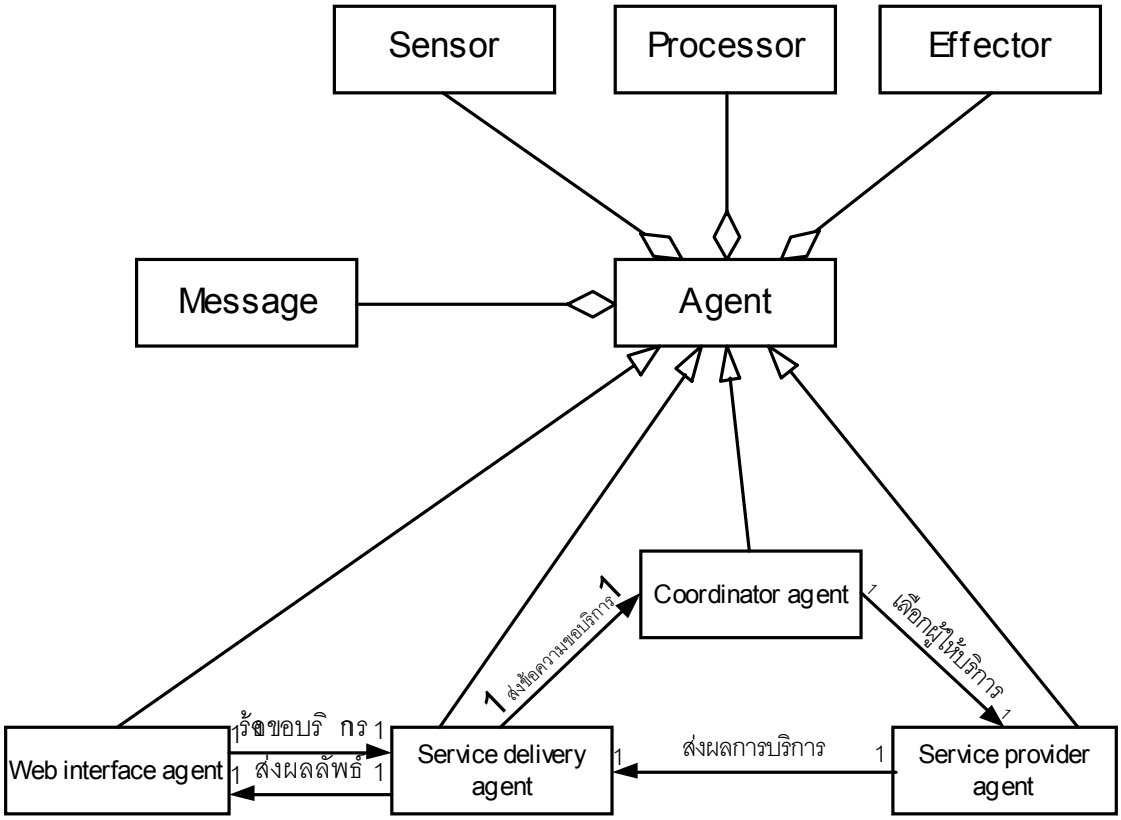


ภาพประกอบ 4.5 ความสัมพันธ์ระหว่างเอเจนท์ที่ทำงานใน SAFWA

เอเจนท์แต่ละตัวที่อยู่ใน SAFWA มีหน้าที่ในการทำงานดังต่อไปนี้คือ

Web interface agent	มีการทำงานเชื่อมต่อกับ Web Server เพื่อ ให้สามารถแสดงผลลัพธ์การทำงานบน Browser ได้ และสามารถส่งข้อความร้องขอบริการ ไปยัง Service delivery agent ได้
Service delivery agent	ทำหน้าที่ในการจัดการตีความข้อความร้องขอบริการ และ จัดเตรียมข้อความสำหรับผลลัพธ์ที่ได้จากการรับบริการ
Coordinator agent	ทำหน้าที่ในการรับข้อความขอบริการ และทำหน้าที่ในการเลือก ผู้ให้บริการที่ตรงกับความต้องการ และส่งข้อความขอบริการไปยังเอเจนท์ที่เลือกไว้
Service provider agent	ทำหน้าที่ให้บริการตามคำร้องขอที่ส่งมาจาก Coordinator agent โดยที่ Service provider agent จะทำงานในส่วนของการติดต่อฐานข้อมูล และอื่น ๆ ขึ้นอยู่กับการออกแบบว่า ระบบของเราจำเป็นต้องมี

งานอะไรบ้าง และหน้าที่อีกอย่างหนึ่งคือการส่งผลลัพธ์กลับไปยัง Service delivery agent



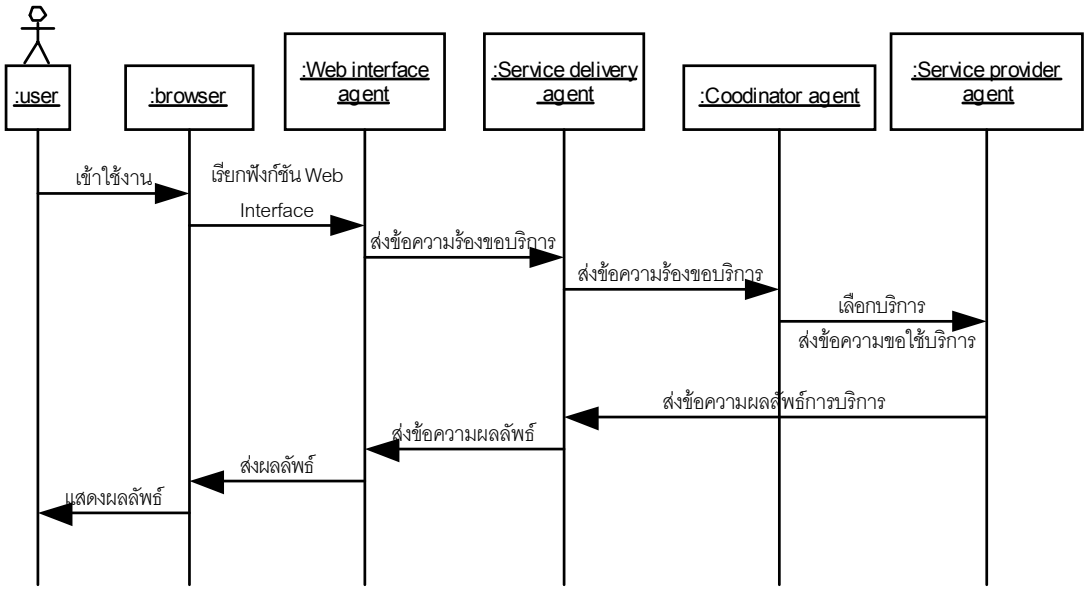
ภาพประกอบ 4.6 แผนภาพความสัมพันธ์ของ SAFWA

เมื่อทำการวิเคราะห์ส่วนประกอบทั้งหมดแล้วก็จะได้แผนภาพแสดงความสัมพันธ์ทั้งหมดระหว่างเอเจนท์พื้นฐานและเอเจนท์ที่ทำหน้าต่างๆ ใน SAFWA ดังแสดงในภาพประกอบ 4.6 โดยที่มีเอเจนท์พื้นฐานเป็นเอเจนท์หลักมีหน้าที่การทำงานคือ การรับส่งข้อความ การประมวลผลข้อความ รวมไปถึงคลาสข้อความที่ทำหน้าที่ในการจัดเก็บข้อความ และหน้าที่อื่นๆ ทีเกี่ยวกับการจัดการข้อความ เมื่อได้เอเจนท์พื้นฐานแล้ว ก็นำมาเพิ่มเติมความสามารถในการทำงานอื่น ๆ เพื่อให้เอเจนท์มีหน้าที่การทำงานที่แตกต่างกัน และเป็นหน้าที่เฉพาะของแต่ละเอเจนท์ โดยที่เอเจนท์แต่ละตัวจะทำงานร่วมกันเพื่อให้ SAFWA สามารถทำงานตามวัตถุประสงค์ที่ต้องการได้ เอเจนท์แต่ละตัวมีการทำงานร่วมกันโดยอาศัยการรับส่งข้อความ โดยที่ข้อความสามารถแบ่งออกได้สอง

ประเภทใหญ่ ๆ คือข้อความร้องขอ และข้อความที่เป็นผลลัพธ์ในการทำงาน เมื่อวิเคราะห์ได้หมดแล้วก็สามารถนำมาประกอบเป็นโครงสร้างพื้นฐานได้

4.1.3 **การวิเคราะห์การทำงานของ** SAFWA

หลังจากผ่านการวิเคราะห์ส่วนประกอบต่าง ๆ ที่เกี่ยวข้องกับ SAFWA แล้ว การวิเคราะห์ลำดับการทำงานของ SAFWA ก็เป็นงานที่ต้องทำในลำดับถัดมา ดังภาพประกอบ 4.7 แสดงให้เห็นลำดับขั้นตอนการทำงานของ SAFWA ที่อยู่บนพื้นฐานแนวคิดการให้บริการ



ภาพประกอบ 4.7 ลำดับขั้นตอนการทำงานของ SAFWA

4.2 **การออกแบบ** SAFWA

การออกแบบ SAFWA เป็นการขยายรายละเอียดขององค์ประกอบต่างๆ ที่ได้จากขั้นตอนการวิเคราะห์โดยที่การออกแบบเริ่มจากการออกแบบเอเจนท์พื้นฐานเป็นอันดับแรก จากนั้นก็ทำการออกส่วนประกอบภายใน SAFWA นั้นก็คือเอเจนท์ที่ทำหน้าที่ต่างๆ ได้แก่ Coordinator agent, Service provider agent, Service delivery agent และ Web interface agent พร้อมกับทำการออกแบบโครงสร้างข้อความสำหรับการสร้างภาษาสื่อสารที่เอเจนท์สามารถที่จะส่งและรับข้อความที่เป็นมาตรฐานเดียวกันได้ การออกแบบเอเจนท์และองค์ประกอบของเอเจนท์มีพื้นฐานอยู่บนการเขียนโปรแกรมเชิงวัตถุด้วยภาษาจาวา และแผนภาพที่ใช้ประกอบมีพื้นฐานอยู่บนการออกแบบโดยใช้ UML (Unified Modeling Language) ดังนั้นองค์ประกอบต่าง จะถูกออกแบบให้อยู่ในรูปของคลาสในภาษาจาวา

4.2.1 การออกแบบเอเจนท์พื้นฐาน

เอเจนท์พื้นฐานที่ได้จากการวิเคราะห์ประกอบไปด้วยส่วนต่างๆ คือ Sensor, Effector, Processor, Message และ Dummy agent โดยแต่ละส่วนมีรายละเอียดการออกแบบดังนี้คือ

4.2.1.1 การออกแบบ Dummy agent

ภาพประกอบ 4.8 แผนภาพคลาส DummyAgent

Dummy agent ทำหน้าที่ในการเริ่มต้นการทำงาน เช่นการอ่านการตั้งค่าต่างๆ การเริ่มต้นการทำงาน และยังเป็นส่วนควบคุมการทำงานทั้งหมด โดยที่การออกแบบเอเจนท์อื่นๆ สามารถทำได้โดยการถ่ายทอดคุณสมบัติจาก Dummy agent และทำการเพิ่มคุณสมบัติใหม่ที่ต้องการเข้าไป

ในภาพประกอบ 4.8 แสดงแผนภาพคลาส DummyAgent ซึ่งสามารถอธิบายความหมายและหน้าที่การทำงานต่างๆ ได้ดังรายละเอียดในตาราง 4.1

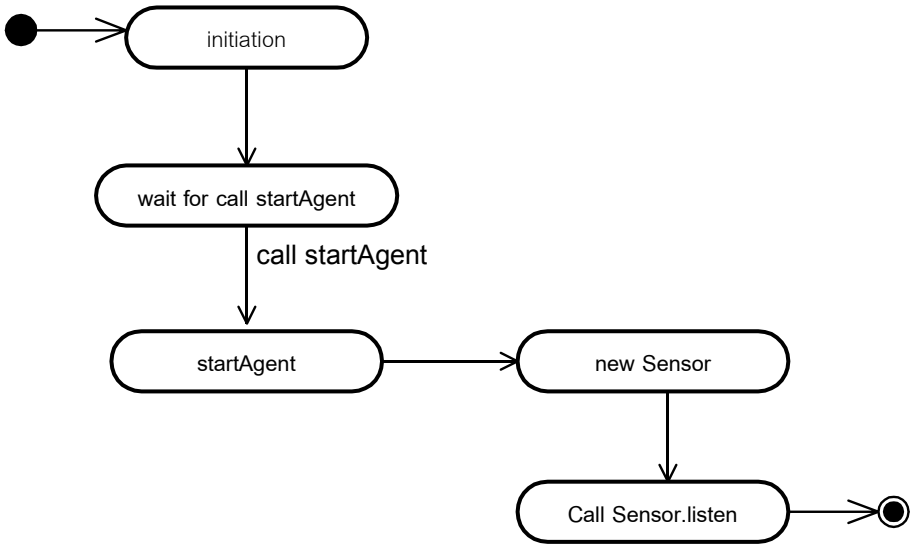
ตาราง 4.1 แสดงการอธิบายตัวแปรและเมธอดของคลาส DummyAgent

ตัวแปร (Data member)	**คำอธิบาย**
- serverPort	ใช้สำหรับค่าตั้งต้นให้ sensor เพื่อทำการเปิด socket สำหรับรอรับข้อความ
- qLength	เป็นค่าเริ่มต้นสำหรับการเปิด socket

ตาราง 4.1 (ต่อ)

ตัวแปร (Data member)	**คำอธิบาย**
- loop	สำหรับกำหนดให้ sensor ทำงานแบบรอรับครั้งเดียวหรือรอรับข้อความตลอดเวลา
- listener	เป็น instance ของ interface AgentListner
เมธอด (Method)	**คำอธิบาย**
+ DummyAgent	Constructor เรียกเมธอด initiation
+ setLoop	ตั้งค่า loop
+ setQLength	ตั้งค่า qLength
+ getQLength	ส่งค่า qLength
+ setServerPort	ตั้งค่า serverPort
+ getServerPort	ส่งค่า serverPort
+ setListener	ส่งค่าสำหรับคลาสที่ implement AgentListener
+ initiation	เมธอดจะถูก overriding ด้วย เมธอดของเอเจนท์ที่ทำหน้าที่เฉพาะ ใช้สำหรับการตั้งค่าเริ่มต้นต่าง ๆ ของเอเจนท์
+ messageRecieved	ทำหน้าที่เรียกการทำงานต่างสำหรับข้อความที่ได้รับ
+ startAgent	ประกอบด้วยคำสั่งให้ sensor เริ่มทำงาน

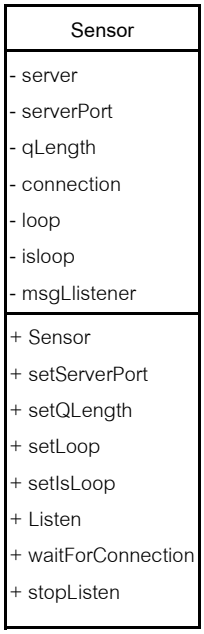
ภาพประกอบ 4.9 แสดงกิจกรรมที่เกิดขึ้นในคลาส DummyAgent เริ่มจากการที่ทำการสร้าง instance ของคลาส DummyAgent ขึ้นมาใหม่โดย Constructor จะทำการเรียกเมธอด initiation หลังจากนั้นเมื่อต้องการให้เอเจนท์ทำงานก็จะเรียกเมธอด startAgent โดยที่เมธอด startAgent มีกระบวนการภายในคือการสร้าง Sensor ขึ้นมาและทำการเรียกให้ Sensor เริ่มทำการรอฟังข้อความ



ภาพประกอบ 4.9 กิจกรรมที่เกิดขึ้นสำหรับคลาส DummyAgent

4.2.1.2 การออกแบบ Sensor

Sensor มีหน้าที่ในการรอรับข้อความที่เข้ามาทางช่องทางสื่อสารที่เปิดไว้ เมื่อได้รับข้อความแล้วจะทำการส่งต่อข้อความนั้นให้คลาส Message เพื่อทำการตีความและเก็บข้อความนั้นไว้ใช้ จากนั้นจะมีการเรียกใช้คลาส Processor เพื่อทำงานตามที่ถูกร้องขอ



ภาพประกอบ 4.10 แผนภาพคลาส Sensor

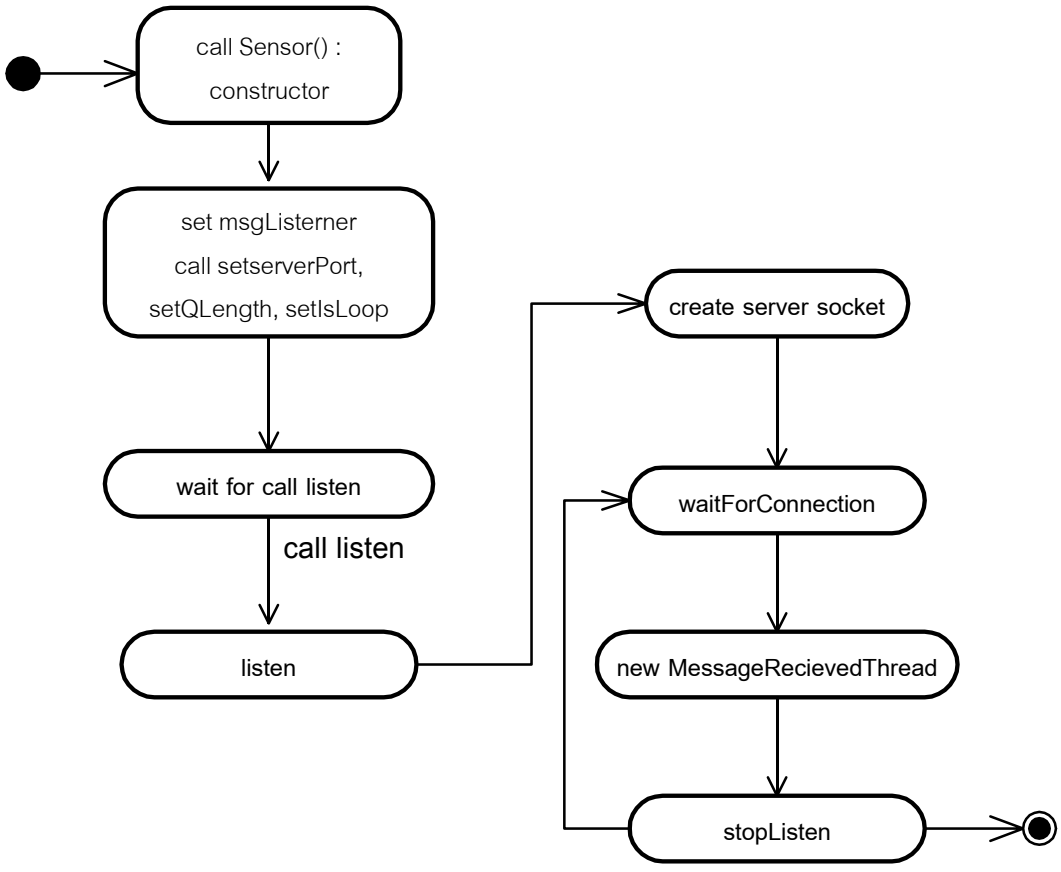
ในภาพประกอบ 4.10 แสดงแผนภาพคลาส Sensor ซึ่งสามารถอธิบายความหมายและหน้าที่การทำงานต่างๆ ได้ดังรายละเอียดในตาราง 4.2

ตาราง 4.2 แสดงการอธิบายตัวแปรและเมธอดของคลาส Sensor

ตัวแปร (Data member)	**คำอธิบาย**
- server	ตัวแปร ServerSocket เพื่อทำหน้าที่เปิด socket สำหรับรอรับการติดต่อ
- serverPort	ใช้สำหรับค่าตั้งต้นให้ sensor เพื่อทำการเปิด socket สำหรับรอรับข้อความ
- qLength	เป็นค่าเริ่มต้นสำหรับการเปิด socket
- connection	ตัวแปร Socket เพื่อสร้าง connection สำหรับการติดต่อ
- loop	สำหรับกำหนดให้ sensor ทำงานแบบรอรับครั้งเดียวหรือ รอรับข้อความตลอดเวลา
- isLoop	ใช้เพื่อตรวจสอบว่าต้องการให้มีการทำงานเป็น loop หรือไม่
- msgListener	เป็น instance ของ interface AgentListner
เมธอด (Method)	**คำอธิบาย**
+ Sensor	Constructor เรียกเมธอด setServerPort ,setQLength ,setIsLoop และตั้งค่า msgListener
+ setServerPort	ตั้งค่า serverPort
+ setQLength	ตั้งค่า qLength
+ setLoop	ตั้งค่า Loop
+ setIsLoop	ตั้งค่า isLoop
+ listen	ถูกเรียกเพื่อรอรับการติดต่อและรับข้อความ ทำการับข้อความโดยการเรียกคลาส MessageRecievedThread
+ waitForConnection	สำหรับรอการติดต่อ
+ stopListen	สำหรับปิดการติดต่อ หลังจากรับข้อความแล้ว

จากภาพประกอบ 4.9 มีสร้าง instance ของคลาส Senser และเรียกใช้เมธอด listen จะเกิดกิจกรรมดังนี้คือ เมื่อทำการสร้าง instance ตัว Constructor ทำการตั้งค่า

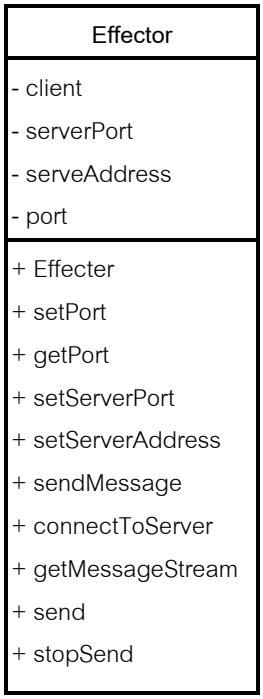
ตัวแปร และเรียกเมธอดสำหรับการตั้งค่าตัวแปรเริ่มต้นคือ setServerPort, setQLength, setIsLoop และตั้งค่าตัวแปร msgListener โดยที่ตัวแปรเหล่านี้ถูกส่งผ่านมาจาก DummyAgent หลังจากทำการตั้งค่าเริ่มต้นของตัวแปรเรียบร้อยแล้ว เมื่อ DummyAgent เรียกใช้เมธอด listen ภายในเมธอด listen เกิดกิจกรรมดังนี้คือ สร้าง socket สำหรับรอรับการติดต่อ ทำการรอรับการติดต่อ เมื่อมีการติดต่อเข้ามาก็จะทำการรับข้อความโดยการเรียกการทำงานของคลาส MessageRecieved Thread แล้วทำการยกเลิกการติดต่อ แล้วกลับไปรอรับการติดต่อครั้งใหม่เพื่อรอข้อความใหม่ต่อไป ดังแสดงในภาพประกอบ 4.11



ภาพประกอบ 4.11 กิจกรรมที่เกิดขึ้นสำหรับคลาส Sensor

4.2.1.3 การออกแบบ Effector

Effector มีหน้าที่ในสร้างช่องทางสำหรับการติดต่อเพื่อส่งข้อความไปยังเอเจนท์ปลายทาง โดยปกติแล้ว Effector จะถูกเรียกให้ส่งข้อความในขั้นตอนของการประมวลผลหรือในส่วนของคลาส Processor ซึ่งขึ้นอยู่กับเมธอดที่สร้างขึ้นสำหรับทำงานของเอเจนท์แต่ละตัว



ภาพประกอบ 4.12 แผนภาพคลาส Effector

ในภาพประกอบ 4.12 แสดงแผนภาพคลาส Effector ซึ่งสามารถอธิบายความหมายและหน้าที่การทำงานต่างๆ ได้ดังรายละเอียดในตาราง 4.3

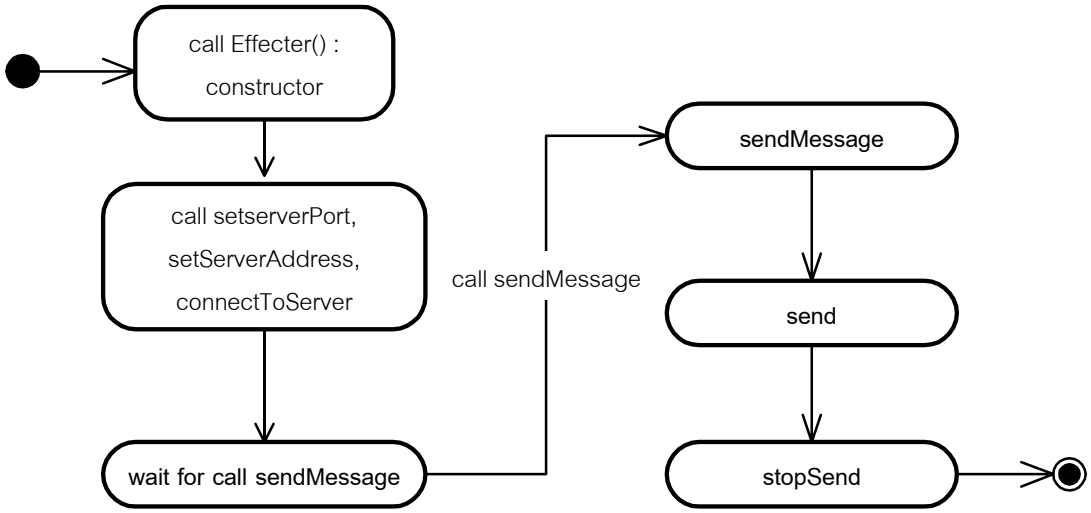
ตาราง 4.3 แสดงการอธิบายตัวแปรและเมธอดของคลาส Effector

ตัวแปร (Data member)	**คำอธิบาย**
- client	ตัวแปร ServerSocket เพื่อทำหน้าที่เปิด socket สำหรับรอรับการติดต่อ
- serverPort	ใช้สำหรับค่าตั้งต้นให้ sensor เพื่อทำการเปิด socket สำหรับรอรับข้อความ
- serverAddress	เป็นค่าเริ่มต้นสำหรับการเปิด socket
- port	ตัวแปร Socket เพื่อสร้าง connection สำหรับการติดต่อ

ตาราง 4.3 (ต่อ)

เมธอด (Method)	คำอธิบาย
+ Effector	Constructor เรียกเมธอด setServerPort, setServerAddress และ connectToServer เพื่อสร้างการติดต่อ
+ setPort	ตั้งค่า serverPort
+ getPort	ส่งค่า port
+ setServerPort	ตั้งค่า serverPort
+ setServerAddress	ตั้งค่า serverAddress
+ sendMessage	ถูกเรียกเพื่อส่งข้อความ ภายในเรียกเมธอด send และ stopSend
+ connectToServer	สำหรับรอการติดต่อ
+ getMessageStream	สำหรับปิดการติดต่อ หลังจากรับข้อความแล้ว
+ send	ทำการส่งข้อความผ่านทางช่องทางติดต่อ
+ stopSend	ปิดช่องทางการติดต่อ

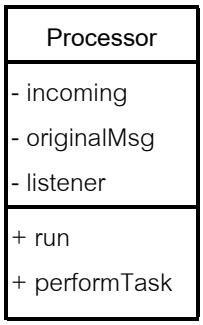
กิจกรรมที่เกิดขึ้นเมื่อมีการเรียกใช้ Effector คือ constructor ทำการตั้งค่า serverPort และ serverAddress ซึ่งเป็นตัวระบุปลายทางที่จะส่งข้อมูลจากนั้นทำการติดต่อเพื่อสร้างช่องทาง จากนั้นก็รอเพื่อให้มีการเรียก sendMessage ในการส่งข้อความ โดยที่มีการเรียกเมธอด send เพื่อส่งข้อความผ่านไปยังช่องทางติดต่อ และเรียก stopSend ในการตัดการติดต่อ กิจกรรมที่เกิดขึ้นแสดงในภาพประกอบ 4.13



ภาพประกอบ 4.13 กิจกรรมที่เกิดขึ้นสำหรับคลาส Effector

4.2.1.4 การออกแบบ Processor

Processor มีหน้าที่สำหรับทำงานตามที่มีการร้องขอ โดยที่ข้อความจะระบุถึงฟังก์ชันที่ต้องการ จากนั้น Processor จะทำการเลือกฟังก์ชันที่ตรงกับที่ระบุมาทำงาน เมื่อทำงานใดๆ เสร็จก็จะมีการสร้างข้อความผลลัพธ์และส่งผลลัพธ์ต่อไปยัง Service delivery agent เพื่อนำไปแสดงผลต่อไป เนื่องจากเอเจนท์แต่ละตัวมีการทำงานที่ต่างกัน ในการออกแบบคลาส Processor จึงต้องออกแบบให้สามารถทำงานที่แตกต่างกันได้ นั่นคือฟังก์ชันที่ต่างกัน สามารถเพิ่มเติมจากตัวคลาส Processor ได้ทันทีโดยไม่ต้องมีการแก้ไขส่วนอื่นภายในคลาส โดยใช้เทคนิคการเรียกใช้ฟังก์ชันในขณะทำงาน (dynamic method calling) นั่นคือสามารถเพิ่มเมธอดที่ต้องการเข้าไปในคลาส ได้ทันทีในขณะที่คำสั่งที่ใช้ในการเรียกฟังก์ชันจะทำการเลือกฟังก์ชันที่จะมาทำงานจาก ตัวแปร function2Call ดังนั้นในเอเจนท์แต่ละตัวก็สามารถที่จะมีเมธอดสำหรับทำงานต่างกันได้แต่ฟังก์ชันสำหรับการเรียกเมธอดนั้นจะเหมือนเดิมไม่ต้องแก้ไข การเรียกเมธอดทำได้โดยการใช้เมธอด performTask แล้วส่งพารามิเตอร์ ที่เป็นชื่อของเมธอดที่เราต้องการให้ทำงาน



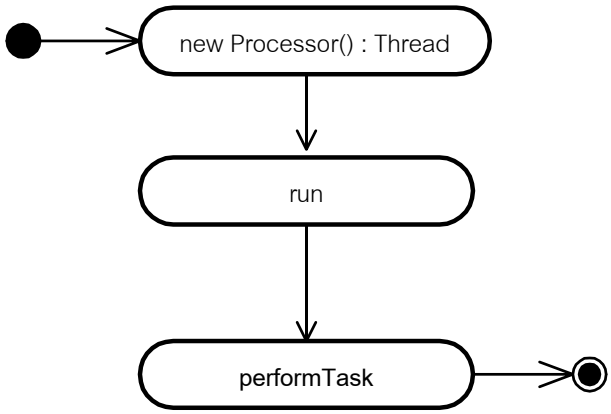
ภาพประกอบ 4.14 แผนภาพคลาส Processor

ในภาพประกอบ 4.14 แสดงแผนภาพคลาส Processor ซึ่งสามารถอธิบายความหมายและหน้าที่การทำงานต่างๆ ได้ดังรายละเอียดในตาราง 4.4

ตารางที่ 4.4 แสดงการอธิบายตัวแปรและเมธอดของคลาส Processor

ตัวแปร (Data member)	**คำอธิบาย**
- incoming	เป็น instance ของคลาส Message สำหรับเก็บข้อความร้องขอบริการ
- originalMsg	เป็น instance ของคลาส Message สำหรับเก็บข้อมูลเริ่มต้นของเอเจนท์เช่น IP address ชื่อเอเจนท์ เป็นต้น
- listener	เป็น instance ของ interface ActionListener อิมพลีเม้นท์โดย เอเจนท์
เมธอด (Method)	**คำอธิบาย**
+ run	เป็นเมธอดที่ ต้องทำการ override เนื่องจากคลาส Processor ถ่ายทอดคุณสมบัติจาก คลาส Thread ทำหน้าที่ในการเลือก ฟังก์ชันในการทำงาน และเรียกเมธอด performTask
+ performTask	มีการทำงานแบบ Dynamic method calling นำเอาค่าพารามิเตอร์ function2Call มาเป็นตัวเลือกเมธอดที่ต้องการทำงาน

คลาส Processor ถูกออกแบบมาให้เขียนโปรแกรมรวมกับ คลาสเอเจนท์ที่ทำหน้าที่ต่างๆ ดังนั้นเมื่อมีการสร้างเอเจนท์ ก็จะต้องทำการสร้างคลาส Processor ควบคู่ไปด้วย เมื่อมีการเรียกใช้งาน ภายในคลาสเกิดกิจกรรมดังนี้คือ สร้างเธรด (Thread) ขึ้นมาใหม่เนื่องจากคลาส Processor ถ่ายทอดคุณสมบัติจากคลาส Thread จากนั้นเมธอด run จะถูกเรียกให้ทำงาน ภายในเมธอด run จะทำการเรียกเมธอด performTask เพื่อทำการเรียกเมธอดที่ต้องการขึ้นมาทำงาน ดังแสดงในภาพประกอบ 4.15



ภาพประกอบ 4.15 กิจกรรมที่เกิดขึ้นสำหรับคลาส Processor

4.2.1.5 การออกแบบ Message

หน้าที่ของคลาส Message ก็คือการนำเอาข้อความที่ได้จากเอเจนท์อื่น มาทำการแปลงให้อยู่ในรูปของคลาสเพื่อนำไปใช้งาน หรือทำการแปลงคลาสให้กลับไปอยู่ในรูปของข้อความเพื่อทำการส่งไปยังเอเจนท์อื่น ดังนั้นเมธอดหลักของคลาส Message คือ translate และ pack โดย translate ทำหน้าที่แปลงข้อความมาเป็นคลาส และ pack ทำหน้าที่นำข้อมูลจากคลาสมาทำให้อยู่ในรูปของข้อความ โดยที่ข้อความอยู่ในรูปแบบของเอกสาร XML การจัดการกับเอกสาร XML นั้นได้ทำการนำเอาเครื่องมือคือ dom4j ซึ่งเป็น API สำหรับจาวาที่ใช้สำหรับการจัดการเอกสาร XML โดยเฉพาะ

ภาพประกอบ 4.16 แผนภาพคลาส Message

ในภาพประกอบ 4.16 แสดงแผนภาพคลาส Message ซึ่งสามารถอธิบายความหมายและหน้าที่การทำงานต่างๆ ได้ดังรายละเอียดในตาราง 4.5

ตาราง 4.5 แสดงการอธิบายตัวแปรและเมธอดของคลาส Message

ตัวแปร (Data member)	**คำอธิบาย**
- source	บอกถึงชื่อของเอเจนท์ที่เป็นผู้ส่งข้อความ เช่น catalog ซึ่งเป็นประเภท Service provider agent
- sourceType	บอกถึงประเภทของเอเจนท์ผู้ส่งข้อความเช่น SPA คือ Service provider agent

ตาราง 4.5 (ต่อ)

ตัวแปร (Data member)	**คำอธิบาย**
- sourceAddr	บอกถึงหมายเลข IP address ของเอเจนท์ผู้ส่งข้อความ
- sourcePort	บอกถึง Port ของเอเจนท์ผู้ส่งข้อความที่เปิดรอรับการส่งข้อความกลับ
- destination	บอกถึงชื่อของเอเจนท์ปลายทางที่รับข้อความ
- destinationAddr	บอกถึง IP address ของเอเจนท์ปลายทางที่รับข้อความ
- destinationPort	บอกถึง Port ของเอเจนท์ปลายทางเปิดรอรับการส่งข้อความ
- messageType	บอกถึงประเภทของข้อความเช่น ประเภทร้องขอบริการ หรือประเภทตอบรับ
- function2Call	บอกถึงฟังก์ชันที่ต้องการให้ทำการบริการ
- content	ใช้สำหรับส่งพารามิเตอร์ในการร้องขอบริการ และใช้สำหรับการส่งผลลัพธ์หลังจากให้บริการแล้ว
เมธอด (Method)	**คำอธิบาย**
+ setSource, + getSource	ตั้งค่าและส่งค่า source
+ setSourceType, + getSourceType	ตั้งค่าและส่งค่า sourceType
+ setSourceAddr, + getSourceAddr	ตั้งค่าและส่งค่า sourceAddr
+ setSourcePort, + getSourcePort	ตั้งค่าและส่งค่า sourcePort
+ setDestination, + getDestination	ตั้งค่าและส่งค่า destination
+ setDestinationAddr, + getDestinationAddr	ตั้งค่าและส่งค่า destinationAddr
+ setDestinationPort, + getDestinationPort	ตั้งค่าและส่งค่า destinationPort
+ setMessageType, + getMessageType	ตั้งค่าและส่งค่า messageType

ตาราง 4.5 (ต่อ)

เมธอด (Method)	คำอธิบาย
+ setFunction2Call, + getFunction2Call	ตั้งค่าและส่งค่า function2Call
+ setContent, + getContent	ตั้งค่าและส่งค่า content
+ translate	ทำหน้าที่แปลงข้อความในรูปแบบเอกสาร XML มาเป็นตัวแปรเก็บไว้ในตัวแปรต่าง ๆ ในคลาส Message
+ pack	ทำหน้าที่นำเอาค่าตัวแปรต่าง ๆ ในคลาส Message มาแปลงให้อยู่ในรูปแบบเอกสาร XML

นอกเหนือจากส่วนประกอบที่ข้างต้นแล้วเอเจนท์พื้นฐานจำเป็นต้องมีส่วนประกอบอื่นเพื่อให้เอเจนท์สามารถทำงานได้ โดยส่วนประกอบอื่นคือ AgentUtil, MessageListener และ RecieveMessageThread โดยหน้าที่และการออกแบบส่วนประกอบเหล่านี้มีดังนี้คือ

4.2.1.6 การออกแบบ AgentUtil

คลาส AgentUtil เป็นคลาสที่รวบรวมเอาเมธอดสำหรับช่วยเหลืองานต่าง ๆ เข้าไว้ด้วยกัน เช่น เมธอดสำหรับการจัดการกับเอกสาร XML เช่นการสร้างเอกสาร XML ใหม่ การอ่านข้อมูลที่ต้องการจากเอกสาร XML ซึ่งคลาสนี้สามารถถูกเรียกใช้โดยคลาสอื่นๆ ถ้าต้องการเมธอดที่ช่วยงานในด้านเอกสาร XML

ภาพประกอบ 4.17 แผนภาพคลาส AgentUtil

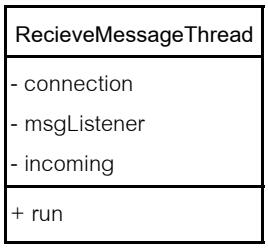
ในภาพประกอบ 4.17 แสดงแผนภาพคลาส AgentUtil ซึ่งสามารถอธิบายความหมายและหน้าที่การทำงานต่างๆ ได้ดังรายละเอียดในตาราง 4.6

ตาราง 4.6 แสดงการอธิบายตัวแปรและเมธอดของคลาส AgentUtil

เมธอด (Method)	คำอธิบาย
+ getAddress	ส่งค่า IP address ของเครื่องที่เอเจนท์กำลังทำงานอยู่
+ createXmlDocument	ใช้สำหรับสร้างเอกสารในรูปแบบของ dom4j
+ getXmlData	ส่งค้าข้อมูลที่อ่านได้จากเอกสารในรูปแบบ dom4j
+ getXmlDataDoc	ส่งค้าข้อมูลที่อ่านได้จากเอกสารในรูปแบบ dom4j และส่งข้อมูลออกมาในรูปแบบเอกสาร dom4j
+ readXml	อ่านเอกสารแฟ้มข้อมูล XML ให้อยู่ในรูปของเอกสาร dom4j
+ serialize2XML	ทำหน้าที่แปลงเอกสารในรูปแบบ dom4j ให้อยู่ในรูปแบบเอกสาร XML
+ transformElement2Doc	แปลงข้อมูลในรูปแบบ Element ให้อยู่ในรูปแบบเอกสาร dom4j

4.2.1.7 การออกแบบ RecieveMessageThread

สำหรับคลาส RecieveMessageThread ในขั้นตอนของคลาส Sensor เมื่อได้รับข้อความแล้วจะทำการสร้างเธรด RecieveMessageThread เพื่อนนำข้อความที่ได้รับมาไปทำงาน โดยที่คลาส RecieveMessageThread นี้หน้าที่คือรับสายของข้อมูล (Data Stream) จากช่องทางติดต่อ และแปลงข้อมูลที่ได้ให้อยู่ในรูปของข้อความและนำไปเก็บได้ในตัวแปรที่เป็นชนิดคลาส Message จากนั้นก็เรียกให้อินเตอร์เฟส MessageListener เรียกเมธอด recievedMessage ที่อยู่ในคลาส DummyAgent



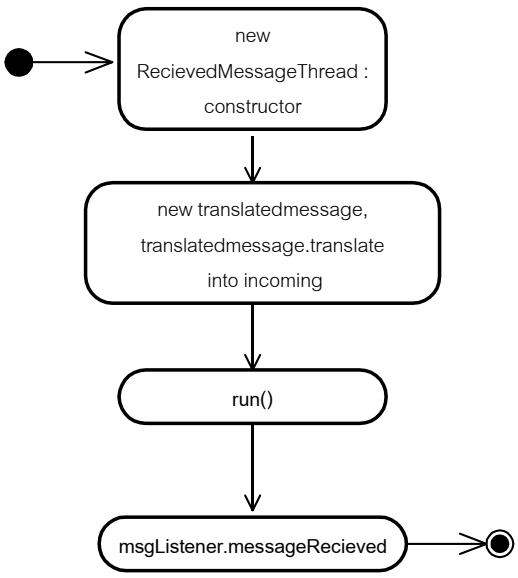
ภาพประกอบ 4.18 แผนภาพคลาส RecieveMessageThread

ในภาพประกอบ 4.18 แสดงแผนภาพคลาส RecieveMessageThread ซึ่งสามารถอธิบายความหมายและหน้าที่การทำงานต่างๆ ได้ดังรายละเอียดในตาราง 4.7

ตาราง 4.7 แสดงการอธิบายตัวแปรและเมธอดของคลาส RecieveMessageThread

ตัวแปร (Data member)	**คำอธิบาย**
- connection	ช่องทางติดต่อสำหรับรับข้อความ
- msgListener	เป็นตัวแปรชนิดอินเตอร์เฟส MessageListener ใช้สำหรับเรียกเมธอด messageRecieved ใน DummyAgent ให้ทำงาน
- incoming	ใช้สำหรับเก็บข้อความที่ได้รับ
เมธอด (Method)	**คำอธิบาย**
+ run	เป็นเมธอดที่ถ่ายทอดมาจาก คลาส Thread ทำหน้าที่สำหรับเรียกการทำงานของ msgListener

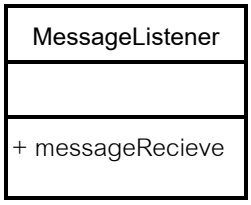
โดยกิจกรรมที่เกิดขึ้นสำหรับ RecieveMessageThread เป็นกระบวนการที่ต่อเนื่องจาก Sensor โดยมีกิจกรรมเกิดขึ้นดังต่อไปนี้คือ เมื่อ Senser ได้รับการส่งข้อความแล้วทำการส่งช่องทางติดต่อให้เธรด RecieveMessageThread โดยตัว constructor ทำหน้าที่ในการรับข้อมูลจากช่องทางติดต่อ จากนั้นทำการแปลงข้อความแล้วเก็บไว้ในตัวแปร incoming เมื่อทำหน้าที่เสร็จเธรดจะทำงานต่อโดยการสั่ง run ภายในเมธอด run จะเป็นการเรียกให้ msgListener ทำงานโดยไปเรียกเมธอด messageRecieved ที่อยู่ใน DummyAgent เพื่อทำงานกับข้อความที่ได้รับต่อไป



ภาพประกอบ 4.19 กิจกรรมที่เกิดขึ้นสำหรับคลาส RecieveMessageThread

4.2.1.8 การออกแบบ MessageListener

การออกแบบอินเตอร์เฟส (Interface) MessageListener เพื่อต้องการใช้เอเจนท์ที่ทำงานสามารถทำงานได้ต่อไปโดยไม่ต้องรอให้มีการทำงานในแต่ละข้อความเสร็จสมบูรณ์โดยหน้าที่ของ MessageListener คือเมื่อมีข้อความเข้ามาก็จะแจ้งไปยังเอเจนท์เพื่อให้เอเจนท์สร้างเธรดสำหรับทำงานกับข้อความนั้นๆ แล้วกลับไปรอรับข้อความใหม่ได้ทันที



ภาพประกอบ 4.20 แผนภาพอินเตอร์เฟส MessageListener

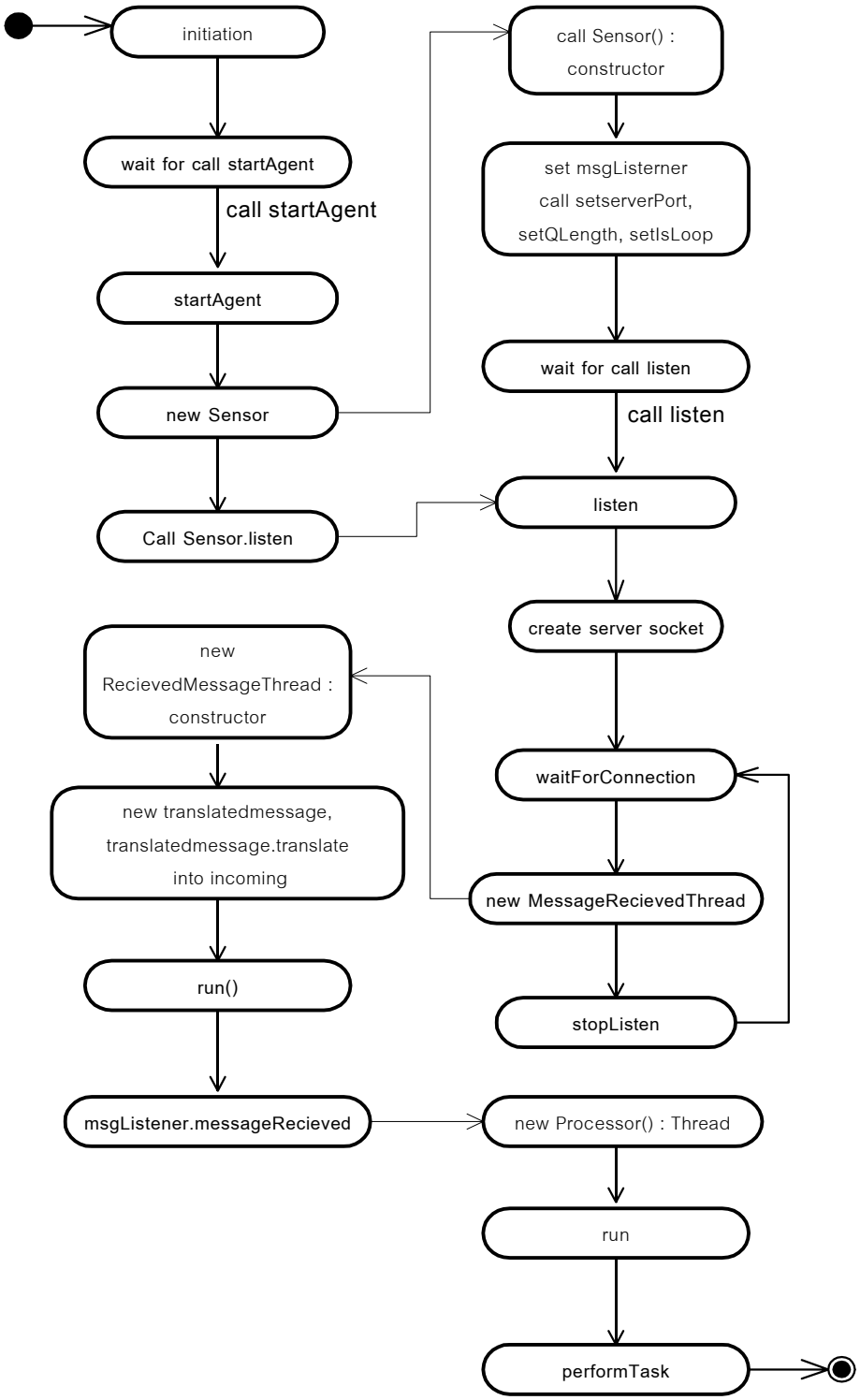
ในภาพประกอบ 4.20 แสดงแผนภาพอินเตอร์เฟส MessageListener ซึ่งสามารถอธิบายความหมายและหน้าที่การทำงานต่างๆ ได้ดังรายละเอียดในตาราง 4.8

ตาราง 4.8 แสดงการอธิบายตัวแปรและเมธอดของอินเตอร์เฟส MessageListener

เมธอด (Method)	คำอธิบาย
+ messageRecieved	ตัวเมธอดที่ทำงานจริงๆ จะอยู่ที่คลาส DummyAgent ซึ่งทำหน้าที่ในการเรียกเมธอดที่ทำงานต่างๆ เมื่อได้รับข้อความจากคลาส Sensor มาแล้ว

ภายในคลาส DummyAgent เมธอด messageRecieved ทำหน้าที่ในการเรียก Processor ขึ้นมาทำงาน

เมื่อออกแบบส่วนประกอบทั้งหมดเรียบร้อยแล้ว ความสัมพันธ์ระหว่างส่วนประกอบต่างๆ ของเอเจนท์พื้นฐานดังแสดงในภาพประกอบ 4.21 ในรูปแสดงกิจกรรมเมื่อเริ่มต้นทำงานของเอเจนท์และได้รับการติดต่อและมีการส่งข้อความ



ภาพประกอบ 4.21 กิจกรรมที่เกิดขึ้นสำหรับเอเจนท์เมื่อเริ่มทำงานและได้รับข้อความร้องขอบริการ

4.2.2 การออกแบบเอเจนท์ที่ทำหน้าที่เฉพาะต่าง ๆ ใน SAFWA

เมื่อได้เอเจนท์พื้นฐานที่สามารถทำงานในส่วนของการติดต่อสื่อสารโดยการรับส่งข้อความได้แล้ว จากนั้นนำเอาเอเจนท์พื้นฐานมาเป็นต้นแบบสำหรับการออกแบบเอเจนท์อื่นๆ ที่เป็นส่วนประกอบสำหรับ SAFWA

4.2.2.1 การออกแบบเอเจนท์โดยการถ่ายทอดคุณสมบัติจาก Dummy agent

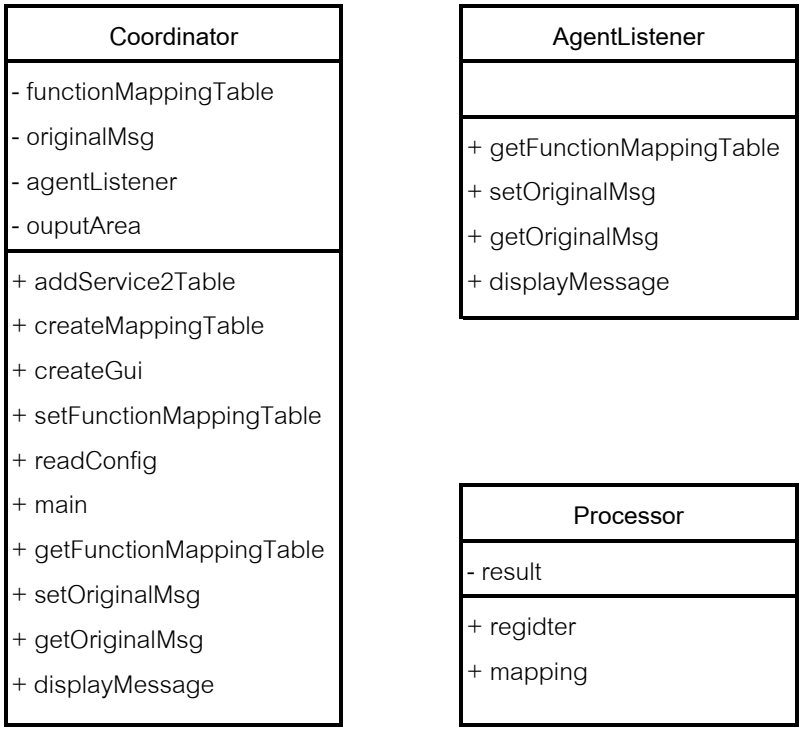
การออกแบบเอเจนท์อาศัยการถ่ายทอดคุณสมบัติจาก Dummy agent ในส่วนของการติดต่อรับส่งข้อมูล ส่วนหลักการในการออกแบบเอเจนท์ใหม่ต้องทำการออกแบบทั้งหมด 3 ส่วนด้วยกันคือ คลาสที่เป็นตัวเอเจนท์ คลาส Processor และ อินเตอร์เฟส AgentListener โดยที่คลาสเอเจนท์ถ่ายทอดคุณสมบัติจาก DummyAgent และทำการเพิ่มเติมฟังก์ชันที่ต้องการใช้งานลงไปในคลาส Processor ส่วนอินเตอร์เฟส AgentListener ทำหน้าที่ในการติดต่อระหว่างคลาสเอเจนท์และคลาส Processor เพื่อใช้ในการส่งค่าตัวแปรหรือทำหน้าที่อื่นๆ ตามที่ต้องการ

4.2.2.2 การออกแบบ Coordinator agent

หน้าของ Coordinator agent ใน SAFWA คือ การจับคู่ผู้ให้บริการกับผู้ร้องขอ ความหมายคือเมื่อมีการร้องขอบริการผู้ร้องขอได้ส่งชื่อของฟังก์ชันที่ต้องการมายัง Coordinator agent เมื่อได้รับข้อความร้องขอบริการแล้ว Coordinator agent จะทำการค้นหาชื่อฟังก์ชันในตารางข้อมูล เมื่อค้นเจอสิ่งที่ได้ Coordinator ได้คือ เอเจน์ผู้ที่ให้บริการ ทำให้สามารถส่งข้อความร้องขอบริการต่อไปยังเอเจนท์ผู้ให้บริการที่ตรงกันได้

การสร้างตารางข้อมูลเพื่อจับคู่ระหว่างฟังก์ชันกับเอเจนท์ผู้ให้บริการนั้นจะถูกสร้างเมื่อเอเจนท์ผู้ให้บริการส่งข้อความลงทะเบียน มายัง Coordinator agent เมื่อได้รับข้อความลงทะเบียน Coordinator agent จะทำการเพิ่มข้อมูลเข้าไปยังตารางข้อมูล ดังนั้นเมื่อมีผู้ให้บริการใหม่ ๆ Coordinator agent ก็จะทำการปรับปรุงตารางข้อมูลเมื่อได้รับข้อความร้องขอการลงทะเบียน

ภาพประกอบ 4.22 แสดงแผนภาพคลาส Coordinator ซึ่งแทน Coordinator agent แสดงแผนภาพคลาส Processor ที่ทำการเพิ่มฟังก์ชันในการทำงานแล้ว และแสดงแผนภาพอินเตอร์เฟส AgentListener โดยที่หน้าที่และความหมายของตัวแปรและเมธอดรายละเอียดในตาราง 4.9



ภาพประกอบ 4.22 แผนภาพคลาส Coordinator, Processor
และแผนภาพอินเตอร์เฟส AgentListener

ตาราง 4.9 แสดงการอธิบายตัวแปรและเมธอดของ Coordinator, Processor

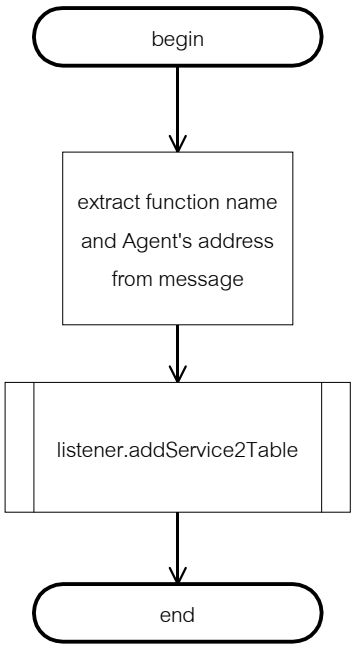
Coordinator	
ตัวแปร (Data member)	**คำอธิบาย**
- functionMappingTable	ใช้สำหรับเก็บข้อมูลคู่ลำดับระหว่างฟังก์ชันกับผู้ให้บริการ
- originalMsg	เก็บค่าเริ่มต้นของเอเจนท์เช่น ชื่อเอเจนท์เป็นต้น
- agentListener	อินเตอร์เฟสสำหรับติดต่อระหว่าง Coordiantor และ Processor
- ouputArea	ใช้สำหรับแสดงการทำงานของเอเจนท์
+ addService2Table	ทำการแทรกข้อมูลเข้าไปในตารางข้อมูลคู่ลำดับ
+ createMappingTable	ใช้สำหรับสร้างตารางคู่ลำดับ
เมธอด (Method)	**คำอธิบาย**
+ createGui	สำหรับสร้าง Windows เพื่อแสดงการทำงาน
+ setFunctionMappingTable	ตั้งค่า functionMappingTable

ตาราง 4.9 (ต่อ)

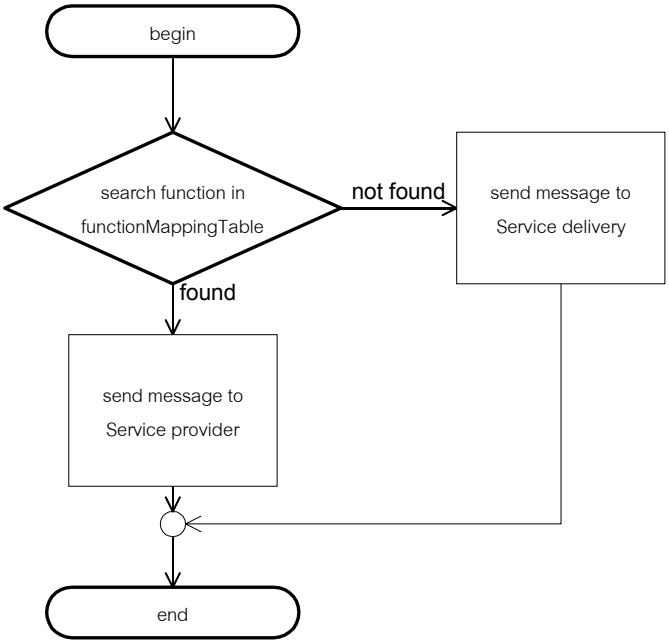
เมธอด (Method)	**คำอธิบาย**
+ readConfig	ใช้สำหรับการอ่านข้อมูลเริ่มต้นของเอเจนท์จากแฟ้มที่เก็บไว้ ข้อมูลเช่น ชื่อเอเจนท์ เป็นต้น
+ main	เรียกเมธอดสำหรับสั่งให้เอเจนท์ทำงาน
+ getFunctionMappingTable	ส่งค่า functionMappingTable
+ setOriginalMsg	ตั้งค่า originalMsg
+ getOriginalMsg	ส่งค่า originalMsg
+ displayMessage	สำหรับแสดงข้อความที่ต้องการบนวินโดว์แสดงการทำงาน
Processor	
ตัวแปร (Data member)	**คำอธิบาย**
- result	ผลลัพธ์ที่เกิดจากการทำงานของฟังก์ชัน
เมธอด (Method)	**คำอธิบาย**
+ register	ทำการลงทะเบียนสำหรับผู้ให้บริการ
+ mapping	เลือกผู้ให้บริการ โดยค้นหาจากชื่อฟังก์ชันที่ได้มาจากข้อความร้องขอบริการ สิ่งที่ได้คือชื่อและที่อยู่ของผู้ให้บริการ

จากตาราง 4.9 มีการอธิบายสำหรับ Coordinator และ Processor เท่านั้น เนื่องจาก เมธอดสำหรับอินเตอร์เฟสนั้นจะถูกเขียนอยู่ใน Coordinator เท่านั้น

กิจกรรมหลักที่เกิดขึ้นสำหรับ Coordinator ก็คือ การลงทะเบียน (register) และการค้นหาผู้ให้บริการ (mapping) โดยขั้นตอนที่เกิดขึ้นสำหรับการลงทะเบียนคือ เมื่อได้รับข้อความที่เป็นชนิดขอลงทะเบียน ทำการดึงข้อมูลต่างๆ เช่นชื่อฟังก์ชันและที่อยู่ของผู้ให้บริการออกจากข้อความร้องขอบริการ จากนั้นนำข้อมูลที่ได้ไปเพิ่มในตารางข้อมูลโดยการเรียกเมธอด addService2Table ซึ่งแสดงในภาพประกอบ 4.23 และขั้นตอนสำหรับการค้นหาผู้ให้บริการ คือ เมื่อได้รับข้อความร้องขอบริการจาก Service delivery agent ทำการดึงข้อมูลที่เป็นชื่อฟังก์ชันออกจากข้อความร้องขอ จากนั้นทำการค้นหาชื่อฟังก์ชันในตาราง หากพบฟังก์ชันที่ต้องการก็จะทำการส่งข้อความร้องขอบริการไปยังเอเจนท์ผู้ให้บริการที่ตรงกับฟังก์ชันนั้น ถ้าไม่พบฟังก์ชันก็จะทำการส่งข้อความกลับไปยัง Service delivery agent เพื่อบอกว่าไม่สามารถให้บริการที่ร้องขอ ซึ่งแสดงในภาพประกอบ 4.24



ภาพประกอบ 4.23 ขั้นตอนในการลงทะเบียนสำหรับผู้ให้บริการ

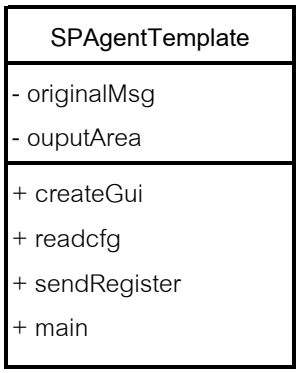


ภาพประกอบ 4.24 ขั้นตอนในค้นหาผู้ให้บริการ

4.2.2.3 การออกแบบ Service provider agent

การออกแบบ Service provider agent ต้องคำนึงถึงโปรแกรมประยุกต์ที่พัฒนาเป็นหลัก แต่การออกแบบในส่วนของ SAFWA จะออกแบบเฉพาะส่วนโครงสร้างเพื่อนำมาเป็นต้นแบบในการสร้างเท่านั้น โดยการแทนเอเจนท์ด้วยคลาส SPAgentTemplate เมื่อต้องการสร้าง เอเจนท์ที่ให้บริการเฉพาะอย่างก็สามารถนำเอาต้นแบบมาทำการปรับปรุงด้วยการเพิ่มฟังก์ชันการทำงานเข้าไปในคลาส Processor เท่านั้น โดยแสดงแผนภาพของคลาส SPAgentTemplate ดังภาพประกอบ 4.25 และคำอธิบายตัวแปรและหน้าที่ของเมธอดดังรายละเอียดในตาราง 4.10

ขั้นตอนแรกของการเริ่มต้นการทำงานของ Service provider agent คือ การลงทะเบียน ด้วยวิธีการส่งข้อความขอลงทะเบียนไปยัง Coordinator agent เพื่อบอกว่าตนเองนั้นให้บริการอะไรบ้าง



ภาพประกอบ 4.25 แผนภาพคลาส SPAgentTemplate

ตาราง 4.10 แสดงการอธิบายตัวแปรและเมธอดของ SPAgentTemplate

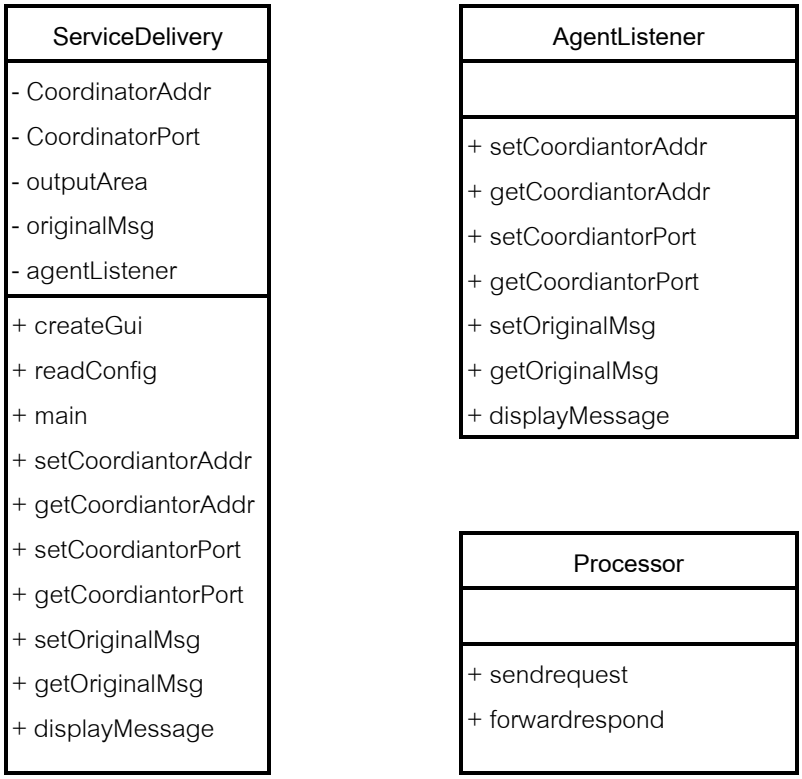
ตัวแปร (Data member)	**คำอธิบาย**
- originalMsg	เก็บค่าเริ่มต้นของเอเจนท์เช่น ชื่อเอเจนท์เป็นต้น
- ouputArea	ใช้สำหรับแสดงการทำงานของเอเจนท์
เมธอด (Method)	**คำอธิบาย**
+ createGui	สำหรับสร้าง Windows เพื่อแสดงการทำงาน
+ readConfig	ใช้สำหรับการอ่านข้อมูลเริ่มต้นของเอเจนท์จากแฟ้มที่เก็บไว้ ข้อมูลเช่น ชื่อเอเจนท์ เป็นต้น

ตาราง 4.10 (ต่อ)

เมธอด (Method)	คำอธิบาย
+ main	เรียกเมธอดสำหรับสั่งให้เอเจนท์ทำงาน
+ sendRegister	ทำการส่งข้อความขอลงทะเบียนไปยัง Coordinator agent โดยส่งข้อมูลฟังก์ชันที่ให้บริการ และ Address ของเอเจนท์

4.2.2.4 การออกแบบ Service delivery agent

หน้าที่สำคัญของ Service delivery agent คือการรับข้อความร้องขอบริการจาก Web interface agent เพื่อส่งต่อไปยัง Coordinator agent และอีกหน้าที่หนึ่งคือ การรับข้อความผลลัพธ์จากการบริการของ Service provider agent เพื่อส่งต่อไปแสดงผลลัพธ์ที่ Web interface agent โดยแสดงแผนภาพของคลาส ServiceDelivery ซึ่งแทน Service delivery agent ในภาพประอบ 4.26 และอธิบายความหมายของตัวแปรและเมธอดดังรายละเอียดในตาราง 4.11



ภาพประกอบ 4.26 แผนภาพคลาส ServiceDelivery, Processor และแผนภาพอินเตอร์เฟส AgentListener

ตาราง 4.11 แสดงการอธิบายตัวแปรและเมธอดของ ServiceDelivery, Processor

ServiceDelivery	
ตัวแปร (Data member)	**คำอธิบาย**
- coordinatorAddr	ค่า IP address ของเครื่องที่ Coordinator agent ทำงานอยู่
- coordinatorPort	ค่า Port ที่ Coordinator agent เปิดรับข้อความ
- originalMsg	เก็บค่าเริ่มต้นของเอเจนท์เช่น ชื่อเอเจนท์เป็นต้น
- agentListener	อินเตอร์เฟสสำหรับติดต่อระหว่าง ServiceDelivery และ Processor
ตัวแปร (Data member)	**คำอธิบาย**
- ouputArea	ใช้สำหรับแสดงการทำงานของเอเจนท์
เมธอด (Method)	**คำอธิบาย**
+ createGui	สำหรับสร้าง Windows เพื่อแสดงการทำงาน
+ readConfig	ใช้สำหรับการอ่านข้อมูลเริ่มต้นของเอเจนท์จากแฟ้มที่เก็บไว้ ข้อมูลเช่น ชื่อเอเจนท์ เป็นต้น
+ main	เรียกเมธอดสำหรับสั่งให้เอเจนท์ทำงาน
+ setCoordinatorAddr	ตั้งค่า CoordinatorAddr
+ getCoordinatorAddr	ส่งค่า CoordinatorAddr
+ setCoordinatorPort	ตั้งค่า CoordinatorPort
+ getCoordinatorPort	ส่งค่า CoordinatorPort
+ setOriginalMsg	ตั้งค่า originalMsg
+ getOriginalMsg	ส่งค่า originalMsg
+ displayMessage	สำหรับแสดงข้อความที่ต้องการบนวินโดว์แสดงการทำงาน
Processor	
เมธอด (Method)	**คำอธิบาย**
+ sendrequest	ทำการรับข้อความร้องขอบริการจาก Web interface agent จากนั้นก็ทำการ ส่งข้อความต่อไปยัง Coordinator agent
+ forwardresponse	รับข้อความผลลัพธ์จาก Service provider agent แล้วทำการส่งต่อไปยัง Web interface agent

4.2.2.5 การออกแบบ Web interface agent

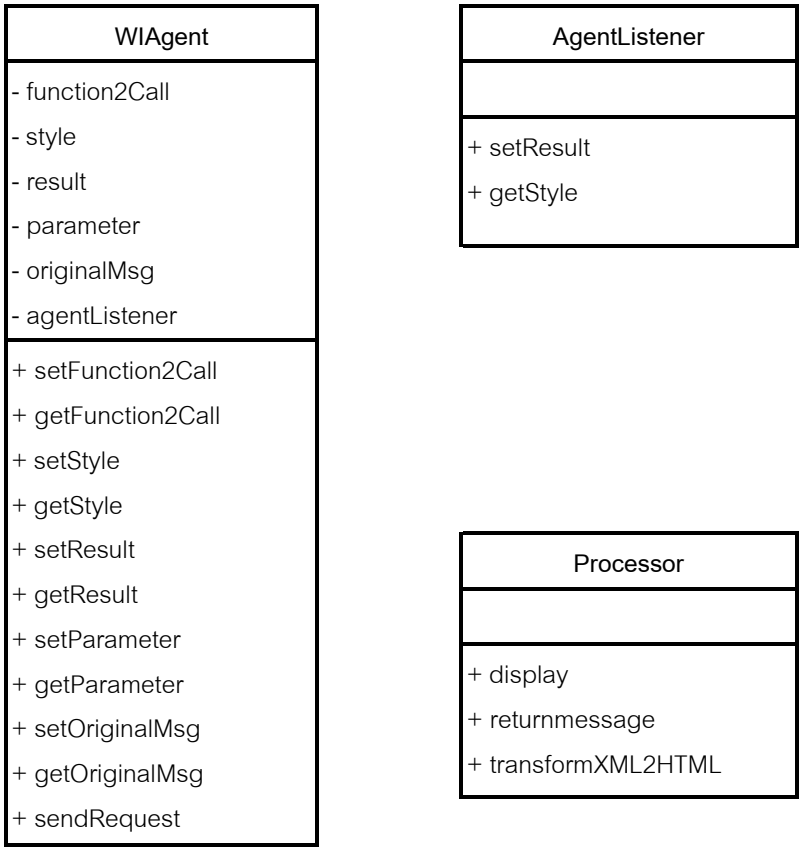
สำหรับการออกแบบ Web interface agent จะแตกต่างจากเอเจนท์อื่น เนื่องจากเอเจนท์จะต้องทำงานร่วมกับการทำงานของเว็บเซิร์ฟเวอร์เพื่อที่จะสามารถให้บริการผ่านทางเว็บได้ ดังนั้นจึงทำการออกแบบ Web interface agent ให้เป็นคลาสในรูปแบบของ Java Bean และใช้การพัฒนาร่วมกับการเขียนโปรแกรมโดยอาศัยเทคโนโลยี JSP (Java Server Page) โดยหลักการทำงานของ Web interface agent มีดังนี้

ในแฟ้มเอกสาร .jsp ที่ต้องการใช้บริการของ SAFWA จะต้องทำการเรียกใช้ชิ้นส่วนที่ชื่อ WIAgent ซึ่งเป็นแทน Web interface agent ทำการส่งพารามิเตอร์ซึ่งประกอบด้วยฟังก์ชันที่ต้องการ พารามิเตอร์ของฟังก์ชันนั้น และระบุรูปแบบของการแสดงผลซึ่งอยู่ในรูปของแฟ้ม .xsl ซึ่งอาศัยเทคโนโลยี XSLT (Extensible Stylesheet Language for Translation) โดยที่เทคโนโลยีนี้ช่วยให้สามารถแปลงเอกสารรูปแบบ XML ซึ่งก็คือผลลัพธ์ที่ส่งมากับข้อความให้สามารถแสดงผลในรูปแบบของ HTML (Hyper-Text Markup Language) ได้ การเรียกใช้มีรูปแบบดังนี้คือ

```
1 <jsp:useBean id="test2" scope="page" class="safieaweb.WIAgent"/>
2 <jsp:setProperty name="test2" property="function2Call" value = "test"/>
3 <jsp:setProperty name="test2" property="style" value = "test4.xsl"/>
4 <jsp:setProperty name="test2" property="parameter" value = "number1=1;number2=2;"/>
5 <jsp:getProperty name="test2" property="result" />
```

บรรทัดแรกต้องทำการสร้างตัวแปรรหัสหมายเลขสำหรับคลาสขึ้นมาก่อนมีชื่อว่า test2 โดยระบุชื่อคลาสคือ safieaweb.WIAgent จากนั้นทำการเรียกฟังก์ชันที่ต้องการในบรรทัดที่สอง นั้นคือให้ค่าตัวแปร function2Call เป็น test จากนั้นทำการระบุแบบของการแสดงผลซึ่งอยู่ในแฟ้ม test4.xsl ในบรรทัดที่ 4 ทำการใส่ค่าพารามิเตอร์สำหรับฟังก์ชัน test จากนั้นให้ดึงค่า result ซึ่งค่า result เป็นผลลัพธ์ที่ได้มาจากการทำงานของฟังก์ชัน test ดังนั้นเมื่อต้องการใช้งานฟังก์ชันอื่นเพียงแค่แทรกคำสั่ง 5 บรรทัดนี้และเปลี่ยนชื่อฟังก์ชันและพารามิเตอร์ต่าง ๆ ก็สามารถเรียกใช้งานฟังก์ชันหรือบริการต่างๆ ที่มีใน Service provider agent ได้

ในภาพประกอบ 4.27 แสดงแผนภาพของคลาส WIAgent ซึ่งแทน Web interface agent และอธิบายความหมายของตัวแปรและเมธอดดังรายละเอียดในตาราง 4.12



ภาพประกอบ 4.27 แผนภาพคลาส WIAgent, Processor
และแผนภาพอินเตอร์เฟส AgentListener

ตารางที่ 4.12 แสดงการอธิบายตัวแปรและเมธอดของ WIAgent, Processor

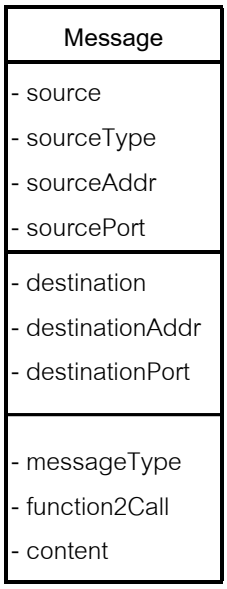
WIAgent	
ตัวแปร (Data member)	**คำอธิบาย**
- function2Call	ชื่อฟังก์ชันที่ต้องการให้ทำงาน
- style	รูปแบบการแสดงผล เก็บในแฟ้มนามสกุล .xsl
- result	ผลลัพธ์ที่ได้จากการทำงานของฟังก์ชัน
- parameter	พารามิเตอร์ต่างๆ ของฟังก์ชันที่ต้องการเรียกใช้
- originalMsg	เก็บค่าเริ่มต้นของเอเจนท์เช่น ชื่อเอเจนท์เป็นต้น
- agentListener	อินเตอร์เฟสสำหรับติดต่อระหว่าง WIAgent และ Processor
เมธอด (Method)	**คำอธิบาย**
+ setFunction2Call	ตั้งค่า function2Call

ตาราง 4.12 (ต่อ)

เมธอด (Method)	คำอธิบาย
+ getFunction2Call	ส่งค่า function2Call
+ setStyle	ตั้งค่า style
+ getStyle	ส่งค่า style
+ setResult	ตั้งค่า result
+ getResult	ส่งค่า result
+ setParameter	ตั้งค่า parameter
+ getParameter	ส่งค่า parameter
+ setOriginalMsg	ตั้งค่า originalMsg
+ getOriginalMsg	ส่งค่า originalMsg
+ sendrequest	ส่งข้อความร้องขอบริการไปยัง Service delivery agent
Processor	
เมธอด (Method)	**คำอธิบาย**
+ display	ส่งผลลัพธ์ที่อยู่ในรูปแบบของ HTML ไปยังหน้าเว็บเพื่อแสดงผล
+ returnmessage	ส่งผลลัพธ์ในรูปของข้อความเพื่อนำไปใช้งาน
+ transformXML2HTML	ทำการแปลงเอกสารในรูปแบบ XML ที่ส่งมาพร้อมข้อความให้อยู่ในรูปแบบ HTML โดยอาศัยการกำหนดรูปแบบการแสดงผลที่อยู่ในแฟ้ม ,xsl ซึ่งระบุมาใน style

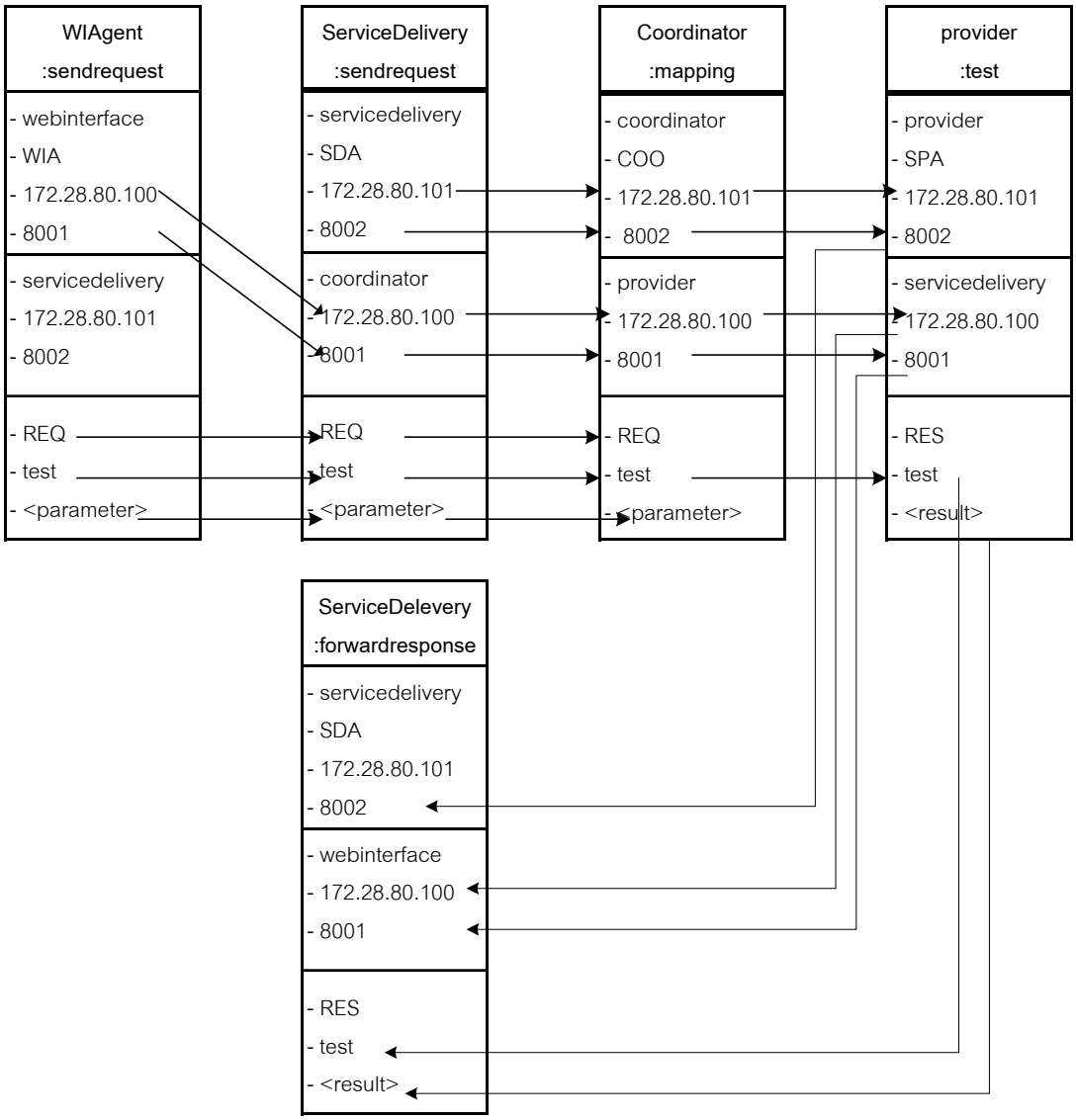
4.2.3 การออกแบบโครงสร้างข้อความสำหรับการติดต่อสื่อสารระหว่างเอเจนท์

การติดต่อสื่อสารระหว่างเอเจนท์อาศัยการส่งและรับข้อความโดยที่ข้อความที่ใช้ในการติดต่อสื่อสารนั้น มีโครงสร้างแบบเดียวกันแต่การกำหนดค่าของแต่ละส่วนในข้อความจะทำให้เกิดความแตกต่างระหว่างข้อความที่ส่งและข้อความที่ได้รับในแต่ละเอเจนท์และต่างกันสำหรับการส่งข้อความระหว่างเอเจนท์ที่ต่างกัน อย่างเช่น การส่งข้อความจาก Web interface agent ไปยัง Service delivery agent กับ การส่งข้อความระหว่าง Service delivery agent ไปยัง Coordinator agent มีโครงสร้างเหมือนกัน แต่มีเนื้อหาของข้อความที่ต่างกัน ในภาพประกอบ 4.28 แสดงโครงสร้างของข้อความที่ใช้ในการติดต่อสื่อสารระหว่างเอเจนท์



ภาพประกอบ 4.28 โครงสร้างของข้อความที่ใช้ในการติดต่อสื่อสาร

จากภาพประกอบ 4.28 สามารถอ่านคำอธิบายความหมายของตัวแปรได้จาก ตารางที่ 4.5 โดยข้อความที่ใช้นี้ถูกสร้างขึ้นให้อยู่ในรูปแบบเอกสาร XML ซึ่งเป็นมาตรฐานสำหรับการรับส่งข้อมูลบนเครือข่ายอินเตอร์เน็ต เอเจนท์แต่ละตัวสามารถกำหนดค่าต่าง ๆ สำหรับข้อความตามที่ต้องการได้ และค่าต่างๆ นั้นเอเจนท์ที่ติดต่อสื่อสารด้วยต้องสามารถเข้าใจได้ตรงกัน ดังนั้นจึงมีการกำหนดรูปแบบของข้อความที่ใช้ในการติดต่อกันภายใน SAFWA ดังแสดงในภาพประกอบ 4.28



ภาพประกอบ 4.29 การกำหนดรูปแบบของข้อความที่ใช้ติดต่อระหว่างเอเจนท์ใน SAFWA

จากภาพประกอบ 4.29 แสดงถึงการกำหนดค่าต่างๆ สำหรับข้อความที่ใช้ในการติดต่อสื่อสารระหว่างเอเจนท์ใน SAFWA ข้อความแต่ละข้อความจะถูกสร้างก่อนที่จะส่งไปยังเอเจนท์ตัวอื่น สำหรับภาพประกอบ 4.29 เป็นตัวอย่างในการขอใช้บริการฟังก์ชัน test กับผู้ให้บริการที่ชื่อ provider โดยที่ WIAgent ทำการส่งข้อความร้องขอบริการ โดย กำหนด source ให้เก็บค่าสำหรับติดต่อกับ WIAgent เมื่อ ServiceDelivery ได้รับข้อความก็จะทำการส่งข้อความต่อแต่เปลี่ยนเอา ค่า source ที่ได้มา ไปเป็น destination เพื่อกำหนดเส้นทางติดต่อ และกำหนด

source ให้เป็นช่องทางติดต่อสำหรับ ServiceDelivery เอง จากนั้นทำการส่งค่าต่อไปยัง Coordinator เพื่อทำการ mapping แล้วส่งข้อความต่อไปยัง provider เพื่อทำงานฟังก์ชัน test เมื่อทำงานเสร็จก็จะนำเอาผลลัพธ์ใส่เข้าไปในข้อความ และส่งข้อความกลับไปยัง ServiceDelivery โดยใช้ช่องทางติดต่อที่มากับ source ของข้อความ เมื่อ ServiceDelivery ได้รับข้อความแล้วก็ทำการส่งข้อความผลลัพธ์ต่อไปยัง WIAgent ด้วยช่องทางติดต่อที่ได้จาก destination เพื่อทำการแสดงผลลัพธ์ต่อไป

การกำหนดรูปแบบของข้อความสามารถเปลี่ยนแปลงได้ตามความต้องการ ซึ่งขึ้นอยู่กับการเปลี่ยนแปลงที่จะเกิดขึ้นกับ SAFWA ดังนั้นไม่จำเป็นต้องเปลี่ยนแปลงโครงสร้างข้อความ แต่สามารถเปลี่ยนแปลงรูปแบบการกำหนดข้อความเพื่อให้ SAFWA สามารถทำงานตามที่ต้องการได้

4.3 การออกแบบการติดตั้ง SAFWA บนระบบงานจริง

การติดตั้ง SAFWA บนระบบงานจริง สามารถที่จะติดตั้งได้หลายแบบตามต้องการเนื่องจากเอเจนท์แต่ละตัวใน SAFWA ติดต่อสื่อสารกันผ่านเครือข่ายอินเตอร์เน็ตซึ่งเป็นมาตรฐาน ดังนั้นการออกแบบการติดตั้งจึงขึ้นอยู่กับการใช้งานโปรแกรมประยุกต์ที่พัฒนาบน SAFWA ว่าต้องการประสิทธิภาพหรือการทำงานอย่างไร ดังนั้นในงานวิจัยนี้จึงเป็นเพียงแค่ตัวอย่างหนึ่งของการออกแบบการติดตั้งเท่านั้น

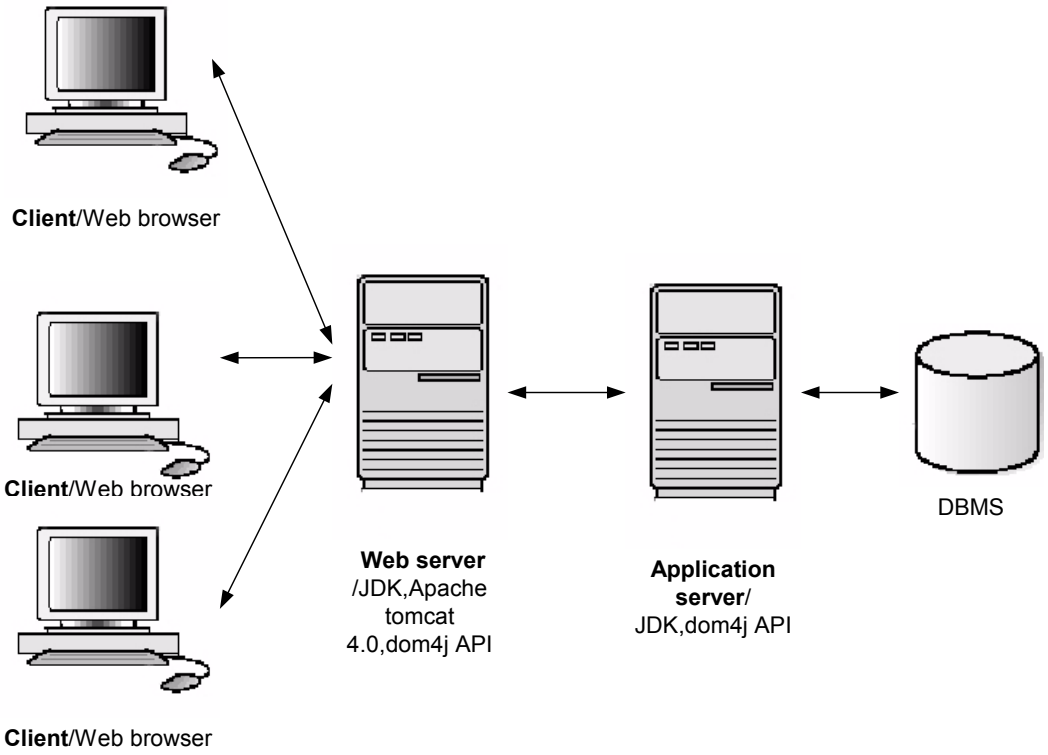
ระบบที่นำมาติดตั้ง SAFWA ประกอบไปด้วยเครื่องคอมพิวเตอร์ลูกข่ายสำหรับการแสดงผลทางเว็บ (Client) เครื่องคอมพิวเตอร์ทีทำหน้าที่เป็นเว็บเซิร์ฟเวอร์ (Web server) เครื่องคอมพิวเตอร์ที่บริการโปรแกรมประยุกต์ (Application server) และ เครื่องที่ติดตั้งระบบการจัดการฐานข้อมูล (Database Server) ดังแสดงในภาพประกอบ 4.30

โดยที่เครื่องคอมพิวเตอร์แต่ละตัวต้องมีการติดตั้งซอฟต์แวร์ดังนี้คือ

Client	ติดตั้ง Web browser สำหรับการใช้งานโปรแกรมประยุกต์บนเว็บ
Web server	ติดตั้ง โปรแกรมเว็บเซิร์ฟเวอร์ คือ Apache Tomcat 4.0 ติดตั้ง JVM (Java Virtual machine) สามารถติดตั้งพร้อมกับ JDK (Java Developer tool kit) ได้ สำหรับให้โปรแกรมจาวาสามารถทำงานได้ พร้อมทั้งติดตั้ง dom4j API เพื่อให้โปรแกรมสามารถจัดการกับเอกสาร XML ได้

Application server ติดตั้ง JVM (Java Virtual machine) และ dom4j API

Database server สามารถเลือกติดตั้ง DBMS (Database Management System) ตัวใดตัวหนึ่งที่มีในท้องตลาดได้ตามต้องการ



ภาพประกอบ 4.30 ตัวอย่างรูปแบบระบบคอมพิวเตอร์สำหรับติดตั้ง SAFWA

ทำการติดตั้ง SAFWA สามารถกระทำได้ดังนี้

Web server ทำการติดตั้ง WIAgent และ ServiceDelevery

Application server ทำการติดตั้ง Coordinator และ Service provider

หลังจากทำการติดตั้ง SAFWA เรียบร้อยแล้ว ก็สามารถที่จะพัฒนาโปรแกรมประยุกต์บน SAFWA ในบทที่ 5 จะเป็นตัวอย่างสำหรับการพัฒนาโปรแกรมประยุกต์โดยอาศัย SAFWA เพื่อสนับสนุนแนวคิดในการพัฒนา SAFWA และเป็นการทดสอบการทำงานของ SAFWA ด้วย