การออกแบบและพัฒนาบริการค้นหาบริการสำหรับระบบกระจาย

นาย วรวุฒิ ศุภสัณฐิติกุล

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2543
ISBN 974-13-0527-3

A DESIGN AND DEVELOPMENT OF A SERVICE DISCOVERY SERVICE FOR DISTRIBUTED SYSTEMS

Mr. Worawut Suphasanthitikul

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering in Computer Engineering
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2000

ISBN 974-13-0527-3

หัวข้อวิทยานิพนธ์	การออกแบบและพัฒนาบริการค้นหาบริการสำหรับระบบกระจาย
โดย	นาย วรวุฒิ ศุภสัณฐิติกุล
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษา	อาจารย์ ดร. ทวิตีย์ เสนีวงศ์ ณ อยุธยา

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

…….. คณบดีคณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร.สมศักดิ์ ปัญญาแก้ว)

คณะกรรมการสอบวิทยานิพนธ์

…… ประธานกรรมการ
(อาจารย์ ดร.ยรรยง เต็งอำนวย)

……... อาจารย์ที่ปรึกษา
(อาจารย์ ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา)

……. กรรมการ
(อาจารย์ ดร.ณัฐวุฒิ หนูไพโรจน์)

……. กรรมการ
(อาจารย์ ดร. ธันวดี สุเนตนันท์)

วรวุฒิ ศุภสัณฐิติกุล : การออกแบบและพัฒนาบริการค้นหาบริการสำหรับระบบกระจาย.
(A DESIGN AND DEVELOPMENT OF A SERVICE DISCOVERY SERVICE FOR DISTRIBUTED SYSTEMS) อ. ที่ปรึกษา : อ.ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา, 121 หน้า. ISBN 974-13-0527-3.

งานวิจัยนี้ได้ทำการออกแบบ และพัฒนาบริการค้นหาบริการ หรือเอสดีเอส เพื่อเพิ่มความสามารถในการค้นหาบริการและแก้ไขข้อจำกัดของการนำเสนอข้อมูลของบริการเทรดเดอร์ในระบบกระจายคอร์บาซึ่งอนุญาตให้ผู้ใช้สามารถค้นหาข้อมูลบริการได้เท่านั้น ทั้งนี้บริการเอสดีเอสจะมีความสามารถเพิ่มเติมโดยอนุญาตให้ผู้ใช้สามารถเลือกค้นหาได้ทั้งข้อมูลชนิดของบริการ ข้อมูลส่วนต่อประ สาน และข้อมูลบริการได้ การค้นหากระทำกับข้อมูลบริการในรูปแบบเอกสารเอ็กซ์เอ็มแอลที่รวบรวมจากบริการเทรดเดอร์หลายๆ แหล่ง และยังสามารถรองรับการค้นหาด้วยภาษาค้นหาหลายรูปแบบ ได้แก่ เอ็กซ์คิวแอล และการค้นหาด้วยคำสำคัญ การค้นหาบริการสามารถรองรับได้ทั้งผู้ใช้คอร์บาและผู้ใช้ผ่านเว็บ นอกจากนี้ยังสามารถเพิ่มขยายการค้นหาด้วยการเชื่อมโยงบริการเอสดีเอสเข้าด้วยกันได้

ภาควิชา	วิศวกรรมคอมพิวเตอร์	ลายมือชื่อนิสิต.....
สาขาวิชา	วิศวกรรมคอมพิวเตอร์	ลายมือชื่ออาจารย์ที่ปรึกษา.....
ปีการศึกษา	2543	ลายมือชื่ออาจารย์ที่ปรึกษาร่วม.....

4270521421 : MAJOR COMPUTER ENGINEERING
KEY WORD: DISCOVERY SERVICE / CORBA / TRADER / XML
WORAWUT SUPHASANTHITIKUL : A DESIGN AND DEVELOPMENT OF A SERVICE DISCOVERY SERVICE FOR DISTRIBUTED SYSTEMS.
THESIS ADVISOR : TWITTIE SENIVONGSE, Ph.D., 121 pp. ISBN 974-13-0527-3.

This research proposes a design and development of a Service Discovery Service (SDS) that enhances service search and resolves a limitation of traders in CORBA distributed systems which allows only search for service offers. SDS users can perform search on service type descriptions, interface definitions, and service offer descriptions, and search is conducted on the XML version of service descriptions gathered from multiple traders. Various kinds of query languages are supported such as XQL and search keywords. An SDS can be accessible by both CORBA users and Web users with a possibility to enlarge search space by the federation of multiple SDSes.

Department	Computer Engineering	Student's signature.....
Field of study	Computer Engineering	Advisor's signature.....
Academic year	2000	Co-advisor's signature.....

กิตติกรรมประกาศ

ขอขอบคุณ อาจารย์ ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา ที่ให้ความกรุณาช่วยเหลือเป็นอย่างมากในการทำงานวิจัยนี้ และสละเวลาให้คำปรึกษาและคำแนะนำที่ดีเสมอมา

ขอขอบคุณ อาจารย์ ดร.ยรรยง เต็งอำนวย อาจารย์ ดร. ณัฐวุฒิ หนูไพโรจน์ และอาจารย์ ดร. ธันวดี สุเนตนันท์ กรรมการวิทยานิพนธ์ ที่กรุณาให้คำแนะนำและตรวจสอบแก้ไขต้นฉบับวิทยานิพนธ์นี้

ขอขอบคุณกำลังใจทุกดวงที่มีให้ ทั้งพี่ๆ เพื่อนๆ และน้องๆ ทั้งหลาย ที่ให้ทั้งคำปรึกษาและความบันเทิง

สุดท้ายนี้ ขอขอบพระคุณ บิดา มารดา และครอบครัว ที่ให้การสนับสนุนในด้านต่างๆ และให้กำลังใจที่ดีเสมอมา

วรวุฒิ ศุภสัณฐิติกุล

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญภาพ

ภาพประกอบ หน้า

สารบัญภาพ (ต่อ)

บทที่ หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

คอร์บา (Common Object Request Broker Architecture (CORBA)) [1] เป็นสถาปัตยกรรมหนึ่งของระบบกระจาย (Distributed System) ที่กำหนดโดยโอเอ็มจี (Object Management Group (OMG)) คอร์บาถูกออกแบบมาเพื่อการทำงานร่วมกันของระบบที่มีความแตกต่างกันในระบบกระจาย บริการต่างๆที่เกิดขึ้นในระบบกระจายจำเป็นต้องมีตัวกลางสำหรับจัดการเกี่ยวกับการจัดเก็บ และการค้นหาบริการในระบบ ซึ่งตัวกลางเหล่านี้ถูกกำหนดเป็นบริการพื้นฐานของคอร์บา และมีมาตรฐานในการพัฒนาที่อ้างอิงจากโอเอ็มจี ได้แก่ บริการชื่อ (Naming Service) และบริการเทรดเดอร์ (Trading Object Service)

สำหรับบริการชื่อ ผู้ใช้งานสามารถกำหนดชื่อบริการของตนเองพร้อมทั้งระบุที่อยู่ แล้วลงทะเบียนไว้กับบริการชื่อ เพื่อให้ผู้อื่นสามารถค้นหาบริการได้จากชื่อเหล่านี้ บริการชื่อเปรียบเทียบได้กับสมุดโทรศัพท์ ซึ่งสามารถค้นหาหมายเลขโทรศัพท์จากชื่อผู้ใช้ได้ ส่วนบริการเทรดเดอร์เปรียบเทียบได้กับสมุดโทรศัพท์หน้าเหลือง ที่นอกจากมีชื่อและหมายเลขโทรศัพท์แล้ว ยังรวมถึงชนิดและคุณสมบัติของบริการด้วย ข้อมูลเหล่านี้ถูกนำมาประกอบการตัดสินใจสำหรับการเลือกบริการของผู้ใช้ การค้นหาและการประกาศใช้บริการใน เทรดเดอร์ จำเป็นต้องใช้ข้อมูลชนิดของบริการในการอ้างถึงรูปแบบของบริการใดๆ ที่มีอยู่ในระบบ เช่น บริการชนิดงานพิมพ์ต้องมีคุณสมบัติต่างๆ อาทิเช่น ชนิดเครื่องพิมพ์ ราคาต่อหน้า ความละเอียด เป็นต้น เมื่อผู้ใช้รู้คุณสมบัติเหล่านี้ ก็สามารถโฆษณาบริการของตนเองโดยการใส่ค่าตามคุณสมบัติเหล่านี้ และผู้ที่มาค้นหาบริการก็ต้องรู้ข้อมูลชนิดของบริการ ก่อนทำการค้นหาบริการเช่นกัน

ปัจจุบันบริการเทรดเดอร์ตามข้อกำหนดของโอเอ็มจี ยังมีข้อจำกัดอยู่ตรงที่ไม่สามารถค้นหาชนิดของบริการได้โดยมีสมมติฐานว่าผู้ใช้งานต้องรู้ชนิดของบริการก่อนหน้าที่จะเลือกใช้บริการ ปัญหาที่พบคือ

- ผู้ใช้ที่ต้องการเลือกใช้บริการใดๆ แต่ไม่รู้ชนิดของบริการว่ามีคุณสมบัติ และรูปแบบการเรียกใช้งานอย่างไร จะไม่สามารถค้นหาบริการนั้นได้จากบริการเทรดเดอร์
- ตามข้อกำหนดของโอเอ็มจีไม่มีการกำหนดส่วนต่อประสาน (Interface) หรือบริการ สำหรับการค้นหาชนิดของบริการในบริการเทรดเดอร์ ดังนั้นผู้พัฒนาบริการเทรดเดอร์จึงไม่มีมาตรฐานของการทำงานส่วนนี้
- รูปแบบการเก็บข้อมูลชนิดของบริการในบริการเทรดเดอร์ ยังไม่กำหนดเป็นมาตรฐาน โดยรูปแบบจะเป็นไปตามความต้องการของผู้พัฒนาบริการเทรดเดอร์แต่ละราย ทำให้รูปแบบไม่เป็นที่เข้าใจร่วมกัน ดังนั้นการใช้งานร่วมกันกับระบบอื่นๆ จึงเป็นไปได้ยาก
- การค้นหาบริการในบริการเทรดเดอร์กำหนดให้ใช้ภาษาบังคับ (Constraint Language) ที่ถูกกำหนดใน [2] ซึ่งไม่มีการใช้รูปแบบการค้นหาที่ซับซ้อน อย่างที่ใช้ในโปรแกรมค้นหา (Search Engine) เช่น การค้นหาโดยใช้คำสำคัญ และการค้นหาบางส่วนของคำ เป็นต้น

งานวิจัยนี้ นำเสนอรูปแบบของบริการค้นหาบริการ หรือเอสดีเอส (Service Discovery Service (SDS)) ซึ่งผู้ใช้สามารถใช้ค้นหาข้อมูลชนิดของบริการ ข้อมูลส่วนต่อประสาน รวมทั้งรายละเอียดของบริการ ได้จาก เทรดเดอร์หลายๆแหล่ง โดยการสร้างโปรแกรมสำหรับดึงข้อมูลชนิดของบริการและรายละเอียดของบริการต่างๆจากเทรดเดอร์ รวมทั้งข้อมูลส่วนต่อประสานจากคลังส่วนต่อประสาน (Interface Repository) แล้วนำมาจัดเก็บลงในฐานข้อมูล โดยเก็บอยู่ในรูปแบบของเอกสารเอ็กซ์เอ็มแอล (Extensible Markup Language (XML)) [3] โดยใช้การทำงานและโครงสร้างข้อมูลจากงานวิจัยส่วนการแปลงคำอธิบายบริการ [4] และสร้างส่วนต่อประสานสำหรับให้ผู้ใช้งานสามารถค้นหาข้อมูลเหล่านี้ ในรูปแบบการค้นหาเอกสารเอ็กซ์เอ็มแอล ทั้งแบบที่เป็นโครงสร้าง คือผู้ใช้งานรู้โครงสร้างของเอกสารก่อนทำการค้นหา โดยจะสามารถค้นหาด้วยภาษาสอบถามเอ็กซ์เอ็มแอล หรือค้นหาแบบไม่เป็นโครงสร้าง คือผู้ใช้สามารถใช้คำสำคัญในการค้นหาข้อมูลทั้งเอกสาร ซึ่งเป็นลักษณะเดียวกับโปรแกรมค้นหาได้

โดยสรุปบริการเอสดีเอสสามารถแก้ปัญหาข้อจำกัดของเทรดเดอร์ และเพิ่มประสิทธิภาพการให้ข้อมูลบริการในระบบกระจายได้ดังนี้

- บริการเอสดีเอสช่วยให้ผู้ใช้สามารถค้นหาชนิดของบริการในเทรดเดอร์ได้ โดยจะกำหนดเป็นส่วนต่อประสานสำหรับการค้นหาชนิดของบริการ และยังรองรับส่วนต่อประสานสำหรับผู้ใช้ในสภาพแวดล้อมอื่นที่นอกเหนือไปจากภายในระบบคอร์บา ซึ่งได้แก่ เวิลด์ไวด์เว็บด้วย
- เนื่องจากข้อมูลบริการถูกจัดเก็บในรูปแบบเอกสารเอ็กซ์เอ็มแอล ซึ่งปัจจุบันเป็นที่นิยมใช้สำหรับอธิบายข้อมูลในอินเทอร์เน็ต เนื่องจากมีความยืดหยุ่นสูง การใช้งานไม่ยึดติดกับโครงสร้างข้อมูล โดยสามารถแลกเปลี่ยนเอกสารกันได้ในขณะที่โครงสร้างต่างกัน นอกจากนี้เอกสารยังมีความหมายในตัวเอง สามารถสร้างและอ่านได้โดยง่าย ดังนั้นลักษณะของเอกสารที่เป็นเอ็กซ์เอ็มแอลจึงมีความเหมาะสมสำหรับการนำมาใช้จัดเก็บ และแลกเปลี่ยนข้อมูลในระบบกระจาย
- ความสามารถอื่นๆในการค้นหาบริการที่เพิ่มขึ้นคือ สามารถค้นหาบริการในลักษณะโปรแกรมค้นหาโดยใช้คำสำคัญได้ รวมทั้งสามารถใช้ภาษาสอบถามสำหรับเอ็กซ์เอ็มแอลเพื่อค้นหาข้อมูลบริการตามโครงสร้างของเอกสารได้ ในขณะเดียวกันบริการเอสดีเอสจะยังคงความสามารถในการค้นหาบริการตามรูปแบบของเทรดเดอร์ไว้ได้

นอกจากนี้บริการเอสดีเอสยังอาจได้รับการออกแบบให้มีความสามารถในการเชื่อมโยงกับบริการเอสดีเอสตัวอื่น ซึ่งเป็นการเพิ่มขยายความสามารถในการค้นหาให้กว้างขึ้นได้

1.2 วัตถุประสงค์

เพื่อออกแบบและพัฒนาโปรแกรมค้นหาข้อมูลชนิดของบริการ ข้อมูลบริการ และข้อมูลของส่วนต่อประสาน ที่รองรับผู้ใช้งานจากระบบกระจายของคอร์บา และจากระบบอื่นเช่น เวิลด์ไวด์เว็บ ได้ โดยมีการออกแบบและพัฒนารูปแบบการจัดเก็บข้อมูลของบริการจากเทรดเดอร์มากกว่า 1 แหล่ง ในรูปแบบของเอกสารเอ็กซ์เอ็มแอล

1.3 ขอบเขตการวิจัย

1. ต้นแบบของบริการเอสดีเอสจะทำงานอยู่บนสถาปัตยกรรมระบบกระจายที่ถูกพัฒนาขึ้นตามข้อกำหนดของคอร์บารุ่นปรับปรุงที่ 2.2 เป็นอย่างน้อย
2. การเฝ้าดูการเรียกใช้ตัวกระทำการสำหรับเทรดเดอร์เอเจนท์ในบริการเทรดเดอร์ จะกระทำเฉพาะกับตัวกระทำการที่กล่าวไว้ในหัวข้อ 3.3 เป็นอย่างน้อย
3. เอกสารเอ็กซ์เอ็มแอลที่ใช้ในงานวิจัยนี้ทั้งหมด เป็นตามข้อกำหนดของเอ็กซ์เอ็มแอล รุ่น 1.0 ของดับเบิลยูทรีซี [3] เป็นอย่างน้อย
4. การแปลงข้อมูลเอ็กซ์เอ็มแอล ทั้งในส่วนของการจัดเก็บและการค้นหา ใช้รูปแบบตามข้อกำหนดของเอ็กซ์เอสแอลที รุ่น 1.0 [5] เป็นอย่างน้อย
5. ส่วนต่อประสานสำหรับการค้นหาจะได้รับการพัฒนาให้รองรับภาษาสอบถามเอ็กซ์เอ็มแอลอย่างน้อย 1 ภาษา รวมทั้งการค้นหาโดยใช้คำสำคัญ
6. รูปแบบกลุ่มข้อมูลสำหรับการค้นหากำหนดไว้ 3 แบบ คือ
   - ค้นหาจากข้อมูลทั้งหมด
   - ค้นหาเฉพาะกลุ่มข้อมูลชนิดของบริการและส่วนต่อประสาน
   - ค้นหาเฉพาะกลุ่มข้อมูลบริการ
7. รูปแบบการค้นหาและการแสดงผลจากการค้นหา สามารถนำเสนอได้ 2 รูปแบบคือ รูปแบบสำหรับผู้ใช้คอร์บาทั่วไป ซึ่งสามารถค้นหาผ่านทางโปรแกรมคอร์บาไคลเอนต์ที่เรียกใช้ส่วนต่อประสานสำหรับการค้นหาได้ และรูปแบบสำหรับโปรแกรมค้นผ่านเว็บ ซึ่งทำงานในลักษณะเดียวกับโปรแกรมค้นหา โดยผู้ใช้สามารถค้นหาข้อมูลบริการและดูผลลัพธ์จากการค้นหาผ่านโปรแกรมค้นผ่านเว็บได้ โดยโปรแกรมค้นผ่านเว็บต้องสนับสนุนเอ็กซ์เอ็มแอล รุ่น1.0 และเอชทีเอ็มแอล รุ่น 4.0 เป็นอย่างน้อย
8. การเชื่อมโยงบริการเอสดีเอสเป็นการออกแบบการเชื่อมโยงตามรูปแบบการทำงานดังที่อธิบายไว้ในหัวข้อ 3.6 เป็นอย่างน้อย

1.4 ขั้นตอนและวิธีการดำเนินงานวิจัย

1. ศึกษาแนวทางการทำงานและการดึงข้อมูลจากบริการเทรดเดอร์ รวมทั้งรูปแบบการจัดการเอกสารเอ็กซ์เอ็มแอล ภาษาสอบถาม และโปรแกรมค้นหาเอกสารเอ็กซ์เอ็มแอล รวมทั้งส่วนประกอบอื่นๆ
2. ออกแบบการทำงานโดยรวมของระบบ และออกแบบส่วนย่อยแต่ละส่วน
3. พัฒนาต้นแบบของเทรดเดอร์เอเจนท์ และบริการเอสดีเอสตามที่ออกแบบไว้
4. พัฒนาโปรแกรมค้นหาข้อมูลสำหรับฝั่งผู้ใช้ ทั้งในรูปแบบคอร์บา และรูปแบบโปรแกรมค้นผ่านเว็บ
5. ทดสอบระบบ
6. สรุปผลและข้อเสนอแนะ
7. จัดทำรายงานวิทยานิพนธ์

1.5 ประโยชน์ที่จะได้รับ

1. ได้บริการค้นหาบริการซึ่งมีความสามารถมากกว่าบริการเทรดเดอร์ในปัจจุบัน โดยสามารถทำการค้นหาทั้งข้อมูลชนิดของบริการและข้อมูลของบริการเองได้ โดยสามารถเก็บรวบรวมข้อมูลบริการจากบริการเทรดเดอร์ และคลังส่วนต่อประสานมากกว่าหนึ่งแหล่ง และสามารถค้นหาข้อมูลได้โดยผู้ใช้คอร์บาทั่วไป และผู้ใช้จากเครือข่ายเวิลด์ไวด์เว็บ

2. ได้ลักษณะการใช้งานของแหล่งข้อมูลบริการในระบบกระจายของคอร์บา สามารถใช้งานได้ในสภาพแวดล้อมที่กว้างขวางขึ้น

1.6 ผลงานตีพิมพ์

ส่วนหนึ่งของวิทยานิพนธ์นี้ได้ตีพิมพ์ และนำเสนอในการประชุมวิชาการดังนี้

1. การประชุมทางวิชาการวิทยาการและวิศวกรรมคอมพิวเตอร์แห่งชาติ ครั้งที่ 4 (The 4th National Computer Science and Engineering Conference (NCSEC 2000)) เมื่อวันที่ 16-17 พฤศจิกายน 2543 ในบทความเรื่อง An Architecture for a Service Discovery Service in CORBA

โดยผู้แต่งคือ Worawut Suphasanthitikul and Twittie Senivongse

2. การประชุมวิชาการทางวิศวกรรมไฟฟ้าครั้งที่ 23 (The 23rd Electrical Engineering Conference (EECON-23)) เมื่อวันที่ 23-24 พฤศจิกายน 2543 ในบทความเรื่อง An Approach to Standardizing Distributed Service Descriptions Format using XML

โดยผู้แต่งคือ Wuttichai Nanekrangsan, Worawut Suphasanthitikul and Twittie Senivongse

3. The 5th International Enterprise Distributed Object Computing Conference EDOC 2001 September 4-7, 2001 Seattle, Washington USA ในบทความเรื่อง An XML-Based Architecture for Service Discovery

โดยผู้แต่งคือ Twittie Senivongse and Worawut Suphasanthitikul (ยังอยู่ในระหว่างการพิจารณา)

บทที่ 2
งานวิจัยและทฤษฎีที่เกี่ยวข้อง

2.1 งานวิจัยที่เกี่ยวข้อง

ปัจจุบันงานวิจัยและการพัฒนาความสามารถด้านการค้นหาบริการในระบบกระจายมีอยู่ในรูปแบบต่างๆ กัน อาทิเช่น การสร้างโปรแกรมค้นหาในอินเตอร์เน็ตสำหรับค้นหาคอมโพเนนท์ในระบบกระจาย หรือการคิดค้นระบบใหม่ๆเพื่อให้การค้นหาบริการง่ายขึ้น เป็นต้น โดยงานวิจัยส่วนหนึ่งมีดังนี้

2.1.1 AGORA ผิดพลาด! ไม่พบแหล่งการอ้างอิง

อะกอรา (AGORA) เป็นต้นแบบของระบบค้นหาคอมโพเนนท์ พัฒนาโดยสถาบันวิศวกรรมซอฟต์แวร์ (Software Engineering Institute) ในมหาวิทยาลัยคาร์เนกี เมลลอน (Carnegie Mellon) ต้นแบบที่ถูกพัฒนาขึ้นสามารถสร้างระบบแยกแยะ (Classification) และตัวบ่งชี้ (Indexing) สำหรับข้อมูลของซอฟต์แวร์บนเวิลด์ไวด์เว็บได้โดยอัตโนมัติ โดยจะสามารถค้นหาชนิดของคอมโพเนนท์ได้หลายรูปแบบ เช่น จาวาบีน (JavaBean) เอ็กทีฟเอ็กซ์ (ActiveX) และคอร์บา เป็นต้น วิธีการรวบรวมข้อมูลของคอมโพเนนท์มีที่มาจาก 2 แหล่งคือ จากโปรแกรมค้นหา และจากคำอธิบายภายในคอมโพเนนท์นั้นๆ ดังเช่นลักษณะของการตรวจสอบข้อมูลของตนเอง (Introspection) ในจาวาบีน สำหรับคอร์บาจะใช้การค้นหาข้อมูลภายในคลังส่วนต่อประสาน (Interface Repository) ข้อมูลที่รวบรวมได้จะนำมาสร้างเป็นฐานข้อมูล แบ่งแยกตามชนิดของคอมโพเนนท์ โดยการทำงานเหล่านี้เป็นกระบวนการที่ทำงานอยู่เบื้องหลัง ส่วนการค้นหาและการรับข้อมูลของ อะกอรามีคุณสมบัติคล้ายกับโปรแกรมค้นหาทั่วไป คือสามารถค้นหาโดยระบุ “AND” “OR” “NOT” และสามารถค้นหาเจาะจงไปยังฟิลด์ใดๆของข้อมูลได้ เช่น property:color OR attribute:color ส่วนการทำงานของการสร้างตัวบ่งชี้ และการเก็บข้อมูลของคอมโพเนนท์จะอาศัยการทำงานจากโปรแกรมช่วยเหลือชื่อ อัลตาวิสตา เอสดีเค (Altavista SDK) โดยสรุปแล้ว อะกอราสามารถทำงานได้ดีกับคอมโพเนนท์ของจาวาบีน แต่ค่อนข้างมีปัญหากับคอมโพเนนท์ของคอร์บา เนื่องจากการหาคอมโพเนนท์ของคอร์บานั้นเป็นการค้นหาข้อมูลที่เก็บอยู่ภายในบริการชื่อและคลังส่วนต่อประสาน ซึ่งอาจไม่มีใช้ในบางระบบ หรือถ้าหากมี จะหาที่อยู่ของบริการชื่อและคลังส่วนต่อประสานได้อย่างไร นอกจากนี้ แม้ว่าอะกอราจะสามารถค้นหาคอมโพเนนท์ของคอร์บาได้ แต่ไม่ได้นำข้อมูลจากบริการเทรดเดอร์มาเกี่ยวข้อง ทั้งๆที่เทรดเดอร์มีข้อมูลที่สมบูรณ์กว่าเกี่ยวกับคอมโพเนนท์ต่างๆ

2.1.2 WebTrader ผิดพลาด! ไม่พบแหล่งการอ้างอิง

เว็บเทรดเดอร์ (WebTrader) เป็นการพัฒนาระบบค้นหาบริการดังเช่นบริการเทรดเดอร์ให้ใช้งานได้บนเวิลด์ไวด์เว็บ ปัญหาที่เว็บเทรดเดอร์ ได้แก้ไขคือ ปัญหาของบริการเทรดเดอร์ในระบบกระจายส่วนใหญ่ถูกสร้างขึ้นโดยผูกติดกับสถาปัตยกรรมที่ใช้งานอยู่เท่านั้น และไม่สามารถจัดการบริการในสถาปัตยกรรมที่ต่างกันอย่างเวิลด์ไวด์เว็บได้ ดังนั้นจุดประสงค์ของงานนี้คือ การออกแบบและสร้างบริการเทรดเดอร์ที่สามารถจัดการกับบริการที่อยู่ในสถาปัตยกรรมระบบกระจายที่แตกต่างกันได้ โดยอ้างอิงกับสถาปัตยกรรมของเวิลด์ไวด์เว็บ ลักษณะการทำงาน

ของเทรดเดอร์ยังคงเป็นรูปแบบเดิม แต่ที่ต่างออกไปคือ การรับส่งข้อมูลเกี่ยวกับบริการจะใช้เป็นเอกสารเอ็กซ์เอ็มแอลแทน เนื่องจากเอ็กซ์เอ็มแอลสามารถให้ความหมายกับข้อมูลได้ดี ถูกใช้งานอย่างแพร่หลายสำหรับการนำเสนอข้อมูลบนอินเทอร์เน็ต รวมทั้งการส่งผ่านข้อมูลสามารถทำได้บนเวิลด์ไวด์เว็บอีกด้วย ผู้ให้บริการจะทำการโฆษณาบริการของตนเองในลักษณะข้อมูลที่เป็นเอ็กซ์เอ็มแอลจากหน้าโฆษณาบริการ (Service Advertisement Page (SAP)) ที่สร้างขึ้น และผู้ใช้ที่ค้นหาบริการต้องส่งข้อมูลการค้นหาเป็นเอ็กซ์เอ็มแอลเช่นกัน โดยต้องเป็นไปตามคำอธิบายชนิดเอกสารเรียกว่าดีทีดี (Document Type Description (DTD)) ที่ถูกกำหนดไว้แล้ว และระบุคุณสมบัติของบริการที่ต้องการมาด้วย จากนั้นที่อยู่อ้างอิง (Object Reference) และรายละเอียดของบริการที่ตรงกับความต้องการจะถูกส่งกลับไปยังผู้ใช้ แม้ว่าเว็บเทรดเดอร์จะสามารถทำงานร่วมกับระบบอื่นได้ แต่ก็ไม่ได้พัฒนาตามมาตรฐานของบริการเทรดเดอร์ใดๆ รวมถึงคำอธิบายบริการและการสอบถามในรูปของเอกสารเอ็กซ์เอ็มแอลยังต้องถูกสร้างด้วยมืออีกด้วย

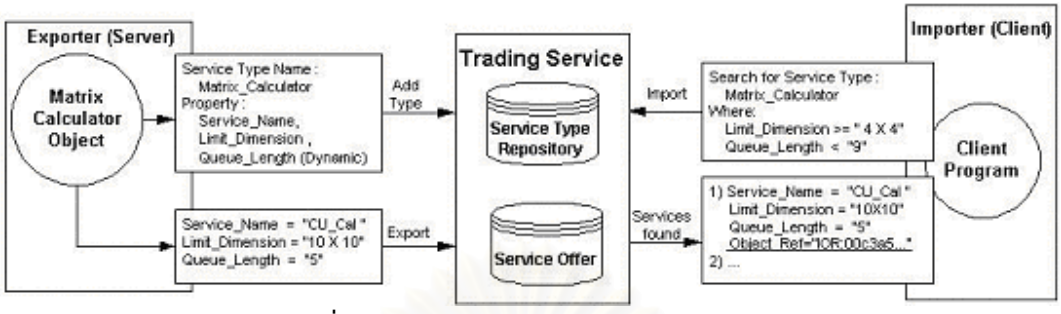
2.1.3 A Component Search Engine Model on Internet ผิดพลาด! ไม่พบแหล่งการอ้างอิง

ในงานวิจัยนี้ นำเสนอระบบค้นหาคอมโพเนนท์ โดยอาศัยหลักการจับคู่ชื่อ (Name Matching) และการจับคู่ความหมาย (Semantic Matching) ในการจับคู่ชื่อ จะอาศัยสัญนิยมการตั้งชื่อ (Name Convention) เป็นหลัก ทำให้การพิจารณาชื่อที่เหมือนกันทำได้ง่าย ส่วนในการจับคู่ความหมายนั้น ทำได้โดยระบบการเขียนเทอมใหม่ (Term Rewriting System (TRS)) ซึ่งคำอธิบายคอมโพเนนท์และการสอบถามข้อมูลจะอยู่ในรูปข้อกำหนดรูปนัยเชิงพีชคณิต (Algebraic Formal Specification) โดยที่หากคอมโพเนนท์มีความสอดคล้องกับข้อมูลที่สอบถาม ระบบค้นหาจะสามารถเขียนข้อกำหนดรูปนัยของการสอบถามได้ใหม่ (Rewrite) โดยอาศัยข้อกำหนดของคอมโพเนนท์ได้ อย่างไรก็ดี เนื่องจากทั้งคำอธิบายคอมโพเนนท์ และการสอบถามข้อมูลจะอยู่ในรูปข้อกำหนดรูปนัย จึงทำให้ยากที่จะเข้าใจโดยระบบงานอื่นและยากในการนำไปใช้งานจริง

2.2 ทฤษฎีที่เกี่ยวข้อง

2.2.1 บริการเทรดเดอร์ (OMG CORBA Trading Object Service) ผิดพลาด! ไม่พบแหล่งการอ้างอิง

บริการเทรดเดอร์เป็นบริการพื้นฐานของคอร์บา ซึ่งทำหน้าที่เป็นตัวกลางสำหรับให้ผู้อื่นใช้ในการจัดเก็บข้อมูลและค้นหาบริการในระบบของคอร์บา โดยผู้ให้บริการที่ประกาศบริการของตนเอง เรียกว่า เอ็กซ์พอร์ตเตอร์ (Exporter) (เทียบได้กับเซิร์ฟเวอร์) และผู้ที่ร้องขอการค้นหาบริการ เรียกว่า อิมพอร์ตเตอร์ (Importer) (เทียบได้กับไคลเอนต์)



รูปที่ 2.1 ตัวอย่างการทำงานของบริการเทรดเดอร์

จากรูปที่ 2.1 เป็นตัวอย่างการทำงานในบริการเทรดเดอร์โดยแยกเป็นขั้นตอนได้ดังนี้

1. เริ่มต้นด้วยการประกาศบริการจากผู้ให้บริการมายังเทรดเดอร์ จากตัวอย่างจะเห็นว่า ผู้ให้บริการเพิ่มข้อมูลชนิดของบริการเข้าสู่เทรดเดอร์ (ขั้นตอนนี้ทำในกรณีที่ยังไม่มีชนิดของบริการนี้อยู่ในเทรดเดอร์) ข้อมูลเหล่านี้จะถูกเก็บในคลังชนิดของบริการ (Service Type Repository) โดยข้อมูลชนิดของบริการประกอบด้วย ชื่อชนิดของบริการและคุณสมบัติที่เกี่ยวข้อง ดังที่ประกาศในข้อกำหนดของบริการเทรดเดอร์ในรูปแบบของสัญกรณ์บีเอ็นเอฟ (BNF) ดังนี้

service <ServiceTypeName>[:<BaseServiceTypeName>
[,<BaseServiceTypeName>]*]{
    interface <InterfaceTypeName>;
    [[mandatory] [readonly] property <IDLType> <PropertyName>;]*
};

จากตัวอย่างในรูปที่ 1 เป็นชนิดของบริการที่สามารถคำนวณเมตริกซ์ได้ (Matrix_Calculator) และมีคุณสมบัติคือ ชื่อบริการ (Service_Name) ขนาดของเมตริกซ์ที่รับได้ (Limit_Dimension) และจำนวนผู้รอใช้งาน (Queue_Length) เป็นต้น

2. หลังจากการประกาศชนิดของบริการ (หรือตรวจสอบว่ามีชนิดของบริการนี้อยู่แล้ว) ผู้ให้บริการจะทำการประกาศบริการเข้าสู่เทรดเดอร์ ซึ่งเรียกว่าการ เอ็กซ์พอร์ต (Export) โดยให้ค่าคุณสมบัติตามที่ระบุไว้ในชนิดของบริการ จากตัวอย่างในรูปที่ 2.1 บริการที่ประกาศมีคุณสมบัติดังนี้ ชื่อบริการ มีค่าเป็น “CU_Cal” ขนาดของเมตริกซ์ที่รับได้ มีค่าเป็น “10X10” และจำนวนผู้รอใช้งาน มีค่าเป็น “5” เป็นต้น

3. ส่วนของผู้เรียกใช้บริการ ทำการค้นหาบริการจากบริการเทรดเดอร์ ซึ่งเรียกว่าการ อิมพอร์ต (Import) โดยเริ่มต้นด้วยการตรวจสอบชนิดของบริการที่ต้องการกับคลังชนิดของบริการ พร้อมกับส่งข้อความสอบถามเข้าสู่บริการเทรดเดอร์ ดังเช่นตัวอย่าง เป็นการสอบถามเพื่อค้นหาบริการ “Matrix_Calculator” ซึ่งต้องมีค่าของคุณสมบัติ ขนาดของเมตริกซ์ที่รับได้ต้องมากกว่าหรือเท่ากับ “4X4” และจำนวนผู้รอใช้งานต้องน้อยกว่า “9” หลังจากนั้นผู้เรียกใช้บริการ จะได้ผลของบริการที่พบ พร้อมทั้งรายละเอียดของบริการ และข้อมูลอ้างถึงวัตถุ (Object Reference) และ

หลังจากนั้นผู้เรียกใช้บริการจะร้องขอบริการนั้นโดยใช้ข้อมูลอ้างถึงวัตถุ ซึ่งเป็นข้อมูลที่ระบุการติดต่อกับบริการได้โดยตรง โดยข้อมูลอยู่ในรูปแบบตัวเลขและตัวอักษรที่เข้ารหัสอยู่เช่น “IOR:003Ca5.....” เป็นต้น

2.2.2 อินเทอร์เซ็พเตอร์ (Interceptor) ผิดพลาด! ไม่พบแหล่งการอ้างอิง

อินเทอร์เซ็พเตอร์เป็นวัตถุที่ช่วยให้ผู้ใช้งานในระบบของคอร์บา สามารถแก้ไข หรือเพิ่มขยายพฤติกรรมการทำงานของออร์บ (Object Request Broker (ORB)) ได้ โดยการกำหนดรูปแบบ และตัวกระทำการ (Method) เพิ่มเพื่อการประมวลผลก่อนหรือหลังการทำงานปกติ อินเทอร์เซ็พเตอร์ทำงานในระดับล่าง สามารถเข้าถึงการทำงานของออร์บได้โดยตรง โดยใช้หลักการดักข้อความจีออพ (General Inter-ORB Protocol (GIOP)) เมื่อมีการติดต่อกับวัตถุใดๆ และสามารถแก้ไขข้อความเหล่านี้ก่อนที่จะส่งต่อไปได้ ซึ่งเป็นประโยชน์ เมื่อต้องการแทรกแซงการทำงานของวัตถุใดๆ โดยไม่ต้องทำการแก้ไขโปรแกรมโดยตรง

อินเทอร์เซ็พเตอร์มีความแตกต่างกันในการเรียกใช้งาน ซึ่งขึ้นอยู่กับบริษัทผู้ผลิต ในที่นี้ได้อ้างอิงรูปแบบของ วิสิโบรกเคอร์ (Visibroker) ของบริษัทอินไพรส์ (Inprise) [9] รูปแบบของอินเทอร์เซ็พเตอร์จะแตกต่างกันในแง่ของชนิดข้อความจีออพที่ทำการดัก โดยมี 3 รูปแบบ คือ

- **ไบนด์อินเทอร์เซ็พเตอร์** (Bind Interceptor) จัดการกับข้อมูลการเชื่อมต่อระหว่างเซิร์ฟเวอร์กับไคลเอนต์ มีตัวกระทำการให้ใช้งาน เช่น bind(..), bind_failed(..), rebind(..) เป็นต้น
- **ไคลเอนต์อินเทอร์เซ็พเตอร์** (Client Interceptor) จัดการกับข้อมูลทั้งการส่งออกและรับเข้าจากฝั่งไคลเอนต์ มีตัวกระทำการให้ใช้งาน เช่น prepare_request(..), send_request(..), receive_reply(..) เป็นต้น
- **เซิร์ฟเวอร์อินเทอร์เซ็พเตอร์** (Server Interceptor) จัดการกับข้อมูลที่ติดต่อเข้ามายังเซิร์ฟเวอร์ จัดการได้ทั้งข้อมูลการร้องขอและการส่งกลับสู่ไคลเอนต์ มีตัวกระทำการให้ใช้งาน เช่น locate(..), receive_request(..), prepare_reply(..) เป็นต้น

ในงานวิจัยนี้ใช้เฉพาะเซิร์ฟเวอร์อินเทอร์เซ็พเตอร์เท่านั้น เพราะต้องการตรวจจับการเรียกใช้ตัวกระทำการในบริการเทรดเดอร์ ซึ่งทำหน้าที่เป็นเซิร์ฟเวอร์ โดยสามารถรู้ได้ว่าไคลเอนต์ที่ติดต่อเข้ามา ใช้ตัวกระทำการใด ส่งข้อมูลอะไรมาบ้าง และดำเนินการสำเร็จหรือไม่ ดังมีรายละเอียดในหัวข้อ 4.1.1.

2.2.3 เอ็กซ์เอ็มแอล (Extensible Markup Language (XML)) [3]

เอ็กซ์เอ็มแอลเป็นข้อกำหนดที่พัฒนาโดยสหภาพเวิลด์ไวด์เว็บ หรือ ดับเบิลยูทรีซี (World Wide Web Consortium (W3C)) ใช้สำหรับอธิบายข้อมูล และเรียกข้อมูลเหล่านั้นว่า “เอกสารเอ็กซ์เอ็มแอล” เอ็กซ์เอ็มแอลเป็นส่วนหนึ่งของเอสจีเอ็มแอล (Standard Generalized Markup Language (SGML)) ที่นำเอาเฉพาะบางส่วนมาใช้งานเท่านั้น เนื่องจากเอสจีเอ็มแอลมีความซับซ้อน และใช้งานยากเกินไป ดังนั้นเอกสารเอ็กซ์เอ็มแอลใดๆสามารถใช้ได้เช่นเดียวกับเอสจีเอ็มแอลด้วย แนวคิดของการออกแบบเอ็กซ์เอ็มแอล คือ

- สามารถใช้งานบนอินเทอร์เน็ตได้
- สนับสนุนงานหลากหลายรูปแบบ
- ขียนโปรแกรมให้สามารถอ่านเอกสารเอ็กซ์เอ็มแอลได้ง่าย

- สามารถปรับปรุงแก้ไขเอกสารได้รวดเร็ว
- การออกแบบเอกสารต้องเป็นไปตามข้อกำหนดและมีความรัดกุม
- ผู้ใช้สามารถอ่าน และสร้างเอกสารได้ง่าย

เอกสารเอ็กซ์เอ็มแอลต้องมีรูปแบบที่ถูกต้อง (Well-Formed) โดยเอกสารที่มีรูปแบบที่ถูกต้อง จะเป็นเอกสารที่สมเหตุสมผล (Valid) ได้ก็ต่อเมื่อ เอกสารนั้นเป็นไปตามคำอธิบายชนิดของเอกสาร หรือ ดีทีดี (Document Type Description (DTD)) ดีทีดีเป็นเอกสารที่ทำหน้าที่กำหนดทั้งโครงสร้างและไวยากรณ์ของเอกสาร ซึ่งรวมอยู่ภายในหรือนอกเอกสารเอ็กซ์เอ็มแอลก็ได้ ตัวอย่างเอกสารเอ็กซ์เอ็มแอล มีดังนี้

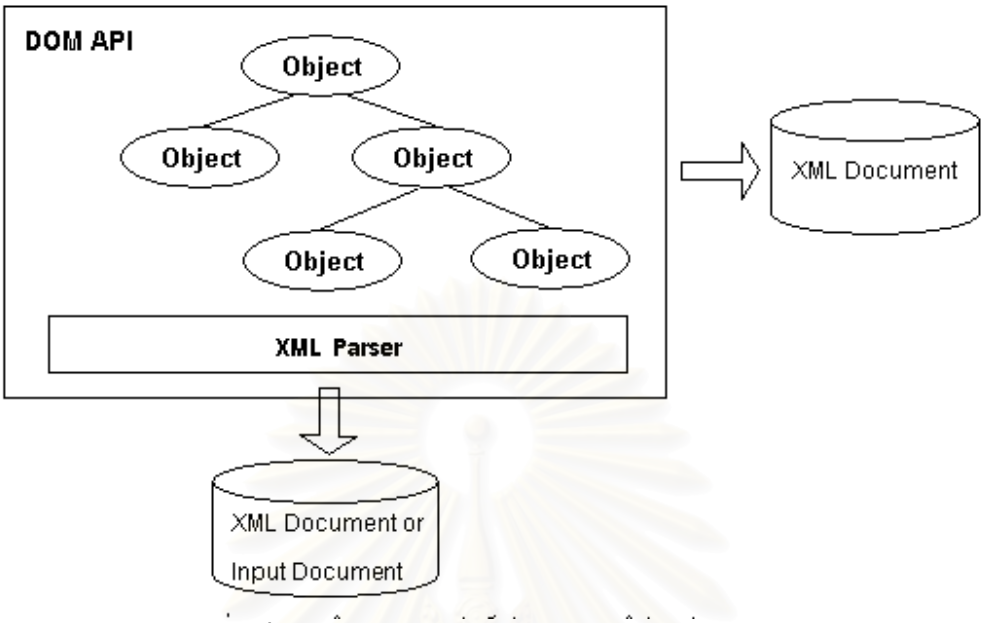
```
1  <?xml version="1.0" encoding="UTF-8" ?>
2  <!DOCTYPE greeting [<!ELEMENT greeting (#PCDATA)>]>
3  <greeting>Hello, world!</greeting>
```

จากตัวอย่างข้างต้น เป็นเอกสารเอ็กซ์เอ็มแอลอย่างง่าย บรรทัดที่ 1 แสดงถึงการเป็นเอกสารเอ็กซ์เอ็มแอลด้วยรูปแบบ <?xml …. ?> ระบุว่าใช้ตามมาตรฐานรุ่นที่ 1 และใช้การเข้ารหัสอักษรแบบ UTF-8 บรรทัดที่ 2 เป็นการระบุว่าเอกสารนี้เป็นชนิดที่มีดีทีดีอยู่ภายใน โดยมีการกำหนดป้ายระบุ (Tag) เพียงป้ายระบุเดียว ชื่อว่า greeting และส่วนที่ใช้งานในการอธิบายข้อมูลคือบรรทัดที่ 3 ข้อมูลจริงจะอยู่ภายใต้ป้ายระบุ <greeting>…</greeting>

เอ็กซ์เอ็มแอลเป็นเทคโนโลยีที่ถูกออกแบบให้เข้าใจง่าย และเป็นภาษาที่มีโครงสร้าง ซึ่งเรียกว่าข้อมูลกึ่งโครงสร้าง (Semi-Structured Data) [10] เพราะไม่ใช่ทั้งรูปแบบข้อมูลเชิงสัมพันธ์และเชิงวัตถุ แต่เป็นรูปแบบที่มีการอธิบายตัวเอง (Self-Describing) คือมีเค้าร่าง (Schema) รวมอยู่กับข้อมูล ข้อดีคือ รูปแบบข้อมูลเหมาะสมกับการใช้งานบนเว็บ เพราะมีรูปแบบที่ยืดหยุ่น ไม่มีโครงสร้างตายตัวเหมือนกับฐานข้อมูลเชิงสัมพันธ์ และอีกประการคือสามารถแลกเปลี่ยนข้อมูลกันระหว่างระบบที่มีรูปแบบฐานข้อมูลต่างกันได้ มาตรฐานของเอ็กซ์เอ็มแอลถูกกำหนดและพัฒนาจากหลายวงการนอกเหนือจากคอมพิวเตอร์ เช่น คณิตศาสตร์ เคมี และอุตสาหกรรมต่างๆ เป็นต้น ซึ่งทำให้เอ็กซ์เอ็มแอลมีแนวโน้มที่จะถูกใช้ในการอธิบายข้อมูลบน อินเทอร์เน็ตได้อย่างมีประสิทธิภาพในอนาคต

2.2.4 Document Object Model (DOM) [11]

ดีโอเอ็ม (Document Object Model (DOM)) คือข้อกำหนดของส่วนต่อประสานในการเขียนโปรแกรมประยุกต์ (Application Programming Interface (API)) เพื่อการจัดการข้อมูลเอกสารเอ็กซ์เอ็มแอลให้อยู่ในรูปของวัตถุที่สามารถเขียนชุดคำสั่งเพื่อเข้าถึง และแก้ไขเนื้อความภายในได้ ข้อกำหนดของดีโอเอ็มระดับหนึ่ง (DOM Level 1) ประกอบไปด้วยส่วนต่อประสานสำคัญสองส่วนคือ ส่วนแรกใช้ในการจัดการเอกสารเอ็กซ์เอ็มแอลในรูปของวัตถุ และส่วนที่สองคือส่วนต่อประสานที่ทำงานร่วมกับส่วนแรก เพื่อเพิ่มความสามารถในการ จัดการกับป้ายระบุของเอกสารเอ็ชทีเอ็มแอล ขั้นตอนการสร้างเอกสารเอ็กซ์เอ็มแอลโดยใช้ดีโอเอ็มอาจแสดงได้ดังรูปที่ 2.2



รูปที่ 2.2 การสร้างเอกสารเอ็กซ์เอ็มแอลโดยใช้ดีโอเอ็ม

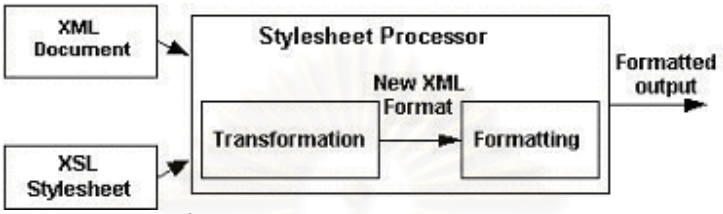
การจัดการเอกสารโดยใช้ดีโอเอ็มอาจเริ่มจากการอ่านเอกสารเอ็กซ์เอ็มแอลที่มีอยู่แล้วเพื่อนำมาแก้ไข หรือสามารถสร้างวัตถุสำหรับเอกสารเข้าขึ้นมาใหม่ก็ได้ เมื่อมีการอ่านเอกสารเอ็กซ์เอ็มแอลที่มีอยู่แล้วเข้ามา เอกสารนั้นจะผ่านการตรวจสอบไวยากรณ์ จากนั้นจึงมีการสร้างวัตถุที่มีเนื้อความภายในเช่นเดียวกับเอกสารขึ้นมา เมื่อผ่านการแก้ไขหรือเพิ่มเติมเนื้อความภายในแล้ว เราสามารถสร้างโปรแกรมเพื่อนำเนื้อความภายในวัตถุมาสร้างเอกสารเอ็กซ์เอ็มแอลได้อีกด้วย การใช้ดีโอเอ็มจะช่วยให้การจัดการ และการเขียนเอกสารเอ็กซ์เอ็มแอลที่อยู่ในรูปวัตถุทำได้ง่ายขึ้น โดยสามารถนำส่วนต่อประสานโปรแกรมประยุกต์เหล่านี้มาพัฒนาสร้างโปรแกรมประยุกต์ให้สามารถนำข้อมูลในรูปของวัตถุมาแสดงผลในรูปของส่วนต่อประสานกราฟิกกับผู้ใช้ (Graphical User Interface (GUI)) หรือนำเสนอผ่านสื่อในลักษณะอื่นได้อีกด้วย

ในปัจจุบันได้มีการพัฒนาดีโอเอ็มภายใต้ข้อกำหนดของดับเบิลยูทรีซีออกมามากมาย เช่น Java Project X จาก Sun Microsystems, Oracle XML Parser จาก Oracle และ MSIE5DOM จาก Microsoft โดยการพัฒนาเหล่านี้ยังได้เพิ่มเติมส่วนจัดการเอกสารที่เป็นประโยชน์ เช่น การสร้างแฟ้มข้อมูลเอกสารเอ็กซ์เอ็มแอล ทำให้สามารถนำการพัฒนาเหล่านี้ไปใช้ในการจัดการเอกสารเอ็กซ์เอ็มแอลได้โดยตรง

2.2.5 **เอ็กซ์เอสแอล** (Extensible Stylesheet Language (XSL)) [12] **และ เอ็กซ์เอสแอลที** (XSL Transformations (XSLT)) [12]

เอ็กซ์เอสแอลเป็นเทคโนโลยีที่ใช้สำหรับการเปลี่ยนรูปแบบของเอกสารเอ็กซ์เอ็มแอล ประกอบด้วย 2 ส่วนหลักคือ ภาษาสำหรับการแปลงรูป (Transformation Language) และภาษาสำหรับการจัดรูปแบบ (Formatting Language) ภาษาสำหรับการแปลงรูป ทำหน้าที่แปลงเอกสารเอ็กซ์เอ็มแอลต้นฉบับให้อยู่ในรูปแบบเอกสารเอ็กซ์เอ็ม

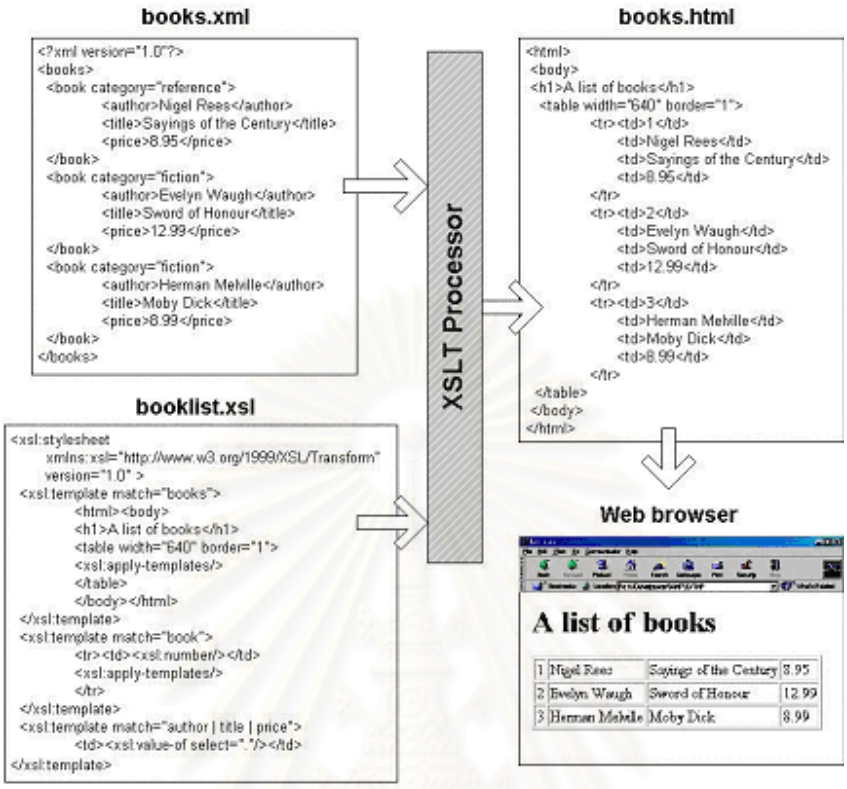
แอลที่ต้องการซึ่งถูกกำหนดเป็นมาตรฐานเรียกว่า “เอ็กซ์เอสแอลที” ส่วนภาษาสำหรับการจัดรูปแบบ ทำหน้าที่จัดรูปเอกสารเอ็กซ์เอ็มแอล เพื่อการนำเสนอสู่โปรแกรมอื่นๆ การจัดรูปแบบมีลักษณะคล้ายกับ ซีเอสเอส (Cascading Stylesheet (CSS)) แต่ต่างกันตรงที่ซีเอสเอสมีวัตถุประสงค์เพื่อใช้งานกับเว็บเพจเท่านั้น แต่เอ็กซ์เอสแอลสามารถใช้กับงานได้หลากหลายรูปแบบกว่า โดยเอ็กซ์เอสแอลมีลักษณะการทำงานดังรูปที่ 2.3



รูปที่ 2.3 รูปแบบการทำงานของเอ็กซ์เอสแอล

จากรูปที่ 2.3 แสดงถึงการทำงานของเอ็กซ์เอสแอล โดยใช้ตัวประมวลผลสไตล์ชีต (Stylesheet Processor) จัดการกับเอกสารเอ็กซ์เอ็มแอล ซึ่งถูกประมวลผลร่วมกับเอกสารเอ็กซ์เอสแอลที่กำหนด โดยสามารถแปลงเป็นเอกสารเอ็กซ์เอ็มแอล อีกรูปแบบหนึ่ง และจัดรูปแบบใหม่เพื่อให้สามารถแสดงผลได้

เอ็กซ์เอสแอลทีเป็นส่วนหนึ่งของเอ็กซ์เอสแอล ที่ทางดับเบิลยูทรีซีกำหนดไวยากรณ์และความหมายของภาษาสำหรับแปลงรูปเอกสารโดยเฉพาะ การแปลงรูปของเอ็กซ์เอสแอลที ใช้แนวคิดของการจับคู่แบบอย่าง (Pattern Matching) และแผ่นแบบ (Templates) ซึ่งในขณะที่เอกสารเอ็กซ์เอ็มแอล ถูกประมวลผลด้วยตัวประมวลผลเอ็กซ์เอสแอลที (XSLT Processor) ตัวประมวลผลจะค้นหาจากแบบอย่าง (Pattern) ในสไตล์ชีตและพยายามจับคู่รูปแบบเหล่านี้กับข้อมูลในเอกสารเอ็กซ์เอ็มแอล เมื่อพบว่าตรงกัน ตัวประมวลผลจะจัดการกับบัพ (Node) ที่พบด้วยแผ่นแบบที่กำหนด ให้อยู่ในรูปแบบที่ต้องการ ดังตัวอย่างในรูปที่ 2.4



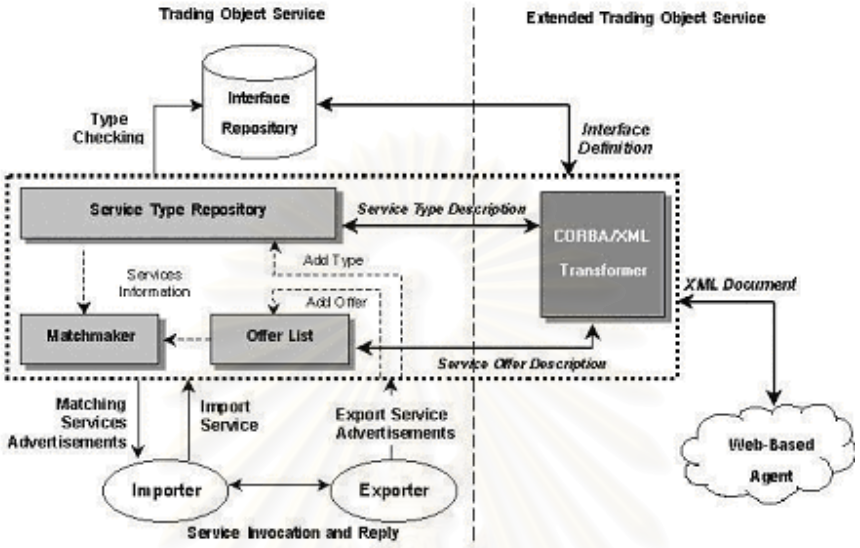
รูปที่ 2.4 ตัวอย่างการทำงานของเอ็กซ์เอสแอลที

จากรูปที่ 2.4 เอกสารเอ็กซ์เอ็มแอลที่ถูกนำมาแปลงรูปคือ เอกสารเอ็กซ์เอ็มแอล ชื่อ books.xml และเอกสารเอ็กซ์เอสแอล ชื่อ booklist.xsl ซึ่งเขียนตามไวยากรณ์ของเอ็กซ์เอสแอลที เริ่มต้นด้วยการระบุ XSLT Namespace เพื่อระบุประเภทของไวยากรณ์ และรุ่น (version) จากนั้น กำหนดให้หาส่วนย่อย (element) ที่ต้องการจาก <xsl:template match="....."> ถ้าพบก็จะเปลี่ยนให้เป็นรูปแบบที่ต้องการ ในที่นี้ถ้าพบส่วนย่อย books ก็จะเปลี่ยนให้เป็นเอกสารเอชทีเอ็มแอล (HTML) ที่แสดงถึงตาราง และแต่ละส่วนย่อย book จะถูกเปลี่ยนเป็นแต่ละแถวของตาราง เมื่อประมวลผลเสร็จทั้งหมด ก็จะได้เอกสารในรูปแบบที่ต้องการ ในที่นี้คือ books.html ที่แสดงข้อมูลได้ด้วยโปรแกรมค้นผ่านเว็บ (Web Browser)

2.2.6 **ส่วนการแปลงคำอธิบายบริการ** (CORBA/XML Transformer) [4]

งานวิจัยนี้ทำการออกแบบกฎสำหรับการแปลงข้อมูลชนิดของบริการ และข้อมูลของบริการภายในบริการเทรดเดอร์ ให้อยู่ในรูปแบบของเอกสารเอ็กซ์เอ็มแอล ที่สามารถทำการสืบค้นภายใต้เกณฑ์วิธี (Protocol) เดียวกันกับในเวิลด์ไวด์เว็บ นอกจากนี้ได้มีการพัฒนาต้นแบบ (Prototype) ของส่วนขยายของ เทรดเดอร์ให้มีความสามารถในการนำข้อมูลที่มีอยู่ภายในบริการเทรดเดอร์ และคลังส่วนต่อประสานมาเปลี่ยนให้อยู่ในรูปของเอ็กซ์เอ็มแอล เพื่อให้สามารถนำไปใช้ในตัวกลางในการค้นหาบริการ และชนิดของบริการภายใต้เวิลด์ไวด์เว็บ หรือค้นหาด้วยโปรแกรมค้นหาได้ โดยบริการเทรดเดอร์มีส่วนเพิ่มขยายดังนี้คือ ส่วนแปลงคำอธิบายบริการ (CORBA/XML Transformer)

และ ส่วนต่อประสานเพื่อการสร้างเอกสารเอ็กซ์เอ็มแอลจากข้อมูลบริการและชนิดของบริการทั้งหมดภายในบริการเทรดเดอร์ หรือตามชื่อของบริการและชนิดของบริการที่ผู้ใช้กำหนดก็ได้ ส่วนประกอบและการทำงานของบริการเทรดเดอร์ที่มีส่วนเพิ่มขยายจะเป็นดังรูปที่ 2.5



รูปที่ 2.5 บริการเทรดเดอร์ที่มีส่วนเพิ่มขยายส่วนการแปลงคำอธิบายบริการ

2.2.7 ภาษาสอบถามสำหรับเอ็กซ์เอ็มแอล (XML Query Language)

ดับเบิลยูทรีซีได้กำหนดมาตรฐานของภาษาที่ใช้เป็นภาษาสอบถามเอ็กซ์เอ็มแอลจากการตกลงกันร่วมกันในการสัมมนาเชิงปฏิบัติการ W3C QL '98 [Query Languages][1] ซึ่งนำไปสู่การนำเสนอข้อกำหนดของภาษาสอบถามเอ็กซ์เอ็มแอล [13] และภาษาสอบถามต่างๆ ซึ่งในปัจจุบันนี้มีข้อกำหนดภาษาสอบถามสำหรับเอ็กซ์เอ็มแอลโดยดับเบิลยูทรีซี ชื่อว่าเอ็กซ์คิวรี (XQuery) [14] แต่ยังคงเป็นฉบับร่าง โดยเอ็กซ์คิวรีรวมคุณสมบัติและข้อดีของภาษาสอบถามต่างๆมาใช้ แต่ปัจจุบันยังไม่มีเครื่องมือค้นหาใดพัฒนาขึ้นมาสนับสนุน ส่วนภาษาสอบถามสำหรับเอ็กซ์เอ็มแอลที่ใช้กันอยู่ในปัจจุบัน มีดังนี้

รูปแบบของภาษาสอบถามที่มีการนำเสนอต่อดับเบิลยูทรีซี ปัจจุบันมี 2 แบบ คือ เอ็กซ์เอ็มแอล-คิวแอล (XML-QL) [15] และเอ็กซ์คิวแอล (XQL) [16] โดยเอ็กซ์เอ็มแอล-คิวแอลมีวัตถุประสงค์เพื่อช่วยแก้ปัญหาของการค้นหาจากเอกสารเอ็กซ์เอ็มแอลที่มีขนาดใหญ่ โดยเฉพาะอีดีไอ (Electronic Data Interchange (EDI)) ซึ่งใช้ในการแลกเปลี่ยนข้อมูลทางธุรกิจ เอ็กซ์เอ็มแอล-คิวแอลมีรูปแบบภาษาและการใช้งานคล้ายกับภาษาซีเควล (Structured Query Language (SQL)) โดยพัฒนาจากรูปแบบมาตรฐานที่ใช้กับระบบฐานข้อมูลในปัจจุบัน ซึ่งมีคุณสมบัติการรวบรวมและการค้นหาข้อมูลจากเอกสารเอ็กซ์เอ็มแอลหลายๆแหล่ง และสามารถสร้างเป็นเอกสารชุดใหม่จากผลของการค้นหาได้

---

[1] สัมมนาเชิงปฏิบัติการ W3C QL '98 [Query Languages] เมื่อวันที่ 3-4 ธันวาคม พ.ศ. 2541 จากกลุ่มผู้เข้าร่วมการสัมมนา 100 คน และกลุ่ม mailing list ประมาณ 170 คน

ส่วนเอ็กซ์คิวแอลต่างกับเอ็กซ์เอ็มแอล-คิวแอลตรงที่ เอ็กซ์คิวแอลพัฒนาจากกลุ่มนักวิจัยด้านการค้นหาข้อมูลจากเอกสาร ซึ่งไม่ได้มีพื้นฐานจากการจัดการฐานข้อมูลอย่างเอ็กซ์เอ็มแอล-คิวแอล โดยเอ็กซ์คิวแอลถือได้ว่าสามารถทำงานเป็นส่วนหนึ่งของเอ็กซ์เอสแอลได้ เอ็กซ์คิวแอลถูกออกแบบให้รัดกุม ง่าย และมีประสิทธิภาพสูง ซึ่งใช้เป็นภาษาค้นหาในลักษณะประโยคเดียว (Single Syntax) ที่สามารถแทรกไปกับโปรแกรม สคริปต์ หรือใช้เป็นลักษณะประจำ (Attribute) ในเอ็กซ์เอ็มแอล หรือเอชทีเอ็มแอลได้

จากตารางที่ 2.1 จะเห็นว่าลักษณะภาษาเอ็กซ์คิวแอลมีความกระทัดรัด และสั้นกว่าการเขียนด้วยภาษาเอ็กซ์เอ็มแอล-คิวแอล ซึ่งผลที่ได้จะมีลักษณะเดียวกัน แต่ความแตกต่างอยู่ที่ลักษณะการนำไปใช้งาน ซึ่งได้กล่าวไว้แล้วว่าเอ็กซ์คิวแอลมุ่งเน้นไปที่ความกระชับของภาษา แต่สำหรับคุณสมบัติที่ซับซ้อนอย่างที่มีในภาษาสอบถามของฐานข้อมูลเชิงสัมพันธ์ในปัจจุบันเอ็กซ์เอ็มแอล-คิวแอลจะทำได้ดีกว่า

ตารางที่ 2.1 ตัวอย่างการใช้งานภาษาค้นหาเอ็กซ์เอ็มแอล-คิวแอลและเอ็กซ์คิวแอล

ลักษณะการค้นหา	เอ็กซ์เอ็มแอล-คิวแอล	เอ็กซ์คิวแอล

ค้นหา book ที่มี name ใน publisher เป็น Addison-Wesley และมีลักษณะประจำ year มากกว่าปี 1991	CONSTRUCT <bib> {
	WHERE
	<bib>
	<book year=$y> <title>$t</title>
	<publisher><name>Addison-Wesley</name>
	</publisher>
	</book> </bib>
	IN "www.bn.com/bib.xml",
	$y > 1991 CONSTRUCT <book year=$y><title>$t
	</title></book>
	} </bib>
		document("http://www.bn.com/bib.xml") {
		book[publisher/name="Addison-Wesley"
		and @year>1991] {
		@year | title
		}
		}

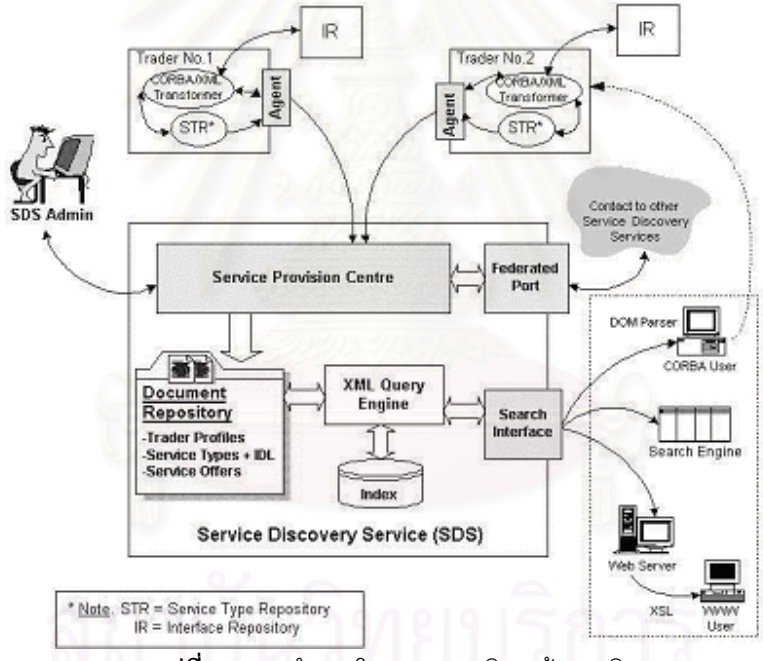
นอกจากภาษาสอบถามที่กล่าวมาแล้วยังมีภาษาอื่น ที่สนับสนุนการค้นหาข้อมูลเอ็กซ์เอ็มแอล ไม่ว่าจะเป็น YATL [17] Lorel [18] รวมทั้งที่กำลังพัฒนาอยู่ในปัจจุบันอีกจำนวนหนึ่ง

บทที่ 3
การออกแบบบริการเอสดีเอส

ในบทนี้จะกล่าวถึงการออกแบบโครงสร้างของบริการเอสดีเอส รวมถึงการทำงานในส่วนย่อยต่างๆ ภายในบริการ

3.1 โครงสร้างของระบบ

งานวิจัยนี้ เป็นการพัฒนาบริการค้นหาบริการหรือเอสดีเอส (Service Discovery Service (SDS)) ที่สามารถรวบรวม และค้นหาข้อมูลรายละเอียดของบริการ ในระบบกระจายของคอร์บา ไม่ว่าจะเป็นข้อมูลบริการจากบริการเทรดเดอร์ และข้อมูลส่วนต่อประสานจากคลังส่วนต่อประสาน ซึ่งเป็นข้อมูลเบื้องต้นที่เพียงพอสำหรับอธิบายคุณสมบัติและลักษณะส่วนต่อประสานของบริการที่มีอยู่ในคอร์บาได้ โดยจะนำข้อมูลเหล่านี้มาจัดเก็บรวบรวมไว้ ในรูปแบบเอกสารเอ็กซ์เอ็มแอล และสร้างส่วนต่อประสานเพื่อให้ผู้ใช้สามารถค้นหาข้อมูลเหล่านี้ได้ โดยมีภาพการทำงานโดยรวมของระบบ ดังรูปที่ 3.1



รูปที่ 3.1 การทำงานโดยรวมของบริการค้นหาบริการ

จากรูปที่ 3.1 การทำงานของระบบเริ่มต้นด้วยการทำงานของเทรดเดอร์เอเจนท์ (Trader Agent) ซึ่งทำหน้าที่ตรวจสอบการเปลี่ยนแปลงของข้อมูลที่เก็บในเทรดเดอร์ จากนั้นส่งข้อมูลการเปลี่ยนแปลงเหล่านี้เข้าสู่ส่วนการแปลงคำอธิบายบริการ (CORBA/XML Transformer) แล้วจึงส่งเอกสารเอ็กซ์เอ็มแอลที่ได้ไปยังบริการเอสดีเอส ซึ่งทำหน้าที่จัดเก็บข้อมูลลงฐานข้อมูล และให้ผู้ใช้สามารถค้นหาข้อมูลเหล่านี้ผ่านส่วนต่อประสานที่สร้างขึ้น โดยสามารถรองรับผู้ใช้ได้หลายลักษณะ เช่น ผู้ใช้คอร์บา หรือ ผู้ใช้จากเวิลด์ไวด์เว็บ เป็นต้น นอกจากนี้บริการเอสดีเอสสามารถเชื่อมต่อไปยังบริการเอสดีเอสอื่นๆ เพื่อการค้นหาที่กว้างขึ้นได้

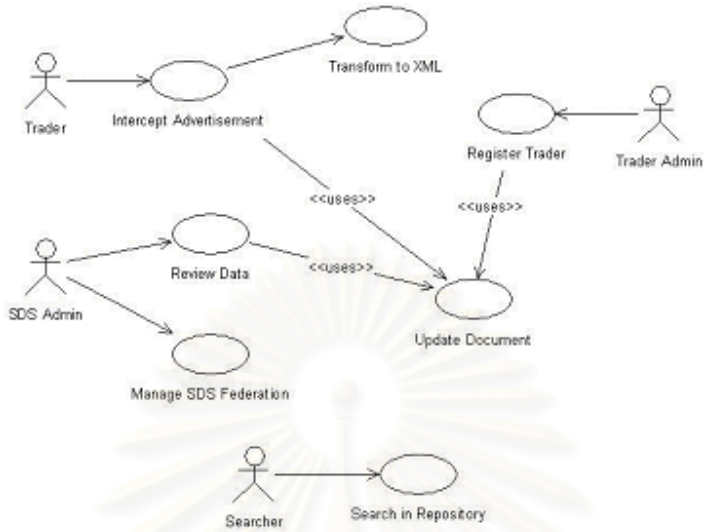
สำหรับครั้งแรกที่มีการลงทะเบียนจากบริการเทรดเดอร์มายังบริการเอสดีเอส ข้อมูลบริการทั้งหมดจากเทรดเดอร์จะถูกดึงมาเก็บไว้อย่างถาวร และบริการเอสดีเอสจะรอรับข้อมูลเพิ่มเติมจากบริการเทรดเดอร์หากมีการเปลี่ยนแปลงข้อมูลเกิดขึ้น ดังนั้นผู้ใช้จึงสามารถค้นหาข้อมูลบริการจากเอสดีเอสได้ แม้ว่าบริการ เทรดเดอร์จะหยุดทำงานก็ตาม

แนวทางการออกแบบบริการเอสดีเอสเป็นการสร้างบริการใหม่ ซึ่งแยกออกจากบริการเทรดเดอร์ แม้ว่าสามารถเพิ่มความสามารถดังที่กล่าวมาแล้ว ไว้ในเทรดเดอร์ได้ ทั้งนี้เพราะ

- ต้องการให้การค้นหากระทำได้ในขอบเขตที่กว้างขึ้น โดยมีการรวบรวมข้อมูลบริการจากบริการ เทรดเดอร์หลายๆแหล่ง
- การเพิ่มความสามารถดังที่กล่าวมาไว้ในเทรดเดอร์โดยตรง จะทำให้ต้องทำการแก้ไขส่วนต่อประสานของบริการเทรดเดอร์ ซึ่งเป็นข้อกำหนดจากโอเอ็มจี อาจทำให้บริการเทรดเดอร์รูปแบบใหม่ที่พัฒนาขึ้น ไม่สามารถใช้งานกับผู้ใช้รูปแบบเดิมหรือติดต่อกับบริการเทรดเดอร์เดิมได้ นอกจากนี้ยังเป็นการเปลี่ยนลักษณะการทำงานของเทรดเดอร์ตามการออกแบบของโอเอ็มจี เพราะการค้นหาบริการจากเทรดเดอร์นั้นมุ่งเน้นเพื่อให้ได้ตัวบริการที่เฉพาะเจาะจงสำหรับการเรียกใช้ตัวบริการนั้นต่อไป ดังนั้นผู้ใช้บริการจะต้องทราบข้อมูลที่ค่อนข้างละเอียดและแน่ชัดของบริการที่ต้องการเช่น ชื่อชนิดของบริการและคุณสมบัติของบริการ ล่วงหน้าก่อน แต่สำหรับบริการเอสดีเอสนั้นสามารถใช้เพื่อค้นหาข้อมูลเกี่ยวกับบริการต่างๆที่อาจเป็นประโยชน์ต่อการใช้งานก่อนที่จะนำไปสู่ตัวบริการที่เฉพาะเจาะจงได้ ผู้ใช้บริการสามารถค้นหาบริการได้โดยไม่ต้องทราบรายละเอียดของบริการที่แน่ชัด มีเพียงแนวทางคร่าวๆ ของบริการที่ต้องการก็จะสามารถใช้ประโยชน์จากบริการเอสดีเอสได้

3.2 **แบบจำลองยูสเคส** (Use Case model)

แบบจำลองยูสเคสใช้สำหรับอธิบายปฏิสัมพันธ์ระหว่างผู้ใช้งานและระบบ ซึ่งสามารถแบ่งเป็นกรณีต่างๆที่ผู้ใช้แต่ละคนกระทำกับกรณีนั้นๆได้ แบบจำลองยูสเคสของบริการเอสดีเอสเป็นดังนี้



รูปที่ 3.2 แบบจำลองยูสเคสของระบบ

จากแบบจำลองยูสเคสในรูปที่ 3.2 ผู้ใช้ในระบบ (Actors) แบ่งได้เป็น 4 กลุ่ม มีรายละเอียดดังนี้

- **เทรดเดอร์** (Trader) เทรดเดอร์เป็นบริการอย่างหนึ่ง ซึ่งถือว่าเป็นผู้ให้ข้อมูลบริการกับระบบ โดยผ่านกรณีดักการโฆษณาบริการ
- **ผู้ดูแลเทรดเดอร์** (Trader Admin) เป็นผู้ทำการลงทะเบียนเพื่อให้ข้อมูลรายละเอียดเกี่ยวกับบริการเทรดเดอร์ที่ตนเองดูแลอยู่กับระบบ
- **ผู้ดูแลบริการค้นหาบริการ** (SDS Admin) ทำหน้าที่ตรวจสอบความถูกต้องของข้อมูล และความผิดพลาดที่เกิดกับข้อมูลในบริการเอสดีเอส รวมทั้งการบริหารการเชื่อมโยงบริการด้วย
- **ผู้ค้นหา** (Searcher) เป็นผู้ใช้งานบริการเอสดีเอส โดยค้นหาบริการที่เก็บอยู่ในคลังจัดเก็บข้อมูลของบริการเอสดีเอส

ส่วนกรณีที่เกิดขึ้นในระบบประกอบด้วย

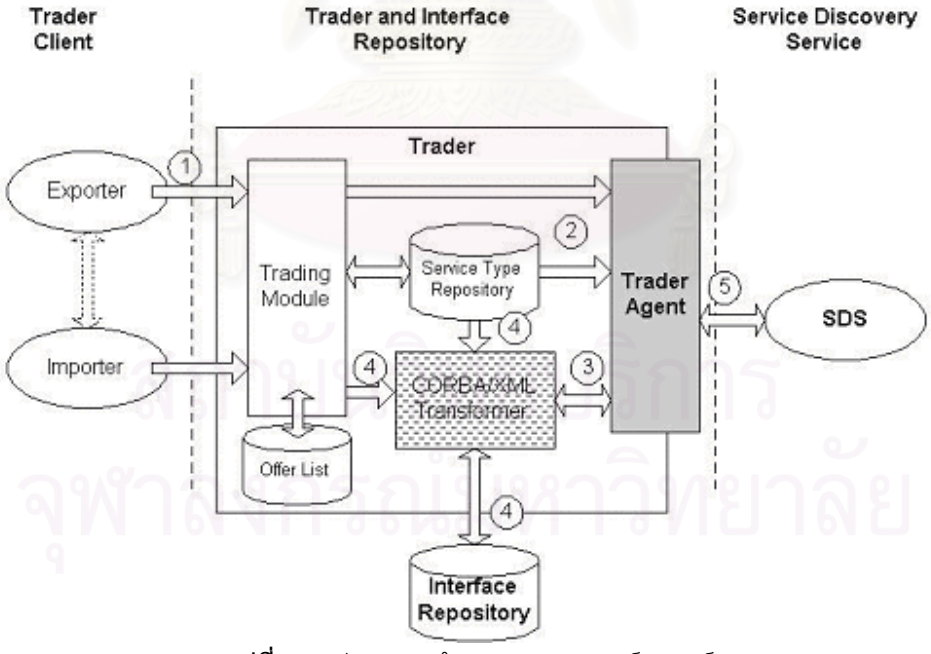
- **กรณีดักการโฆษณาบริการ** (Intercept Advertisement) กระทำโดยบริการเทรดเดอร์ เพื่อดักการเปลี่ยนแปลงข้อมูลบริการในเทรดเดอร์ และนำมาจัดการและจัดเก็บโดยเรียกใช้กรณีปรับปรุงเอกสาร
- **กรณีแปลงข้อมูลเป็นเอ็กซ์เอ็มแอล** (Transform to XML) กระทำโดยกรณีดักโฆษณาบริการ เพื่อแปลงข้อมูลให้เป็นเอกสารเอ็กซ์เอ็มแอล ก่อนการส่งไปจัดเก็บ
- **กรณีลงทะเบียนบริการเทรดเดอร์** (Register Trader) กระทำโดยผู้ดูแลบริการเทรดเดอร์ เพื่อเพิ่มข้อมูลรายละเอียดของเทรดเดอร์เข้าสู่บริการเอสดีเอส โดยเรียกใช้กรณีปรับปรุงเอกสารเพื่อการจัดเก็บข้อมูล

- **กรณีตรวจสอบข้อมูล** (Review Data) กระทำโดยผู้ดูแลบริการค้นหาบริการ เพื่อตรวจสอบและแก้ไขบริการที่อยู่ในคลังจัดเก็บข้อมูลของบริการเอสดีเอส โดยเรียกใช้กรณีปรับปรุงเอกสาร เมื่อจำเป็นต้องเปลี่ยนแปลงข้อมูลใดๆ ในคลังจัดเก็บข้อมูล
- **กรณีปรับปรุงเอกสาร** (Update Document) ถูกเรียกใช้โดย กรณีดักการโฆษณาบริการ กรณีลงทะเบียนบริการเทรดเดอร์ และกรณีตรวจสอบความสมเหตุสมผลของข้อมูล โดยจะทำหน้าที่ปรับปรุงข้อมูลในคลังจัดเก็บข้อมูล โดยการรองรับการเพิ่ม ลบ และแก้ไขข้อมูลบริการในคลังจัดเก็บข้อมูลของเอสดีเอส
- **กรณีจัดการการเชื่อมโยงบริการค้นหาบริการ** (Manage SDS Federation) ถูกใช้โดยผู้ดูแลบริการค้นหาบริการ เพื่อบริหารการเชื่อมโยงไปสู่บริการอื่น
- **กรณีค้นหาในคลังจัดเก็บ** (Search in Repository) กระทำโดยผู้ค้นหา โดยสามารถค้นหาข้อมูลบริการที่มีในคลังจัดเก็บข้อมูลได้

จากโครงสร้างของระบบและแบบจำลองยูสเคส บริการเอสดีเอสแบ่งได้เป็น 4 ส่วนหลัก รายละเอียดของแต่ละส่วนประกอบเป็นดังนี้

3.3 **เทรดเดอร์เอเจนท์** (Trader Agent)

เทรดเดอร์เอเจนท์ทำงานอยู่ร่วมกับบริการเทรดเดอร์ในลักษณะของโปรแกรมคอร์บาอินเทอร์เซ็พเตอร์ ซึ่งทำหน้าที่คอยตรวจสอบการใช้งานตัวกระทำการที่เกี่ยวข้องกับการเปลี่ยนแปลงข้อมูลภายในบริการเทรดเดอร์ ทั้งข้อมูลชนิดของบริการและข้อมูลบริการ การทำงานของเทรดเดอร์เอเจนท์เป็นดังรูปที่ 3.3



รูปที่ 3.3 รูปแบบการทำงานของเทรดเดอร์เอเจนท์

ขั้นตอนที่ 1 เป็นการทำงานโดยปกติของบริการเทรดเดอร์ คือผู้ให้บริการติดต่อเพื่อเพิ่มหรือแก้ไขบริการในบริการเทรดเดอร์ ขั้นตอนที่ 2 เทรดเดอร์เอเจนท์ทำหน้าที่เฝ้าดูตัวกระทำการที่ผู้ให้บริการเรียกใช้ โดยใช้วิธีการของเซิร์ฟเวอร์อินเทอร์เซ็พเตอร์เพื่อดักตัวกระทำการที่เรียกเข้ามายังบริการเทรดเดอร์ ไม่ว่าจะเป็นตัวกระทำการจากโมดูลของเทรดเดอร์หรือจากคลังชนิดของบริการ หากตัวกระทำการนั้นทำให้เกิดการเปลี่ยนแปลงข้อมูลภายในเทรดเดอร์ เทรดเดอร์เอเจนท์จะทำงานต่อในขั้นตอนที่ 3 เทรดเดอร์เอเจนท์จะทำการร้องขอการดึงข้อมูล และการแปลงข้อมูลของบริการเป็นเอกสารเอ็กซ์เอ็มแอล จากส่วนการแปลงคำอธิบายบริการ และในขั้นตอนที่ 4 เป็นหน้าที่ส่วนการแปลงคำอธิบายบริการ ที่ทำการดึงข้อมูลจากแหล่งข้อมูลต่างๆ แล้วแปลงเป็นเอกสารเอ็กซ์เอ็มแอล จากนั้นทำการส่งข้อมูลกลับสู่เทรดเดอร์เอเจนท์ซึ่งจะเรียกใช้บริการเอสดีเอสเพื่อเก็บข้อมูลลงในฐานข้อมูลต่อไปในขั้นตอนที่ 5

การดักการเปลี่ยนแปลงข้อมูลของเทรดเดอร์จะกระทำเมื่อมีการเรียกใช้ตัวกระทำการของเทรดเดอร์ (ตามข้อกำหนดของ [12]) ที่มีผลกับการเปลี่ยนแปลงของข้อมูลบริการในเทรดเดอร์ ซึ่งมีดังนี้

<u>ตัวกระทำการจากส่วนต่อประสานคลังจัดเก็บชนิดของบริการ</u>

- เพิ่มชนิดของบริการ จากตัวกระทำการดังนี้

```
IncarnationNumber add_type (
  in CosTrading::ServiceTypeName name,
  in Identifier if_name,
  in PropStructSeq props,
  in ServiceTypeNameSeq super_types
)
```

- ลบชนิดของบริการ จากตัวกระทำการดังนี้

```
void remove_type (
  in CosTrading::ServiceTypeName name
)
```

<u>ตัวกระทำการจากส่วนต่อประสานการลงทะเบียนบริการ (Register Interface) ของเทรดเดอร์</u>

- เพิ่มบริการ จากตัวกระทำการดังนี้

```
OfferId export (
  in Object reference,
  in ServiceTypeName type,
  in PropertySeq properties
)
```

- ยกเลิกบริการ จากตัวกระทำการดังนี้

```
void withdraw (
  in OfferId id
)
```

- เปลี่ยนแปลงค่าคุณสมบัติบริการ จากตัวกระทำการดังนี้

```
void modify (
  in OfferId id,
  in PropertyNameSeq del_list,
  in PropertySeq modify_list
)
```

3.4 ศูนย์จัดเตรียมบริการ (Service Provision Centre)

ศูนย์จัดเตรียมบริการทำหน้าที่เป็นตัวกลางจัดการข้อมูลบริการ คอยรับข้อมูลจากเทรดเดอร์เอเจนท์และผู้ดูแลเทรดเดอร์ (Trader Admin) แล้วนำมาเก็บลงในฐานข้อมูล โดยแบ่งลักษณะข้อมูลออกได้เป็น 3 รูปแบบ คือ

- ข้อมูลรายละเอียดของบริการเทรดเดอร์ (Trader Profile) คือ ข้อมูลที่อธิบายรายละเอียด และข้อมูลการติดต่อที่ใช้อ้างถึงบริการเทรดเดอร์ใดๆ ข้อมูลเหล่านี้ได้มาจากการลงทะเบียนครั้งแรกของเทรดเดอร์เพื่อติดต่อกับบริการเอสดีเอส รูปแบบข้อมูลประกอบด้วย

  1. ชื่อเทรดเดอร์ (Trader Name)
  2. ข้อมูลในการอ้างถึงวัตถุของเทรดเดอร์
  3. ข้อมูลผู้ดูแลบริการเทรดเดอร์ประกอบด้วย ชื่อ ที่อยู่ อีเมล์ และอื่นๆ
  4. ข้อมูลอื่นๆ

  ในการติดต่อกับบริการเอสดีเอส ชื่อบริการเทรดเดอร์ที่ได้มาจากการลงทะเบียน จะถูกใช้อ้างถึงบริการเทรดเดอร์นั้นๆ และใช้ร่วมกับการเรียกตัวกระทำการต่างๆของศูนย์จัดเตรียมบริการ ข้อมูลรายละเอียดของบริการเทรดเดอร์เป็นไปตามรูปแบบดีทีดีที่กำหนดไว้ตามภาคผนวก ก. ซึ่งข้อมูลเหล่านี้จะถูกนำมาใช้เมื่อต้องการอ้างถึงเทรดเดอร์ใดๆ ที่เข้ามาลงทะเบียนในระบบ

- ข้อมูลชนิดของบริการจากคลังจัดเก็บชนิดของบริการ และข้อมูลส่วนต่อประสานจากคลังจัดเก็บส่วนต่อประสาน ลักษณะของข้อมูล เป็นไปตามที่อธิบายในหัวข้อที่ 2.2.1 โดยเอกสารเอ็กซ์เอ็มแอลที่ส่งมาจากเทรดเดอร์เอเจนท์ จะมีรูปแบบตามที่กำหนดไว้ในงานวิจัย [4] แต่ข้อมูลดังกล่าวมาจากบริการเทรดเดอร์หลายแหล่ง ดังนั้นจำเป็นต้องแทรกชื่อของบริการเทรดเดอร์เพื่อเป็นตัวระบุแหล่งที่มาเข้าไปในเอกสาร โดยใช้ชื่อว่า TraderNameซึ่งเป็นส่วนย่อยเพิ่มเติมจากดีทีดีใน[4](ภายใต้รากของส่วนย่อย <ServiceTypeDescription>) ดังนี้

  <!ELEMENT TraderName (#PCDATA)>

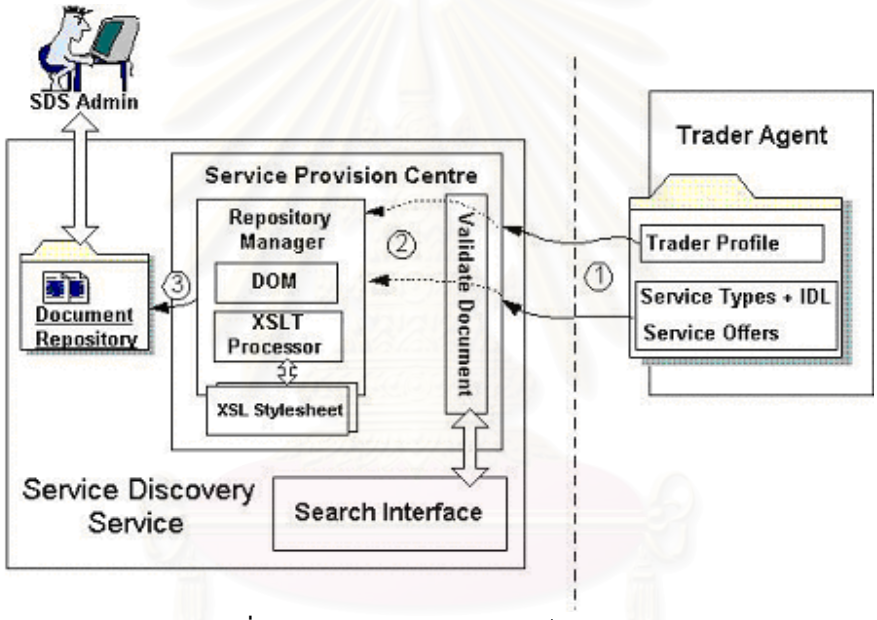
  ดูรายละเอียดของดีทีดีได้ในภาคผนวก ก

- ข้อมูลบริการจากบริการเทรดเดอร์ จัดเก็บในลักษณะเดียวกับชนิดของบริการ โดยเอกสารจะมีรูปแบบตามที่กำหนดไว้ในงานวิจัย [4] ดังได้นำมาแสดงไว้ในภาคผนวก ก เช่นกัน แต่ได้เพิ่มเติมส่วนที่ใช้ระบุแหล่งที่มาของข้อมูลคือ TraderName และ OfferId ในรูปแบบลักษณะประจำภายใต้รากของเอกสารดีทีดีเดิม (รากเอกสารใช้ส่วนย่อยชื่อ <ServiceOfferDescription>) ดังนี้

  <!ATTLIST ServiceOfferDescription TraderName CDATA #REQUIRED
  OfferId CDATA #REQUIRED>

รูปที่ 3.4 แสดงการทำงานภายในของศูนย์จัดเตรียมบริการ ในขั้นตอนที่ 1 เป็นการนำข้อมูลเข้าสู่บริการเอสดีเอสโดยผ่านเทรดเดอร์เอเจนท์ ขั้นตอนที่ 2 ทำการตรวจสอบข้อมูลทั้งความถูกต้องของรูปแบบและความสัมพันธ์กับข้อมูลเดิมในระบบ ในขั้นตอนนี้จะเรียกใช้การทำงานของส่วนต่อประสานสำหรับการค้นหาเพื่อค้นหาข้อมูลที่มีอยู่ในระบบด้วย ทั้งนี้เพื่อตรวจสอบความซ้ำซ้อนและความมีอยู่ของข้อมูล จากนั้นจัดส่งข้อมูลไปยังตัวจัดการคลังเก็บข้อมูล (Repository Manager) ซึ่งทำหน้าที่ติดต่อกับคลังเก็บข้อมูลโดยตรง โดยการจัดเก็บข้อมูลรายละเอียดของ

บริการเทรดเดอร์ จะเก็บในรูปแบบเดิมอย่างที่ได้รับมาจากเทรดเดอร์ เอเจนท์ (เพราะเป็นรูปแบบข้อมูลที่ถูกกำหนดเพื่อบริการเอสดีเอสโดยเฉพาะ) ส่วนข้อมูลชนิดของบริการ ส่วนต่อประสาน และบริการ จำเป็นต้องผ่านการแปลงให้อยู่ในรูปแบบที่เหมาะสมก่อนที่จะเก็บ คือมีการเพิ่มเติมข้อมูลเกี่ยวกับเทรดเดอร์ที่มา ดังที่กล่าวมาแล้ว ซึ่งการทำงานส่วนนี้สามารถใช้ดีโอเอ็มมาจัดการได้ เพราะดีโอเอ็มมีตัวกระทำการที่สามารถแทรกส่วนย่อยและลักษณะประจำลงในเอกสารเดิมได้ นอกจากนี้แม้ว่างานวิจัยนี้จะใช้ดีทีดีดังในภาคผนวก ก เพื่ออธิบายข้อมูลบริการภายในเทรดเดอร์เท่านั้น แต่ในความเป็นจริงแล้ว เทรดเดอร์หลายแหล่งที่ให้ข้อมูลกับบริการเอสดีเอสเดียวกันอาจใช้ดีทีดีคนละแบบกันในการอธิบายข้อมูลบริการ ซึ่งอาจมีโครงสร้างต่างกันก็เป็นไปได้ ดังนั้นผู้วิจัยจึงออกแบบให้บริการเอสดีเอสมีความสามารถในการปรับเปลี่ยนเอกสารเอ็กซ์เอ็มแอลที่ได้จากเทรดเดอร์ทั้งหลายเหล่านี้ให้เป็นเอกสารที่มีสกีมา (Schema) หรือโครงสร้างตามดีทีดีที่บริการเอสดีเอสยึดถืออยู่ได้ด้วย ในส่วนนี้จึงมีการนำตัวประมวลผลเอ็กซ์เอสแอลทีมาช่วยในการปรับเปลี่ยนเอกสารข้อมูลบริการก่อนทำการจัดเก็บ



รูปที่ 3.4 การทำงานของศูนย์จัดเตรียมบริการ

ตัวจัดการคลังเก็บข้อมูลจะแปลงเอกสารโดยใช้เอกสารเอ็กซ์เอสแอลสไตล์ชีต ที่เขียนขึ้นตามไวยากรณ์ของเอ็กซ์เอสแอลที ในงานวิจัยนี้บริการเอสดีเอสและเทรดเดอร์จะยึดถือดีทีดีของข้อมูลบริการแบบเดียวกัน ดังนั้นตัวประมวลผลเอ็กซ์เอสแอลทีจะทำหน้าที่เพียงดึงเฉพาะส่วนที่ต้องการเกี่ยวกับชนิดของบริการหรือตัวบริการเองจากเอกสารเอ็กซ์เอ็มแอลที่ถูกประมวลผลเท่านั้น โดยตัดส่วนที่ไม่จำเป็น เช่น ที่อยู่ของดีทีดี ซึ่งระบุในเอกสารออกไป เอกสารเอ็กซ์เอสแอลสไตล์ชีตสำหรับชนิดของบริการจะเป็นดังนี้

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
<xsl:template match="ServiceTypeDescription">
   <xsl:copy>
            <xsl:copy-of select="*"/>
   </xsl:copy>
</xsl:template>
</xsl:stylesheet>
```

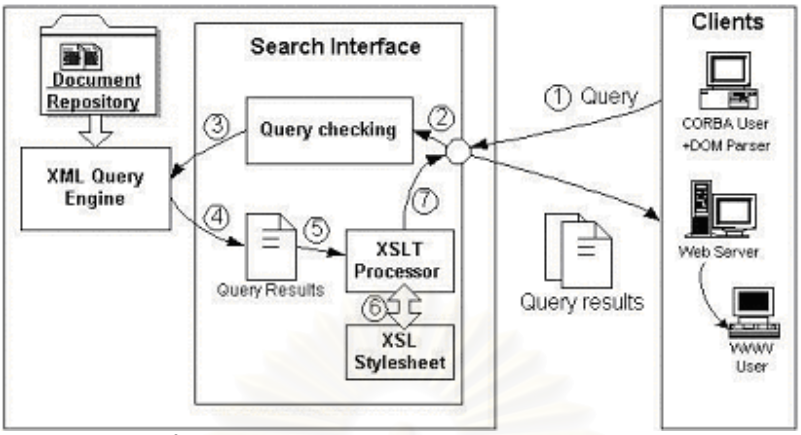
เอกสารเอ็กซ์เอสแอลสไตล์ชีตข้างต้นทำหน้าที่จับคู่และสำเนาข้อมูลภายใต้ส่วนย่อยที่ชื่อ ServiceTypeDescription เท่านั้น ส่วนเอกสารเอ็กซ์เอสแอลสไตล์ชีตสำหรับบริการทำงานในลักษณะเดียวกัน แต่ดึงข้อมูลภายใต้ส่วนย่อย ServiceOfferDescription ดังนี้

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
<xsl:template match="ServiceOfferDescription">
   <xsl:copy>
            <xsl:copy-of select="*"/>
   </xsl:copy>
</xsl:template>
</xsl:stylesheet>
```

หลังจากนั้นในขั้นตอนที่ 3 จะทำการเก็บข้อมูลลงในคลังเก็บข้อมูล หลังจากข้อมูลถูกจัดเก็บแล้วหากมีข้อผิดพลาดเกิดขึ้นผู้ดูแลระบบสามารถเข้าแก้ไขข้อมูลเหล่านี้ได้โดยตรง ศูนย์จัดเตรียมบริการจะเตรียมส่วนต่อประสานให้สามารถ เพิ่ม ลบ และแก้ไขข้อมูลเหล่านี้ได้ โดยเอกสารที่จัดเก็บจะถูกจัดการในรูปแบบของ ดีโอเอ็ม เพราะสามารถนำไปประมวลผลด้วยตัวแจง (Parser) และทำการแปลงรูปเอกสารโดยใช้ตัวประมวลผลเอ็กซ์เอสแอลทีได้ทันที

3.5 **ส่วนต่อประสานสำหรับการค้นหา** (Search Interface)

ส่วนต่อประสานสำหรับการค้นหาเป็นส่วนหลักของบริการเอสดีเอส โดยสามารถพัฒนาให้รองรับการสอบถามโดยใช้ภาษาสอบถามเอ็กซ์เอ็มแอลหรือภาษาอื่นๆได้ ทั้งนี้ขึ้นอยู่กับผู้พัฒนาบริการเอสดีเอสว่าต้องการให้ค้นหาข้อมูลด้วยภาษาใดได้บ้าง ซึ่งผู้สอบถามข้อมูลสามารถระบุภาษาที่ทำการสอบถามผ่านส่วนต่อประสานที่กำหนดได้ ส่วนต่อประสานสำหรับการค้นหามีรูปแบบการทำงานดังรูปที่ 3.5



รูปที่ 3.5 การทำงานของส่วนต่อประสานสำหรับการค้นหา

จากรูปที่ 3.5 การทำงานขั้นที่ 1 ผู้ใช้ส่งการสอบถามมายังส่วนต่อประสานสำหรับการค้นหา และจากนั้นในขั้นที่ 2 จะมีการตรวจสอบว่าผู้ใช้งานใช้ภาษาสอบถามใด และตรวจไวยากรณ์ว่าถูกต้องหรือไม่ จากนั้นก็ทำการค้นหาผ่านโปรแกรมค้นหาในขั้นตอนที่ 3 เมื่อได้ผลการค้นหาจากขั้นตอนที่ 4 ก็จะเข้าสู่ขั้นตอนที่ 5 และ 6 ซึ่งเป็นการแปลงข้อมูลผลที่ได้ ให้อยู่ในรูปแบบที่เหมาะสมก่อนการส่งกลับให้ผู้ใช้ในขั้นตอนที่ 7 ต่อไป การค้นหาสามารถทำได้กว้างขึ้น หากมีการพัฒนาการเชื่อมโยงบริการเอสดีเอสเข้าด้วยกัน ซึ่งหมายความว่าสามารถค้นหาไปยังแหล่งข้อมูลอื่นได้

<u>คุณสมบัติของส่วนต่อประสานสำหรับการค้นหามีดังนี้</u>

1) รูปแบบผู้ใช้งานส่วนต่อประสานสำหรับการค้นหามี 2 ลักษณะดังนี้

- ผู้ใช้งานคอร์บา สามารถเรียกผ่านส่วนต่อประสานการค้นหาของเอสดีเอส ซึ่งเป็นลักษณะเดียวกับการเรียกใช้บริการพื้นฐานของคอร์บาทั่วไป หรือเรียกใช้ผ่านข้อมูลอ้างอิงวัตถุได้
- ผู้ใช้ผ่านเกณฑ์วิธีเอชทีทีพี (HTTP Protocol) เช่นผู้ใช้งานผ่านโปรแกรมค้นผ่านเว็บ หรือเป็นโปรแกรมค้นหา หรือโปรแกรมประยุกต์ใดๆที่ต้องการค้นหาข้อมูลเหล่านี้เพื่อนำไปใช้ ทั้งนี้ต้องเพิ่มโปรแกรมเพื่อรองรับการทำงานส่วนนี้อาจเป็น จาวาเซิร์ฟเล็ต (JavaServlet) หรือซีจีไอ (CGI) ที่สามารถเรียกใช้งานส่วนต่อประสานของคอร์บาได้

2) รูปแบบการค้นหา กำหนดเป็น 2 ลักษณะคือ

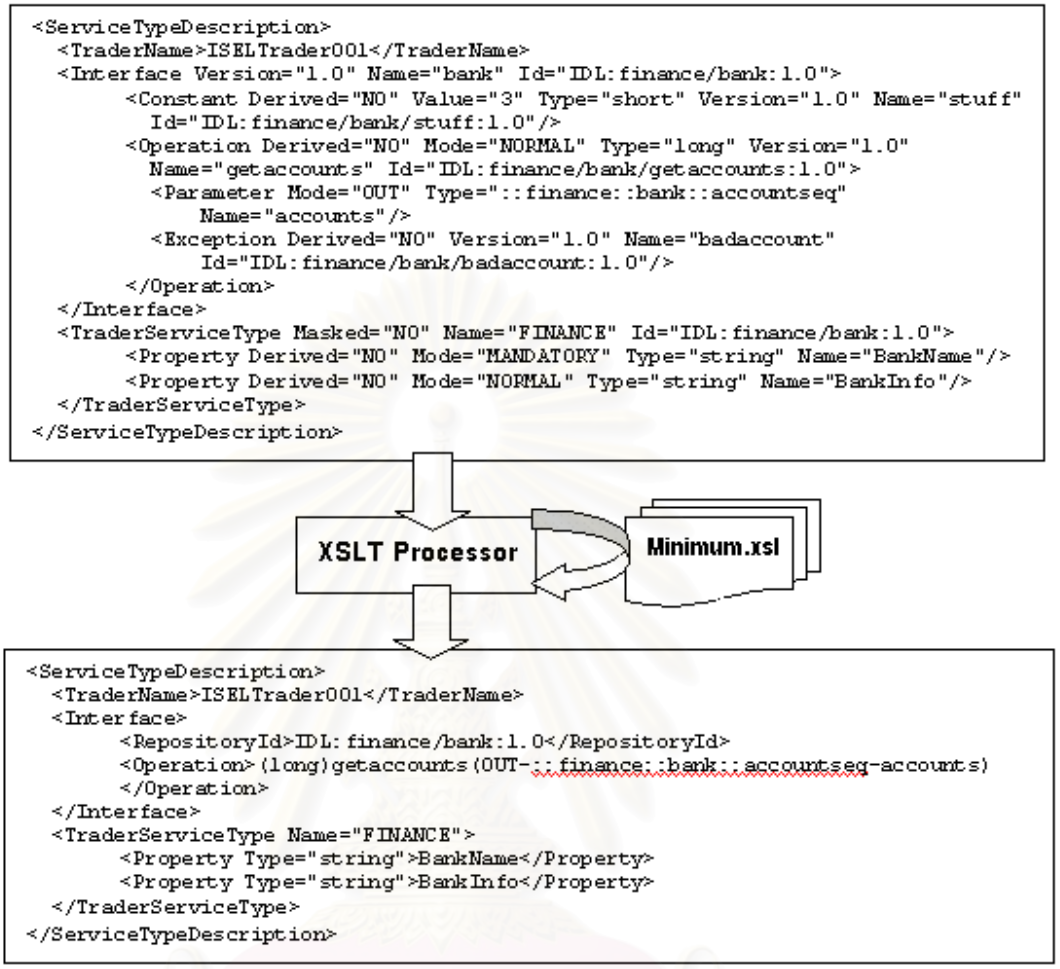
- การค้นหาโดยรู้โครงสร้างของเอกสารเอ็กซ์เอ็มแอล ทั้งนี้ใช้รูปแบบของภาษาสอบถามเอ็กซ์เอ็มแอล โดยความสามารถในการค้นหาขึ้นอยู่กับความสามารถของโปรแกรมค้นหาที่นำมาพัฒนาร่วมกับระบบ และความเชี่ยวชาญในภาษาสอบถามนั้นๆ ของผู้ใช้
- การค้นหาโดยไม่รู้โครงสร้างเอกสาร ซึ่งใช้ลักษณะเดียวกันกับโปรแกรมค้นหาโดยทั่วไป ที่ใช้เฉพาะคำสำคัญ หรือคำที่เกี่ยวข้อง

3) รูปแบบของผลลัพธ์ของการค้นหา ผลของการค้นหาจะเป็นเอกสารเอ็กซ์เอ็มแอลที่ได้จากโปรแกรมค้นหาเอ็กซ์เอ็มแอลที่นำมาใช้ แต่ได้รับการแปลงให้อยู่ในรูปแบบที่เหมาะสมก่อนที่จะส่งกลับให้ผู้ใช้จริง ทั้งนี้จะใช้ตัวประมวลผลเอ็กซ์เอสแอลทีกับเอกสารเอ็กซ์เอสแอลเพื่อการแปลงผลลัพธ์ (ดูภาคผนวก ข) เพื่อทำงานในส่วนนี้ รูปแบบของผลการค้นหาได้กำหนดไว้ 2 ลักษณะคือ

- แบบสมบูรณ์ (MAXIMUM) จะแสดงรายละเอียดทั้งหมดของข้อมูล ไม่ว่าจะเป็นผลลัพธ์ของข้อมูลชนิดของบริการหรือข้อมูลบริการ โดยกำหนดให้ผลลัพธ์แต่ละส่วนย่อยที่พบ อยู่ภายใต้ส่วนย่อย <row> และผลลัพธ์ทั้งหมดอยู่ภายใต้ส่วนย่อยรากชื่อ <ResultSet>
- แบบย่อ (MINIMUM) จะมีผลเฉพาะกับผลลัพธ์ของข้อมูลชนิดของบริการหรือส่วนต่อประสานเท่านั้น โดยจะเลือกแสดงเฉพาะข้อมูลบางส่วนเนื่องจากข้อมูลดังกล่าวเป็นข้อมูลที่มีรายละเอียดมาก อีกทั้งข้อมูลบางส่วนไม่ค่อยมีการนำมาใช้จริงมากนัก สามารถแสดงได้ด้วยส่วนของดีทีดีดังนี้

```
<!ELEMENT ServiceTypeDescription (TraderName, Interface?,
            TraderServiceType)>
<!ELEMENT TraderName (#PCDATA)>
<!ELEMENT Interface (RepositoryId*, Attribute*, Operation*)>
            <!ELEMENT RepositoryId (#PCDATA)>
            <!ELEMENT Attribute (#PCDATA)>
            <!ELEMENT Operation (#PCDATA)>
<!ELEMENT TraderServiceType (Property*)>
            <!ATTLIST TraderServiceType Name CDATA #REQUIRED>
            <!ELEMENT Property (#PCDATA)>
             <!ATTLIST Property Type CDATA #REQUIRED>
```

ส่วนของดีทีดีนี้แสดงถึงข้อมูลในส่วนย่อยของ ServiceTypeDescription ซึ่งจะเห็นว่ามีการลดข้อมูลไปบางส่วนเมื่อเทียบกับดีทีดีเดิม (ในภาคผนวก ก) ตัวอย่างข้อมูลที่มีการแสดงผลแบบย่อ แสดงไว้ดังรูปที่ 3.6



รูปที่ 3.6 ตัวอย่างการแปลงผลลัพธ์เป็นแบบย่อ

การแสดงผลการค้นหาข้อมูลบริการแบบย่อจะให้ผลเหมือนกับการแสดงผลแบบสมบูรณ์ เพราะข้อมูลมีจำนวนน้อยอยู่แล้ว โดยข้อมูลผลลัพธ์ทั้งหมดใช้ส่วนย่อยสำหรับกำหนดผลลัพธ์เช่นเดียวกับแบบสมบูรณ์

3.6 **ส่วนเชื่อมโยงบริการ** (Federated Port)

ส่วนเชื่อมโยงบริการเป็นรูปแบบการเชื่อมต่อบริการแบบเดียวกันในระบบ ซึ่งใช้กันโดยทั่วไปในบริการพื้นฐานของคอร์บา เช่น บริการเทรดเดอร์ หรือบริการชื่อ เป็นต้น ในที่นี้บริการเอสดีเอสถูกออกแบบให้สามารถรองรับการเชื่อมโยงไปยังบริการเอสดีเอสตัวอื่นได้ การเชื่อมโยงบริการมีตัวกระทำการที่สำคัญคือ เพิ่มการเชื่อมโยง (Add Link) ลบการเชื่อมโยง (Remove Link) และแก้ไขการเชื่อมโยง (Modify Link) โดยการทำงานพื้นฐานของการเชื่อมโยงคือ

1. การเพิ่มการเชื่อมโยงจากบริการอื่น ซึ่งต้องมีการระบุข้อมูลที่จำเป็นสำหรับการเชื่อมโยง เช่น ชื่อการเชื่อมโยง ข้อมูลอ้างอิงบริการ และข้อจำกัดต่างๆ

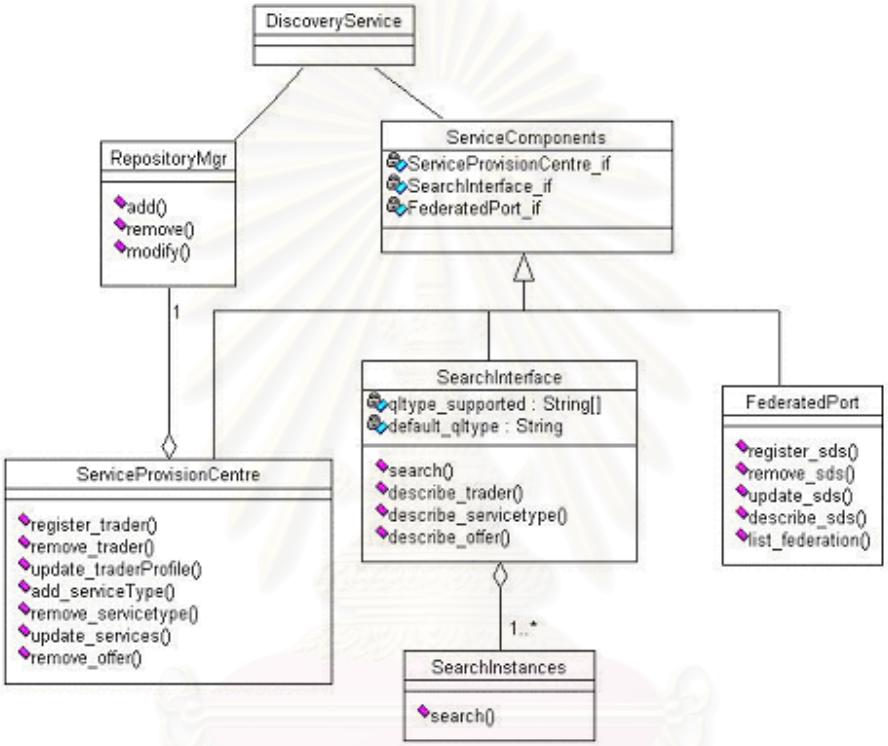
2. เมื่อค้นหาข้อมูลภายในระบบของตนเองไม่พบ และการค้นหานี้สามารถค้นหาไปยังบริการอื่นได้ บริการก็จะทำการสืบค้นไปยังบริการอื่นที่อยู่ในรายการเชื่อมโยง เมื่อพบก็จะส่งค่ากลับมารวบรวมไว้ แล้วส่งต่อให้ผู้ใช้ต่อไป

3. นอกจากนี้ยังสามารถแก้ไข หรือลบข้อมูลการเชื่อมโยง จากส่วนต่อประสานที่พัฒนาขึ้นได้

วิทยานิพนธ์นี้จะทำการออกแบบการเชื่อมโยงบริการเอสดีเอสในรูปแบบ 3 ข้อข้างต้น และกำหนดรูปแบบส่วนต่อประสานสำหรับการเชื่อมโยงเท่านั้น โดยจะสามารถนำไปพัฒนาให้ใช้งานได้จริงและขยายให้มีความซับซ้อนต่อไปได้

บทที่ 4
ต้นแบบบริการเอสดีเอส

ในบทนี้จะอธิบายต้นแบบบริการเอสดีเอสที่พัฒนาขึ้นตามการออกแบบจากบทที่ 3 โดยจะกล่าวถึงรูปแบบโครงสร้างหลักของระบบ รวมทั้งส่วนต่อประสานของบริการเอสดีเอสที่กำหนดขึ้น และอธิบายในส่วนของการพัฒนาส่วนประกอบต่างๆ โดยการออกแบบในเบื้องต้นเป็นไปตามแบบจำลองคลาส (Class Model) ดังนี้



รูปที่ 4.1 แบบจำลองคลาสของบริการเอสดีเอส

จากแบบจำลองคลาสในรูปที่ 4.1 สามารถแบ่งเป็นคลาสต่างๆได้ดังนี้

- DiscoveryService เป็นเสมือนคลาสหลักสำหรับเรียกใช้บริการเอสดีเอส ซึ่งจะเรียกใช้การทำงานของคลาสต่างๆ ที่เกี่ยวข้อง รวมทั้งการตั้งค่าเริ่มต้นทั้งหมดของบริการเอสดีเอส เพื่อส่งไปให้ส่วนต่างๆ ทำงาน เช่น ไดเร็คทอรีที่ใช้เก็บข้อมูล ที่อยู่ของดีทีดี และที่อยู่ของเอกสารเอ็กซ์เอสแอล เป็นต้น
- RepositoryMgr เป็นคลาสที่ใช้เชื่อมต่อกับฐานข้อมูลจริง สามารถเพิ่ม ลบ แก้ไขข้อมูลที่จัดเก็บ ผ่านคลาสนี้ได้
- ServiceComponents เป็นคลาสที่รวมเอาส่วนต่อประสานของทุกส่วนประกอบในเอสดีเอสไว้ด้วยกัน โดยส่วนต่อประสานเหล่านี้ต้องทำการสืบทอดคลาสนี้ เพื่อที่จะสามารถเรียกใช้ส่วนต่อประสานอื่นๆ ได้จากจุดเดียวกัน

- ServiceProvisionCentre เป็นคลาสที่รองรับการทำงานของศูนย์จัดเตรียมบริการ มีตัวกระทำการที่ใช้สำหรับการจัดการข้อมูลในรูปแบบต่างๆ กัน
- SearchInterface เป็นคลาสที่รองรับการทำงานของส่วนต่อประสานสำหรับการค้นหา โดยสามารถทำการค้นหา และเรียกดูบริการโดยละเอียดจากตัวกระทำการที่กำหนดได้ คลาสนี้มีลักษณะประจำเป็นรายการของภาษาค้นหาที่สนับสนุน และค่าโดยปริยายของภาษาค้นหา
- SearchInstances เป็นคลาสที่สร้างขึ้นเพื่อการพัฒนาเครื่องมือค้นหา (Query Engine) ที่รองรับภาษาค้นหาหนึ่งๆ ได้ โดยสามารถพัฒนาได้หลายๆ คลาส ขึ้นอยู่กับความต้องการที่จะสนับสนุนภาษาค้นหาใด ซึ่งคลาสนี้ถูกเรียกใช้โดยคลาส SearchInterface
- FederatedPort เป็นคลาสที่รองรับการทำงานของส่วนเชื่อมโยงบริการ ทำหน้าที่รับการลงทะเบียนสำหรับการเชื่อมโยงจากบริการเอสดีเอสอื่น รวมทั้งมีตัวกระทำการให้สามารถแก้ไขปรับปรุงข้อมูลเหล่านี้ได้

4.1 ส่วนต่อประสานของบริการเอสดีเอส

ส่วนต่อประสานของบริการเอสดีเอสกำหนดขึ้นเพื่อรองรับการเรียกใช้ตัวกระทำการต่างๆ ผ่านส่วนต่อประสานในรูปแบบของบริการคอร์บา ซึ่งมีรายละเอียดดังนี้

```
module SDService {
    interface ServiceProvisionCentre;
    interface SearchInterface;
    interface FederatedPort;
    typedef string TraderName;
    typedef string TraderProfile;
    typedef string ServiceTypeName;
    typedef string ServiceTypeInfo;
    typedef string SourceURL;
    typedef string ServiceInfo;
    typedef string ServiceID;
    typedef string OfferId;
```

ส่วนนี้เป็นการประกาศชื่อโมดูล ส่วนต่อประสานที่มีทั้งหมด และชนิดของตัวแปรที่ถูกกำหนดขึ้นเพื่อใช้งานร่วมกันในส่วนต่อประสานต่างๆ

เอ็กซ์เซ็พชันของบริการเอสดีเอส

เอ็กซ์เซ็พชันเหล่านี้เป็นเอ็กซ์เซ็พชันร่วม ซึ่งส่วนต่อประสานอื่นสามารถเรียกใช้ได้ มีดังนี้

เอ็กซ์เซ็พชัน	คำอธิบาย

exception UnknownServiceType {
ServiceTypeName st_name;
};
	เอ็กซ์เซ็พชันที่เกิดขึ้นเมื่อไม่พบชนิดของบริการ โดยจะส่งชื่อชนิดของบริการกลับไป

exception UnknownTraderName {
TraderName name;
};
	เอ็กซ์เซ็พชันที่เกิดขึ้นเมื่อไม่พบชื่อของบริการ เทรดเดอร์ที่มาลงทะเบียนโดยจะส่งชื่อบริการ เทรดเดอร์กลับไป

4.1.1 ส่วนต่อประสานส่วนประกอบของบริการเอสดีเอส

```
interface ServiceComponents {
        readonly attribute ServiceProvisionCentre
        ServiceProvisionCentre_if;
    readonly attribute SearchInterface
        SearchInterface_if;
    readonly attribute FederatedPort
        FederatedPort_if;
};
```

ส่วนต่อประสานนี้ประกอบด้วยที่เป็นส่วนต่อประสานของส่วนประกอบของบริการเอสดีเอส กำหนดขึ้นเพื่อให้ส่วนต่อประสานของส่วนประกอบได้สืบทอด เพื่อให้แต่ละส่วนต่อประสานสามารถเรียกใช้ส่วนต่อประสานอื่นได้ เช่น ศูนย์จัดเตรียมบริการจำเป็นต้องใช้ส่วนต่อประสานสำหรับการค้นหา เพื่อค้นหาว่าชนิดของบริการมีอยู่หรือไม่ ก่อนทำการลบหรือแก้ไขชนิดของบริการนั้นออกจากระบบ เป็นต้น

4.1.2 ส่วนต่อประสานของศูนย์จัดเตรียมบริการ

```
interface ServiceProvisionCentre : ServiceComponents {
```

เป็นการประกาศส่วนต่อประสาน ซึ่งสืบทอดมาจากส่วนต่อประสานของส่วนประกอบบริการเอสดีเอส ดังที่กล่าวข้างต้น

เอ็กซ์เซ็พชันของศูนย์จัดเตรียมบริการ

เอ็กซ์เซ็พชัน	คำอธิบาย

exception TraderNameExists {
TraderName name;
};
	เอ็กซ์เซ็พชันนี้เกิดขึ้น เมื่อพบว่าบริการเทรดเดอร์ที่มาลงทะเบียนมีชื่ออยู่ในระบบแล้ว โดยจะส่งชื่อของบริการเทรดเดอร์นั้นกลับไป

exception IllegalProfile { TraderProfile tp;	เอ็กซ์เซ็พชันนี้เกิดขึ้น เมื่อพบว่าข้อมูลของบริการเทรดเดอร์ที่ส่งมาลงทะเบียน หรือแก้ไข มีข้อผิดพลาด หรือไม่

};	ตรงตามรูปแบบดีทีดี โดยจะส่งข้อมูลเทรดเดอร์นั้นกลับไป

exception ServiceTypeExists {
ServiceTypeName name;
};
	เอ็กซ์เซ็พชันนี้เกิดขึ้น เมื่อพบว่าชนิดของบริการมีอยู่ในระบบแล้ว โดยจะส่งชื่อชนิดของบริการกลับไป

exception IllegalServiceType {
ServiceTypeInfo st_info;
};
	เอ็กซ์เซ็พชันนี้เกิดขึ้น เมื่อพบว่าข้อมูลชนิดของบริการที่ส่งมามีข้อผิดพลาด หรือไม่ถูกต้องตามรูปแบบดีทีดี โดยจะส่งข้อมูลของชนิดของบริการกลับไป

exception IllegalURL {
SourceURL url;
};
	เอ็กซ์เซ็พชันนี้เกิดขึ้น เมื่อพบว่ายูอาร์แอล (URL) ที่ใช้อ้างอิงข้อมูลผิดรูปแบบ หรือไม่พบว่ามีอยู่จริง โดยจะส่งค่ายูอาร์แอลนั้นกลับไป

exception IllegalServiceInfo {
ServiceInfo s_info;
};
	เอ็กซ์เซ็พชันนี้เกิดขึ้น เมื่อพบว่าข้อมูลบริการที่ส่งมามีข้อผิดพลาด หรือไม่ถูกต้องตามรูปแบบดีทีดี โดยจะส่งข้อมูลของบริการกลับไป

exception UnknownOfferId {
OfferId oid;
};
	เอ็กซ์เซ็พชันนี้เกิดขึ้น เมื่อไม่พบ Offer Id ที่ใช้ระบุถึงบริการนั้นๆ โดยจะส่งค่า Offer Id นั้นกลับไป

<u>ตัวกระทำการของศูนย์จัดเตรียมบริการ</u>

- TraderName register_trader(
in SourceURL trader_profile
) raises (
TraderNameExists,
IllegalProfile
);

ตัวกระทำการที่ใช้สำหรับลงทะเบียนบริการเทรดเดอร์เข้าสู่ระบบของบริการเอสดีเอส โดยรับค่าพารามิเตอร์เป็นข้อมูลบริการเทรดเดอร์ ในที่นี้พัฒนาให้สามารถรับข้อมูลเป็นยูอาร์แอลที่อ้างถึงข้อมูลได้ โดยค่าที่ส่งกลับเป็นชื่อของบริการเทรดเดอร์นั้นๆ

- boolean remove_trader(
in TraderName trader_name
) raises (
UnknownTraderName
);

ตัวกระทำการสำหรับลบบริการเทรดเดอร์ออกจากระบบ ซึ่งใช้ชื่อบริการเทรดเดอร์เป็นชื่ออ้างอิงในการลบข้อมูล ค่าที่ส่งกลับเป็นชนิด boolean ค่าเป็นจริงเมื่อสามารถลบบริการได้ และเป็นเท็จเมื่อไม่สามารถลบบริการได้

```
- TraderName update_trader_profile(
                in TraderProfile tp
    ) raises (
        UnknownTraderName,
        IllegalProfile
    );
```

เป็นตัวกระทำการสำหรับการแก้ไขข้อมูลบริการเทรดเดอร์ โดยรับค่าพารามิเตอร์เป็นข้อมูลที่แก้ไขแล้วหรือยูอาร์แอลที่อ้างถึง ค่าที่ส่งกลับเป็นชื่อของบริการเทรดเดอร์

```
- ServiceTypeName add_service_type(
                                in TraderName trader_name,
    in SourceURL st_url
    ) raises (
    UnknownTraderName,
    ServiceTypeExists,
    IllegalServiceType,
    IllegalURL
    );
```

ตัวกระทำการสำหรับการเพิ่มข้อมูลชนิดบริการเข้าสู่บริการเอสดีเอส ค่าพารามิเตอร์คือชื่อของบริการเทรดเดอร์ที่เก็บชนิดของบริการนั้น และข้อมูลชนิดของบริการที่อ้างถึงด้วยยูอาร์แอล ค่าที่ส่งกลับคือชื่อชนิดของบริการนั้น

```
- boolean remove_service_type(
                in TraderName trader_name,
        in ServiceTypeName st_name
    ) raises (
        UnknownTraderName,
        UnknownServiceType
    );
```

ตัวกระทำการสำหรับการลบข้อมูลชนิดของบริการ ค่าพารามิเตอร์คือชื่อของบริการเทรดเดอร์ และชื่อชนิดของบริการ หากสามารถลบชนิดบริการได้ ค่าที่ส่งกลับจะเป็นค่าจริง และหากลบไม่ได้ก็จะส่งค่ากลับเป็นค่าเท็จ

```
- OfferId update_services(
                        in TraderName trader_name,
            in SourceURL s_url,
            in OfferId oid
        ) raises (
            UnknownTraderName,
            UnknownServiceType,
            IllegalServiceInfo,
             IllegalURL
        );
```

ตัวกระทำการสำหรับการแก้ไขบริการ ตัวกระทำการนี้รวมการเพิ่ม และการแก้ไขบริการไว้ด้วยกัน หากพบว่าบริการไม่มีอยู่ ก็จะถือว่าเป็นบริการใหม่และเพิ่มข้อมูลเข้าสู่ระบบ และหากพบว่ามีบริการนี้อยู่แล้ว ก็จะทำการปรับปรุงข้อมูลบริการเป็นข้อมูลล่าสุดที่ส่งไป พารามิเตอร์คือชื่อบริการเทรดเดอร์ ยูอาร์แอลที่อ้างถึงข้อมูลบริการและค่าเลขที่บริการของบริการที่ทำการแก้ไข โดยค่าที่ส่งกลับเป็นค่าเลขที่บริการของบริการนั้น

```
- boolean remove_offer(
                        in TraderName trader_name,
        in OfferId oid
    ) raises (
    UnknownTraderName,
    UnknownOfferId
    );
```

ตัวกระทำการสำหรับการลบบริการ ค่าพารามิเตอร์คือ ชื่อของบริการเทรดเดอร์ และค่าเลขที่บริการที่ต้องการลบ หากสามารถลบได้ก็จะส่งค่าจริงกลับไป หากลบไม่ได้ก็จะส่งค่าเท็จ

4.1.3 <u>ส่วนต่อประสานสำหรับการค้นหา</u>

```
interface SearchInterface : ServiceComponents {
        typedef string Keyword;
        typedef string Result;
        typedef boolean isURLresult;
        typedef string QLType;
        typedef sequence<QLType> QLTypeSeq;
        enum SearchType {
            TRADERS,
            SERVICE_TYPES,
            SERVICE_OFFERS,
            ALL
        };
        enum ResultFormat {
            MAXIMUM,
            MINIMUM,
           NO_TRANSFORM
        };
```

ส่วนกำหนดตัวแปรของส่วนต่อประสานสำหรับการค้นหา ซึ่งกำหนดชนิดของภาษาค้นหาโดยปริยายไว้เป็นตัวแปรชนิดสายอักขระ และชนิดของภาษาค้นหาทั้งหมดที่สนับสนุนเป็นลำดับของสายอักขระ ส่วนรูปแบบการค้นหา (SearchType) และ รูปแบบการแสดงผลประกาศไว้เป็นชนิดแจงนับ (Enumerate) ทั้งนี้เพื่อให้ผู้ใช้เลือกอย่างใดอย่างหนึ่งเท่านั้น

เอ็กซ์เซ็พชันของส่วนต่อประสานสำหรับการค้นหา

เอ็กซ์เซ็พชัน	**คำอธิบาย**

exception IllegalKeyword {
Keyword keyw;
};
	เอ็กซ์เซ็พชันนี้เกิดขึ้น เมื่อคำสำคัญที่ใช้ค้นหาไม่ถูกต้อง หรือผิดไวยากรณ์ในภาษาค้นหานั้นๆ ค่าที่ส่งกลับไปคือ คำสำคัญนั้น

Exception UnknownQLType {
QLType ql_type;
};
	เอ็กซ์เซ็พชันนี้เกิดขึ้น เมื่อค่าของตัวแปรชนิด QLType ที่ใช้ระบุชนิดของภาษาค้นหาไม่มีในระบบ โดยส่งค่าชนิดภาษาค้นหากลับไป

Exception UnknownSearchType {
SearchType search_type;
};
	เอ็กซ์เซ็พชันนี้เกิดขึ้น เมื่อการระบุชนิดข้อมูลที่จะค้นหาผิดพลาด โดยส่งกลับค่าชนิดข้อมูลนั้นไปด้วย

Exception UnknownServiceID {
ServiceID sid;
};
	เอ็กซ์เซ็พชันนี้เกิดขึ้น เมื่อไม่พบค่า Offer Id ของบริการนั้น ค่าที่ส่งไปด้วยคือ Offer Id ของบริการ

exception IllegalResultFormat {
ResultFormat rf;
};
	เอ็กซ์เซ็พชันนี้เกิดขึ้น เมื่อรูปแบบการแสดงผลที่ระบุเกิดข้อผิดพลาดหรือไม่ถูกต้อง โดยค่าที่ส่งไปด้วยคือค่าชนิดการแสดงผล

ลักษณะประจำของส่วนต่อประสานการค้นหา

- readonly attribute QLTypeSeq qltype_supported;

ใช้ระบุค่าของภาษาค้นหาที่ระบบสนับสนุน ซึ่งเป็นตัวแปรชนิดลำดับของสายอักขระ

- readonly attribute QLType default_qltype;

ใช้ระบุค่าโดยปริยายของภาษาค้นหา หากผู้ใช้ไม่ได้ระบุภาษาค้นหามาด้วย

ตัวกระทำการของส่วนต่อประสานการค้นหา

```
- Result search (
                    in Keyword keyw,
                    in QLType ql_type,
                    in SearchType search_type,
                    in isURLresult url_result,
                    in ResultFormat result_format
                ) raises (
```

```
                IllegalKeyword,
                UnknownQLType,
                UnknownSearchType,
                IllegalResultFormat
        );
```

ตัวกระทำการค้นหา เป็นตัวกระทำการหลักของส่วนต่อประสานสำหรับการค้นหา ซึ่งให้ผู้ใช้สามารถเรียกใช้เพื่อค้นหาข้อมูลต่างๆ ได้ โดยมีค่าพารามิเตอร์ดังนี้

- keyw คือคำสำคัญที่ใช้สำหรับค้นหา
- ql_type คือชนิดของภาษาค้นหาที่ผู้ใช้ระบุ
- search_type คือชนิดของข้อมูลที่จะทำการค้นหา เป็นตัวแปรชนิด SearchType ซึ่งสามารถระบุได้ 4 รูปแบบคือ

  TRADERS - ค้นหาเฉพาะข้อมูลเทรดเดอร์

  SERVICE_TYPES - ค้นหาเฉพาะชนิดของข้อมูล (รวมทั้งส่วนต่อประสาน)

  SERVICE_OFFERS - ค้นหาเฉพาะข้อมูลบริการ

  ALL - ค้นหาจากข้อมูลทั้งหมด

- url_result ใช้ระบุผลลัพธ์ที่ส่งกลับ เป็นตัวแปรชนิด boolean ถ้าถูกระบุเป็นค่าจริง ผลลัพธ์ที่ได้จะถูกเก็บไว้เป็นไฟล์และจะถูกส่งกลับในรูปแบบยูอาร์แอล ถ้าระบุเป็นค่าเท็จ ผลลัพธ์จะถูกส่งกลับเป็นสายอักขระ
- result_format ใช้ระบุรูปแบบผลลัพธ์ ซึ่งเป็นตัวแปรชนิด ResultFormat ซึ่งสามารถระบุได้ 3 รูปแบบ

  MAXIMUM - ผลลัพธ์จะถูกแปลงให้แสดงในแบบสมบูรณ์ โดยไม่มีการตัดข้อมูลส่วนใดออก

  MINIMUM - ผลลัพธ์จะถูกแปลงให้แสดงในแบบย่อ โดยเลือกเฉพาะส่วนสำคัญมาแสดงเท่านั้น

  NO_TRANSFORM - ผลลัพธ์ไม่มีการแปลงใดๆทั้งสิ้น เป็นผลลัพธ์ที่ได้จากเครื่องมือค้นหาโดยตรง

```
- TraderProfile describe_trader(
                in TraderName trader_name
    ) raises (
        IllegalKeyword,
        UnknownTraderName
    );
```

ตัวกระทำการสำหรับอธิบายรายละเอียดของบริการเทรดเดอร์ ทำการค้นหาข้อมูลบริการเทรดเดอร์ด้วยชื่อของบริการเทรดเดอร์นั้นๆ

```
- ServiceTypeInfo describe_service_type (
                in TraderName trader_name,
        in ServiceTypeName st_name
    ) raises (
        IllegalKeyword,
        UnknownServiceType
    );
```

ตัวกระทำการสำหรับอธิบายชนิดของบริการ ทำการค้นหาข้อมูลชนิดของบริการด้วยชื่อของบริการ เทรดเดอร์และชื่อชนิดของบริการ ทั้งนี้เพราะชื่อชนิดของบริการอาจซ้ำกันเมื่อข้อมูลมากจากเทรดเดอร์หลายๆแหล่ง ดังนั้นจึงต้องใช่ชื่อบริการเทรดเดอร์เพื่อมากำกับ

```
- ServiceInfo describe_service_offer (
                    in TraderName trader_name,
               in ServiceID sid
         ) raises (
              IllegalKeyword,
              UnknownServiceID
         );
```

ตัวกระทำการสำหรับอธิบายบริการ ทำการค้นหาของบริการด้วยชื่อของบริการเทรดเดอร์และค่าเลขที่บริการ ทั้งนี้เพราะในเทรดเดอร์หลายๆแหล่งค่าเลขที่บริการใดๆอาจซ้ำกัน

4.1.4 ส่วนต่อประสานส่วนเชื่อมโยงบริการ

```
interface FederatedPort : ServiceComponents {
        typedef string SDSName;
        typedef string SDSInfo;
        typedef sequence<SDSName> SDSSeq;
```

ส่วนเชื่อมโยงบริการกำหนดตัวแปรชื่อของบริการเอสดีเอสเป็นชนิดสายอักขระ เพื่อการอ้างถึงเมื่อต้องการติดต่อเชื่อมโยง และกำหนดตัวแปรลำดับของสายอักขระชื่อ SDSSeq เพื่อใช้สำหรับการเรียกดูรายการชื่อบริการเอสดีเอสที่มาเชื่อมต่อได้

เอ็กซ์เซ็พชันสำหรับส่วนเชื่อมโยงบริการ

เอ็กซ์เซ็พชัน	คำอธิบาย

Exception IllegalSDSInfo {
SDSInfo sds_info;
};
	เอ็กซ์เซ็พชันนี้เกิดขึ้น เมื่อค่าข้อมูลของบริการเอสดีเอสมีข้อผิดพลาด หรือไม่ถูกต้อง ค่าที่ส่งกลับไปคือข้อมูลบริการเอสดีเอส

Exception DuplicateSDSName {
SDSName sds_name;
};
	เอ็กซ์เซ็พชันนี้เกิดขึ้น เมื่อชื่อบริการเอสดีเอสมีอยู่แล้วในระบบ ค่าที่ส่งกลับไปด้วยคือชื่อบริการเอสดีเอสที่ใช้ลงทะเบียนเชื่อมโยง

Exception UnknownSDSName {
SDSName sds_name;
};
	เอ็กซ์เซ็พชันนี้เกิดขึ้น เมื่อชื่อบริการเอสดีเอสไม่มีอยู่ในระบบ เมื่อมีการเรียกดูข้อมูลบริการเอสดีเอสนั้นๆ ข้อมูลที่ส่งกลับไปด้วยคือชื่อบริการเอสดีเอสนั้น

ตัวกระทำการของส่วนเชื่อมโยงบริการ

```
- SDSName register_sds  (
                    in SDSName sds_name,
              in SDSInfo sds_info
        ) raises (
              DuplicateSDSName,
              IllegalSDSInfo
       );
```

ตัวกระทำการสำหรับการลงทะเบียนการเชื่อมโยงของบริการเอสดีเอส เพื่อเก็บข้อมูลของบริการเอสดีเอสนั้นเข้าสู่ระบบ สำหรับใช้ในการติดต่อขอเชื่อมโยงบริการ ค่าพารามิเตอร์คือ ชื่อบริการเอสดีเอส และข้อมูลบริการเอสดีเอส ผลที่ส่งกลับคือชื่อของบริการเอสดีเอสนั้น

```
- SDSName remove_sds (
                    in SDSName sds_name
        ) raises (
              UnknownSDSName
        );
```

ตัวกระทำการสำหรับลบข้อมูลบริการเอสดีเอสออกจากระบบ กรณีที่ยกเลิกการเชื่อมโยงบริการเอสดีเอสนั้นๆ สามารถส่งชื่อของบริการมาเพื่อร้องขอการลบออกจากการเชื่อมโยงได้

```
- SDSName update_sds (
                    in SDSName sds_name,
              in SDSInfo sds_info
         ) raises (
              UnknownSDSName,
              IllegalSDSInfo
         );
```

ตัวกระทำการแก้ไขข้อมูลการเชื่อมโยง หากมีการเปลี่ยนแปลงข้อมูลการเชื่อมโยง บริการเอสดีเอสนั้นๆ สามารถส่งข้อมูลใหม่มาปรับปรุงได้

```
- SDSInfo describe_sds (
                    in SDSName sds_name
       ) raises (
              UnknownSDSName
      );
```

ตัวกระทำการสำหรับอธิบายการเชื่อมโยงของบริการเอสดีเอส พารามิเตอร์คือชื่อบริการเอสดีเอส โดยจะส่งข้อมูลการเชื่อมโยงกลับไป

```
SDSSeq list_federation ( );
```

ตัวกระทำการสำหรับแสดงรายชื่อบริการเอสดีเอสที่เชื่อมโยงอยู่กับระบบ ค่าที่ส่งกลับไปเป็นชนิดตัวแปร SDSSeq ซึ่งเป็นค่าลำดับของสายอักขระที่เก็บชื่อของบริการเอสดีเอสที่มาเชื่อมโยงทั้งหมด

4.2 ต้นแบบเทรดเดอร์เอเจนท์

เทรดเดอร์เอเจนท์เป็นโปรแกรมคอร์บาอินเทอร์เซ็ฟเตอร์ ที่ดักการเรียกใช้ตัวกระทำการของบริการเทรดเดอร์ทั้งในส่วนของบริการ และคลังชนิดของบริการ ซึ่งกล่าวไว้แล้วในบทที่ 3 โปรแกรมอินเทอร์เซ็ฟเตอร์ในวิทยานิพนธ์นี้ใช้รูปแบบเซิร์ฟเวอร์อินเตอร์เซ็ฟเตอร์ ของวิสิโบรกเคอร์ [9] ซึ่งผูกติดอยู่กับบริการเทรดเดอร์ของ JacORB [19] (ซึ่งถูกแปลงให้ใช้งานได้กับวิสิโบรกเกอร์ และโมดูลของส่วนการแปลงคำอธิบายบริการแล้ว) โดยโปรแกรมอินเตอร์เซ็ฟเตอร์ของเอเจนท์ทำงานร่วมกับโปรแกรมคอร์บาที่ทำหน้าที่ติดต่อกับโมดูลของ CORBA/XML Transformer และบริการเอสดีเอส โดยใช้ชื่อว่า TraderAgentMgr ซึ่งมีโครงสร้างแพคเกจดังรูปที่ 4.2

รูปที่ 4.2 โครงสร้างแพคเกจของเทรดเดอร์เอเจนท์

การเรียกใช้งานเทรดเดอร์เอเจนท์ทำได้โดยใช้คำสั่ง

vbj -DORBservices=tagent jacorb.trading.TradingService TS_Ref

คำสั่งนี้เป็นการใช้งานบริการเทรดเดอร์ในรูปแบบของวิสิโบรกเคอร์ ด้วยคำสั่ง vbj โดยพารามิเตอร์ที่ระบุการเรียกใช้อินเทอร์เซ็ฟเตอร์ คือ -DORBservices ซึ่งค่าของพารามิเตอร์คือ ชื่อของแพคเก็จอินเตอร์เซ็ฟเตอร์ ในที่นี้คือ tagent ส่วนพารามิเตอร์อื่นเป็นการเรียกใช้บริการเทรดเดอร์ โดยรายละเอียดของโปรแกรม อินเตอร์เซ็ฟเตอร์ของเทรดเดอร์เอเจนท์เป็นดังนี้

- คลาส TraderAgentFactory และ Init เป็นคลาสที่จำเป็นสำหรับวิสิโบรกเคอร์ ซึ่ง TraderAgentFactory ใช้สำหรับสร้างวัตถุของ ServerTraderAgent เพื่อมารองรับการติดต่อจากไคลเอนต์ และ Init ใช้สำหรับการเริ่มต้นสร้างวัตถุของ TraderAgentFactory และกำหนดชื่ออ้างอิงสำหรับบริการเทรดเดอร์ด้วย รายละเอียดสามารถดูได้ที่ ผิดพลาด! ไม่พบแหล่งการอ้างอิง
- คลาส ServerTraderAgent ทำหน้าที่หลักสำหรับการดักข้อมูล ซึ่งต้องเป็นคลาสที่ถูกสร้างขึ้นโดยการสืบทอดมาจากคลาส com.visigenic.vbroker.interceptor.DefaultServerInterceptor ของ วิสิโบรกเคอร์ ซึ่งในวิทยานิพนธ์นี้ได้ใช้งาน 2 ตัวกระทำการ คือ receive_request() และ send_reply() ซึ่งมีรายละเอียดดังนี้

```
public org.omg.CORBA.portable.InputStream receive_request (
 com.visigenic.vbroker.GIOP.RequestHeader     hdr,
 org.omg.CORBA.ObjectHolder                   object,
 org.omg.CORBA.portable.InputStream           buf,
com.visigenic.vbroker.interceptor.Closure     closure)
```

ตัวกระทำการ receive_request() ทำหน้าที่ดักข้อมูลขณะที่ผู้ใช้เรียกใช้ตัวกระทำการของวัตถุเซิร์ฟเวอร์ ซึ่งก็คือบริการเทรดเดอร์นั่นเอง โดยชื่อตัวกระทำการที่เรียกใช้จะถูกเก็บอยู่ในตัวแปร hdr ซึ่งเป็นข้อมูลชนิดเฮดเดอร์ และจะสามารถดึงชื่อตัวกระทำการได้โดยการอ้างถึง hdr.operation ข้อมูลของพารามิเตอร์ที่ส่งมาจะเก็บอยู่ใน buf ซึ่งเป็นชนิด InputStream ของคอร์บา ซึ่งสามารถดึงข้อมูลออกมาดูได้ และไม่ว่าตัวกระทำการนั้นจะทำงานสำเร็จหรือไม่ สามารถดักข้อมูลที่เซิร์ฟเวอร์ส่งกลับด้วยตัวกระทำการ send_reply() ดังนี้

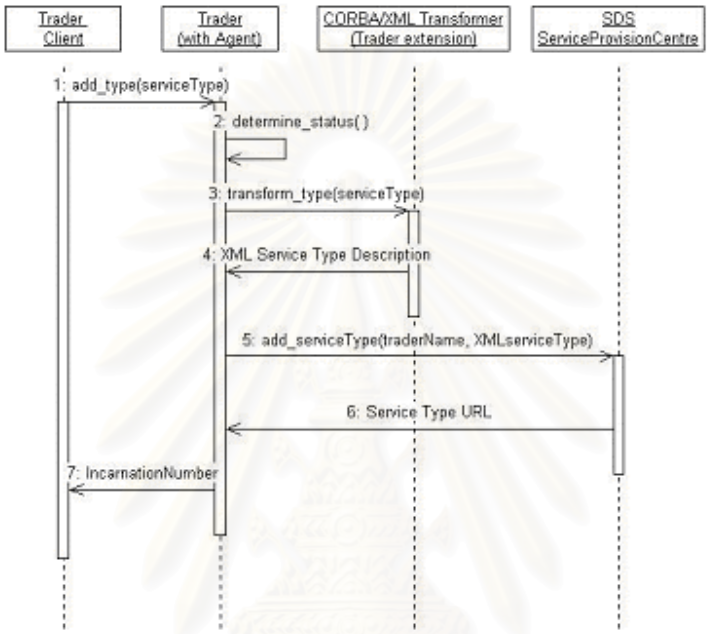
```
public org.omg.CORBA.portable.OutputStream send_reply(
 com.visigenic.vbroker.GIOP.RequestHeader reqHdr,
 com.visigenic.vbroker.GIOP.ReplyHeader hdr,
 org.omg.CORBA.Object target,
 org.omg.CORBA.portable.OutputStream buf,
 org.omg.CORBA.Environment env,
 com.visigenic.vbroker.interceptor.Closure closure)
```

ตัวกระทำการ send_reply() สามารถดักข้อมูลชื่อตัวกระทำการได้เช่นเดียวกัน และสามารถดักข้อความเอ็กซ์เซ็พชันหากเกิดขึ้นได้ ด้วยการเรียก env.exception() และหากการทำงานสำเร็จเป็นปกติ ก็สามารถอ่านข้อมูลที่ส่งกลับ ผ่านตัวแปร buf ได้เช่นเดียวกัน

ส่วนโมดูล TraderAgentMgr ประกอบด้วยส่วนต่อประสาน tagentMgr ซึ่งจะถูกเรียกใช้จาก ServerTraderAgent โดย tagentMgr จะทำหน้าที่ติดต่อกับส่วนการแปลงคำอธิบายบริการในเทรดเดอร์ และติดต่อกับศูนย์จัดเตรียมบริการในบริการเอสดีเอส โดยรายละเอียดของโมดูล TraderAgentMgr มีดังนี้

```
module TraderAgentMgr
{
interface tagentMgr
 {
    typedef string TypeName;
    typedef string OfferID;
    oneway void add_type(in TypeName tname);
    oneway void remove_type(in TypeName tname);
    oneway void update_offer(in OfferID oid);
    oneway void remove_offer(in OfferID oid);
 };
};
```

จะเห็นว่าตัวกระทำการของส่วนต่อประสานที่กำหนดขึ้นเป็นแบบ oneway ทั้งนี้เพราะต้องการให้การบริการเทรดเดอร์ทำงานได้อย่างต่อเนื่อง เพราะตัวกระทำการแบบ oneway จะไม่เกิดเอ็กซ์เซ็พชันและไม่ต้องรอผลตอบกลับ ซึ่งจะไม่เป็นการเพิ่มภาระงานให้กับเทรดเดอร์มากนัก ส่วนผลจากการทำงานของ TraderAgentMgr จะถูกบันทึกโดยคลาสสำหรับการเขียนข้อมูลการใช้งาน (Log) ซึ่งมีชื่อว่า LogWriter ทั้งนี้เพื่อใช้ในการตรวจสอบผลการทำงานในภายหลัง



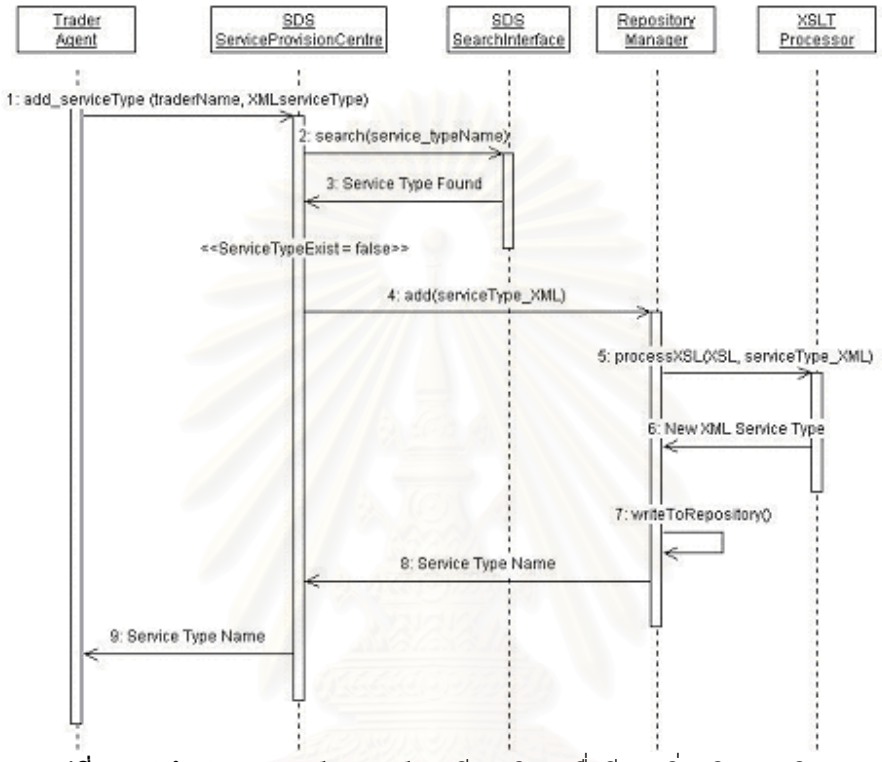
รูปที่ 4.3 ผังลำดับเหตุการณ์ของเทรดเดอร์เอเจนท์สำหรับตัวกระทำการ add_type

รูปที่ 4.3 เป็นผังลำดับเหตุการณ์ของตัวอย่างการเพิ่มข้อมูลชนิดของบริการในเทรดเดอร์ (มีการเรียกใช้ตัวกระทำการ add_type) เมื่อบริการเทรดเดอร์ที่มีเอเจนท์รวมอยู่ด้วย ถูกเรียกใช้ตัวกระทำการเพิ่มชนิดของบริการ เทรดเดอร์เอเจนท์จะตรวจดูสถานะของการเรียกใช้งาน หากไม่มีเอ็กซ์เซ็พชันเกิดขึ้น เทรดเดอร์เอเจนท์จะเรียกใช้ตัวกระทำการ transform_type(serviceType) ของส่วนการแปลงคำอธิบายบริการ เพื่อร้องขอการแปลงข้อมูลชนิดบริการนั้นเป็นเอกสารเอ็กซ์เอ็มแอล จากนั้นก็จะติดต่อกับบริการเอสดีเอสในส่วนของศูนย์จัดเตรียมบริการ โดยเรียกใช้ตัวกระทำการ add_serviceType(traderName, XMLserviceType) ซึ่งจะทำการจัดเก็บข้อมูลลงในคลังเก็บข้อมูลของบริการเอสดีเอส

ข้อมูลของบริการหรือชนิดของบริการที่ถูกแปลงแล้ว จะถูกจัดเก็บไว้ในไดเร็คทรอรีที่กำหนด ซึ่งอ้างถึงงานวิจัย [4] โดยใช้ชื่อว่า xmldb อยู่ภายใต้ไดเร็คทรอรี db ในรากของบริการเทรดเดอร์ ซึ่งจัดเก็บตามชื่อชนิดของบริการ หรือชื่อชนิดของบริการรวมกับชื่ออ้างอิงบริการ ตามด้วย .xml ซึ่งผู้ดูแลบริการเทรดเดอร์สามารถใช้เพื่อการแก้ไขข้อผิดพลาดจากการส่งได้ (ซึ่งสามารถดูได้จากไฟล์บันทึกการทำงานของเอเจนท์)

4.3 ต้นแบบศูนย์จัดเตรียมบริการ

ศูนย์จัดเตรียมบริการทำหน้าที่เป็นส่วนต่อประสานที่รองรับการจัดการข้อมูลบริการ เช่น การเพิ่ม ลบ หรือแก้ไขข้อมูลบริการ ส่วนต่อประสานที่ใช้สำหรับพัฒนาศูนย์จัดเตรียมบริการมีชื่อว่า ServiceProvisionCentre โดยต้นแบบที่พัฒนามีผังลำดับเหตุการณ์ดังรูปที่ 4.4



รูปที่ 4.4 ผังลำดับเหตุการณ์ของศูนย์จัดเตรียมบริการเมื่อมีการเพิ่มชนิดของบริการ

รูปที่ 4.4 เป็นตัวอย่างผังลำดับเหตุการณ์ของศูนย์จัดเตรียมบริการ เมื่อเทรดเดอร์เอเจนท์มีการเรียกใช้ตัวกระทำการเพิ่มข้อมูลชนิดของบริการคือ add_serviceType โดยส่งพารามิเตอร์ไปเป็นชื่อเทรดเดอร์ และข้อมูลชนิดของบริการที่ผ่านการแปลงเป็นเอกสารเอ็กซ์เอ็มแอลแล้ว จากนั้นศูนย์จัดเตรียมบริการจะติดต่อกับส่วนต่อประสานสำหรับการค้นหา เพื่อตรวจสอบดูว่าชื่อชนิดของบริการนั้นมีอยู่แล้วหรือไม่ ถ้ายังไม่มีก็จะทำงานต่อโดยติดต่อเรียกใช้ตัวจัดการคลังเก็บข้อมูล (Repository Manager) ให้ทำการเพิ่มข้อมูลโดยจะทำการแปลงข้อมูลให้อยู่ในรูปแบบที่เหมาะสมก่อนการจัดเก็บ จากนั้นหากการทำงานถูกต้องก็จะส่งค่าชื่อชนิดของบริการกลับไป หากผิดพลาดจะส่งกลับเป็นค่า NULL

เนื่องจากต้นแบบบริการเอสดีเอสสนับสนุนการทำงานของภาษาค้นหาเอ็กซ์คิวแอล (ดูในหัวข้อ 4.4) รูปแบบข้อมูลจึงต้องเป็นไปตามเครื่องมือค้นหานั้นๆ ในงานวิจัยนี้ใช้ GMD-IPSI [20] ซึ่งจัดเก็บข้อมูลแบบถาวร ในรูปแบบพีดีโอเอ็ม (Persistence DOM) ส่วนการจัดการกับเอกสารเอ็กซ์เอ็มแอลทั้งหมดใช้ตัวแจงผ่านของ [21] ซึ่งสามารถใช้เอพีไอได้ทั้งดีโอเอ็มและเอ็กซ์เอสแอลที

4.4 ต้นแบบส่วนต่อประสานสำหรับการค้นหา

บริการเอสดีเอสพัฒนาส่วนต่อประสานสำหรับการค้นหา ให้รองรับการสอบถามสำหรับภาษาเอ็กซ์เอ็มแอล 1 ภาษา คือเอ็กซ์คิวแอล และการค้นหาแบบใช้คำสำคัญ โดยมีรายละเอียดดังนี้

1. ภาษาสอบถามเอ็กซ์คิวแอล

งานวิจัยนี้ใช้ GMD-IPSI ซึ่งเป็นโปรแกรมค้นหาที่ได้รับการพัฒนาให้มีฟังก์ชันการทำงานของภาษาเอ็กซ์คิวแอลครบถ้วน และสามารถเก็บข้อมูลแบบถาวรได้ แต่เนื่องจากบริการ เอสดีเอสยังต้องการคงคุณสมบัติการค้นหาของบริการเทรดเดอร์ให้มากที่สุด ดังนั้นจึงต้องพัฒนาส่วนที่ภาษาสอบถามเอ็กซ์คิวแอลทำไม่ได้ อันได้แก่ฟังก์ชันการคำนวณทางคณิตศาสตร์ซึ่งจำเป็นต้องใช้การสอบถามจากเทรดเดอร์ดยใช้ภาษาบังคับ การเพิ่มเติมในส่วนนี้จำทำในรูปแบบของฟังก์ชันบัพ (Node Function) ซึ่งเป็นคุณสมบัติที่สามารถพัฒนาเพิ่มเติมได้ใน GMD-IPSI โดยมีตัวอย่างดังนี้

```
XQL.addNodeFunction("sum",
      new XQLNodeFunction() {
          public Object call(int refNodeIndex, XQLResult refNodeSet, Object[] args) {
        double d = 0.0;
        for(int i=0; i<args.length; i++)
        if (args[i] instanceof XQLExpression) {
         XQLResult r = new XQLResult();
        ((XQLExpression)args[i]).eval(refNodeIndex, refNodeSet, r);
        for(int j=0; j<r.getLength(); j++)
          d = d + XQL.number(r.getItem(j));
         } else {
         d = d + XQL.number(args[i]);
         }
        return new Double(d);
        }
      }
);
```

จากตัวอย่างข้างต้น เป็นการกำหนดฟังก์ชันการรวม (Sum) ซึ่งทำหน้าที่หาผลรวมของข้อมูลที่เป็นตัวเลขหรือข้อความสอบถามของเอ็กซ์คิวแอล ตัวอย่างเช่น การค้นหาบริการที่ชื่อชนิดของบริการเท่ากับ Calculator และมีผลรวมของค่าคุณสมบัติ Delay_Time และ Process_Time น้อยกว่า 800 โดยสามารถเขียนเป็นนิพจน์ของเอ็กซ์คิวแอลได้ดังนี้

```
ServiceOfferDescription[(OfferType[(@Name = 'Calculator')])
  $and$ sum((Property[(@Name = Delay_Time')]/@Value),
      Property[(@Name= ' Process_Time ')]/@Value)) < 800]
```

การพัฒนาส่วนต่อประสานสำหรับการค้นหาของบริการเอสดีเอส กำหนดให้มีฟังก์ชันการคำนวณเพิ่มขึ้น ได้แก่ การรวม (sum) การลบ (sub) การคูณ (mul) การหาร (div) ซึ่งทั้งหมดไม่มีอยู่ในข้อกำหนดของเอ็กซ์คิวแอล

2. การค้นหาแบบใช้คำสำคัญ

เนื่องจากบริการเอสดีเอสสนับสนุนการค้นหาแบบใช้คำสำคัญ ซึ่งใช้คุณสมบัติของเครื่องมือค้นหาเอ็กซ์คิวแอล การทำงานแบ่งออกได้เป็น 2 ส่วนคือ

(1) การแยกคำสำคัญออกจากกัน และการจัดรูปแบบให้เป็นนิพจน์ของเอ็กซ์คิวแอล การแยกคำสำคัญเรียกใช้โมดูลของนิพจน์ปรกติ (Regular Expression) มาทำการแยกคำด้วยนิพจน์ดังนี้

(\\-?'[^']+')|(\\-?\\\"[^\\\"]+\\\")| (\\-?[^\\s]+)

นิพจน์นี้ใช้สำหรับแยกคำสำคัญในลักษณะเดียวกับโปรแกรมค้นหา โดยมีรายละเอียดในแต่ละส่วนดังนี้

นิพจน์	ความหมาย
(\\-?'[^']+')	จับคู่คำที่มีเครื่องหมาย – นำหน้า 0 หรือ 1 ตัว และตามด้วยเครื่องหมาย ' ตามด้วยตัวอักษรใดๆ (ที่ไม่ใช่เครื่องหมาย ') และปิดท้ายด้วยเครื่องหมาย ' การใช้งาน นิพจน์รูปแบบนี้ใช้สำหรับค้นหาคำที่มีรูปแบบตรงกับคำที่ระบุแบบแน่นอน (Exact Match) เช่น จะค้นหาเอกสารที่มีคำว่า 'Laser Printer' แบบนี้เท่านั้น หากมีเครื่องหมายลบนำหน้าจะหมายความว่า จะไม่เลือกเอาเอกสารที่มีคำที่ตรงกับที่ระบุ (Exclude) เช่น -'HP'
(\\-?\\\"[^\\\"]+\\\")	ลักษณะเดียวกับนิพจน์ข้างต้น แต่อยู่ภายใต้เครื่องหมาย "
(\\-?[^\\s]+)	อักษรใดๆ (ที่ไม่ใช่ช่องว่าง) มีเครื่องหมาย – นำหน้า 0 หรือ 1 ตัว การใช้งาน นิพจน์รูปแบบนี้ใช้สำหรับค้นหาคำที่มีส่วนหนึ่งมีรูปแบบตรงกับคำที่ระบุ (Substring) โดยสามารถมีตัวอักษรเล็ก-ใหญ่ต่างกันได้ เช่นคำว่า hotel สามารถจับคู่ได้กับคำว่า Hotel หรือ hotels เป็นต้น และแต่ละนิพจน์สามารถมีเครื่องหมายลบนำหน้าได้เช่นเดียวกัน

(2) การรวมคำสำคัญให้เป็นนิพจน์เอ็กซ์คิวแอล หลังจากที่ได้คำสำคัญโดยจัดกลุ่มตามรูปแบบข้างต้นมาแล้ว โปรแกรมจะทำการจัดรูปแบบคำสำคัญให้เป็นรูปแบบของภาษาเอ็กซ์คิวแอล เพื่อส่งไปยังเครื่องมือค้นหา โดยมีรูปแบบดังนี้

- นิพจน์ระบุการค้นหาข้อมูลภายในป้ายระบุ และภายในลักษณะประจำ ซึ่งจะค้นหาในข้อมูลที่ผู้ใช้งานกำหนด
- สำหรับการค้นหาแบบแน่นอน (คำที่อยู่ภายใต้เครื่องหมาย "" หรือ '') จะใช้เครื่องหมาย = ในการจับคู่
- สำหรับการค้นหาแบบคำปรกติ (คำที่ไม่อยู่ภายใต้เครื่องหมาย "" หรือ '') จะใช้เครื่องหมาย $icontains$ ในการจับคู่
- สำหรับเครื่องหมายลบ (-) จะใช้เครื่องหมาย not ระบุหน้านิพจน์ที่สร้างขึ้น

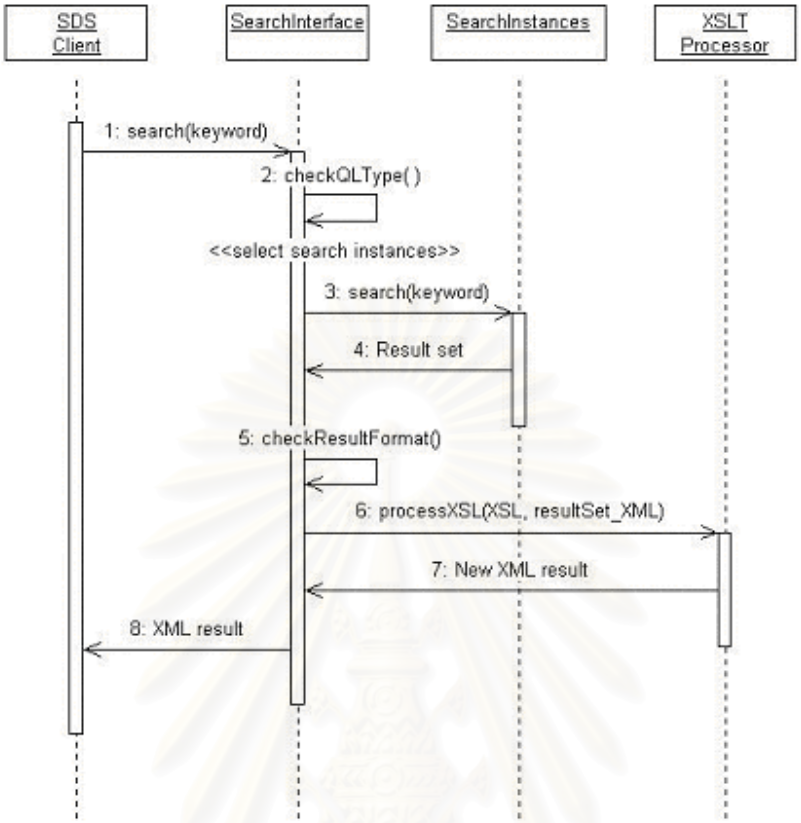
ตัวอย่างเช่น

การค้นหาด้วยคำสำคัญ "Laser Printer" Color –HP

จะถูกแปลงเป็น

```
ServiceOfferDescription [
(((. = "Laser Printer") $or$ (.//@* = "Laser Printer"))
$and$ ((. $icontains$ 'Color') $or$ (.//@* $icontains$ 'Color')))
$and$ $not$ ((. $icontains$ 'HP') $or$ (.//@* $icontains$ 'HP')) ]
```

รูปที่ 4.5 แสดงตัวอย่างผังลำดับเหตุการณ์ของส่วนต่อประสานสำหรับการค้นหา เมื่อมีการเรียกใช้ตัวกระทำการ search ส่วนต่อประสานสำหรับการค้นหาจะตรวจสอบจากพารามิเตอร์ที่ส่งมาว่าผู้ใช้ระบุภาษาค้นหาใด หลังจากนั้นจะเรียกตัวกระทำการ search ของเครื่องมือค้นหานั้นๆ ส่วนต่อประสานสำหรับการค้นหาจะทำการแปลงรูปผลลัพธ์ที่ได้กลับมาให้อยู่ในรูปแบบที่ผู้ใช้ระบุ หลังจากนั้นผลลัพธ์สุดท้ายจะถูกส่งกลับไปยังผู้ใช้

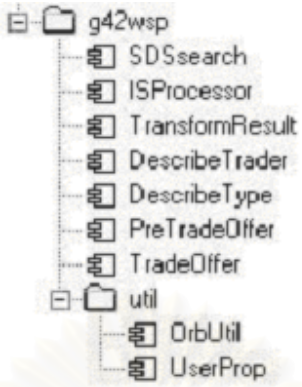


รูปที่ 4.5 ผังลำดับเหตุการณ์ของส่วนต่อประสานสำหรับการค้นหา

เนื่องจากการพัฒนาให้สามารถรองรับภาษาสอบถามได้หลายภาษา ส่วนต่อประสานสำหรับการค้นหาจึงกำหนดส่วนต่อประสานของ SearchInstances ขึ้นมาเพื่อ พัฒนาให้เป็นวัตถุที่เชื่อมต่อกับเครื่องมือค้นหาสำหรับแต่ละภาษา ซึ่งส่วนต่อประสานสำหรับการค้นหาจะทำการสร้างวัตถุของ SearchInstances แต่ละตัวขึ้นมา เพื่อเตรียมไว้สำหรับผู้ใช้ที่เรียกใช้ภาษานั้นๆ

4.5 ส่วนต่อประสานสำหรับการค้นหาในส่วนผู้ใช้งานผ่านเกณฑ์วิธีเอชทีทีพี

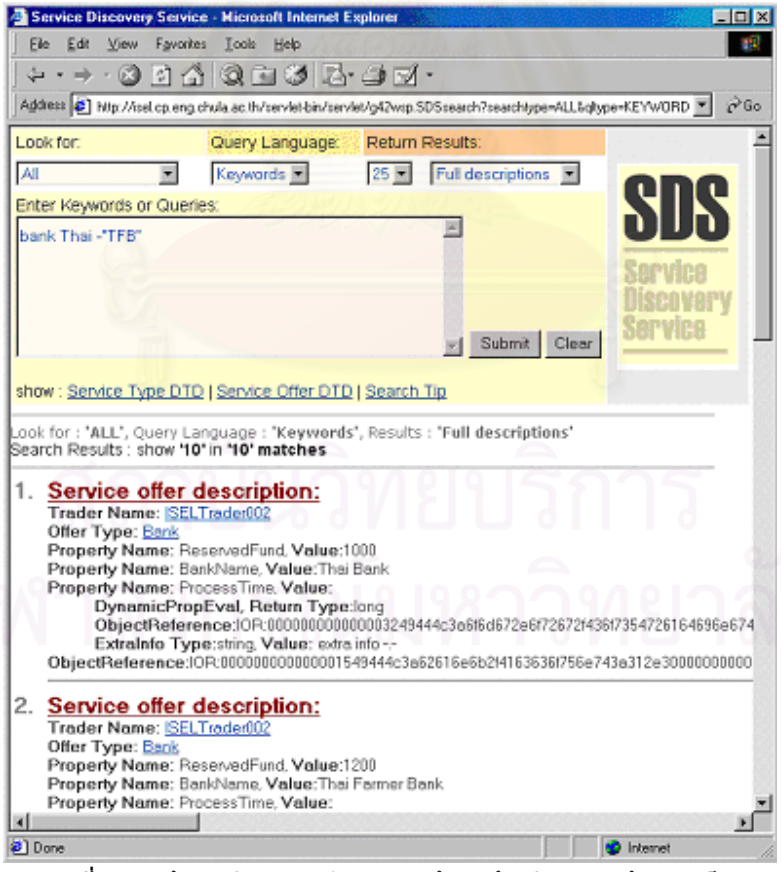
ในการพัฒนาส่วนต่อประสานสำหรับการค้นหาให้สามารถใช้ได้กับผู้ใช้ผ่านโปรแกรมค้นหาผ่านเว็บ หรือในรูปแบบเดียวกับโปรแกรมค้นหา จะใช้จาวาเซิร์ฟเล็ตในการติดต่อกับส่วนต่อกับส่วนต่อประสานสำหรับการค้นหาและใช้การประมวลผลการแปลงรูปผลลัพธ์ให้เป็นเอกสารเอชทีเอ็มแอล ผลลัพธ์จะถูกส่งมาแสดงทางโปรแกรมค้นผ่านเว็บด้วยโปรแกรมเจเอสพี โดยรูปแบบเพคเกจของโปรแกรมในส่วนจาวาเซิร์ฟเล็ตเป็นดังนี้



รูปที่ 4.6 แพคเกจของโปรแกรมจาวาเซิร์ฟเล็ตของส่วนต่อประสานสำหรับการค้นหา

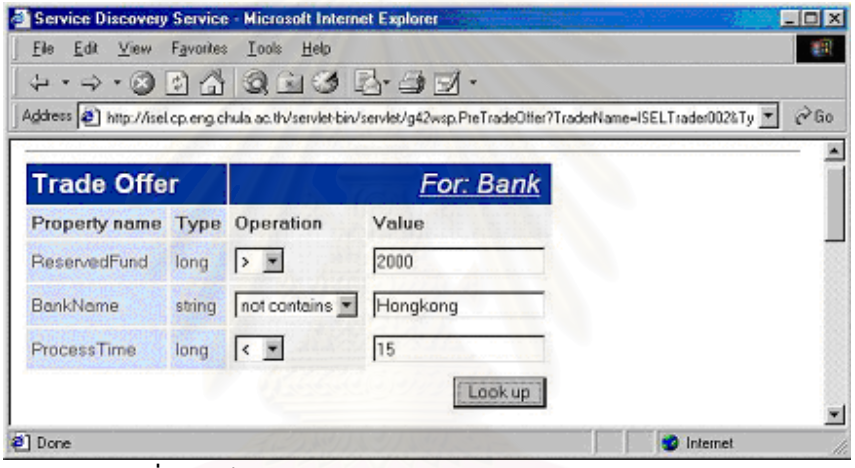
แพคเกจในรูปที่ 4.6 มีรายละเอียดดังนี้

- SDSsearch เป็นคลาสเซิร์ฟเล็ตหลักที่ทำหน้าที่รับข้อมูลจากแบบฟอร์มรับข้อมูล ทั้งจากโปรแกรมเอชทีเอ็มแอลเริ่มต้น หรือโปรแกรมเจเอสพีหลังจากการค้นหา โดยเรียกตัวประมวลผลต่างๆ ให้ทำงาน แล้วส่งผลไปยังโปรแกรม SearchResults.jsp เพื่อแสดงผล และรอรับข้อมูลสำหรับทำการค้นหาต่อไป ดังตัวอย่างดังรูปที่ 4.7



รูปที่ 4.7 หน้าต่างโปรแกรมสำหรับการค้นหาด้วยโปรแกรมค้นผ่านเว็บ

- ISProcessor ทำหน้าที่ติดต่อกับส่วนต่อประสานสำหรับการค้นหา ซึ่งมีลักษณะเป็นโปรแกรมคอร์บาไคลเอนต์ โดยมีตัวกระทำการที่สามารถเรียกใช้การค้นหาข้อมูลในคลังเก็บเอกสารเอ็กซ์เอ็มแอลได้
- TransformResult ใช้สำหรับแปลงผลลัพธ์ให้เป็นเอกสารเอชทีเอ็มแอล รองรับการแสดงผลทั้งแบบสมบูรณ์และแบบย่อ การทำงานส่วนนี้เรียกใช้การทำงานของตัวประมวลผลเอ็กซ์เอสแอลที โดยเอกสารเอ็กซ์เอสแอลสำหรับการแปลงเป็นเอชทีเอ็มแอลแสดงใน ภาคผนวก ข
- DescribeTrader ทำหน้าที่อธิบายรายละเอียดของบริการเทรดเดอร์ โดยส่งผลลัพธ์ให้กับ DescribeTrader.jsp
- DescribeType ทำหน้าที่อธิบายรายละเอียดของชนิดของบริการ โดยส่งผลลัพธ์ให้กับDescribeType.jsp
- PreTradeOffer ทำหน้าที่รองรับการค้นหาบริการ โดยทำการตรวจสอบชนิดของคุณสมบัติของชนิดของบริการที่ต้องการค้นหา และแสดงหน้าจอให้ผู้ใช้กรอกข้อมูลด้วยโปรแกรม PreTradeOffer.jsp ดังรูปที่ 4.8

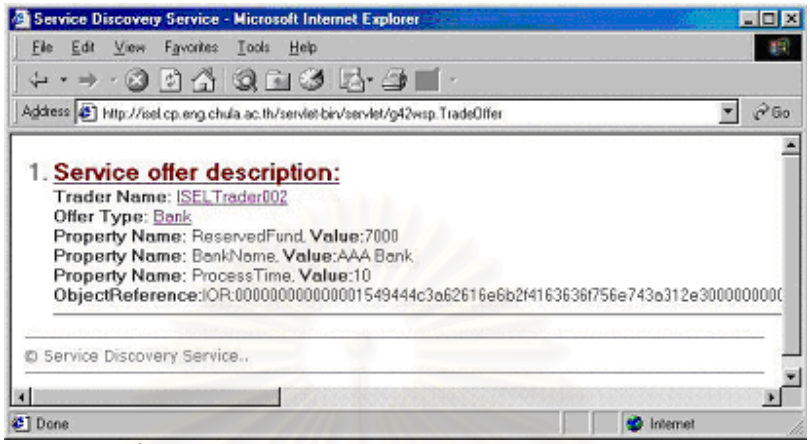


รูปที่ 4.8 หน้าต่างแสดงการทำงานของโปรแกรม PreTradeOffer

- TradeOffer รองรับข้อมูลจาก PreTradeOffer และทำการค้นสร้างนิพจน์เอ็กซ์คิวแอล เช่นดังรูปที่ 4.8 สามารถแปลงได้เป็นดังนี้

```
//ServiceOfferDescription[(@TraderName = 'ISELTrader002')
 $and$ (OfferType[(@Name = 'Bank')])
 $and$ (Property[(@Name= 'ReservedFund')
        $and$ (@Value > 2000)])
 $and$ $not$ (Property[(@Name='BankName')
        $and$ (@Value $icontains$ 'Hongkong')])
 $and$ (Property[(@Name= 'ProcessTime')
        $and$ (@Value < 15)])]
```

จากนั้นจะส่งนิพจน์ดังกล่าวไปยังโปรแกรม ISProcessor เพื่อจับคู่บริการตามที่กำหนด และรอรับผลลัพธ์กลับมา จากนั้นส่งต่อให้กับโปรแกรม TradeOffer.jsp ดังรูปที่ 4.9



รูปที่ 4.9 หน้าต่างแสดงผลการทำงานของโปรแกรม TradeOffer

รายละเอียดของหน้าจอการทำงานต่างๆสามารถดูเพิ่มเติมได้ในบทที่ 5

บทที่ 5
การทดสอบการใช้งานและคุณสมบัติของบริการเอสดีเอส

ในบทนี้จะกล่าวถึงรายละเอียดของการทดสอบการทำงานของบริการเอสดีเอสในส่วนต่างๆ คือ เทรดเดอร์เอเจนท์ ศูนย์จัดเตรียมบริการ และส่วนต่อประสานสำหรับการค้นหา และรวมทั้งคุณสมบัติการค้นหาเมื่อเปรียบเทียบกับบริการเทรดเดอร์ โดยรายละเอียดการทดสอบมีดังนี้

5.1 สภาวะที่ใช้ในการทดสอบ

1. เครื่องเซิร์ฟเวอร์ Intel PII 350 แรม 128 เมกะไบต์ ระบบปฏิบัติการลินุกซ์ (Linux) รุ่น 2.2.13
2. เครื่องมือพัฒนาภาษาจาวา JDK 1.2.2
3. วิสิโบรกเกอร์รุ่น 3.4 สำหรับภาษาจาวา
4. บริการเทรดเดอร์จำนวน 3 ตัว ซึ่งเป็นของ JacORB [19] ที่แก้ไขสำหรับใช้งานกับวิสิโบรกเกอร์และรวมโมดูลของส่วนการแปลงคำอธิบายบริการแล้ว เทรดเดอร์ทั้งสามนี้จะใช้คลังส่วนต่อประสาน 1 ตัว ร่วมกันซึ่งเก็บส่วนต่อประสานจำนวน 24 ชุด บริการเทรดเดอร์แต่ละตัวจำมีจำนวนชนิดของบริการและบริการดังตารางที่ 5.1
5. บริการเซิร์ฟเล็ตและเจเอสพี Tomcat รุ่น 3.1
6. บริการเอสดีเอส 1 ตัว
7. โปรแกรมค้นผ่านเว็บ Internet Explorer รุ่น 5.5 บนระบบปฏิบัติการ Windows98 สำหรับผู้ใช้ผ่านเว็บ

ตารางที่ 5.1 จำนวนข้อมูลชนิดของบริการและบริการสำหรับการทดสอบ

เทรดเดอร์	จำนวนชนิดของบริการ	จำนวนบริการ
ISELTrader001	11	110
ISELTrader002	13	142
ISELTrader003	2	10

ลำดับขั้นตอนการเริ่มต้นโปรแกรมและบริการต่างๆ แสดงไว้ในภาคผนวก ค

5.2 การทดสอบเทรดเดอร์เอเจนท์

การทดสอบเทรดเดอร์เอเจนท์กระทำในส่วนต่างๆดังนี้

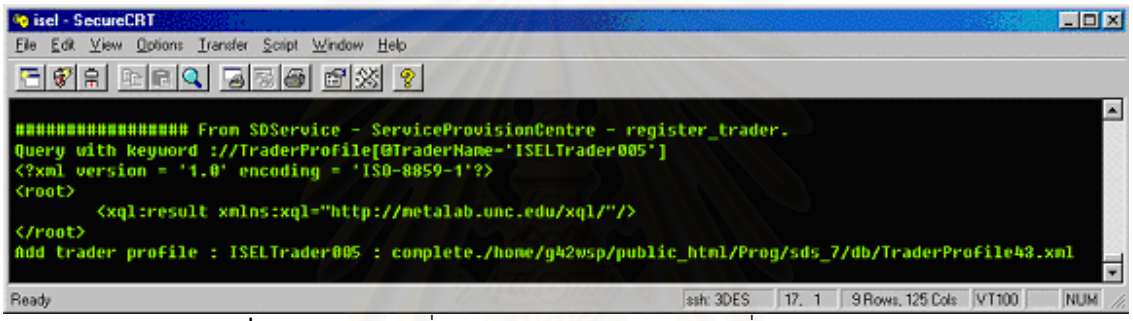
5.2.1 การทดสอบการลงทะเบียนของบริการเทรดเดอร์

- การเพิ่มบริการเทรดเดอร์เข้าสู่ระบบ

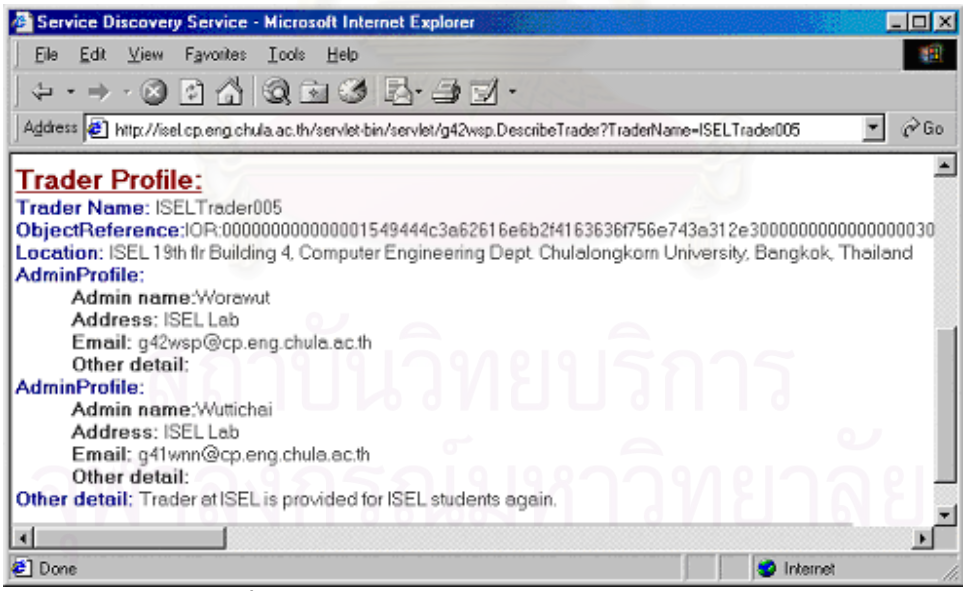
โปรแกรมสำหรับการลงทะเบียนทำการเรียกใช้ศูนย์จัดเตรียมบริการ โดยสามารถเข้าถึงจากส่วนต่อประสานสำหรับการค้นหาที่ทำการ narrow วัตถุอ้างอิงของบริการเอสดีเอส โดยพารามิเตอร์ของตัวกระทำการ register_trader() จะเป็นยูอาร์แอลของไฟล์ข้อมูลบริการเทรดเดอร์ดังนี้

```
SDService.SearchInterface ss = SDService.SearchInterfaceHelper.narrow(obj);
SDService.ServiceProvisionCentre spc = ss.ServiceProvisionCentre_if();
String result = spc.register_trader("http://isel.cp.eng.chula.ac.th/~g42wsp/ts/TraderProfile.xml");
```

หากโปรแกรมทำงานสำเร็จจะส่งค่าชื่อบริการเทรดเดอร์กลับ ผลทางฝั่งบริการเอสดีเอสและผลจากการใช้งานโปรแกรมอธิบายบริการเทรดเดอร์จะเป็นดังรูปที่ 5.1 และ 5.2



รูปที่ 5.1 ผลจากการเพิ่มบริการเทรดเดอร์เข้าสู่ระบบในฝั่งบริการเอสดีเอส



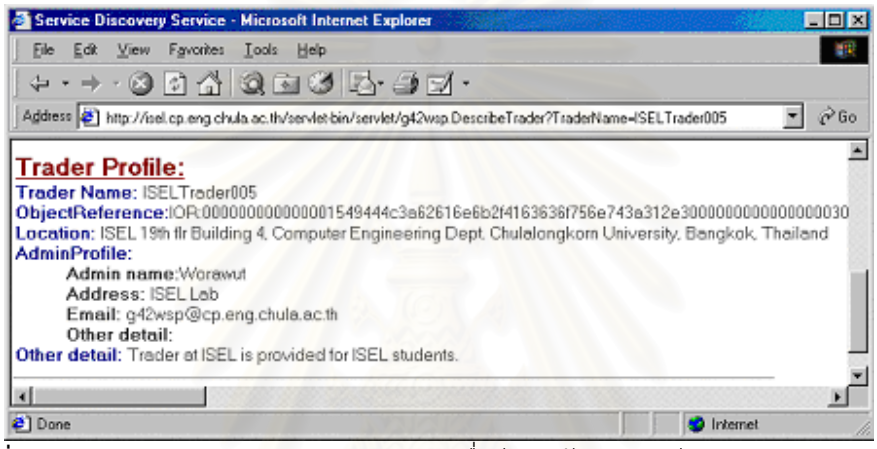
รูปที่ 5.2 ผลจากการใช้โปรแกรมอธิบายบริการเทรดเดอร์

- การแก้ไขข้อมูลบริการเทรดเดอร์

การแก้ไขข้อมูลบริการเทรดเดอร์จะทำการเรียกใช้ศูนย์จัดเตรียมบริการ ด้วยตัวกระทำการ update_trader_profile() โดยพารามิเตอร์ของตัวกระทำการจะเป็นยูอาร์แอลของไฟล์ข้อมูลบริการเทรดเดอร์ที่แก้ไขใหม่ รูปแบบการเรียกตัวกระทำการเป็นดังนี้

```
String result =
    spc. update_trader_profile("http://isel.cp.eng.chula.ac.th/~g42wsp/ts/new_TraderProfile.xml");
```

ผลจากการทำงานจะเป็นลักษณะเดียวกับการเพิ่มบริการเทรดเดอร์ แต่ข้อมูลจากโปรแกรมอธิบายบริการเทรดเดอร์จะเปลี่ยนไปเป็นข้อมูลใหม่ ดังรูปที่ 5.3



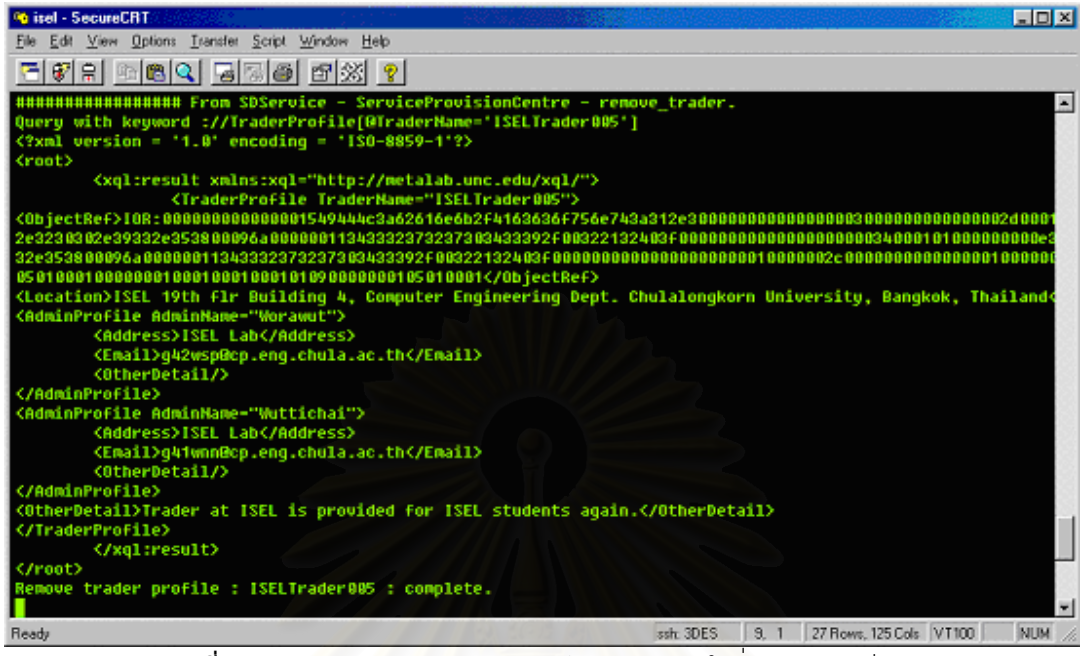
รูปที่ 5.3 ผลจากโปรแกรมอธิบายบริการเทรดเดอร์เมื่อมีการแก้ไขรายละเอียดของบริการเทรดเดอร์

- การลบบริการเทรดเดอร์ออกจากระบบ

การลบบริการเทรดเดอร์ออกจากระบบจะทำการเรียกใช้ศูนย์จัดเตรียมบริการด้วยตัวกระทำการ remove_trader() โดยพารามิเตอร์จะเป็นชื่อของบริการเทรดเดอร์ รูปแบบการใช้งานจะเป็นดังนี้

```
if(spc.remove_trader("ISELTrader005")) System.out.println("can remove");
```

ผลจากการลบบริการเทรดเดอร์จะเป็นชนิด boolean ซึ่งถ้าลบได้ค่าจะเป็นจริง ถ้าไม่ได้ค่าที่ส่งกลับจะเป็นเท็จ โดยผลจากการลบข้อมูลเทรดเดอร์ในฝั่งบริการเอสดีเอสเป็นดังรูปที่ 5.4



รูปที่ 5.4 ผลจากการลบบริการเทรดเดอร์ออกจากระบบในฝั่งบริการเอสดีเอส

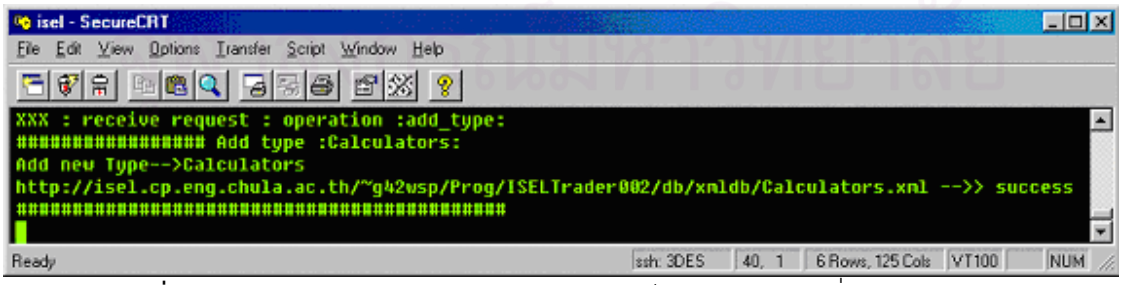
ผลจากการลบข้อมูลบริการเทรดเดอร์จะแสดงรายละเอียดของข้อมูลบริการเทรดเดอร์ที่ค้นหาพบ และแสดงสถานะของผลการทำงาน

5.2.2 การทดสอบการเปลี่ยนแปลงข้อมูลที่เกี่ยวกับบริการของเทรดเดอร์จากการเรียกใช้ตัวกระทำการต่างๆ

การเรียกใช้ตัวกระทำการที่มีผลต่อการเปลี่ยนแปลงข้อมูลเกี่ยวกับบริการ จะทำให้เทรดเดอร์เอเจนท์ทำงานดังที่กล่าวไว้ในบทที่ 3 การทดสอบจะเป็นการเรียกใช้ตัวกระทำการเหล่านี้ แล้วทำการตรวจสอบผลการทำงาน โดยแบ่งการทดสอบด้วยตัวกระทำการดังนี้

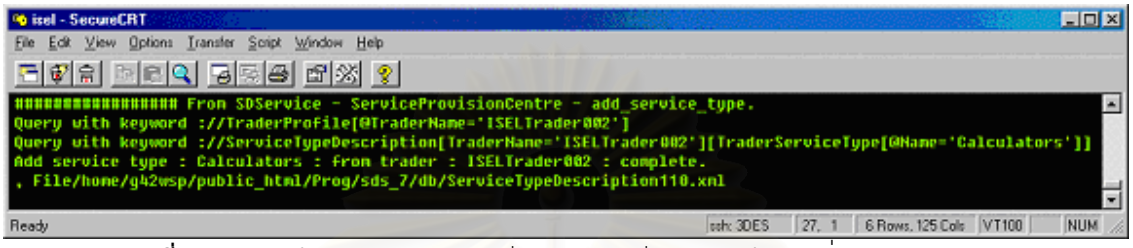
- การเพิ่มชนิดของบริการ (add_type())

เมื่อผู้ใช้บริการเทรดเดอร์ทำการเพิ่มชนิดของบริการ โดยเรียกใช้ตัวกระทำการ add_type() จากบริการเทรดเดอร์ที่มีเทรดเดอร์เอเจนท์ โปรแกรมเทรดเดอร์เอเจนท์จะทำงาน โดยมีการรายงานผลการดักข้อมูล แล้วทำการส่งข้อมูลไปยังบริการเอสดีเอสด้วยยูอาร์แอลของข้อมูลชนิดของบริการที่ผ่านการแปลงเป็นเอกสารเอ็กซ์เอ็มแอลแล้ว ผลการทำงานในส่วนบริการเทรดเดอร์หลังจากการเพิ่มชนิดของบริการ Calculators เป็นดังรูปที่ 5.5

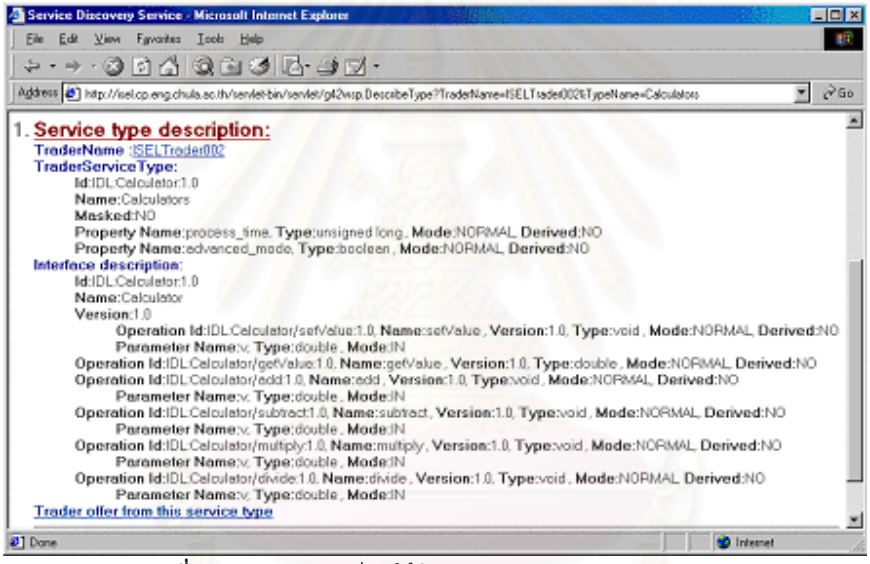


รูปที่ 5.5 ผลการทำงานของเทรดเดอร์เอเจนท์จากการเรียกตัวกระทำการเพิ่มชนิดของบริการ

ส่วนของบริการเอสดีเอส เมื่อมีการเพิ่มชนิดบริการ ระบบจะเรียกใช้ส่วนต่อประสานสำหรับการค้นหาเพื่อตรวจสอบว่าบริการเทรดเดอร์และชนิดของบริการนั้น มีอยู่แล้วหรือไม่ ในที่นี้เป็นการค้นหาเทรดเดอร์ชื่อ ISELTrader002 และชนิดของบริการชื่อ Calculators จากนั้นจึงจัดเก็บข้อมูลลงในคลังข้อมูล โดยผลจากการทำงานเป็นดังรูปที่ 5.6 และผลจากการเรียกใช้โปรแกรมอธิบายชนิดของบริการที่อธิบายชนิดของบริการที่เพิ่มเข้าไปใหม่เป็นดังรูปที่ 5.7



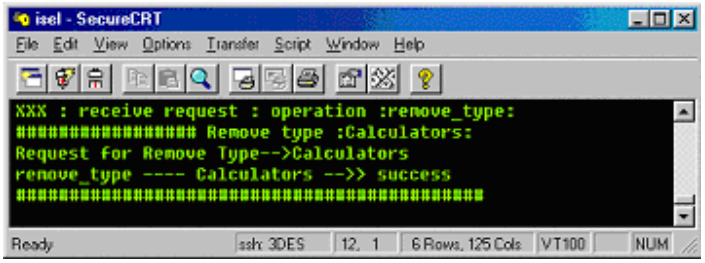
รูปที่ 5.6 ผลการทำงานของบริการเอสดีเอสจากการเรียกตัวกระทำการเพิ่มชนิดของบริการ



รูปที่ 5.7 ผลจากการเรียกใช้โปรแกรมอธิบายชนิดของบริการ

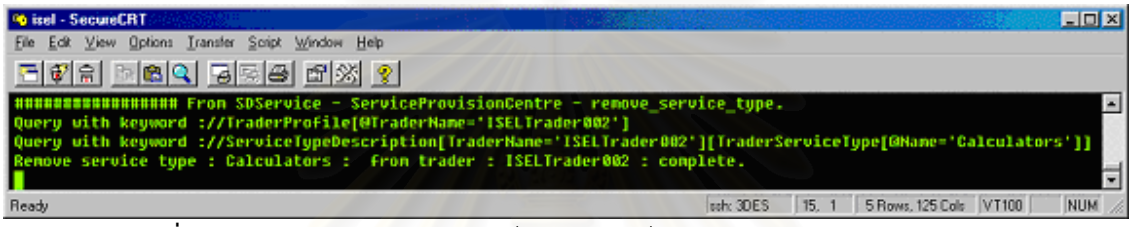
- การลบชนิดของบริการ (remove_type())

เมื่อผู้ใช้บริการเทรดเดอร์ทำการลบชนิดของบริการ โดยเรียกตัวกระทำการ remove_type() จากบริการเทรดเดอร์ที่มีเทรดเดอร์เอเจนท์ โปรแกรมเทรดเดอร์เอเจนท์จะทำงาน โดยมีการรายงานผลการดักข้อมูล แล้วเรียกใช้ตัวกระทำการลบชนิดของบริการไปยังบริการเอสดีเอส การทดสอบกระทำโดยการลบชนิดของบริการชื่อ Calculators ผลที่แสดงในส่วนบริการเทรดเดอร์เป็นดังรูปที่ 5.8



รูปที่ 5.8 ผลการทำงานของเทรดเดอร์จากการเรียกตัวกระทำการลบชนิดของบริการ

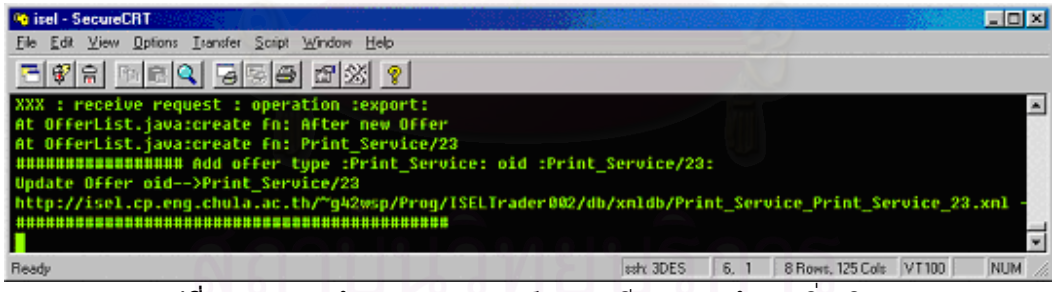
ส่วนของบริการเอสดีเอส เมื่อมีการลบชนิดของบริการ ระบบจะเรียกใช้ส่วนต่อประสานสำหรับการค้นหาเพื่อตรวจสอบว่าบริการเทรดเดอร์และชนิดของบริการนั้น มีอยู่จริงหรือไม่ จากนั้นจึงลบข้อมูลออกจากคลังข้อมูล โดยผลการทำงานส่วนของบริการเอสดีเอสเป็นดังรูปที่ 5.9



รูปที่ 5.9 ผลการทำงานของบริการเอสดีเอสจากการเรียกตัวกระทำการลบชนิดของบริการ

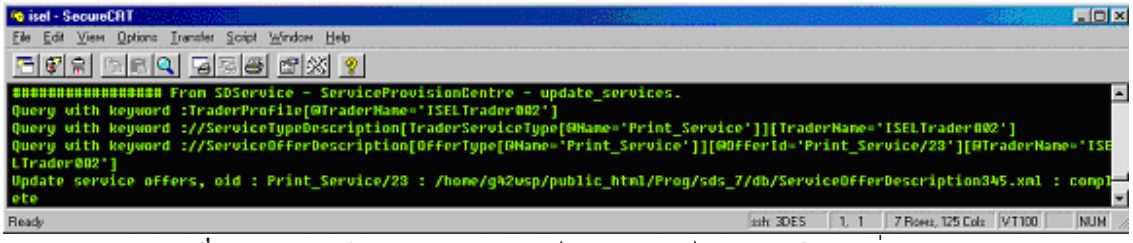
- การเพิ่มบริการ (export())

เมื่อผู้ใช้บริการเทรดเดอร์ทำการเพิ่มบริการ โดยเรียกตัวกระทำการ export() จากบริการเทรดเดอร์ที่มีเทรดเดอร์เอเจนท์ โปรแกรมเทรดเดอร์เอเจนท์จะทำงาน โดยรายงานผลการดักข้อมูล แล้วส่งข้อมูลการเพิ่มบริการไปยังบริการเอสดีเอส การทดสอบเป็นการเพิ่มบริการที่มีชนิดของบริการคือ Print_Service ซึ่งบริการนี้ได้ค่า Offer Id คือ Print_Service/23 ผลการทำงานของบริการเทรดเดอร์แสดงดังรูปที่ 5.10

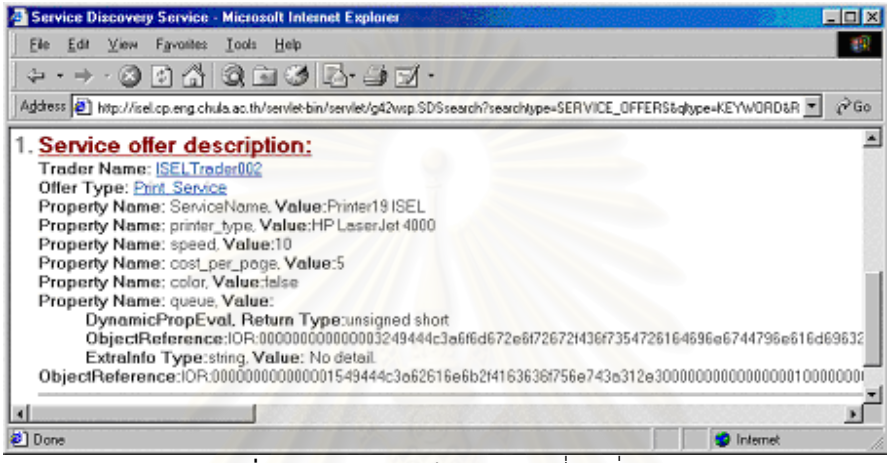


รูปที่ 5.10 ผลการทำงานของเทรดเดอร์จากการเรียกตัวกระทำการเพิ่มบริการ

ส่วนของบริการเอสดีเอส เมื่อมีการเพิ่มบริการ ระบบจะเรียกใช้ส่วนต่อประสานสำหรับการค้นหา เพื่อตรวจสอบว่าบริการเทรดเดอร์ ชนิดของบริการ และค่า Offer Id ของบริการนั้น มีอยู่แล้วหรือไม่ หากไม่มีค่า Offer Id ซ้ำในระบบ ก็จะทำการเพิ่มข้อมูลบริการเข้าไปในคลังเก็บข้อมูล แต่หากมีอยู่แล้ว ระบบจะปรับปรุงให้เป็นข้อมูลใหม่จากการทดสอบการเพิ่มบริการใหม่ ผลการทำงานของบริการเอสดีเอสเป็นดังรูปที่ 5.11 และผลจากการเพิ่มบริการสามารถตรวจสอบโดยค้นหาได้จากโปรแกรมการค้นหาบริการ โดยแสดงผลดังรูปที่ 5.12



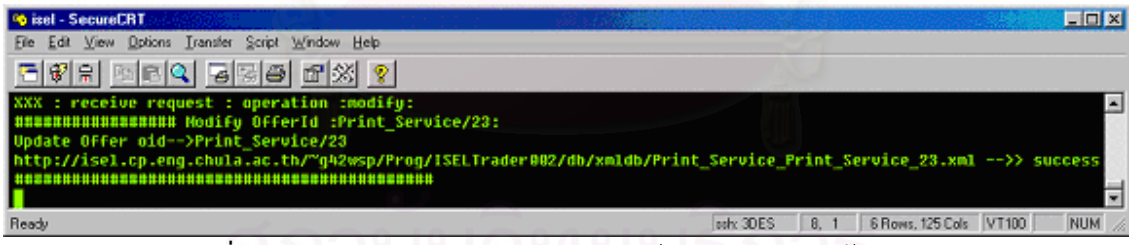
รูปที่ 5.11 ผลการทำงานของบริการเอสดีเอสจากการเรียกตัวกระทำการเพิ่มบริการ



รูปที่ 5.12 ผลจากการค้นหาบริการที่ถูกเพิ่มใหม่

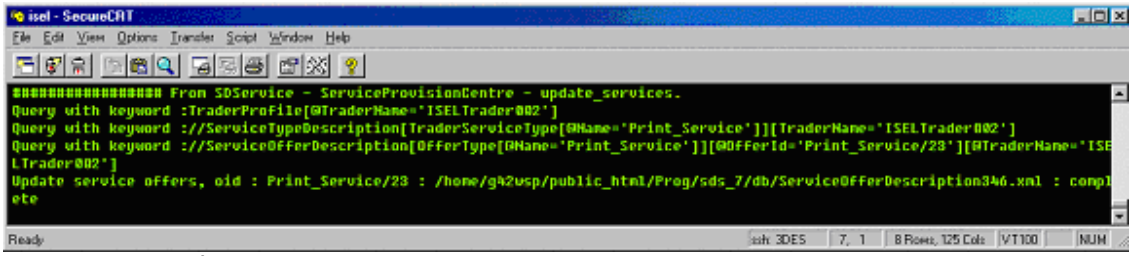
- การแก้ไขบริการ (modify())

เมื่อผู้ใช้ทำการแก้ไขบริการ โดยการเรียกตัวกระทำการ modify() ของบริการเทรดเดอร์ที่มีเทรดเดอร์เอเจนท์โปรแกรมเทรดเดอร์เอเจนท์จะทำงาน โดยมีการรายงานผลการดักข้อมูล แล้วส่งข้อมูลบริการที่ถูกแก้ไขไปยังบริการเอสดีเอส การทดสอบแก้ไขบริการ Print_Service/23 มีผลของการทำงานในส่วนของบริการเทรดเดอร์เป็นดังรูปที่ 5.13

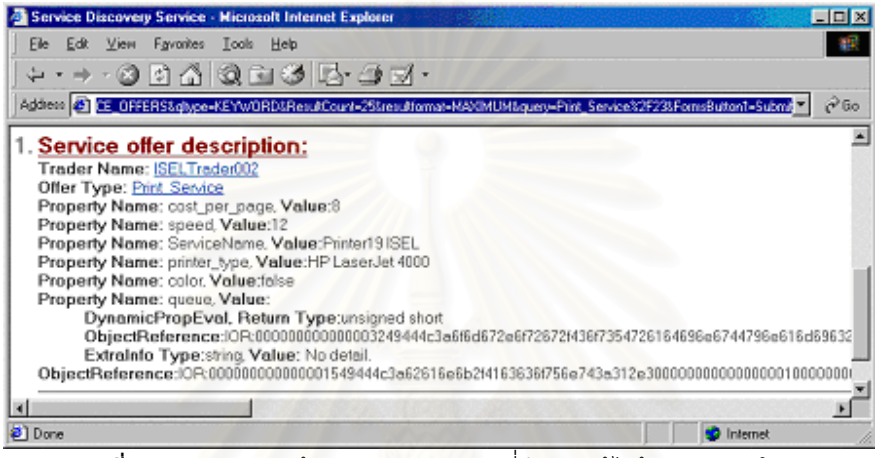


รูปที่ 5.13 ผลการทำงานของเทรดเดอร์จากการเรียกตัวกระทำการแก้ไขบริการ

ส่วนของบริการเอสดีเอส เมื่อมีการแก้ไขบริการ ระบบจะเรียกใช้ส่วนต่อประสานสำหรับการค้นหาเพื่อตรวจสอบว่า ชื่อบริการเทรดเดอร์ ชนิดของบริการ และค่า Offer Id ของบริการนั้น มีอยู่จริงหรือไม่ หากมีอยู่จริงจะทำการแก้ไขข้อมูลให้ ผลของบริการเอสดีเอสจากการแก้ไขข้อมูลบริการเป็นดังรูปที่ 5.14 และผลจากการแก้ไขบริการสามารถค้นหาดูได้จากโปรแกรมการค้นหาบริการ ดังรูปที่ 5.15 ซึ่งจากการทดสอบจะเห็นว่าข้อมูลบริการนี้ถูกเปลี่ยนไปจากเดิมในส่วนของค่าคุณสมบัติ cost_per_page และ speed



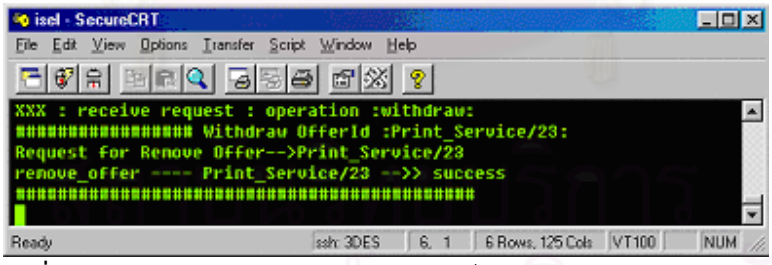
รูปที่ 5.14 ผลการทำงานของบริการเอสดีเอสจากการเรียกตัวกระทำการแก้ไขบริการ



รูปที่ 5.15 ผลจากการค้นหาบริการหลังจากที่ทำการแก้ไขข้อมูลบริการใหม่

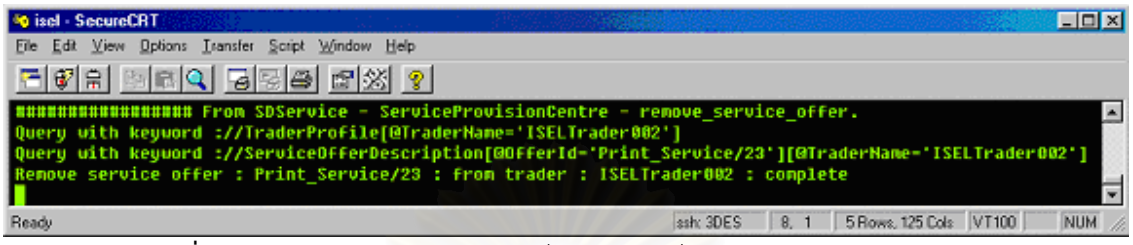
- การยกเลิกบริการ (withdraw())

เมื่อผู้ใช้ทำการยกเลิกบริการ โดยเรียกตัวกระทำการ withdraw() จากบริการเทรดเดอร์ที่มีเทรดเดอร์เอเจนท์ โปรแกรมเทรดเดอร์เอเจนท์จะทำงาน โดยมีการรายงานผลการดักข้อมูล แล้วเรียกตัวกระทำการลบข้อมูลบริการของบริการเอสดีเอส การทดสอบเป็นการยกเลิกบริการที่มีค่า Offer Id เท่ากับ Print_Service/23 ผลการทำงานของบริการเทรดเดอร์เป็นดังรูปที่ 5.16



รูปที่ 5.16 ผลการทำงานของเทรดเดอร์จากการเรียกตัวกระทำการยกเลิกบริการ

ส่วนของบริการเอสดีเอส เมื่อมีการยกเลิกบริการ ระบบจะเรียกใช้ส่วนต่อประสานสำหรับการค้นหาเพื่อตรวจสอบว่าบริการเทรดเดอร์ ชนิดของบริการ และค่า Offer Id ของบริการนั้นมีอยู่จริงหรือไม่ จากนั้นจึงลบข้อมูลออกจากคลังข้อมูล ผลการทำงานในส่วนของบริการเอสดีเอสเป็นดังรูปที่ 5.17



รูปที่ 5.17 ผลการทำงานของบริการเอสดีเอสจากการเรียกตัวกระทำการยกเลิกบริการ

5.3 การทดสอบศูนย์จัดเตรียมบริการ

5.3.1 การทดสอบการทำงานของตัวกระทำการของศูนย์จัดเตรียมบริการ

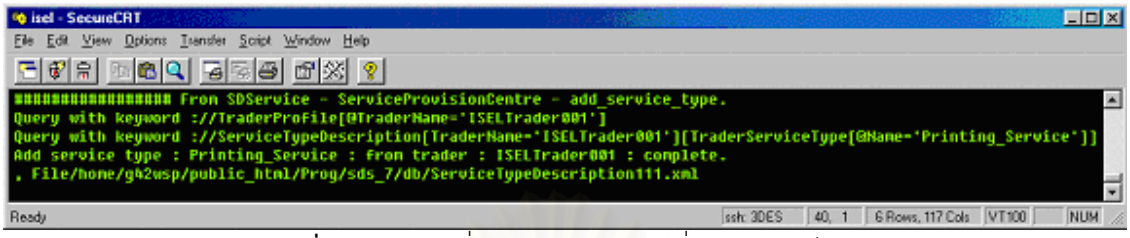
เนื่องจากการทดสอบการทำงานของตัวกระทำการสำหรับการจัดเก็บข้อมูลของศูนย์จัดเตรียมบริการมีรูปแบบที่เกี่ยวเนื่องกันกับการทำงานของเทรดเดอร์เอเจนท์ดังที่ได้กล่าวไปแล้ว ซึ่งผลลัพธ์จากการทำงานของบริการเอสดีเอสเป็นรูปแบบเดียวกัน จึงไม่นำมาอธิบายเพิ่มเติมในหัวข้อนี้

ในหัวข้อนี้จะทำการทดสอบกรณีที่ผู้ใช้ ได้แก่ผู้ดูแลบริการเอสดีเอส และผู้ดูแลบริการเทรดเดอร์ ต้องการที่จะจัดการกับข้อมูลในบริการเอสดีเอสด้วยตนเอง ไม่ว่าจะเป็นการเพิ่มข้อมูลที่เกิดความผิดพลาดในการส่ง หรือการลบข้อมูลที่ไม่ต้องการ ผู้ดูแลบริการเอสดีเอสสามารถใช้การทำงานของศูนย์จัดเตรียมบริการได้ตามขั้นตอนดังนี้

1. เตรียมข้อมูลสำหรับการปรับปรุงคลังเอกสาร เช่น ผู้ดูแลบริการเทรดเดอร์ต้องการเพิ่มข้อมูลชนิดของบริการด้วยตัวเอง ซึ่งอาจเกิดขึ้นได้ในกรณีที่การสื่อสารเกิดข้อผิดพลาดขณะส่งข้อมูล (ซึ่งสามารถอ่านได้จากไฟล์บันทึกผลการทำงาน) ผู้ดูแลบริการเทรดเดอร์จำเป็นต้องจัดเตรียมข้อมูลชนิดของบริการนั้น โดยสามารถนำข้อมูลที่แปลงไว้แล้วซึ่งจัดเก็บอยู่ในไดเร็คทอรีที่กำหนด แล้วทำการกำหนดยูอาร์แอลที่อ้างถึงไฟล์ชนิดของบริการในรูปแบบเอ็กซ์เอ็มแอล
2. เนื่องจากยังไม่มีการพัฒนาส่วนต่อประสานผู้ใช้ (User Interface) สำหรับผู้ดูแลบริการเทรดเดอร์ในการใช้งานศูนย์จัดเตรียมบริการ ดังนั้นผู้ดูแลบริการเทรดเดอร์จะต้องทำการเขียนโปรแกรมขึ้นเอง เพื่อเรียกใช้ตัวกระทำการ add_service_type() จากวัตถุของศูนย์จัดเตรียมบริการ ดังส่วนของโปรแกรมดังนี้

```
SDService.SearchInterface ss = SDService.SearchInterfaceHelper.narrow(obj);
SDService.ServiceProvisionCentre spc = ss.ServiceProvisionCentre_if();
String result = spc.add_service_type("ISELTrader001"',
"http://isel.cp.eng.chula.ac.th/~g42wsp/ts/db/xmldb/Printing_Service.xml");
```

3. เมื่อทำการคอมไพล์โปรแกรมและใช้งาน ก็จะได้ผลลัพธ์ลักษณะเดียวกับการเพิ่มชนิดของบริการโดยเทรดเดอร์เอเจนท์ โดยผลการทำงานในส่วนของบริการเอสดีเอสเป็นดังในรูปที่ 5.18



รูปที่ 5.18 ผลการเพิ่มชนิดของบริการทางฝั่งบริการเอสดีเอส

สำหรับการปรับปรุงคลังข้อมูลด้วยตัวกระทำการอื่น เช่น การลบ หรือการแก้ไขข้อมูล ก็สามารถทำได้ในลักษณะเดียวกัน เพียงแต่เลือกใช้ตัวกระทำการที่เหมาะสมเท่านั้น

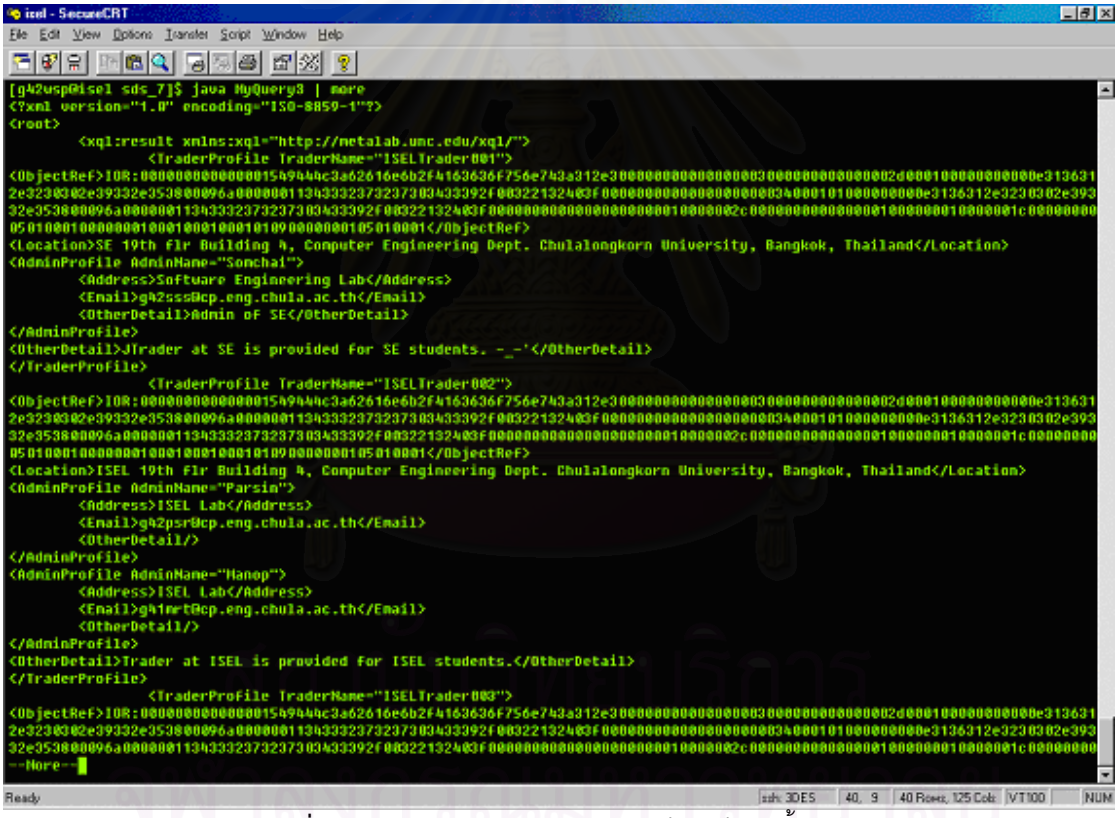
5.3.2 <u>การทดสอบการแปลงรูปข้อมูลและรูปแบบข้อมูลที่จัดเก็บ</u>

สำหรับความถูกต้องของการแปลงข้อมูลสามารถตรวจสอบได้ โดยการแสดงข้อมูลที่เก็บทั้งหมด แล้วทำการตรวจสอบรูปแบบข้อมูลว่าเป็นไปตามที่กำหนดไว้หรือไม่ (ตามดีทีดีในภาคผนวก ก) การทดสอบกระทำโดยการแสดงข้อมูลในคลังข้อมูลของบริการเอสดีเอสโดยตรง โดยโปรแกรมในการทดสอบมีส่วนที่สำคัญดังนี้

```
1.  import de.gmd.ipsi.xql.*;
2.  import de.gmd.ipsi.pdom.*;
3.  import de.gmd.ipsi.domutil.*;
4.  import org.w3c.dom.*;
5.  …..
6.  final static String query = "*";
7.  static final String fileName1    = "db/trader.pdom";
8.  static final String fileName2    = "db/stype.pdom";
9.  static final String fileName3    = "db/offer.pdom";
10. PDocument doc1, doc2, doc3;
11. doc1 = new PDocument ( fileName1 );
12. doc2 = new PDocument ( fileName2 );
13. doc3 = new PDocument ( fileName3 );
14. Document resultDoc = DOMUtil.createDocument();
15. Element root = resultDoc.createElement("root");
16. resultDoc.appendChild( root );
17. XQL.execute( query,  doc1, root);
18. XQL.execute( query,  doc2, root);
19. XQL.execute( query,  doc3, root);
```

```
20. XMLWriter out = new XMLWriter(System.out, "ISO-8859-1");
21. out.write( resultDoc );
22. out.writeln();
23. out.flush();
```

ส่วนของโปรแกรมข้างต้น ทำการกำหนดนิพจน์เพื่อการค้นหาข้อมูลทั้งหมดด้วย "*" (บรรทัดที่ 6) ทำการสร้างวัตถุของเอกสารพีดีโอเอ็มจากไฟล์พีดีโอเอ็ม (บรรทัดที่ 7-13) แล้วนำมาประมวลผลร่วมกับนิพจน์สำหรับค้นหาข้อมูลทั้งหมด โดยใช้ตัวประมวลผลชื่อ XQL (บรรทัดที่ 17-19) ผลลัพธ์ที่ได้จะถูกเก็บในเอกสารเอ็กซ์เอ็มแอลที่สร้างขึ้นมาใหม่ด้วยส่วนย่อยรากคือ root (บรรทัดที่ 15-16) ซึ่งถูกส่งไปเป็นพารามิเตอร์ของการประมวลผลด้วย จากนั้นนำผลจากการค้นหาที่อยู่ในส่วนย่อย root มาพิมพ์ออกทางจอภาพด้วยวัตถุชนิด XMLWriter (บรรทัดที่ 20-23) ซึ่งข้อมูลบางส่วนของผลจากการทำงานของโปรแกรมแสดงดังรูปที่ 5.19 โดยผลการตรวจสอบรูปแบบข้อมูลพบว่าตรงตามรูปแบบที่กำหนด



รูปที่ 5.19 ส่วนของผลลัพธ์จากโปรแกรมค้นหาข้อมูลทั้งหมด

5.4 การทดสอบส่วนต่อประสานสำหรับการค้นหา

5.4.1 การทดสอบการค้นหาข้อมูลตามรูปแบบผู้ใช้งาน

- ผู้ใช้คอร์บา

โปรแกรมทดสอบสำหรับผู้ใช้ส่วนต่อประสานสำหรับการค้นหาในรูปแบบคอร์บา เป็นโปรแกรมที่ทำงานในรูปแบบอักษร (Text Mode) ซึ่งมีส่วนของโปรแกรมที่สำคัญดังนี้

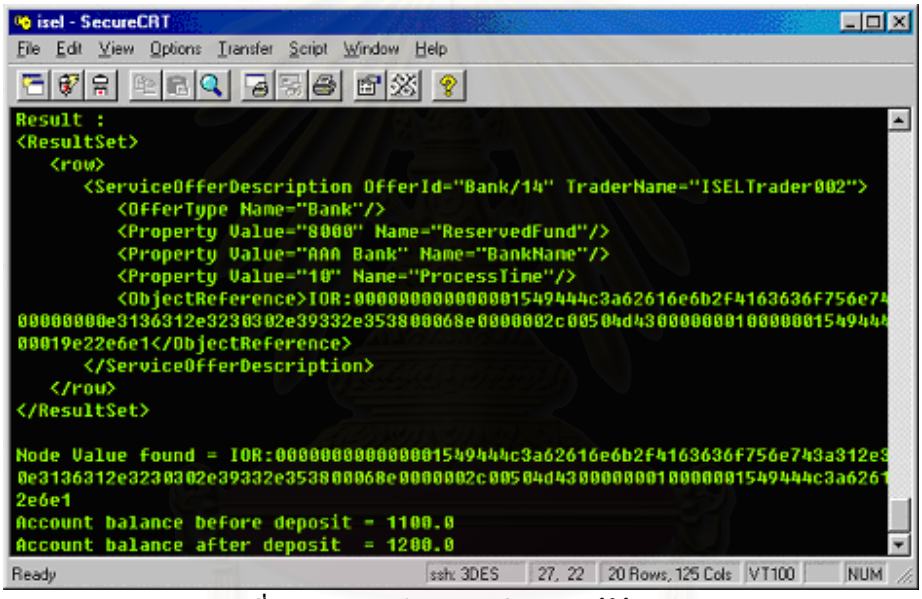
```
1. SDService.SearchInterface ss = SDService.SearchInterfaceHelper.narrow(obj);
2. String ql_type = “XQL”;
3. String keyw = "ServiceOfferDescription[(OfferType/@Name = 'Bank') $and$ " +
4. "(Property[(@Name= 'ReservedFund')  $and$ (@Value > 5000)])]";
5. SDService.SearchInterfacePackage.SearchType search_type =
6. SearchType.SERVICE_OFFERS;
7. boolean isUrl = false;
8. SDService.SearchInterfacePackage.ResultFormat rsf = ResultFormat.MINIMUM;
9. String results = ss.search(keyw, ql_type, search_type, isUrl, rsf);
```

จากส่วนของโปรแกรมข้างต้น เมื่อได้วัตถุของส่วนต่อประสานสำหรับการค้นหา (บรรทัดที่ 1) และกำหนดค่าพารามิเตอร์ของตัวกระทำการต่างๆ (บรรทัดที่ 2-8) ซึ่งจากตัวอย่างเป็นการค้นหาบริการ ที่มีชื่อของบริการคือ Bank และค่าคุณสมบัติ ReservedFund มากกว่า 5000 จากนั้นจึงเรียกใช้ตัวกระทำการ search() ซึ่งผลจากการค้นหาจะถูกเก็บในตัวแปร results (บรรทัดที่ 9) หลังจากนั้นเป็นการเลือกข้อมูลจากผลการค้นหาข้างต้น เพื่อที่จะนำมาใช้งานจริง ในตัวอย่างการทดสอบใช้รูปแบบของการนำส่วนย่อยแรกที่พบมาประมวลผล โดยใช้การทำงานของดีโอเอ็ม ดังส่วนของโปรแกรมดังนี้

```
1. StringReader in_res = new StringReader(results);
2. parser.parse(in_res);
3. Document xml_result = parser.getDocument();
4. NodeList nl = xml_result.getElementsByTagName("ObjectReference");
5. String tx = "";
6. if(nl.item(0) != null){
7. Element ts = (Element)nl.item(0);
8. tx = ((Text)ts.getChildNodes().item(0)).getNodeValue();
9. System.out.println("Node Value found = " + tx);
10. } else {
11. System.out.println("No result found..");
12. }
13. org.omg.CORBA.Object objRef = orb.string_to_object(tx);
```

```
14. bank.Account mybank = bank.AccountHelper.narrow(objRef);
15. System.out.println("Account balance before deposit = " + mybank.balance());
16. mybank.deposit(100);
17. System.out.println("Account balance after deposit = " + mybank.balance());
```

จากส่วนของโปรแกรมข้างต้น เป็นการเลือกวัตถุอ้างอิงของบริการจากผลการค้นหา เพื่อนำมาใช้งาน โดยนำผลลัพธ์ที่ได้มาแปลงให้เป็นวัตถุของดีโอเอ็ม (บรรทัดที่ 1-3) จากนั้นจึงเลือกข้อมูลในส่วนย่อยชื่อ ObjectReference และนำค่าแรกที่พบออกมา (บรรทัดที่ 4-12) หลังจากนั้นนำค่าที่ได้มาแปลงเป็นวัตถุ(บรรทัดที่ 13) และนำมา narrow ให้เป็นวัตถุของ bank.Account (บรรทัดที่ 14) ซึ่งเป็นตัวอย่างวัตถุของธนาคาร จากนั้นเรียกใช้ตัวกระทำการ deposit() เพื่อฝากเงินจำนวน 100 และทำการพิมพ์ค่ายอดเงินคงเหลือโดยการเรียกตัวกระทำการ balance() (บรรทัดที่ 15-17) ผลจากการทำงานเป็นดังรูปที่ 5.20



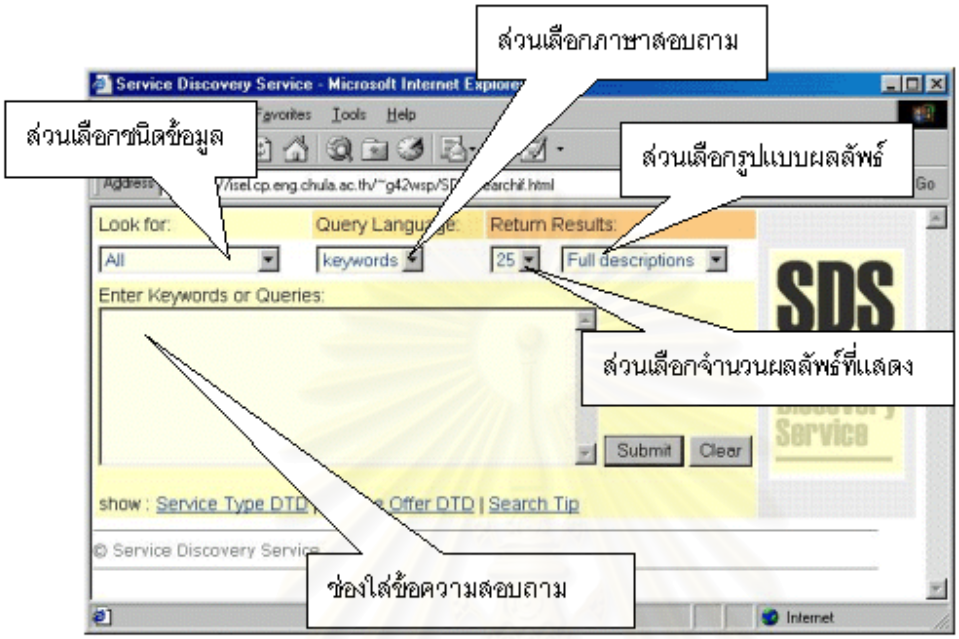
รูปที่ 5.20 ผลการทำงานของโปรแกรมผู้ใช้คอร์บา

- ผู้ใช้ผ่านโปรแกรมค้นผ่านเว็บ

สำหรับผู้ใช้ผ่านเว็บจำเป็นต้องอาศัยโปรแกรมค้นผ่านเว็บในการเรียกใช้งานส่วนต่อประสานสำหรับการค้นหา ข้อดีของโปรแกรมใช้งานผ่านเว็บคือช่วยให้ผู้ใช้ใช้งานได้สะดวกขึ้น โดยต้นแบบส่วนต่อประสานสำหรับการค้นหาผ่านเว็บในวิทยานิพนธ์นี้ทำงานอยู่บนเว็บเซิร์ฟเวอร์ที่กำหนดยูอาร์แอลเริ่มต้นของการทำงานไว้คือ

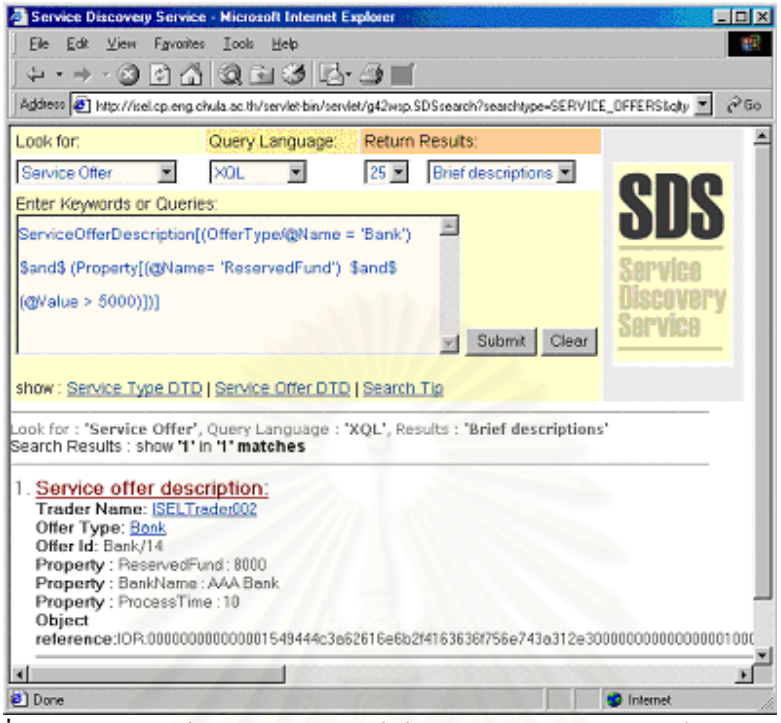
http://isel.cp.eng.chula.ac.th/~g42wsp/SDS/searchif.html

ยูอาร์แอลนี้แสดงผลผ่านโปรแกรมค้นผ่านเว็บดังรูปที่ 5.21 ซึ่งข้อมูลที่ผู้ใช้ใส่ในแบบฟอร์มจะถูกส่งไปประมวลผลโดยโปรแกรมเซิร์ฟเล็ตชื่อ SDSsearch ซึ่งอธิบายไว้ในบทที่ 4



รูปที่ 5.21 รายละเอียดของโปรแกรมผ่านเว็บส่วนต่อประสานสำหรับการค้นหา

ผลจากการทดสอบการค้นหาบริการด้วยนิพจน์เดียวกับตัวอย่างในรูปแบบผู้ใช้คอร์บาข้างต้น แสดงดังรูปที่ 5.22 หลังจากนั้นการนำข้อมูลไปใช้งานก็เป็นไปในลักษณะเดียวกัน โดยผู้ใช้ต้องเลือกเอาข้อมูลอ้างอิงบริการ (จากตัวอย่างคือค่าของ ObjectReference) มาแปลงเป็นวัตถุเสียก่อนด้วยตัวกระทำการ string_to_object() ของออร์บเพื่อที่จะนำไปเรียกใช้งานต่อไปในโปรแกรมได้



รูปที่ 5.22 ผลจากการค้นหาบริการสำหรับผู้ใช้ส่วนต่อประสานสำหรับการค้นหาผ่านเว็บ

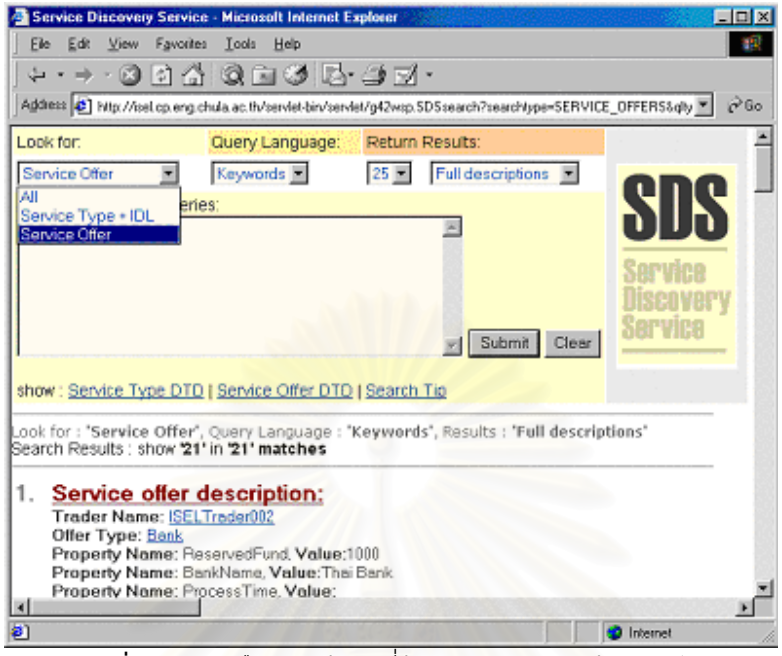
5.4.2 <u>การทดสอบคุณสมบัติการค้นหาของเอสดีเอส</u>

- การเลือกเฉพาะกลุ่มข้อมูลที่ต้องการค้นหา

การเลือกเฉพาะกลุ่มข้อมูลที่ต้องการค้นหามี 3 รูปแบบดังที่อธิบายไว้ในบทที่ 3 ผู้ใช้สามารถเลือกค้นหาได้โดยการระบุค่าพารามิเตอร์ SearchType ให้กับตัวกระทำการ search() ดังเช่น

```
SDService.SearchInterfacePackage.SearchType search_type = SearchType.SERVICE_OFFERS;
```

เป็นการเลือกค้นหาเฉพาะข้อมูลของบริการ หากผู้ใช้ทำการค้นหาผ่านโปรแกรมค้นผ่านเว็บก็สามารถเลือกชนิดข้อมูลที่จะค้นหาได้ดังรูปที่ 5.23

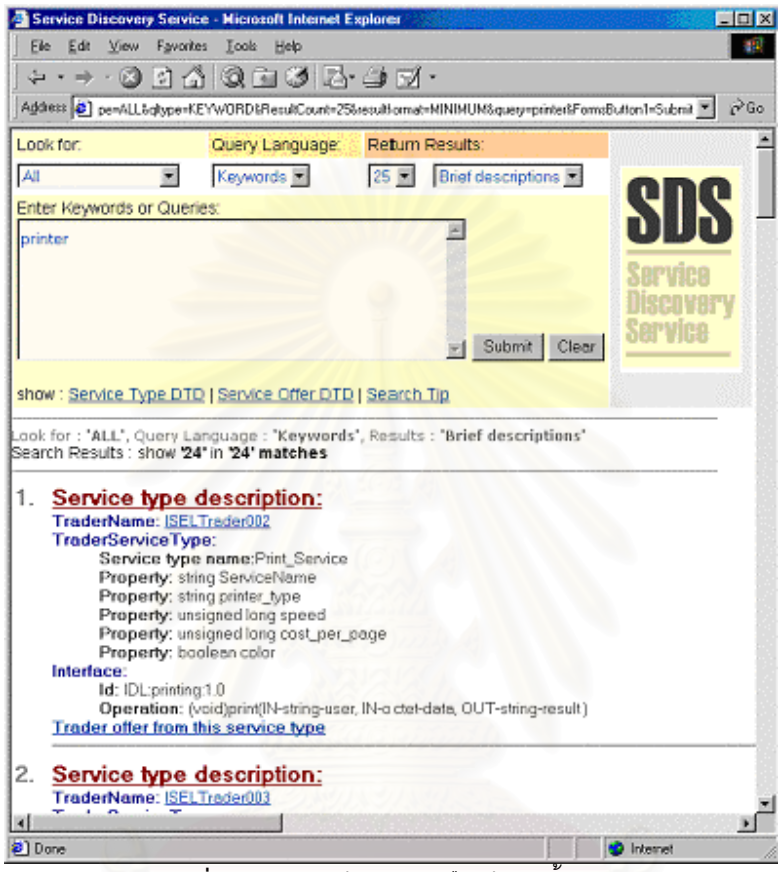


รูปที่ 5.23 การเลือกชนิดข้อมูลที่ค้นหาผ่านโปรแกรมค้นผ่านเว็บ

ตัวอย่างการทดสอบการเลือกชนิดข้อมูลที่จะค้นหา กระทำโดยการค้นหาด้วยคำสำคัญคือ printer และทำการตรวจสอบจำนวนและชนิดของผลลัพธ์ว่าตรงกับชนิดข้อมูลที่เลือกค้นหาหรือไม่ โดยแบ่งได้ดังนี้

- ค้นหาข้อมูลทั้งหมด

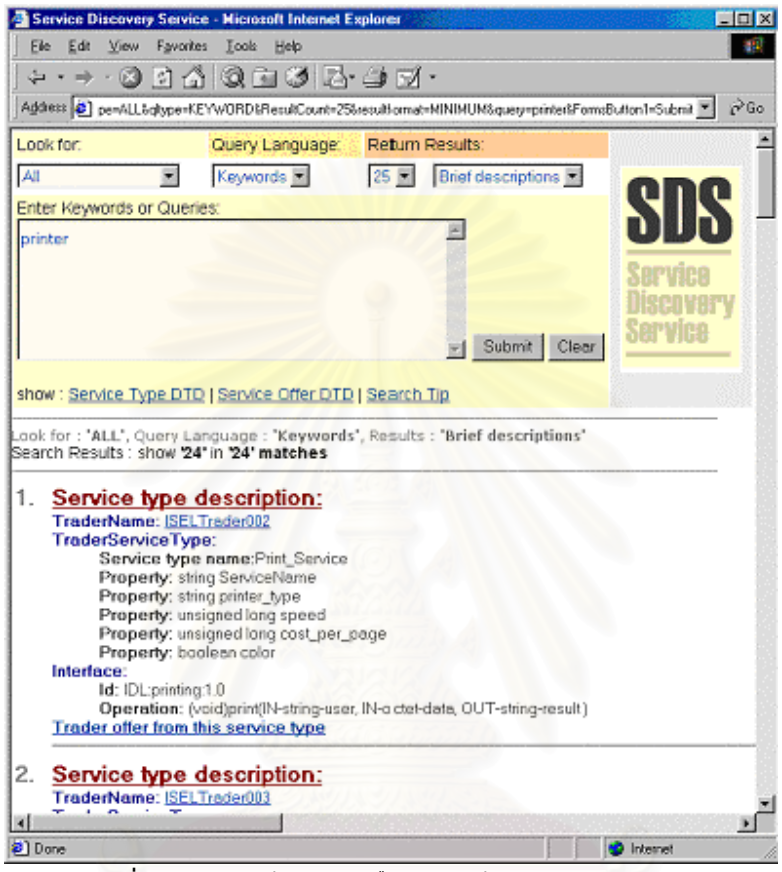
การค้นหาข้อมูลทั้งหมด คือการค้นหาทั้งข้อมูลบริการและชนิดบริการ จากรูปที่ 5.24 เป็นผลการค้นหาซึ่งได้จำนวนทั้งสิ้น 24 ผลลัพธ์ เป็นข้อมูลชนิดของบริการ 2 ชุด และข้อมูลบริการ 22 ชุด



รูปที่ 5.24 ผลการค้นหาโดยเลือกข้อมูลทั้งหมด

- เลือกค้นหาเฉพาะข้อมูลชนิดของบริการ

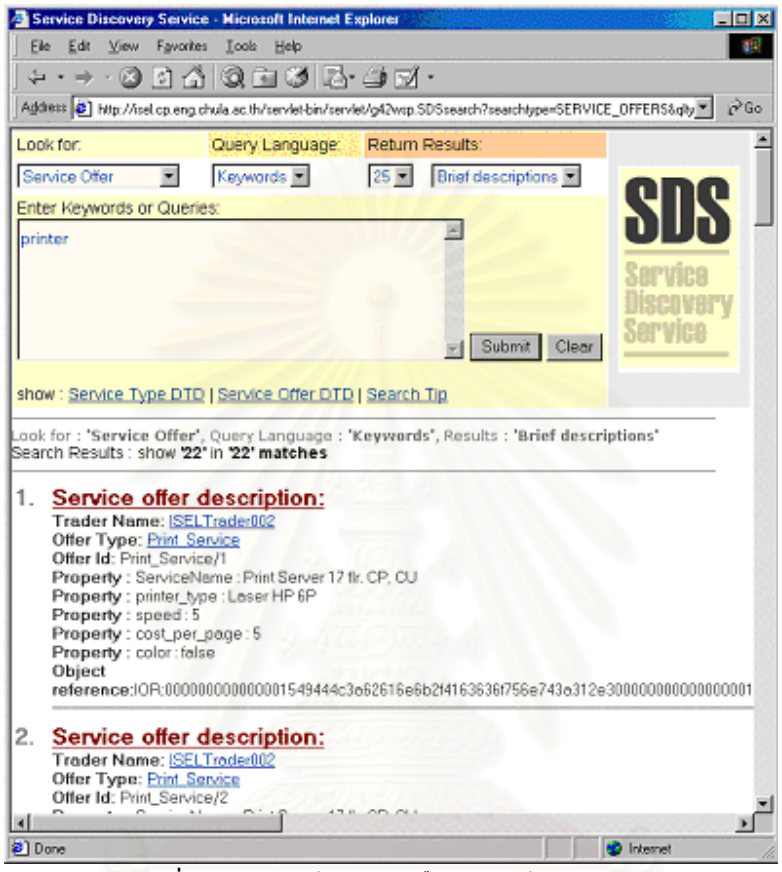
การเลือกเฉพาะข้อมูลบริการ ด้วยคำสำคัญเดียวกันกับข้างต้น ได้จำนวนผลลัพธ์เท่ากับ 2 ดังแสดงในรูปที่ 5.25



รูปที่ 5.25 ผลการค้นหาโดยเลือกเฉพาะข้อมูลชนิดของบริการ

- เลือกค้นหาเฉพาะข้อมูลบริการ

สำหรับการค้นหาข้อมูลบริการด้วยคำสำคัญเดียวกันนี้ พบจำนวนบริการทั้งสิ้น 22 ผลลัพธ์ ดังแสดงในรูปที่ 5.26

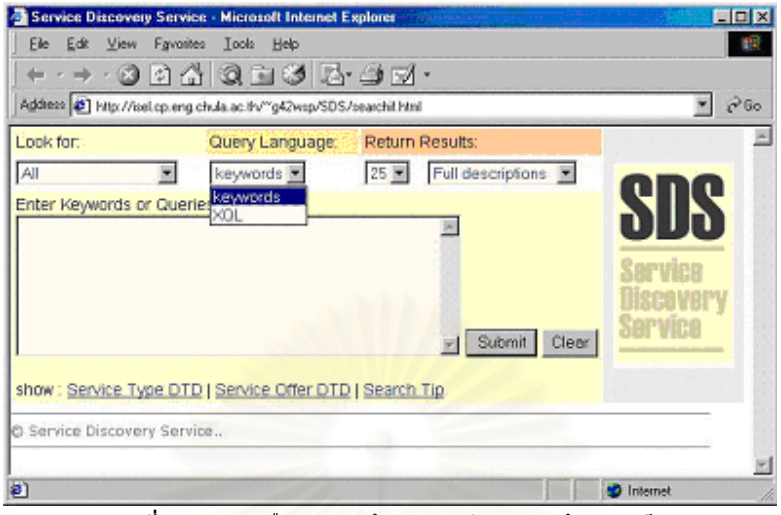


รูปที่ 5.26 ผลการค้นหาโดยเลือกเฉพาะข้อมูลบริการ

จากการค้นหาข้างต้น ผลลัพธ์จากการค้นหาด้วยข้อมูลทั้งหมด จะเท่ากับจำนวนผลลัพธ์จากการค้นหาเฉพาะข้อมูลชนิดของบริการ รวมกับจำนวนผลลัพธ์จากการค้นหาเฉพาะข้อมูลบริการ

- การเลือกภาษาสอบถาม

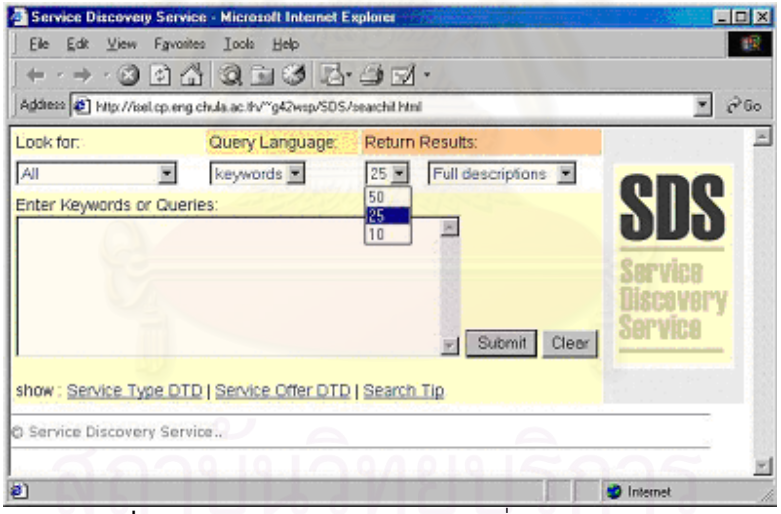
ผู้ใช้สามารถเลือกภาษาค้นหาได้จากการระบุค่าในพารามิเตอร์ QLType ในตัวกระทำการ search() โดยสามารถเลือกระบุได้ 2 แบบคือ “XQL” และ “KEYWORD” ส่วนผู้ใช้ผ่านเว็บสามารถเลือกได้จากหน้าต่าง ดังรูปที่ 5.27 โดยรายละเอียดคุณสมบัติของภาษาสอบถามแต่ละชนิดแสดงในหัวข้อ 5.4.3



รูปที่ 5.27 การเลือกภาษาค้นหาผ่านโปรแกรมค้นผ่านเว็บ

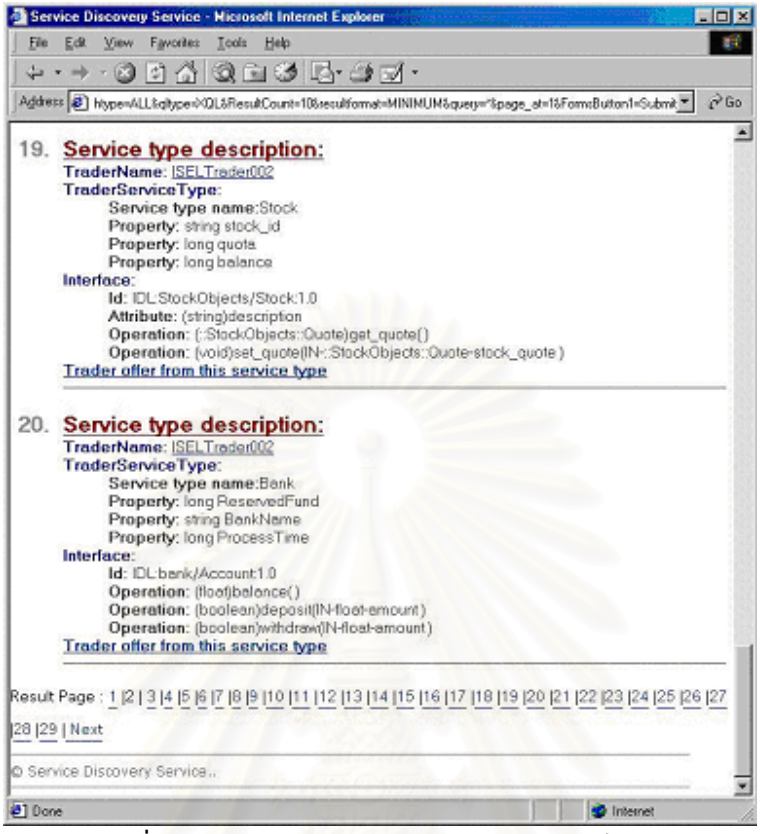
- การระบุจำนวนผลลัพธ์ที่แสดงต่อการค้นหา

ผู้ใช้สามารถกำหนดจำนวนผลลัพธ์ที่จะแสดงต่อการค้นหาได้ ซึ่งคุณสมบัตินี้มีเฉพาะในส่วนต่อประสานสำหรับการค้นหาของผู้ใช้ผ่านเว็บเท่านั้น โดยกำหนดให้ผู้ใช้สามารถเลือกดูผลลัพธ์ได้เป็น 10, 25 และ 50 ชุดต่อหน้า รูปแบบการเลือกใช้เป็นดังรูปที่ 5.28



รูปที่ 5.28 รูปแบบการระบุจำนวนผลลัพธ์ที่แสดงต่อการค้นหา

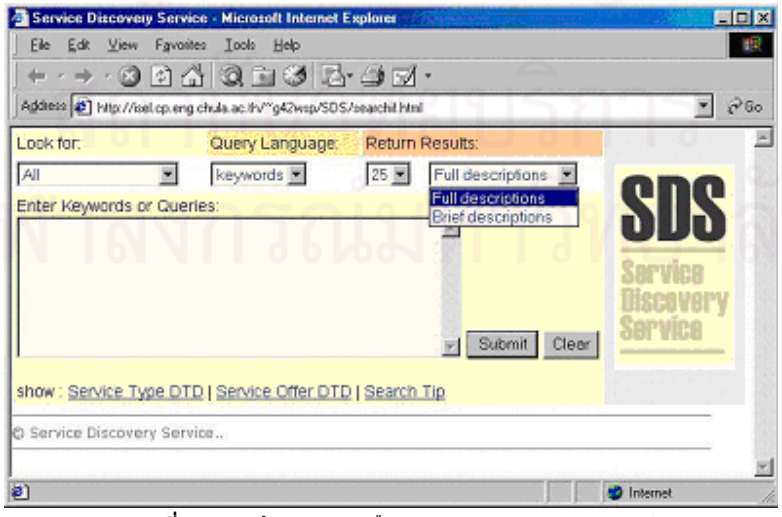
จากการเลือกจำนวนผลลัพธ์ดังกล่าว หากผลลัพธ์จากการค้นหามีมากกว่าค่าที่เลือก โปรแกรมจะสร้างแถบของจำนวนผลลัพธ์ที่พบทั้งหมด ซึ่งแบ่งผลลัพธ์เป็นหน้าย่อยๆ ผู้ใช้สามารถเลือกดูได้ รูปที่ 5.29 เป็นตัวอย่างการค้นหาด้วยการระบุให้แสดงผลลัพธ์ 10 ชุดต่อหน้า โดยมีทั้งสิ้น 29 หน้า



รูปที่ 5.29 ตัวอย่างแถบของจำนวนผลลัพธ์จากการค้นหา

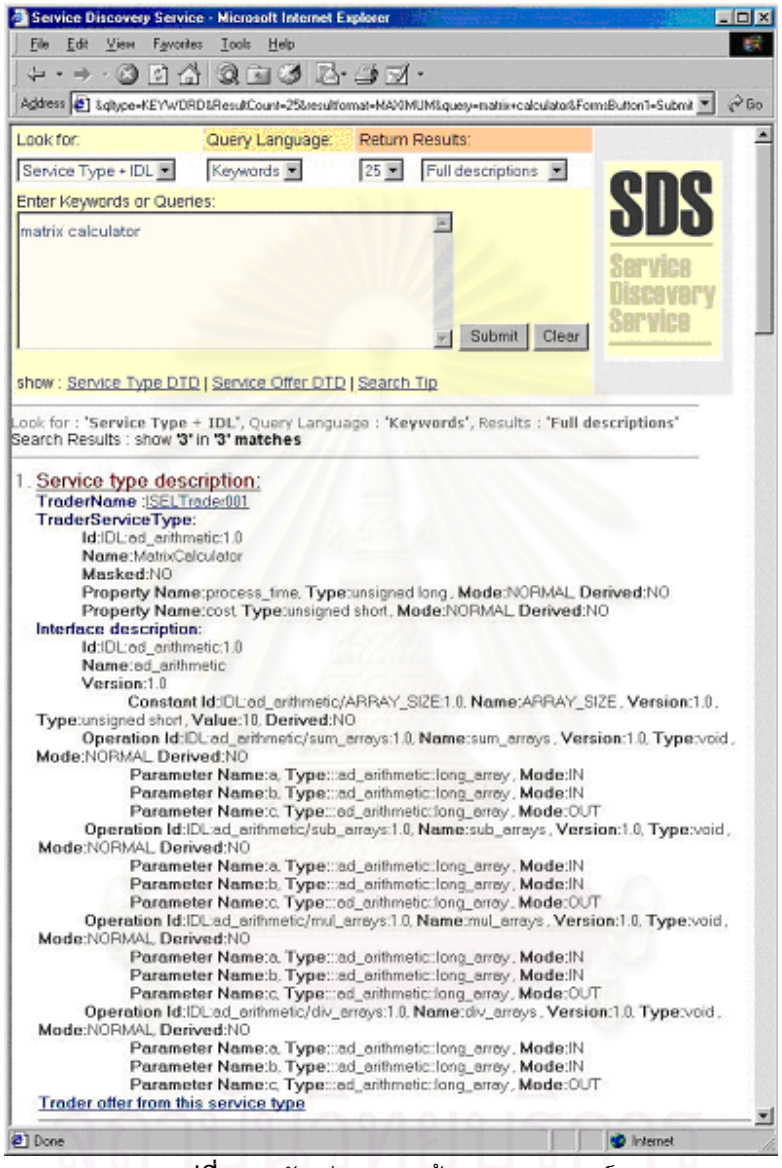
- การเลือกรูปแบบผลการค้นหา

ผู้ใช้สามารถเลือกรูปแบบผลลัพธ์ที่จะแสดงได้ 2 แบบ คือแบบสมบูรณ์และแบบย่อ ดังได้กล่าวไว้ในบทที่ 3 โดยรูปแบบของผลลัพธ์สำหรับผู้ใช้แบบคอร์บาจะเป็นรูปแบบเอ็กซ์เอ็มแอล แต่รูปแบบผลลัพธ์สำหรับผู้ใช้ผ่านเว็บจะถูกแปลงรูปเป็นเอชทีเอ็มแอล ซึ่งหน้าต่างการเลือกรูปแบบผลลัพธ์ของส่วนต่อประสานสำหรับการค้นหาผ่านเว็บแสดงดังรูปที่ 5.30

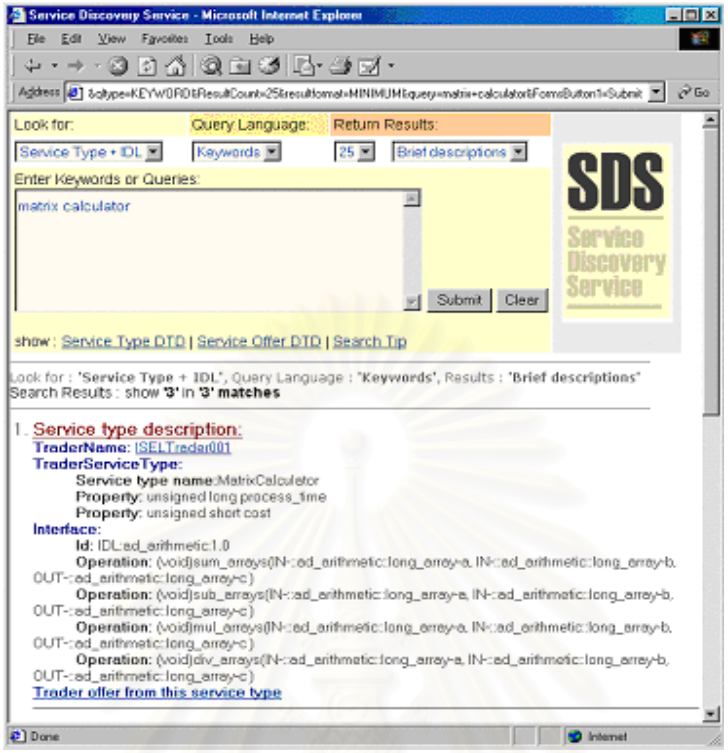


รูปที่ 5.30 หน้าต่างการเลือกรูปแบบการแสดงผลลัพธ์

ผลจากการค้นหาทั้ง 2 รูปแบบ โดยการใช้คำสำคัญเดียวกัน แล้วทำการเปลี่ยนเฉพาะรูปแบบการแสดงผล จะได้ผลดังตัวอย่างในรูปที่ 5.31 และ 5.32



รูปที่ 5.31 ตัวอย่างผลการค้นหาแบบสมบูรณ์



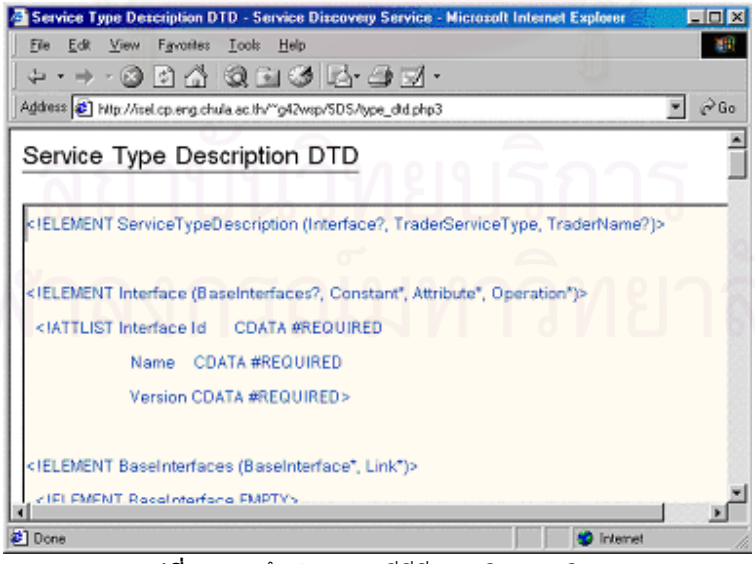
รูปที่ 5.32 ตัวอย่างผลการค้นหาแบบย่อ

- ระบบช่วยเหลือของส่วนต่อประสานสำหรับการค้นหาผ่านเว็บ

ระบบช่วยเหลือที่สร้างขึ้นเพื่ออำนวยความสะดวกแก่ผู้ใช้ มีส่วนต่างๆดังนี้

- การแสดงดีทีดีของชนิดของบริการ และบริการ

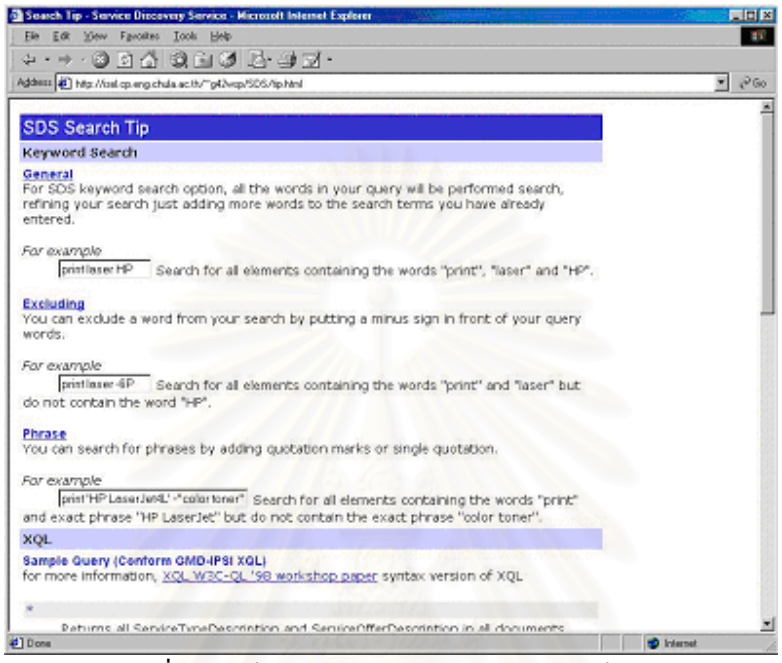
การแสดงดีทีดีก็เพื่อให้ผู้ใช้ที่ต้องการค้นหาข้อมูลด้วยภาษาเอ็กซ์คิวแอล แต่ไม่รู้โครงสร้างของเอกสาร สามารถทำการค้นหาได้ โดยมีส่วนของหน้าต่างดังรูปที่ 5.33



รูปที่ 5.33 หน้าต่างแสดงดีทีดีของชนิดของบริการ

- เทคนิคและตัวอย่างการค้นหา

การแสดงเทคนิคและตัวอย่างการค้นหาจะช่วยให้ผู้ใช้ทำความเข้าใจกับรูปแบบการใช้งานภาษาสอบถามได้ดีขึ้น โดยรูปแบบของส่วนช่วยเหลือนี้แสดงดังรูปที่ 5.34



รูปที่ 5.34 หน้าต่างแสดงเทคนิคและตัวอย่างการค้นหา

5.4.3 การเปรียบเทียบคุณสมบัติการค้นหากับรูปแบบการค้นหาระหว่างบริการเทรดเดอร์กับบริการเอสดีเอส และความสามารถอื่นๆในการค้นหาของบริการเอสดีเอส

- ข้อเปรียบเทียบระหว่างรูปแบบการค้นหาบริการของบริการเทรดเดอร์ซึ่งใช้ภาษาบังคับ [2] กับรูปแบบการค้นหาบริการของบริการเอสดีเอสซึ่งใช้เอ็กซ์คิวแอลเป็นดังตารางที่ 5.2

ตารางที่ 5.2 ผลการเปรียบเทียบคุณสมบัติการค้นหาบริการระหว่างบริการเทรดเดอร์และบริการเอสดีเอส

บริการเทรดเดอร์	บริการเอสดีเอส
ฟังก์ชันการเปรียบเทียบ (Comparative Function):	
== (เท่ากับ)	eq , =
!= (ไม่เท่ากับ)	ne , !=
>	gt , >
>=	ge , >=
<	lt , <
<=	le , <=
~ (ค้นหาส่วนของอักขระ (Substring Match))	$contains$ and $icontains$ (คุณสมบัติเฉพาะใน GMD-IPSI)
in (ค้นหาส่วนย่อยในลำดับ (Element in	

Sequence))	ไม่สนับสนุน
การเชื่อมต่อตรรกะ (Boolean Connective): And Or Not	 and or not
การมีอยู่ของคุณสมบัติ (Property Existence): Exist	 สามารถค้นหาได้จากส่วนย่อย <property>
ตัวกระทำการทางคณิตศาสตร์ (Mathematical Operation): +, -, *, /	 สร้างฟังก์ชันเพิ่มเติม ได้แก่ sum, sub, mul, div (คุณสมบัติเฉพาะใน GMD-IPSI)
ตัวกระทำการสำหรับจัดกลุ่ม (Grouping Operation): (,)	 ( )
ขนาดของตัวอักษรในนิพจน์ มีผลต่อการค้นหา (Case Sensitive)	ขนาดของตัวอักษรในนิพจน์ มีผลต่อการค้นหา แต่หากต้องการให้ขนาดของตัวอักษรไม่มีผลต่อการค้นหา (Case Insensitive) สามารถใช้: ieq ine ilt ile igt ige (คุณสมบัติเฉพาะใน GMD-IPSI)

จากข้อเปรียบเทียบดังตารางที่ 5.2 จะเห็นว่าบริการเอสดีเอสสามารถทำการค้นหาบริการได้ใกล้เคียงกับคุณสมบัติการค้นหาบริการของบริการเทรดเดอร์ โดยส่วนที่ไม่สามารถทำได้เนื่องมาจากข้อจำกัดของส่วนการแปลงคำอธิบายบริการ โดยส่วนที่ยังไม่สามารถทำได้ ได้แก่ การค้นหาที่เกี่ยวข้องกับชนิดของข้อมูลที่ผู้ใช้กำหนด (User-Defined Type) เนื่องจากส่วนการแปลงคำอธิบายบริการที่นำมาใช้ในต้นแบบนั้นยังไม่ครอบคลุมการแปลงชนิดข้อมูลที่ผู้ใช้กำหนดลงในเอกสารเอ็กซ์เอ็มแอลที่เป็นผลลัพธ์ ดังเช่น คุณสมบัติ Addr ของบริการหนึ่ง มีชนิดเป็น Address ซึ่งผู้ใช้กำหนดให้เป็น Struct ของ Number, Street และ City ผลที่ได้จากการแปลง จะได้เอกสารเอ็กซ์เอ็มแอลที่เป็นคำอธิบายของบริการนี้ โดยมีคุณสมบัติ Addr ที่มีชนิด Address ปรากฏอยู่ แต่จะไม่มีรายละเอียดของ Address ว่าประกอบด้วยข้อมูลอะไร ดังนั้นผู้ใช้จะไม่สามารถทำการค้นหาบนเอกสารเอ็กซ์เอ็มแอลนี้ โดยระบุรายละเอียดของ Address ได้ เช่น ไม่สามารถค้นหาบริการที่มี City = 'Bangkok' ได้ นอกจากนี้ยังไม่สามารถค้นหาส่วนย่อยในลำดับ (Element in Sequence) แบบที่เทรดเดอร์ทำได้ เนื่องจากเป็นการค้นหาที่เกี่ยวข้องกับรายละเอียดภายในลำดับ ซึ่งถือเป็นข้อมูลที่ผู้ใช้กำหนดเองแบบหนึ่ง อย่างไรก็ดี หากได้มีการปรับปรุงส่วนการแปลงคำอธิบายบริการที่นำมาใช้ ก็จะทำให้บริการเอสดีเอสสามารถทำการค้นหาแบบเดียวกับบริการเทรดเดอร์ได้

- ข้อเปรียบเทียบระหว่างความสามารถในการค้นหาชนิดของบริการโดยบริการเทรดเดอร์กับรูปแบบการค้นหาโดยบริการเอสดีเอสซึ่งใช้เอ็กซ์คิวแอล ผลการเปรียบเทียบเป็นดังตารางที่ 5.3

ตารางที่ 5.3 ผลการเปรียบเทียบความสามารถในการค้นหาชนิดของบริการระหว่างบริการเทรดเดอร์กับบริการเอสดีเอส

บริการเทรดเดอร์	บริการเอสดีเอส
ในการค้นหาชนิดของบริการ ผู้ใช้จะเรียกใช้ตัวกระทำการจากคลังเก็บชนิดของบริการตามลำดับดังนี้ • list_type() - เป็นตัวกระทำการสำหรับแสดงรายชื่อชนิดของบริการ ผลที่ได้จากการเรียกใช้งานคือ รายชื่อชนิดของบริการทั้งหมดที่มีอยู่ในคลังเก็บชนิดของบริการนั้น ผู้ใช้ต้องพิจารณาจากรายชื่อทั้งหมดเองว่าชนิดของบริการใดคือสิ่งที่สนใจ จากนั้นจึงเรียกใช้ describe_type() หรือ fully_describe_type() ต่อไป • describe_type(ServiceTypeName) - เป็นตัวกระทำการสำหรับอธิบายชนิดของบริการ โดยส่งชื่อชนิดของบริการนั้นเป็นพารามิเตอร์ ผลที่ได้คือรายละเอียดของชนิดของบริการนั้น • fully_describe_type(ServiceTypeName) - มีลักษณะเช่นเดียวกับ describe_type() แต่ผลลัพธ์จะรวมเอาคุณสมบัติของชนิดของบริการที่สืบทอดมาด้วย	ผู้ใช้สามารถค้นหาข้อมูลชนิดของบริการและข้อมูลส่วนต่อประสานได้โดยตรงจากคลังข้อมูลที่อยู่ในรูปแบบเอกสารเอ็กซ์เอ็มแอล โดยมีตัวอย่างดังนี้ • การค้นหาข้อมูลในส่วนย่อยของชนิดของบริการ เช่น ServiceTypeDescription[TraderServiceType[.//BaseServiceType[@Name="Calculator"]] and not (.//Property[@Name="Cost"])] เป็นการค้นหาข้อมูลชนิดของบริการที่มีชื่อว่า Calculator ที่ไม่มีคุณสมบัติชื่อว่า Cost • การค้นหาข้อมูลชนิดของบริการที่มีการอ้างถึงส่วนต่อประสาน เช่น ServiceTypeDescription[.//Operation[@Name="deposit"]/Parameter/@Type ="double"] เป็นการค้นหาชนิดของบริการที่รองรับตัวกระทำการชื่อ deposit ซึ่งมีพารามิเตอร์ชนิด double

- สำหรับการค้นหาแบบใช้คำสำคัญในการค้นหาทั้งเอกสาร ได้มีการเปรียบเทียบความสามารถของการค้นหาดังแสดงในตารางที่ 5.4

ตารางที่ 5.4 ผลการเปรียบเทียบความสามารถในการค้นหาแบบใช้คำสำคัญระหว่างบริการเทรดเดอร์กับบริการเอสดีเอส

บริการเทรดเดอร์	บริการเอสดีเอส
ในการค้นหาด้วยคำสำคัญ หรือการค้นหาบางส่วนของอักขระ จะสามารถทำได้กับเฉพาะค่าคุณสมบัติของบริการเท่านั้น เช่น 'Laser' ~ PrintType เป็นการค้นหาบริการที่มีคุณสมบัติ PrintType ที่มีค่าหรือบางส่วนของค่าเป็น Laser	สำหรับการค้นหาด้วยคำสำคัญ บริการเอสดีเอสนำรูปแบบการทำงานของโปรแกรมค้นหาโดยทั่วไปมาใช้ ซึ่งมีรูปแบบดังนี้ • คำสำคัญที่ผู้ใช้กำหนดจะถูกนำไปใช้ค้นหากับเอกสารที่มีอยู่ ทั้งนี้ก็ขึ้นอยู่กับการเลือกชนิดของข้อมูลที่จะค้นหาว่าต้องการค้นหาข้อมูลใด ดังที่กล่าวไว้ในหัวข้อ 5.4.2 • การค้นหาด้วยคำสำคัญทั่วไป จะค้นหาในส่วนของอักขระ โดยไม่คำนึงถึงขนาดตัวอักษร และใช้การเชื่อมต่อตรรกะเป็น "และ" เช่น bank Thai คือการค้นหาข้อมูลที่ต้องมีคำว่า bank และ Thai ในเอกสารนั้น อย่างเช่นเอกสารที่มีคำว่า Thai Farmer Bank หรือ Bank thailand ก็จะถูกพบโดยคำสำคัญดังกล่าว • หากต้องการค้นหาเอกสารที่ไม่มีคำสำคัญที่ระบุรวมอยู่ด้วย จะต้องใช้เครื่องหมายลบนำหน้าคำสำคัญนั้น (แบบติดกัน) เช่น print -laser คือการค้นหาเอกสารที่มีคำ (หรือส่วนของคำ) ว่า print แต่ต้องไม่มีคำว่า laser ในเอกสารนั้น • หากต้องการให้ค้นหาเอกสารที่มีคำที่ตรงกับที่ต้องการ โดยคำนึงถึงทั้งขนาดอักษร การจัดเรียงคำ และการเว้นช่องว่าง จะสามารถทำได้โดยการใช้เครื่องหมาย ". . ." หรือ '. . .' กับคำสำคัญที่ต้องการ เช่น "Windows NT" backup service คือการค้นหาเอกสารที่ต้องมีคำว่า Windows NT ซึ่ง

	ต้องตรงทั้งตามรูปแบบและขนาดตัวอักษร รวมทั้งต้องมีคำว่า backup และ service ซึ่งสามารถอยู่ในส่วนใดๆ ของเอกสารก็ได้ และขนาดตัวอักษรเล็กหรือใหญ่ก็ได้ ส่วนกรณีที่ไม่ต้องการคำนี้รวมอยู่ก็สามารถทำได้โดยระบุเครื่องหมายลบเช่นเดียวกัน เช่น -"Windows NT"

บทที่ 6
สรุปผลการวิจัยและข้อเสนอแนะ

ในบทนี้จะกล่าวถึงผลสรุปของงานวิจัย ปัญหาและข้อจำกัดในส่วนต่างๆของระบบ รวมทั้งข้อเสนอแนะที่สามารถนำไปปรับปรุงและพัฒนาได้ต่อไปในอนาคต โดยมีรายละเอียดดังนี้

6.1 สรุปผลการวิจัย

งานวิจัยนี้ได้ทำการออกแบบ และพัฒนาบริการค้นหาบริการ ทั้งนี้เพื่อเพิ่มคุณสมบัติการค้นหาบริการ และแก้ไขข้อจำกัดของการนำเสนอข้อมูลของบริการเทรดเดอร์ในระบบกระจายคอร์บา โดยผลที่ได้รับคือ โครงสร้างและต้นแบบของบริการเอสดีเอสที่รองรับการค้นหาบริการในระบบกระจายของคอร์บาได้ โดยช่วยแก้ไขปัญหาข้อจำกัดที่แต่เดิมผู้ใช้บริการเทรดเดอร์สามารถค้นหาได้เฉพาะข้อมูลบริการเท่านั้น แต่ผู้ใช้บริการเอสดีเอสสามารถเลือกค้นหาได้ทั้งชนิดของบริการ ส่วนต่อประสาน และบริการได้จากเทรดเดอร์หลายๆแหล่ง และยังสามารถรองรับการค้นหาด้วยภาษาค้นหาหลายรูปแบบ โดยเปิดโอกาสให้ผู้พัฒนาสามารถเลือกใช้เครื่องมือค้นหาที่สนับสนุนภาษาค้นหาใดก็ได้ และการค้นหาบริการสามารถใช้งานได้ทั้งผู้ใช้คอร์บาและผู้ใช้ผ่านเว็บ ซึ่งเป็นการเพิ่มทางเลือกให้กับผู้ใช้นอกเหนือจากระบบเดิมที่จำกัดอยู่ในโดเมนของคอร์บาเท่านั้น

6.2 ปัญหาและข้อจำกัดของงานวิจัย

1. เมื่อเปรียบเทียบรูปแบบการค้นหาบริการกับภาษาบังคับของเทรดเดอร์ สิ่งที่บริการเอสดีเอสไม่สามารถทำได้คือการค้นหาข้อมูลที่มีชนิดที่กำหนดโดยผู้ใช้ ซึ่งปัญหานี้เป็นผลมาจากข้อจำกัดของส่วนการแปลงคำอธิบายบริการที่ไม่สามารถแปลงข้อมูลที่มีชนิดเหล่านี้ได้ โดยจะแสดงได้เฉพาะชื่อของคุณสมบัติเท่านั้น ไม่สามารถค้นหาค่าภายในได้
2. ภาษาค้นหาเอ็กซ์เอ็มแอลที่งานวิจัยนี้ได้เลือกใช้คือเอ็กซ์คิวแอล ซึ่งมีโอกาสที่จะถูกเลิกใช้งานได้ในอนาคต เพราะเนื่องจากปัจจุบันนี้ได้เริ่มมีการกำหนดข้อเสนอเพื่อเป็นมาตรฐานสำหรับภาษาสอบถามเอ็กซ์เอ็มแอล ซึ่งก็คือเอ็กซ์คิวรี ดังที่กล่าวไว้ในบทที่ 2 แต่ในขณะนี้ยังไม่มีเครื่องมือค้นหาใดที่สนับสนุนเอ็กซ์คิวรี เพราะรูปแบบภาษายังอยู่ในช่วงการดำเนินการเพื่อให้เป็นมาตรฐาน
3. รูปแบบการเก็บเอกสารเอ็กซ์เอ็มแอลของบริการเอสดีเอสขึ้นอยู่กับรูปแบบการเก็บข้อมูลของเครื่องมือค้นหาที่นำมาใช้ ดังนั้นจึงเป็นข้อจำกัดในการเลือกใช้รูปแบบการเก็บข้อมูลเอ็กซ์เอ็มแอล
4. ในส่วนของผู้ดูแลบริการเอสดีเอสและผู้ดูแลบริการเทรดเดอร์ ยังไม่มีส่วนต่อประสานผู้ใช้ ที่จะอำนวยความสะดวกในการใช้งาน ไม่ว่าจะเป็นการลงทะเบียนบริการเทรดเดอร์ หรือการตรวจสอบและจัดการข้อมูลของบริการเอสดีเอส ซึ่งผู้ดูแลบริการจำเป็นต้องเขียนโปรแกรมเรียกใช้ส่วนต่อประสานเอง
5. ส่วนของการเชื่อมโยง ยังไม่ได้รับการพัฒนาในต้นแบบของงานวิจัยนี้ ดังนั้นจึงไม่ได้มีการตรวจสอบหรือวัดผลการทำงานว่าถูกต้องตามที่ได้ออกแบบไว้แล้วหรือไม่

6.3 ข้อเสนอแนะ

1. ควรปรับปรุงหรือพัฒนาให้ส่วนการแปลงคำอธิบายบริการสามารถรองรับการแปลงข้อมูลได้ทุกชนิด ทั้งนี้เพื่อให้บริการเอสดีเอสสามารถค้นหาข้อมูลได้สมบูรณ์ขึ้น
2. เนื่องจากช่วงเวลาที่เริ่มพัฒนาบริการเอสดีเอส การตัดสินใจเลือกใช้เครื่องมือค้นหาค่อนข้างจำกัด เพราะเครื่องมือค้นหาสำหรับเอ็กซ์เอ็มแอลยังมีการพัฒนาไม่มากนัก แต่ปัจจุบันมีเครื่องมือค้นหาหรือฐานข้อมูลที่สนับสนุนการค้นหาด้วยภาษาสอบถามเอ็กซ์เอ็มแอลเพิ่มมากขึ้น ดังนั้นหากมีการพัฒนาบริการ เอสดีเอสต่อไปสามารถนำเครื่องมือค้นหาที่มีประสิทธิภาพที่ดีขึ้นมาใช้งานได้
3. การพัฒนาบริการเอสดีเอส สามารถเพิ่มขยายส่วนของการรับข้อมูลให้สามารถรองรับข้อมูลที่มีโครงสร้างหรือดีทีดีต่างจากที่กำหนดได้ แต่ต้องเพิ่มรูปแบบการแปลงข้อมูลของเอกสารเอ็กซ์เอสแอลให้สามารถใช้กับส่วนย่อยที่หลากหลายขึ้นด้วย นอกจากนี้ยังสามารถพัฒนาให้รองรับคำอธิบายของคอมโพเนนท์ชนิดอื่นๆ ในระบบกระจายได้

รายการอ้างอิง

[1] Object Management Group. The Common Object Request Broker: Architecture and Specification Revision 2.2 1998. (February 1998)

[2] Object Management Group. Trading Object Service Specification Revised Edition 1997. (March 1997)

[3] Bray, T., Paoli, J. and McQueen, S. Extensible Markup Language (XML) 1.0 Specification. World Wide Web Consortium Recommendation 10 February 1998, Available from: http://www.w3.org/TR/REC-xml

[4] Nanekrangsan, W. A Transformation of Service Descriptions between the CORBA Trader Format and XML. Master's Thesis, Department of Computer Engineering, Graduate School, Chulalongkorn University, 2001.

[5] World Wide Web Consortium. XSL Transformations (XSLT) Version 1.0. World Wide Web Consortium Recommendation 16 November 1999, Available from: http://www.w3.org/TR/1999/REC-xslt-19991116

[6] Seacord, R.C., Hissam, S.A. and Wallnau, K.C. AGORA: A Search Engine for Software Components. IEEE Internet Computing 2 6 (November-December 1998): 62-70.

[7] Vasudevan, V. and Bannon, T. WebTrader: Discovery and Programmed Access to Web-Based Services (Draft). OBJS Technical Report 1999, Available from: http://www.objs.com/agility/tech-reports/9812-web-trader-paper/WebTraderPaper.html

[8] Tong, R. and Haolin, S. A Component Search Engine Model on Internet. In Proceedings of Tools'24 on Technology of Object-Oriented Languages 1997 1998: 393-396.

[9] McCaffery, M. and Scott, B. The Official VisiBroker for Java Handbook. Indiana, U.S.: Sams Publishing, 1999.

[10] Buneman, P. Semistructured Data. In Proceeding of the Sixteenth ACM SIGMOD Symposium on Principles of Database Systems (Tucson, Arizona) 1997: 117-121.

[11] Apparao, V., et al. Document Object Model (DOM) Level 1 Specification Version 1.0: World Wide Web Consortium Recommendation 1 Octoer 1998, Available from: http://www.w3.org/TR/ REC-DOM-Level-1

[12] World Wide Web Consortium. Extensible Stylesheet Language (XSL) Version 1.0. W3C Working Draft 27 March 2000, Available from: http://www.w3.org/TR/2000/WD-xsl-20000327/

[13] Fankhauser, P., Marchiori, M. and Robie, J. XML Query Requirements. World Wide Web Consortium Working Draft 31 January 2000, Available from: http://www.w3.org/TR/2000/WD-xmlquery-req-20000131

[14] World Wide Web Consortium. XQuery: A Query Language for XML. World Wide Web Consortium Working Draft 15 February 2001, Available from: http://www.w3.org/TR/xquery

[15] World Wide Web Consortium. XML-QL: A Query Language for XML. Submission to the World Wide Web Consortium 19 August 1998, Available from: http://www.w3.org/TR/1998/NOTE-xml-ql-19980819

[16] Robie, J., Lapp J. and Schach D. XML Query Language (XQL). Position paper of W3C QL'98 - The Query Languages Workshop 1998, Available from: http://www.w3.org/TandS/QL/QL98/pp/xql.html

[17] Cluet, S. and Simeon, J. YATL: a Functional and Declarative Language for XML. Submitted to ICFP'2000, Available from: http://www-db.research.bell-labs.com/user/simeon/icfp.ps

[18] Goldman, R., McHugh, J. and Widom, J. From semistructured data to XML: Migrating the Lore data model and query language. In Proceedings of the 2nd International Workshop on the Web and Databases (WebDB '99), Philadelphia, Pennsylvania, June 1999.

[19] JacORB - a free Java ORB, Available from: http://www.inf.fu-berlin.de/~brose/jacorb.

[20] GMD-IPSI XQL Engine, Available from: http://xml.darmstadt.gmd.de/xql

[21] Oracle Parser v2, Available from: http://technet.oracle.com

ภาคผนวก

ภาคผนวก ก

ดีทีดีสำหรับข้อมูลที่จัดเก็บโดยศูนย์จัดเตรียมบริการภายในบริการเอสดีเอส

1. ดีทีดีสำหรับข้อมูลชนิดของบริการ (Service Type Description DTD)

```
<!ELEMENT ServiceTypeDescription (Interface?, TraderServiceType, TraderName?)>
<!ELEMENT Interface (BaseInterfaces?, Constant*, Attribute*, Operation*)>
<!ATTLIST Interface   Id     CDATA #REQUIRED Name   CDATA #REQUIRED
                      Version CDATA #REQUIRED>
<!ELEMENT BaseInterfaces (BaseInterface*, Link*)>
<!ELEMENT BaseInterface EMPTY>
<!ATTLIST BaseInterface  Id   CDATA #REQUIRED Name CDATA #REQUIRED>
<!ELEMENT Link EMPTY>
<!ATTLIST Link Source CDATA #REQUIRED Dest   CDATA #REQUIRED>
<!ELEMENT Constant EMPTY>
<!ATTLIST Constant Id CDATA #REQUIRED  Name   CDATA #REQUIRED
                Version CDATA #REQUIRED Type     CDATA #REQUIRED
                Value   CDATA #REQUIRED  Derived (YES | NO) "NO">
<!ELEMENT Attribute EMPTY>
<!ATTLIST Attribute Id  CDATA #REQUIRED     Name   CDATA #REQUIRED
                Version CDATA #REQUIRED  Type    CDATA #REQUIRED
                Mode    (NORMAL | READONLY) "NORMAL"
                Derived (YES | NO) "NO">
<!ELEMENT Operation (Parameter*, Exception*, Context*)>
<!ATTLIST Operation  Id  CDATA #REQUIRED  Name   CDATA #REQUIRED
                Version CDATA #REQUIRED  Type    CDATA #REQUIRED
                Mode    (NORMAL | ONEWAY) "NORMAL"
                Derived (YES | NO) "NO">
<!ELEMENT Parameter EMPTY>
<!ATTLIST Parameter Name  CDATA #REQUIRED  Type   CDATA #REQUIRED
                Mode    (IN | OUT | INOUT) "IN">
<!ELEMENT Exception (Member)*>
<!ATTLIST Exception  Id     CDATA #REQUIRED     Name   CDATA #REQUIRED
                Version CDATA #REQUIRED  Derived (YES | NO) "NO">
<!ELEMENT Member EMPTY>
<!ATTLIST Member Name  CDATA #REQUIRED  Type   CDATA #REQUIRED>
<!ELEMENT Context (#PCDATA)>
<!ELEMENT TraderServiceType (BaseServiceTypes?, Property*)>
<!ATTLIST TraderServiceType Id CDATA #REQUIRED Name CDATA REQUIRED
                Masked (YES | NO) "NO">
<!ELEMENT BaseServiceTypes (BaseServiceType*, Link*)>
<!ELEMENT BaseServiceType EMPTY>
<!ATTLIST BaseServiceType Name CDATA #REQUIRED>
<!ELEMENT Property EMPTY>
<!ATTLIST Property Name CDATA #REQUIRED  Type CDATA #REQUIRED
                Mode (NORMAL|READONLY|MANDATORY|
                MANDATORY_READONLY) "NORMAL"
                Derived (YES | NO) "NO">
<!ELEMENT TraderName (#PCDATA)>
```

2. ดีทีดีสำหรับข้อมูลบริการ (Service Offer DTD)

```
<!ELEMENT ServiceOfferDescription (OfferType, Property*, ObjectReference)>
<!ATTLIST ServiceOfferDescription TraderName CDATA #REQUIRED
                  OfferId CDATA #REQUIRED>
<!ELEMENT OfferType EMPTY>
<!ATTLIST OfferType Name CDATA #REQUIRED>
<!ELEMENT Property (DynamicPropEval, ExtraInfo)?>
<!ATTLIST Property Name CDATA #REQUIRED
                  Value CDATA #IMPLIED>
<!ELEMENT DynamicPropEval (#PCDATA)>
<!ATTLIST DynamicPropEval ReturnType CDATA #REQUIRED>
<!ELEMENT ExtraInfo EMPTY>
<!ATTLIST ExtraInfo Type  CDATA #REQUIRED
                  Value CDATA #REQUIRED>
<!ELEMENT ObjectReference (#PCDATA)>
```

3. ดีทีดีสำหรับข้อมูลบริการเทรดเดอร์ (Trader Profile DTD)

```
<!ELEMENT TraderProfile (ObjectRef+, Location?, AdminProfile+, OtherDetail?)>
   <!ATTLIST TraderProfile TraderName CDATA #REQUIRED>
<!ELEMENT ObjectRef (#PCDATA)>
<!ELEMENT Location (#PCDATA)>
<!ELEMENT AdminProfile (Address*, Email+, OtherDetail?)>
   <!ATTLIST AdminProfile AdminName CDATA #REQUIRED>
<!ELEMENT Address (#PCDATA)>
<!ELEMENT Email (#PCDATA)>
<!ELEMENT OtherDetail (#PCDATA)>
```

ภาคผนวก ข
เอกสารเอ็กซ์เอสแอลสำหรับการแสดงผลลัพธ์
ของส่วนต่อประสานสำหรับการค้นหา

1. เอกสารเอ็กซ์เอสแอลสำหรับการแปลงผลลัพธ์ให้อยู่ในรูปแบบสมบูรณ์ (MAXIMUM)

```
1 <xsl:stylesheet
2         xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
3         version="1.0"
4 >
5 <xsl:template match="/">
6   <ResultSet>
7     <xsl:apply-templates select="*"/>
8  </ResultSet>
9 </xsl:template>
10
11 <xsl:template match="ServiceTypeDescription">
12 <row>
13  <ServiceTypeDescription>
14  <xsl:copy-of select="*"/>
15  </ServiceTypeDescription>
16 </row>
17 </xsl:template>
18
19 <xsl:template match="ServiceOfferDescription">
20 <row>
21 <xsl:variable name="oid" select="@OfferId"/>
22 <xsl:variable name="tname" select="@TraderName"/>
23  <ServiceOfferDescription OfferId="{$oid}" TraderName="{$tname}">
24  <xsl:copy-of select="*"/>
25  </ServiceOfferDescription>
26 </row>
27 </xsl:template>
28
29 </xsl:stylesheet>
```

คำอธิบายเอกสารเอ็กซ์เอสแอลสำหรับการแปลงผลลัพธ์ให้อยู่ในรูปแบบสมบูรณ์

บรรทัดที่	คำอธิบาย
1-4	ระบุว่าเป็นเอกสารเอ็กซ์เอสแอลที่เขียนขึ้นตามข้อกำหนดของเอ็กซ์เอสแอลที รุ่นที่ 1.0
5-9	แสดงการเลือกข้อมูลทั้งหมดจากเอกสารมาประมวลผล โดยใช้ป้ายระบุ <ResultSet> เป็นรากของเอกสาร
11	เป็นการจับคู่กับแบบอย่างของเอกสารที่เป็นชนิดของบริการ <ServiceTypeDescription>

บรรทัดที่	คำอธิบาย
12-16	ทำการเพิ่มป้ายระบุ <row> กับบัพชนิดของบริการที่พบ และ กำหนดป้ายระบุ <ServiceTypeDescription> เพื่อระบุว่าเป็นข้อมูลชนิดของบริการ และเลือกข้อมูลภายในทั้งหมดมาแสดงด้วยป้ายระบุ <xsl:copy-of select="*"/>
19	เป็นการจับคู่กับแบบอย่างของเอกสารที่เป็นบริการ <ServiceOfferDescription>
20	ทำการเพิ่มป้ายระบุ <row> กับบัพของบริการที่พบ เช่นเดียวกันกับข้างต้น
21-22	กำหนดค่าของลักษณะประจำ TraderName ให้กับตัวแปรชื่อ tname และค่าของ OfferId ให้กับตัวแปรชื่อ oid เพื่อนำไปใช้งานต่อไป
23	กำหนดลักษณะประจำ TraderName และ OfferId และกำหนดค่าจากตัวแปรข้างต้น เพื่อรวมเข้ากับป้ายระบุ <ServiceOfferDescription> ดังตัวอย่างเช่น <ServiceOfferDescription OfferId="Accont/1" TraderName="CU_Trader">
24	เลือกค่าเดิมทั้งหมดของบริการ

2. เอกสารเอ็กซ์เอสแอลสำหรับการแปลงผลลัพธ์ให้อยู่ในรูปแบบย่อ (MINIMUM)

```
1 <xsl:stylesheet
2         xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
3         version="1.0"
4 >
5 <xsl:template match="/">
6  <ResultSet>
7    <xsl:apply-templates select="*"/>
8  </ResultSet>
9 </xsl:template>
10
11 <xsl:template match="ServiceTypeDescription">
12 <row>
13  <ServiceTypeDescription>
14   <xsl:apply-templates select="TraderName"/>
15   <xsl:apply-templates select="Interface"/>
16   <xsl:apply-templates select="TraderServiceType"/>
17  </ServiceTypeDescription>
18 </row>
19 </xsl:template>
20
21 <xsl:template match="TraderName">
22 <xsl:copy-of select="."/>
23 </xsl:template>
24
25 <xsl:template match="Interface">
26  <Interface>
27  <RepositoryId><xsl:value-of select="@Id"/></RepositoryId>
28  <xsl:apply-templates select="Attribute"/>
29  <xsl:apply-templates select="Operation"/>
30  </Interface>
31 </xsl:template>
```

```
32 <xsl:template match="Attribute">
33 <Attribute>(<xsl:value-of select="@Type"/>)<xsl:text> </xsl:text>
34 <xsl:value-of select="@Name"/>
35 </Attribute>
36 </xsl:template>
37
38 <xsl:template match="Operation">
39 <Operation>(<xsl:value-of select="@Type"/>)<xsl:text> </xsl:text>
40 <xsl:value-of select="@Name"/>
41 (<xsl:apply-templates select="Parameter"/> )</Operation>
42 </xsl:template>
43
44 <xsl:template match="Parameter">
45       <xsl:value-of select="@Mode"/>
46       <xsl:text>-</xsl:text><xsl:value-of select="@Type"/>
47       <xsl:text>-</xsl:text><xsl:value-of select="@Name"/>
48 <xsl:if test="not(position()=last())">,
49 </xsl:if>
50 </xsl:template>
51
52 <xsl:template match="TraderServiceType">
53       <TraderServiceType Name="{@Name}">
54          <xsl:apply-templates select="Property"/>
55       </TraderServiceType>
56 </xsl:template>
57 <xsl:template match="Property">
58       <Property Type="{@Type}"><xsl:value-of select="@Name"/></Property>
59 </xsl:template>
60
61 <xsl:template match="ServiceOfferDescription">
62 <row>
63 <xsl:variable name="oid" select="@OfferId"/>
64 <xsl:variable name="tname" select="@TraderName"/>
65  <ServiceOfferDescription OfferId="{$oid}" TraderName="{$tname}">
66  <xsl:copy-of select="*"/>
67  </ServiceOfferDescription>
68 </row>
69 </xsl:template>
70 </xsl:stylesheet>
```

คำอธิบายสำหรับเอกสารเอ็กซ์เอสแอลสำหรับการแปลงผลลัพธ์ให้อยู่ในรูปแบบย่อ

บรรทัดที่	คำอธิบาย
1-13	ลักษณะเดียวกับรูปแบบสมบูรณ์ข้างต้น
14-16	เป็นการระบุว่าจะเลือกส่วนย่อยใดที่อยู่ภายใต้ป้ายระบุ <ServiceTypeDescription> มาประมวลผลบ้าง ซึ่งการแสดงผลรูปในแบบย่อจะเลือกใช้ส่วนย่อยดังนี้ - <TraderName> สำหรับข้อมูลชื่อบริการเทรดเดอร์ - <Interface> สำหรับข้อมูลส่วนต่อประสาน - <TraderServiceType> สำหรับข้อมูลชนิดของบริการในบริการเทรดเดอร์ ซึ่งในแต่ละส่วนย่อยจะถูกกำหนดในรายละเอียดต่อไป
21-23	เป็นรายละเอียดของส่วนย่อย <TraderName> โดยดึงข้อมูลออกมาแสดงด้วยป้ายระบุ <xsl:copy-of select="."/>
25-31	เป็นการจัดการกับข้อมูลในส่วนย่อย <Interface> โดยนำเอาค่าของลักษณะประจำ Id มากำหนดในป้ายระบุ <RepositoryId> และเลือกส่วนย่อย <Attribute> และ <Operation> ซึ่งทั้งหมดถูกเก็บในป้ายระบุ <Interface>
32-36	เป็นรายละเอียดของการนำเสนอข้อมูลส่วนย่อย <Attribute> โดยแสดงด้วยรูปแบบดังนี้ (ชนิดของลักษณะประจำ) ชื่อของลักษณะประจำ เช่น (String) processName
38-42	เป็นรายละเอียดของการแสดงข้อมูลส่วนย่อย <Operation> โดยแสดงด้วยรูปแบบดังนี้ (ชนิดของค่าที่ส่งกลับ) ชื่อของตัวกระทำการ (พารามิเตอร์) เช่น (void) sum_arrays (in-long_array-arr1, in-long_array-arr2, out-long_aray-result)
44-50	เป็นรายละเอียดของการเลือกข้อมูลส่วนย่อย <Parameter> จากป้ายระบุ <Operation> ข้างต้น โดยพารามิเตอร์ประกอบด้วยลักษณะประจำทั้ง 3 แบบและคั่นด้วยเครื่องหมาย "-" ดังนี้ Mode-Type-Name และแยกพารามิเตอร์แต่ละตัวด้วยเครื่องหมาย "," ไปจนถึงพารามิเตอร์ตัวสุดท้าย ดังตัวอย่างข้างต้น
42-56	เป็นรายละเอียดของการนำเสนอข้อมูลในส่วนย่อย <TraderServiceType> โดยนำเอาลักษณะประจำ Name ซึ่งเป็นชื่อชนิดของบริการมารวมในป้ายระบุ <TraderServiceType> และเลือกเฉพาะข้อมูลใน <Property> มาแสดงเท่านั้น
57-59	เป็นรายละเอียดการเลือกข้อมูลในส่วนย่อย <Property> มาจัดรูปแบบใหม่ โดยดึงลักษณะประจำ Type ใส่ไว้ในป้ายระบุ <Property> และกำหนดให้ค่าของลักษณะประจำ Name อยู่ภายใต้ส่วนย่อย <Property> ตัวอย่างเช่น <Property Type="string">PrinterName</Property>
61-69	เป็นการดึงข้อมูลของส่วนย่อย <ServiceOfferDescription> ซึ่งเป็นลักษณะเดียวกับรูปแบบข้างต้น เพราะข้อมูลบริการมีน้อยอยู่แล้ว ดังนั้นจึงไม่มีการตัดส่วนใดทิ้งไป

3. เอกสารเอ็กซ์เอสแอลสำหรับการแปลงผลลัพธ์เป็นเอกสารเอชทีเอ็มแอลในรูปแบบสมบูรณ์

```
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"
>

<xsl:template match="ResultSet">
<xsl:apply-templates select="row"/>
</xsl:template>

<xsl:template match="row">
<xsl:apply-templates select="ServiceTypeDescription"/>
<xsl:apply-templates select="ServiceOfferDescription"/>
</xsl:template>

<xsl:template match="ServiceTypeDescription">
<tr><td valign="top"><B><font size="+0.5" color="gray">
<xsl:number level="any" from="ResultSet"/>.</font></B></td>
<td><B><U><font size="+0.5" color="maroon">Service type description:</font>
</U></B><BR/>
<xsl:apply-templates select="TraderName"/>
<xsl:apply-templates select="TraderServiceType"/>
<xsl:apply-templates select="Interface"/>
<hr/></td></tr>
</xsl:template>

<xsl:template match="TraderName">
<B><font color="navy">TraderName :</font></B>
<xsl:variable name="tname" select="."/>
<A HREF="http://isel.cp.eng.chula.ac.th/servlet-bin/servlet/g42wsp.DescribeTrader?
TraderName={$tname}">
<xsl:value-of select="."/></A><BR/>
</xsl:template>

<!—MATCHING TRADER SERVICE TYPE -->

<xsl:template match="TraderServiceType">
<B><font color="navy">TraderServiceType:</font></B><BR/>
<DD/><B>Id:</B> <xsl:value-of select="@Id"/><BR/>
<DD/><B>Name:</B> <xsl:value-of select="@Name"/><BR/>
<DD/><B>Masked:</B> <xsl:value-of select="@Masked"/><BR/>
<xsl:apply-templates select="BaseServiceTypes"/>
<xsl:apply-templates select="Property"/>
</xsl:template>

<xsl:template match="BaseServiceTypes">
<DD/><B>BaseServiceTypes:</B> <BR/>
<xsl:apply-templates select="BaseServiceType"/>
<xsl:apply-templates select="Link"/>
</xsl:template>

<xsl:template match="BaseServiceType">
<DD/><DD/><B>ServiceType Name:</B> <xsl:value-of select="@Name"/><BR/>
```

```
52 </xsl:template>
53 <xsl:template match="Link">
54 <DD/><DD/><B>Link Source: </B><xsl:value-of select="@Source"/>,
55 <B>Dest:</B> <xsl:value-of select="@Dest"/><BR/>
56 </xsl:template>
57
58 <xsl:template match="TraderServiceType/Property">
59 <DD/><B>Property Name:</B> <xsl:value-of select="@Name"/>,
60 <B>Type:</B> <xsl:value-of select="@Type"/>,
61 <B>Mode:</B> <xsl:value-of select="@Mode"/>,
62 <B>Derived:</B> <xsl:value-of select="@Derived"/><BR/>
63 </xsl:template>
64
65 <!-- MATCHING INTERFACE DEFINITION -->
66
67 <xsl:template match="Interface">
68 <B><font color="navy">Interface description:</font></B> <BR/>
69 <DD/><B>Id:</B> <xsl:value-of select="@Id"/><BR/>
70 <DD/><B>Name:</B> <xsl:value-of select="@Name"/><BR/>
71 <DD/><B>Version:</B> <xsl:value-of select="@Version"/><BR/>
72
73 <DD/><xsl:apply-templates select="BaseInterfaces"/>
74 <xsl:apply-templates select="Constant"/>
75 <xsl:apply-templates select="Attribute"/>
76 <xsl:apply-templates select="Operation"/>
77
78 <xsl:variable name="tname" select="../TraderName"/>
79 <xsl:variable name="stname" select="../TraderServiceType/@Name"/>
80 <B><U><font color="maroon">
81 <A HREF="http://isel.cp.eng.chula.ac.th/servlet-bin/servlet/g42wsp.PreTradeOffer?
82 TraderName={$tname}&amp;amp;TypeName={$stname}">
83 Trader offer from this service type</A></font></U></B>
84 </xsl:template>
85
86 <xsl:template match="BaseInterfaces">
87 <B>BaseInterfaces :</B> <BR/>
88 <xsl:apply-templates select="BaseInterface"/>
89 <xsl:apply-templates select="Link"/>
90 </xsl:template>
91
92 <xsl:template match="BaseInterface">
93 <DD/><DD/><B>Interface Id: </B><xsl:value-of select="@Id"/>,
94 <B>Name:</B> <xsl:value-of select="@Name"/><BR/>
95 </xsl:template>
96
97 <xsl:template match="Constant">
98 <DD/><B>Constant Id:</B> <xsl:value-of select="@Id"/>,
99 <B>Name:</B> <xsl:value-of select="@Name"/>
100, <B>Version:</B> <xsl:value-of select="@Version"/> ,
101 <B>Type:</B> <xsl:value-of select="@Type"/>
102, <B>Value:</B> <xsl:value-of select="@Value"/>,
103 <B>Derived:</B> <xsl:value-of select="@Derived"/><BR/>
104 </xsl:template>
105
```

```
106 <xsl:template match="Attribute">
107 <DD/><B>Attribute Id:</B> <xsl:value-of select="@Id"/>,
108 <B>Name:</B> <xsl:value-of select="@Name"/>
109 , <B>Version:</B> <xsl:value-of select="@Version"/>,
110 <B>Type:</B> <xsl:value-of select="@Type"/>
111 , <B>Mode:</B> <xsl:value-of select="@Mode"/>,
112 <B>Derived:</B> <xsl:value-of select="@Derived"/><BR/>
113 </xsl:template>
114
115 <xsl:template match="Operation">
116 <DD/><B>Operation Id:</B> <xsl:value-of select="@Id"/>,
117 <B>Name:</B> <xsl:value-of select="@Name"/>
118 , <B>Version:</B> <xsl:value-of select="@Version"/>,
119 <B>Type:</B> <xsl:value-of select="@Type"/>
120 , <B>Mode:</B> <xsl:value-of select="@Mode"/>,
121 <B>Derived:</B> <xsl:value-of select="@Derived"/><BR/>
122 <xsl:apply-templates select="Parameter"/>
123 <xsl:apply-templates select="Exception"/>
124 <xsl:apply-templates select="Context"/>
125 </xsl:template>
126
127 <xsl:template match="Parameter">
128 <DD/><DD/><B>Parameter Name:</B> <xsl:value-of select="@Name"/>,
129 <B>Type:</B> <xsl:value-of select="@Type"/>
130 , <B>Mode:</B> <xsl:value-of select="@Mode"/><BR/>
131 </xsl:template>
132
133 <xsl:template match="Exception">
134 <DD/><DD/><B>Exception Id:</B> <xsl:value-of select="@Id"/>,
135 <B>Name:</B> <xsl:value-of select="@Name"/>
136 , <B>Version:</B> <xsl:value-of select="@Version"/>,
137 <B>Derived:</B> <xsl:value-of select="@Derived"/><BR/>
138 <xsl:apply-templates select="Member"/>
139 </xsl:template>
140
141 <xsl:template match="Member">
142 <DD/><DD/><DD/><B>Member Name:</B> <xsl:value-of select="@Name"/>,
143 <B>Type:</B> <xsl:value-of select="@Type"/><BR/>
144 </xsl:template>
145
146 <xsl:template match="Context">
147 <DD/><DD/><B>Context: </B><xsl:value-of select="."/><BR/>
148 </xsl:template>
149
150 <!-- MATCHING SERVICE OFFER DESCRIPTION -->
151
152 <xsl:template match="ServiceOfferDescription">
153 <tr><td valign="top"><B><font size="+0.5" color="gray">
154 <xsl:number level="any" from="ResultSet"/>.</font></B></td>
155 <td><B><U><font size="+0.5" color="maroon">Service offer description:
156 </font></U></B><BR/>
157 <B>Trader Name: </B><xsl:text> </xsl:text>
158 <xsl:variable name="tname" select="@TraderName"/>
159 <A HREF="http://isel.cp.eng.chula.ac.th/servlet-bin/servlet/g42wsp.DescribeTrader?
```

```
160 TraderName={$tname}">
161 <xsl:value-of select="@TraderName"/></A><BR/>
162
163 <xsl:apply-templates select="OfferType"/>
164 <xsl:apply-templates select="Property"/>
165 <xsl:apply-templates select="ObjectReference"/>
166 <hr/></td></tr>
167 </xsl:template>
168
169 <xsl:template match="OfferType">
170 <B>Offer Type: </B><xsl:text> </xsl:text>
171 <xsl:variable name="stname" select="@Name"/>
172 <A HREF="http://isel.cp.eng.chula.ac.th/servlet-bin/servlet/g42wsp.DescribeType?
173 TraderName={$tname}&amp;TypeName={$stname}">
174 <xsl:value-of select="@Name"/></A><BR/>
175 </xsl:template>
176
177 <xsl:template match="Property">
178 <B>Property Name: </B><xsl:value-of select="@Name"/>,
179 <B>Value:</B> <xsl:value-of select="@Value"/><BR/>
180 <xsl:apply-templates select="DynamicPropEval"/>
181 <xsl:apply-templates select="ExtraInfo"/>
182 </xsl:template>
183
184 <xsl:template match="DynamicPropEval">
185 <DD/><B>DynamicPropEval, Return Type:</B>
186 <xsl:value-of select="@ReturnType"/>
187 <DD/><B>ObjectReference:</B> <xsl:value-of select="."/><BR/>
188 </xsl:template>
189
190 <xsl:template match="ExtraInfo">
191 <DD/><B>ExtraInfo Type:</B> <xsl:value-of select="@Type"/>,
192 <B>Value: </B><xsl:value-of select="@Value"/><BR/>
193 </xsl:template>
194
195 <xsl:template match="ObjectReference">
196 <B>ObjectReference:</B> <xsl:value-of select="."/><BR/>
197 </xsl:template>
198
199 </xsl:stylesheet>
```

คำอธิบายเอกสารเอ็กซ์เอสแอลสำหรับการแปลงผลลัพธ์เป็นเอกสารเอชทีเอ็มแอลในรูปแบบสมบูรณ์

บรรทัดที่	คำอธิบาย
5-7	ทำการค้นหารากของเอกสารด้วยส่วนย่อยชื่อ <ResultSet> และเลือกส่วนย่อย <row> ที่อยู่ภายใน
9-12	เมื่อพบส่วนย่อย <row> ก็จะเลือกส่วนย่อย <ServiceTypeDescription> และ <ServiceOfferDescription>

บรรทัดที่	คำอธิบาย
14-23	เมื่อพบส่วนย่อย <ServiceTypeDescription> จะกำหนดให้ข้อมูลทั้งหมดอยู่ภายใต้ส่วนย่อยแถวของตาราง และใส่หมายเลขลำดับรวมทั้งข้อความกำกับให้แต่ละชนิดของบริการที่พบ และเลือกส่วนย่อยภายในคือ <TraderName>, <TraderServiceType> และ <Interface>
25-31	เป็นรายละเอียดของการนำเสนอข้อมูลในส่วนย่อย <TraderName> ซึ่งเป็นข้อมูลชื่อของบริการเทรดเดอร์ โดยดึงข้อมูลภายในส่วนย่อยออกมาแสดงและเพิ่มข้อมูลเชื่อมโยงไปยังโปรแกรมอธิบายบริการเทรดเดอร์ โดยสร้างเป็นตัวแปรที่เก็บค่าชื่อของเทรดเดอร์ คือ tname และส่งค่าของตัวแปรนี้ไปเป็นพารามิเตอร์ของโปรแกรม
35-42	เมื่อพบส่วนย่อย <TraderServiceType> จะเลือกข้อมูลของ Id, Name และ Masked ออกมาแสดง ซึ่งทั้งหมดเป็นลักษณะประจำภายในป้ายระบุนี้ และเลือกข้อมูลของ <BaseServiceTypes> และ <Property> มาแสดง
44-48	สำหรับข้อมูลในส่วนย่อย <BaseServiceTypes> สามารถมี <BaseServiceType> และ <Link> ได้หลายชุด ดังนั้นจึงทำการเลือกข้อมูลภายในต่อไป
50-56	เป็นรายละเอียดของการนำเสนอข้อมูลในส่วนย่อย <BaseServiceType> และ <Link>
58-63	เป็นรายละเอียดของการนำเสนอข้อมูลในส่วนย่อย <Property> โดยการอ้างถึงจำเป็นต้องระบุว่า เป็นส่วนย่อย <Property> ที่อยู่ภายใต้ส่วนย่อย <TraderServiceType> เพราะการนำเสนอข้อมูลของส่วนย่อย <Property> จะใช้ชื่อเดียวกันกับของส่วนย่อย <ServiceOfferDescription> โดยจะดึงค่าของลักษณะประจำ Name, Type, Mode และ Derived ออกมาแสดงผล
67-71	เป็นรายละเอียดของการนำเสนอข้อมูลในส่วนย่อย <Interface> ซึ่งเป็นข้อมูลส่วนต่อประสานของชนิดของบริการนี้ โดยจะแสดงข้อมูลในลักษณะประจำ Id, Name และ Version ของส่วนต่อประสาน
73-76	เลือกจับคู่ส่วนย่อย BaseInterfaces, Constant, Attribute และ Operation
78-84	เป็นการสร้างการเชื่อมโยงให้กับชนิดของบริการ ซึ่งสามารถเชื่อมโยงไปยังโปรแกรมสำหรับค้นหาบริการ PreTradeOffer ได้ โดยการกำหนดตัวแปรให้เก็บค่าชื่อบริการเทรดเดอร์ (บรรทัดที่ 78) และชื่อชนิดของบริการ (บรรทัดที่ 79) และส่งค่าเหล่านี้ผ่านทางพารามิเตอร์ของโปรแกรม
86-90	เป็นรายละเอียดของการนำเสนอข้อมูลในส่วนย่อย <BaseInterfaces> ซึ่งสามารถมีส่วนย่อย <BaseInterface> และ <Link> ได้มากกว่า 1 ชุด

บรรทัดที่	คำอธิบาย
92-113	เป็นรายละเอียดของการนำเสนอข้อมูลในส่วนย่อย <BaseInterface>, <Constant> และ <Attribute> ซึ่งทั้งหมดเป็นการดึงข้อมูลจากลักษณะประจำภายในป้ายระบุมาแสดงผล
115-125	เป็นรายละเอียดของการนำเสนอข้อมูลในส่วนย่อย <Operation> โดยดึงข้อมูลจากลักษณะประจำมาแสดงผล และจับคู่ส่วนย่อย Parameter, Exception และ Context
127-148	เป็นรายละเอียดของการนำเสนอข้อมูลในส่วนย่อย <Parameter>, <Exception> และ <Context> ซึ่งทั้งหมดเป็นการดึงข้อมูลจากลักษณะประจำภายในป้ายระบุมาแสดง
152-161	สำหรับส่วนย่อยของบริการ <ServiceOfferDescription> ก็จะทำลักษณะเดียวกันกับชนิดของบริการคือ การกำหนดแถวของตาราง ใส่ค่าตัวเลขลำดับ และระบุว่าเป็นข้อมูลบริการ จากนั้นจะนำข้อมูลในลักษณะประจำ TraderName มาแสดง และเพิ่มข้อมูลเชื่อมโยงไปยังโปรแกรมแสดงรายละเอียดของบริการเทรดเดอร์ โดยส่งค่าพารามิเตอร์เป็นชื่อของบริการเทรดเดอร์จากตัวแปร tname
163-167	เลือกส่วนย่อย <OfferType>, <Property> และ <ObjectReference>
169-175	เป็นรายละเอียดของการนำเสนอข้อมูลในส่วนย่อย <OfferType> โดยการดึงข้อมูลจากลักษณะประจำ Name มาแสดง และสร้างส่วนเชื่อมโยงไปยังโปรแกรมสำหรับดูรายละเอียดของชนิดของบริการ DescribeType โดยส่งค่าชื่อบริการเทรดเดอร์และชื่อชนิดของบริการเป็นพารามิเตอร์
177-182	เป็นรายละเอียดของการนำเสนอข้อมูลในส่วนย่อย <Property> ซึ่งนำข้อมูลลักษณะประจำ Name และ Value มาแสดง และเลือกส่วนย่อย DynamicPropEval และ ExtraInfo
184-188	สำหรับข้อมูลส่วนย่อย <DynamicPropEval> เป็นการระบุว่าคุณสมบัตินี้เป็นแบบไดนามิค ซึ่งจะแสดงค่าลักษณะประจำ ReturnType และค่าอ้างอิงวัตถุของคุณสมบัตินี้
190-197	เป็นรายละเอียดของการนำเสนอข้อมูลในส่วนย่อย <ExtraInfo> และ <ObjectReference>

4.เอกสารเอ็กซ์เอสแอลสำหรับการแปลงผลลัพธ์เป็นเอกสารเอชทีเอ็มแอลในรูปแบบย่อ

```
1   <xsl:stylesheet
2         xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"
3   >
4
5   <xsl:template match="ResultSet">
6   <xsl:apply-templates select="row"/>
7   </xsl:template>
8
9   <xsl:template match="row">
10  <xsl:apply-templates select="ServiceTypeDescription"/>
```

```
11 <xsl:apply-templates select="ServiceOfferDescription"/>
12 </xsl:template>
13
14 <xsl:template match="ServiceTypeDescription">
15 <tr><td valign="top"><B><font size="+0.5" color="gray">
16 <xsl:number level="any" from="ResultSet"/>.</font></B></td>
17 <td><B><U><font size="+0.5" color="maroon">Service type description:
18 </font></U></B><BR/>
19 <xsl:apply-templates select="TraderName"/>
20 <xsl:apply-templates select="TraderServiceType"/>
21 <xsl:apply-templates select="Interface"/>
22 <hr/></td></tr>
23 </xsl:template>
24
25 <xsl:template match="TraderName">
26 <B><font color="navy">TraderName: </font></B>
27 <xsl:variable name="tname" select="."/>
28 <A HREF="http://isel.cp.eng.chula.ac.th/servlet-bin/servlet/g42wsp.DescribeTrader?
29 TraderName={$tname}">
30 <xsl:value-of select="."/></A><BR/>
31 </xsl:template>
32
33 <!-- MATCHING SERVICE TYPE DESCRIPTION -->
34
35 <xsl:template match="TraderServiceType">
36 <B><font color="navy">TraderServiceType:</font></B><BR/>
37 <DD/><B>Service type name:</B> <xsl:value-of select="@Name"/> <BR/>
38 <xsl:apply-templates select="Property"/>
39 </xsl:template>
40
41 <xsl:template match="TraderServiceType/Property">
42 <DD/><B>Property: </B><xsl:value-of select="@Type"/>
43 <xsl:text> </xsl:text><xsl:value-of select="."/><BR/>
44 </xsl:template>
45
46 <!-- MATCHING INTERFACE DESCRIPTION -->
47
48 <xsl:template match="Interface">
49 <font color="navy"><B>Interface:</B></font><BR/>
50 <xsl:apply-templates select="RepositoryId"/>
51 <xsl:apply-templates select="Attribute"/>
52 <xsl:apply-templates select="Operation"/>
53
54 <xsl:variable name="tname" select="../TraderName"/>
55 <xsl:variable name="stname" select="../TraderServiceType/@Name"/>
56 <B><U><font color="maroon">
57 <A HREF="http://isel.cp.eng.chula.ac.th/servlet-bin/servlet/g42wsp.PreTradeOffer?
58 TraderName={$tname}&amp;TypeName={$stname}">
59 Trader offer from this service type</A></font></U></B>
60 </xsl:template>
61
62 <xsl:template match="RepositoryId">
63 <DD/><B>Id: </B><xsl:value-of select="."/><BR/>
64 </xsl:template>
```

```
65
66 <xsl:template match="Attribute">
67 <DD/><B>Attribute: </B><xsl:value-of select="."/><BR/>
68 </xsl:template>
69
70 <xsl:template match="Operation">
71 <DD/><B>Operation: </B><xsl:value-of select="."/><BR/>
72 </xsl:template>
73
74 <!-- MATCHING SERVICE OFFER DESCRIPTION -->
75
76 <xsl:template match="ServiceOfferDescription">
77 <tr><td valign="top"><B><font size="+0.5" color="gray">
78 <xsl:number level="any" from="ResultSet"/>.</font></B></td>
79 <td><B><U><font size="+0.5" color="maroon">Service offer description:
80 </font></U></B><BR/>
81 <B>Trader Name: </B><xsl:text> </xsl:text>
82 <xsl:variable name="tname" select="@TraderName"/>
83 <A HREF="http://isel.cp.eng.chula.ac.th/servlet-bin/servlet/g42wsp.DescribeTrader?
84 TraderName={$tname}">
85 <xsl:value-of select="@TraderName"/></A><BR/>
86
87 <xsl:apply-templates select="OfferType"/>
88 <B>Offer Id: </B><xsl:text> </xsl:text><xsl:value-of select="@OfferId"/><BR/>
89 <xsl:apply-templates select="Property"/>
90 <xsl:apply-templates select="ObjectReference"/>
91 <hr/></td></tr>
92 </xsl:template>
93
94 <xsl:template match="OfferType">
95 <B>Offer Type: </B><xsl:text> </xsl:text>
96 <xsl:variable name="stname" select="@Name"/>
97 <A HREF="http://isel.cp.eng.chula.ac.th/servlet-bin/servlet/g42wsp.DescribeType?
98 TraderName={$tname}&amp;amp;TypeName={$stname}">
99 <xsl:value-of select="@Name"/></A><BR/>
100 </xsl:template>
101
102 <xsl:template match="Property">
103 <B>Property : </B><xsl:value-of select="@Name"/><xsl:text> :
104 </xsl:text><xsl:value-of select="@Value"/><BR/>
105 <xsl:apply-templates select="DynamicPropEval"/>
106 <xsl:apply-templates select="ExtraInfo"/>
107 </xsl:template>
108
109 <xsl:template match="DynamicPropEval">
110 <DD/><B>DynamicPropEval :</B> <xsl:value-of select="@ReturnType"/>
111 <xsl:text> : </xsl:text>
112 <DD/><xsl:value-of select="."/><BR/>
113 </xsl:template>
114
115 <xsl:template match="ExtraInfo">
116 <DD/><B>ExtraInfo:</B> <xsl:value-of select="@Type"/>
117 <xsl:text> : </xsl:text><xsl:value-of select="@Value"/><BR/>
118 </xsl:template>
```

```
119
120 <xsl:template match="ObjectReference">
121 <B>Object reference:</B> <xsl:value-of select="."/><BR/>
122 </xsl:template>
123
124 </xsl:stylesheet>
```

คำอธิบายเอกสารเอ็กซ์เอสแอลสำหรับการแปลงผลลัพธ์เป็นเอกสารเอชทีเอ็มแอลในรูปแบบย่อ

บรรทัดที่	คำอธิบาย
1-31	ลักษณะเดียวกับรูปแบบสมบูรณ์ข้างต้น
35-39	เป็นรายละเอียดของการนำเสนอข้อมูลในส่วนย่อย <TraderServiceType> โดยนำลักษณะประจำ Name มาแสดง และเลือกข้อมูลเฉพาะ <Property>
41-44	เป็นการเลือกคุณสมบัติของบริการ โดยนำข้อมูลลักษณะประจำ Type และค่าภายในป้ายระบุ <Property> มาแสดง
48-52	เป็นการเลือกข้อมูลส่วนต่อประสาน <Interface> ซึ่งเลือกเฉพาะ RepositoryId, Attribute และ Operation
54-60	เป็นการสร้างส่วนเชื่อมโยงให้สามารถเรียกโปรแกรมสำหรับการค้นหาบริการตามชนิดของบริการนี้ คือ PreTradeOffer โดยส่งค่าชื่อบริการเทรดเดอร์และชื่อชนิดของบริการเป็นพารามิเตอร์ของโปรแกรม
62-72	เป็นรายละเอียดของการนำเสนอข้อมูลในส่วนย่อย <RepositoryId>, <Attribute> และ <Operation> โดยการดึงเอาเฉพาะข้อมูลในลักษณะประจำมาแสดง
76-122	เป็นการนำเสนอข้อมูลของบริการภายใต้ป้ายระบุ <ServiceOfferDescription> ลักษณะเดียวกับรูปแบบสมบูรณ์ข้างต้น เนื่องจากรูปแบบของข้อมูลบริการเหมือนกันทั้งแบบสมบูรณ์และแบบย่อ

ภาคผนวก ค
ลำดับการทำงานของระบบและการเรียกใช้ส่วนต่าง ๆ ของบริการเอสดีเอส

ลำดับการเริ่มต้นบริการเอสดีเอสและบริการอื่นๆที่เกี่ยวข้อง

1. เริ่มโปรแกรม osagent ของวิสิโบรกเกอร์ ด้วยคำสั่ง

   osagent [-options]

   คำสั่ง osagent สามารถระบุพอร์ตด้วยการระบุค่าในตัวแปรสภาวะ (Environment Variable) OSAGENT_PORT หรือทางอาร์กิวเมนต์ของคำสั่ง เช่น osagent –p=14001 ได้ ทั้งนี้เพื่อให้สภาวะแวดล้อมการทำงานของระบบแยกจากกันได้ ซึ่งหากไม่มีการระบุ โปรแกรมจะใช้ค่าโดยปริยายคือ 14000 ในวิทยานิพนธ์นี้ใช้วิธีกำหนดพอร์ตสำหรับการใช้งานเทรดเดอร์มากกว่า 1 ตัวในเครื่องเดียวกัน ซึ่งเป็นการจำลองการทำงานของบริการเทรดเดอร์หลายๆ ตัวพร้อมกันได้
2. เริ่มโปรแกรมคลังส่วนต่อประสาน ด้วยคำสั่ง

   irep [-options] <irep name> [<idl storage file name>]

   ผู้ใช้สามารถใช้ทางเลือก -console เพื่อให้ทำงานในโหมดอักษรได้ และสามารถระบุชื่อคลังส่วนต่อประสานเพื่อใช้เป็นชื่ออ้างอิงในการเรียกใช้ และสามารถกำหนดชื่อไฟล์เพื่อเก็บข้อมูลส่วนต่อประสานแบบถาวรได้ได้
3. เริ่มบริการเทรดเดอร์ ด้วยคำสั่ง

   vbj –DORBservices=tagent –DORBagentPort=14001 jacorb.trading.TradingService –TS_Ref –d db

   อาร์กิวเมนต์ของการเริ่มบริการเทรดเดอร์มีดังนี้

   –DORBservices=tagent เป็นการระบุการใช้งานบริการเทรดเดอร์ร่วมกับเทรดเดอร์เอเจนท์ ดังที่กล่าวไว้ในบทที่ 4

   –DORBagentPort=14001 เป็นการระบุพอร์ตของ osagent (หากบริการเริ่มต้นโดยไม่ระบุพอร์ต จะได้ใช้ osagent ที่ใช้พอร์ต 14000)

   jacorb.trading.TradingService เป็นการเรียกเพคเกจของบริการเทรดเดอร์

   –TS_Ref เป็นการระบุชื่อไฟล์สำหรับเก็บข้อมูลอ้างถึงวัตถุของบริการเทรดเดอร์

–d db เป็นการระบุไดเร็คทอรีสำหรับเก็บข้อมูลบริการของบริการเทรดเดอร์

4. เริ่มบริการเอสดีเอส ด้วยคำสั่ง

vbj DiscoveryService SDS_Ref

การกำหนดอาร์กิวเมนต์ SDS_Ref เพื่อระบุเป็นชื่อไฟล์สำหรับเก็บข้อมูลอ้างถึงวัตถุของบริการเอสดีเอส

5. สำหรับครั้งแรกที่บริการเทรดเดอร์ติดต่อกับบริการเอสดีเอส ผู้ดูแลบริการเทรดเดอร์จำเป็นต้องลงทะเบียนกับบริการเอสดีเอส โดยเรียกใช้ตัวกระทำการ register_trader() จากศูนย์จัดเตรียมบริการ หลังจากนั้นก็นำข้อมูลชนิดของบริการและข้อมูลบริการที่มีอยู่ ส่งไปยังบริการเอสดีเอส ซึ่งผู้ดูแลบริการเทรดเดอร์สามารถทำงานในขั้นตอนดังกล่าวได้โดยการเรียกใช้โปรแกรมการลงทะเบียนดังนี้

vbj SDSregister

ภาคผนวก ง
ผลงานตีพิมพ์

1. การประชุมทางวิชาการวิทยาการและวิศวกรรมคอมพิวเตอร์แห่งชาติ ครั้งที่ 4 (The 4th National Computer Science and Engineering Conference (NCSEC 2000)) เมื่อวันที่ 16-17 พฤศจิกายน 2543 ในบทความเรื่อง An Architecture for a Service Discovery Service in CORBA

โดยผู้แต่งคือ Worawut Suphasanthitikul and Twittie Senivongse

An Architecture for a Service Discovery Service in CORBA

Worawut Suphasanthitikul[1] and Twittie Senivongse[2]
Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University
Pathumwan, Bangkok, Thailand 10330
Phone: (66-2) 2186991, Fax: (66-2) 2186955
E-mail: g42wsp@cp.eng.chula.ac.th[1], twittie.s@chula.ac.th[2]

Abstract: A trader facilitates the advertising and discovery of services in CORBA distributed environment. Service providers can publish their service types and offers of those types with the trader so as for clients can look for required functionality. At present, trader search is limited to only search for service offers while discovery of service types is not possible. This paper reports on a progress of our proposal for an architecture of a Service Discovery Service (SDS) which is based on the working of the trader but can support a wider variety of search including service type search. The search will be done on the XML version of service descriptions via XML query languages and search keywords with interfaces available for both CORBA and WWW users. The SDS adds to the current functionality of the trader with an enlarged search space by also cooperating with several traders and SDS instances.
Key words: Service Type, Trading Object Service, CORBA, XML

1. Introduction

Trading Object Service [1] has been proposed by the Object Management Group (OMG) to facilitate the advertising and discovery of services in CORBA (Common Object Request Broker) distributed environment [2]. Trader is an instance of the Trading Object Service that maintains information of available service types and service offers (i.e. instances of service types). The sequence of interaction in the trader is shown in Figure 1. Exporters publish service exports to the trader describing computational characteristic (e.g. interface signature) and non-computational properties. Such service descriptions are maintained by the Service Type Repository and Service Offer Directory. The trader processes service import requests by looking for matching service offers from its collection of service descriptions before returning the search results to the importers. Discovered service offers will then be contacted directly by the importers.

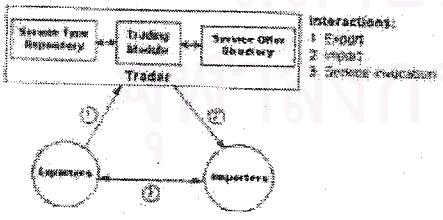


Figure 1: Interaction between exporters, trader and importers.

Several problems remain with the trader including the following.

- Importers can only trade for service offers; importers specify service type names and names and values of service properties as search criteria for the trader to select appropriate offers.
- Importers must have exact knowledge of service types and properties in order to specify search requests; no features of advanced search or keyword search as of WWW search engines are supported.
- The representation of service descriptions and the implementation of trader search are dependent on the data structures and the storage model used by particular trader vendors. This makes the utilisation of trader information by other environment (such as HTTP) difficult.

To solve these problems, we propose an architecture for a Service Discovery Service (SDS) in CORBA. The SDS will provide both search for service types and search for service offers based on service types, service properties, and interface definitions of services. Existing service descriptions within a trader will be transformed into an XML format [3] by using the C2X Transformer [4]. The SDS will maintain these XML service descriptions and perform search on them. We choose to do the search on the XML version of the service descriptions because we do not want to restrict our search strategy to a vendor-specific representation of the service descriptions, so we choose to perform

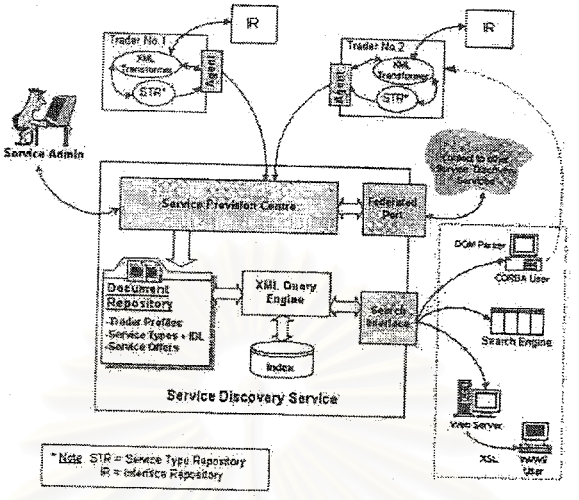


Figure 2: SDS architecture

search on the service descriptions on a more standardised format like XML. As XML information is semi-structured [5] with schema associated with the data, its self-describing characteristic can be useful for the search. The SDS will also provide a search interface for CORBA and WWW users to query by XML query languages or keywords, and this will make its search functionality available for not only CORBA environment but also the Internet community.

This paper is organised as follows. In Section 2, we mention related research on searching for information within distributed environments. We then describe our design of the SDS in Section 3 followed by the status of the prototype implementation in Section 4. Section 5 summarises our present work and discusses some of the future work.

2. Related Work

Several architectures have been designed and developed to discover several kinds of distributed resources. Agora [6] is a component search engine that can search for software components of some types on the Internet. For CORBA components, a CORBA client, acting as an Agora agent, perform the search in the CORBA Interface Repository (IR) [6] that maintains interface definitions of all CORBA components. We think that the search should be better done on the information from the trader because service descriptions of exported components within the trader also describe non-computational properties of the components apart from their interface definitions. WebTrader [7] is another architecture that provides an infrastructure for a general Web-based service market where users can export and import services in XML documents. However, it does not adopt any distributed system standard and so does not provide for all characteristics of CORBA service descriptions. Other research work includes [8] in which the search for software components is based on theorem proving of algebraic specifications of the components, something that could be more difficult for practical use. The work in [9] describes a form of trader federation in which service descriptions in several traders are pooled together to provide for larger search space but search is still restricted to search for service offers.

3. Service Discovery Services (SDS)

The architecture of the SDS is depicted in Figure 2. The SDS acts as a common service in CORBA that can gather service description information from several traders and do the service search.

Service descriptions comprise service type descriptions and service offer descriptions from the trader and interface definitions from the IR[1]. By using a Trader Agent that can retrieve service descriptions from a trader, the SDS can obtain service descriptions in an XML format through a C2X Transformer. The C2X Transformer implements extended operations in a trader and can transform normal CORBA service descriptions into

[1]A service type description defines a service type name and a list of properties of a service whereas a service offer description describes information of a particular object of a service according to its service type. An interface definition specifies computational interface signature of a service that supports it.

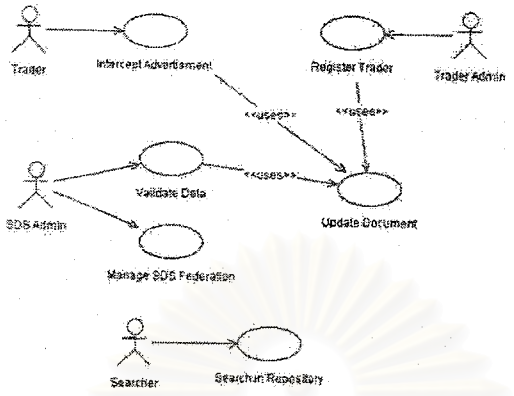


Figure 3: SDS use case model

XML documents according to the predefined Document Type Definitions (DTDs) for service types and service offers (see http://isel.cp.eng.chula.ac.th/~g42wsp/SDS/).

The SDS provides a powerful XML repository that accommodates search indexes for XML service descriptions. A search interface supports multiple search strategy and is available for CORBA clients and Web users. Several SDS instances can be federated so as to establish a wider search environment. The rest of this section describes parts and features that comprise the SDS.

3.1. Use Case Model of SDS

The use case model of SDS is presented in Figure 3.

Actors of the system are classified into four types:

1. **Trader** provides service descriptions to the SDS. In fact, service information from the trader can be dynamically intercepted when exporters advertise or modify their services.
2. **Trader Admin** is responsible for registering and managing the profile of a trader participating with the SDS.
3. **SDS Admin** is responsible for inspecting and correcting the information stored in the SDS document repository. The SDS admin also manages link information with other groups of traders through federated SDSes.
4. **Searcher** uses search abilities of the SDS search interface to search service descriptions in the SDS document repository.

Six use cases for the SDS have been identified:

1. **Intercept Advertisement** is used by a Trader for trapping service export and modification operations performed on it so that new or up-to-date service descriptions are recorded in the SDS document repository using the **Update Document** use case.
2. **Register Trader** enables a Trader Admin to register a trader profile to the SDS. It uses the **Update Document** use case to update the SDS document repository with the new trader profile.
3. **Validate Data** is used by an SDS Admin to review and validate information in the SDS document repository with help of the **Update Document** use case.
4. **Update Document** is used by previous use cases for updating their information within the SDS document repository.
5. **Manage SDS Federation** is used by an SDS Admin to establish and manage federation between SDSes.
6. **Search in Repository** lets a Searcher search on service descriptions within the SDS document repository.

3.2. SDS Components

According to the overview architecture in Figure 2 and the use case diagram, each SDS components will be examined.

3.2.1. Trader Agent

The trader agent, as a CORBA interceptor [10], is a part of the SDS that is bound to a trader. It comes to life with the trader to detect some exporter requests to the trader which can effectively modify service type and service offer information (i.e. add_type() and remove_type() for service types and export(), withdraw(), and modify() for service offers). Such interception will trigger a C2X Transformer to transform corresponding service descriptions into XML documents which will then be forwarded to the main SDS.

3.2.2. Service Provision Centre

The service provision centre interfaces with traders and trader admins to acquire necessary information for the work of the SDS. The information is stored as XML documents within the SDS document repository and includes service descriptions from trader agents as well as trader profiles from trader admins. XML service descriptions will be transformed into new XML documents that are suitable for storage in the document repository, and we employ an XSLT processor using several XSL stylesheet documents for this purpose [11]. Trader admins need to register profiles of the participating traders (e.g. trader admin contacts, object references of traders) before further connection between the SDS and traders can be established. The profile information is specified according to a predefined Trader Profile DTD (see http://isel.cp.eng.chula.ac.th/~g42wsp/SDS/).

The service provision centre also incorporates a query engine for its indexing and search capabilities.

3.2.3. Search Interface

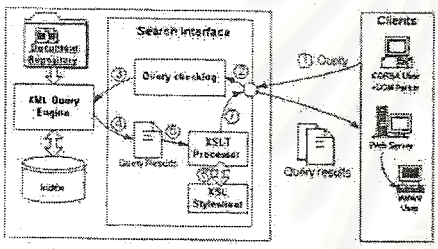


Figure 4: Search Interface

The search interface supports query operations on service descriptions within the SDS document repository. Figure 4 illustrates the architecture of the search interface. First, a CORBA client or Web user can query the SDS through the search interface using the query language supported by the SDS (1). Note that for Web users, Web client programs (e.g. JavaServlet, JSP and CGI) must be provided to mediate between HTTP and the CORBA system. The query will be parsed for syntax checking (2) and further processed by an XML query engine (3) with matched results returned as XML documents (4). An XSLT processor will transform the results (5) into appropriate formats, specified within the client's query, using supported XSL stylesheet documents (6). Finally, XML query results are returned to the client (7).

3.2.4. Federated Port

The federated port is designed to enlarge search space of the SDS. It allows one SDS to use the search interface of another. A query to an SDS can be propagated to other SDSes through the list of SDS links until the query is answered satisfactorily. Link information consists of necessary attributes for SDS federation such as link name, link target, and some behavior policies for controlling the propagation of queries. The function of the federated port is to manage link information such as add, remove, modify links etc.

4. Prototype Implementation

The SDS is being developed as a CORBA service mainly on Visibroker 3.4 for Java [10] with an exception on the trader that we use the implementation from JacORB [12]. The class model of the current implementation of the SDS is presented in Figure 5.

An SDS can be implemented to support several XML query languages by integrating corresponding query engines. Despite a number of XML query languages available, our current implementation chooses to support XQL [13] and XML-QL [14] as they have been submitted to W3C. XQL has been developed by the document processing community

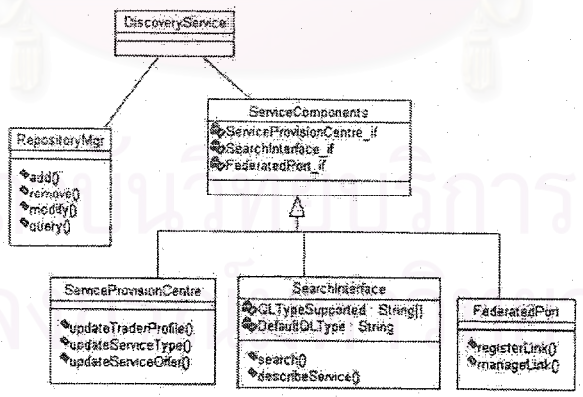


Figure 5: SDS class model

who is concerned with full text search and query of structured documents. We use the XQL query engine from Fatdog Software [15] to produce search indexes for XML documents and provide general XQL query facilities including keyword search. XML-QL is a query language from the database research community, designed for querying a large amount of data on the Web, and has clauses similar to those in SQL. We use the implementation of an XML-QL query engine from [16].

Since more than one query language can be supported, clients will specify in their queries what query language they are using. The search interface will then check to see whether the specified query language is supported by the current implementation of the SDS, and if not, an UnknownQLType exception will be raised. In this way, we can implement a multilingual SDS that can be extended to support more query languages in the future.

```
// local types
    typedef lstring Keyword;
    typedef lstring Result;
    typedef lstring QLType;
    typedef sequence<QLType> QLTypeSeq;
    enum SearchType {
        SERVICE_TYPES,
        SERVICE_OFFERS,
        ALL
    };
    struct SearchPolicy {
        unsigned long max_results;
        lstring mark_row_tag;
    };
    enum ResultFormat {
        MAXIMUM,
        MINIMUM,
        NAME_ONLY
    };
// attributes
readonly attribute QLTypeSeq
        qltype_supported;
readonly attribute QLType default_qltype;
// operation signatures
Result search (
    in Keyword keyw,
    in QLType ql_type,
    in SearchType search_type,
    in SearchPolicy search_policy,
    in ResultFormat return_format )
```

Figure 6: Interface definition for Search Interface

Figure 6 shows the detail of the search interface. Clients can query for service types or service offers and can select whether the search should be done either on service type descriptions only (including service properties and interface definitions), on service offer descriptions only, or on all descriptions. Some policy (e.g. maximum returned results) and result format (e.g. full description, minimal description, or name only) can be specified for returned results. To format the presentation of the results, our prototype currently provides some stylesheet documents that conform to the XSLT specification and correspond to the service type description DTD and the service offer description DTD in [4]. See http://isel.cp.eng.chula.ac.th/~g42wsp/SDS/ for details.

To support Web users, we have implemented Web clients in the form of JavaServlet and JSP. These programs act as a bridge from the Web to invoke CORBA interfaces. An example of a query via HTTP is shown in Figure 7.

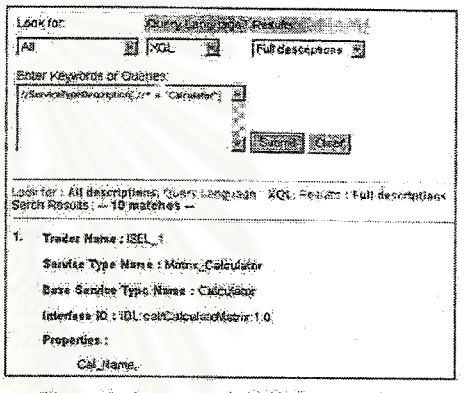


Figure 7: An example of WWW interface.

5. Conclusion and Future work

Our design of the SDS can augment the current search capability of CORBA traders. With the SDS, clients can query for available service types or service offers in various ways based on their types, properties, and interfaces, and can use several query languages. The search space is large as it can span several traders as well as several SDSes. The SDS is also easier to access from non-CORBA environment such as the Internet.

The development of the fully functioned SDS is in progress. At present, the capability of our SDS is dependent on the capability of the integrated XML query engines. In the future, we plan to integrate more advanced query engine, when available, to perform advanced search like WWW search engines. We are also interested in extending the SDS to support the discovery of a wide variety of distributed service components such as JavaBeans, DCOM, ActiveX etc. This extension will involve the mechanisms to acquire metadata of those components and to adapt such information to our common XML format.

Acknowledgement

This work is supported by the Development Grants for New Faculty/Researchers of Chulalongkorn University.

References

[1] Object Management Group, "Trading Object Service Specification", Revised Edition, March 1997.

[2] Object Management Group, "The Common Object Request Broker: Architecture and Specification", Revision 2.2, Feb. 1998.

[3] T. Bray, J. Paoli, and S. McQueen, "Extensible Markup Language (XML) 1.0 Specification", World Wide Web Consortium Recommendation, 10 Feb. 1998, http://www.w3.org/TR/REC-xml

[4] W. Nanekrangsan, W. Suphasanthitikul and T. Senivongse, "An Approach to Standardizing Distributed Service Descriptions Format using XML", In Proceedings of the 23rd Electrical Engineering Conference (EECON23), 2000.

[5] P. Buneman, "Semistructured Data", In Proceedings of the Sixteenth ACM SIGMOD Symposium on Principles of Database Systems (Tucson, Arizona), 1997, pp. 117-121

[6] R.C. Seacord, S.A. Hissam, and K.C. Wallnau, "Agora: A Search Engine for Software Components", Technical Report CMU/SEI-98-TR-011, August 1998.

[7] V. Vasudevan and T. Bannon, "WebTrader: Discovery and Programmed Access to Web-Based Services (Draft)". OBJS Techinical Report, 1999, http://www.objs.com/agility/tech-reports/9812-web-trader-paper/WebTraderPaper.html

[8] R. Tong and S. Haolin, "A Component Search Engine Model on Internet", Technology of Object-Oriented Languages,1997, In Proceedings of Tools'24, 1998, pp. 393-396.

[9] H. Tanaka, "Integrated environment for service-type repository management (IE-STREM)", In Proceedings of Global Convergence of Telecommunications and Distributed Object Computing (TINA 97), 1998, pp. 363 –371.

[10] Inprise Visibroker, http://www.inprise.com

[11] World Wide Web Consortium, "XSL Transformations (XSLT) Version 1.0", World Wide Web Consortium Recommendation, 16 November 1999, http://www.w3.org/TR/1999/REC-xslt-19991116

[12] JacORB – a free Java ORB, http://www.inf.fu-berlin.de/~brose/jacorb

[13] J. Robie, J. Lapp and D. Schach, "XML Query Language (XQL)", Position paper of W3C QL'98 - The Query Languages Workshop, 1998, http://www.w3.org/TandS/QL/QL98/pp/xql.html

[14] World Wide Web Consortium, "XML-QL: A Query Language for XML", Submission to the World Wide Web Consortium, 19 August 1998, http://www.w3.org/TR/1998/NOTE-xml-ql-19980819

[15] XML Query Engine, http://www.fatdog.com

[16] XML-QL: A Query Language for XML, http://www.research.att.com/sw/tools/xmlql

2. การประชุมวิชาการทางวิศวกรรมไฟฟ้าครั้งที่ 23 (The 23rd Electrical Engineering Conference (EECON-23)) เมื่อวันที่ 23-24 พฤศจิกายน 2543 ในบทความเรื่อง An Approach to Standardizing Distributed Service Descriptions Format using XML

โดยผู้แต่งคือ Wuttichai Nanekrangsan, Worawut Suphasanthitikul and Twittie Senivongse

An Approach to Standardizing Distributed Service Descriptions Format using XML

Wunicha Nanekrangsan, Worawut Suphasanthitikul and Twittie Senivongse
Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University
Pathumwan, Bangkok, Thailand 10330.
Phone: (66-2) 2186991 Fax: (66-2) 2186955
E-mail: {g41wnn, g42wsp}@cp.eng.chula.ac.th, twittie.s@chula.ac.th

Abstract

Distributed services advertised in a distributed system normally are restricted to the underlying distributed architecture. For example, services advertised in a directory service, called a trader, of the CORBA distributed architecture are accessible only within the CORBA platform. This paper proposes a way to increase accessibility of service information in CORBA environment so that such information is also available for access from other environment such as the Internet. We define XML Document Type Descriptions (DTDs) for CORBA service descriptions and use them to produce XML documents for those descriptions. We also design and implement an extension to a CORBA trader to accommodate the transformation of conventional CORBA service descriptions into XML. It is foreseen that this approach can provide a way to expose CORBA service descriptions to the Internet and facilitate discovery of CORBA components from the Web.

Keywords : Trading Object Service, CORBA, XML

1. Introduction

CORBA [1] is one of the major standard architectures for distributed object systems. It provides a Trading Object Service or a trader [2] as a directory service that maintains information of available service types and service offers within the system. It allows service exporters to describe service types and advertise service offers (i.e. service instances) according to those types and also enables service importers to discover offers of particular service types. At present, these service descriptions are accessible only within a CORBA environment. There has been an attempt to search for CORBA components via the Internet [3] but the search is restrictedly achieved via a CORBA agent as the search is performed only on the interface definitions of the services. As the interface definition is only one part of a service description in the trader, we see that it would be better if the search is conducted on the whole set of information from the trader while such information is also in a format convenient for Internet search. We aim to provide trader service descriptions in the XML format [4]. XML seems to be an appropriate choice as it is becoming a de facto standard for data interchange on the Web. Describing data in a semi-structured way [5], XML conveys semantic information that is useful for the search.

In fact, the idea of describing software components for the Web has already been addressed in many research projects. WebTrader [6] provides an infrastructure to handle Web-Based service market where users can export and import services via predefined XML documents format. However, it does not provide for all characteristics of CORBA service descriptions. Other work focuses on the XML templates for software component descriptions, e.g. OSD [7] and DSD [8], but these templates are aimed for deployment management of components and are too general for describing services in CORBA environment.

In Section 2, we present our ongoing work to define XML Document Type Descriptions (DTDs) which are the schema for XML documents describing service types and service offers. Part of the implementation of the CORBA to XML transformer that can generate XML service descriptions according to those DTDs is discussed in Section 3. Section 4 concludes our present work and discusses some of the future work.

2. CORBA to XML Service Descriptions

This section describes conventional CORBA service descriptions in a trader and how to represent them by our XML DTDs.

2.1. Service Description in Trader

The description of a CORBA service advertised in a trader consists of three types of information:

1. Interface definition describes the interface of a particular service, i.e. interface id, interface name, base interfaces from which this interface inherits, and a set of operations (methods) available. This information is stored in a CORBA Interface Repository (IR) [1] but can be obtained indirectly from the trader.
2. Service type is a template for describing service offers of that type. It specifies service name, service interface, base service types from which this service type inherits, and a set of properties. This information is trader-own and stored in the Service Type Repository module of the trader.

3. Service offer is a description of a particular instance of a service type including service type name, name-value pairs of service properties, and object reference used to locate the instance. This information is trader-own and stored in the Offer List module in the trader.

Since service descriptions in the trader are composed of type level and offer level information, we define two DTDs: service type description DTD describing the interface definition and service type information and service offer description DTD describing service offer information.

2.2. Service Type Description DTD

Figure 1 shows the service type description DTD. This DTD is composed of two main parts: the interface definition (starting from line 2) and the service type information (starting from line 49).

The interface definition describes the computational signature of the service comprising the interface id within the IR (line 3), base interfaces (line 6-12) represented as a structured graph adapted from [9], a list of its constants (line 13-19), attributes (line 20-27) and a list of its operation signatures (line 28-47).

The service type information consists of the interface id (line 50), service type name (line 51), base service types from which this type is derived (line 53), and a list of service properties (line 56-62) specified by name, type, and mode of each property.

```
1 <!ELEMENT ServiceTypeDescription (Interface?, TradeServiceType?) >
2 <!ELEMENT Interface (BaseInterface*, Constant*, Attribute*, Operation*) >
3 <!ATTLIST Interface          Id        CDATA  #REQUIRED
4                              Name      CDATA  #REQUIRED
5                              Version   CDATA  #REQUIRED >
6 <!ELEMENT BaseInterface (BaseInterface*, Link*) >
7 <!ELEMENT BaseInterface EMPTY >
8 <!ATTLIST BaseInterface      Id        CDATA  #REQUIRED
9                              Name      CDATA  #REQUIRED >
10 <!ELEMENT Link EMPTY >
11 <!ATTLIST Link              Source    CDATA  #REQUIRED
12                             Dest      CDATA  #REQUIRED >
13 <!ELEMENT Constant EMPTY>
14 <!ATTLIST Constant          Id        CDATA  #REQUIRED
15                             Name      CDATA  #REQUIRED
16                             Version   CDATA  #REQUIRED
17                             Type      CDATA  #REQUIRED
18                             Value     CDATA  #REQUIRED
19                             Derived   (YES | NO) "NO" >
20 <!ELEMENT Attribute EMPTY >
21 <!ATTLIST Attribute         Id        CDATA  #REQUIRED
22                             Name      CDATA  #REQUIRED
23                             Version   CDATA  #REQUIRED
24                             Type      CDATA  #REQUIRED
25                             Mode      (NORMAL | READONLY)
26                                       "NORMAL"
27                             Derived   (YES | NO) "NO" >
28 <!ELEMENT Operation (Parameter*, Exception*, Context*) >
29 <!ATTLIST Operation          Id        CDATA  #REQUIRED
30                             Name      CDATA  #REQUIRED
31                             Version   CDATA  #REQUIRED
32                             Type      CDATA  #REQUIRED
33                             Mode      (NORMAL | ONEWAY)
34                                       "NORMAL"
35                             Derived   (YES | NO) "NO" >
36 <!ELEMENT Parameter EMPTY >
37 <!ATTLIST Parameter          Name      CDATA  #REQUIRED
38                             Type      CDATA  #REQUIRED
39                             Mode      (IN | OUT | INOUT) "IN" >
40 <!ELEMENT Exception (Member*) >
41 <!ATTLIST Exception          Id        CDATA  #REQUIRED
42                             Name      CDATA  #REQUIRED
43                             Version   CDATA  #REQUIRED
44                             Derived   (YES | NO) "NO">
45 <!ELEMENT Member EMPTY >
46 <!ATTLIST Member             Name      CDATA  #REQUIRED
47                             Type      CDATA  #REQUIRED >
48 <!ELEMENT Context (#PCDATA) >
49 <!ELEMENT TradeServiceType (BaseServiceType*, Property*) >
50 <!ATTLIST TradeServiceType   Id        CDATA  #REQUIRED
51                             Name      CDATA  #REQUIRED
52                             Mutant    (YES | NO) "NO" >
53 <!ELEMENT BaseServiceType (BaseServiceType*, Link*) >
54 <!ELEMENT BaseServiceType EMPTY >
55 <!ATTLIST BaseServiceType    Name      CDATA  #REQUIRED >
56 <!ELEMENT Property EMPTY>
57 <!ATTLIST Property           Name      CDATA  #REQUIRED
58                             Type      CDATA  #REQUIRED
59                             Mode      (NORMAL | READONLY |
60                                       MANDATORY |
61                                       MANDATORY_READONLY)
62                                       "NORMAL" >
```

Figure 1: Service Type Description DTD

2.3. Service Offer Description DTD

The service offer description DTD constrains XML documents of service offers as shown in Figure 2. An offer is described by its service type (line 3), zero or more properties (line 4-11) specified as (property name, property value) pairs, and the reference to this offer instance (line 12).

3. CORBA to XML Transformer

In this section, we describe our design and some implementation details of the extension to the trader, called the CORBA to XML transformer or C2X transformer, that uses the DTDs above to generate XML documents for service descriptions.

```
1<!ELEMENT ServiceOfferDescription (OfferType, Property*,
2                                  ObjectReference) >
3<!ELEMENT OfferType  Name  CDATA    #REQUIRED >
4<!ELEMENT Property (DynamicPropEval, ExtraInfo)? >
5<!ATTLIST Property     Name  CDATA    #REQUIRED
6                       Value CDATA    #IMPLIED >
7<!ELEMENT DynamicPropEval (#PCDATA) >
8<!ATTLIST DynamicPropEval ReturnType CDATA #REQUIRED >
9<!ELEMENT ExtraInfo EMPTY >
10<!ATTLIST ExtraInfo  Type  CDATA    #REQUIRED
11                     Value CDATA    #REQUIRED >
12<!ELEMENT ObjectReference (#PCDATA) >
```

Figure 2: Service Offer Description DTD

3.1. Design of C2X Transformer

The C2X transformer is added to the architecture of the trader and is depicted by Figure 3. The C2X transformer provides operations to transform a service type description according to the service type description DTD by obtaining the service type information from the Service Type Repository and the associated interface definition from the IR. For conversion of a service offer, it uses the service offer description DTD to transform the corresponding service offer description in the Offer List module into XML offer documents. We design this extended trader in such a way that the addition of the C2X transformer has no impact on the conventional operations of the trader.

3.2. Implementation of C2X Transformer

Our C2X transformer uses the implementation of the Document Object Model (DOM)[10] called Java Project X [11] to build an XML instance from service information. The prototype has been developed on JacORB [12], which is an implementation of CORBA architecture and uses the implementation of the IR from VisiBroker for Java [13].

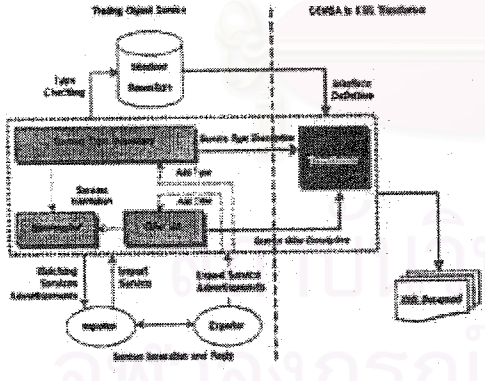


Figure 3: Trader with C2X Transformer

The prototype provides four operations to transform service descriptions into XML documents. The interface of the C2X transformer described in CORBA interface definition language is:

```
interface Corba2XmlTransformer {
   void transformAllServiceTypeDescription( );
   void transformServiceTypeDescription(
                              string serviceTypeName);
   void transformAllServiceOfferDescription( );
   void transformServiceOfferDescription(
                              string serviceTypeName);
};
```

The C2X transformer can be asked to transform all service type descriptions or all service offer descriptions maintained by the trader. We may also ask for the transformation of a particular service type description specifying the service type name, or for the transformation of all service offers of a particular service type specifying the service type name.

Client applications may simply call these operations through normal CORBA operation invocation. After each operation is called, the C2X transformer reads appropriate service descriptions and creates DOM document objects for such descriptions according to the two proposed DTDs. These document objects can be written out as an XML service description file using a variety of text encodings. An example of the transformation is shown in Figure 4.

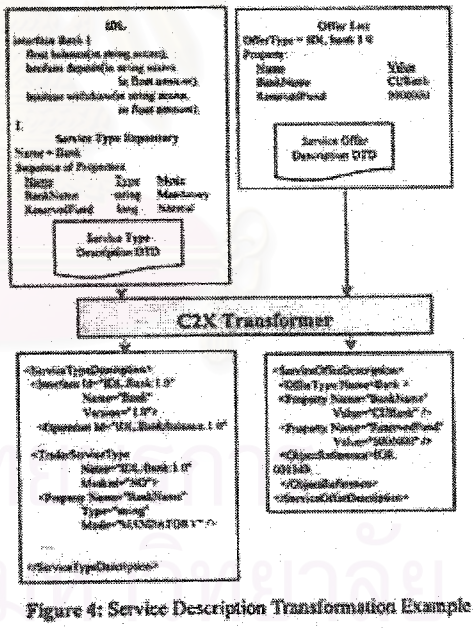


Figure 4: Service Description Transformation Example

4. Web-Based Client Prototype System

We have developed a client prototype to demonstrate the use of the C2X transformer from the Web environment using Java Servlet and JavaServer Pages (JSP) in conjunction with JacORB. It has been tested on Linux 2.2.13 using Apache 1.3.6 as a Web server and Tomcat 3.1 as a Servlet/JSP engine. We also tested successfully on Windows 98 using Sun's JavaServer Web Development Kit (JSWDK) 1.0.1 as a Web server and a Servlet/JSP engine. The Servlet mediates between the browser and a Java ORB client, which invokes the C2X transformer's operations, to provide a CORBA service for the browser. The client prototype uses an X2H transformer to render HTML from XML using the XSL and XSLT technology.

Figure 5 shows a Web-based client screen of the trader. Through this client prototype, a user can browse contents within the trader through a Web browser and can simply request a transformation into XML for a service type or service offer description without having to write a CORBA client program. The generated XML document can be viewed either in XML or HTML. The user may also use the XML description further with other XML-based applications.

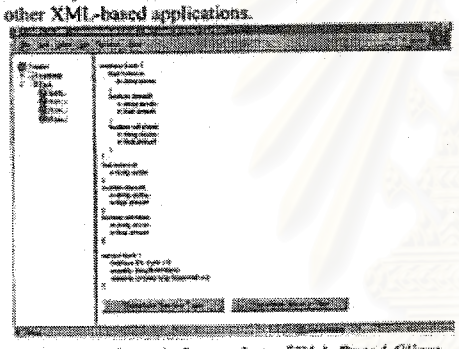


Figure 5: Sample Screenshot of Web-Based Client

5. Conclusion and Future Work

We have presented the design and implementation of the C2X transformer that uses the XML technology to describe service types and service offers in a trader in order to promote a Web-Based CORBA component descriptions. The two proposed DTDs cover most of the details of CORBA service type descriptions, service offer descriptions, and service interface definitions. The human-readable XML descriptions will facilitate the discovery of CORBA components from the Web.

We are still progressing on the counterfunction of the current C2X transformer prototype, enabling it to import into the trader the service types and service offers from XML service descriptions. We also plan to enhance the ability of the trader which is currently restricted to service offers lookup. The enhanced trader will have the ability of an XML search engine that can discover both service types and service offers from the XML descriptions generated by the C2X transformer. This trader is planned to be accessible from within CORBA environment and Web browsers.

References

[1] Object Management Group, "The Common Object Request Broker: Architecture and Specification", Revision 2.2, February 1998.

[2] Object Management Group, "Trading Object Service Specification", Revised Ed., March 1997.

[3] R.C. Seacord, S.A. Hissam, and K.C. Wallnau, "Agora: A Search Engine for Software Components", Technical Report CMU/SEI-98-TR-011, August 1998.

[4] T. Bray, J. Paoli, and S. McQueen, "Extensible Markup Language (XML) 1.0 Specification", World Wide Web Consortium Recommendation, 10 Feb. 1998, http://www.w3.org/TR/REC-xml

[5] P. Buneman, "Semistructured Data", In Proceeding of the Sixteenth ACM SIGMOD Symposium on Principles of Database Systems (Tucson, Arizona), 1997, pp. 117-121.

[6] V. Vasudevan and T. Bannon, "WebTrader: Discovery and Programmed Access to Web-Based Services (Draft)", OBJS Technical Report, 1999, http://www.objs.com/agility/tech-reports/9812-web-trader-paper/WebTraderPaper.html

[7] World Wide Web Consortium, "The Open Software Description Format (OSD) ", Submission to the World Wide Web Consortium, 13 August 1997, http://www.w3.org/TR/NOTE-OSD.html

[8] D. Heimbigner, R. Hall, A. Wolf, "Analyzing Configurations of Deployable Software Systems", SERL Technical Report CU-SERL-208-99, Software Engineering Research Laboratory, Department of Computer Science, University of Colorado, April 1999.

[9] O. Liechti, M. J. Sifer, T. Ichikawa, "Structured graph format: XML metadata for describing Web site structure", Computer Networks and ISDN Systems, 1998, pp. 11-21.

[10] V. Apparao, "Document Object Model (DOM) Level 1 Specification Version 1.0: World Wide Web Consortium Recommendation", 1998, http://www.w3.org/TR/REC-DOM-Level-1

[11] Java Project X, http://java.sun.com/xml

[12] JacORB – a free Java ORB, http://www.inf.fu-berlin.de/~brose/jacorb

[13] Inprise Visibroker, http://www.inprise.com

[14] A. Sharon et. al., Extensible Stylesheet Language (XSL), 2000, http://www.w3c.org/TR/xsl

[15] C. James, XSL Transformations (XSLT), 1999, http://www.w3c.org/TR/xslt

3. The 5th International Enterprise Distributed Object Computing Conference EDOC 2001 September 4-7, 2001 Seattle, Washington USA ในบทความเรื่อง An XML-Based Architecture for Service Discovery

โดยผู้แต่งคือ Twittie Senivongse and Worawut Suphasanthitikul (ยังอยู่ในระหว่างการพิจารณา)

An XML-Based Architecture for Service Discovery

Twittie Senivongse and Worawut Suphasanthiitikul
Department of Computer Engineering, Faculty of Engineering
Chulalongkorn University, Pathumwan, Bangkok 10330 Thailand
Phone: (662) 218-6996 Fax: (662) 218-6955
E-mail: twittie.s@chula.ac.th, g42wsp@cp.eng.chula.ac.th

Abstract

Limitations in the trading function (e.g. that offered by a CORBA Trading Object Service) are the inability to discover service types and the ability to trade only within a particular distributed environment (e.g. within a CORBA platform). We propose a Service Discovery Service (SDS) that can gather service descriptions from multiple traders and provide search for service types and interface definitions as well as service offers. Search is conducted on the XML version of service descriptions and query is possible via several XML query languages and search keywords. An SDS can be thought of as a super trader that is accessible by both CORBA and WWW users with a possibility of enlarged search space through the federation of multiple SDSes.

1. Introduction

Distributed architectures define a trading function for service discovery. In OMG CORBA [1], such service is called the Trading Object Service or trader [2]. A trader facilitates advertising of services by service providers (exporters) and querying for service offers by clients (importers). The interactions between exporters and importers are shown in Figure 1. After an exporter has registered its service type and service offer with the trader, an importer can query the trader for a service offer that matches its requirement, and after that, the communication between the exporter and importer is direct.

Within the trader, the Service Type Repository maintains descriptions of service types. A service type (e.g. Printer) consists of an interface type and zero or more named property types. An interface type specifies the computational signature of the service interface (e.g. operation Print()) whose definition can be found in the Interface Repository (IR). Property types refer to non-computational descriptions (e.g. printertype). The Service Offer Directory stores information of particular instances of service types. For example, information of two instances of the service type Printer may be stored. They both conform to the same interface definition but one has printertype = laser and the other has printertype = inkjet.

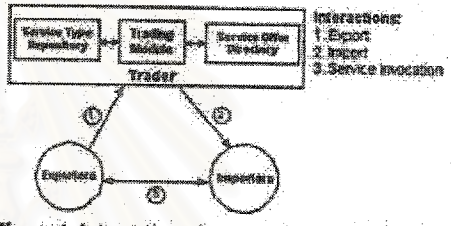


Figure 1. Interactions between Exporters, Trader, and Importers

Currently, there are still some limitations within the trader:

- Importers can only import for service offers. It is assumed that the knowledge of the service type and property types is known at import time, so service selection is based on evaluation of the importer's requirement against offers of the specified type. However, the importer may not know of the service type information and prefer to consider it before importing a service instance. One way the importer can do to find out about the services that might be of interest is to browse the whole content of the Service Type Repository but this is not convenient. At present, querying for service types directly is not possible.
- Service descriptions are platform-dependent. That is, they are packaged in a form that is understandable only within their own environment, e.g. service descriptions in CORBA can be obtained and understood only by those who learn about CORBA. Clients from other environment such as the Internet cannot use or have difficulties in using such information.

several registered traders. The Trader Agent associated with each trader will help keep the SDS up-to-date by detecting requests for addition or change of service descriptions that are sent to the trader. All service descriptions and updates are transformed by a CORBA/XML Transformer into XML documents before being sent to the Service Provision Centre and stored in its Document Repository component. Query on these documents is by query languages or keywords and the Search Interface is accessible by both CORBA users and WWW users. Federation of several SDSes can provide a view of an integrated SDS with wider search space.

Figure 3 illustrates the use case model of the SDS. There are four types of actors within the system:

1. Trader
Trader provides service information. The SDS can intercept service advertisement or update requests sent to the trader from exporters (e.f. the Trader Agent).
2. Trader Admin
Trader Admin is responsible for registering and managing profile information of a trader associated with the SDS.
3. SDS Admin
SDS Admin is responsible for verification and correction of the information stored within the SDS (e.f. Document Repository) and also manages the federation with other SDSes.
4. Searcher
Searcher is a user of the SDS who uses search facility of the SDS (e.f. Search Interface) to search for service descriptions.

The system consists of seven use cases:

1. Intercept Advertisement
Intercept Advertisement is a function for a Trader actor to detect updates of service information. Such information will internally be transformed to XML using the Transform to XML use case and then stored within the SDS using the Update Document use case.
2. Transform to XML
Transform to XML is a function used by the Intercept Advertisement use case to transform a service description into XML format.
3. Register Trader
Register Trader is a function for a Trader Admin actor to register the trader profile to the SDS and to store the profile using the Update Document use case.
4. Review Data
Review Data is a function for an SDS Admin actor to review and manage the information stored in the SDS using the Update Document use case.
5. Update Document
Update Document is an internal use case for updating information within the SDS (e.f. Document Repository)
6. Manage SDS Federation
Manage SDS Federation is a function of the SDS used by an SDS Admin actor to manage the federation of the SDSes.
7. Search in Repository
Search in Repository is a function of the SDS invoked by a Searcher actor to search for service descriptions.

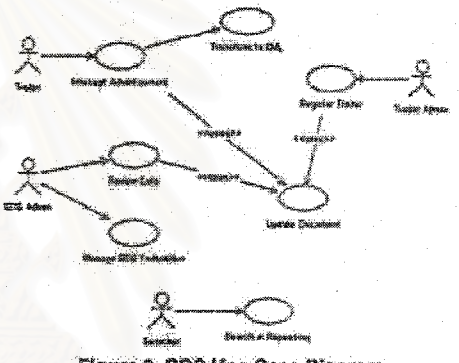


Figure 3. SDS Use Case Diagram

From the overview and use case model of the SDS, we can now discuss each part in more details.

3.1. CORBA/XML Transformer

Traders are sources of service information for the SDS. Since the SDS operates on XML service descriptions, it makes use of the result of our previous work on the provision for XML service descriptions by a CORBA/XML Transformer [5].

We define a CORBA/XML Transformer as an additional module to existing trading components (Figure 4). It interacts with three repositories, i.e. the Interface Repository to obtain interface definitions, the Service Type Repository to acquire service type descriptions, and the Offer List for service offer descriptions. With these descriptions, it then generates XML documents according to two predefined Document Type Definitions (DTDs), namely service type DTD and service offer DTD (Table 1-2). The service type DTD comprises the schemas for a

service type and its interface definition, while the service offer DTD defines the structure for a service offer description.

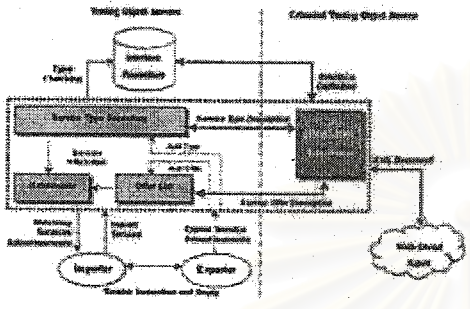


Figure 4. Trader with CORBA/XML Transformer

The interface definition of the CORBA/XML transformer describes several operations such as transform_type() for transformation of a particular service type description, transform_offer() for transformation of offer descriptions of a service type, and transform_offer_id() for transformation of a particular service offer etc. A CORBA/XML Transformer works in conjunction with a Trader Agent.

3.2. Trader Agent

Service descriptions within the SDS can be thought of as the XML version of those in the associated traders. A Trader Agent is an agent that tries to keep these copies consistent. It is a CORBA interceptor [10] that is bound to a trader and is able to detect addition or change of service information by trapping requests sent to the trader. The trapped requests are:

- add_type() and remove_type() which are sent to the Service Type Repository to add or remove a service type description.
- export(), withdraw(), and modify() which are sent to the Register module of the trader to add, remove, or change a service offer description.

If the request succeeds, the Trader Agent will call a CORBA/XML Transformer to transform the service description in question into XML, if necessary. Then, it will invoke the SDS Service Provision Centre to update the Document Repository. Figure 5 shows a sequence diagram of the Trader Agent when add_type() is detected. The new service type will be transformed and added to the Document Repository.

3.3. Service Provision Centre

The Service Provision Centre provides operations to add, remove, and modify information within the Document Repository and hence is called when the information needs to be updated. There are three types of such information, all in XML format:

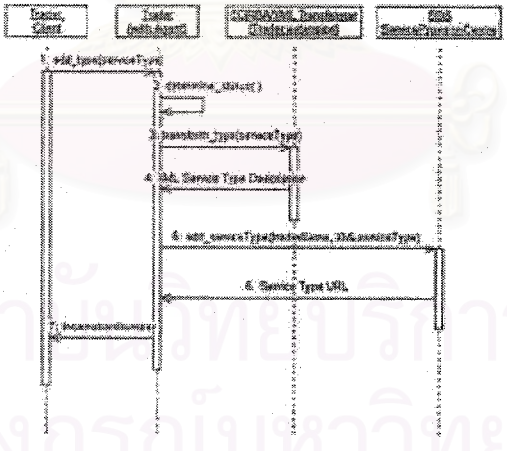


Figure 5. Sequence Diagram of Trader Agent when a New Service Type is Added

1. *Trader Profile* is the information about each associated trader, registered by a trader admin at the time the association is first established. The Trader Profile DTD (Table 3) describes the name and object reference of the trader together with other trader admin information.

2. *Service type description* comprises details of a service type and its interface definition as constrained by the service type DTD previously shown in Table 1.

3. *Service offer description* is the information of a particular service offer as defined by the service offer DTD in Table 2 above.

Table 1. Service Type DTD

```
<!ELEMENT ServiceTypeDescription (Interface?, TraderServiceType, TraderName*)>
<!ELEMENT Interface (BaseInterfaces?, Constant*, Attribute*, Operation*)>
<!ATTLIST Interface  Id   CDATA #REQUIRED Name  CDATA #REQUIRED
        Version CDATA #REQUIRED>
<!ELEMENT BaseInterfaces (BaseInterface*, Link*)>
<!ELEMENT BaseInterface EMPTY>
<!ATTLIST BaseInterface Id  CDATA #REQUIRED Name CDATA #REQUIRED>
<!ELEMENT Link EMPTY>
<!ATTLIST Link Source CDATA #REQUIRED Dest  CDATA #REQUIRED>
<!ELEMENT Constant EMPTY>
<!ATTLIST Constant Id CDATA #REQUIRED  Name  CDATA #REQUIRED
        Version CDATA #REQUIRED Type   CDATA #REQUIRED
        Value  CDATA #REQUIRED  Derived (YES | NO) "NO">
<!ELEMENT Attribute EMPTY>
<!ATTLIST Attribute Id  CDATA #REQUIRED   Name  CDATA #REQUIRED
        Version CDATA #REQUIRED  Type   CDATA #REQUIRED
        Mode   (NORMAL | READONLY) "NORMAL"
        Derived (YES | NO) "NO">
<!ELEMENT Operation (Parameter*, Exception*, Context*)>
<!ATTLIST Operation  Id  CDATA #REQUIRED  Name  CDATA #REQUIRED
        Version CDATA #REQUIRED  Type   CDATA #REQUIRED
        Mode   (NORMAL | ONEWAY) "NORMAL"
        Derived (YES | NO) "NO">
<!ELEMENT Parameter EMPTY>
<!ATTLIST Parameter Name  CDATA #REQUIRED  Type   CDATA #REQUIRED
        Mode   (IN | OUT | INOUT) "IN">
<!ELEMENT Exception (Member)*>
<!ATTLIST Exception  Id   CDATA #REQUIRED   Name  CDATA #REQUIRED
        Version CDATA #REQUIRED  Derived (YES | NO) "NO">
<!ELEMENT Member EMPTY>
<!ATTLIST Member Name  CDATA #REQUIRED  Type  CDATA #REQUIRED>
<!ELEMENT Context (#PCDATA)>
<!ELEMENT TraderServiceType (BaseServiceTypes?, Property*)>
<!ATTLIST TraderServiceType Id CDATA #REQUIRED Name CDATA REQUIRED
        Masked (YES | NO) "NO">
<!ELEMENT BaseServiceTypes (BaseServiceType*, Link*)>
<!ELEMENT BaseServiceType EMPTY>
<!ATTLIST BaseServiceType Name CDATA #REQUIRED>
<!ELEMENT Property EMPTY>
<!ATTLIST Property Name CDATA #REQUIRED  Type CDATA #REQUIRED
        Mode (NORMAL|READONLY|MANDATORY|
              MANDATORY_READONLY) "NORMAL"
        Derived (YES | NO) "NO">
<!ELEMENT TraderName (#PCDATA)>
```

Table 2. Service Offer DTD

```
1. <!ELEMENT ServiceOfferDescription (OfferType, Property*, ObjectReference)>
2. <!ATTLIST ServiceOfferDescription TraderName CDATA #REQUIRED OfferId CDATA #REQUIRED>
3. <!ELEMENT OfferType EMPTY>
4. <!ATTLIST OfferType Name CDATA #REQUIRED>
5. <!ELEMENT Property (DynamicPropEval, ExtraInfo)?>
6. <!ATTLIST Property Name CDATA #REQUIRED  Value CDATA #IMPLIED>
7. <!ELEMENT DynamicPropEval (#PCDATA)>
8. <!ATTLIST DynamicPropEval ReturnType CDATA #REQUIRED>
9. <!ELEMENT ExtraInfo EMPTY>
10. <!ATTLIST ExtraInfo Type  CDATA #REQUIRED  Value CDATA #REQUIRED>
11. <!ELEMENT ObjectReference (#PCDATA)>
```

Table 3. Trader Profile DTD

```
1. <!ELEMENT TraderProfile (ObjectRef+, Location?, AdminProfile+, OtherDetail?)>
2. <!ATTLIST TraderProfile TraderName CDATA #REQUIRED>
3. <!ELEMENT ObjectRef (#PCDATA)>
4. <!ELEMENT Location (#PCDATA)>
5. <!ELEMENT AdminProfile (Address*, Email+, OtherDetail?)>
6. <!ATTLIST AdminProfile AdminName CDATA #REQUIRED>
7. <!ELEMENT Address (#PCDATA)>
8. <!ELEMENT Email (#PCDATA)>
9. <!ELEMENT OtherDetail (#PCDATA)>
```

The Service Provision Centre is called when there is change that involves a trader profile, service type description, or service offer description. The activity of the Service Provision Centre is shown in Figure 6. It parses the input XML document to validate if it conforms to the corresponding DTD and checks to see if the input document involves change in a document that already exists in the SDS, by invoking the Search Interface (1). If so, the input update is applied; otherwise it is new information and then added (2). The Repository Manager is connected to the Document Repository, which is the physical storage of the documents, to update the information (3). The SDS Admin is able to manage (e.g. view, change, remove) these documents if required.

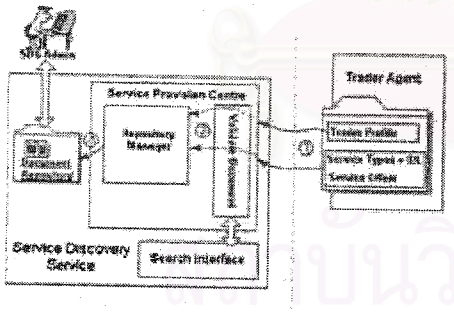


Figure 6. Activity of Service Provision Centre

3.4. Search Interface

The Search Interface provides searching for service descriptions in the Document Repository. This searching function is flexible since the Search Interface is designed to support multiple query languages. SDS clients can specify the query language they use at query time. Figure 7 shows the activity of the Search Interface. A client who may be a CORBA or Web user invokes the Search Interface specifying search criteria (1). After that, the query language and syntax are checked (2) before passing the query to its query engine (3). Note that, if the client's query language is not supported directly by any query engine, the Search Interface can be made responsible for mapping to a supported language. Search results in XML format (4) are sent to an XSLT processor [11] (5) so that they are transformed to some specific markup format for output (6). Finally, the output results are returned to the client (7).

Features of the Search Interface are described below:

1. Support for two groups of clients
   - CORBA clients can perform search on the SDS in the same manner as invoking other CORBA common services, by writing CORBA client programs to call the Search Interface. They can connect with the SDS by using resolve_initial_reference() method or directly binding with its known IOR.
   - Web clients who are human users or programs that use the HTTP connection can view the SDS as a search engine through the Search Interface.

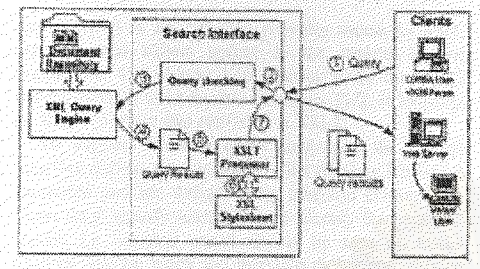


Figure 7. Search Interface

2. Search methodology
- Users with knowledge of the structure or DTD of the documents may construct a query using a supported XML query language.
- Users without knowledge of the structure or DTD of the documents can use keyword search as with normal search engines. The Search Interface has a feature to construct an XML query language from input keywords.

3. Result format
- With Maximum format, results contain details of all elements of service type or service offer descriptions.
- With Minimum format, results contain details of only some elements in the DTDs. An example of a transformed result in Minimum format is shown in Figure 8.

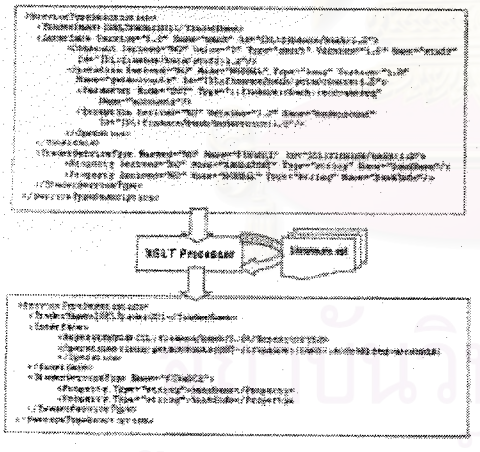


Figure 8. Example of Minimum Result Format Transformation

3.5. Federated Port

The Federated Port provides links from one SDS to other SDSes in order to enlarge search space. That is, one SDS is allowed to further invoke the Search Interface of the others. The federation mechanism is similar to the federation of Trading Service or Naming Service. Link information consists of attributes such as link name, link target and some behavioural policies that control propagation of queries. A query can be forwarded to the group of linked SDSes and the combination of their results is returned to the user. The Federated Port interface provides operations to manage link information such as register_sds(), remove_sds(), update_sds() etc.

4. Prototype Implementation

The SDS has been developed as a common service in CORBA using VisiBroker for Java [10]. We use JacORB implementation of a trader [12] with a Trader Agent implemented as a VisiBroker server interceptor.

The SDS class model is shown in Figure 9. The ServiceComponents class contains the references to all SDS interfaces and is derived by other SDS component classes. The RepositoryMgr class is a storage connector whose instance is used by ServiceProvisionCentre to directly store and update documents in a physical repository. SearchInterface is composed of one or more SearchInstances of query engines that the SDS supports.

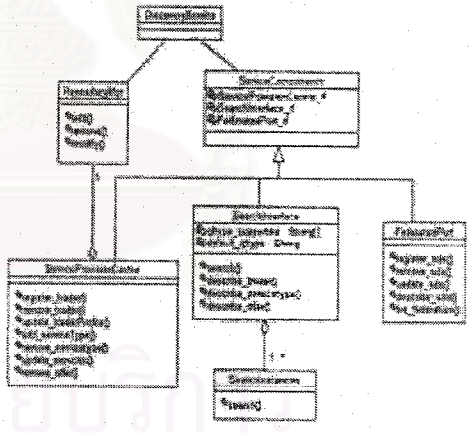


Figure 9. SDS Class Model

For simple manipulation (add, remove, or replace) of XML documents, the Document Object Model (DOM) is

used [13]. It is an API that provides access to XML elements and allows update to the content of the documents. The Oracle XML Parser v2 is the DOM implementation used for this prototype [14]. It includes an XSL Processor that conforms to XSL1.

To support multiple query languages, the Search Interface defines two attributes: a list of query languages supported and the default query language (Figure 10). Users must specify search type to identify the scope of search, i.e. whether search should be done on trader profiles, service types and interface definitions, service offers, or all types of documents. The ResultFormat variable can be either Maximum or Minimum format. If the "isURLresult" variable is set to "true", search results will be written to a file and its URL is returned to the user. The Search Interface also provides operations to describe each service type description, service offer description, and trader profile.

```
// local type
        typedef string Keyword;
        typedef string Result;
        typedef boolean isURLresult;
        typedef string QLType;
        typedef sequence<QLType> QLTypeSeq;
        enum SearchType {
          TRADERS,
          SERVICE_TYPES,
          SERVICE_OFFERS,
          ALL
        };
        enum ResultFormat {
          MAXIMUM,
          MINIMUM,
          NO_TRANSFORM
        };
// attribute
readonly attribute QLTypeSeq qltype_supported;
readonly attribute QLType default_qltype;
// operation signatures
Result search (
        in Keyword keyw,
        in QLType ql_type,
        in SearchType search_type,
        in isURLresult url_result,
        in ResultFormat result_format
)
ServiceTypeInfo describe_service_type (
        in TraderName trader_name,
        in ServiceTypeName st_name
)
```

Figure 10. Part of the Definition of Search Interface

We have considered several XML query languages for our prototype to support. At present, the XML Query Working Group has published a working draft document on XML Query Requirements [15] that specifies goals for the W3C XML Query data model and query language. As a result, XQuery [16] is designed to meet that requirement. XQuery borrows features from several XML query languages such as XQL [17], XML-QL [18] etc. However, it is very new and still has no supporting implementation.

By the time we started developing the prototype, the adoption process for a standard XML query language was still ongoing. We decided to adopt XQL here because it has been proposed to W3C by the document processing community and it is powerful with full text search and query of structured documents. There are also many XQL query engines around and we use the implementation of GMD-IPSI XQL Engine version 1.0.2 [19] that supports most of the functionality of XQL as well as persistence storage of XML documents through persistent DOM. We use its customisable method feature to enhance the query engine with functions (e.g. sum, minus, multiply, and divide) that will be used to evaluate queries. This enables the SDS to exhibit the characteristics of a normal trader where search criteria are defined using the trader's constraint language [2] and query evaluation is evaluation of expressions. Currently, our prototype still supports only XQL query engine.

We use Java Servlet and JavaServer Page (JSP) to maintain the WWW interface for the SDS. The Servlet program directly invokes the Search() operation in the Search Interface and processes according to specified parameters and keywords. The results are forwarded to an appropriate JSP program for display.

Figure 11 shows the Minimum results of a query for all service types with a property named process_time. The results may contain hyperlinks to other part of the information, e.g. the profiles of the traders that maintain the listed services or related service types of the queried offers. Users can query for service types, service offers, or trader profiles, and can specify search criteria similar to when importing from a trader or can list them using keywords. The SDS prototype provides search tips for users to conveniently and effectively use XQL or keywords, and takes care of the mapping from keyword search to XQL queries.

We contrast the query features of the SDS XQL prototype and those of a normal trader in Table 4. The features consist of search for service offers, search for service types, and full text search:

1. Search for service offers. The SDS aims to maintain the functionality of a normal trader by making it possible to trade for service offers in the same way importers can do when trading with a trader. Section 1 of

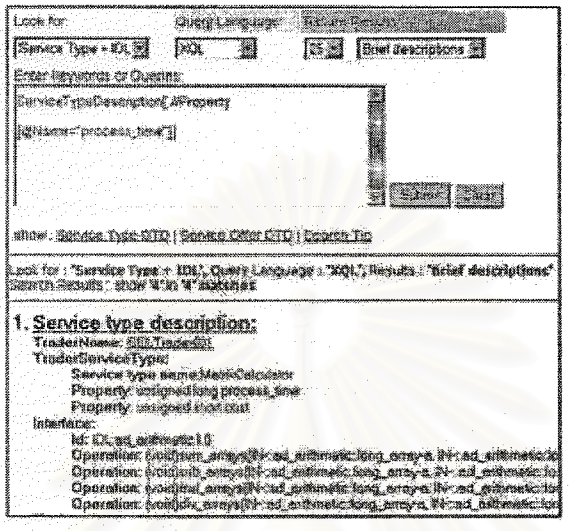


Figure 11. Example of Web Interface

Table 4 shows the XQL constructs that are equivalent to the OMG constraint language constructs used to import service offers. However, there are few kinds of query that can be issued to the trader but cannot be to the prototype due to the limitation of the CORBA/XML Transformer that we use. The limitation is that the Transformer supports only transformation of primitive types; there is no support for user-defined types yet. This means the Transformer cannot give detailed descriptions of user-defined types in XML service descriptions that it produces. This limitation makes the SDS that operates on such XML documents unable to conduct search based on details of user-defined types or to search for an element in a sequence (sequence may be seen as a user-defined type). For example, if a service property Addr is of type Address (i.e. a user-defined structure of Number, Street, and City), only the property name Addr will appear in corresponding XML service descriptions without details of Addr or property values. Therefore, we cannot issue, for instance, a query for an offer with Addr.City = 'Seattle' because the documents do not contain enough information. However, if the Transformer is improved in this respect, this limitation will be overcome.

2. Search for service types. Section 2 of Table 4 shows this advantage that the SDS has over the trader. With the SDS, we can look for service types including interface definitions by specifying required characteristics as keywords. The examples show a query for a service type with a required base type name and without a particular property, and a query for a required operation and parameter. This kind of search is not supported by the trader. To get the details of service types and interface definitions from the trader, knowledge of service type names is assumed. That is, one has to invoke provided operations to browse the trader in order to obtain a specific service type name of interest before using that name to query for its description.

3. Full text search. Section 3 of Table 4 shows the features of full text search over the XML documents within the SDS. Keywords can be used to search on any part of service descriptions, i.e. service type description and interface definition only, service offer description only, or all service description. We can direct the search by indicating ignored or exact keywords. With a normal trader, text search applies only on property values when service offers are evaluated against queries.

5. Conclusions

This paper presents an architecture of the SDS that enhances search features of a CORBA trader. Clients can query for information about service types, interface definitions, and service offers from several trader domains. Search can be done by query language selection and by both CORBA and non-CORBA clients such as Internet users.

Table 4. Query Features Comparison between OMG Trading Object Service and SDS XQL

OMG Trading Object Service	SDS (XQL)
1. Service Offer Description	
Comparative function: == (equality) != (inequality) > >= < <= ~ (substring match) in (element in sequence)	 eq , = ne , != gt , > ge , >= lt , < le , <= $contains$ and $icontains$ (GMD-IPSI feature) Not applicable
Boolean connectives: And Or Not	 and or not
Property existence: Exist	 Search for element tag of property
Mathematical operations: +,-,*,/	 Customised function (GMD-IPSI feature)
Grouping operation: (,)	 ( )
Identifiers used in the query are case sensitive.	Identifiers used in the query are case sensitive. For case insensitive search, use: ieq ine ilt ile igt ige (GMD-IPSI feature)
2. Service Type Description and Interface Definition	
Clients can only invoke operations from Service Type Repository. • list_type( ) - to obtain service type names. • describe_type(ServiceTypeName) - to obtain the detail of the specified service type name. • fully_describe_type(ServiceTypeName) - similar to describe_type() with all properties inherited from its super types returned.	Clients can search on XML service type descriptions including interface definitions. • Search for service type element tag, e.g. ServiceTypeDescription[TraderServiceType [.//BaseServiceType[@Name="Calculator"]] and not (.//Property[@Name="Cost"])] • Search for service type with particular operation, e.g. ServiceTypeDescription[.//Operation[@Name= "deposit"]/Parameter/@Type ="double"]
3. Full text search	
Text search on property values of service offers only.	For full text search, the SDS adopts basic operations from normal search engines. • Scope for keyword search can be specified, i.e. service type and interface definition, service offer, or all document. All the words in the query will be searched within the scope. • To exclude a word from search result, use a minus sign in front of keyword, e.g. print -laser • For exact word, use double or single quotation mark , e.g. print "HP"

Our design and prototype still assume that all traders registered with an SDS are CORBA compliant, and all CORBA/XML Transformers and the SDS adopt the same service type and service offer DTDs. In fact, it is possible that several types of traders, using different DTDs to describe their service descriptions, can be connected to the same SDS. Therefore, the SDS will have to support mapping between the schemas used by those traders and the ones it uses. This can be done using the XSLT to select and filter information within the documents from those traders and transform them into XML documents that the SDS understands.

In the future, we plan to integrate more advanced query engine such as a WWW search engine and also support the XQuery language that will soon become a recommendation of W3C. We are also interested in extending the SDS to support discovery of other distributed components such as JavaBeans, DCOM, ActiveX etc.

Acknowledgement

This work is supported by the Thailand-Japan Technology Transfer Project (TJTTP-OECF) and the Development Grants for New Faculty/Researchers of Chulalongkorn University.

References

[1] Object Management Group, "The Common Object Request Broker: Architecture and Specification", Revision 2.2, February 1998.

[2] Object Management Group, "Trading Object Service Specification", Revised Edition, March 1997.

[3] T. Bray, J. Paoli, and S. McQueen, "Extensible Markup Language (XML) 1.0 Specification", *W3C Recommendation*, 10 February 1998, http://www.w3.org/TR/REC-xml

[4] P. Buneman, "Semistructured Data", *In Proceedings of the Sixteenth ACM SIGMOD Symposium on Principles of Database Systems*, Tucson, Arizona, 1997, pp. 117-121.

[5] W. Nanckrangsan, "A Transformation of Service Descriptions between the CORBA Trader Format and XML", *Master's Thesis*, Department of Computer Engineering, Graduate School, Chulalongkorn University, Thailand, 2001.

[6] R.C. Seacord, S.A. Hissam, and K.C. Wallnau, "Agora: A Search Engine for Software Components", *Technical Report CMU/SEI-98-TR-011*, August 1998.

[7] V. Vasudevan and T. Bannon, "WebTrader: Discovery and Programmed Access to Web-Based Services (Draft)", *OBJS Technical Report*, 1999, http://www.objs.com/agility/tech-reports/9812-web-trader-paper/WebTraderPaper.html

[8] R. Tong and S. Haolin, "A Component Search Engine Model on Internet", Technology of Object-Oriented Languages.1997, *In Proceedings of Tools'24*, 1998, pp. 393-396.

[9] H. Tanaka, "Integrated environment for service-type repository management (IE-STREM)", *In Proceedings of Global Convergence of Telecommunications and Distributed Object Computing (TINA 97)*, 1998, pp. 363 – 371.

[10] Inprise VisiBroker, http://www.inprise.com

[11] World Wide Web Consortium, "XSL Transformations (XSLT) Version 1.0",*W3C Recommendation*, 16 November 1999, http://www.w3.org/TR/1999/ REC-xslt-19991116

[12] JacORB — a free Java ORB, http://www.inf.fu-berlin.de/~brose/jacorb

[13] V. Apparao, "Document Object Model (DOM) Level 1 Specification Version 1.0", *W3C Recommendation*, 1998, http://www.w3.org/TR/REC-DOM-Level-1

[14] Oracle Parser v2, http://technet.oracle.com

[15] D. Chamberlin, P. Fankhauser, M. Marchiori and J. Robie, "XML Query Requirements", *W3C Working Draft*, 15 February 2001, http://www.w3.org/TR/xmlquery-req

[16] World Wide Web Consortium, "XQuery: A Query Language for XML", *W3C Working Draft*, 15 February 2001, http://www.w3.org/TR/xquery

[17] J. Robie, J. Lapp and D. Schach, "XML Query Language (XQL)", *Position paper of W3C QL'98 - The Query Languages Workshop*, 1998, http://www.w3.org/TandS/QL/QL98/pp/xql.html

[18] World Wide Web Consortium, "XML-QL: A Query Language for XML", *Submission to the W3C*, 19 August 1998, http://www.w3.org/TR/1998/NOTE-xml-ql-19980819

[19] GMD-IPSI XQL Engine, http://xml.darmstadt.gmd.de/xql

ประวัติผู้เขียนวิทยานิพนธ์

นายวรวุฒิ ศุภสัณฐิติกุล เกิดเมื่อวันที่ 13 กันยายน 2518 ที่จังหวัดพัทลุง สำเร็จการศึกษาหลักสูตรวิศวกรรมศาสตรบัณฑิต (วศ.บ.) สาขาวิศวกรรมคอมพิวเตอร์ จากภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยสงขลานครินทร์ เมื่อปีการศึกษา 2540 และเข้าศึกษาต่อหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต (วศ.ม.) สาขาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2542