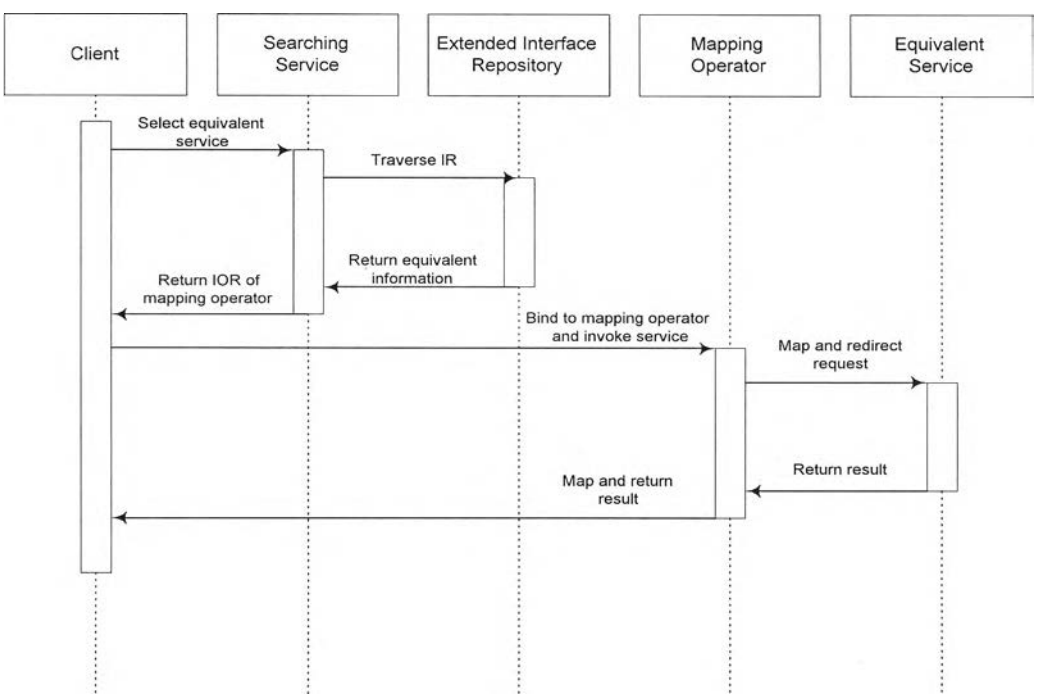
บทที่ 3
การออกแบบและพัฒนาโปรแกรมพรีโพรเซสเซอร์

จากหัวข้อ 1.2 แนวคิดในการทำวิจัย ซึ่งได้แบ่งการพัฒนากลไกส่วนขยายให้คอร์บารองรับการแทนที่กันของบริการที่มีความสัมพันธ์แบบเท่าเทียมกันในงานวิจัยนี้ออกเป็น 2 ส่วน ในบทนี้จะขอกล่าวถึงรายละเอียดในส่วนที่เกี่ยวข้องกับการออกแบบและพัฒนาโปรแกรมพรีโพรเซสเซอร์ (Preprocessor) ซึ่งเป็นกลไกส่วนขยายหนึ่งสำหรับใช้ในการแทรกคำสั่งการทำงานลงในโปรแกรมของผู้รับบริการ คำสั่งการทำงานที่ถูกแทรกเข้าไปนี้จะช่วยสร้างกลไกการค้นหาบริการที่สามารถทำงานแทนที่จากคลังจัดเก็บส่วนต่อประสานของงานวิจัย [4] เมื่อโปรแกรมของผู้รับบริการไม่สามารถเรียกใช้งานบริการที่ต้องการด้วยคำสั่ง bind ได้ตามปกติ

3.1 การทำงานของโปรแกรมพรีโพรเซสเซอร์

งานวิจัยนี้มีเป้าหมายที่จะพัฒนาโปรแกรมพรีโพรเซสเซอร์สำหรับใช้ในการวิเคราะห์และแทรกคำสั่งการทำงานที่เหมาะสมลงในโปรแกรมของผู้รับบริการเพื่อสร้างกลไกการแทนที่กันของบริการในระดับอินสแตนซ์ให้เป็นไปตามซีเควนซ์ไดอะแกรม (Sequence Diagram) ในรูปที่ 3.1



รูปที่ 3.1 ซีเควนซ์ไดอะแกรมการทำงานของคำสั่งที่แทรกโดยโปรแกรมพรีโพรเซสเซอร์

งานวิจัยนี้ได้จำกัดขอบเขตของภาษาคอมพิวเตอร์ที่ใช้ในการพัฒนาโปรแกรมของผู้รับบริการซึ่งโปรแกรมพรีโพรเซสเซอร์จะสามารถวิเคราะห์และแทรกคำสั่งการทำงานได้เพียงภาษาเดียวเท่านั้นคือภาษาจาวา[7]

โปรแกรมพรีโพรเซสเซอร์ที่ถูกพัฒนาขึ้นจะทำการแทรกคำสั่งลงในโปรแกรมผู้รับบริการโดยคำสั่งการทำงานนี้จะคอยตรวจจับความผิดพลาดที่อาจเกิดขึ้นในขณะที่โปรแกรมของผู้รับบริการทำการค้นหาข้อมูลอ้างอิงของบริการที่ต้องการใช้งานผ่านทางออร์บด้วยการเรียกใช้คำสั่ง bind ของซอฟต์แวร์วิสิโบรคเกอร์ การดักจับความผิดพลาดที่เกิดขึ้นนี้จะใช้การดักจับที่เอ็กซ์เซพชันของระบบแบบ org.omg.CORBA.NO_IMPLEMENT ซึ่งออร์บของคอร์บาจะทำให้เกิดขึ้นเมื่อออร์บไม่สามารถส่งกลับข้อมูลอ้างอิงของบริการที่ผู้รับบริการต้องการใช้งานได้ตามรูปแบบที่ระบุไว้ในข้อกำหนดคอร์บาของโอเอ็มจี [1] เมื่อเกิดเอ็กซ์เซพชันขึ้นจากการเรียกใช้คำสั่ง bind คำสั่งการทำงานที่แทรกลงในโปรแกรมของผู้รับบริการจะทำการติดต่อไปยังบริการค้นหา (Searching Service) เพื่อทำการค้นหาบริการอื่นที่สามารถใช้งานแทนที่บริการที่ไม่พร้อมทำงานได้ในขณะนั้น บริการค้นหาจะท่องไปในข้อมูลความสัมพันธ์แบบเท่าเทียมกันของบริการที่ถูกจัดเก็บไว้ภายในคลังจัดเก็บส่วนต่อประสาน แล้วทำการคัดเลือกบริการที่สามารถใช้งานแทนที่ได้ก่อนที่จะส่งข้อมูลอ้างอิงของตัวดำเนินการแปลงกลับไปให้โปรแกรมของผู้รับบริการ โปรแกรมผู้รับบริการจะทำการติดต่อและเรียกใช้งานตัวดำเนินการแปลงที่ได้รับจากบริการค้นหาเพื่อทำงานตามกลไกที่ได้กล่าวถึงไปแล้วในงานวิจัย [9] ต่อไป

เนื่องจากคำสั่ง bind ในซอฟต์แวร์วิสิโบรคเกอร์มีรูปแบบการค้นหาข้อมูลอ้างอิงของบริการที่ผู้รับบริการสามารถเรียกใช้งานได้แตกต่างกันทั้งสิ้น 3 แบบ (รายละเอียดของรูปแบบคำสั่ง bind ทั้ง 3 แบบอยู่ในหัวข้อที่ 2.1) คำสั่ง bind ทั้ง 3 แบบจะใช้พารามิเตอร์ในการค้นหาอินสแตนซ์ของบริการที่มาลงทะเบียนประกาศการให้บริการไว้ในสมาร์ทเอเจนท์ที่แตกต่างกันอาทิเช่นคำสั่ง bind แบบที่ 1 จะถูกใช้เมื่อผู้รับบริการต้องการใช้งานอินสแตนซ์ใดๆของบริการที่มีส่วนต่อประสานตามที่ระบุมาในคำสั่ง bind (ผู้รับบริการเจาะจงเฉพาะส่วนต่อประสานของบริการ) ในขณะที่คำสั่ง bind แบบที่ 2 จะหมายถึงการที่ผู้รับบริการเจาะจงการใช้งานบริการในระดับอินสแตนซ์โดยระบุทั้งส่วนต่อประสานตลอดจนชื่ออินสแตนซ์ของบริการที่ต้องการใช้งาน

ด้วยสาเหตุดังกล่าวนี้ ส่งผลให้โปรแกรมพรีโพรเซสเซอร์ในงานวิจัยนี้ต้องทำการแทรกคำสั่งการทำงานลงในโปรแกรมผู้รับบริการเพื่อสร้างกลไกการค้นหาบริการแทนที่ในรูปแบบที่แตกต่างกันไปตามรูปแบบการเรียกใช้คำสั่ง bind ด้วยเช่นกัน การค้นหาบริการแทนที่ที่แตกต่างกันนี้จะอยู่ในรูปของการเรียก

---

[7] ภาษาจาวา (Java Language) เป็นภาษาโปรแกรมเชิงวัตถุ (Object-Oriented Programming) ที่ถูกพัฒนาขึ้นโดยซันไมโครซิสเต็มซ์ (Sun Microsystems) ให้เป็นภาษาที่เป็นอิสระต่อแพล็ตฟอร์ม (Platform Independent) โดยใช้แนวความคิดของการแปลงโปรแกรมต้นฉบับไปเป็นไบต์โค้ด (Bytecode) ก่อนที่จะนำไปใช้งานในจาวาเวอร์ชวลแมชชีน (Java Virtual Machine)

ใช้งานเมทอดของบริการค้นหาที่แตกต่างกันไป (รายละเอียดเมทอดต่างๆของบริการค้นหาที่จะถูกเรียกใช้งานในการค้นหาบริการแทนที่อยู่ในหัวข้อที่ 3.2)

เมื่อผู้พัฒนาโปรแกรมของผู้รับบริการเรียกใช้งานโปรแกรมพรีโพรเซสเซอร์เพื่อทำการแทรกคำสั่งการทำงานลงในโปรแกรมผู้รับบริการ โปรแกรมพรีโพรเซสเซอร์จะแทรกคำสั่งให้โปรแกรมของผู้รับบริการเรียกใช้งานอินสแตนซ์ของบริการค้นหาโดยใช้การแปลงข้อมูลอ้างอิงของบริการค้นหาที่ถูกจัดเก็บอยู่ภายในแฟ้มข้อมูลในรูปของอักขระไปเป็นวัตถุคอร์บา (CORBA Object) ก่อนที่จะสามารถเรียกใช้เมทอดของบริการค้นหาได้ การแปลงข้อมูลอ้างอิงของบริการค้นหาในลักษณะดังกล่าวนี้จะกระทำผ่านเมทอด string_to_object ของออร์บตามรูปแบบที่กำหนดไว้ในข้อกำหนดคอร์บา [1,5] ตัวอย่างคำสั่งที่ถูกแทรกลงในโปรแกรมผู้รับบริการโดยโปรแกรมพรีโพรเซสเซอร์เพื่อทำการเรียกใช้เมทอดของบริการค้นหาเป็นดังรูปที่ 3.2

```
// สมมติให้ข้อมูลอ้างอิงของบริการค้นหาถูกจัดเก็บอยู่ในแฟ้มข้อมูลชื่อ irserv.ior ภายใต้ไดเร็คทอรีชื่อ ior
// โปรแกรมพรีโพรเซสเซอร์จะแทรกคำสั่งการอ่านข้อมูลอ้างอิงของบริการค้นหาจากแฟ้มข้อมูลและจัดเก็บลง
// ในตัวแปร gv_client_m2
BufferedReader gv_client_m1 = new BufferedReader(new FileReader("c:\\ior\\irserv.ior"));
String gv_client_m2 = gv_client_m1.readLine();
org.omg.CORBA.ORB gv_client_m3 = org.omg.CORBA.ORB.init();
// แปลงข้อมูลอ้างอิงของบริการค้นหาไปเป็นวัตถุคอร์บาก่อนเรียกใช้งานเมทอด search_equ
EQIR.extend_ir gv_client_m4 = EQIR.extend_irHelper.narrow
                                        (gv_client_m3.string_to_object(gv_client_m2));
// เรียกใช้เมทอด search_equ ของบริการค้นหาเพื่อทำการค้นหาบริการแทนที่จากคลังจัดเก็บส่วนต่อประสาน
String gv_client_m5 = gv_client_m4.search_equ("IDL:service_by_id/account_by_id:1.0");
If(!gv_client_m5.trim().equals(""))
    // เมื่อพบบริการแทนที่โปรแกรมผู้รับบริการจะแปลงข้อมูลอ้างอิงของตัวดำเนินการแปลงไปเป็น
    // วัตถุคอร์บาก่อนที่จะจัดเก็บลงในตัวแปร ser1 เพื่อนำไปใช้งาน
    ser1 = service_by_id.account_by_idHelper.narrow(gv_client_m3.string_to_object(gv_client_m5));
else{
    // โปรแกรมผู้รับบริการจะหยุดทำงานเมื่อไม่พบบริการแทนที่ภายในคลังจัดเก็บส่วนต่อประสาน
    System.out.println("No any equivalent service");
    System.exit(1);
}
```

รูปที่ 3.2 ตัวอย่างคำสั่งที่แทรกลงในโปรแกรมผู้รับบริการเพื่อค้นหาบริการแทนที่

3.2 **บริการค้นหา** (Searching Service)

งานวิจัยนี้ได้ออกแบบให้บริการค้นหาทำงานเป็นบริการชนิดหนึ่งตามข้อกำหนดของคอร์บาโดยมีหน้าที่ให้บริการค้นหาบริการอื่นที่สามารถใช้งานแทนที่บริการที่ผู้ใช้งานต้องการได้ บริการค้นหาจะถูกเรียกใช้งานเมื่อโปรแกรมของผู้รับบริการไม่สามารถเรียกใช้บริการใดๆที่ต้องการได้ตามปกติ คำสั่งที่แทรกลงในโปรแกรมของผู้รับบริการโดยโปรแกรมพรีโพรเซสเซอร์จะทำการติดต่อและเรียกใช้งานเมทอดภายในบริการค้นหาโดยอัตโนมัติ เมื่อถูกเรียกใช้งานบริการค้นหาจะท่องไปในข้อมูลความสัมพันธ์แบบเท่าเทียมกันภายในคลังจัดเก็บส่วนต่อประสานตามงานวิจัย [4] เพื่อทำการคัดเลือกและส่งกลับข้อมูลอ้างอิงของตัวดำเนินการแปลงที่กำหนดขึ้นโดยผู้ให้บริการทดแทนเพื่อให้โปรแกรมผู้รับบริการนำไปใช้งานในขั้นตอนอื่นต่อไป

งานวิจัยนี้ได้ทำการออกแบบและพัฒนาโปรแกรมของบริการค้นหา โดยมีรายละเอียดของส่วนต่อประสานตามรูปที่ 3.3

```
#ifndef EQIR_IDL
#define EQIR_IDL
#include <CosTrading.idl>
  module EQIR{
      interface extend_ir {
          string search_equ(in string orgrepid);                          (1)
          string search_equ_obj(in string orgrepid,in string orgobjname);  (2)
          string search_equ_host(in string orgrepid,in string orgobjname,  (3)
                  in string   orghost);
          CosTrading::OfferSeq search_equ_trader(
                  in CosTrading::ServiceTypeName name);                    (4)
      };
  };
#endif
```

รูปที่ 3.3 ส่วนต่อประสานของบริการค้นหา

บริการค้นหาในงานวิจัยนี้มีชื่อส่วนต่อประสานเป็น extend_ir ซึ่งส่วนต่อประสานนี้จะมีเมทอดให้โปรแกรมผู้รับบริการสามารถเรียกใช้งานเพื่อทำการค้นหาบริการแทนที่ได้ 4 เมทอดดังนี้

1. เมทอด search_equ

เป็นเมทอดของบริการค้นหาซึ่งถูกใช้ในการค้นหาบริการแทนที่จากคลังจัดเก็บส่วนต่อประสานโดยเมทอดนี้จะรับค่าพารามิเตอร์ด้านอินพุต 1 ค่าคือค่าไอดีของการจัดเก็บ (Repository ID[8]) เมื่อโปรแกรมผู้รับบริการเกิดเอ็กซ์เซพชันขึ้นขณะเรียกใช้งานคำสั่ง bind แบบที่ 1 คำสั่งที่ถูกแทรกลงในโปรแกรมของผู้รับบริการจะทำการติดต่อและเรียกใช้เมทอด search_equ ของบริการค้นหาโดยทำการระบุชื่อส่วนต่อประสานตลอดจนรุ่นของบริการที่ต้องการใช้งานมาในพารามิเตอร์ค่านี้ ตัวอย่างเช่นถ้าโปรแกรมผู้รับบริการทำการระบุค่าไอดีของการจัดเก็บให้เท่ากับ IDL:service_by_id/ account_by_id:1.0 จะหมายถึงการที่โปรแกรมของผู้รับบริการไม่สามารถเรียกใช้อินสแตนซ์ใดๆของบริการที่มีส่วนต่อประสานเป็น ::service_by_id::account_by_id ได้ ดังนั้นโปรแกรมของผู้รับบริการจึงต้องการให้บริการค้นหาส่งกลับข้อมูลอ้างอิงของตัวดำเนินการแปลงที่จะช่วยทำให้โปรแกรมของผู้รับบริการสามารถเรียกใช้งานบริการแทนที่ใดๆที่สามารถทำงานได้เท่าเทียมกับบริการซึ่งมีส่วนต่อประสานเป็น ::service_by_id::account_by_id และมีรุ่นของบริการเป็น 1.0 เป็นต้น (รูปที่ 3.4)

ตัวอย่างการเรียกใช้คำสั่ง bind รูปแบบที่ 1 ในโปรแกรมของผู้รับบริการ

```
service_by_id.account_by_id ser1 = service_by_id.account_by_idHelper.bind(orb);
```

คำสั่งที่แทรกโดยโปรแกรมพรีโพรเซสเซอร์เพื่อเรียกใช้งานบริการค้นหา

```
String mapior = search_service.search_equ("IDL:service_by_id/account_by_id:1.0");
```

รูปที่ 3.4 การเรียกใช้เมทอด search_equ เมื่อผู้รับบริการเรียกใช้คำสั่ง bind ในรูปแบบที่ 1

เมื่อเมทอด search_equ ถูกเรียกใช้งานบริการค้นหาจะทำการค้นหาข้อมูลจากคลังจัดเก็บส่วนต่อประสานและส่งกลับข้อมูลอ้างอิงของตัวดำเนินการแปลงตัวแรกที่พบว่าวัตถุอินเตอร์เฟสเดฟนั้นมีชื่อส่วนต่อประสานตรงกับค่าที่ระบุมาในพารามิเตอร์ บริการค้นหาจะนำข้อมูลอ้างอิงของตัวดำเนินการแปลงภายในวัตถุอิควิวาเลนซ์เดฟของวัตถุอินเตอร์เฟสเดฟนั้นส่งไปยังโปรแกรมผู้รับบริการ การค้นหาบริการแทนที่ในเมทอดนี้จะให้ความสนใจเฉพาะชื่อส่วนต่อประสานเท่านั้นโดยจะไม่พิจารณาถึงชื่ออินสแตนซ์ของบริการตลอดจนชื่อโฮสต์ที่บริการนั้นทำงานอยู่แต่อย่างใด บริการค้นหาจะส่งกลับข้อมูลอ้างอิงของตัวดำเนินการแปลงที่กำหนดไว้ภายในวัตถุอิควิวาเลนซ์เดฟของบริการทดแทนในรูปของสายอักขระ

---

[8] ข้อกำหนดคอร์บาของโอเอ็มจีใช้ค่าไอดีของการจัดเก็บ (Repository ID) เป็นคีย์ในการค้นหารายละเอียดส่วนต่อประสานของบริการใดๆภายในคลังจัดเก็บส่วนต่อประสาน (Interface Repository)

(String) ไปยังโปรแกรมผู้รับบริการ ซึ่งโปรแกรมผู้รับบริการจะเรียกใช้เมทอด string_to_object ในออร์บเพื่อทำการแปลงสายอักขระนี้ไปเป็นวัตถุคอร์บาก่อนที่จะสามารถเรียกใช้งานตัวดำเนินการแปลงนี้ได้ ในกรณีที่บริการค้นหาไม่พบบริการแทนที่ใดๆเลยภายในคลังจัดเก็บส่วนต่อประสานที่เหมาะสมกับค่าพารามิเตอร์ที่ผู้ใช้บริการกำหนดมา บริการค้นหาจะส่งกลับค่า Null ไปยังโปรแกรมผู้รับบริการ ซึ่งจะส่งผลให้โปรแกรมผู้รับบริการหยุดการทำงาน

2. เมทอด search_equ_obj

เมทอดนี้จะถูกเรียกใช้งานเมื่อโปรแกรมของผู้รับบริการเกิดเอ็กซ์เซพชันจากการเรียกใช้คำสั่ง bind ในรูปแบบที่ 2 หรือเมื่อโปรแกรมผู้รับบริการพยายามที่จะเรียกใช้บริการจากข้อมูลอ้างอิงของบริการใดๆที่ถูกจัดเก็บเอาไว้ในแฟ้มข้อมูล (Persistent IOR) แต่บริการนั้นไม่ได้ทำงานอยู่ เมทอด search_equ_obj จะรับค่าพารามิเตอร์ด้านอินพุต 2 ค่าคือค่าไอดีของการจัดเก็บ ซึ่งระบุถึงข้อมูลของบริการที่ผู้ใช้งานต้องการเช่นเดียวกันกับกรณีของเมทอด search_equ ส่วนค่าพารามิเตอร์ที่ 2 จะเป็นชื่ออินสแตนซ์ของบริการ (Service Name) ที่ผู้รับบริการต้องการใช้งานนั่นเอง

ตัวอย่างเช่นถ้าโปรแกรมของผู้รับบริการต้องการใช้งานอินสแตนซ์ของบริการชื่อ service1 ที่มีส่วนต่อประสานเป็น ::service_by_id::account_by_id จึงทำการเรียกใช้คำสั่ง bind ในรูปแบบที่ 2

หากอินสแตนซ์ของบริการดังกล่าวนี้ไม่ได้ทำงานอยู่จะส่งผลให้ออร์บในฝั่งของโปรแกรมผู้รับบริการทำให้เกิดเอ็กซ์เซพชันขึ้น โปรแกรมผู้รับบริการที่ผ่านการแทรกคำสั่งโดยโปรแกรมพรีโพรเซสเซอร์จะทำการติดต่อไปยังบริการค้นหาเพื่อค้นหาบริการแทนที่ใดๆที่สามารถทำงานได้เท่าเทียมกันกับบริการที่ต้องการใช้งานโดยเรียกไปยังเมทอด search_equ_obj ของบริการค้นหาและทำการระบุค่าไอดีของการจัดเก็บเป็น IDL:service_by_id/account_by_id:1.0 ตลอดจนกำหนดชื่ออินสแตนซ์ของบริการเป็นค่า service1 ตามลำดับ (รูปที่ 3.5)

ตัวอย่างการเรียกใช้คำสั่ง bind แบบที่ 2 ในโปรแกรมของผู้รับบริการ

```
service_by_id.account_by_id ser1 = service_by_id.account_by_idHelper.bind(orb,"service1");
```

คำสั่งที่แทรกโดยโปรแกรมพรีโพรเซสเซอร์เพื่อเรียกใช้งานบริการค้นหา

```
String mapior = search_service.search_equ_obj("IDL:service_by_id/account_by_id:1.0","service1");
```

รูปที่ 3.5 การเรียกใช้เมทอด search_equ_obj เมื่อผู้ใช้บริการเรียกใช้คำสั่ง bind ในรูปแบบที่ 2

เมื่อเมทอด search_equ_obj ถูกเรียกใช้งานบริการค้นหาจะทำการค้นหาข้อมูลจากคลังจัดเก็บส่วนต่อประสานโดยใช้เงื่อนไขในการค้นหาที่แตกต่างกันกับกรณีของเมทอด search_equ ทั้งนี้เมทอด search_equ_obj จะส่งกลับข้อมูลอ้างอิงของตัวดำเนินการแปลงตัวแรกจากวัตถุอิควิวาเลนซ์เดฟภายใน

วัตถุอินเตอร์เฟสเดฟที่พบว่าวัตถุอินเตอร์เฟสเดฟนั้นมีชื่อส่วนต่อประสานและชื่ออินสแตนซ์ตรงกับค่าที่ระบุมาในพารามิเตอร์ การค้นหาบริการแทนที่ในเมทอดนี้จะให้ความสนใจทั้งชื่อส่วนต่อประสานตลอดจนชื่ออินสแตนซ์ของบริการแต่จะไม่พิจารณาชื่อโฮสต์ที่บริการนั้นทำงานอยู่แต่อย่างใด

บริการค้นหาจะส่งกลับข้อมูลอ้างอิงของตัวดำเนินการแปลงที่พบในรูปของสายอักขระ ซึ่งโปรแกรมผู้รับบริการจะเรียกใช้เมทอด string_to_object ของออร์บเพื่อทำการแปลงสายอักขระนี้ไปเป็นวัตถุคอร์บาก่อนที่จะสามารถนำไปใช้งานได้

3. เมทอด search_equ_host

เมทอดนี้จะถูกเรียกใช้งานเมื่อโปรแกรมของผู้รับบริการเกิดเอ็กซ์เซพชันขึ้นจากการเรียกใช้คำสั่ง bind ในรูปแบบที่ 3 โดยเมทอด search_equ_host จะรับค่าพารามิเตอร์ด้านอินพุตทั้งสิ้น 3 ค่าได้แก่ ค่าไอดีของการจัดเก็บ ชื่ออินสแตนซ์ของบริการที่ผู้ใช้งานต้องการ ตลอดจนโฮสต์ที่อินสแตนซ์ของบริการนั้นทำงานอยู่

การค้นหาบริการแทนที่โดยเมทอดนี้จะใช้ค่าพารามิเตอร์ทั้ง 3 ค่าในการค้นหาบริการแทนที่จากคลังจัดเก็บส่วนต่อประสาน นั่นคือบริการค้นหาจะส่งกลับข้อมูลอ้างอิงของตัวดำเนินการแปลงตัวแรกจากวัตถุอิควิวาเลนซ์เดฟภายในวัตถุอินเตอร์เฟสเดฟที่พบว่าวัตถุอินเตอร์เฟสเดฟนั้นมีชื่อส่วนต่อประสานและชื่ออินสแตนซ์ตลอดจนชื่อโฮสต์ที่ตรงกันกับค่าที่ระบุมาในค่าพารามิเตอร์ เมทอด search_equ_host จะส่งกลับข้อมูลอ้างอิงของตัวดำเนินการแปลงในรูปของสายอักขระกลับไปยังโปรแกรมของผู้รับบริการเช่นเดียวกันกับเมทอด search_equ และ search_equ_obj ที่กล่าวถึงไปแล้ว

ตัวอย่างเช่นถ้าโปรแกรมผู้รับบริการต้องการใช้งานอินสแตนซ์ของบริการชื่อ service1 ที่มีส่วนต่อประสานเป็น ::service_by_id::account_by_id โดยบริการนี้ทำงานอยู่ในโฮสต์ชื่อ host1 (มีเลขที่อยู่ไอพีเป็น 155.5.1.9) จึงทำการเรียกใช้คำสั่ง bind ในรูปแบบที่ 3 หากอินสแตนซ์ของบริการดังกล่าวนี้ไม่ได้ทำงานอยู่จะส่งผลให้ออร์บในฝั่งของโปรแกรมผู้รับบริการทำให้เกิดเอ็กซ์เซพชันขึ้นภายหลังการเรียกใช้คำสั่ง bind โปรแกรมของผู้รับบริการที่ผ่านการแทรกคำสั่งโดยโปรแกรมพรีโพรเซสเซอร์จะติดต่อไปยังบริการค้นหาเพื่อค้นหาบริการแทนที่ใดๆที่สามารถทำงานได้เท่าเทียมกับบริการที่ผู้ใช้งานต้องการโดยเรียกใช้เมทอด search_equ_host และทำการระบุค่าไอดีของการจัดเก็บ ชื่ออินสแตนซ์ของบริการ ตลอดจนชื่อโฮสต์ที่บริการนั้นทำงานอยู่เป็นค่า IDL:service_by_id/account_by_id:1.0, service1 และ 155.5.1.9 ตามลำดับ (รูปที่ 3.6)

ตัวอย่างการเรียกใช้คำสั่ง bind แบบที่ 3 ในโปรแกรมของผู้รับบริการ

```
service_by_id.account_by_id ser1 = service_by_id.account_by_idHelper.bind(orb,"service1",
                                    "155.5.1.9", bindopt);
```

คำสั่งที่แทรกโดยโปรแกรมพรีโพรเซสเซอร์เพื่อเรียกใช้งานบริการค้นหา

```
String mapior = search_service.search_equ_host("IDL:service_by_id/account_by_id:1.0",
                    "service1","155.5.1.9");
```

รูปที่ 3.6 การเรียกใช้เมทอด search_equ_host เมื่อผู้ใช้บริการเรียกใช้คำสั่ง bind ในรูปแบบที่ 3

4. เมทอด search_equ_trader

การค้นหาบริการแทนที่ในเมทอดนี้จะแตกต่างกับเมทอดทั้ง 3 ที่กล่าวถึงมาแล้ว โดยงานวิจัยนี้ได้กำหนดให้เมทอด search_equ_trader ถูกเรียกใช้งานจากบริการเทรดเดอร์ในกรณีที่บริการเทรดเดอร์ไม่พบข้อเสนอบริการใดเลยที่ตรงกับความต้องการของอิมพอร์ตเตอร์ (ดูรายละเอียดในบทที่ 4)

เมทอด search_equ_trader ของบริการค้นหาจะรับชื่อชนิดของบริการ (Service Type Name) เข้ามาเป็นค่าพารามิเตอร์ด้านอินพุต ตัวอย่างเช่นถ้าโปรแกรมผู้รับบริการต้องการอิมพอร์ตข้อเสนอบริการใดๆ ที่มีชนิดของบริการเป็น ::servicetype1 จากเทรดเดอร์ โปรแกรมของผู้รับบริการจะเรียกใช้เมทอด query ของเทรดเดอร์โดยทำการกำหนดชนิดของบริการให้เป็น ::servicetype1 แต่ถ้าหากภายในเทรดเดอร์ไม่มีผู้ให้บริการใดเลยมาลงทะเบียนประกาศการให้บริการชนิดนี้เอาไว้ กลไกส่วนขยายในเทรดเดอร์ที่ถูกแก้ไขเพิ่มเติมดังในบทที่ 4 จะทำการเรียกใช้เมทอด search_equ_trader ของบริการค้นหาโดยกำหนดค่าพารามิเตอร์ของเมทอดนี้ให้เท่ากับ ::servicetype1 เพื่อให้บริการค้นหาส่งกลับกลุ่มข้อเสนอบริการ (Service Offer Sequence) ใดๆที่อิมพอร์ตเตอร์อาจใช้งานแทนที่ได้กลับไปให้เทรดเดอร์เพื่อทำการพิจารณาคุณสมบัติของข้อเสนอบริการนั้นอีกครั้ง โดยข้อเสนอบริการที่ส่งกลับไปนี้อาจมีชนิดของบริการเป็นชื่ออื่นๆที่ไม่ใช่ ::servicetype1 แต่เป็นชนิดของบริการที่ทำงานได้เท่าเทียมกัน บริการเทรดเดอร์จะพิจารณากลุ่มข้อเสนอบริการที่ได้รับจากบริการค้นหาตามเงื่อนไขการค้นหา (Constraint) ที่อิมพอร์ตเตอร์ระบุมาในเมทอด query ก่อนที่จะส่งกลับเฉพาะข้อเสนอบริการที่สอดคล้องกับค่าพารามิเตอร์ในเมทอด query เท่านั้นไปยังอิมพอร์ตเตอร์

3.3 การจำลองส่วนขยายของคลังจัดเก็บส่วนต่อประสาน

งานวิจัยนี้ไม่สามารถพัฒนาโปรแกรมในส่วนเมทอดต่างๆของบริการค้นหาให้ทำการคัดเลือกบริการแทนที่จากงานการพัฒนาส่วนขยายให้กับคลังจัดเก็บส่วนต่อประสานของคอร์บาเพื่อรองรับข้อมูลความสัมพันธ์แบบเท่าเทียมกัน [4] ได้โดยตรงทั้งนี้เนื่องจากงานดังกล่าวแล้วเสร็จล่าช้ากว่ากำหนด ส่งผลให้

งานวิจัยนี้ต้องใช้การจำลองคลังและทำการจัดเก็บข้อมูลต่างๆเท่าที่จำเป็นต่อการทำวิจัยลงในแฟ้มข้อมูลเพื่อใช้งานร่วมกับบริการค้นหา งานวิจัยนี้ได้ออกแบบโครงสร้างการจัดเก็บข้อมูลความสัมพันธ์แบบเท่าเทียมกันของบริการลงในแฟ้มข้อมูลชื่อ config.eq ตามรูปที่ 3.7

```
<start>
desire_repositoryid,desire_instance,desire_host_ip
eq_repositoryid,eq_instance,eq_host_ip
mapping operator's IOR file
servicetypename,no of service_property
property_name,property_type,property_value
<end>
```

คำอธิบาย

- *<start> และ <end> เป็นโทเคนที่ใช้แสดงขอบเขตของค่าติดตั้งระหว่างคู่อินสแตนซ์ของบริการใดๆ*
- *desire_repositoryid คือค่าไอดีของการจัดเก็บของบริการหลัก*
- *desire_instance คือชื่ออินสแตนซ์ของบริการหลัก*
- *desire_host_ip คือค่าเลขที่อยู่ไอพีของโฮสต์ซึ่งมีบริการหลักทำงานอยู่*
- *eq_repositoryid คือค่าไอดีของการจัดเก็บของบริการที่แทนที่*
- *eq_instance คือชื่ออินสแตนซ์ของบริการที่แทนที่*
- *eq_host_ip คือค่าเลขที่อยู่ไอพีของโฮสต์ซึ่งมีบริการแทนที่ทำงานอยู่*
- *mapping operator's IOR file คือชื่อแฟ้มข้อมูลที่จัดเก็บข้อมูลอ้างอิงของตัวดำเนินการแปลงสำหรับใช้สร้างกลไกการแทนที่กันของบริการ*
- *servicetypename คือชื่อชนิดของบริการหลัก*
- *no of service_property คือจำนวนคุณสมบัติของบริการหลัก*
- *property_name คือชื่อคุณสมบัติของบริการหลัก*
- *property_type คือชนิดคุณสมบัติของบริการหลัก*
- *property_value คือค่าคุณสมบัติของบริการหลัก*

รูปที่ 3.7 โครงสร้างแฟ้มข้อมูลที่ใช้จัดเก็บข้อมูลความสัมพันธ์แบบเท่าเทียมกันของบริการ

ผู้ให้บริการแทนที่ซึ่งมีหน้าที่รับผิดชอบในการกำหนดข้อมูลความสัมพันธ์แบบเท่าเทียมกันของบริการจะเป็นผู้กำหนดค่าลงในแฟ้มข้อมูล config.eq ได้ตามตัวอย่างดังต่อไปนี้

ตัวอย่างเช่นสมมติให้ผู้ให้บริการแทนที่กำหนดข้อมูลความสัมพันธ์ระหว่างคู่อินสแตนซ์ของบริการ ser2 ที่มีส่วนต่อประสานเป็น IDL:service_by_name/account_by_name:1.0 และทำงานอยู่บนโฮสต์ที่มีเลขที่อยู่ไอพีเป็น 155.7.1.200 ให้สามารถทำงานแทนที่บริการ ser1 ที่มีส่วนต่อประสานเป็น IDL:service_by_id/account_by_id:1.0 และทำงานอยู่ในโฮสต์ที่มีเลขที่อยู่ไอพีเป็น 155.7.2.100 ได้โดยกำหนดให้โปรแกรมของผู้รับบริการต้องทำการเรียกใช้บริการ ser2 ผ่านทางตัวดำเนินการแปลงซึ่งมีข้อมูลอ้างอิงถูกจัดเก็บอยู่ในแฟ้มข้อมูล map1.ior สมมติให้บริการ ser1 เป็นบริการที่ใช้ในการค้นหายอดเงินคงเหลือของบัญชีเงินฝากจากเลขที่บัญชี ซึ่งบริการนี้จะมีชนิดของบริการเป็น balance_by_id ชนิดของบริการแบบนี้จะมีคุณสมบัติของบริการทั้งสิ้น 4 ค่าได้แก่

- ค่า searchby เป็นคุณสมบัติที่มีชนิดเป็นสายอักขระใช้สำหรับแสดงคีย์ที่บริการ ser1 ใช้ในการค้นหาค่ายอดคงเหลือของบัญชีจากฐานข้อมูลเงินฝากในที่นี้สมมติให้เท่ากับ id
- ค่า area เป็นคุณสมบัติที่มีชนิดเป็นสายอักขระสำหรับใช้ในการแสดงพื้นที่การให้บริการที่บริการ ser1 สามารถทำการค้นหาข้อมูลยอดคงเหลือของบัญชีได้ ในที่นี้สมมติให้เท่ากับ NSEW (North, South, East และ West)
- ค่า cost เป็นคุณสมบัติที่ใช้แสดงค่าใช้จ่ายในการค้นหาข้อมูลด้วยบริการ ser1 ต่อการค้นหา 1 ครั้ง สมมติให้มีค่าเท่ากับ 10 บาท/ครั้ง
- ค่า responsetime เป็นคุณสมบัติของบริการที่ใช้แสดงเวลาที่ต้องใช้ในการค้นหาข้อมูลจากฐานข้อมูล สมมติให้มีค่าไม่เกิน 40 ms/ครั้ง

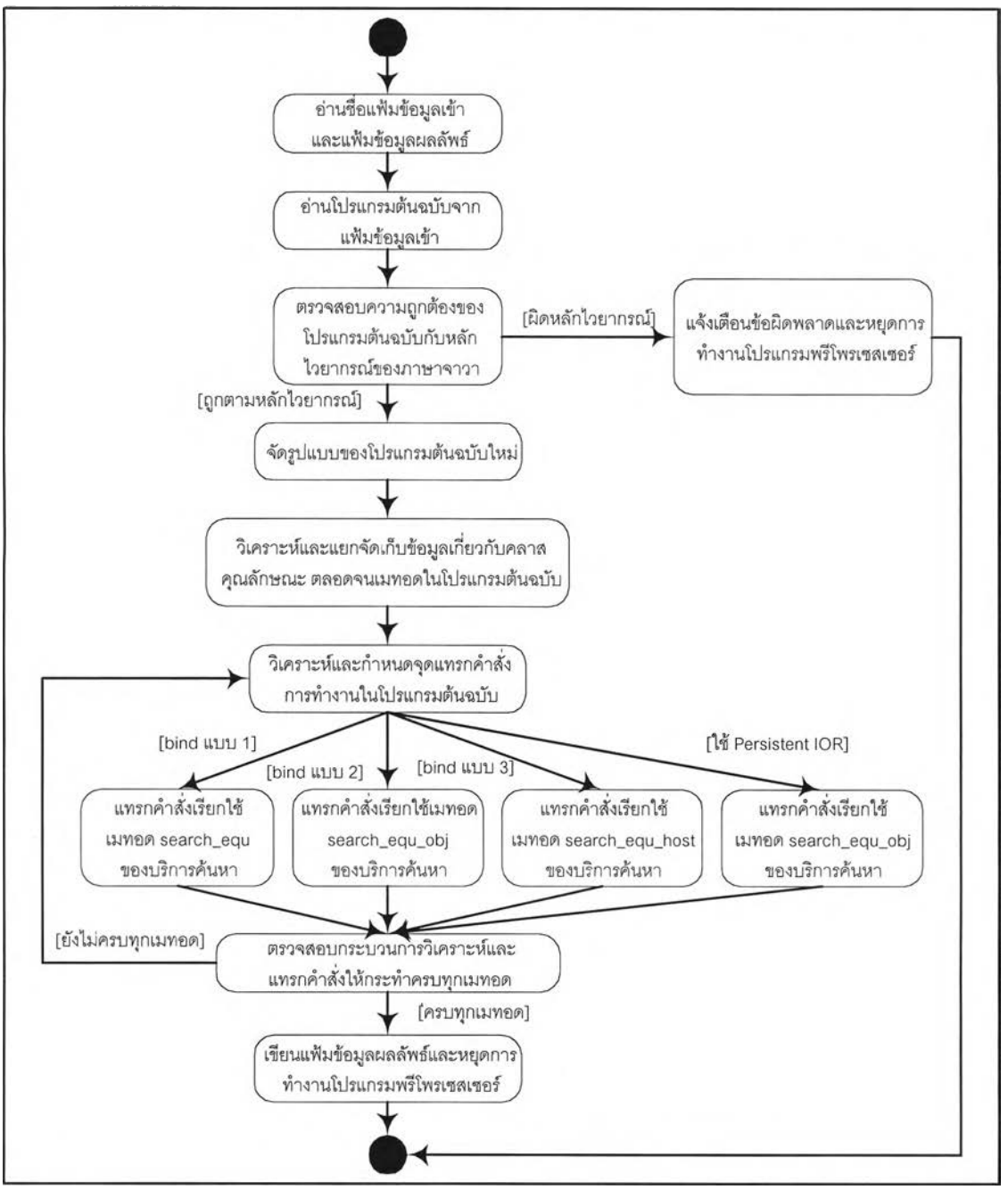
ผู้ให้บริการแทนที่จะต้องกำหนดค่าติดตั้งลงในแฟ้มข้อมูล config.eq ตามรูปที่ 3.8

```
<start>
IDL:service_by_id/account_by_id:1.0,ser1,155.7.2.100
IDL:service_by_name/account_by_name:1.0,ser2,155.7.1.200
c:\thesis\ior\map1.ior
balance_by_id,4
searchby,string,id
area,string,NSEW
cost,float,30.00
responsetime,float,40.00
<end>
```

รูปที่ 3.8 ตัวอย่างการกำหนดข้อมูลความสัมพันธ์แบบเท่าเทียมกันลงในแฟ้มข้อมูล config.eq

3.4 การออกแบบและพัฒนาโปรแกรมพรีโพรเซสเซอร์

งานวิจัยนี้ได้ออกแบบโปรแกรมพรีโพรเซสเซอร์ให้ทำการวิเคราะห์โปรแกรมต้นฉบับ (Source Program) ของผู้รับบริการที่ถูกพัฒนาขึ้นด้วยภาษาจาวาและทำการแทรกคำสั่งลงในตำแหน่งที่โปรแกรมของผู้รับบริการทำการค้นหาข้อมูลอ้างอิงของบริการต่างๆผ่านทางออร์บ งานวิจัยนี้ได้ออกแบบให้โปรแกรมพรีโพรเซสเซอร์มีขั้นตอนการทำงานตามแอ็คทิวิตีไดอะแกรม (Activity Diagram) ในรูปที่ 3.9



รูปที่ 3.9 แอ็คทิวิตีไดอะแกรมการทำงานของโปรแกรมพรีโพรเซสเซอร์

ในการเรียกใช้โปรแกรมพรีโพรเซสเซอร์ของงานวิจัยนี้เพื่อทำการแทรกคำสั่งการทำงานลงในโปรแกรมของผู้รับบริการนั้นผู้ใช้จะต้องระบุชื่อแฟ้มข้อมูลเข้าที่ใช้จัดเก็บโปรแกรมผู้รับบริการ (Input File) ตลอดจนชื่อแฟ้มข้อมูลที่ใช้จัดเก็บผลลัพธ์ของโปรแกรมต้นฉบับที่ผ่านการแทรกคำสั่งการทำงานเรียบร้อยแล้ว (Output File) เพื่อจัดส่งเป็นค่าพารามิเตอร์ไปยังโปรแกรมพรีโพรเซสเซอร์ (ผ่านทางคลาส cli2eq) โปรแกรมพรีโพรเซสเซอร์จะเริ่มประมวลผลโดยทำการอ่านโปรแกรมต้นฉบับที่ถูกจัดเก็บอยู่ภายในแฟ้มข้อมูลที่ผู้ใช้งานระบุมาลงในหน่วยความจำ จากนั้นจึงเริ่มทำกระบวนการตรวจสอบความถูกต้องของโปรแกรมต้นฉบับนั้นว่าโปรแกรมผู้รับบริการมีรูปแบบของคำสั่ง (Syntax) สอดคล้องตามหลักไวยากรณ์ของภาษาจาวา (Java Grammar[9]) หรือไม่ งานวิจัยนี้ได้กำหนดให้โปรแกรมพรีโพรเซสเซอร์ทำการแทรกคำสั่งลงเฉพาะในโปรแกรมต้นฉบับที่ผ่านการตรวจสอบแล้วว่ามีความถูกต้องตามหลักไวยากรณ์เท่านั้น โปรแกรมพรีโพรเซสเซอร์จะแสดงข้อความแจ้งเตือนความผิดพลาดของโปรแกรมต้นฉบับ (Error Message) และยุติการทำงานลงถ้าพบว่าโปรแกรมต้นฉบับของผู้รับบริการนั้นมีรูปแบบของคำสั่งที่ไม่ถูกต้อง

งานวิจัยนี้กำหนดให้โปรแกรมพรีโพรเซสเซอร์ทำการตรวจสอบความผิดพลาดโดยใช้คลาส JavaParser ซึ่งเป็นคลาสที่ถูกสร้างขึ้นด้วยโปรแกรมเจทีบีและจาวาซีซี (JTB and JavaCC) การใช้คลาส JavaParser ในการทำพาร์สชิง (Parsing) โปรแกรมต้นฉบับของผู้รับบริการในงานวิจัยนี้จะมีความถูกต้องและเป็นมาตรฐานมากกว่าการพัฒนาคลาสขึ้นใช้งานเอง ทั้งนี้เนื่องจากโปรแกรมเจทีบีและจาวาซีซีเป็นเครื่องมือ (Tools) สำหรับใช้ในการสร้างพาร์สเซอร์ (Parser Generator) ที่ได้รับการรับรองการใช้งานจากบริษัทซันไมโครซิสเต็มซ์ ซึ่งเป็นบริษัทที่เป็นผู้พัฒนาและออกข้อกำหนดภาษาจาวา (Java Language Specification) ขึ้น รายละเอียดของการสร้างคลาส JavaParser ด้วยโปรแกรมเจทีบีและจาวาซีซีมีอยู่ในภาคผนวก ข

ภายหลังจากที่โปรแกรมพรีโพรเซสเซอร์ได้ทำการตรวจสอบความถูกต้องของโปรแกรมผู้รับบริการเสร็จเรียบร้อยแล้ว คลาส JavaParser จะสร้างซินแท็กซ์ทรี (Abstract Syntax Tree - AST) ขึ้นในหน่วยความจำเพื่อใช้แทน (Representing) องค์ประกอบต่างๆของโปรแกรมต้นฉบับด้วยโหนด (Node) ของซินแท็กซ์ทรี โปรแกรมพรีโพรเซสเซอร์จะใช้คลาส TreeFormatter ในการท่องซินแท็กซ์ทรีในหน่วยความจำเพื่อทำการจัดรูปแบบ (Format) ของโปรแกรมต้นฉบับใหม่ จากนั้นจึงเรียกใช้คลาส TreeDumper ในการอ่านข้อมูลจากหน่วยความจำเพื่อจัดเก็บโปรแกรมต้นฉบับที่ผ่านการจัดรูปแบบให้ง่ายต่อการวิเคราะห์แล้วลงในแฟ้มข้อมูลชั่วคราว (Temporary File) (ทั้งคลาส TreeFormatter และ TreeDumper เป็นคลาสที่ถูกสร้างขึ้นด้วยโปรแกรมเจทีบีและจาวาซีซี เช่นเดียวกัน) โปรแกรมพรีโพรเซสเซอร์จะเรียกใช้

---

[9] งานวิจัยนี้ใช้แฟ้มข้อมูลหลักไวยากรณ์จาวา (Java Grammar File) ชื่อ Java1.1.jj ของบริษัทซันไมโครซิสเต็มซ์ร่วมกับโปรแกรมจาวาซีซีในการสร้างคลาส JavaParser สำหรับใช้ในการตรวจสอบความถูกต้องของโปรแกรมต้นฉบับของผู้รับบริการ

คลาส cli2eq ในการอ่านโปรแกรมต้นฉบับจากแฟ้มข้อมูลชั่วคราวที่สร้างขึ้นนี้เพื่อนำไปผ่านกระบวนการวิเคราะห์และแทรกคำสั่งในขั้นตอนต่อไป

โปรแกรมพรีโพรเซสเซอร์จะเริ่มกระบวนการวิเคราะห์โปรแกรมต้นฉบับของผู้รับบริการโดยใช้เมทอดต่างๆภายในคลาส cli2eq และ util ในการแยกรายละเอียดเกี่ยวกับคอมเมนต์ (Comment) คำสั่งส่วนการอิมพอร์ตคลาสไลบรารี (Import Class Library) และคำสั่งส่วนการระบุแพ็คเกจ (Package) ของคลาสในโปรแกรมต้นฉบับของผู้รับบริการออกไปเพื่อให้โปรแกรมของผู้รับบริการคงเหลือเฉพาะคำสั่งส่วนรายละเอียดการทำงานของคลาส (Class Declaration and Content) เท่านั้น จากนั้นโปรแกรมพรีโพรเซสเซอร์จะทำการแยกแยะรายละเอียดส่วนของการประกาศคลาส (Class Declaration) คุณลักษณะของคลาส (Class Attribute) และเมทอดต่างๆภายในคลาส (Class Method) ออกจากกันและจัดเก็บรายละเอียดส่วนต่างๆที่กล่าวถึงนี้แยกออกเป็นส่วนๆลงในตัวแปรที่มีชนิดเป็นเวคเตอร์ (Vector)[10]

เมื่อโปรแกรมต้นฉบับของส่วนเมทอดต่างๆในคลาสของโปรแกรมผู้รับบริการถูกแยกจัดเก็บลงในเวคเตอร์ที่กำหนดขึ้นแล้ว คลาส cli2eq ของโปรแกรมพรีโพรเซสเซอร์จะนำโปรแกรมต้นฉบับส่วนนั้นๆในเวคเตอร์ไปทำการวิเคราะห์ว่าภายในเมทอดต่างๆของโปรแกรมผู้รับบริการมีการเรียกใช้คำสั่ง bind ของซอฟต์แวร์วิสิโบรคเกอร์เพื่อทำการค้นหาข้อมูลอ้างอิงของบริการผ่านทางออร์บของคอร์บาหรือไม่ โดยใช้เมทอด analysis_method การทำงานภายในเมทอด analysis_method นี้จะแบ่งส่วนการวิเคราะห์ย่อยออกเป็น 2 ลักษณะได้แก่ส่วนการวิเคราะห์และแทรกคำสั่งเมื่อโปรแกรมของผู้รับบริการค้นหาข้อมูลอ้างอิงของบริการผ่านการเรียกใช้คำสั่ง bind ทั้ง 3 แบบของซอฟต์แวร์วิสิโบรคเกอร์ ในกรณีนี้เมทอด analysis_method จะเรียกใช้เมทอด analyse_bind เพื่อทำการประมวลผลและแทรกคำสั่งการทำงานส่วนการเรียกใช้บริการค้นหาลงในโปรแกรมของผู้รับบริการตามรูปแบบที่ปรากฏอยู่ในหัวข้อที่ 3.1 และ 3.2 การวิเคราะห์และแทรกคำสั่งโดยเมทอด analysis_method ในลักษณะที่ 2 จะใช้งานเมื่อโปรแกรมของผู้รับบริการใช้การแปลงข้อมูลอ้างอิงของบริการที่ถูกจัดเก็บอยู่ในแฟ้มข้อมูล (Persistent IOR) ไปเป็นวัตถุคอร์บาเพื่อนำไปใช้งานแทนการค้นหาข้อมูลอ้างอิงของบริการที่ต้องการผ่านการเรียกใช้คำสั่ง bind โปรแกรมของผู้รับบริการจะเกิดเอ็กซ์เซพชั่นขึ้นเช่นเดียวกันถ้าหากว่าอินสแตนซ์ของบริการนั้นๆไม่ได้ทำงานอยู่ในขณะถูกเรียกใช้งาน ในกรณีนี้โปรแกรมพรีโพรเซสเซอร์จะเรียกใช้เมทอด analyse_narrow เพื่อทำการประมวลผลและแทรกคำสั่งการทำงานลงในโปรแกรมผู้รับบริการแทนการเรียกใช้เมทอด analyse_bind ในการแทรกคำสั่งลงในโปรแกรมของผู้รับบริการนั้นโปรแกรมพรี

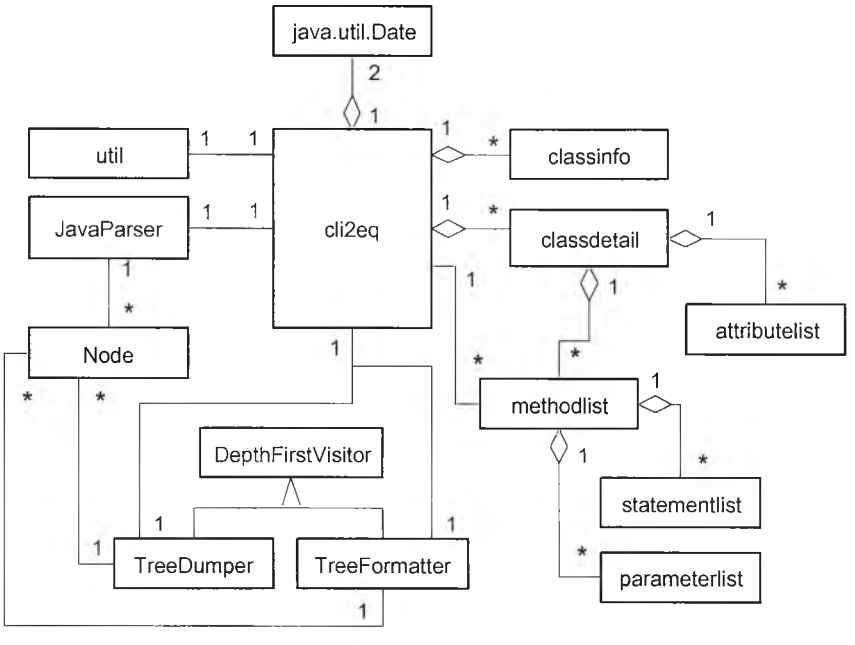
---

[10] ตัวแปรชนิดเวคเตอร์ในภาษาจาวาถูกออกแบบขึ้นสำหรับใช้ในการจัดเก็บข้อมูลที่มีลักษณะเดียวกันกับอะเรย์แต่เวคเตอร์จะมี คุณสมบัติพิเศษที่สามารถเพิ่ม/ลดช่องจัดเก็บสมาชิกได้อย่างพลวัต (Dynamic) ทำให้มีความเหมาะสมอย่างยิ่งในการนำเวคเตอร์มาใช้จัดเก็บเมทอดต่างๆของโปรแกรมผู้รับบริการที่มีจำนวนไม่แน่นอน [11]

โพรเซสเซอร์จะเรียกใช้เมทอด get_newvar_name ของคลาส cli2eq เพื่อทำหน้าที่สร้างชื่อตัวแปรที่ไม่ซ้ำกันกับชื่อตัวแปรที่ปรากฏอยู่แล้วในโปรแกรมของผู้รับบริการ (รายละเอียดของเมทอดนี้อยู่ในตารางที่ 3.2)

เมื่อเมทอดทั้งหมดในโปรแกรมผู้รับบริการได้รับการแทรกคำสั่งโดยโปรแกรมพรีโพรเซสเซอร์แล้ว คลาส cli2eq ของโปรแกรมพรีโพรเซสเซอร์จะทำการสร้างแฟ้มข้อมูลผลลัพธ์เพื่อใช้จัดเก็บโปรแกรมต้นฉบับที่ผ่านการแทรกคำสั่งแล้วโดยใช้ชื่อแฟ้มข้อมูลเป็นชื่อเดียวกันกับที่ผู้ใช้งานระบุมาในค่าพารามิเตอร์ของโปรแกรมพรีโพรเซสเซอร์ จากนั้นจึงแสดงเวลาทั้งหมดที่โปรแกรมพรีโพรเซสเซอร์ใช้ (Elapsing Time) ในกระบวนการวิเคราะห์และแทรกคำสั่งการทำงานลงในโปรแกรมผู้รับบริการก่อนที่จะสิ้นสุดการทำงาน

คลาสไดอะแกรมของโปรแกรมพรีโพรเซสเซอร์ในงานวิจัยนี้จะเป็นไปตามรูปที่ 3.10



รูปที่ 3.10 คลาสไดอะแกรมของโปรแกรมพรีโพรเซสเซอร์ในงานวิจัยนี้

งานวิจัยนี้ได้พัฒนาโปรแกรมพรีโพรเซสเซอร์โดยใช้แนวคิดของการโปรแกรมเชิงวัตถุด้วยภาษาจาวา โปรแกรมพรีโพรเซสเซอร์ที่ถูกพัฒนาขึ้นจะประกอบด้วยคลาสสำคัญๆ (รูปที่ 3.10) ดังรายละเอียดต่อไปนี้

- คลาส cli2eq

เป็นคลาสหลัก (Main Class) ของโปรแกรมพรีโพรเซสเซอร์ที่มีหน้าที่ในการ

- รับและจัดเก็บชื่อแฟ้มข้อมูลของโปรแกรมผู้รับบริการและชื่อแฟ้มข้อมูลที่ใช้จัดเก็บโปรแกรมต้นฉบับที่ผ่านการแทรกคำสั่งการทำงานแล้ว

- อ่านโปรแกรมต้นฉบับของผู้รับบริการจากแฟ้มข้อมูลที่ผู้ใช้งานระบุมาในค่าพารามิเตอร์ของโปรแกรมพรีโพรเซสเซอร์
- เรียกใช้งานคลาส JavaParser ในการทำพาร์สซิงโปรแกรมต้นฉบับของผู้รับบริการ
- เรียกใช้งานคลาส TreeFormatter และ TreeDumper ในการจัดรูปแบบของโปรแกรมต้นฉบับของผู้รับบริการใหม่พร้อมทั้งจัดเก็บลงในแฟ้มข้อมูลชั่วคราว
- วิเคราะห์และแทรกคำสั่งการค้นหาบริการแทนที่ลงในโปรแกรมของผู้รับบริการโดยใช้การเรียกไปยังเมทอดต่างๆตามที่กล่าวถึงไปแล้วของบริการค้นหา
- เรียกใช้คลาส util เพื่อทำการเขียนโปรแกรมต้นฉบับที่ผ่านการแทรกคำสั่งแล้วลงในแฟ้มข้อมูลผลลัพธ์
- คำนวณและแสดงเวลาทั้งหมดที่โปรแกรมพรีโพรเซสเซอร์ใช้ในการแทรกคำสั่งการทำงานลงในโปรแกรมของผู้รับบริการ

คลาส cli2eq จะเรียกใช้งานคลาสอื่นๆประกอบกับเมทอดต่างๆภายในคลาสเพื่อทำงานวิเคราะห์และแทรกคำสั่งการทำงานลงในโปรแกรมของผู้รับบริการตามแอ็คทิวิตีไดอะแกรมในรูปที่ 3.9 คลาส cli2eq จะประกอบไปด้วยคุณลักษณะและเมทอดต่างๆที่สำคัญๆดังตารางที่ 3.1 และ 3.2 ตามลำดับ

ตารางที่ 3.1 คุณลักษณะที่สำคัญของคลาส cli2eq

ชื่อคุณลักษณะ (Class Attribute)	**ชนิด** (Type)	**หน้าที่** (Function)
inputfile	Java.lang.String	จัดเก็บค่าพารามิเตอร์อินพุตที่เป็นชื่อแฟ้มข้อมูลของโปรแกรมผู้รับบริการ
outputfile	Java.lang.String	จัดเก็บค่าพารามิเตอร์อินพุตที่เป็นชื่อแฟ้มข้อมูลที่ใช้จัดเก็บโปรแกรมต้นฉบับที่ผ่านการแทรกคำสั่งการทำงานแล้ว
incon	Java.lang.String	จัดเก็บโปรแกรมต้นฉบับของผู้รับบริการที่ได้จากการอ่านแฟ้มข้อมูลที่มีชื่ออยู่ในคุณลักษณะ inputfile
outcon	Java.lang.String	จัดเก็บโปรแกรมต้นฉบับที่ผ่านการแทรกคำสั่งการทำงานโดยโปรแกรมพรีโพรเซสเซอร์แล้ว
afterparse	Java.lang.String	จัดเก็บโปรแกรมต้นฉบับที่ผ่านการตรวจสอบความถูกต้องโดยคลาส JavaParser และผ่านการจัดรูปแบบโดยคลาส TreeFormatter และ TreeDumper แล้ว

ตารางที่ 3.1 คุณลักษณะที่สำคัญของคลาส cli2eq (ต่อ)

importcon	Java.lang.String	จัดเก็บโปรแกรมต้นฉบับของผู้รับบริการเฉพาะส่วนการอิมพอร์ตคลาสไลบรารี
contentcon	Java.lang.String	จัดเก็บโปรแกรมต้นฉบับของผู้รับบริการเฉพาะส่วนรายละเอียดการทำงานของคลาสต่างๆ
classcon	Java.util.Vector	จัดเก็บรายละเอียดของคลาสแต่ละคลาสในโปรแกรมของผู้รับบริการลงเป็นสมาชิกแต่ละเอลิเมนท์ของเวคเตอร์
detailcon	java.util.Vector	จัดเก็บอินสแตนซ์ของคลาส classdetail ซึ่งใช้แทนรายละเอียดเกี่ยวกับคุณลักษณะและเมทอดต่างๆภายในคลาสของโปรแกรมผู้รับบริการลงเป็นสมาชิกแต่ละเอลิเมนท์ของเวคเตอร์
starttime	java.util.Date	จัดเก็บข้อมูลเกี่ยวกับวันเวลาที่โปรแกรมพรีโพรเซสเซอร์เริ่มต้นการประมวลผล
endtime	java.util.Date	จัดเก็บข้อมูลเกี่ยวกับวันเวลาที่โปรแกรมพรีโพรเซสเซอร์สิ้นสุดการประมวลผล
allvarlist	java.util.Vector	จัดเก็บชื่อตัวแปรทั้งหมดที่ถูกประกาศการใช้งานในโปรแกรมของผู้รับบริการลงเป็นสมาชิกแต่ละเอลิเมนท์ของเวคเตอร์
genvarrecord	java.util.Vector	จัดเก็บชื่อตัวแปรทั้งหมดที่โปรแกรมพรีโพรเซสเซอร์สร้างขึ้นสำหรับใช้แทรกลงในโปรแกรมของผู้รับบริการลงเป็นสมาชิกแต่ละเอลิเมนท์ของเวคเตอร์
afterpp	java.lang.String	จัดเก็บรายละเอียดภายในแต่ละเมทอดที่ผ่านการแทรกคำสั่งการทำงานแล้วชั่วคราวก่อนที่จะนำข้อมูลนี้ไปกำหนดค่าลงในตัวแปร detailcon
iorfile	java.lang.String	จัดเก็บชื่อแฟ้มข้อมูลที่ใช้จัดเก็บข้อมูลอ้างอิงของบริการค้นหา ข้อมูลนี้จะถูกใช้งานเมื่อโปรแกรมพรีโพรเซสเซอร์ทำการแทรกคำสั่งให้โปรแกรมของผู้รับบริการเรียกใช้งานเมทอดของบริการค้นหาเพื่อสร้างกลไกการค้นหาบริการแทนที่โดยอัตโนมัติ

ตารางที่ 3.2 เมทอดที่สำคัญของคลาส cli2eq

รูปแบบของเมทอด (Method Signature)	public static void main(String[] args);
หน้าที่ (Function)	วิเคราะห์และแทรกคำสั่งตลอดจนยุติการประมวลผลโปรแกรมพรีโพรเซสเซอร์
การทำงาน (Procedure)	เมทอดนี้จะรับค่าพารามิเตอร์เข้ามาทั้งหมด 2 ค่าเพื่อจัดเก็บลงในคุณลักษณะ inputfile และ outputfile ของคลาส cli2eq เมทอด main จะเรียกใช้งานเมทอดต่างๆภายในคลาส cli2eq ตลอดจนคลาสอื่นเช่นคลาส JavaParser, util, TreeFormatter และ TreeDumper เป็นต้น เพื่อทำงานวิเคราะห์และแทรกคำสั่งตามแอ็คทิวิตีไดอะแกรมในรูปที่ 3.9
รูปแบบของเมทอด (Method Signature)	public static void identify_classinfo(String);
หน้าที่ (Function)	แยกรายละเอียดของคลาสแต่ละคลาสภายในโปรแกรมต้นฉบับของผู้รับบริการออกจากกันและจัดเก็บข้อมูลนั้นลงในเอลิเมนท์ของเวคเตอร์ classcon
การทำงาน (Procedure)	เมทอดนี้จะรับโปรแกรมต้นฉบับของผู้รับบริการเข้ามาทางค่าพารามิเตอร์ด้านอินพุตก่อนที่จะทำการแยกรายละเอียดของแต่ละคลาสภายในโปรแกรมต้นฉบับของผู้รับบริการออกจากกันโดยใช้การค้นหาส่วนการประกาศคลาสตามรูปแบบดังต่อไปนี้ [12] class_declaration ::= {modifier} "class" identifier ["extends" class_name] ["implements" interface_name {"," interface_name}] "{" {field_declaration} "}" modifier ::= "public" | "private" | "protected" | "static" | "final" | "native" | "synchronized" | "abstract" | "transient" class_name ::= identifier | (package_name "." identifier) interface_name ::= identifier | (package_name "." identifier) package_name ::= identifier | (package_name "." identifier) field_declaration ::= (method_declaration | constructor_declaration | variable_declaration) | static_initializer | ";" การค้นหาขอบเขตและแยกคลาสต่างๆออกจากกันนั้นจะใช้การจับที่คำหลัก "class" ที่พิมพ์ไว้ติดกันกับอักขระ space
รูปแบบของเมทอด (Method Signature)	public static void identify_classdetail();
หน้าที่ (Function)	นำรายละเอียดของแต่ละคลาสในโปรแกรมต้นฉบับของผู้รับบริการไปสร้างเป็นอินสแตนซ์ของคลาส classdetail จากนั้นจึงนำทุกๆอินสแตนซ์ที่เมทอดนี้สร้าง

	ขึ้นไปจัดเก็บลงในแต่ละเอลิเมนท์ของเวคเตอร์ detailcon
การทำงาน (Procedure)	นำข้อมูลในเวคเตอร์ classcon มาทำการวิเคราะห์เพื่อหารายละเอียดในส่วนของคุณลักษณะตลอดจนเมทอดต่างๆภายในคลาส จากนั้นจึงทำการสร้างอินสแตนซ์ของคลาส classdetail และจัดเก็บอินสแตนซ์นั้นลงในเวคเตอร์ detailcon โดยเรียกใช้เมทอด addElement ของคลาสเวคเตอร์
รูปแบบของเมทอด (Method Signature)	public static Vector make_vec_attributelist(String);
หน้าที่ (Function)	เมทอดนี้จะรับค่าคุณลักษณะทั้งหมดภายในคลาสหนึ่งๆมาเป็นค่าพารามิเตอร์อินพุต จากนั้นจึงนำมาวิเคราะห์และแยกจัดเก็บรายละเอียดของคุณลักษณะต่างๆภายในคลาสของโปรแกรมผู้รับบริการลงในเวคเตอร์ ซึ่งภายในเอลิเมนท์หนึ่งๆของเวคเตอร์นี้จะใช้จัดเก็บอินสแตนซ์ของคลาส attributelist เพื่อใช้แทนค่าคุณลักษณะ 1 ค่า เมทอดนี้จะส่งกลับเวคเตอร์ผลลัพธ์ไปยังคลาส cli2eq
การทำงาน (Procedure)	แยกคุณลักษณะภายในโปรแกรมต้นฉบับโดยใช้รูปแบบดังนี้ [12] variable_declaration ::= {modifier} type variable_declarator { "," variable_declarator } ";" variable_declarator ::= identifier { "[" "]" } ["=" variable_initializer] variable_initializer ::= expression | ("{" [variable_initializer { "," variable_initializer } [","]] "}") type ::= type_specifier { "[" "]" } type_specifier ::= "boolean" | "byte" | "char" | "short" | "int" | "float" | "long" | "double" | class_name | interface_name
รูปแบบของเมทอด (Method Signature)	public static Vector make_vec_methodlist(String);
หน้าที่ (Function)	เมทอดนี้จะรับรายละเอียดของเมทอดทั้งหมดภายในคลาสหนึ่งๆมาเป็นค่าพารามิเตอร์อินพุต จากนั้นจึงนำมาวิเคราะห์และแยกจัดเก็บรายละเอียดของเมทอดต่างๆภายในคลาสของโปรแกรมผู้รับบริการลงในเวคเตอร์ ซึ่งภายในเอลิเมนท์หนึ่งๆของเวคเตอร์นี้จะใช้จัดเก็บอินสแตนซ์ของคลาส methodlist เพื่อใช้แทนเมทอด 1 เมทอด เมทอดนี้จะส่งกลับเวคเตอร์ผลลัพธ์ไปยังคลาส cli2eq
การทำงาน (Procedure)	แยกเมทอดภายในโปรแกรมต้นฉบับโดยใช้รูปแบบดังนี้ [12] method_declaration ::= {modifier} type identifier "(" [parameter_list] ")" { "[" "]" } (statement_block | ";") parameter_list ::= parameter { "," parameter } parameter ::= type identifier { "[" "]" } statement_block ::= "{" { statement } "}"

รูปแบบของเมทอด (Method Signature)	public static Vector make_vec_paramlist(String);
หน้าที่ (Function)	เมทอดนี้จะรับค่าพารามิเตอร์ทั้งหมดภายในเมทอดหนึ่งๆมาเป็นค่าพารามิเตอร์อินพุต จากนั้นจึงนำมาวิเคราะห์และแยกจัดเก็บรายละเอียดของพารามิเตอร์ต่างๆภายในเมทอดของโปรแกรมผู้รับบริการลงในเวคเตอร์ ซึ่งภายในเอลิเมนท์หนึ่งๆของเวคเตอร์นี้จะใช้จัดเก็บอินสแตนซ์ของคลาส parameterlist เพื่อใช้แทนค่าพารามิเตอร์ 1 ค่า เมทอดนี้จะส่งกลับเวคเตอร์ผลลัพธ์ไปยังคลาส cli2eq
การทำงาน (Procedure)	แยกค่าพารามิเตอร์ต่างๆภายในเมทอดของคลาสโดยใช้รูปแบบดังนี้ [12] parameter_list ::= parameter { "," parameter } parameter ::= type identifier { "[" "]" }
รูปแบบของเมทอด (Method Signature)	public static Vector make_vec_statementlist(String);
หน้าที่ (Function)	เมทอดนี้จะรับรายละเอียดของคำสั่งทั้งหมดภายในเมทอดหนึ่งๆมาเป็นค่าพารามิเตอร์อินพุต จากนั้นจึงนำมาวิเคราะห์และแยกจัดเก็บรายละเอียดของคำสั่ง (Statement) ลงในเวคเตอร์ ซึ่งภายในเอลิเมนท์หนึ่งๆของเวคเตอร์นี้จะใช้จัดเก็บอินสแตนซ์ของคลาส statementlist เพื่อใช้แทนคำสั่ง 1 คำสั่งของเมทอด เมทอดนี้จะส่งกลับเวคเตอร์ผลลัพธ์ไปยังคลาส cli2eq
การทำงาน (Procedure)	แยกคำสั่ง (statement) ต่างๆภายในเมทอดของคลาสโดยใช้อักขระ "\r\n" เป็นตัวแยกแต่ละคำสั่ง (statement) ออกจากกันโดยเรียกใช้คลาส StringTokenizer ของจาวา
รูปแบบของเมทอด (Method Signature)	Public static boolean setmstub(String,String,String);
หน้าที่ (Function)	เมทอดนี้จะรับค่าพารามิเตอร์อินพุตทั้งสิ้น 3 ค่าได้แก่ชื่อของคลาส ชื่อของเมทอดและรายละเอียดของคำสั่งในเมทอดที่ผู้ใช้งานต้องการให้เมทอด setmstub จัดเก็บค่าใหม่นี้ลงในคุณลักษณะ detailcon ของคลาส cli2eq
การทำงาน (Procedure)	เมทอด setmstub จะนำชื่อคลาสและชื่อเมทอดที่รับเข้ามาไปตรวจสอบกับคุณลักษณะ name ในอินสแตนซ์ของคลาส classdetail และคลาส methodlist เมื่อพบค่าที่สอดคล้องกัน เมทอด setmstub จะทำการกำหนดค่าคุณลักษณะ mstub ของอินสแตนซ์ที่มีชนิดเป็น methodlist ให้มีค่าเท่ากับค่าพารามิเตอร์

	อินพุตที่ 3
รูปแบบของเมทอด (Method Signature)	public static void setgenrecord(String,String,int);
หน้าที่ (Function)	เมทอดนี้จะรับค่าพารามิเตอร์อินพุตทั้งสิ้น 3 ค่าได้แก่ชื่อของคลาส ชื่อของเมทอดและเลขลำดับที่ใช้ในการสร้างตัวแปรโดยงานวิจัยนี้กำหนดให้โปรแกรมพรีโพรเซสเซอร์นำชื่อของคลาสและชื่อของเมทอดที่จะถูกแทรกคำสั่งมาร่วมสร้างตัวแปรใหม่เพื่อนำไปใช้ในขั้นตอนของการแทรกคำสั่งการทำงานด้วยคลาส cli2eq จะใช้เมทอด get_newvar_name เพื่อสร้างชื่อตัวแปรใหม่ที่ไม่ซ้ำกันกับชื่อตัวแปรที่มีอยู่แล้วในโปรแกรมของผู้รับบริการโดยใช้รูปแบบของตัวแปรดังนี้ newvariable_name = <classname>_<methodname>_<varid.> เมทอด setgenrecord จะถูกเรียกใช้เพื่อทำหน้าที่จัดเก็บค่า varid ที่เป็นเลขลำดับล่าสุดลงในคุณลักษณะ genvarrecord ของคลาส cli2eq ให้สอดคล้องตามชื่อของคลาสและเมทอด
การทำงาน (Procedure)	ค้นหาเอลิเมนท์ของเวคเตอร์ genvarrecord ซึ่งใช้จัดเก็บค่า varid ที่สอดคล้องกันกับชื่อของคลาสและเมทอดที่ผู้ใช้งานระบุมา จากนั้นจึงทำการกำหนดค่า varid (ค่าพารามิเตอร์ที่ 3) ใหม่ลงในคุณลักษณะ gnum ของคลาส genrecord ในเอลิเมนท์ของเวคเตอร์ genvarrecord
รูปแบบของเมทอด (Method Signature)	public static void analysis_method(String,String,String);
หน้าที่ (Function)	เมทอดนี้จะรับชื่อคลาส ชื่อเมทอดของโปรแกรมผู้รับบริการตลอดจนรายละเอียดของคำสั่งต่างๆภายในเมทอดนั้นมาทำการวิเคราะห์รูปแบบเพื่อนำไปผ่านกระบวนการแทรกคำสั่งการทำงานโดยใช้การเรียกไปยังเมทอด analyse_bind และเมทอด analyse_narrow ต่อไป ภายหลังจากที่เมทอดทั้งหมดของโปรแกรมของผู้รับบริการผ่านการแทรกคำสั่งแล้ว เมทอด analysis_method จะนำโปรแกรมของผู้รับบริการไปจัดเก็บลงในคุณลักษณะ afterpp ของคลาส cli2eq เพื่อรอการนำไปเขียนลงในแฟ้มข้อมูลผลลัพธ์ต่อไป
การทำงาน (Procedure)	เมทอดนี้จะเรียกใช้งานเมทอด analyse_bind และ analyse_narrow ตามลำดับ จากนั้นจึงนำโปรแกรมต้นฉบับที่ผ่านการแทรกคำสั่งแล้วจัดเก็บลงในคุณลักษณะ afterpp ของคลาส cli2eq

รูปแบบของเมทอด (Method Signature)	public static String analyse_bind(String,String,String);
หน้าที่ (Function)	เมทอดนี้จะรับค่าพารามิเตอร์อินพุตทั้งสิ้น 3 ค่าคือชื่อคลาส ชื่อเมทอดตลอดจนรายละเอียดของคำสั่งการทำงานภายในเมทอดของโปรแกรมผู้รับบริการจากเมทอด analysis_method จากนั้นจึงทำการวิเคราะห์คำสั่งการทำงานภายในเมทอดว่ามีการเรียกใช้คำสั่ง bind ของซอฟต์แวร์วิสิโบรคเกอร์ทั้ง 3 รูปแบบหรือไม่ ถ้ามีการเรียกใช้คำสั่ง bind ภายในโปรแกรมของผู้รับบริการแล้ว เมทอดนี้จะทำการแทรกคำสั่งการทำงานให้สอดคล้องกับรูปแบบของคำสั่ง bind ภายในโปรแกรมของผู้รับบริการ ก่อนที่จะส่งกลับเมทอดของโปรแกรมผู้รับบริการนี้กลับไปยังเมทอด analysis_method ต่อไป
การทำงาน (Procedure)	เมทอดนี้จะทำการวิเคราะห์คำสั่งการทำงานภายในเมทอดที่รับมาทางค่าพารามิเตอร์อินพุต ถ้าพบว่าเมทอดที่รับเข้ามามีการเรียกใช้คำสั่ง bind ของซอฟต์แวร์วิสิโบรคเกอร์แล้ว เมทอดนี้จะทำการเรียกใช้ - เมทอด verify_bind_case1 และ insert_bind_case1 เพื่อทำการตรวจสอบและแทรกคำสั่งเมื่อผู้ใช้เรียกคำสั่ง bind รูปแบบที่ 1 - เมทอด verify_bind_case2 และ insert_bind_case2 เพื่อทำการตรวจสอบและแทรกคำสั่งเมื่อผู้ใช้เรียกคำสั่ง bind รูปแบบที่ 2 - เมทอด verify_bind_case3 และ insert_bind_case3 เพื่อทำการตรวจสอบและแทรกคำสั่งเมื่อผู้ใช้เรียกคำสั่ง bind รูปแบบที่ 3 จากนั้นจึงส่งกลับเมทอดของโปรแกรมผู้รับบริการที่ผ่านการแทรกคำสั่งแล้วกลับไปยังเมทอด analysis_method เพื่อใช้งานต่อไป
รูปแบบของเมทอด (Method Signature)	public static String analyse_narrow(String,String,String);
หน้าที่ (Function)	เมทอดนี้จะรับค่าพารามิเตอร์อินพุตทั้งสิ้น 3 ค่าคือชื่อคลาส ชื่อเมทอดตลอดจนรายละเอียดของคำสั่งการทำงานภายในเมทอดของโปรแกรมผู้รับบริการจากเมทอด analysis_method จากนั้นจึงทำการวิเคราะห์คำสั่งการทำงานภายในเมทอดว่ามีการเรียกใช้คำสั่ง narrow หรือไม่ ถ้ามีการเรียกใช้คำสั่ง narrow ภายในโปรแกรมของผู้รับบริการแล้ว เมทอดนี้จะทำการแทรกคำสั่งการทำงานให้โปรแกรมของผู้รับบริการเรียกใช้บริการค้นหาเมื่อเกิดเอ็กซ์เซพชันขึ้นจากการเชื่อมต่อไปยังอินสแตนซ์ของบริการที่ต้องการใช้งาน เมทอด analyse_narrow

	จะส่งกลับเมทอดของโปรแกรมผู้รับบริการที่ผ่านการแทรกคำสั่งแล้วกลับไปยังเมทอด analysis_method ของคลาส cli2eq ต่อไป
การทำงาน (Procedure)	เมทอดนี้จะทำการวิเคราะห์คำสั่งการทำงานภายในเมทอดที่รับมาทางค่าพารามิเตอร์อินพุต ถ้าพบว่าเมทอดที่รับเข้ามามีการเรียกใช้คำสั่ง narrow ของออร์บเพื่อทำการแปลงข้อมูลอ้างอิงของบริการภายในแฟ้มข้อมูล (Persistent IOR) ไปเป็นอินสแตนซ์ของบริการใดๆแล้ว เมทอดนี้จะทำการเรียกใช้เมทอด verify_narrow_case และ insert_narrow_case เพื่อทำการตรวจสอบและแทรกคำสั่งลงในโปรแกรมของผู้รับบริการต่อไป เมทอดนี้จะส่งกลับโปรแกรมของผู้รับบริการที่ผ่านการแทรกคำสั่งแล้วกลับไปยังเมทอด analysis_method เพื่อใช้งานต่อไป
รูปแบบของเมทอด (Method Signature)	public static boolean verify_bind_case1(Vector,boolean);
หน้าที่ (Function)	เมทอดนี้จะรับค่าพารามิเตอร์อินพุตทั้งสิ้น 2 ค่าคือค่า bparam ซึ่งมีชนิดของตัวแปรเป็นเวคเตอร์และค่า usehelper ซึ่งมีชนิดของตัวแปรเป็น boolean เมทอดนี้จะทำหน้าที่ตรวจสอบว่าเมทอด bind ที่พบในโปรแกรมของผู้รับบริการมีค่าพารามิเตอร์ที่ตรงกันกับรูปแบบที่กำหนดไว้ในคำสั่ง bind รูปแบบที่ 1 ของซอฟต์แวร์วิสิโบรคเกอร์หรือไม่ ถ้าตรงกัน เมทอดนี้จะส่งค่าจริง (True) และถ้าไม่ตรง เมทอดนี้จะส่งค่าเท็จ (False) กลับไปยังเมทอด analyse_bind
การทำงาน (Procedure)	เมทอดนี้จะตรวจสอบว่าหากค่า bparam มีสมาชิกทั้งสิ้น 1 จำนวนเป็นชื่อของออร์บและค่า usehelper เป็นจริงจะทำการส่งกลับค่าจริงไปยังเมทอด analyse_bind
รูปแบบของเมทอด (Method Signature)	Public static boolean verify_bind_case2(Vector,boolean);
หน้าที่ (Function)	เมทอดนี้จะรับค่าพารามิเตอร์อินพุตทั้งสิ้น 2 ค่าคือค่า bparam ซึ่งมีชนิดของตัวแปรเป็นเวคเตอร์และค่า usehelper ซึ่งมีชนิดของตัวแปรเป็น boolean เมทอดนี้จะทำหน้าที่ตรวจสอบว่าเมทอด bind ที่พบในโปรแกรมของผู้รับบริการมีค่าพารามิเตอร์ที่ตรงกันกับรูปแบบที่กำหนดไว้ในคำสั่ง bind รูปแบบที่ 2 ของซอฟต์แวร์วิสิโบรคเกอร์หรือไม่ ถ้าตรงกัน เมทอดนี้จะส่งค่าจริงและถ้าไม่ตรง เมทอดนี้จะส่งค่าเท็จกลับไปยังเมทอด analyse_bind
การทำงาน (Procedure)	เมทอดนี้จะตรวจสอบว่าหากค่า bparam มีสมาชิกทั้งสิ้น 2 จำนวนเป็นชื่อของ

	ออร์บและชื่ออินสแตนซ์ของบริการใดๆ (มีชนิดเป็นสายอักขระ) และค่า usehelper เป็นจริงจะทำการส่งกลับค่าจริงไปยังเมทอด analyse_bind
รูปแบบของเมทอด (Method Signature)	public static boolean verify_bind_case3(Vector,boolean);
หน้าที่ (Function)	เมทอดนี้จะรับค่าพารามิเตอร์อินพุตทั้งสิ้น 2 ค่าคือค่า bparam ซึ่งมีชนิดของตัวแปรเป็นเวคเตอร์และค่า usehelper ซึ่งมีชนิดของตัวแปรเป็น boolean เมทอดนี้จะทำหน้าที่ตรวจสอบว่าเมทอด bind ที่พบในโปรแกรมของผู้รับบริการมีค่าพารามิเตอร์ที่ตรงกันกับรูปแบบที่กำหนดไว้ในคำสั่ง bind รูปแบบที่ 3 ของซอฟต์แวร์วิสิโบรคเกอร์หรือไม่ ถ้าตรงกัน เมทอดนี้จะส่งค่าจริงและถ้าไม่ตรงเมทอดนี้จะส่งค่าเท็จกลับไปยังเมทอด analyse_bind
การทำงาน (Procedure)	เมทอดนี้จะตรวจสอบว่าหากค่า bparam มีสมาชิกทั้งสิ้น 4 จำนวนได้แก่ 1) ชื่อของออร์บ 2) ชื่ออินสแตนซ์ของบริการ 3) ชื่อของโฮสต์ที่อินสแตนซ์ของบริการทำงานอยู่ 4) ตัวแปรที่มีชนิดเป็น BindOptions และค่า usehelper เป็นจริงจะทำการส่งกลับค่าจริงไปยังเมทอด analyse_bind
รูปแบบของเมทอด (Method Signature)	public static String insert_bind_case1(String,Vector,Vector, Boolean,String,String,String,String,boolean);
หน้าที่ (Function)	แทรกคำสั่งเรียกใช้บริการค้นหาลงในโปรแกรมของผู้รับบริการตามรูปที่ 3.2 และ 3.4
การทำงาน (Procedure)	- หาค่าไอดีของการจัดเก็บจากคำสั่ง bind - แทรกคำสั่งให้โปรแกรมของผู้รับบริการทำการตรวจสอบค่าพารามิเตอร์ 1 ค่าของคำสั่ง bind ว่ามีชนิดเป็น com.visigenic.vbroker.orb.ObjectImpl หรือไม่ ถ้าใช่จึงทำการแทรกคำสั่งส่วนการตรวจจับเอ็กซ์เซพชันและส่วนการเรียกใช้บริการค้นหาเพื่อสร้างกลไกการทำงานตามซีเควนซ์ไดอะแกรมในรูปที่ 3.1
รูปแบบของเมทอด (Method Signature)	public static String insert_bind_case2(String,Vector,Vector, Boolean,String,String,String,String,boolean);
หน้าที่ (Function)	แทรกคำสั่งเรียกใช้บริการค้นหาลงในโปรแกรมของผู้รับบริการตามรูปที่ 3.2 และ 3.5

การทำงาน (Procedure)	- หาค่าไอดีของการจัดเก็บและชื่ออินสแตนซ์ของบริการจากคำสั่ง bind - แทรกคำสั่งให้โปรแกรมของผู้รับบริการทำการตรวจสอบค่าพารามิเตอร์ ทั้ง 2 ค่าของคำสั่ง bind ว่ามีชนิดเป็น com.visigenic.vbroker.orb.ObjectImpl และ java.lang.String ตามลำดับหรือไม่ ถ้าใช่จึงทำการแทรกคำสั่งส่วนการตรวจจับเอ็กซ์เซพชันและส่วนการเรียกใช้บริการค้นหาเพื่อสร้างกลไกการทำงานตามซีเควนซ์ไดอะแกรมในรูปที่ 3.1
รูปแบบของเมทอด (Method Signature)	public static String insert_bind_case3(String,Vector,Vector,Boolean, String,String,String,String,boolean);
หน้าที่ (Function)	แทรกคำสั่งเรียกใช้บริการค้นหาลงในโปรแกรมของผู้รับบริการตามรูปที่ 3.2 และ 3.6
การทำงาน (Procedure)	- หาค่าไอดีของการจัดเก็บ ชื่ออินสแตนซ์ของบริการ ตลอดจนชื่อโฮสต์ที่บริการนั้นทำงานอยู่จากคำสั่ง bind - แทรกคำสั่งให้โปรแกรมของผู้รับบริการทำการตรวจสอบค่าพารามิเตอร์ ทั้ง 4 ค่าของคำสั่ง bind ว่ามีชนิดเป็น com.visigenic.vbroker.orb.ObjectImpl, java.lang.String, java.lang.String และ org.omg.CORBA.BindOptions ตามลำดับหรือไม่ ถ้าใช่จึงทำการแทรกคำสั่งส่วนการตรวจจับเอ็กซ์เซพชันและส่วนการเรียกใช้บริการค้นหาเพื่อสร้างกลไกการทำงานตามซีเควนซ์ไดอะแกรมในรูปที่ 3.1
รูปแบบของเมทอด (Method Signature)	public static String insert_narrow_case(String,Vector,Vector,Boolean, String,String,String,String,boolean);
หน้าที่ (Function)	แทรกคำสั่งเรียกใช้บริการค้นหาลงในโปรแกรมของผู้รับบริการตามรูปที่ 3.2 และ 3.5
การทำงาน (Procedure)	- หาค่าไอดีของการจัดเก็บจากคำสั่ง narrow - แทรกคำสั่งเพื่อทำการค้นหาชื่ออินสแตนซ์ของบริการจากเมทอด _repository_id ของวัตถุคอร์บา - แทรกคำสั่งให้โปรแกรมของผู้รับบริการทำการตรวจสอบค่าพารามิเตอร์ 1 ค่าของคำสั่ง narrow ว่ามีชนิดเป็น com.visigenic.vbroker.orb.ObjectImpl หรือไม่ ถ้าใช่จึงทำการแทรกคำสั่งส่วนการตรวจจับเอ็กซ์เซพชันและส่วนการเรียกใช้บริการค้นหาเพื่อสร้างกลไกการทำงานตามซีเควนซ์ไดอะแกรมในรูปที่ 3.1

รูปแบบของเมทอด (Method Signature)	public static String get_newvar_name(String, String);
หน้าที่ (Function)	เมทอดนี้จะรับค่าพารามิเตอร์อินพุตทั้งสิ้น 2 ค่าได้แก่ชื่อของคลาสและ ชื่อของเมทอดที่จะถูกแทรกคำสั่งเพื่อนำมาใช้ในการสร้างชื่อของตัวแปรที่ไม่ซ้ำกันกับชื่อของตัวแปรที่มีอยู่เดิมในโปรแกรมของผู้รับบริการ
การทำงาน (Procedure)	เมทอดนี้จะใช้การสร้างชื่อตัวแปรตามรูปแบบดังต่อไปนี้ newvariable_name = <classname>_<methodname>_<varid.> การตรวจสอบว่าชื่อตัวแปรที่ตั้งขึ้นจะไม่ซ้ำกับชื่อตัวแปรที่มีอยู่แล้วนั้นจะใช้การตรวจสอบจากคุณลักษณะ allvarlist และ genvarrecord ของคลาส cli2eq โดยคุณลักษณะ allvarlist จะเป็นตัวแปรเวคเตอร์ที่ใช้จัดเก็บชื่อตัวแปรทั้งหมดในโปรแกรมของผู้รับบริการภายหลังการทำพาร์สซิงโปรแกรมต้นฉบับ ส่วนคุณลักษณะ genvarrecord จะเป็นตัวแปรเวคเตอร์ที่ใช้จัดเก็บชื่อตัวแปรทั้งหมดที่ถูกแทรกลงในโปรแกรมผู้รับบริการโดยโปรแกรมพรีโพรเซสเซอร์

- คลาส util

เป็นคลาสที่ถูกพัฒนาขึ้นเพื่อใช้ประโยชน์ในงานทั่วไป (Utility Class) อาทิเช่นการอ่านโปรแกรมต้นฉบับจากแฟ้มข้อมูลใดๆที่ผู้ใช้งานระบุมา ตลอดจนการอ่านค่าติดตั้งจากแฟ้มข้อมูล config.pp เป็นต้น คลาส util จะถูกเรียกใช้งานโดยคลาสอื่นๆผ่านทางเมทอดต่างๆดังรายละเอียดในตารางที่ 3.3 และ 3.4 ตามลำดับ

ตารางที่ 3.3 คุณลักษณะที่สำคัญของคลาส util

ชื่อคุณลักษณะ (Class Attribute)	**ชนิด** (Type)	**หน้าที่** (Function)
conf_file	java.lang.String	จัดเก็บชื่อแฟ้มข้อมูลที่ใช้กำหนดค่าติดตั้ง (Config File) ของโปรแกรมพรีโพรเซสเซอร์

ตารางที่ 3.4 เมทอดที่สำคัญของคลาส util

รูปแบบของเมทอด (Method Signature)	Public static String get_conf(String);
หน้าที่ (Function)	อ่านและส่งกลับข้อมูลที่กำหนดไว้ในแฟ้มข้อมูล config.pp ตามชื่อแท็ก (Tag) ที่ระบุมาในค่าพารามิเตอร์อินพุต เช่น แท็ก ir_file ใช้สำหรับอ่านชื่อแฟ้มข้อมูลที่ใช้จัดเก็บข้อมูลอ้างอิงของบริการค้นหาเป็นต้น
การทำงาน (Procedure)	เมทอดนี้จะทำการอ่านข้อมูลทั้งหมดที่ถูกกำหนดค่าไว้ในแฟ้มข้อมูล config.pp โดยเรียกใช้เมทอด readf ของคลาส cli2eq จากนั้นจึงนำสายอักขระที่ได้รับมาจากเมทอด readf มาทำการค้นหาข้อมูลตามแท็กที่ผู้ใช้งานระบุมาด้วยเมทอด indexOf ของคลาส String ถ้าพบแท็กที่ต้องการเมทอด get_conf จะส่งกลับข้อมูลจากแท็กนั้นไปยังผู้เรียก แต่ถ้าไม่พบค่าแท็ก เมทอดนี้จะส่งกลับค่า Null
รูปแบบของเมทอด (Method Signature)	public static void write_file(String,String);
หน้าที่ (Function)	เมทอดนี้จะรับค่าพารามิเตอร์อินพุตทั้งสิ้น 2 ค่าคือชื่อแฟ้มข้อมูลและรายละเอียดของข้อมูลที่ผู้ใช้งานต้องการให้เมทอดนี้จัดสร้างขึ้น เมทอด write_file ของงานวิจัยนี้จะถูกเรียกใช้งานเมื่อคลาส cli2eq ต้องการสร้างแฟ้มข้อมูลที่ใช้จัดเก็บโปรแกรมของผู้รับบริการที่ผ่านการแทรกคำสั่งการทำงานแล้ว
การทำงาน (Procedure)	เมทอดนี้จะเรียกใช้เมทอด write ของคลาส FileWriter เพื่อสร้างและเขียนข้อมูลที่ผู้ใช้งานระบุมาลงในแฟ้มข้อมูลที่กำหนด
รูปแบบของเมทอด (Method Signature)	public static String cutcomment(String);
หน้าที่ (Function)	เมทอดนี้จะรับโปรแกรมต้นฉบับของผู้รับบริการเข้ามาเป็นค่าพารามิเตอร์อินพุตเพื่อทำการตัดรายละเอียดเกี่ยวกับคอมเมนต์ออกไปจากโปรแกรมต้นฉบับก่อนที่จะทำการส่งกลับโปรแกรมของผู้รับบริการที่ปราศจากคอมเมนต์แล้วกลับไปยังคลาส cli2eq
การทำงาน (Procedure)	เมทอดนี้จะตัดรายละเอียดเกี่ยวกับคอมเมนต์ภายในโปรแกรมของผู้รับบริการที่อยู่ภายหลังเครื่องหมาย // หรืออยู่ภายในขอบเขตของเครื่องหมาย /* */ ทิ้งไป จากนั้นจึงทำการส่งกลับโปรแกรมส่วนที่เหลือไปยัง

	เมทอดอื่นที่เรียกใช้เมทอดนี้
รูปแบบของเมทอด (Method Signature)	public static String extractimportline(String);
หน้าที่ (Function)	เมทอดนี้จะรับโปรแกรมต้นฉบับของผู้รับบริการเข้ามาทำการวิเคราะห์และส่งกลับเฉพาะรายละเอียดส่วนการอิมพอร์ตคลาสไลบรารีและส่วนประกาศแพ็คเกจของคลาสกลับไปยังผู้เรียกใช้เมทอด
การทำงาน (Procedure)	เมทอดนี้จะเรียกใช้เมทอด indexOf ของคลาส String เพื่อทำการค้นหาคำหลัก import และ package จากโปรแกรมผู้รับบริการ เมื่อพบคำหลักที่ค้นหา เมทอด extractimportline จะรวบรวมรายละเอียดของคำสั่ง import และ package เพื่อส่งกลับไปยังผู้เรียกใช้เมทอด
รูปแบบของเมทอด (Method Signature)	Public static String removeimportline(String);
หน้าที่ (Function)	เมทอดนี้จะรับโปรแกรมต้นฉบับของผู้รับบริการเข้ามาทำการวิเคราะห์และส่งกลับเฉพาะรายละเอียดส่วนการประกาศคลาสโดยแยกตัดรายละเอียดส่วนการอิมพอร์ตคลาสไลบรารีและส่วนการประกาศแพ็คเกจของคลาสทิ้งไป
การทำงาน (Procedure)	เมทอดนี้จะเรียกใช้เมทอด indexOf ของคลาส String เพื่อทำการค้นหาคำหลัก import และ package จากโปรแกรมผู้รับบริการ เมื่อพบคำหลักที่ค้นหาแล้วเมทอด removeimportline จะตัดรายละเอียดของคำสั่ง import และ package ทิ้งไปโดยจะส่งเฉพาะรายละเอียดส่วนการประกาศคลาสกลับไปยังเมทอดที่เรียกใช้เมทอดนี้
รูปแบบของเมทอด (Method Signature)	public static String get_repid(String,String);
หน้าที่ (Function)	เมทอดนี้จะรับชื่อโมดูลและชื่อส่วนต่อประสานของบริการใดๆเข้ามาเป็นค่าพารามิเตอร์ด้านอินพุตจากนั้นจึงทำการสร้างและส่งกลับค่าไอดีของการจัดเก็บไปยังผู้เรียกใช้เมทอด
การทำงาน (Procedure)	สร้างและส่งกลับค่าไอดีของการจัดเก็บตามรูปแบบดังต่อไปนี้ "IDL:"+ชื่อโมดูล+"/"+ชื่อส่วนต่อประสาน+":1.0" เช่น "IDL:module1/interface1:1.0"

รูปแบบของเมทอด (Method Signature)	public static String readf();
หน้าที่ (Function)	อ่านและส่งกลับข้อมูลทั้งหมดที่ถูกกำหนดค่าไว้ในแฟ้มข้อมูล config.pp
การทำงาน (Procedure)	เมทอดนี้จะทำการตรวจสอบว่าแฟ้มข้อมูล config.pp มีอยู่จริงภายในระบบแฟ้มข้อมูล (File System) โดยเรียกใช้เมทอด exists ของคลาส File จากนั้นจึงทำการอ่านข้อมูลทั้งหมดจากแฟ้มข้อมูลโดยเรียกใช้เมทอด readLine ของคลาส BufferedReader ท้ายสุดเมทอด readf จะส่งกลับข้อมูลทั้งหมดที่อ่านได้กลับไปยังผู้เรียกในรูปของสายอักขระ
รูปแบบของเมทอด (Method Signature)	Public static String read_ior(String);
หน้าที่ (Function)	อ่านและส่งกลับข้อมูลอ้างอิงของบริการจากแฟ้มข้อมูลที่ผู้ใช้ระบุชื่อมาในค่าพารามิเตอร์อินพุต
การทำงาน (Procedure)	เมทอดนี้จะทำการตรวจสอบว่าแฟ้มข้อมูลที่ผู้ใช้งานระบุมามีอยู่จริงภายในระบบแฟ้มข้อมูล (File System) โดยเรียกใช้เมทอด exists ของคลาส File จากนั้นจึงทำการอ่านข้อมูลอ้างอิงของบริการจากแฟ้มข้อมูลโดยเรียกใช้เมทอด readLine ของคลาส BufferedReader ท้ายสุดเมทอด read_ior จะส่งกลับข้อมูลอ้างอิงของบริการกลับไปยังเมทอดที่เรียกใช้เมทอดนี้ในรูปของสายอักขระ

- คลาส JavaParser

เป็นคลาสที่ถูกสร้างขึ้นจากโปรแกรมเจทีบีและจาวาซีซีสำหรับใช้ในการทำพาร์สซิงโปรแกรมต้นฉบับของผู้รับบริการที่ถูกพัฒนาด้วยภาษาจาวา การพาร์สซิงโปรแกรมต้นฉบับของผู้รับบริการจะกระทำผ่านทางเมทอด CompilationUnit ของคลาสนี้ ดังรายละเอียดต่อไปนี้

ตารางที่ 3.5 เมทอดที่สำคัญของคลาส JavaParser

รูปแบบของเมทอด (Method Signature)	public static final CompilationUnit CompilationUnit() throws ParseException
หน้าที่ (Function)	ตรวจสอบความถูกต้องของคำสั่งการทำงานในโปรแกรมผู้รับบริการว่ามีรูปแบบของคำสั่ง (Syntax) สอดคล้องตรงตามหลักไวยากรณ์ของภาษาจาวา (Java Grammar) หรือไม่ ถ้าคลาส JavaParser พบว่าโปรแกรมผู้รับบริการมีรูปแบบที่ไม่ถูกต้อง เมทอด CompilationUnit จะทำให้เกิดเอ็กซ์เซพชันชนิด ParserException ขึ้น (แจ้งเตือนรายละเอียดของความผิดพลาด) แต่ถ้ามีรูปแบบที่ถูกต้องตามไวยากรณ์แล้ว เมทอด CompilationUnit จะส่งกลับโหนดราก (Root Node) ของซินแท็กซ์ทรี (Syntax Tree) นั้นกลับไปยังผู้เรียกใช้เมทอด
การทำงาน (Procedure)	ทำการพาร์สซิงโปรแกรมต้นฉบับของผู้รับบริการและจัดสร้างซินแท็กซ์ทรีเพื่อใช้แทนส่วนประกอบต่างๆของโปรแกรม เมทอดนี้จะส่งโหนดรากกลับไปยังคลาส cli2eq เพื่อใช้ในการจัดรูปแบบของคำสั่งใหม่โดยใช้คลาส TreeFormatter และ TreeDumper

- คลาส TreeFormatter

เป็นคลาสที่สืบทอดคุณสมบัติมาจากคลาส DepthFirstVisitor โดยคลาสนี้จะถูกสร้างขึ้นจากโปรแกรมเจทีบีและจาวาซีซี งานวิจัยนี้ใช้คลาส TreeFormatter ในการจัดรูปแบบโปรแกรมของผู้รับบริการใหม่ให้ง่ายต่อการวิเคราะห์ส่วนประกอบโดยเมทอดในคลาส cli2eq ต่อไป อาทิเช่นถ้าโปรแกรมผู้รับบริการถูกพัฒนาขึ้นเป็น

```
public        static
void main     (String []     args){
<classbody>...}
```

การจัดรูปแบบโปรแกรมผู้รับบริการใหม่โดยคลาส TreeFormatter จะอยู่ในรูปของ

```
public static void main(String[] args)
{
  <classbody>...
}
```

ภายหลังจากที่คลาส JavaParser ทำการพาร์สซิงโปรแกรมผู้รับบริการเสร็จเรียบร้อยแล้ว คลาส JavaParser จะจัดสร้างซินแท็กซ์ทรีที่ใช้แทนส่วนประกอบต่างๆของโปรแกรมผู้รับบริการขึ้นในหน่วย

ความจำ โดยงานวิจัยนี้จะใช้หลักการของวิสิเตอร์แพทเทิร์น (Visitor Pattern) ในการท่องซินแท็กซ์ทรีเพื่อจัดรูปแบบของโปรแกรมต้นฉบับใหม่ด้วยคำสั่งดังนี้

```
syntaxtree.Node root = parser.CompilationUnit();
root.accept(new visitor.TreeFormatter(3,0));
```

การจัดรูปแบบของโปรแกรมต้นฉบับโดยคลาส TreeFormatter จะถูกใช้งานร่วมกับคลาส TreeDumper ในการเขียนข้อมูลจากโหนดของซินแท็กซ์ทรีภายในหน่วยความจำลงในแฟ้มข้อมูลชั่วคราว

- คลาส TreeDumper

เป็นคลาสที่ได้รับการสืบทอดมาจากคลาส DepthFirstVisitor โดยคลาสนี้ถูกสร้างขึ้นจากโปรแกรมเจทีบีและจาวาซีซีเช่นเดียวกัน งานวิจัยนี้กำหนดให้ใช้งานคลาส TreeDumper ร่วมกับคลาส TreeFormatter โดยใช้คำสั่งภายในโปรแกรมพรีโพรเซสเซอร์ดังนี้

```
syntaxtree.Node root = parser.CompilationUnit();
afterparse=inputfile.trim()+".tran";
final visitor.TreeDumper dumper = new visitor.TreeDumper(new
                                        java.io.FileOutputStream(afterparse));
root.accept(new visitor.TreeFormatter(3,0));
dumper.resetPosition();
root.accept(dumper);
```

งานวิจัยนี้จะใช้คลาส TreeDumper ในการท่องโหนดของซินแท็กซ์ทรีที่ผ่านการจัดรูปแบบโดยคลาส TreeFormatter แล้วภายในหน่วยความจำเพื่อเขียน (Dump) โปรแกรมของผู้รับบริการนั้นลงในแฟ้มข้อมูลชั่วคราว โปรแกรมพรีโพรเซสเซอร์จะใช้ชื่อแฟ้มข้อมูลชั่วคราวเป็นชื่อเดียวกันกับชื่อแฟ้มข้อมูลที่ผู้ใช้งานระบุมาในค่าพารามิเตอร์อินพุตแต่จะกำหนดให้แฟ้มข้อมูลชั่วคราวนี้มีนามสกุลเป็น .tran

- คลาส classinfo

เป็นคลาสที่ออกแบบขึ้นเพื่อใช้จัดเก็บชื่อคลาส รายละเอียดภายในคลาส ตลอดจนลำดับของคลาสที่ปรากฏภายในโปรแกรมของผู้รับบริการ (ตารางที่ 3.6) คลาส cli2eq ในโปรแกรมพรีโพรเซสเซอร์จะทำการวิเคราะห์และแยกจัดเก็บรายละเอียดของคลาสต่างๆลงในอินสแตนซ์ของคลาส classinfo จากนั้นจึงรวบรวมอินสแตนซ์ของคลาส classinfo ทั้งหมดไปจัดเก็บเป็นสมาชิกภายในเอลิเมนท์ของเวคเตอร์ classcon ในคลาส cli2eq เพื่อรอการวิเคราะห์รายละเอียดของคลาสต่างๆต่อไป

ตารางที่ 3.6 คุณลักษณะที่สำคัญของคลาส classinfo

ชื่อคุณลักษณะ (Class Attribute)	ชนิด (Type)	หน้าที่ (Function)
name	java.lang.String	จัดเก็บชื่อคลาสที่ใช้อินสแตนซ์ของคลาสนี้เก็บข้อมูล
order	int	จัดเก็บลำดับของคลาสที่ปรากฏอยู่ในโปรแกรมต้นฉบับของผู้รับบริการ
content	java.lang.String	จัดเก็บรายละเอียดภายในคลาสที่ใช้อินสแตนซ์ของคลาสนี้เก็บข้อมูล
head	java.lang.String	จัดเก็บข้อมูลส่วนการประกาศคลาสที่อยู่ก่อนหน้าชื่อของคลาสเช่น คำหลัก abstract, final, public, private หรือ protected
tail	java.lang.String	จัดเก็บข้อมูลส่วนการประกาศคลาสที่อยู่ภายหลังชื่อของคลาสเช่น คำหลัก extends หรือ implements

- คลาส classdetail

ภายหลังจากที่คลาส cli2eq เรียกใช้เมทอด identify_classinfo เพื่อทำการแยกรายละเอียดและจัดเก็บคลาสต่างๆในโปรแกรมของผู้รับบริการลงในเวคเตอร์ classcon แล้วคลาส cli2eq จะเรียกเมทอด identify_classdetail เพื่อนำสมาชิกของเวคเตอร์ classcon (ซึ่งเป็นอินสแตนซ์ของคลาส classinfo) มาผ่านการวิเคราะห์หาคุณลักษณะและเมทอดภายในคลาสและจัดเก็บลงในอินสแตนซ์ของคลาส classdetail อีกครั้งหนึ่ง

คลาส classdetail ถูกออกแบบขึ้นเพื่อใช้จัดเก็บรายละเอียดเกี่ยวกับชื่อคลาส คุณลักษณะของคลาสตลอดจนเมทอดต่างๆภายในคลาสของโปรแกรมผู้รับบริการลงในเวคเตอร์ detailcon เพื่อรอการแทรกคำสั่งโดยคลาส cli2eq ต่อไป (ตารางที่ 3.7)

ตารางที่ 3.7 คุณลักษณะที่สำคัญของคลาส classdetail

ชื่อคุณลักษณะ (Class Attribute)	ชนิด (Type)	หน้าที่ (Function)
name	java.lang.String	จัดเก็บชื่อคลาสที่ใช้อินสแตนซ์ของคลาสนี้เก็บข้อมูล
alist	java.util.Vector	เป็นเวคเตอร์ที่ใช้จัดเก็บอินสแตนซ์ของคลาส attributelist ซึ่งแต่ละอินสแตนซ์จะใช้แทนแต่ละคุณลักษณะของคลาส
mlist	java.util.Vector	เป็นเวคเตอร์ที่ใช้จัดเก็บอินสแตนซ์ของคลาส methodlist ซึ่งแต่ละอินสแตนซ์จะใช้แทนแต่ละเมทอดภายในคลาส

ตารางที่ 3.7 คุณลักษณะที่สำคัญของคลาส classdetail (ต่อ)

astub	Java.lang.String	จัดเก็บคุณลักษณะทั้งหมดภายในคลาสที่ใช้อินสแตนซ์ของคลาสนี้เก็บข้อมูล
head	Java.lang.String	จัดเก็บข้อมูลส่วนการประกาศคลาสที่อยู่ก่อนหน้าชื่อของคลาสเช่น คำหลัก abstract, final, public, private หรือ protected
tail	Java.lang.String	จัดเก็บข้อมูลส่วนการประกาศคลาสที่อยู่ภายหลังชื่อของคลาสเช่น คำหลัก extends หรือ implements

- คลาส attributelist

เป็นคลาสที่ถูกออกแบบขึ้นเพื่อใช้จัดเก็บข้อมูลเกี่ยวกับคุณลักษณะต่างๆของคลาส (ตารางที่ 3.8)

ตารางที่ 3.8 คุณลักษณะที่สำคัญของคลาส attributelist

ชื่อคุณลักษณะ (Class Attribute)	**ชนิด (Type)**	**หน้าที่ (Function)**
name	Java.lang.String	จัดเก็บชื่อของคุณลักษณะ
type	Java.lang.String	จัดเก็บชนิดของคุณลักษณะ

- คลาส methodlist

เป็นคลาสที่ถูกออกแบบขึ้นเพื่อใช้จัดเก็บข้อมูลเกี่ยวกับเมทอดต่างๆภายในคลาส (ตารางที่ 3.9)

ตารางที่ 3.9 คุณลักษณะที่สำคัญของคลาส methodlist

ชื่อคุณลักษณะ (Class Attribute)	**ชนิด (Type)**	**หน้าที่ (Function)**
head	java.lang.String	จัดเก็บข้อมูลที่อยู่ก่อนหน้าการประกาศเมทอดเช่นคำหลัก public, private, protected, static, synchronized เป็นต้น
name	java.lang.String	จัดเก็บชื่อของเมทอด
returntype	java.lang.String	จัดเก็บชนิดของค่าส่งกลับของเมทอด

ตารางที่ 3.9 คุณลักษณะที่สำคัญของคลาส methodlist (ต่อ)

slist	java.util.Vector	เป็นเวคเตอร์ที่ใช้จัดเก็บอินสแตนซ์ของคลาส statementlist ซึ่งแต่ละอินสแตนซ์จะใช้แทนแต่ละคำสั่ง (Statement) ภายในเมทอด
plist	java.util.Vector	เป็นเวคเตอร์ที่ใช้จัดเก็บอินสแตนซ์ของคลาส parameterlist ซึ่งแต่ละอินสแตนซ์จะใช้แทนแต่ละค่าพารามิเตอร์ของเมทอด
tail	java.lang.String	จัดเก็บข้อมูลที่อยู่ภายหลังการประกาศเมทอดเช่นคำหลัก throw exception
mstub	java.lang.String	จัดเก็บรายละเอียดของเมทอดใดๆที่ใช้อินสแตนซ์ของคลาสนี้เก็บข้อมูล

- คลาส parameterlist

เป็นคลาสที่ถูกออกแบบขึ้นเพื่อใช้จัดเก็บข้อมูลเกี่ยวกับพารามิเตอร์ต่างๆภายในเมทอด (ตารางที่ 3.10)

ตารางที่ 3.10 คุณลักษณะที่สำคัญของคลาส parameterlist

ชื่อคุณลักษณะ (Class Attribute)	**ชนิด (Type)**	**หน้าที่ (Function)**
name	java.lang.String	จัดเก็บชื่อของพารามิเตอร์
type	java.lang.String	จัดเก็บชนิดของพารามิเตอร์
head	java.lang.String	จัดเก็บข้อมูลที่อยู่ก่อนหน้าชนิดของพารามิเตอร์เช่นคำหลัก final

- คลาส statementlist

เป็นคลาสที่ถูกออกแบบขึ้นเพื่อใช้จัดเก็บข้อมูลเกี่ยวกับคำสั่ง (Statement) ต่างๆภายในเมทอด (ตารางที่ 3.11)

ตารางที่ 3.11 คุณลักษณะที่สำคัญของคลาส statementlist

ชื่อคุณลักษณะ (Class Attribute)	**ชนิด (Type)**	**หน้าที่ (Function)**
number	int	จัดเก็บเลขลำดับของคำสั่งนี้ในเมทอด
statement	java.lang.String	จัดเก็บรายละเอียดของคำสั่ง
space	java.lang.String	จัดเก็บอักขระ space ที่อยู่ข้างหน้าคำสั่ง

3.5 การใช้งานโปรแกรมพรีโพรเซสเซอร์

เมื่อผู้พัฒนาโปรแกรมประยุกต์พัฒนาโปรแกรมของผู้รับบริการให้เรียกใช้งานบริการต่างๆของคอร์บาเสร็จเรียบร้อยแล้ว ผู้พัฒนาโปรแกรมจะสามารถเรียกใช้โปรแกรมพรีโพรเซสเซอร์ให้ทำการแทรกคำสั่งเพื่อใช้สร้างกลไกการค้นหาบริการแทนที่ลงในโปรแกรมผู้รับบริการได้ โปรแกรมของผู้รับบริการที่ถูกแทรกคำสั่งแล้วจะมีความสามารถในการทำงานเพิ่มขึ้นในแง่ของความสามารถในการเรียกใช้งานบริการอื่นที่ทำงานได้เท่าเทียมกันเพื่อทดแทนบริการที่ผู้ใช้งานต้องการแต่ยังไม่พร้อมให้บริการได้ในขณะนั้น ผู้ให้บริการแทนที่จะสามารถควบคุมรูปแบบของการทำงานแทนที่กันของบริการโดยกำหนดที่อินสแตนซ์ของตัวดำเนินการแปลงที่จะถูกเรียกใช้งานโดยผู้รับบริการ (ระบุข้อมูลดังกล่าวลงภายในแฟ้มข้อมูล config.eq) การเรียกใช้งานโปรแกรมพรีโพรเซสเซอร์ของงานวิจัยนี้เป็นดังต่อไปนี้

```
java cli2eq c:\thesis\client\client1.java –o c:\thesis\client\eqclient.java
```

รูปที่ 3.11 ตัวอย่างการเรียกใช้โปรแกรมพรีโพรเซสเซอร์

จากตัวอย่างในรูปที่ 3.11 โปรแกรมพรีโพรเซสเซอร์จะทำการแทรกคำสั่งลงในโปรแกรมต้นฉบับที่จัดเก็บอยู่ภายในแฟ้มข้อมูล client1.java และจัดสร้างแฟ้มข้อมูล eqclient1.java ขึ้นใหม่เพื่อใช้จัดเก็บโปรแกรมต้นฉบับที่ผ่านการแทรกคำสั่งการทำงานแล้ว จากนั้นโปรแกรมพรีโพรเซสเซอร์จะแสดงข้อมูลผลการเรียกใช้โปรแกรมตามรูปต่อไปนี้

รูปที่ 3.12 ผลการเรียกใช้โปรแกรมพรีโพรเซสเซอร์

สมมติให้โปรแกรมต้นฉบับของผู้รับบริการมีรายละเอียดของคำสั่งส่วนการค้นหาและเรียกใช้บริการต่างๆของคอร์บาด้วยวิธีการเรียกใช้คำสั่ง bind ทั้ง 3 รูปแบบรวมถึงการใช้วิธีการอ่านและแปลงข้อมูลอ้างอิงของบริการที่ถูกจัดเก็บอยู่ภายในแฟ้มข้อมูล (Persistent IOR) ไปเป็นอินสแตนซ์ของบริการที่ผู้ใช้ต้องการ ตามตัวอย่างโปรแกรมในรูปที่ 3.13

```
(1) import java.util.*;
(2) import java.io.*;
(3) import java.net.*;
(4) import service_by_id.*;
(5) public class client1 {
(6)   public static void main  (String[] args)
(7)   {
(8)     try{
(9)          org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args,null);
(10)         org.omg.CORBA.BOA boa = orb.BOA_init();
(11)         System.out.println("<<Bind - Case1>>");
(12)         service_by_id.account_by_id  ser1=service_by_id.account_by_idHelper.bind(orb);
(13)         String ret=String.valueOf(ser1.getbalance(1));
(14)         System.out.println("    Invoke Service -> ID : 1  have saving balance = "+ret);
(15)         System.out.println("<<Bind - Case 2>>");
(16)         String objname="ser1";
(17)         ser1=service_by_id.account_by_idHelper.bind(orb,objname);
(18)         ret=String.valueOf(ser1.getbalance(1));
(19)         System.out.println("    Invoke Service -> ID : 1  have saving balance = "+ret);
(20)         System.out.println("<<Bind - Case 3>>");
(21)         String host = "155.7.1.2";
(22)         org.omg.CORBA.BindOptions bindopt = new org.omg.CORBA.BindOptions();
(23)         ser1=service_by_id.account_by_idHelper.bind(orb,objname,host,bindopt);
(24)         ret=String.valueOf(ser1.getbalance(1));
(25)         System.out.println("    Invoke Service -> ID : 1  have saving balance = "+ret);
(26)         System.out.println("<<Using Persistence IOR>>");
(27)         String ior = util.read_ior("c:\\thesis\\ior\\ser1.ior");
```

รูปที่ 3.13 ตัวอย่างโปรแกรมของผู้รับบริการก่อนการแทรกคำสั่ง

```
(28)        org.omg.CORBA.Object x = orb.string_to_object(ior);
(29)        ser1 = service_by_id.account_by_idHelper.narrow(orb.string_to_object(ior));
(30)        ret=String.valueOf(ser1.getbalance(1));
(31)        System.out.println("    Invoke Service -> ID : 1  have saving balance = "+ret);
(32)    }
 (33)   catch(Exception ex){
(34)        ex.printStackTrace();
(35)        System.exit(1);
(36)    }
(37) }
(38)}
```

รูปที่ 3.13 ตัวอย่างโปรแกรมของผู้รับบริการก่อนการแทรกคำสั่ง (ต่อ)

คำอธิบายของโปรแกรมตัวอย่างในรูปที่ 3.13

บรรทัดที่ 1-4 เป็นคำสั่งส่วนการอิมพอร์ตคลาสไลบรารีที่ใช้ในโปรแกรมของผู้รับบริการโดยภายในคลาสไลบรารี service_by_id จะประกอบไปด้วยคลาสสตับ (Client Stub) ซึ่งถูกสร้างขึ้นโดยโปรแกรม idl2java ของซอฟต์แวร์วิสิโบรคเกอร์เพื่อใช้งานในขั้นตอนของการสร้างการเชื่อมต่อ (Binding) ระหว่างโปรแกรมของผู้รับบริการและโปรแกรมของผู้ให้บริการ

บรรทัดที่ 5 เป็นคำสั่งส่วนการประกาศคลาสของโปรแกรมผู้รับบริการ ซึ่งในตัวอย่างนี้กำหนดให้โปรแกรมของผู้รับบริการมีชื่อคลาสเป็น client1

บรรทัดที่ 6 เป็นคำสั่งส่วนการประกาศเมทอดหลัก (Main Method) ของคลาส client1 โดยคำสั่งต่างๆภายในเมทอดนี้จะทำงานเมื่อผู้รับบริการเรียกใช้คลาส client1 ผ่านทางจาวาเวอร์ชวลแมชชีน

บรรทัดที่ 9-10 เป็นคำสั่งส่วนการติดตั้งค่าโดยปริยายให้กับออร์บ (ORB) และออบเจ็กต์อะแดปเตอร์ (Object Adapter) ของคอร์บา

บรรทัดที่ 12 เป็นตัวอย่างคำสั่งส่วนการค้นหาข้อมูลอ้างอิงของบริการใดๆที่มีชื่อส่วนต่อประสานเป็น ::service_by_id::account_by_id โปรแกรมของผู้รับบริการจะเรียกใช้คำสั่ง bind ในรูปแบบที่ 1 จากแฟ้มข้อมูลตัวช่วยของคลาสไลบรารี service_by_id ที่คลาส client1 อิมพอร์ตเข้ามา การเรียกใช้คำสั่ง bind จะส่งผลให้ออร์บของคอร์บาสร้างการเชื่อมต่อและค้นหาข้อมูลอ้างอิงของบริการที่ต้องการจากสมาร์ทเอเจนท์ ซึ่งถ้าออร์บค้นพบบริการใดๆในสมาร์ทเอเจนท์ที่มีชื่อส่วนต่อประสานตามที่ผู้ใช้งานระบุมาในคำสั่ง bind ออร์บในฝั่งของโปรแกรมผู้รับบริการจะสร้างการเชื่อมต่อระหว่างอินสแตนซ์ของ

บริการนั้นกับวัตถุพร็อกซีท้องถิ่นที่ออร์บสร้างขึ้นพร้อมจัดส่งข้อมูลอ้างอิงของบริการที่พบกลับไปยังตัวแปร ser1 ตามคำสั่งในบรรทัดที่ 12 โปรแกรมของผู้รับบริการจะสามารถเรียกใช้เมทอด getbalance ของบริการ ser1 ได้ตามคำสั่งในบรรทัดที่ 13 แต่ถ้าหากออร์บไม่พบบริการใดเลยในสมาร์ทเอเจนท์ที่มีชื่อของส่วนต่อประสานตามที่ผู้ใช้งานระบุมาแล้ว ออร์บจะทำให้โปรแกรมของผู้รับบริการเกิดเอ็กซ์เซพชันขึ้นเมื่อโปรแกรมผู้รับบริการประมวลผลคำสั่งในบรรทัดที่ 12

บรรทัดที่ 17 เป็นตัวอย่างคำสั่งส่วนการค้นหาข้อมูลอ้างอิงของบริการที่มีชื่อส่วนต่อประสานเป็น ::service_by_id::account_by_id และมีชื่ออินสแตนซ์ของบริการเป็น ser1 ตามค่าในตัวแปร objname โปรแกรมผู้รับบริการจะเรียกใช้คำสั่ง bind ในรูปแบบที่ 2 จากแฟ้มข้อมูลตัวช่วยของคลาสไลบรารี service_by_id เพื่อให้ออร์บสร้างการเชื่อมต่อระหว่างอินสแตนซ์ของบริการที่พบกับวัตถุพร็อกซีท้องถิ่นที่ออร์บสร้างขึ้นพร้อมทั้งส่งกลับข้อมูลอ้างอิงของบริการที่พบกลับไปใช้งานในคำสั่งการเรียกใช้บริการในบรรทัดที่ 18 ต่อไป

บรรทัดที่ 23 เป็นตัวอย่างคำสั่งส่วนการค้นหาข้อมูลอ้างอิงของบริการที่มีชื่อส่วนต่อประสานเป็น ::service_by_id::account_by_id และมีชื่ออินสแตนซ์ของบริการเป็น ser1 นอกจากนั้นแล้วผู้รับบริการยังกำหนดชื่อโฮสต์ที่บริการนั้นทำงานอยู่โดยให้มีค่าเลขที่อยู่ไอพีเป็น 155.7.1.2 อีกด้วย โปรแกรมของผู้รับบริการจะเรียกใช้คำสั่ง bind ในรูปแบบที่ 3 โดยทำการระบุค่าตัวเลือกที่ใช้ในการ bind (Bind Option) ให้มีค่าตามค่าโดยปริยาย (Default) ของซอฟต์แวร์วิสิโบรคเกอร์

คำสั่ง bind ในบรรทัดที่ 23 จะส่งผลให้ออร์บของโปรแกรมผู้รับบริการส่งกลับข้อมูลอ้างอิงของบริการตามเงื่อนไขที่ผู้ใช้งานต้องการกลับไปทำงานในคำสั่งส่วนการเรียกใช้บริการ (บรรทัดที่ 24) ต่อไป

บรรทัดที่ 27-29 เป็นตัวอย่างคำสั่งส่วนการเรียกใช้บริการ ser1 โดยใช้การแปลงข้อมูลอ้างอิงของบริการจากแฟ้มข้อมูล (Persistent IOR) ไปเป็นวัตถุคอร์บาด้วยการเรียกใช้คำสั่ง string_to_object ของออร์บ ก่อนที่ผู้รับบริการจะทำการเรียกใช้เมทอด narrow จากแฟ้มข้อมูลตัวช่วยในคลาสไลบรารี service_by_id เพื่อทำการกำหนดชนิดของบริการ (Cast Service Type) ให้เป็น service_by_id.account_by_id ตามคำสั่งในบรรทัดที่ 29

บรรทัดที่ 8 และ 33 เป็นคำสั่ง try และ catch ของภาษาจาวา ซึ่งถูกใช้ในการดักจับเอ็กซ์เซพชันต่างๆที่เกิดขึ้นในโปรแกรมของผู้รับบริการ เมื่อคำสั่งใดๆในขอบเขตของคำสั่ง try (คำสั่งระหว่างบรรทัดที่ 9-31) เกิดข้อผิดพลาดในการทำงานขึ้น คำสั่งที่อยู่ในขอบเขตของคำสั่ง catch (บรรทัดที่ 34-35) จะถูกเรียกใช้งานโดยอัตโนมัติ ในตัวอย่างนี้จะใช้เมทอด printStackTrace ของคลาส Exception ในการพิมพ์ข้อความแจ้งเตือนความผิดพลาดที่เกิดขึ้นบนหน้าจอ ส่วนคำสั่ง System.exit ในบรรทัดที่ 35 จะใช้ยุติการทำงานของโปรแกรมผู้รับบริการ

ถ้าผู้รับบริการทำการคอมไพล์และเรียกใช้โปรแกรม client1 ตามรูปแบบปกติ โปรแกรมของผู้รับบริการจะแสดงข้อความแจ้งเตือนความผิดพลาดพร้อมทั้งหยุดการทำงานลงในทันทีที่ออร์บค้นพบความ

ผิดพลาดเมื่อทำงานในคำสั่ง bind ทั้ง 3 รูปแบบหรือเมื่อโปรแกรมของผู้รับบริการพยายามที่จะเรียกใช้บริการที่ได้รับข้อมูลอ้างอิงของบริการจากแฟ้มข้อมูลแต่อินสแตนซ์ของบริการนั้นไม่ได้ทำงานอยู่จริงในขณะเรียกใช้งาน

ผู้พัฒนาโปรแกรมของผู้รับบริการจะสามารถเพิ่มเติมคุณสมบัติในการเรียกใช้งานบริการอื่นที่ทำงานได้เท่าเทียมกันกับบริการที่ต้องการเพื่อใช้งานแทนที่เมื่อเกิดเอ็กซ์เซพชันต่างๆดังที่กล่าวถึงมาแล้วได้โดยเรียกใช้โปรแกรมพรีโพรเซสเซอร์เพื่อทำการแทรกคำสั่งการทำงานลงในแฟ้มข้อมูล client1.java โปรแกรมพรีโพรเซสเซอร์จะแทรกคำสั่งลงในโปรแกรมต้นฉบับของผู้รับบริการแล้วทำการจัดเก็บลงในแฟ้มข้อมูล eqclient1.java (รูปที่ 3.14)

```
(1) import java.util.*;
(2) import java.io.*;
(3) import java.net.*;
(4) import service_by_id.*;
(5) public class client1
(6) {
(7)    public static void main(String[] args)
(8)    {
(9)       try
(10)     {
(11)        org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args, null);
(12)        org.omg.CORBA.BOA boa = orb.BOA_init();
(13)        System.out.println("<<Bind - Case1>>");
(14)        service_by_id.account_by_id ser1= null;
(15)        try
(16)        {
(17)           if(orb.getClass().getName().equals("com.visigenic.vbroker.orb.ORB"))
(18)           {
(19)            try
(20)            {
```

รูปที่ 3.14 ตัวอย่างโปรแกรมของผู้รับบริการที่ผ่านการแทรกคำสั่งแล้ว

```
(21)          ser1 = service_by_id.account_by_idHelper.bind(orb);
(22)        }
(23)        catch(org.omg.CORBA.NO_IMPLEMENT gv_client1_main_1)
(24)        {
(25)          BufferedReader gv_client1_main_2 = new BufferedReader
                                                  (new FileReader("c:\\thesis\\ior\\irserv.ior"));
(26)          String gv_client1_main_3 = gv_client1_main_2.readLine();
(27)          org.omg.CORBA.ORB gv_client1_main_4 = org.omg.CORBA.ORB.init();
(28)          EQIR.extend_ir gv_client1_main_5 = EQIR.extend_irHelper.narrow
                                        (gv_client1_main_4.string_to_object(gv_client1_main_3));
(29)          String gv_client1_main_6 = gv_client1_main_5.search_equ
                                                    ("IDL:service_by_id/account_by_id:1.0");
(30)          if(!gv_client1_main_6.trim().equals(""))
(31)            ser1 = service_by_id.account_by_idHelper.narrow
                                      (gv_client1_main_4.string_to_object(gv_client1_main_6));
(32)          else
(33)          {
(34)            System.out.println("No any equivalent service.");
(35)            System.exit(1);
(36)          }
(37)        }
(38)      }
(39)      else
(40)      {
(41)        ser1 = service_by_id.account_by_idHelper.bind(orb);
(42)      }
(43)    }
(44)    catch(Exception gv_client1_main_7)
(45)    {
(46)      gv_client1_main_7.printStackTrace();
(47)      System.exit(1);
```

รูปที่ 3.14 ตัวอย่างโปรแกรมของผู้รับบริการที่ผ่านการแทรกคำสั่งแล้ว (ต่อ)

```
(48)    }
(49)    String ret = String.valueOf(ser1.getbalance(1));
(50)    System.out.println("    Invoke Service -> ID : 1  have saving balance = " + ret);
(51)    System.out.println("<<Bind - Case 2>>");
(52)    String objname = "ser1";
(53)    try
(54)    {
(55)      if((orb.getClass().getName().equals("com.visigenic.vbroker.orb.ORB"))
            &&(objname.getClass().getName().equals("java.lang.String")))
(56)      {
(57)        try
(58)        {
(59)          ser1 = service_by_id.account_by_idHelper.bind(orb,objname);
(60)        }
(61)        catch(org.omg.CORBA.NO_IMPLEMENT gv_client1_main_8)
(62)        {
(63)          BufferedReader gv_client1_main_9 = new BufferedReader
                                                  (new FileReader("c:\\thesis\\ior\\irserv.ior"));
(64)          String gv_client1_main_10 = gv_client1_main_9.readLine();
(65)          org.omg.CORBA.ORB gv_client1_main_11 = org.omg.CORBA.ORB.init();
(66)          EQIR.extend_ir gv_client1_main_12 = EQIR.extend_irHelper.narrow
(67)                                    (gv_client1_main_11.string_to_object (gv_client1_main_10));
(68)          String gv_client1_main_13 = gv_client1_main_12.search_equ_obj
                                                  ("IDL:service_by_id/account_by_id:1.0",objname);
(69)          if(!gv_client1_main_13.trim().equals(""))
(70)            ser1 = service_by_id.account_by_idHelper.narrow
                                    (gv_client1_main_11.string_to_object(gv_client1_main_13));
(71)          else
(72)          {
(73)            System.out.println("No any equivalent service.");
(74)            System.exit(1);
(75)          }
```

รูปที่ 3.14 ตัวอย่างโปรแกรมของผู้รับบริการที่ผ่านการแทรกคำสั่งแล้ว (ต่อ)

```
(76)          }
(77)        }
(78)        else
(79)        {
(80)          ser1 = service_by_id.account_by_idHelper.bind(orb,objname);
(81)        }
(82)      }
(83)      catch(Exception gv_client1_main_14)
(84)      {
(85)        gv_client1_main_14.printStackTrace();
(86)        System.exit(1);
(87)      }
(88)      ret = String.valueOf(ser1.getbalance(1));
(89)      System.out.println("    Invoke Service -> ID : 1  have saving balance = " + ret);
(90)      System.out.println("<<Bind - Case 3>>");
(91)      String host = "155.7.1.2";
(92)      org.omg.CORBA.BindOptions bindopt = new org.omg.CORBA.BindOptions();
(93)      try
(94)      {
(95)        if((orb.getClass().getName().equals("com.visigenic.vbroker.orb.ORB"))
(96)            &&(objname.getClass().getName().equals("java.lang.String"))
(97)            &&(host.getClass().getName().equals("java.lang.String"))
(98)            &&(bindopt.getClass().getName().equals("org.omg.CORBA.BindOptions")))
(99)        {
(100)         try
(101)         {
(102)           ser1 = service_by_id.account_by_idHelper.bind(orb,objname,host,bindopt);
(103)           boolean gv_client1_main_15 = bindopt.defer_bind;
(104)           if(gv_client1_main_15)
(105)           {
(106)             if(ser1._non_existent())
(107)             {
```

รูปที่ 3.14 ตัวอย่างโปรแกรมของผู้รับบริการที่ผ่านการแทรกคำสั่งแล้ว (ต่อ)

```
(108)            BufferedReader gv_client1_main_17 = new BufferedReader
                                                    (new FileReader("c:\\thesis\\ior\\irserv.ior"));
(109)            String gv_client1_main_18 = gv_client1_main_17.readLine();
(110)            org.omg.CORBA.ORB gv_client1_main_19 = org.omg.CORBA.ORB.init();
(111)            EQIR.extend_ir gv_client1_main_20 = EQIR.extend_irHelper.narrow
                                    (gv_client1_main_19.string_to_object(gv_client1_main_18));
(112)            String gv_client1_main_21 = gv_client1_main_20.search_equ_host
                                    ("IDL:service_by_id/account_by_id:1.0",objname,host);
(113)             if(!gv_client1_main_21.trim().equals(""))
(114)               ser1 = service_by_id.account_by_idHelper.narrow
                                (gv_client1_main_19.string_to_object(gv_client1_main_21));
(115)             else
(116)             {
(117)               System.out.println("No any equivalent service.");
(118)               System.exit(1);
(119)             }
(120)           }
(121)         }
(122)       }
(123)       catch(org.omg.CORBA.NO_IMPLEMENT gv_client1_main_16)
(124)       {
(125)          BufferedReader gv_client1_main_17 = new BufferedReader
                                                    (new FileReader("c:\\thesis\\ior\\irserv.ior"));
(126)          String gv_client1_main_18 = gv_client1_main_17.readLine();
(127)          org.omg.CORBA.ORB gv_client1_main_19 = org.omg.CORBA.ORB.init();
(128)          EQIR.extend_ir gv_client1_main_20 = EQIR.extend_irHelper.narrow
                                    (gv_client1_main_19.string_to_object(gv_client1_main_18));
(129)          String gv_client1_main_21 = gv_client1_main_20.search_equ_host
                                    ("IDL:service_by_id/account_by_id:1.0",objname,host);
(130)          if(!gv_client1_main_21.trim().equals(""))
(131)            ser1 = service_by_id.account_by_idHelper.narrow
                                (gv_client1_main_19.string_to_object(gv_client1_main_21));
```

รูปที่ 3.14 ตัวอย่างโปรแกรมของผู้รับบริการที่ผ่านการแทรกคำสั่งแล้ว (ต่อ)

```
(132)           else
(133)           {
(134)             System.out.println("No any equivalent service.");
(135)             System.exit(1);
(136)           }
(137)         }
(138)       }
(139)       else
(140)       {
(141)         ser1 = service_by_id.account_by_idHelper.bind(orb,objname,host,bindopt);
(142)       }
(143)     }
(144)     catch(Exception gv_client1_main_22)
(145)     {
(146)       gv_client1_main_22.printStackTrace();
(147)       System.exit(1);
(148)     }
(149)     ret = String.valueOf(ser1.getbalance(1));
(150)     System.out.println("    Invoke Service -> ID : 1  have saving balance = " + ret);
(151)     System.out.println("<<Using Persistence IOR>>");
(152)     String ior = util.read_ior("c:\\thesis\\ior\\ser1.ior");
(153)     org.omg.CORBA.Object x = orb.string_to_object(ior);
(154)     try
(155)     {
(156)       ser1 = service_by_id.account_by_idHelper.narrow(orb.string_to_object(ior));
(157)       if(ser1._non_existent())
(158)       {
(159)         BufferedReader gv_client1_main_24 = new BufferedReader
                                                  (new FileReader("c:\\thesis\\ior\\irserv.ior"));
(160)         String gv_client1_main_25 = gv_client1_main_24.readLine();
(161)         org.omg.CORBA.ORB gv_client1_main_26 = org.omg.CORBA.ORB.init();
```

รูปที่ 3.14 ตัวอย่างโปรแกรมของผู้รับบริการที่ผ่านการแทรกคำสั่งแล้ว (ต่อ)

```
(162)          EQIR.extend_ir gv_client1_main_27 = EQIR.extend_irHelper.narrow
                                        (gv_client1_main_26.string_to_object(gv_client1_main_25));
(163)          String gv_client1_main_29 = ser1._repository_id();
(164)          String gv_client1_main_30 = ser1._object_name();
(165)          String gv_client1_main_28 = gv_client1_main_27.search_equ_obj
                                                    (gv_client1_main_29,gv_client1_main_30);
(166)          if(!gv_client1_main_28.trim().equals(""))
(167)            ser1 = service_by_id.account_by_idHelper.narrow
                                        (gv_client1_main_26.string_to_object(gv_client1_main_28));
(168)          else
(169)          {
(170)             System.out.println("No any equivalent service.");
(171)             System.exit(1);
(172)          }
(173)       }
(174)    }
(175)    catch(Exception gv_client1_main_23)
(176)    {
(177)       gv_client1_main_23.printStackTrace();
(178)       System.exit(1);
(179)    }
(180)    ret = String.valueOf(ser1.getbalance(1));
(181)    System.out.println("    Invoke Service -> ID : 1  have saving balance = " + ret);
(182) }
(183) catch (Exception ex)
(184) {
(185)    ex.printStackTrace();
(186)    System.exit(1);
(187) }
(188)}
```

รูปที่ 3.14 ตัวอย่างโปรแกรมของผู้รับบริการที่ผ่านการแทรกคำสั่งแล้ว (ต่อ)

คำอธิบายโปรแกรมตัวอย่างในรูปที่ 3.14

บรรทัดที่ 1-6 จะมีคำสั่งการทำงานเหมือนเดิม ทั้งนี้โปรแกรมพรีโพรเซสเซอร์จะไม่ทำการเปลี่ยนแปลงใดๆในคำสั่งส่วนการอิมพอร์ตคลาสไลบรารี คำสั่งการประกาศคลาส ตลอดจนคำสั่งการประกาศเมทอดหลักในโปรแกรมผู้รับบริการ

บรรทัดที่ 14-48 จะเป็นกลุ่มคำสั่งที่ถูกแทรกลงในโปรแกรมของผู้รับบริการเพื่อใช้สร้างกลไกการค้นหาบริการแทนที่จากคลังจัดเก็บส่วนต่อประสานที่ได้รับการเพิ่มขยาย เมื่อโปรแกรมผู้รับบริการเรียกใช้คำสั่ง bind ในรูปแบบที่ 1 โปรแกรมพรีโพรเซสเซอร์จะเริ่มแทรกคำสั่งเพื่อใช้ตรวจสอบตัวแปรที่เป็นพารามิเตอร์ของคำสั่ง bind ว่ามีชนิดเป็น com.visigenic.vbroker.orb.ORB หรือไม่ ในบรรทัดที่ 17 ถ้าไม่ใช่โปรแกรมของผู้รับบริการจะทำงานตามคำสั่งในบรรทัดที่ 41 ไปตามปกติ แต่ถ้าใช่แสดงว่าคำสั่ง bind ในโปรแกรมของผู้รับบริการนั้นเป็นคำสั่งที่ผู้รับบริการทำการค้นหาข้อมูลอ้างอิงของบริการจากสมาร์ทเอเจนท์จริง ดังนั้นโปรแกรมพรีโพรเซสเซอร์จะทำการแทรกคำสั่ง try และ catch ซึ่งใช้ในการดักจับเอ็กซ์เซพชันที่อาจจะเกิดขึ้นเมื่อออร์บในฝั่งของโปรแกรมผู้รับบริการไม่สามารถค้นหาข้อมูลอ้างอิงของบริการได้ตามปกติ การดักจับเอ็กซ์เซพชันที่เกิดขึ้นนี้จะเป็นไปตามการแทรกคำสั่ง catch ในบรรทัดที่ 23

คำสั่งในบรรทัดที่ 25-36 จะถูกเรียกใช้งานเมื่อโปรแกรมของผู้รับบริการเกิดเอ็กซ์เซพชันขึ้นในขณะรันไทม์ โปรแกรมพรีโพรเซสเซอร์จะแทรกคำสั่งให้โปรแกรมผู้รับบริการเรียกใช้เมทอด search_equ ของบริการค้นหาโดยเริ่มคำสั่งจากการสร้างการเชื่อมต่อไปยังอินสแตนซ์ของบริการค้นหาตามคำสั่งในบรรทัดที่ 25-28 จากนั้นจึงทำการเรียกใช้เมทอด search_equ ของบริการค้นหาในบรรทัดที่ 29 โดยระบุค่าไอดีของการจัดเก็บเป็น IDL:service_by_id/account_by_id:1.0 ซึ่งจะสอดคล้องกับรูปแบบการเรียกใช้คำสั่ง bind รูปแบบที่ 1 ข้อมูลอ้างอิงของตัวดำเนินการแปลงที่บริการค้นหาส่งกลับมาจากเมทอด search_equ จะถูกจัดเก็บลงในตัวแปร gv_client1_main_6 ในบรรทัดที่ 29 ซึ่งโปรแกรมพรีโพรเซสเซอร์จะแทรกคำสั่งตรวจสอบผลการค้นหานี้ว่ามีบริการอื่นๆที่ผู้รับบริการจะสามารถเรียกใช้งานแทนที่บริการที่ต้องการได้หรือไม่ในบรรทัดที่ 30 ถ้าค่าในตัวแปร gv_client1_main_6 ไม่เท่ากับ Null โปรแกรมของผู้รับบริการจะทำการแปลงชนิดและจัดเก็บข้อมูลอ้างอิงของตัวดำเนินการแปลงลงในตัวแปร ser1 ตามคำสั่งในบรรทัดที่ 31 เพื่อรอการเรียกใช้งานต่อไปแต่ถ้าบริการค้นหาไม่พบบริการใดๆที่สามารถใช้งานแทนที่ได้เลย บริการค้นหาจะส่งกลับค่า Null มาที่ตัวแปร gv_client1_main_6 ซึ่งจะส่งผลให้โปรแกรมของผู้รับบริการแสดงข้อความแจ้งเตือนว่า “No any equivalent service” และหยุดการทำงานลงในบรรทัดที่ 35

อนึ่งเมื่อได้รับข้อมูลอ้างอิงของตัวดำเนินการแปลงมาใช้งานแทนที่บริการที่ต้องการแล้ว โปรแกรมของผู้รับบริการจะสามารถเรียกใช้งานตัวดำเนินการแปลงเพื่อทำงานตามกลไกในซีเควนซ์ไดอะแกรมที่ 3.1 ได้ตามคำสั่งในบรรทัดที่ 49 งานวิจัยนี้จะมีการกำหนดรูปแบบของชื่อตัวแปรที่ถูกสร้างขึ้นเพื่อใช้แทรกลงในโปรแกรมผู้รับบริการอย่างชัดเจนกล่าวคือโปรแกรมพรีโพรเซสเซอร์จะนำชื่อคลาสและ

ชื่อเมทอดที่จะแทรกคำสั่งมาประกอบกับเลขลำดับการแทรกคำสั่งเพื่อกำหนดเป็นชื่อตัวแปร อาทิเช่นตัวแปรที่ถูกแทรกลงในเมทอด main ของคลาส client1 ค่าที่ 1 จะถูกตั้งชื่อเป็น gv_client1_main_1 เป็นต้น

บรรทัดที่ 53-87 เป็นกลุ่มคำสั่งที่ถูกแทรกลงในโปรแกรมของผู้รับบริการเพื่อใช้สร้างกลไกการค้นหาบริการแทนที่จากคลังจัดเก็บส่วนต่อประสานที่เพิ่มขยาย เมื่อโปรแกรมผู้รับบริการเรียกใช้คำสั่ง bind รูปแบบที่ 2 ความแตกต่างของการแทรกคำสั่งระหว่างกรณีที่ผู้รับบริการเรียกคำสั่ง bind แบบที่ 1 และแบบที่ 2 อยู่ที่การตรวจสอบค่าตัวแปรที่เป็นพารามิเตอร์ของคำสั่ง bind โดยสำหรับรูปแบบที่ 2 จะต้องตรวจสอบชนิดของพารามิเตอร์ทั้ง 2 ค่าโดยกำหนดให้พารามิเตอร์ที่ 1 จะต้องมีชื่อชนิดเป็น com.visigenic.vbroker.orb.ORB ส่วนพารามิเตอร์ที่ 2 จะต้องมีชื่อชนิดเป็น java.lang.String เท่านั้น โปรแกรมของผู้รับบริการจึงจะถูกเรียกทำงานในคำสั่งตามบรรทัดที่ 63-76 เพื่อทำการค้นหาบริการแทนที่จากคลังจัดเก็บส่วนต่อประสานโดยเรียกไปยังเมทอด search_equ_obj ของบริการค้นหาแทนการเรียกใช้เมทอด search_equ ตามคำสั่งในบรรทัดที่ 68

บรรทัดที่ 93-148 เป็นกลุ่มคำสั่งที่ถูกแทรกลงในโปรแกรมของผู้รับบริการเพื่อใช้สร้างกลไกการค้นหาบริการแทนที่จากคลังจัดเก็บส่วนต่อประสานที่ได้รับการเพิ่มขยาย เมื่อโปรแกรมผู้รับบริการเรียกใช้คำสั่ง bind รูปแบบที่ 3 การแทรกคำสั่งโดยโปรแกรมพรีโพรเซสเซอร์เมื่อผู้รับบริการเรียกใช้คำสั่ง bind รูปแบบที่ 3 จะแตกต่างกับเมื่อใช้คำสั่ง bind ทั้ง 2 แบบแรกตรงที่การแทรกคำสั่งส่วนการตรวจสอบชนิดของพารามิเตอร์ของคำสั่ง bind แบบที่ 3 จะต้องทำการตรวจสอบพารามิเตอร์ทั้งสิ้น 4 ค่าตามคำสั่งในบรรทัดที่ 95-98 จากนั้นโปรแกรมพรีโพรเซสเซอร์จึงจะแทรกคำสั่งเพื่อใช้ตรวจสอบตัวเลือกของการ bind ว่ามีค่า defer_bind เป็นค่าจริงหรือไม่ (บรรทัดที่ 103-104) การตรวจสอบการทำงานในส่วนนี้มีความจำเป็นจะต้องดำเนินการทั้งนี้เนื่องจากเมื่อใดก็ตามที่โปรแกรมผู้รับบริการกำหนดค่า defer_bind เป็นจริงในช่วงของการ bind แบบที่ 3 แล้วจะส่งผลให้ออร์บของคอร์บาทำการส่งกลับข้อมูลอ้างอิงของบริการที่พบในสมาร์ทเอเจนท์กลับไปยังโปรแกรมผู้รับบริการในทันทีโดยไม่ได้ทำการสร้างการเชื่อมต่อไปยังอินสแตนซ์ของบริการนั้นจริงๆ (การเชื่อมต่อจะเกิดขึ้นเมื่อมีการเรียกใช้บริการครั้งแรก) นั่นคือออร์บจะยังไม่ได้ตรวจสอบว่าบริการนั้นทำงานอยู่ ณ ขณะนั้นจริง ดังนั้นถ้าหาก defer_bind เป็นจริงโปรแกรมพรีโพรเซสเซอร์จะทำการแทรกคำสั่งให้ผู้รับบริการตรวจสอบการทำงานอินสแตนซ์ของบริการที่ได้รับข้อมูลอ้างอิงของบริการผ่านทางคำสั่ง bind ว่าทำงานอยู่ ณ ขณะนั้นหรือไม่ในเครือข่ายโดยโปรแกรมของผู้รับบริการจะเรียกไปยังเมทอด _non_existent ในบรรทัดที่ 106 การเรียกใช้เมทอด _non_existent จะส่งผลให้ออร์บพยายามสร้างการเชื่อมต่อระหว่างโปรแกรมผู้รับบริการไปยังอินสแตนซ์ของบริการในทันที (ไม่ต้องรอไปจนถึงเวลาที่โปรแกรมผู้รับบริการเรียกใช้งานเมทอดแรกของบริการนั้นๆ) ถ้าเมทอด _non_existent ส่งกลับค่าจริงมายังโปรแกรมผู้รับบริการจะแสดงว่าบริการนั้นๆไม่ได้ทำงานอยู่ในขณะเรียกใช้งาน ดังนั้นโปรแกรมพรีโพรเซสเซอร์จะแทรกคำสั่งส่วนการเรียกใช้เมทอด search_equ_host ของบริการค้นหาตามรูปแบบของคำสั่งในบรรทัดที่ 108-120 คำสั่งในบรรทัดที่ 124-136 ซึ่งอยู่ภายในขอบเขตของคำสั่ง catch

จะถูกเรียกใช้งานเช่นเดียวกันเมื่อโปรแกรมของผู้รับบริการเกิดเอ็กซ์เซพชันจากการทำคำสั่ง bind ในบรรทัดที่ 102 ซึ่งในที่นี้โปรแกรมพรีโพรเซสเซอร์จะแทรกคำสั่งให้โปรแกรมผู้รับบริการเรียกใช้เมทอด search_equ_host ของบริการค้นหาเช่นเดียวกับคำสั่งในบรรทัดที่ 112

บรรทัดที่ 157-174 เป็นกลุ่มคำสั่งที่ถูกแทรกลงในโปรแกรมของผู้รับบริการเพื่อใช้สร้างกลไกการค้นหาบริการแทนที่จากคลังจัดเก็บส่วนต่อประสานที่เพิ่มขยาย เมื่อโปรแกรมผู้รับบริการเรียกใช้งานบริการใดๆที่ได้รับข้อมูลอ้างอิงของบริการจากแฟ้มข้อมูล (Persistent IOR) โปรแกรมพรีโพรเซสเซอร์จะแทรกคำสั่งให้ผู้รับบริการเรียกใช้เมทอด _non_existent ในบรรทัดที่ 157 เพื่อทำการตรวจสอบการทำงานอินสแตนซ์ของบริการที่จะใช้งานว่าทำงานอยู่จริงภายในเครือข่ายก่อนที่จะทำการเรียกใช้บริการนั้นๆได้ คำสั่งในบรรทัดที่ 159-173 จะถูกเรียกใช้งานเมื่อออร์บในฝั่งของโปรแกรมผู้รับบริการไม่พบอินสแตนซ์ของบริการที่ต้องการใช้งานทั้งนี้เมทอด _non_existent จะส่งกลับค่าจริงกลับมายังโปรแกรมของผู้รับบริการเพื่อให้ผู้รับบริการเรียกใช้เมทอด search_equ_obj ของบริการค้นหาตามคำสั่งในบรรทัดที่ 165 ได้