บทที่ 3
การออกแบบคลังชนิดของบริการ ที่รองรับความสัมพันธ์แบบเท่าเทียมกัน

ดังที่กล่าวไปแล้วในบทที่ 2 ว่าส่วนต่อประสานของบริการเป็นส่วนหนึ่งของข้อมูลชนิดของบริการซึ่งอาร์เอ็ม-โอดีพีกำหนดรวมไว้ในคลังชนิดของบริการ แต่เนื่องจากแต่ละสถาปัตยกรรมมีความแตกต่างในการจัดเก็บส่วนต่อประสานของบริการ ดังเช่นในสถาปัตยกรรมแบบคอร์บา จะทำการเก็บข้อมูลส่วนต่อประสานของบริการไว้ที่องค์ประกอบที่ชื่อว่าคลังส่วนต่อประสาน ดังนั้นวิทยานิพนธ์นี้ซึ่งใช้คอร์บาเป็นสถาปัตยกรรมต้นแบบ จะทำการเพิ่มเติมข้อกำหนดของคลังส่วนต่อประสานดังกล่าว ให้สามารถรองรับข้อมูลความสัมพันธ์แบบเท่าเทียมกันได้ด้วย อย่างไรก็ดีข้อกำหนดที่เพิ่มเติมนี้จะสามารถนำไปประยุกต์ให้เข้ากับคลังชนิดของบริการในสถาปัตยกรรมอื่นๆ ได้

3.1 ความสัมพันธ์ในลักษณะการบรรจุของข้อมูลภายในคลังชนิดของบริการที่รองรับความสัมพันธ์แบบเท่าเทียมกัน

คลังชนิดของบริการ (หรือคลังส่วนต่อประสานของคอร์บาในวิทยานิพนธ์นี้) ซึ่งรองรับความสัมพันธ์แบบเท่าเทียมกันนั้น จะเป็นส่วนเพิ่มเติมจากคลังส่วนต่อประสานเดิม โดยจะเก็บเฉพาะข้อมูลเพิ่มเติมที่แสดงการทดแทนกันได้ในระหว่างส่วนต่อประสาน

ที่มาของข้อมูลความสัมพันธ์แบบเท่าเทียมกันนี้ มาจากการนำคลังส่วนต่อประสานดังกล่าวไปใช้งาน โดยการออกแบบข้อมูลต่างๆที่จะมีได้ในคลังส่วนต่อประสานนี้ ได้คำนึงถึงการนำข้อมูลนั้นไปใช้เป็นหลัก ดังรูปที่ 3.1 โดยจะต้องสามารถรองรับการนำไปใช้ดังต่อไปนี้

1. การนำข้อมูลความสัมพันธ์แบบเท่าเทียมกันไปใช้โดยโปรแกรมสร้างตัวดำเนินการแปลง (Mapping-Operator-Generating Program) ในงานวิจัย [1] ซึ่งข้อมูลที่โปรแกรมสร้างตัวดำเนินการแปลงต้องการใช้ได้แก่
   - ระดับของการแปลง (Evolve Level) เป็นข้อมูลที่ระบุว่าการแปลงที่เกิดขึ้นเป็นการแปลงในระดับชนิดหรือระดับอินแสตนซ์
   - ชื่อวัตถุต้นทาง (Source Object Name) ชื่อผู้ให้บริการต้นทาง (Source Server Name) และชื่อเครื่องแม่ข่ายต้นทาง (Source Host Name) ซึ่งข้อมูลทั้งสามนี้จะใช้ในการระบุถึงวัตถุของส่วนต่อประสานรุ่นเก่าที่ต้องการทำการแปลง
   - ชื่อวัตถุปลายทาง (Destination Object Name) ชื่อผู้ให้บริการปลายทาง (Destination Server Name) และชื่อเครื่องแม่ข่ายปลายทาง (Destination Host

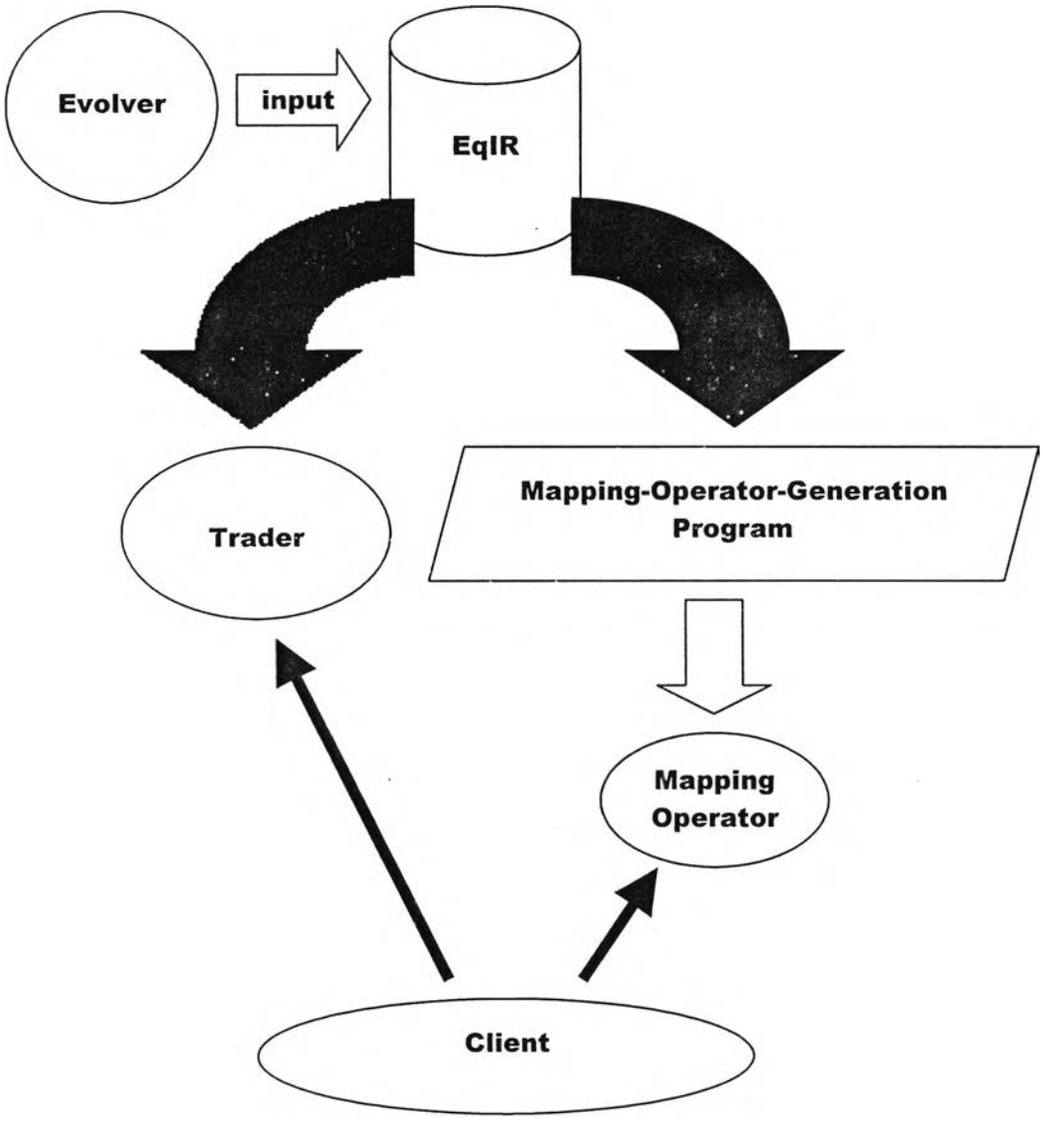
Name) ซึ่งข้อมูลทั้งสามนี้ จะใช้ในการระบุถึงวัตถุของส่วนต่อประสานรุ่นใหม่ที่มีความสัมพันธ์แบบเท่าเทียมกันกับวัตถุของส่วนต่อประสานรุ่นเก่า

- โหมดการสร้าง (Generating Mode) เป็นข้อมูลที่ระบุโหมดในการสร้างฟังก์ชันการแปลง ซึ่งมีได้ 2 รูปแบบคือ direct และ user defined [1] (ดูหัวข้อ 3.2.1)
- โหมดการแปลง (Map Mode) เป็นข้อมูลที่ระบุลักษณะของฟังก์ชันการแปลงว่าเป็นฟังก์ชันที่ถูกสร้างขึ้นเพื่อแปลงองค์ประกอบใดในส่วนต่อประสานที่เปลี่ยนไป (ดูหัวข้อ 3.2.1)
- แฟ้มฟังก์ชันการแปลง (Mapping Function File) เป็นข้อมูลที่ระบุวิธีการแปลงที่จะกระทำโดยตัวดำเนินการแปลง โดยแฟ้มฟังก์ชันการแปลงจะถูกแปล (Compile) เป็นส่วนหนึ่งของตัวดำเนินการแปลงนั้นๆ เพื่อทำหน้าที่แปลงคำร้องสำหรับวัตถุของส่วนต่อประสานต้นทางให้กับวัตถุของส่วนต่อประสานปลายทาง [1] โดยปกติแล้วตัวดำเนินการแปลงที่ถูกสร้างขึ้นมานั้น จะต้องมีความรู้หลักๆ 2 ส่วนคือ รู้จักวัตถุต้นทางและปลายทาง ที่มันจะเป็นตัวกลาง และรู้จักว่ามันต้องใช้ฟังก์ชันการแปลงใดบ้าง และใช้เมื่อใด ดังนั้นผู้ทำการเปลี่ยนรุ่นจะต้องระบุรายละเอียดของวัตถุต้นทางและปลายทางของการแปลงให้ชัดเจน พร้อมทั้งรายละเอียดของฟังก์ชันการแปลงแบบผู้ใช้กำหนด (user-defined) ทั้งหมด ได้แก่ ระดับการแปลง โหมดการสร้าง โหมดการแปลง และวิธีการแปลง ดังข้างต้น

2. การนำข้อมูลความสัมพันธ์แบบเท่าเทียมกันไปใช้สำหรับเทรดเดอร์ เทรดเดอร์อาจมีความต้องการเรียกใช้งานคลังส่วนต่อประสานที่รองรับความสัมพันธ์แบบเท่าเทียมกันเพื่อใช้ในการค้นหาบริการที่เท่าเทียมกับบริการที่มีผู้รับบริการร้องขอมา ในกรณีที่ไม่สามารถหาบริการที่ถูกร้องขอมาได้โดยตรง โดยข้อมูลที่เทรดเดอร์ใช้ในการพิจารณาได้แก่

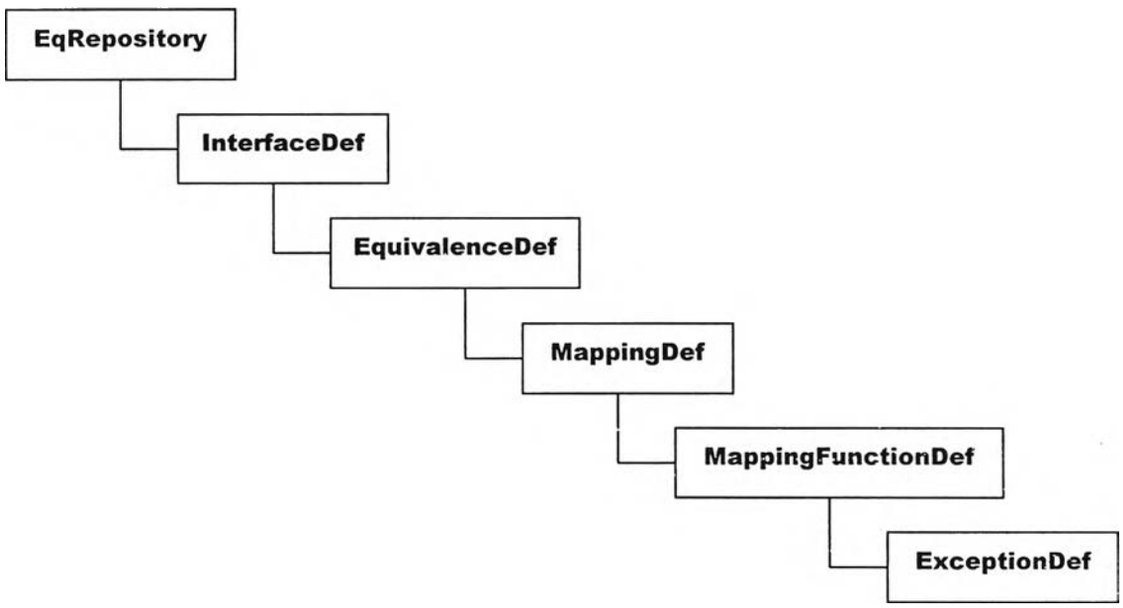
   - ชื่อชนิดของบริการ (Service Type Name) เป็นชื่อของบริการที่วัตถุต้นทางและปลายทางของการแปลงได้รับการประกาศไว้ยังเทรดเดอร์
   - รายการคุณสมบัติของชนิดของบริการ (Property List) เป็นข้อมูลรายการคุณสมบัติต่างๆของวัตถุต้นทางและปลายทางของการแปลงที่ได้รับการประกาศไว้ยังเทรดเดอร์
   - ข้อมูลอ้างอิงที่ใช้ในการเข้าถึงตัวดำเนินการแปลง (Interoperable Object Reference of Mapping Operator) เทรดเดอร์จะสามารถค้นหาบริการที่เท่าเทียมกับบริการที่ผู้รับบริการร้องขอมาได้โดยการพิจารณาจากชื่อชนิดของบริการและรายการคุณสมบัติของวัตถุต้นทางว่าสอดคล้องกับสิ่งที่ผู้รับบริการต้องการ

หรือไม่ หากสอดคล้อง แสดงว่าเทรดเดอร์สามารถส่งข้อมูลอ้างอิงที่ใช้ในการเข้าถึงตัวดำเนินการแปลงที่เกี่ยวข้องกับวัตถุต้นทางนั้นกลับไปให้ผู้รับบริการได้ ซึ่งจะทำให้ผู้รับบริการสามารถใช้งานวัตถุปลายทางที่เกี่ยวข้องผ่านตัวดำเนินการแปลงนั้นทดแทนการใช้งานวัตถุต้นทางได้



รูปที่ 3.1 การนำข้อมูลความสัมพันธ์แบบเท่าเทียมกันไปใช้โดยโปรแกรมสร้างตัวดำเนินการแปลง และเทรดเดอร์

รูปที่ 3.1 แสดงการนำข้อมูลความสัมพันธ์แบบเท่าเทียมกันเข้าสู่คลังส่วนต่อประสานโดยผู้ที่ทำการเปลี่ยนรุ่น หลังจากที่จัดเก็บข้อมูลเข้าไปแล้ว ข้อมูลดังกล่าวอาจถูกนำไปใช้โดยโปรแกรมที่ทำการสร้างตัวดำเนินการแปลง หรือเทรดเดอร์ ซึ่งการออกแบบข้อมูลที่จะจัดเก็บต้องคำนึงถึงข้อมูลที่จะถูกนำไปใช้ได้จากทั้ง 2 กรณีเป็นสำคัญ โดยนิยามของข้อมูลที่เพิ่มเติมนี้จะแสดงได้ในรูปของโครงสร้างความสัมพันธ์ในลักษณะการบรรจุ ดังรูปที่ 3.2



รูปที่ 3.2 โครงสร้างความสัมพันธ์ในลักษณะการบรรจุ (Containment Relationship) ของข้อมูลภายในคลังส่วนต่อประสานที่รองรับความสัมพันธ์แบบเท่าเทียมกัน

จากรูปที่ 3.2 จะเห็นได้ว่าการออกแบบคลังส่วนต่อประสานที่รองรับความสัมพันธ์แบบเท่าเทียมกันในมุมมองเชิงตรรกะจะมีการเชื่อมโยงเข้ากับคลังส่วนต่อประสานเดิมผ่านทางข้อมูลประเภท InterfaceDef โดยข้อมูลที่แสดงความสัมพันธ์แบบเท่าเทียมกันจะถูกบรรจุอยู่ภายในข้อมูล InterfaceDef เป็นลำดับชั้นกันลงไป อันประกอบไปด้วยข้อมูล 3 ลำดับชั้นในการอธิบายถึงแต่ละชุดของความสัมพันธ์แบบเท่าเทียมกัน ได้แก่ข้อมูลชนิด EquivalenceDef ข้อมูล MappingDef และข้อมูล MappingFunctionDef ข้อมูลที่บรรจุอยู่ในข้อมูล InterfaceDef เป็นลำดับชั้นแรกคือข้อมูลชนิด EquivalenceDef ซึ่งเป็นข้อมูลที่บ่งบอกถึงการมีความสัมพันธ์แบบเท่าเทียมกันของข้อมูล InterfaceDef คู่หนึ่งๆ (ดูหัวข้อ 3.3.2) ลำดับชั้นต่อไปคือข้อมูล MappingDef ซึ่งเป็นข้อมูลที่นำไปสร้างตัวดำเนินการแปลงระหว่างคู่อินสแตนซ์ของข้อมูลประเภท InterfaceDef ที่มีความสัมพันธ์แบบเท่าเทียมกัน (ดูหัวข้อ 3.3.3) และลำดับชั้นสุดท้ายคือ MappingFunctionDef ซึ่งเป็นข้อมูลที่ระบุฟังก์ชันการแปลง (ดูหัวข้อ 3.3.4) ข้อมูลที่แสดงความสัมพันธ์แบบเท่าเทียมกันนั้นเป็น

ข้อมูลที่มีการสืบทอดคุณลักษณะพื้นฐานมาจากคลาสพื้นฐานบางตัวได้แก่ EqIRObject, EqContained และ EqContainer ซึ่งจากคุณลักษณะพื้นฐานดังกล่าวทำให้เกิดเป็นโครงสร้างความสัมพันธ์ในลักษณะการบรรจุ และสามารถนำข้อมูลเหล่านั้นไปบรรจุอยู่ภายในข้อมูล InterfaceDef เป็นลำดับชั้นกันลงไปได้

โดยสรุปแล้วข้อกำหนดคลังชนิดของส่วนต่อประสานที่ออกแบบนี้ จะประกอบด้วยข้อกำหนด 2 ส่วนคือ

- ข้อกำหนดสำหรับข้อมูลพื้นฐาน (ดูหัวข้อ 3.2) ได้แก่นิยามชนิดข้อมูลสนับสนุน (Supporting Type Definition) และนิยามส่วนต่อประสาน EqIRObject, EqContained และ EqContainer นิยามของข้อมูลพื้นฐานเหล่านี้จะใช้ในการนิยาม ข้อกำหนดสำหรับข้อมูลความสัมพันธ์แบบเท่าเทียมกันที่เพิ่มเติมไว้ในคลังส่วนต่อประสาน
- ข้อกำหนดสำหรับข้อมูลความสัมพันธ์แบบเท่าเทียมกัน ได้แก่นิยามของส่วนต่อประสาน EqRepository, EquivalenceDef, MappingDef และ MappingFunctionDef (ดูหัวข้อ 3.3) ข้อมูลเหล่านี้จะสอดคล้องกับข้อมูลที่เพิ่มเติมเข้ามา ดังรูป 3.1 ข้างต้น

จากข้อมูลทั้ง 2 กลุ่มที่กล่าวไปแล้วข้างต้น สามารถแบ่งออกได้เป็น 7 ส่วนต่อประสานหลักๆ ด้วยกัน (ดูหัวข้อ 3.2) ซึ่งเหตุผลในการแบ่งข้อมูลดังกล่าวออกเป็น 7 ส่วนต่อประสาน ก็เนื่องมาจากการเลียนแบบโครงสร้างของคลังส่วนต่อประสานเดิมที่ได้กำหนดไว้แล้วของคอร์บา ซึ่งในข้อกำหนดคลังส่วนต่อประสานของคอร์บา เองก็มีโครงสร้างที่ชัดเจนและแบ่งกลุ่มข้อมูลไว้อย่างมีเหตุผลตามแนวคิดเชิงวัตถุรวมทั้งมีการกำหนดให้ส่วนต่อประสานซึ่งอธิบายข้อมูล มีความสัมพันธ์ในลักษณะการบรรจุกัน ซึ่งการออกแบบคลังส่วนต่อประสานที่รองรับความสัมพันธ์แบบเท่าเทียมกันก็ได้ประยุกต์แนวคิดดังกล่าวเข้ามาใช้คือ จัดกลุ่มของข้อมูลพื้นฐานที่ซ้ำซ้อนกันเอาไว้ที่ส่วนต่อประสานพื้นฐาน อันได้แก่ ส่วนต่อประสาน EqIRObject ส่วนต่อประสาน EqContained และส่วนต่อประสาน EqContainer สำหรับข้อมูลที่รองรับความสัมพันธ์แบบเท่าเทียมกันก็ได้แบ่งไว้ในส่วนต่อประสานที่เหลือ และมีความสัมพันธ์ในลักษณะการบรรจุกันด้วย (ดูหัวข้อ 3.3) จากการเลียนแบบโครงสร้างของคลังส่วนต่อประสานของคอร์บาทำให้คลังส่วนต่อประสานที่รองรับความสัมพันธ์แบบเท่าเทียมกันสามารถทำงานร่วมกับคลังส่วนต่อประสานของคอร์บาได้

เนื้อหาที่จะกล่าวต่อไปนี้จะเป็นข้อกำหนดคลังชนิดของส่วนต่อประสานที่สามารถรองรับข้อมูลความสัมพันธ์แบบเท่าเทียมกัน โดยจะแสดงด้วยภาษากำหนดนิยามส่วนต่อประสานของโอเอ็มจี (OMG IDL – Object Management Group Interface Definition Language)

3.2 ข้อกำหนดสำหรับข้อมูลพื้นฐาน

3.2.1 นิยามชนิดข้อมูลของส่วนสนับสนุน (Supporting Type Definition)

ชนิดข้อมูลเพิ่มเติมที่กำหนดขึ้นเพื่อใช้ในการกำหนดนิยามของข้อมูลที่แสดงความสัมพันธ์แบบเท่าเทียมกัน มีดังนี้

```
(1)  #include <IntRep.idl>
(2)  #include <CosTrading.idl>

(3)  module EquivalenceIR {
(4)  module idl2java {

(5)    enum EquivalenceDefinitionKind {
(6)        eq_dk_none, eq_dk_all, eq_dk_Repository,
(7)        eq_dk_EquivalenceDef, eq_dk_MappingDef,
(8)        eq_dk_MappingFunctionDef
(9)    };

(10)   enum MapMode {
(11)       mm_none, mm_Attribute, mm_Argument, mm_Result
(12)       ,mm_Operation, mm_Exception, mm_Instance
(13)   };

(14)   enum GenMode {
(15)       gm_none, gm_Direct, gm_User_Defined
(16)   };

(17)   enum EvolveLevel {
(18)     el_none, el_Instance, el_Type
(19)   };
(20) };
(21) };
```

โดยแต่ละส่วนสามารถอธิบายได้ดังนี้

`#include <IntRep.idl>` (1) เป็นการรวมแฟ้มนิยามส่วนต่อประสานของคลังส่วนต่อประสานเดิมเข้ามาในคลังส่วนต่อประสานที่รองรับความสัมพันธ์แบบเท่าเทียมกัน เพื่อให้สามารถใช้ข้อมูลพื้นฐานที่จำเป็นเช่น

```
CORBA::RepositoryId     id;
CORBA::Identifier       name;
CORBA::VersionSpec      version;
```

ข้อมูล id, name และ version เป็นลักษณะประจำพื้นฐานที่สำคัญและทุกๆส่วนต่อประสานต้องมี

`#include <CosTrading.idl>` (2) เป็นการรวมแฟ้มนิยามส่วนต่อประสานของเทรดเดอร์ซึ่งเป็นบริการไดเรกทอรี่ (Directory Service) ของคอร์บาเข้ามาในนิยามคลังส่วนต่อประสานที่รองรับความสัมพันธ์แบบเท่าเทียมกัน เพื่อใช้ข้อมูลพื้นฐานของเทรดเดอร์ เช่น ชื่อชนิดของบริการ

(ServiceTypeName) และรายการคุณสมบัติของชื่อชนิดของบริการนั้น (property_list) ซึ่งข้อมูลของเทรดเดอร์เหล่านี้ จะระบุอยู่ใน MappingDef (ดูหัวข้อ 3.3.3) ข้อมูลดังกล่าวจะสามารถเชื่อมโยงการทำงานจากเทรดเดอร์เข้ามาสู่คลังส่วนต่อประสานที่รองรับความสัมพันธ์แบบเท่าเทียมกันได้ โดยเทรดเดอร์จะสามารถ ค้นหา บริการที่เท่าเทียมกันกับบริการที่ผู้รับบริการร้องขอได้จากคลังส่วนต่อประสานนี้ (ดูหัวข้อ 6.3)

EquivalenceDefinitionKind (5-9) หมายถึงชนิดของข้อมูลในคลังส่วนต่อประสานที่กำหนดเพิ่มเติมเพื่อรองรับความสัมพันธ์แบบเท่าเทียมกัน ซึ่งจะสามารถใช้ในการตรวจสอบชนิดของวัตถุในคลังส่วนต่อประสานได้ ชนิดข้อมูลที่เพิ่มเติมมีทั้งสิ้น 4 ชนิดดังนี้

- **eq_dk_Repository** ระบุชนิดคลังส่วนต่อประสานที่รองรับความสัมพันธ์แบบเท่าเทียมกัน
- **eq_dk_EquivalenceDef** ระบุชนิดความสัมพันธ์แบบเท่าเทียมกันระหว่างส่วนต่อประสาน
- **eq_dk_MappingDef** ระบุชนิดตัวดำเนินการแปลงที่ทำการแปลงข้อมูลระหว่างส่วนต่อประสานที่มีความสัมพันธ์แบบเท่าเทียมกัน
- **eq_dk_MappingFunctionDef** ระบุชนิดฟังก์ชันการแปลงข้อมูลที่ถูกเรียกใช้โดยตัวดำเนินการแปลง เพื่อทำให้เกิดความสัมพันธ์แบบเท่าเทียมกันในลักษณะโปร่งใส

(สำหรับ eq_dk_none และ eq_dk_all กำหนดไว้เพื่อให้เกิดความสะดวกในการเขียนโปรแกรม)

MapMode (10-13) ระบุว่าฟังก์ชันการแปลงที่สร้างขึ้น มีไว้สำหรับแปลงองค์ประกอบใดของส่วนต่อประสานที่เกิดการเปลี่ยนแปลง โดยองค์ประกอบที่สามารถเปลี่ยนแปลงได้มีอยู่ทั้งสิ้น 6 กรณี คือ ลักษณะประจำ (Attribute), อาร์กิวเมนต์ (Argument), ผลลัพธ์ (Result), การดำเนินงาน (Operation), ข้อยกเว้น (Exception) และอินสแตนซ์ (Instance) (สำหรับ mm_none กำหนดไว้เพื่อให้เกิดความสะดวกในการเขียนโปรแกรม)

GenMode (14-16) กำหนดรูปแบบการสร้างฟังก์ชันการแปลงจากงานวิจัย [1] ซึ่งรูปแบบการสร้างมี 2 รูปแบบคือ สร้างอัตโนมัติโดยตรงจากโปรแกรมในงานวิจัย [1] (Direct Mapping Function) หรือเป็นโปรแกรมที่ผู้ใช้เป็นผู้สร้าง (User-Defined Mapping Function) (สำหรับ gm_none กำหนดไว้เพื่อให้เกิดความสะดวกในการเขียนโปรแกรม)

EvolveLevel (17-19) กำหนดระดับของความสัมพันธ์แบบเท่าเทียมกันระหว่างส่วนต่อประสาน (วัตถุชนิด InterfaceDef) ซึ่งสอดคล้องกับงานวิจัย [1] ระดับของความสัมพันธ์แบบเท่าเทียมกันมี 2 ระดับ คือ ระดับอินสแตนซ์ (Instance Level) และระดับชนิด (Type Level) (สำหรับ el_none กำหนดไว้เพื่อให้เกิดความสะดวกในการเขียนโปรแกรม)

3.2.2 นิยามส่วนต่อประสานของ EqIRObject

EqIRObject เป็นส่วนต่อประสานพื้นฐานที่ทุกๆส่วนต่อประสานของข้อมูลภายในคลังส่วนต่อประสานที่รองรับความสัมพันธ์แบบเท่าเทียมกันจะต้องทำการสืบทอดคุณลักษณะ (Inherit) โดยอธิบายได้ดังนี้

```
(1)  interface EqIRObject {
(2)    // read interface
(3)    readonly attribute EquivalenceDefinitionKind
                                                eq_def_kind;

(4)    // write interface
(5)    void destroy();
(6)  };
```

eq_def_kind (3) เป็นลักษณะประจำแบบอ่านอย่างเดียวที่ใช้ระบุชนิดของข้อมูลในคลังส่วนต่อประสานที่รองรับความสัมพันธ์แบบเท่าเทียมกัน โดยค่าที่สามารถกำหนดให้กับข้อมูลจะมีความสอดคล้องกับชนิดข้อมูลใน EquivalenceDefinitionKind ที่ได้กล่าวไปแล้วในหัวข้อ 3.2.1

destroy() (5) เป็นรูปแบบการให้บริการที่ใช้ในการลบชนิดของข้อมูลในคลังส่วนต่อประสานที่รองรับความสัมพันธ์แบบเท่าเทียมกัน ซึ่งจากการที่ EqIRObject เป็นส่วนต่อประสานพื้นฐาน ส่งผลให้ destroy() จะมีอยู่ในทุกๆส่วนต่อประสานของข้อมูลแสดงความสัมพันธ์แบบเท่าเทียมกัน ดังนั้นผู้ใช้สามารถเรียกใช้บริการ destroy() เพื่อลบข้อมูลต่างๆได้ และหากวัตถุที่ถูกเรียกใช้บริการ destroy() มีชนิดเป็น EqContainer การทำงานของ destroy() จะลบข้อมูลทั้งหมดที่บรรจุอยู่ใน EqContainer นั้นด้วย

3.2.3 นิยามส่วนต่อประสานของ EqContained

EqContained เป็นส่วนต่อประสานพื้นฐานสำหรับข้อมูลความสัมพันธ์แบบเท่าเทียมกันที่ถูกบรรจุอยู่ภายในข้อมูลอื่น ส่วนต่อประสานของข้อมูลที่สืบทอดคุณลักษณะจาก EqContained ได้แก่ EquivalenceDef, MappingDef และ MappingFunctionDef ส่วนต่อประสาน EqContained เป็นดังนี้

```
(1)  interface EqContained : EqIRObject {
(2)    // read/write interface
(3)    attribute CORBA::RepositoryId    id;
(4)    attribute CORBA::Identifier      name;
(5)     attribute CORBA::VersionSpec     version;

(6)     // read interface
(7)     readonly attribute Object                defined_in;
(8)    readonly attribute CORBA::ScopedName    absolute_name;
(9)    readonly attribute EqRepository containing_repository;
(10)  };
```

id, name, version, absolute_name (3-5, 8) เป็นลักษณะประจำที่ใช้ระบุถึง รหัส ชื่อ รุ่น และชื่อสัมบูรณ์ของชนิดของข้อมูลในคลังส่วนต่อประสานที่รองรับความสัมพันธ์แบบเท่าเทียมกัน ซึ่งข้อมูลเหล่านี้ได้ถูกกำหนดไว้แล้วในนิยามของคลังส่วนต่อประสานของคอร์บา

defined_in (7) เป็นลักษณะประจำที่ใช้ระบุถึงวัตถุที่เป็นตัวบรรจุ (Container) ของวัตถุ EqContained นี้ โดยสามารถสื่อความหมายของข้อมูลชนิด InterfaceDef และ EqContained ดังนั้นจึงกำหนดให้มีชนิดเป็นออบเจคต์ (Object) เพื่อที่จะสามารถรองรับชนิดข้อมูลทั้งสองได้

containing_repository (9) เป็นลักษณะประจำที่ใช้ระบุถึงวัตถุ EqRepository อันหมายถึงคลังส่วนต่อประสานที่ขยายเพื่อรองรับข้อมูลความสัมพันธ์แบบเท่าเทียมกัน ซึ่งบรรจุวัตถุ EqContained นี้อยู่ (ดูหัวข้อ 3.3.1)

3.2.4 นิยามส่วนต่อประสานของ EqContainer

EqContainer ทำให้เกิดความสัมพันธ์ในลักษณะการบรรจุของวัตถุต่างๆในคลังส่วนต่อประสานที่รองรับความสัมพันธ์แบบเท่าเทียมกัน วัตถุที่ได้รับการถ่ายทอดคุณลักษณะมาจาก EqContainer สามารถบรรจุวัตถุที่ได้รับการถ่ายทอดคุณลักษณะมาจาก EqContained เป็นจำนวนเท่าใดก็ได้ จึงก่อให้เกิดความสัมพันธ์ของโครงสร้างการบรรจุขึ้นได้ ส่วนต่อประสาน EqContainer เป็นดังนี้

```
(1)  interface EqContainer : EqIRObject {
(2)    // read interface
(3)    EqContainedSeq eq_contents();

(4)    // write interface
(5)    EquivalenceDef create_equivalence(
(6)      in CORBA::RepositoryId    id,
(7)      in CORBA::Identifier      name,
(8)      in CORBA::VersionSpec     version,
```

```
 (9)       in CORBA::InterfaceDef    original_interface,
(10)       in CORBA::InterfaceDef    equivalence_interface
(11)    );
(12)  };
```

eq_contents() (3) เป็นรูปแบบการให้บริการที่จะส่งคืนรายการ (list) ของวัตถุทั้งหมดที่ถูกบรรจุอยู่ในวัตถุ EqContainer นี้ หากมีการเรียกใช้ eq_contents() ที่วัตถุชนิด EqRepository จะได้รับรายการของวัตถุทั้งหมดที่บรรจุอยู่ในคลังส่วนต่อประสานที่รองรับความสัมพันธ์แบบเท่าเทียมกัน

create_equivalence() (5) เป็นรูปแบบการให้บริการที่ใช้ในการสร้างวัตถุชนิด EquivalenceDef (ดูหัวข้อ 3.3.2) โดยมีการรับข้อมูลพื้นฐานเข้ามาคือข้อมูลรหัส ชื่อ และรุ่นสำหรับวัตถุชนิด EquivalenceDef ที่จะถูกสร้างขึ้นใหม่ นอกจากนั้นยังมีการรับวัตถุชนิด InterfaceDef ของคอร์บาเข้ามาสองวัตถุ ซึ่งบ่งบอกถึงความสัมพันธ์แบบเท่าเทียมกันระหว่างส่วนต่อประสานทั้งสอง โดย original_interface หมายถึงส่วนต่อประสานก่อนเกิดการเปลี่ยนแปลงหรือส่วนต่อประสานที่ถูกทดแทนได้ ส่วน equivalence_interface หมายถึงส่วนต่อประสานที่ได้จากการเปลี่ยนแปลงหรือส่วนต่อประสานที่ทดแทน original_interface ได้ กล่าวได้ว่า equivalence_interface มีความเท่าเทียมกับ original_interface

3.3 ข้อกำหนดสำหรับข้อมูลความสัมพันธ์แบบเท่าเทียมกัน

3.3.1 นิยามส่วนต่อประสานของ EqRepository

EqRepository เปรียบเสมือนจุดเข้า (entry point) ในการเข้าถึงข้อมูลต่างๆที่มีอยู่ในคลังส่วนต่อประสานที่รองรับความสัมพันธ์แบบเท่าเทียมกัน เนื่องจาก EqRepository เป็นส่วนต่อประสานที่ได้รับการถ่ายคุณลักษณะมาจาก EqContainer จึงสามารถบรรจุวัตถุชนิดต่างๆ ซึ่งแสดงข้อมูลความสัมพันธ์แบบเท่าเทียมกันอันได้แก่วัตถุของส่วนต่อประสาน EquivalenceDef, MappingDef และ MappingFunctionDef ทั้งนี้ในการใช้งานจริงอาจมีมากกว่าหนึ่งวัตถุที่มีชนิดเป็น EqRepository ได้ในสภาพแวดล้อมหนึ่ง

หากต้องการเดินทาง (Navigate) หรือทราบรายการของวัตถุทั้งหมดในคลังส่วนต่อประสานที่รองรับความสัมพันธ์แบบเท่าเทียมกัน ก็ให้เรียกใช้ eq_contents() ที่วัตถุ EqRepository ซึ่ง eq_contents() เป็นรูปแบบการให้บริการของ EqContainer ที่ถูกถ่ายทอดคุณลักษณะมายัง EqRepository

ส่วนต่อประสาน EqRepository เป็นดังนี้

```
(1)  interface EqRepository : EqContainer {
(2)    // readinterface
(3)    EqContained    eq_lookup_id(
(4)      in CORBA::RepositoryId    search_id
(5)    );

(6)    CORBA::InterfaceDefSeq list_of_equivalence_objs(
(7)      in CORBA::RepositoryId    original_interface_id
(8)    );
(9)  };
```

eq_lookup_id() (3) เป็นรูปแบบการให้บริการที่ทำหน้าที่ในการค้นหาวัตถุใดๆ จากวัตถุทั้งหมดที่บรรจุอยู่ในคลังส่วนต่อประสานที่รองรับความสัมพันธ์แบบเท่าเทียมกัน โดยรับข้อมูลรหัสที่เป็นหนึ่งเดียว (unique) ของวัตถุที่ต้องการค้นหา ซึ่งรหัสมีชนิดเป็น RepositoryId ที่ถูกกำหนดไว้แล้วในนิยามของส่วนต่อประสานของคอร์บา จากการเรียกใช้ eq_lookup_id() จะได้ผลลัพธ์เป็นวัตถุชนิด EqContained เพียงหนึ่งวัตถุเท่านั้น

list_of_equivalence_objs() (6) เป็นรูปแบบการให้บริการที่จะส่งคืนรายการ (list) ของวัตถุชนิด InterfaceDef ของคอร์บาที่มีความสัมพันธ์แบบเท่าเทียมกันกับส่วนต่อประสานที่มีรหัสที่เป็นหนึ่งเดียวตามที่ระบุ

3.3.2 นิยามส่วนต่อประสานของ EquivalenceDef

EquivalenceDef เป็นส่วนต่อประสานที่ใช้ในการบ่งบอกความสัมพันธ์แบบเท่าเทียมกันระหว่างส่วนต่อประสานชนิด InterfaceDef ของคอร์บา EquivalenceDef สามารถบรรจุส่วนต่อประสาน MappingDef และ MappingFunctionDef ซึ่งระบุรายละเอียดของข้อมูลการแปลงอันแสดงความสัมพันธ์แบบเท่าเทียมกัน วัตถุชนิด EquivalenceDef จะถูกสร้างโดยการเรียก create_equivalence() ภายใน EqContainer โดยที่สำหรับส่วนต่อประสานชนิด InterfaceDef หนึ่งๆจะสามารถมีข้อมูล EquivalenceDef ได้หลายวัตถุ ซึ่งหมายถึงว่า InterfaceDef นี้สามารถถูกทดแทนได้ด้วยหลายส่วนต่อประสานนั่นเอง ส่วนต่อประสาน EquivalenceDef เป็นดังนี้

```
(1)  struct EquivalenceDescription {
(2)    CORBA::Identifier          name;
(3)    CORBA::RepositoryId        id;
(4)    CORBA::RepositoryId        defined_in;
(5)    CORBA::VersionSpec         version;
(6)    CORBA::RepositoryId        original_interface;
(7)    CORBA::RepositoryIdSeq     equivalence_interface;
(8)    EvolveLevel                evolve_level;
(9)    MappingDescriptionSeq      mapping;
(10) };
```

```
(11)  interface EquivalenceDef : EqContained, EqContainer {
(12)    // read/write interface
(13)    attribute CORBA::InterfaceDef      original_interface;
(14) attribute CORBA::InterfaceDef      equivalence_interface;
(15)    attribute EvolveLevel               evolve_level;

(16)    // read interface
(17)    boolean equivalence_with(
(18)      in CORBA::RepositoryId    interface_id
(19)    );

(20)    EquivalenceDescription describe_equivalence();

(21)    // write interface
(22)    MappingDef create_mapping(
(23)      in CORBA::RepositoryId                              id,
(24)      in CORBA::Identifier                              name,
(25)      in CORBA::VersionSpec                          version,
(26)      in EquivalenceSourceObjectPair   eq_source_obj_pair,
(27)   in EquivalenceDestinationObjectPairSeq  eq_dest_obj_pair
(28)    );
(29)  };
```

original_interface (6) เป็นลักษณะประจำที่ใช้ระบุถึงวัตถุชนิด InterfaceDef ของคอร์บาซึ่งหมายถึงส่วนต่อประสานเดิมก่อนเกิดการเปลี่ยนแปลง โดยจะสามารถทดแทนได้ด้วยวัตถุชนิด InterfaceDef อื่นๆ คือ equivalence_interface original_interface นี้จะหมายถึง InterfaceDef ที่บรรจุข้อมูล EquivalenceDef นี้อยู่

equivalence_interface (7) เป็นลักษณะประจำที่ใช้ระบุถึงวัตถุชนิด InterfaceDef ที่มีความสัมพันธ์แบบเท่าเทียมกันกับวัตถุ original_interface โดยเป็นนิยามของส่วนต่อประสานที่ได้รับการเปลี่ยนแปลงหรือเป็นนิยามที่ใช้แทนนิยามของ original_interface ได้

evolve_level (8) เป็นลักษณะประจำที่ใช้ระบุถึงระดับของความสัมพันธ์แบบเท่าเทียมกันระหว่างวัตถุชนิด InterfaceDef (อันหมายถึงระหว่าง original_interface และ equivalence_interface) ตามแนวทางของงานวิจัย [1] ซึ่งได้แก่ระดับอินสแตนซ์ และระดับชนิด

equivalence_with() (17) จุดมุ่งหมายของรูปแบบการให้บริการนี้คือต้องการจะตรวจสอบวัตถุชนิด InterfaceDef ที่บรรจุข้อมูล EquivalenceDef นี้อยู่ว่ามีความสัมพันธ์แบบเท่าเทียมกันกับอีกวัตถุชนิด InterfaceDef หรือไม่ โดยจะมีการรับข้อมูลรหัสที่เป็นหนึ่งเดียวที่มีชนิดเป็น RepositoryId ของวัตถุชนิด InterfaceDef ที่ต้องการสอบถามเข้ามา หากพบว่า InterfaceDef ที่สอบถามปรากฏเป็น equivalence_interface ภายในวัตถุ EquivalenceDef ใดๆที่ InterfaceDef

(original_interface) บรรจุอยู่ จะส่งคืนค่าความเป็นจริงที่เป็นจริง แต่หากไม่พบก็จะส่งคืนค่าความจริงที่เป็นเท็จ

describe_equivalence() (20) เป็นรูปแบบการให้บริการที่ให้ข้อมูลอธิบายวัตถุชนิด EquivalenceDef โดยข้อมูลดังกล่าวมีรูปแบบเป็นไปตามโครงสร้างข้อมูลชนิด EquivalenceDescription ซึ่งประกอบไปด้วยข้อมูลต่อไปนี้

- ชื่อ
- รหัสของวัตถุ EquivalenceDef นี้
- รหัสของวัตถุที่บรรจุวัตถุชนิด EquivalenceDef นี้ (คือรหัสที่เป็นหนึ่งเดียวของ original_interface)
- รุ่น
- วัตถุ original_interface
- วัตถุ equivalence_interface
- ระดับของความสัมพันธ์แบบเท่าเทียมกันระหว่างวัตถุชนิด InterfaceDef
- รายการของข้อมูลของส่วนต่อประสาน MappingDef (ดูหัวข้อ 3.3.3)

create_mapping() (22) เป็นรูปแบบการให้บริการที่ใช้ในการสร้างวัตถุชนิด MappingDef ที่จะถูกบรรจุอยู่ในวัตถุชนิด EquivalenceDef โดยในการสร้างวัตถุชนิด MappingDef มีการรับข้อมูลพื้นฐานเข้ามาคือข้อมูลรหัส ชื่อ และรุ่น เพื่อใช้เป็นข้อมูลพื้นฐานสำหรับวัตถุชนิด MappingDef ที่จะถูกสร้างขึ้นใหม่ นอกจากนั้นยังมีการรับวัตถุที่ช่วยเสริมรายละเอียดระหว่างวัตถุชนิด InterfaceDef ที่มีความสัมพันธ์แบบเท่าเทียมกัน (คือระหว่าง original_interface และ equivalence_interface) เข้ามาอีกด้วย อันได้แก่วัตถุชนิด EquivalenceSourceObjectPair และรายการของวัตถุชนิด EquivalenceDestinationObjectPair โดยรายละเอียดจะได้กล่าวต่อไปในหัวข้อ 3.3.3

3.3.3 นิยามส่วนต่อประสานของ MappingDef

MappingDef เป็นส่วนต่อประสานที่ใช้ในการเสริมรายละเอียดของความสัมพันธ์แบบเท่าเทียมกันระหว่างวัตถุชนิด InterfaceDef ของคอร์บา MappingDef สามารถบรรจุส่วนต่อประสาน MappingFunctionDef เพื่อใช้ในการบ่งบอกถึงรายละเอียดในเชิงพฤติกรรมในการทำให้เกิดความโปร่งใสในการเปลี่ยนแปลงชนิดของบริการ ส่วนต่อประสาน MappingDef เป็นดังนี้

```
(1)  struct EquivalenceSourceObjectPair {
(2)    CORBA::Identifier          source_obj_name;
(3)    CORBA::Identifier          source_server_name;
(4)    CORBA::Identifier          source_host_name;
(5)    CosTrading::Istring        ServiceTypeName;
(6)    CosTrading::PropertySeq    property_list;
(7)  };

(8)  struct EquivalenceDestinationObjectPair {
(9)    CORBA::Identifier          dest_obj_name;
(10)   CORBA::Identifier          dest_server_name;
(11)   CORBA::Identifier          dest_host_name;
(12)   CosTrading::Istring        ServiceTypeName;
(13)   CosTrading::PropertySeq    property_list;
(14) };

(15) struct MappingDescription {
(16)   CORBA::Identifier                      name;
(17)   CORBA::RepositoryId                    id;
(18)   CORBA::RepositoryId                    defined_in;
(19)   CORBA::VersionSpec                     version;
(20)   EquivalenceSourceObjectPair          eq_source_obj_pair;
(21)   EquivalenceDestinationObjectPairSeq  eq_dest_obj_pair;
(22)   MappingFnDescriptionSeq                mappingFns;
(23)   string                                 mapping_ref;
(24) };

(25) interface MappingDef : EqContained, EqContainer {
(26)   // read/write interface
(27)   attribute EquivalenceSourceObjectPair
                                        eq_source_obj_pair;
(28)   attribute EquivalenceDestinationObjectPairSeq
                                          eq_dest_obj_pair;
(29)   attribute string mapping_ref;

(30)   // read interface
(31)   MappingDescription describe_mapping();

(32)   // write interface
(33)   MappingFunctionDef create_mapping_function (
(34)     in CORBA::RepositoryId         id,
(35)     in CORBA::Identifier           name,
(36)     in CORBA::VersionSpec          version,
(37)     in CORBA::IDLType              result,
(38)     in CORBA::OperationMode        mode,
(39)     in CORBA::ParDescriptionSeq    params,
(40)     in CORBA::ExceptionDefSeq      exceptions,
(41)     in GenMode                     gen_mode,
(42)     in MapMode                     map_mode
(43)   );
(44) };
```

eq_source_obj_pair (27) เป็นลักษณะประจำที่มีชนิดเป็น EquivalenceSourceObjectPair ซึ่งเป็นโครงสร้างของข้อมูลที่บ่งชี้ถึงวัตถุหนึ่งซึ่งเป็นวัตถุเดิมก่อนเกิดการเปลี่ยนแปลงชนิดของบริการหรือเป็นวัตถุที่สามารถแทนที่ได้โดยวัตถุของชนิด InterfaceDef อื่น (คือวัตถุของ original_interface นั่นเอง) โดยข้อมูลดังกล่าวประกอบไปด้วย ชื่อวัตถุต้นทาง (source_obj_name) ชื่อผู้ให้บริการต้นทาง (source_server_name) ชื่อแม่ข่ายต้นทาง (source_host_name) ชื่อชนิดของบริการ (ServiceTypeName) และรายการคุณสมบัติ (property_list) โดยข้อมูลชื่อวัตถุต้นทาง ชื่อผู้ให้บริการต้นทาง และชื่อแม่ข่ายต้นทาง ของวัตถุเดิม หรือวัตถุที่จะถูกแทนที่จะมีชนิดเป็น Identifier ซึ่งเป็นชนิดข้อมูลพื้นฐานที่ได้กำหนดไว้แล้วในนิยามส่วนต่อประสานของคลังส่วนต่อประสานของคอร์บา ส่วนชื่อชนิดของบริการและรายการคุณสมบัติ เป็นข้อมูลเพิ่มเติมของวัตถุหากวัตถุได้รับการลงทะเบียนไว้กับเทรดเดอร์ด้วย โดยจะระบุชนิดของบริการที่วัตถุได้รับการประกาศไว้ในเทรดเดอร์ รวมทั้งข้อมูลคุณสมบัติอื่นๆเฉพาะตัวของวัตถุนั้น ข้อมูลทั้งสองมีชนิดเป็น Istring และ PropertySeq ตามลำดับ ซึ่งเป็นชนิดข้อมูลพื้นฐานที่ได้กำหนดไว้แล้วในนิยามส่วนต่อประสานของเทรดเดอร์

eq_dest_obj_pair (28) เป็นลักษณะประจำที่มีชนิดเป็น EquivalenceDestinationObjectPairSeq ซึ่งเป็นโครงสร้างของข้อมูลที่บ่งชี้ถึงวัตถุที่มีการเปลี่ยนแปลงชนิดของบริการหรือวัตถุที่สามารถแทนที่วัตถุของชนิด InterfaceDef อื่นได้ (คือวัตถุของ equivalence_interface นั่นเอง) โดยโครงสร้างข้อมูลนี้จะอยู่ในรูปของรายการโครงสร้างข้อมูลชนิด EquivalenceDestinaitonObjectPair เนื่องจากในบางกรณีวัตถุต้นทางหนึ่งวัตถุสามารถทดแทนได้ด้วยวัตถุปลายทางมากกว่าหนึ่งวัตถุ ข้อมูลดังกล่าวมีความคล้ายคลึงกับข้อมูลของ eq_source_obj_pair ซึ่งประกอบไปด้วย ชื่อวัตถุปลายทาง (dest_obj_name) ชื่อผู้ให้บริการปลายทาง (dest_server_name) ชื่อแม่ข่ายปลายทาง (dest_host_name) ชื่อชนิดของบริการ (ServiceTypeName) และรายการคุณสมบัติ (property_list) โดยข้อมูลชื่อวัตถุปลายทาง ชื่อผู้ให้บริการปลายทาง และชื่อแม่ข่ายปลายทาง ของวัตถุใหม่หลังการเกิดการเปลี่ยนแปลงชนิดของบริการหรือวัตถุที่จะทำการแทนที่จะมีชนิดเป็น Identifier ซึ่งเป็นชนิดข้อมูลพื้นฐานที่ได้กำหนดไว้แล้วในนิยามส่วนต่อประสานของคลังส่วนต่อประสานของคอร์บา ส่วนชื่อชนิดของบริการและรายการคุณสมบัติ เป็นข้อมูลเพิ่มเติมของวัตถุหากวัตถุได้รับการลงทะเบียนไว้กับเทรดเดอร์ด้วย โดยจะระบุชนิดของบริการที่วัตถุได้รับการประกาศไว้ในเทรดเดอร์ รวมทั้งข้อมูลคุณสมบัติอื่นๆเฉพาะตัวของวัตถุนั้น ข้อมูลทั้งสองมีชนิดเป็น Istring และ PropertySeq ตามลำดับ ซึ่งเป็นชนิดข้อมูลพื้นฐานที่ได้กำหนดไว้แล้วในนิยามส่วนต่อประสานของเทรดเดอร์

mapping_ref (29) เป็นลักษณะประจำที่มีชนิดเป็นสายอักขระ (string) ใช้ในการเก็บข้อมูลที่ใช้ในการเข้าถึงวัตถุที่เป็นตัวดำเนินการแปลง หรือกล่าวอีกนัยหนึ่งได้ว่าเป็นวัตถุที่ถูกสร้างขึ้นมาแทนที่วัตถุ eq_source_obj_pair เพื่อให้กลไกการทำงานจริงมีการเรียกใช้งาน eq_dest_obj_pair ได้อย่างโปร่งใสในสภาพแวดล้อมของคอร์บา ข้อมูล mapping_ref คือข้อมูลอ้างอิงวัตถุเพื่อการดำเนินการร่วมกัน (Interoperable Object Reference) [5]

describe_mapping() (31) เป็นรูปแบบการให้บริการที่ให้ข้อมูลเกี่ยวกับวัตถุชนิด MappingDef โดยข้อมูลดังกล่าวมีรูปแบบเป็นไปตามโครงสร้างข้อมูลชนิด MappingDescription ซึ่งประกอบไปด้วยข้อมูลต่อไปนี้

- ชื่อ
- รหัสของวัตถุ MappingDef นี้
- รหัสของวัตถุที่บรรจุวัตถุชนิด MappingDef นี้ (คือรหัสที่เป็นหนึ่งเดียวของวัตถุชนิด EquivalenceDef ที่บรรจุวัตถุ MappingDef นี้)
- รุ่น
- วัตถุชนิด EquivalenceSourceObjectPair
- รายการของวัตถุชนิด EquivalenceDestinationObjectPair
- รายการของข้อมูลของส่วนต่อประสาน MappingFunctionDef (ดูหัวข้อ 3.3.4)
- ข้อมูลอ้างอิงที่ใช้ในการเข้าถึงวัตถุที่เป็นตัวดำเนินการแปลง

create_mapping_function (33) เป็นรูปแบบการให้บริการที่ใช้ในการสร้างวัตถุชนิด MappingFunctionDef ที่จะถูกบรรจุอยู่ในวัตถุชนิด MappingDef โดยในการสร้างวัตถุชนิด MappingFunctionDef จะมีการรับข้อมูลพื้นฐานเข้ามาคือข้อมูลรหัส ชื่อ รุ่น ผลลัพธ์ ภาวะของการดำเนินการ (Operaiton Mode) รายการของพารามิเตอร์ และรายการของข้อยกเว้น เพื่อใช้เป็นข้อมูลพื้นฐานสำหรับวัตถุชนิด MappingFunctionDef ที่จะถูกสร้างขึ้นใหม่ นอกจากนี้ยังมีการรับข้อมูลชนิด GenMode ที่ใช้กำหนดรูปแบบการสร้างฟังก์ชันการแปลงจากงานวิจัย [1] และข้อมูล Mapmode ที่ระบุว่าฟังก์ชันการแปลงที่สร้างขึ้นมีไว้สำหรับการแปลงองค์ประกอบใดของส่วนต่อประสานที่มีการเปลี่ยนแปลง ซึ่งชนิดข้อมูลทั้งสองนี้ได้อธิบายไปแล้วในหัวข้อ 3.2.1

3.3.4 นิยามส่วนต่อประสานของ MappingFunctionDef

MappingFunctionDef เป็นส่วนต่อประสานที่ใช้บ่งบอกถึงรายละเอียดในเชิงพฤติกรรมในการทำให้เกิดความโปร่งใสในการเปลี่ยนแปลงชนิดของบริการ และสามารถอธิบายได้ดังนี้

```
(1)  struct MappingFunctionDescription {
(2)    CORBA::Identifier           name;
(3)    CORBA::RepositoryId         id;
(4)    CORBA::RepositoryId         defined_in;
(5)    CORBA::VersionSpec          version;
(6)    TypeCode                    result;
(7)    CORBA::OperationMode        mode;
(8)    CORBA::ParDescriptionSeq    parameters;
(9)    CORBA::ExcDescriptionSeq    exceptions;
(10)   GenMode                     gen_mode;
(11)   MapMode                     map_mode;
(12)   MappingFunctionBody         mapFnBody;
(13) };

(14) interface MappingFunctionDef : EqContained {
(15)   // read/write interface
(16)   attribute CORBA::ParDescriptionSeq    params;
(17)   attribute CORBA::OperationMode        mode;
(18)   attribute CORBA::ExceptionDefSeq      exceptions;
(19)   attribute GenMode                     gen_mode;
(20)   attribute MapMode                     map_mode;
(21)   attribute MappingFunctionBody         mapFnBody;

(22)   // read interface
(23)   readonly attribute TypeCode result;
(24)  MappingFunctionDescription describe_mapping_function();
(25) };
```

params (16) เป็นลักษณะประจำที่มีชนิดเป็น ParDescription [5] ซึ่งเป็นโครงสร้างของข้อมูลที่บ่งบอกถึงข้อมูลพารามิเตอร์ของแต่ละฟังก์ชันการแปลง โดยที่ลำดับในรายการของพารามิเตอร์มีความสำคัญและพารามิเตอร์จะอยู่ในรูปของคู่ลำดับระหว่างชื่อและชนิด โครงสร้างข้อมูลนี้ได้มีการกำหนดไว้แล้วในนิยามของคลังส่วนต่อประสานของคอร์บา

mode (17) เป็นลักษณะประจำที่มีชนิดเป็น OperationMode [5] ซึ่งเป็นข้อมูลที่บ่งบอกถึงภาวะของการดำเนินการ โดยค่าที่เป็นไปได้ก็คือ oneway หรือ normal ซึ่งได้มีการกำหนดไว้แล้วในนิยามของคลังส่วนต่อประสานของคอร์บา โดย oneway หมายถึงการดำเนินการจะไม่ส่งผลลัพธ์กลับ และ normal หมายถึงการดำเนินการจะส่งผลลัพธ์กลับ สำหรับฟังก์ชันการแปลงนั้นจะส่งผลลัพธ์กลับเสมอ

exceptions (18) เป็นลักษณะประจำที่มีชนิดเป็น ExceptionDef [5] ซึ่งเป็นข้อมูลที่บ่งบอกถึงรายการของข้อยกเว้นที่อาจเกิดจากการทำงานของฟังก์ชันการแปลง โดยได้มีการกำหนดไว้แล้วในนิยามของคลังส่วนต่อประสานของคอร์บา

gen_mode (19) เป็นลักษณะประจำที่มีชนิดเป็น GenMode ซึ่งมีค่าที่แสดงรูปแบบการสร้างฟังก์ชันการแปลงจากงานวิจัย [1] อันได้แก่ gm_none, gm_Direct และ gm_User_Defined ดังที่อธิบายไปแล้วในหัวข้อ 3.2.1

map_mode (20) เป็นลักษณะประจำที่มีชนิดเป็น MapMode ซึ่งมีค่าที่แสดงว่าฟังก์ชันการแปลงถูกสร้างขึ้นเพื่อแปลงองค์ประกอบใดในส่วนต่อประสานที่เปลี่ยนไป อันได้แก่ mm_none, mm_Attribute, mm_Argument, mm_Result, mm_Operation, mm_Exception และ mm_Instance ดังที่อธิบายไปแล้วในหัวข้อ 3.2.1

mapFnBody (21) เป็นลักษณะประจำที่มีชนิดเป็น MappingFunctionBody ซึ่งก็คือแฟ้มข้อมูลของฟังก์ชันการแปลงในรูปแบบที่เป็นรหัสต้นฉบับ (Source Code) โดยในงานวิจัย [1] จะนำแฟ้มข้อมูลของฟังก์ชันการแปลงในรูปแบบที่เป็นรหัสต้นฉบับ ไปทำการแปลโปรแกรม (Compile) และเชื่อมโยงรวม (Link) เข้ากับกระบวนการสร้างตัวดำเนินการแปลงโดยอัตโนมัติ นอกจากนั้นตัวรหัสต้นฉบับเอง ยังสามารถเป็นสารสนเทศเชิงพฤติกรรมในการศึกษาการทำให้เกิดความโปร่งใสในการเปลี่ยนแปลงชนิดของบริการได้อีกด้วย

result (23) เป็นลักษณะประจำที่มีชนิดเป็น TypeCode [5] ซึ่งเป็นข้อมูลที่บ่งบอกถึงชนิดของข้อมูลผลลัพธ์จากการดำเนินการของฟังก์ชันการแปลง โดยได้มีการกำหนดไว้แล้วในนิยามของคลังส่วนต่อประสานของคอร์บา

describe_mapping_function() (24) เป็นรูปแบบการให้บริการที่ให้ข้อมูลเกี่ยวกับวัตถุชนิด MappingFunctionDef โดยข้อมูลดังกล่าวมีรูปแบบเป็นไปตามโครงสร้างข้อมูลชนิด MappingFunctionDescription ซึ่งประกอบไปด้วยข้อมูลต่อไปนี้

- ชื่อ
- รหัสของวัตถุ MappingFunctionDef นี้
- รหัสของวัตถุที่บรรจุวัตถุชนิด MappingFunctionDef นี้ (คือรหัสที่เป็นหนึ่งเดียวของวัตถุชนิด MappingDef ที่บรรจุวัตถุ MappingFunctionDef นี้)
- รุ่น

- ชนิดของข้อมูลที่เป็นผลลัพธ์จากการดำเนินการของฟังก์ชันการแปลง
- ข้อมูลภาวะของการดำเนินการของฟังก์ชันการแปลง
- รายการของข้อมูลพารามิเตอร์ของฟังก์ชันการแปลง
- รายการของข้อมูลข้อยกเว้นที่อาจเกิดขึ้นได้จากฟังก์ชันการแปลง
- ข้อมูลแสดงรูปแบบการสร้างฟังก์ชันการแปลงจากงานวิจัย [1]
- ข้อมูลที่ใช้ระบุว่าฟังก์ชันการแปลงนี้มีไว้สำหรับแปลงองค์ประกอบใดของส่วนต่อประสานที่เปลี่ยนไป
- แฟ้มข้อมูลของฟังก์ชันการแปลงในรูปแบบที่เป็นรหัสต้นฉบับ

ในบทนี้ได้อธิบายถึงนิยามของคลังชนิดของบริการที่ได้ออกแบบให้สามารถรองรับความสัมพันธ์แบบเท่าเทียมกันอย่างละเอียด จากนิยามดังกล่าวสามารถนำไปทำให้เกิดผลเพื่อใช้ในระบบกระจาย โดยไม่ขึ้นอยู่กับภาษาการโปรแกรมหรือระบบปฏิบัติการใด ในบทถัดไปจะอธิบายถึงแนววิธีการที่จะนำนิยามดังกล่าวไปทำให้เกิดผลในการพัฒนาต้นแบบคลังชนิดของบริการที่รองรับความสัมพันธ์แบบเท่าเทียมกัน