ภาคผนวก ก

กฎการแปลงแบบจำลอง UML เป็นแบบจำลองจาวา

การแปลงแบบจำลองจาก UML เป็นแบบจำลองของเพลตฟอร์มภาษาจาวาที่เป็นพื้นฐานของภาษาจาวาสามารถจับคู่อิลิเมนต์ของเมตาโมเดลของแบบจำลอง UML ซึ่งเป็นแบบจำลองต้นทางกับอิลิเมนต์ของเมตาโมเดลของแบบจำลองจาวาซึ่งเป็นแบบจำลองปลายทางได้ โดยแสดงในภาพที่ ก.1

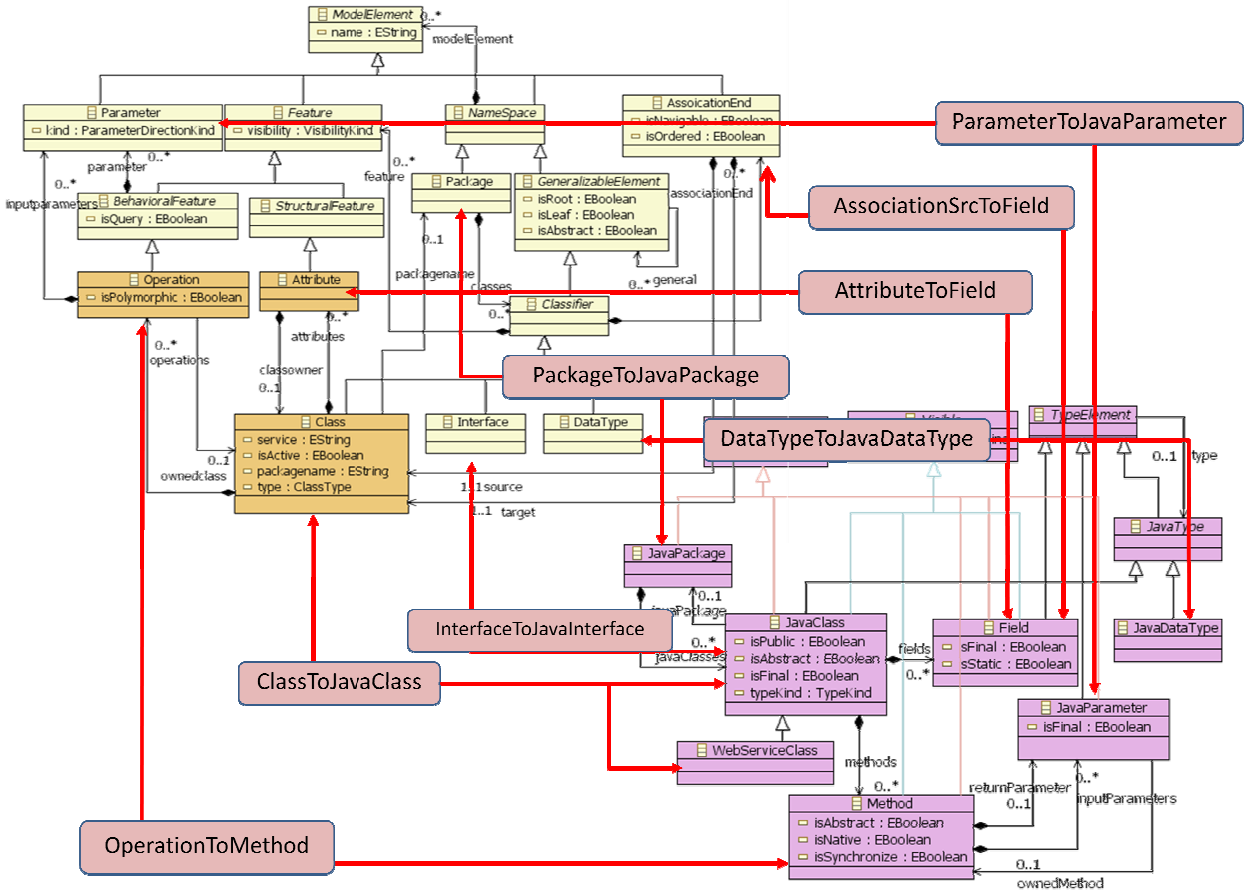
การจับคู่ความสัมพันธ์ระหว่างเมตาโมเดลของ UML และเมตาโมเดลของจาวา เพื่อใช้ในการกำหนดกฎการแปลงแบบจำลองได้ดังนี้

- กฎ PackageToJavaPackage การจับคู่ระหว่าง UML Package กับ Java Package

```
--UML Package to Java Package
top relation PackageToJavaPackage {
      pn: String;
      checkonly domain umlsec p:umlsec::Package {
            classes = uc:umlsec::Class{},
            name = pn
      };

      enforce domain javaws jp:javasec::JavaPackage {
            javaClasses = jc:javasec::JavaClass{},

            name = pn
      };

      where{
            ClassToJavaClass(uc, jc) or
            ServiceClassToJavaWSClass(uc, jc);

      }
```

ภาพที่ ก.1

การจับคู่ (Mapping) ระหว่างเมตาโมเดลของ UML กับเมตาโมเดลของจาวา



- กฎ ClassToJavaClass การจับคู่ระหว่าง UML Class กับ Java Class

```
relation ClassToJavaClass {
	cn, kname: String;
	isAbstract : Boolean;
	checkonly domain umlsec c:umlsec::Class {
		name = cn,
		kind = kname,
		isAbstract = isAbstract
	};

	enforce domain javaws jc:javasec::JavaClass {
		name = cn,
		typeKind = javasec::TypeKind::TypeClass,
		isPublic = true,
		isAbstract = isAbstract
	};

	when {
		kname = '';
	}

	where {
		AttributeToField(c, jc);
		OperationToMethod(c, jc);
	}
}
```

- กฎ InterfaceToJavaInterface การจับคู่ระหว่าง UML Interface กับ Java Class ประเภท Interface

```
relation InterfaceToJavaInterface {
	iName: String;
	isAbstract : Boolean;
	checkonly domain umlsec i:umlsec::Interface {
		name = iName,
		isAbstract = isAbstract
	};

	enforce domain javaws ji:javasec::JavaClass {
		name = iName,
		isPublic = true,
		isAbstract = isAbstract,
		typeKind = javasec::TypeKind::TypeInterface
	};

	where {
		InfAttributeToField(i, ji);
		InfOperationToMethod(i, ji);
	}
}
```

- กฎ OperationTOMethod การจับคู่ระหว่าง UML Operation กับ Java Method

```
relation OperationToMethod {
	opName : String;
	checkonly domain umlsec c:umlsec::Class {
		operations = o:umlsec::Operation {
			name = opName
		}
	};

	enforce domain javaws jc:javasec::JavaClass {
		methods = jm:javasec::Method{
			name = opName
		}
	};

	where {
		parameterToInputJavaParameter(o, jm);
		parameterToReturnJavaParameter(o, jm);
	}
}
```

- กฎ DataTypeToJavaDataType การจับคู่ระหว่าง UML Data Type กับ Java Data Type

```
top relation DataTypeToJavaDataType {
	dtName : String;
	checkonly domain umlsec dt:umlsec::DataType {
		name = dtName
	};

	enforce domain javaws jd:javasec::JavaDataType {
		name = dtName
	};
}
```

- กฎ AttributeToField การจับคู่ระหว่าง UML Attribute กับ Java Field

```
-- UMLAtrribute to Java Field
relation AttributeToField {
	checkonly domain umlsec c:umlsec::Class {};
	enforce domain javaws jc:javasec::JavaClass {};


	where {
		PrimitiveAttributeToField(c, jc);
	}
}

-- PrimitiveType Attribute to Java Field
relation PrimitiveAttributeToField {
	an : String;
	checkonly domain umlsec c:umlsec::Class{
		attributes = a:umlsec::Attribute {
			name = an,
			datatype = dt:umlsec::DataType{},

			visibility = umlsec::VisibilityKind
					::public
		}
	};

	enforce domain javaws jc : javasce::JavaClass{
		fields = f : javasec::Field {
		name = an,
		type = jdt:javasec::JavaDataType{},
			visibilityKind =
				javasec::VisibilityKind
					::public
		}
	};

	when {
		DataTypeToJavaDataType(dt, jdt);

	}
}
```

- กฎ ParameterTOJavaParameter การจับคู่ระหว่าง UML Parameter กับ Java Package

```
relation parameterToInputJavaParameter {
	ipn, dtName : String;
	checkonly domain umlsec o:umlsec::Operation {
		inputparameters = ip:umlsec::Parameter{
			name = ipn,
			kind = umlsec::
				ParameterDirectionKind::IN,
			type = dt:umlsec::DataType{
				name = dtName
			}
		}
	};
	enforce domain javaws jm:javasec::Method {
		inputParameters = ipj:javasec::JavaParameter{
			name = ipn,
			ownedmethod = jm:javasec::Method {},
			type = jdt:javasec::JavaDataType{
				name = dtName
			}
		}
	};
	when {
		DataTypeToJavaDataType(dt, jdt);
	}
}
relation parameterToReturnJavaParameter {
	dTypeName, pName : String;
	checkonly domain umlsec o:umlsec::Operation {
		inputparameters = ip:umlsec::Parameter{

			name = pName,
			kind = umlsec::ParameterDirectionKind
					::RETURN,
			type = dt:umlsec::DataType{
				name = dTypeName
			}
		}
	};
	enforce domain javaws jm:javasec::Method {
		returnParameter = rp:javasec::JavaParameter{
			name = pName,
			isFinal = false,
			ownedmethod = jm:javas::Method {},
			type = jdt:javasec::JavaDataType{
				name = dTypeName
			}
		}
	};
	when {
		DataTypeToJavaDataType(dt, jdt);
	}
}
```

- กฎ AssociationSrcToField การจับคู่ระหว่าง UML Association กับ Java Field

```
-- UMLAssociation to JavaField
top relation AssociationSrcToField {
	assoName, cName, tName : String;

	checkonly domain umlsec ass:umlsec::AssociationEnd{
		name = assoName,
		source = sClass:umlsec::Class{
			name = cName
		},
		target = tClass:umlsec::Class {
			name = tName
		}
	};

	enforce domain javaws jc: javasec::JavaClass {
		fields = f : javasec::Field {
			name = assoName,
			type = jc: javasec::JavaClass{}
		}
	};

	when {
		ClassToJavaWSClass(sClass, jc)
		and ClassToJavaClass(sClass, jc);

	}
}
```