รายการอ้างอิง

[1] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesleys, 1995.

[2] Joseph W. Yoder & Reza Razavi. "Metadata and Adaptive Object-Models", ECOOP'2000 Workshop Reader; Lecture Notes in Computer Science, vol. no. 1964; Springer Verlag 2000.

[3] Ralph Johnson and Bobby Woolf, "The Type Object Pattern", Pattern Languages of Program Design. Vol. 3. Addison-Wesley, 1997.

[4] Joseph W. Yoder, Federico Balaguer and Ralph Johnson, "Architecture and Design of Adaptive Object-Models", ACM SIG-PLAN Notices 36, No.12 pp.50-60, 2001.

[5] Martin Fowler, "Analysis Patterns: reusable object models", Addison Wesley. 1997

[6] Matthew Flatt and Matthias Felleisen, "Units: Cool Modules for HOT Languages", ACM SIGPLAN Notices, Proceedings of the ACM SIGPLAN 1998 conference on Programming language design and implementation PLDI '98.

[7] Sean McDirmid, Matthew Flatt, Wilson Hsieh, "Jiazzi: new-age components for old-fasioned Java ", ACM SIGPLAN Notices , Proceedings of the 16th ACM SIGPLAN conference on Object oriented programming, systems, languages, and applications OOPSLA '01.

[8] N. E. Fenton and S. L. Pfleeger, Software Metrics: A Rigorous and Practical Approach: PWS Publishing Company, 1997.

[9] http://www.sdmetrics.com

[10] Eric Eide, Alastair Reid, John Regehr and Jay Lepreau, "Static and Dynamic Structure in Design Patterns" Software Engineering, 2002. ICSE 2002. Proceedings of the 24rd International Conference on 2002 Page(s): 208 - 218

[11] Marcela Genero, and Mario Piattini, "Empirical validation of measures for class diagram structural complexity through controlled experiments", 5th International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering, 2002.

[12] M. Fowler, Reducing Coupling, IEEE Software July August, 2001, pp. 102-105

ภาคผนวก

ภาคผนวก ก

การใช้งานเครื่องมือยูนิตเจนเนอร์เรเตอร์ (Unit Generator)

ในภาคผนวกนี้ จะอธิบายถึงการใช้งานเครื่องมือเพื่อช่วยในออกแบบเอโอเอ็มด้วย Jiazzi (Unit Model) โดยมีฟังก์ชันการทำงานหลัก 3 ส่วนคือ การสร้างแพ็กเกจซิกเนเจอร์, อะตอมยูนิต และ คอม-พาวด์ยูนิต

นอกจากนั้นเครื่องมือได้จัดเตรียม แพ็กเกจซิกเนเจอร์, อะตอมยูนิต และ คอมพาวด์ยูนิต สำหรับเอโอเอ็ม ที่ผู้ออกแบบสามารถนำไปใช้ในการออกแบบแอพพลิเคชันด้วยยูนิตโมเดลต่อไปได้

ก.1. หน้าจอหลักของเครื่องมือ

เมื่อผู้ใช้เข้าสู่โปรแกรม ผู้ใช้สามารถมองเห็นไฟล์ของแพ็กเกจซิกเนเจอร์และยูนิตต่างๆ ใน Working Directory ทางด้านซ้ายมือของหน้าจอ ในรูปแบบของแผนภาพต้นไม้ แยกตามประเภทของคลาส ผู้ใช้สามารถเลือกคลาสที่ต้องการแก้ไข โดยการดับเบิ้ลคลิกที่แพ็กเกจซิกเนเจอร์หรือยูนิตที่ต้องการได้ ดังรูปที่ ก.1

หลักในการตั้งชื่อของแพ็กเกจซิกเนเจอร์, อะตอมยูนิต และ คอมพาวด์ยูนิต นั้นจะขึ้นต้นด้วยตัวอักษร s_, au_ และ cu_ ตามลำดับ

รูปที่ ก.1 หน้าจอหลักของเครื่องมือยูนิตเจนเนอร์เรเตอร์

ก.2. การสร้างแพ็กเกจซิกเนเจอร์

เมื่อผู้ใช้ต้องการใช้เครื่องมือในการสร้างแพ็กเกจซิกเนเจอร์ ผู้ใช้สามารถเลือกเมนูทูลบาร์ Signature ระบบจะแสดงหน้าจอให้ผู้ใช้ใส่ข้อมูลดังรูปที่ ก.2 ซึ่งประกอบด้วย

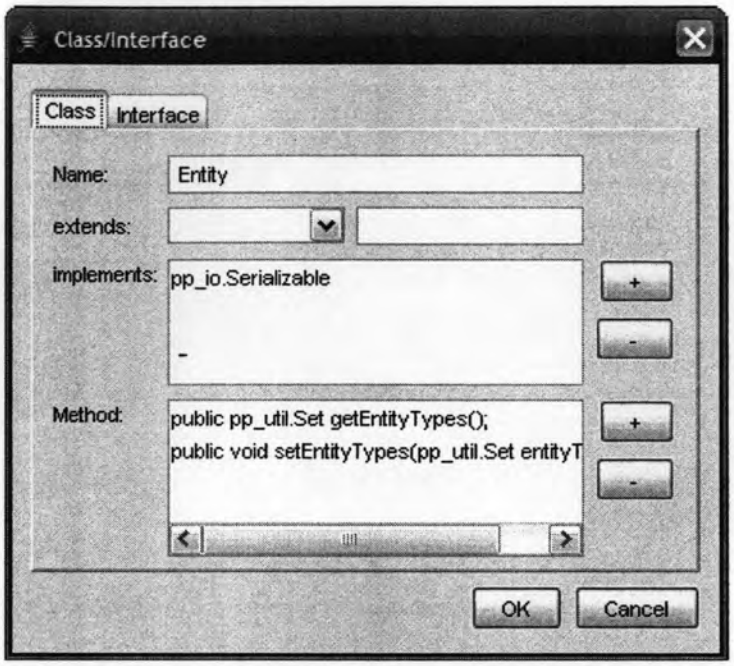
- Package Parameter ซึ่งเป็นพารามิเตอร์ของชื่อแพ็กเกจที่จะถูกใช้ในการผูก (Bind) กับชื่อแพ็กเกจจริงของจาวา
- Signature Name ชื่อของแพ็กเกจซิกเนเจอร์
- Classes/Interface รายชื่อคลาสหรืออินเตอร์เฟสของแพ็กเกจซิกเนเจอร์นี้

รูปที่ ก.2 หน้าจอการสร้างซิกเนเจอร์

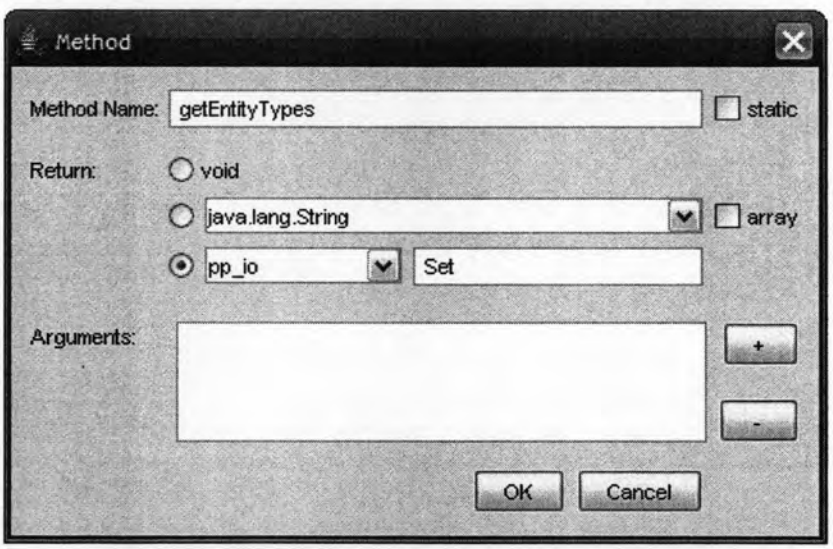
เมื่อผู้ใช้ต้องการเพิ่มคลาสให้กับแพ็กเกจซิกเนเจอร์ ให้ทำการกดปุ่มที่เครื่องหมาย + เพื่อกำหนดรายละเอียดเกี่ยวกับคลาสหรืออินเตอร์เฟส ดังแสดงในรูปที่ ก.3 ซึ่งผู้ใช้จะต้องกรอกข้อมูลเกี่ยวตัวคลาสที่ โดยการกำหนดชื่อคลาส ชื่อคลาสแม่ที่จะสืบทอด อินเตอร์เฟสที่จะอิมพลิเมนต์และเมท็อด

เนื่องจากเมท็อดมีได้หลายเมท็อดและรายละเอียดหลายส่วน ดังนั้นผู้ใช้จะต้องกดปุ่มเครื่องหมาย + เพื่อกำหนดชื่อเมท็อด พารามิเตอร์ ประเภทของข้อมูลที่จะคืนค่า ดังแสดงในรูปที่ ก.4 และ ก.5

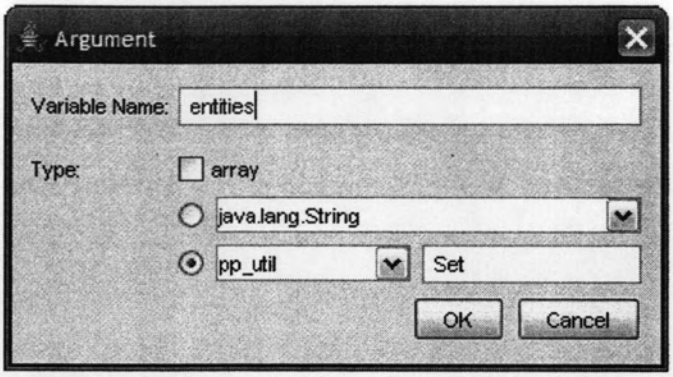
ในการสร้างแพ็กเกจซิกเนเจอร์นั้น จะเห็นได้ว่า หากมีการอ้างถึงคลาสใดๆก็ตาม แพ็กเกจพารามิเตอร์ จะถูกนำมาใช้ในการอ้างถึงเสมอ เนื่องจากแพ็กเกจพารามิเตอร์จะเป็นตัวกำหนดว่าคลาสดังกล่าวนั้น หมายถึงคลาสที่มาจากแพ็กเกจใด ซึ่งแพ็กเกจที่แท้จริงนั้นจะถูกกำหนดขึ้นในตอนที่สร้างอะตอมหรือคอมพาวด์ยูนิต



รูปที่ ก.3 หน้าจอการเพิ่มคลาสหรืออินเตอร์เฟสให้กับซิกเนเจอร์



รูปที่ ก.4 หน้าจอการเพิ่มเมท็อดให้กับคลาสหรืออินเตอร์เฟส



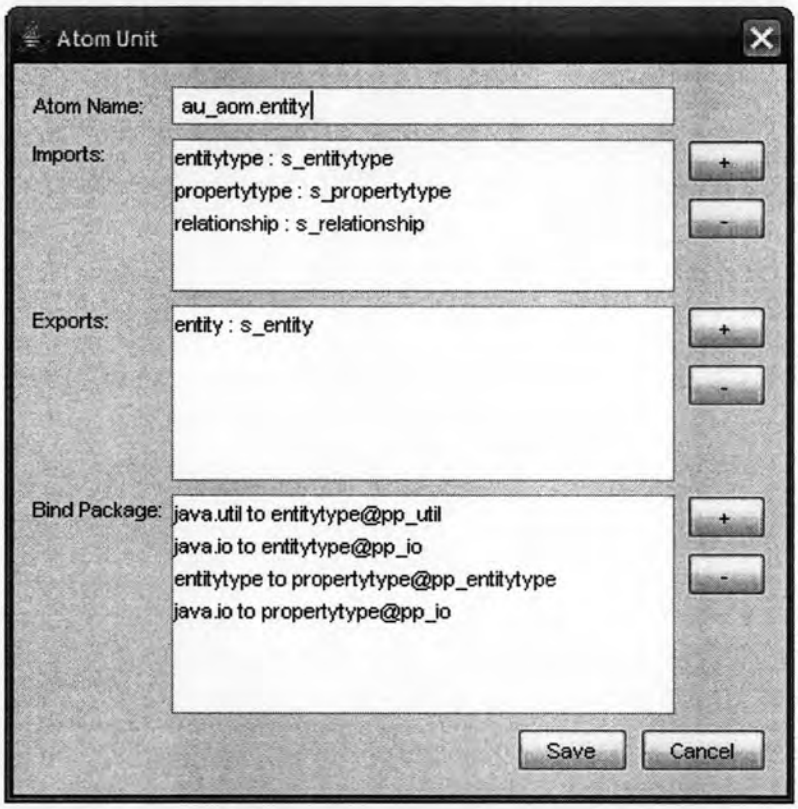
รูปที่ ก.5 หน้าจอการเพิ่มพารามิเตอร์ให้กับเมท็อด

หลังจากที่ผู้ใช้ได้กรอกรายละเอียดของแพ็กเกจซิกเนเจอร์แล้ว ให้ทำการกดปุ่ม Save เพื่อบันทึก โปรแกรมก็จะทำการสร้างแพ็กเกจซิกเนเจอร์ออกมา ซึ่งจะปรากฏไฟล์ของแพ็กเกจซิกเนเจอร์อยู่ใน Working Directory

ก.3. การสร้างอะตอมยูนิต

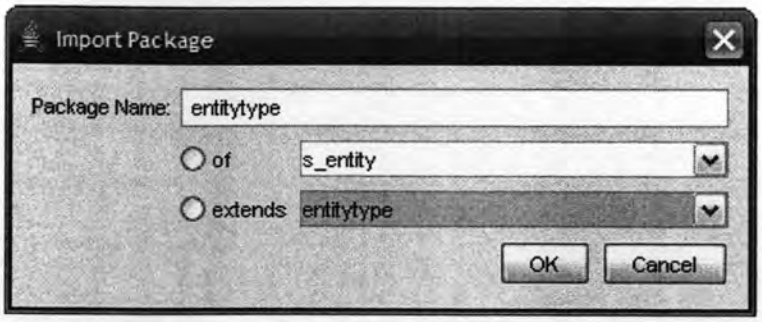
เมื่อผู้ใช้ต้องการสร้างอะตอมยูนิต ผู้ใช้สามารถเลือกเมนูทูลบาร์ Atom ระบบจะแสดงหน้าจอให้ผู้ใช้ใส่ข้อมูลดังรูปที่ ก.6 ซึ่งประกอบด้วย

- Atom Name ชื่อของอะตอมยูนิต
- Imports กลุ่มของแพ็กเกจซิกเนเจอร์ที่ยูนิตนี้ต้องการอิมพอร์ตเข้ามาใช้ในการอ้างถึง
- Exports กลุ่มของแพ็กเกจซิกเนเจอร์ที่ยูนิตนี้ เอ็กซ์พอร์ตออกมาเพื่อให้ยูนิตอื่นสามารถอ้างถึงได้
- Bind Package การผูกแพ็กเกจพารามิเตอร์ที่ได้ประกาศไว้ในแพ็กเกจซิกเนเจอร์



รูปที่ ก.6 หน้าจอการสร้างอะตอมยูนิต

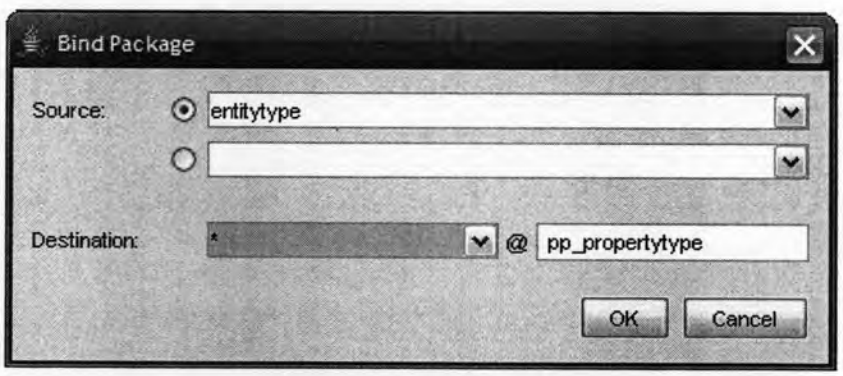
เมื่อผู้ใช้ต้องการเพิ่มแพ็กเกจซิกเนเจอร์ให้กับยูนิตให้กดปุ่มเครื่องหมาย + ที่ปรากฏอยู่ในส่วนการอิมพอร์ต ก็จะปรากฏหน้าจอดังรูปที่ ก.7 ซึ่งผู้ใช้จะต้องตั้งชื่อให้กับแพ็กเกจซิกเนเจอร์และเลือกแพ็กเกจซิกเนเจอร์ที่ต้องการ หรือ สามารถเลือกสืบทอดจากแพ็กเกจซิกเนเจอร์อื่นๆได้ และในกรณีเอ็กซ์พอร์ตก็จะใช้วิธีการเดียวกัน



รูปที่ ก.7 หน้าจอการเลือกอิมพอร์ตซิกเนเจอร์ให้กับอะตอมยูนิต

หลังจากที่ได้กำหนดแพ็กเกจซิกเนเจอร์ให้กับส่วนของการอิมพอร์ตและเอ็กซ์พอร์ตแล้ว ผู้ใช้จะต้องทำการผูกแพ็กเกจ ที่ได้ประกาศเป็นพารามิเตอร์ไว้ในแพ็กเกจซิกเนเจอร์เข้าด้วยกัน ดังรูปที่ ก.8 โดยมีรายละเอียด ดังนี้

- Source หมายถึง คลาสที่อยู่ในแพ็กเกจซิกเนเจอร์นี้ จะถูกส่งไปให้กับปลายทางเพื่อบอกว่าคลาสที่อ้างถึงของปลายทางคือคลาสเหล่านี้นั่นเอง นอกจากนั้นผู้ใช้สามารถเลือกแพ็กเกจใดๆก็ตามปรากฏอยู่ใน Classpath ได้ ในตัวเลือกที่สองของ Source
- Destination หมายถึง แพ็กเกจซิกเนเจอร์และแพ็กเกจพารามิเตอร์ (ชื่อหลังเครื่องหมาย @) ปลายทางที่ต้องการรับค่า และสำหรับเครื่องหมาย * เป็นการบอกว่าแพ็กเกจซิกเนเจอร์ต้นทางจะถูกส่งให้กับทุกๆแพ็กเกจซิกเนเจอร์ปลายทางที่มีชื่อพารามิเตอร์เดียวกันกับที่ระบุไว้



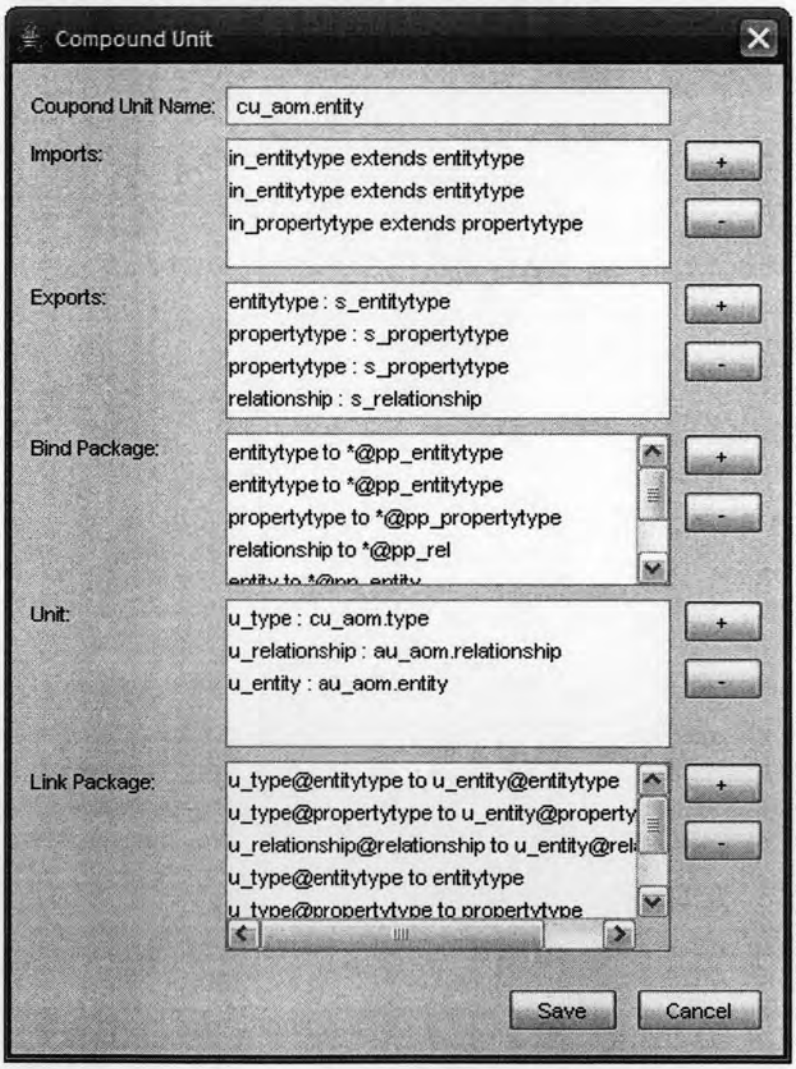
รูปที่ ก.8 หน้าจอการผูกแพ็กเกจ

หลังจากที่ผู้ใช้ได้กรอกรายละเอียดของอะตอมยูนิตแล้ว ให้ทำการกดปุ่ม Save เพื่อบันทึก โปรแกรมก็จะทำการสร้างอะตอมยูนิตออกมา ซึ่งจะปรากฏไฟล์ของยูนิตอยู่ใน Working Directory

ก.4. การสร้างคอมพาวด์ยูนิต

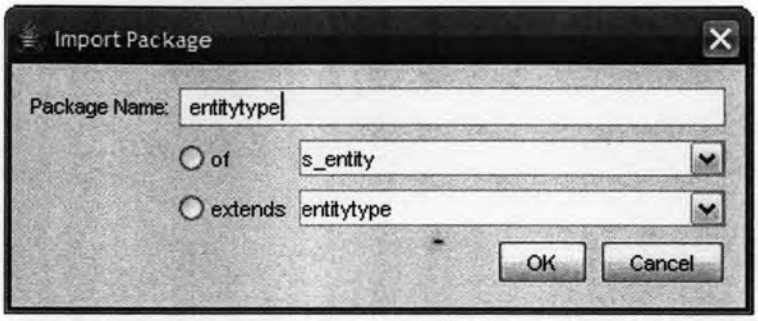
เมื่อผู้ใช้ต้องการสร้างคอมพาวด์ยูนิต ผู้ใช้สามารถเลือกเมนูทูลบาร์ Compound ระบบจะแสดงหน้าจอให้ผู้ใช้ใส่ข้อมูลดังรูปที่ ก.9 ซึ่งประกอบด้วย

- Compound Name ชื่อของอะตอมยูนิต
- Imports กลุ่มของแพ็กเกจซิกเนเจอร์ที่ยูนิตนี้ต้องการอิมพอร์ตเข้ามาใช้ในการอ้างถึง
- Exports กลุ่มของแพ็กเกจซิกเนเจอร์ที่ยูนิตนี้ เอ็กซ์พอร์ตออกมาเพื่อให้ยูนิตอื่นสามารถอ้างถึงได้
- Bind Package การผูกแพ็กเกจพารามิเตอร์ที่ได้ประกาศไว้ในแพ็กเกจซิกเนเจอร์
- Unit เป็นกำหนดว่าคอมพาวด์ยูนิตนี้ ภายในประกอบด้วยยูนิตใดบ้าง
- Link Package เป็นการกำหนดวิธีการเชื่อมต่อยูนิตที่ประกาศไว้เข้าด้วยกัน รวมถึงการเชื่อมต่อแพ็กเกจในส่วนของการอิมพอร์ตและเอ็กซ์พอร์ต

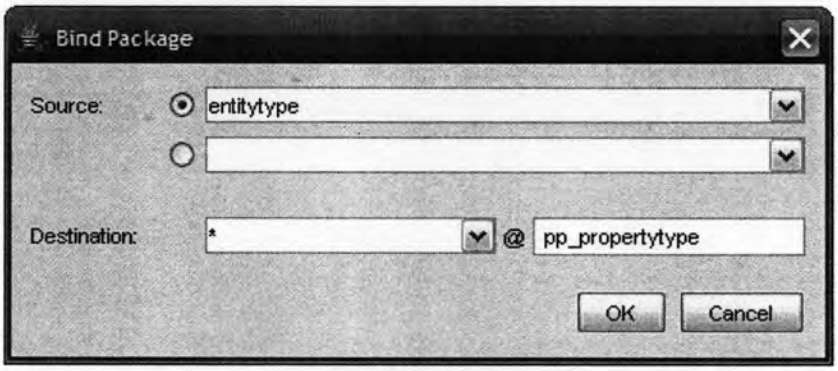


รูปที่ ก.9 หน้าจอการสร้างคอมพาวด์ยูนิต

ในส่วน 4 ส่วนแรกของการสร้างคอมพาวด์ยูนิต นั้นจะเหมือนกับการสร้างอะตอมยูนิตได้แก่ ส่วนการกำหนดชื่อ อิมพอร์ต เอ็กซ์พอร์ต และการผูกแพ็กเกจ ดังรูปที่ ก.10 และ ก.11

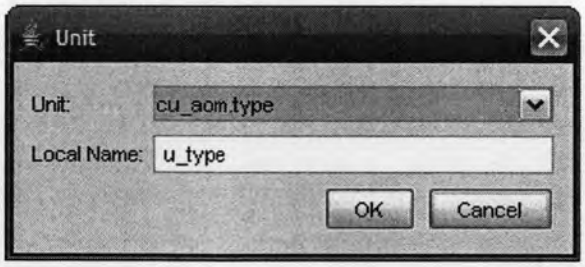


รูปที่ ก.10 หน้าจอการเลือกอิมพอร์ตแพ็กเกจของคอมพาวด์ยูนิต



รูปที่ ก.11 หน้าจอการผูกแพ็กเกจที่ต้องการ

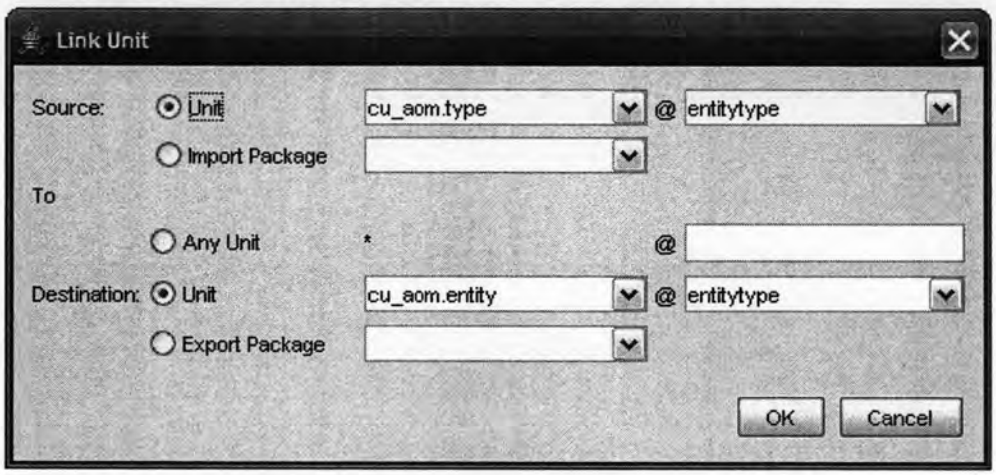
เนื่องจากคอมพาวด์ยูนิตนั้นสามารถประกอบกันขึ้นมากจากยูนิตใดๆก็ได้ ดังนั้น ภายในคอมพาวด์ยูนิตจะต้องมีส่วนที่ประกาศยูนิตที่ต้องการนำมาประกอบกัน ซึ่งผู้ใช้จะต้องกดปุ่มเครื่องหมาย + ด้านขวา ก็จะปรากฏหน้าจอดังรูปที่ ก.12 ให้ผู้ใช้เลือกยูนิตที่ต้องการพร้อมทั้งกำหนดชื่อให้กับยูนิต



รูปที่ ก.12 หน้าจอการเลือกยูนิตเพื่อประกอบกันเป็นยูนิตใหม่ของคอมพาวด์ยูนิต

หลังจากที่ได้กำหนดว่าภายในประกอบด้วยยูนิตใดแล้ว ผู้ใช้จะต้องทำการเชื่อมต่อยูนิตเข้าด้วยกัน โดยมีหลักคือ ส่วนอิมพอร์ตแพ็กเกจซิกเนเจอร์จะต้องเชื่อมต่อเข้ากับส่วนที่เอ็กพอร์ตซิกเนเจอร์เท่านั้น

- กำหนดแพ็กเกจต้นทางของยูนิต หรือ เลือกแพ็กเกจที่อิมพอร์ตเข้ามา อย่างใดอย่างหนึ่ง
- กำหนดแพ็กเกจของยูนิตปลายทาง หรือ เลือกแพ็กเกจที่ต้องการเอ็กซ์พอร์ต หรือเลือกเครื่องหมาย * และกำหนดชื่อแพ็กเกจพารามิเตอร์ อย่างใดอย่างหนึ่ง

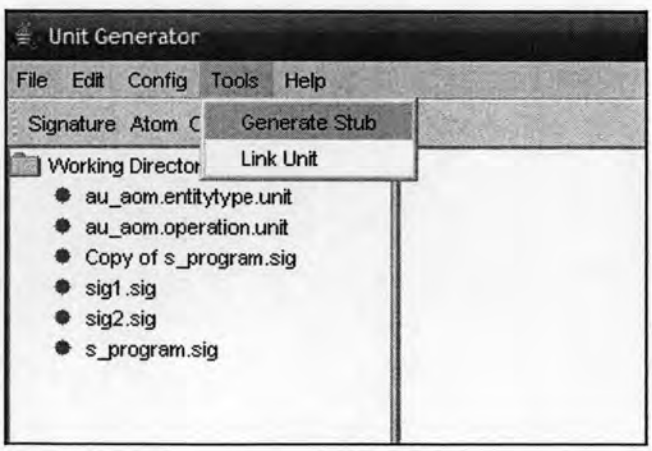


รูปที่ ก.13 หน้าจอการเชื่อมต่อของยูนิตแต่ละยูนิตที่ถูกเลือกมาประกอบกันเป็นคอมพาวด์ยูนิต

หลังจากที่ผู้ใช้ได้กรอกรายละเอียดของคอมพาวด์ยูนิตแล้ว ให้ทำการกดปุ่ม Save เพื่อบันทึก โปรแกรมก็จะทำการสร้างคอมพาวด์ยูนิตออกมา ซึ่งจะปรากฏไฟล์ของยูนิตอยู่ใน Working Directory

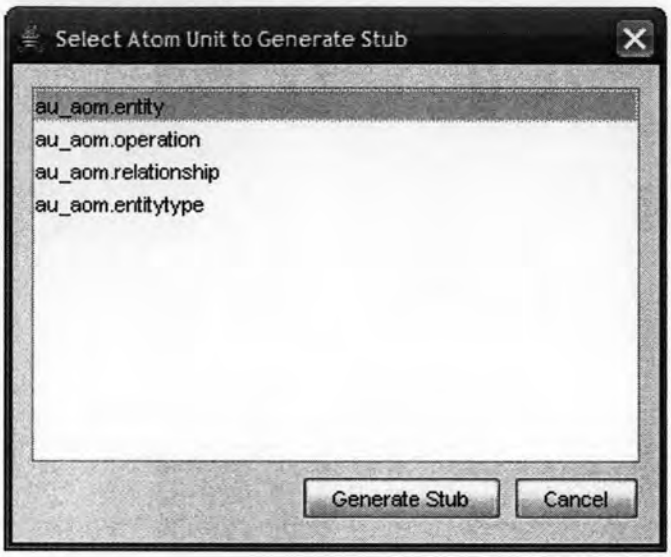
ก.5. การสร้างสตับเพื่อนำไปอิมพลิเมนต์

หลังจากที่ได้อะตอมยูนิตแล้ว ผู้ใช้สามารถเลือกสร้างสตับของอะตอมยูนิตได้ โดยการเลือกที่เมนู Tools -> Generate Stub ดังรูปที่ ก.14



รูปที่ ก.14 เมนูการใช้เครื่องมือสร้างสตับเพื่อนำไปใช้ในการอิมพลิเมนต์ซอร์สโค้ด

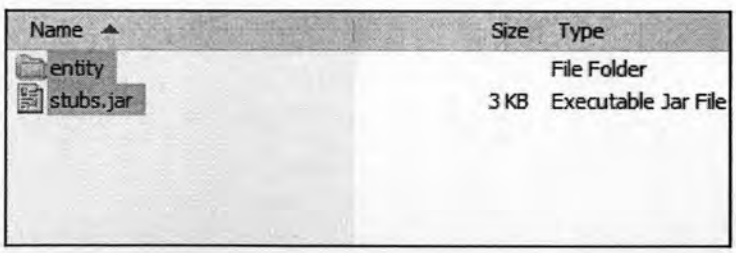
โปรแกรมจะจะแสดงหน้าจอให้ผู้ใช้เลือกยูนิตที่ต้องการดังรูปที่ ก.15 เมื่อเลือกยูนิตแล้วจะต้องกดปุ่ม Generate Stub โปรแกรมก็จะทำการสร้างสตับให้พร้อมทั้งโฟลเดอร์ที่ภายในบรรจุไฟล์ของ .java ที่ยังไม่ได้ทำการอิมพลิเมนต์ให้สมบูรณ์



รูปที่ ก.15 รายการยูนิตที่ผู้ใช้สามารถเลือกสร้างสตับได้

รูปที่ ก.16 แสดงผลที่ได้จากการสร้างสตับของยูนิต au_aolm.entity ซึ่งยูนิตนี้ได้กำหนดไว้ว่าจะเอ็กซ์พอร์ตแพ็กเกจชิกเนเจอร์ชื่อ entity ซึ่งนักพัฒนาจะต้องนำไฟล์ .java ดังกล่าวไปพัฒนาและคอมไพล์ต่อให้สมบูรณ์

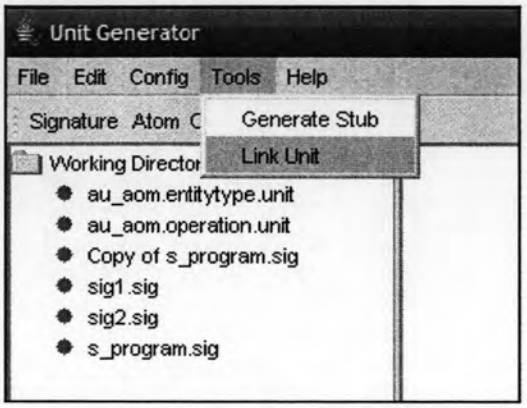
ในการคอมไพล์นั้นผู้ใช้จะต้องนำเอาสตับ (stubs.jar) ไปกำหนดไว้ใน Classpath ด้วย ตัวอย่างเช่น "javac –classpath .;stubs.jar; entity/*.java" ก็เพื่อให้สามารถคอมไพล์การอ้างถึงคลาสที่อิมพอร์ตเข้ามาได้ การทำเช่นนี้จะให้นักพัฒนาสามารถแยกกันพัฒนาแต่ละยูนิตได้โดยอิสระแม้ว่ายูนิตอื่นๆจะยังไม่ถูกพัฒนาขึ้นมาก็ตาม



รูปที่ ก.16 ผลลัพธ์ที่ได้จากการสร้างสตับของอะตอมยูนิตชื่อ au_aom.entity

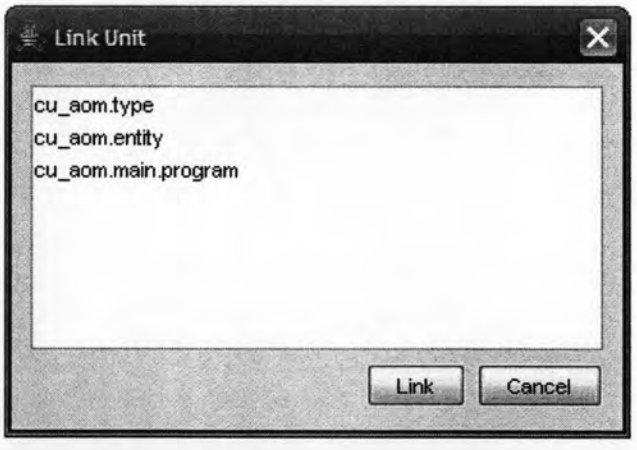
ก.6. การลิงค์ยูนิตเข้าด้วยกัน

หลังจากที่ได้พัฒนาแต่ละยูนิตแล้ว ผู้ใช้สามารถลิงค์แต่ละยูนิตที่กำหนดไว้ในคอมพาวด์ยูนิตเข้าด้วยกัน โดยการเลือกที่เมนู Tools -> Link Unit ดังรูปที่ ก.17

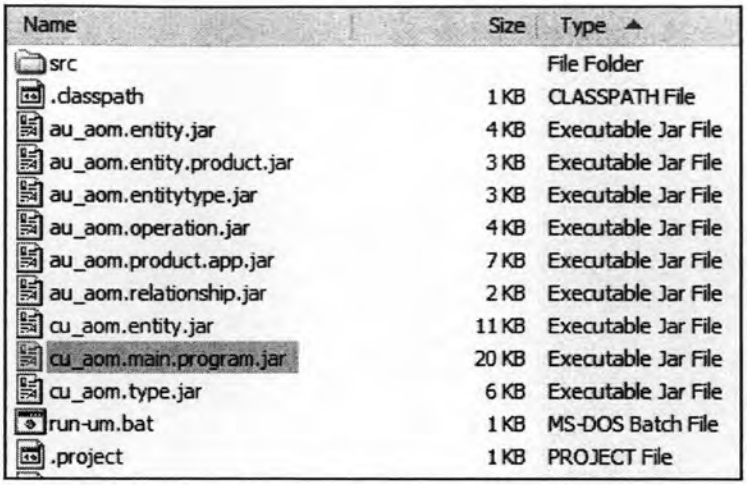


รูปที่ ก.17 เมนูการใช้เครื่องมือเพื่อลิงค์ยูนิตเป็นคอมพาวด์ยูนิต

หลังจากนั้น โปรแกรมจะจะแสดงหน้าจอให้ผู้ใช้เลือกยูนิตที่ต้องการดังรูปที่ ก.18 เมื่อเลือกยูนิตแล้วจะต้องกดปุ่ม Link โปรแกรมก็จะทำการลิงค์ยูนิตให้ ยูนิตผลลัพธ์ที่ได้จะถูกบันทึกลงไว้ใน Working Directory ดังรูปที่ ก.19 ซึ่งเป็นไฟล์ .jar



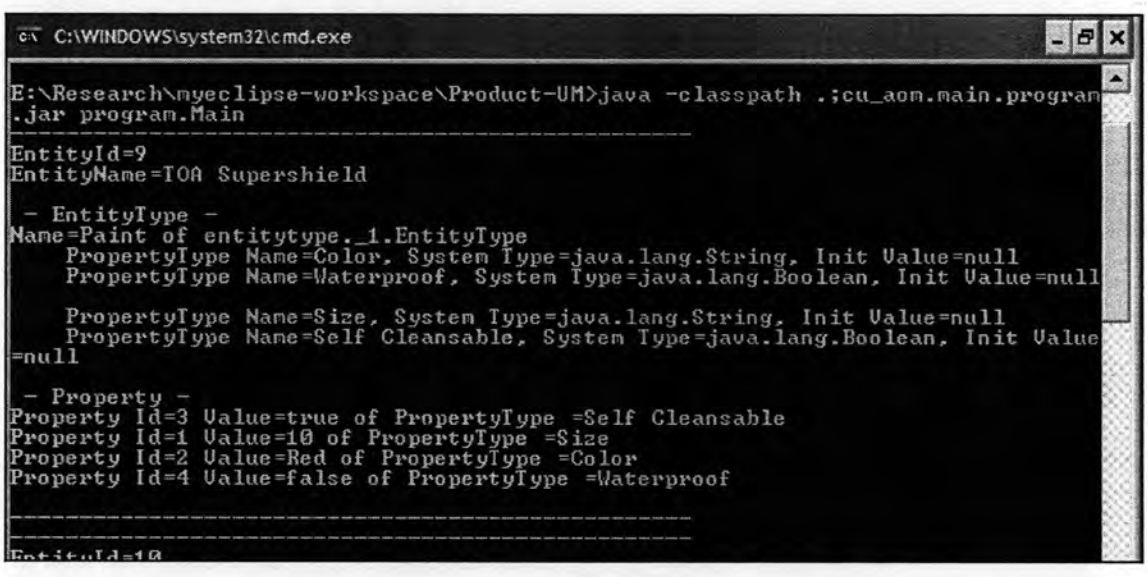
รูปที่ ก.18 รายการคอมพาวด์ยูนิตที่ผู้ใช้สามารถลิงค์ยูนิตที่กำหนดไว้เป็นยูนิตใหม่ได้



Name	Size	Type
src		File Folder
.classpath	1 KB	CLASSPATH File
au_aom.entity.jar	4 KB	Executable Jar File
au_aom.entity.product.jar	3 KB	Executable Jar File
au_aom.entitytype.jar	3 KB	Executable Jar File
au_aom.operation.jar	4 KB	Executable Jar File
au_aom.product.app.jar	7 KB	Executable Jar File
au_aom.relationship.jar	2 KB	Executable Jar File
cu_aom.entity.jar	11 KB	Executable Jar File
cu_aom.main.program.jar	20 KB	Executable Jar File
cu_aom.type.jar	6 KB	Executable Jar File
run-um.bat	1 KB	MS-DOS Batch File
.project	1 KB	PROJECT File

รูปที่ ก.19 ผลลัพธ์ที่ได้จากการลิงค์ยูนิต cu_aom.main.program

ไฟล์ .jar นั้นอยู่ในรูปของไบนารีที่คอมไพล์แล้ว นักพัฒนาสามารถนำไปใช้ในการพัฒนาต่อเป็นยูนิตใหม่หรือนำไปใช้งานได้ทันที ดังรูปที่ ก.20



```
C:\WINDOWS\system32\cmd.exe

E:\Research\myeclipse-workspace\Product-UM>java -classpath .;cu_aom.main.progran
.jar program.Main
-------------------------------------------------
EntityId=9
EntityName=TOA Supershield

 - EntityType -
Name=Paint of entitytype._1.EntityType
    PropertyType Name=Color, System Type=java.lang.String, Init Value=null
    PropertyType Name=Waterproof, System Type=java.lang.Boolean, Init Value=null

    PropertyType Name=Size, System Type=java.lang.String, Init Value=null
    PropertyType Name=Self Cleansable, System Type=java.lang.Boolean, Init Value
=null

 - Property -
Property Id=3 Value=true of PropertyType =Self Cleansable
Property Id=1 Value=10 of PropertyType =Size
Property Id=2 Value=Red of PropertyType =Color
Property Id=4 Value=false of PropertyType =Waterproof

-------------------------------------------------
-------------------------------------------------
EntityId=10
```

รูปที่ ก.20 ตัวอย่างผลลัพธ์ของการใช้งานคอมพาวด์ยูนิต

ภาคผนวก ข.

ซอร์สโค้ดยูนิตโมเดลสำหรับเอโอเอ็ม

ข.1. ซอร์สโค้ดของซิกเนเจอร์

ข.1.1. s_entitytype.sig

```
signature s_entitytype = {
  package pp_util, pp_io;
 public class EntityType implements pp_io.Serializable {
        public EntityType();
        public Integer getEntityTypeId();
        public void setEntityTypeId(Integer entityTypeId);
        public pp_util.Set getEntities();
        public void setEntities(pp_util.Set entities);
        public pp_util.Set getOperationNames();
        public void setOperationNames(pp_util.Set operationNames);
        public String getName();
        public void setName(String name);
        public pp_util.Set getPropertyTypes();
        public void setPropertyTypes(pp_util.Set propertyTypes);
        public Object invokeOperation(String operationName, Object[] parameters);
 }
}
```

ข.1.2. s_operation.sig

```
signature s_operation = {
 package pp_util;
 class Operation extends Object {
        Operation();
        public int getOperationId();
        public void setOperationId(int operationId);
        public String getName();
        public void setName(String name);
        public pp_util.Set getEntityTypes();
        public void setEntityTypes(pp_util.Set entityTypes);
        public static Object invoke(String opName, Object[] parameters);
 }

 interface IOperation {
  public Object operate(Object[] parameters);
 }
}
```

ข.1.3. s_propertytype.sig

```
signature s_propertytype = {
  package pp_entitytype, pp_io;
  public class PropertyType implements pp_io.Serializable {
          public PropertyType();
          public Integer getPropertyTypeId();
          public void setPropertyTypeId(Integer propertyTypeId);
          public pp_entitytype.EntityType getEntityType();
          public void setEntityType(pp_entitytype.EntityType entityType);
          public String getName();
          public void setName(String name);
          public String getSystemDataType();
          public void setSystemDataType(String systemDataType);
          public Object getInitValue();
          public void setInitValue(Object initValue);
          public Class getSystemType();
  }
}
```

ข.1.4. s_relationship.sig

```
signature s_relationship = {
  package pp_entitytype, pp_io, pp_util;
 public class AccountabilityType implements pp_io.Serializable {
        public AccountabilityType();
        public AccountabilityType(String name);
        public Integer getAccTypeId();
        public void setAccTypeId(Integer accTypeId);
        public String getName();
        public void setName(String name);
        public pp_util.Set getAccountabilities();
        public void setAccountabilities(pp_util.Set accountabilities);
        public void addConnectionRule(pp_entitytype.EntityType parent,
pp_entitytype.EntityType child);

 }
}
```

ข.1.5. s_entity.sig

```
signature s_entity = {
 package pp_entity, pp_entitytype, pp_propertytype, pp_rel, pp_util, pp_io;
 public class Entity implements pp_io.Serializable {
        public Entity();
        public Integer getEntityId();
        public void setEntityId(Integer entityId);
        public String getName();
        public void setName(String name);
        public pp_util.Set getEntityTypes();
        public void setEntityTypes(pp_util.Set entityTypes);
        public pp_util.Set getProperties();
        public void setProperties(pp_util.Set properties);
        public pp_util.Set getParentAccountabilities();
        public void setParentAccountabilities(pp_util.Set parentAccountabilities);
        public pp_util.Set getChildAccountabilities();
        public void setChildAccountabilities(pp_util.Set childAccountabilities);
 }
 public class Property implements pp_io.Serializable {
        public Property();
        public Integer getPropertyId();
        public void setPropertyId(Integer propertyId);
        public pp_propertytype.PropertyType getPropertyType();
        public void setPropertyType(pp_propertytype.PropertyType propertyType);
        public Object getValue();
        public void setValue(Object value);
        public pp_entity.Entity getEntity();
        public void setEntity(pp_entity.Entity entity);
 }
```

ข.1.5. s_entity.sig (ต่อ)

```
 public class Accountability implements pp_io.Serializable {
        public Accountability();
        public Accountability(pp_entity.Entity parent, pp_entity.Entity child,
pp_rel.AccountabilityType type);
        public Integer getAccId();
        public void setAccId(Integer accId);
        public pp_rel.AccountabilityType getAccountabilityType();
        public void setAccountabilityType(pp_rel.AccountabilityType accountabilityType);
        public pp_entity.Entity getChild();
        public void setChild(pp_entity.Entity child);
        public pp_entity.Entity getParent();
        public void setParent(pp_entity.Entity parent);
        public static pp_entity.Accountability create(pp_entity.Entity parent, pp_entity.Entity
child, pp_rel.AccountabilityType type);
        public static boolean canCreate(pp_entity.Entity parent, pp_entity.Entity child,
pp_rel.AccountabilityType type);
 }
}
```

ข.2. ซอร์สโค้ดของอะตอมยูนิต

ข.2.1. au_aom.entitytype.unit

```
atom au_aom.entitytype {
 import operation : s_operation;
 export entitytype : s_entitytype;
 export propertytype : s_propertytype;
 bind package
        java.util to *@pp_util,
        java.io to *@pp_io,
        entitytype to *@pp_entitytype;
}
```

ข.2.2. au_aom.operation.unit

```
atom au_aom.operation {
  export operation : s_operation;
  bind package
        java.util to operation@pp_util;
}
```

ข.2.3. au_aom.relationship.unit

```
atom au_aom.relationship {
  import entitytype : s_entitytype;
  export relationship : s_relationship;
  bind package
        entitytype to *@pp_entitytype,
        java.io to *@pp_io,
        java.util to *@pp_util;
}
```

ข.2.4. au_aom.entity.unit

```
atom au_aom.entity {
  import entitytype : s_entitytype;
  import propertytype : s_propertytype;
  import relationship : s_relationship;
  export entity : s_entity;
  bind package
        entity to *@pp_entity,
        java.util to *@pp_util,
        java.io to *@pp_io,
        entitytype to *@pp_entitytype,
        propertytype to *@pp_propertytype,
        relationship to *@pp_rel;
}
```

ข.3. ซอร์สโค้ดของคอมพาวด์ยูนิต

ข.3.1. cu_aom.type.unit

```
compound cu_aom.type {
  export entitytype : s_entitytype;
  export propertytype : s_propertytype;
  bind package
          java.util to entitytype@pp_util,
          java.io to entitytype@pp_io,
          entitytype to propertytype@pp_entitytype,
          java.io to propertytype@pp_io;
}{
  local u_entitytype : au_aom.entitytype, u_op : au_aom.operation;
  link package
    u_op@operation to u_entitytype@operation,
    u_entitytype@entitytype to entitytype,
    u_entitytype@propertytype to propertytype;
}
```

ข.3.2. cu_aom.entity.unit

```
compound cu_aom.entity {
  export entitytype : s_entitytype;
  export propertytype : s_propertytype;
  export relationship : s_relationship;
  export entity : s_entity;
  bind package
          entitytype to *@pp_entitytype,
          propertytype to *@pp_propertytype,
          relationship to *@pp_rel,
          entity to *@pp_entity,
          java.util to *@pp_util,
          java.io to *@pp_io;
} {
  local u_type : cu_aom.type, u_relationship : au_aom.relationship, u_entity : au_aom.entity;
  link package
    u_type@entitytype to u_relationship@entitytype,
    u_type@entitytype to u_entity@entitytype,
    u_type@propertytype to u_entity@propertytype,
    u_relationship@relationship to u_entity@relationship;

  link package
    u_type@entitytype to entitytype,
    u_type@propertytype to propertytype,
    u_relationship@relationship to relationship,
    u_entity@entity to entity;

}
```

ประวัติผู้เขียนวิทยานิพนธ์

นาย นฤชิต ดำรงวิถีธรรม เกิดเมื่อวันที่ 10 กรกฎาคม 2522 ที่จังหวัดตรัง สำเร็จการศึกษาปริญญาวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ในปีการศึกษา 2544 หลังจากสำเร็จการศึกษาได้เข้าทำงานในตำแหน่งนักพัฒนาซอฟต์แวร์ ระหว่างทำงานได้เข้าศึกษาต่อระดับปริญญามหาบัณฑิต ปีการศึกษา 2547 หลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย