สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

เฟรมเวิร์คสำหรับแอพลิเคชั่นบันทึกข้อมูลบนอุปกรณ์พกพา

FRAMEWORK FOR DATA COLLECTION APPLICATION ON MOBILE DEVICE

โดย

นายภูวสิทธิ์ ศรีโภคา

นายวรา สงกรานต์

อาจารย์ที่ปรึกษา

ดร.ชุติเมษฎ์ ศรีนิลทา

เลขหมู่.....
เลขทะเบียน 61376
วัน,เดือน,ปี 17 ก.ค. 2549

.b.....
.i.....

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

ปริญญานิพนธ์ปีการศึกษา 2547

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เฟรมเวิร์คสำหรับแอพลิเคชั่นบันทึกข้อมูลบนอุปกรณ์พกพา

FRAMEWORK FOR DATA COLLECTION APPLICATION ON MOBILE DEVICE

ผู้จัดทำ

1. นาย ภูวสิทธิ์ ศรีโภคา รหัสประจำตัว 44010367
2. นาย วรา สงกรานต์ รหัสประจำตัว 44010419

ชุติเมษฎ์ ศรีนิลทา อาจารย์ที่ปรึกษา

(ดร.ชุติเมษฎ์ ศรีนิลทา)

เฟรมเวิร์สำหรับแอพลิเคชั่นบันทึกข้อมูลบนอุปกรณ์พกพา

นายภูวสิทธิ์ ศรีโภคา 44010367
นายวรา สงกรานต์ 44010419
ดร.ชุติเมษฎ์ ศรีนิลทา อาจารย์ที่ปรึกษา
ปีการศึกษา 2547

บทคัดย่อ

โครงงานนี้เป็นการพัฒนา framework สำหรับ mobile application ที่ใช้ในการเก็บหรือบันทึกข้อมูลซึ่งใช้กันแพร่หลาย โดยจะทำให้สามารถพัฒนา mobile application ที่รองรับงานลักษณะนี้ได้ง่ายและรวดเร็วขึ้น framework จะทำการกำหนดแบบฟอร์มในการบันทึกข้อมูลโดยอ้างอิงกับ XML และใช้เทคนิค XML-relational mapping ในการเชื่อมโยงข้อมูลกับ relational database ขององค์กร โดยข้อมูลจาก database จะถ่ายโอนมายัง mobile device โดยอัตโนมัติตามเงื่อนไขที่กำหนดไว้ เมื่อผู้ใช้เข้าใช้งานและป้อน username และ password ได้ถูกต้อง การแก้ไขและเปลี่ยนแปลงข้อมูล XML จะทำผ่านผ่าน XMLForm ซึ่งเป็น application ที่สร้าง user interface จาก XML configuration file และจะสามารถโอนข้อมูลที่แก้ไขแล้วกลับสู่ database ได้อย่างถูกต้องเมื่อผู้ใช้กดปุ่ม update นอกจากการพัฒนา framework แล้ว โครงงานนี้ยังรวมถึงการพัฒนา application จาก framework นั้นด้วย

FRAMEWORK FOR DATA COLLECTION APPLICATION ON MOBILE DEVICE

Mr. Puwasit Sripoka
Mr. Wara Songkran
Dr. Chutimet Srininta Advisor
Academic Year 2004

ABSTRACT

This project aim to build a framework for mobile application that is used to collect or record data. The framework provides developer an easy and fast way to develop mobile application by using the form that based on XML and use XML-Relational mapping technique to synchronize data with organizational relational database. Required data from relational database will be converted to XML according to predefined conditions and downloaded to mobile device automatically when user connects to the "Sync Server" with valid username and password. XMLForm , the application that runs on Pocket PC will be responsible for modifying XML document through a user interface generated form XML configuration file. When the modification is done, the user can update the data by tapping on the "Update" button. We developed the framework and an application, using such framework.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้คงไม่อาจเสร็จได้ด้วยดี หากไม่ได้รับการช่วยเหลือและร่วมมือจากหลายๆ ฝ่ายด้วยกัน บุคคลแรกที่ต้องกล่าวถึงเพราะเป็นส่วนสำคัญที่ทำให้วิทยานิพนธ์นี้เสร็จลงได้ก็คืออาจารย์ ชุติเมษฎ์ ศรีนิลทา อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ให้ความเอาใจใส่ แนะนำ และช่วยเหลือเสมอมา ซึ่งต้องขอขอบพระคุณเป็นอย่างมาก

ขอขอบพระคุณบุคคลสำคัญที่สุดที่ทำให้ข้าพเจ้ามีวันนี้คือ บิดา มารดา อันเป็นที่เคารพรักยิ่ง ซึ่งได้เลี้ยงดูผู้เขียนมาเป็นอย่างดี และขอขอบคุณเพื่อนๆ พี่ๆ น้องๆ ทุกคนที่ให้ความร่วมมือในด้านต่างๆ จนงานนี้ประสบความสำเร็จได้ด้วยดี จึงขอขอบพระคุณมายัง ณ ที่นี้

ภูวสิทธิ์ ศรีโภคา
วรา สงกรานต์

สารบัญ

สารบัญภาพ

บทที่1

บทนำ

1.1 ความเป็นมาและความสำคัญ

ธุรกิจในปัจจุบันต้องการความคล่องตัวในการทำงาน และเนื่องจากความก้าวหน้าในด้าน mobile computing ทำให้มีการนำ mobile device โดยเฉพาะ PDA มาช่วยในการดำเนินธุรกิจมากขึ้น เนื่องจากคุณสมบัติและข้อจำกัดของ PDA ในปัจจุบันคือมีขนาดพื้นที่แสดงผลเล็กเพื่อให้สะดวกในการพกพา มีหน่วยความจำจำกัดและมีความสามารถในการประมวลผลต่ำ ทำให้ PDA ถูกนำไปใช้ในการเก็บข้อมูลหรือบันทึกข้อมูลในระหว่างการทำงานเป็นส่วนใหญ่

Application ที่ใช้จัดเก็บและบันทึกข้อมูลบน PDA นี้แม้จะมีความแตกต่างกันในแง่ของรูปแบบข้อมูลที่จะจัดเก็บแต่มีกระบวนการในการทำงานเหมือนกัน คือผู้ใช้ download ข้อมูลที่ต้องใช้ในการทำงานลง PDA จากนั้นจึงเปลี่ยนแปลงหรือเพิ่มข้อมูลใน PDA แล้วจึงส่งข้อมูลที่ได้เปลี่ยนแปลง upload คืนให้ server ดังนั้นหากมี framework ที่จะสามารถลดเวลาในการพัฒนา mobile application ประเภทนี้จะทำให้องค์กรสามารถลดเวลาและทรัพยากรที่ใช้ในการพัฒนา application ได้มาก

เนื่องจากข้อมูลในองค์กรส่วนใหญ่มักเก็บไว้ใน relational database ซึ่งทำงานบน computer ที่มีประสิทธิภาพสูง แต่เนื่องจากข้อมูลที่จะใช้บน PDA มีขนาดเล็กและไม่ต้องการคุณสมบัติในการควบคุม constraints ของข้อมูลมากเท่า relational database จึงเลือกใช้ XML มาใช้ในการเก็บข้อมูลชั่วคราวบน PDA และเนื่องจาก XML มีขนาดเล็กและ Library ที่ใช้ในการ process XML ก็มีขนาดเล็กทำให้ประหยัดหน่วยความจำ นอกจากนี้ XML ยังมักถูกใช้เป็นตัวกลางในการแลกเปลี่ยนข้อมูลระหว่างระบบหรือ application ดังนั้นการเลือกใช้ XML จึงทำให้การแลกเปลี่ยนข้อมูลระหว่าง application บน PDA กับระบบอื่นในอนาคตทำได้ง่ายขึ้น

โดย framework นี้จะสามารถ

1. ให้นักพัฒนากำหนดรูปแบบของ form ที่ใช้ในการบันทึกข้อมูลในรูปแบบ XML
2. ถ่ายโอนข้อมูลระหว่าง XML ที่ใช้บน PDA กับ relational database ใน server ขององค์กรได้โดยผ่าน mapping file ที่กำหนดไว้ในรูปแบบ XML
3. เมื่อผู้ใช้ติดต่อเข้าสู่ระบบผ่าน PDA ระบบจะส่งข้อมูลให้ PDA ตามที่เงื่อนไขที่กำหนดไว้ในรูปแบบ XML
4. ผู้ใช้สามารถเลือก update ข้อมูลบางส่วนได้โดยระบบจะส่งเฉพาะ XML ส่วนที่มีการแก้ไขหรือเพิ่มเติมกลับมายัง Server

1.2 วัตถุประสงค์ของปริญญานิพนธ์

1. เพื่อพัฒนาระบบเก็บและบันทึกข้อมูลโดยใช้ PDA
2. เพื่อศึกษาถึงการนำ XML มาใช้เก็บข้อมูลชั่วคราวใน application
3. เพื่อศึกษาถึงการเชื่อมโยงข้อมูลระหว่าง XML กับ relational database

1.3 ขอบเขตของปริญญานิพนธ์

1. พัฒนา framework สำหรับ mobile application ที่ใช้ในการบันทึกข้อมูล
2. พัฒนา application โดยใช้ framework ในข้อ 1

1.4 วิธีการดำเนินงาน

1. ศึกษาเทคโนโลยีเกี่ยวข้องที่จำเป็นสำหรับการนำไปใช้ในโครงการ
2. ศึกษา platform ที่จะใช้ในการพัฒนา application บน Pocket PC
3. ศึกษาและทดลองพัฒนา Win32 application บน Pocket PC โดยใช้ Python programming language
4. ศึกษาและทดลองการถ่ายโอนข้อมูลระหว่าง relational database และ XML
5. ออกแบบ XML configuration file ทั้งหมดที่ใช้ใน framework
6. นำ component ของระบบทั้งหมดมาประกอบเข้าด้วยกัน
7. สร้าง Management Interface ของระบบ

บทที่ 2
ทฤษฎี

2.1. ภาษา XML (Extensive Markup Language)

XML ถูกกำหนดโดยกลุ่มทำงาน XML ของสถาบัน World Wide Web Consortium (W3C) กลุ่มทำงานกลุ่มนี้ได้บรรยายถึงภาษา XML ไว้ว่า Extensible Markup Language เป็นฟอร์แมตที่อธิบายถึงรายละเอียดของโครงสร้างและแบบของข้อมูลเป็นภาษาหรือชุดคำสั่งเกี่ยวกับข้อมูลบนเว็บ ที่ให้การพัฒนาและมีศักยภาพในส่วนของโครงสร้างข้อมูลจากหลากหลายแอพพลิเคชันมานำเสนอบนเครื่องเดสก์ทอป ด้วย XML จะทำให้การจัดการข้อมูลหรือเรียกใช้ข้อมูลจากแอพพลิเคชันต่างๆ จะเข้าสู่มาตรฐานเดียวกัน

XML และ HTML จะเป็นส่วนหนึ่งของภาษา SGML ซึ่ง XML จะให้รายละเอียดเกี่ยวกับข้อมูล เช่น ชื่อเมือง อุณหภูมิ ความกดอากาศ ส่วน HTML เป็นการกำหนด tag ต่างๆ ที่จะกำหนดรูปแบบการแสดงผลของข้อมูลบนหน้าเว็บ ซึ่งข้อมูลจะสามารถแสดงออกมาได้หลายรูปแบบ ขึ้นอยู่กับการกำหนดของ HTML และในปัจจุบันนี้ ด้วย XML จะมีการให้รายละเอียดของเนื้อหาเอกสารที่เรียกว่า Document Type Definition (DTD) ที่ให้รายละเอียดเกี่ยวกับตัวเอกสารว่าจะแสดงหรือซ่อนส่วนไหนของเอกสารบ้าง ซึ่ง DTD จะเป็นส่วนที่เพิ่มเติมสำหรับ XML ถ้าหากมีการส่งข้อมูลในรูปแบบ DTD ก็จะรู้กันว่าเป็น XML มีความหมายหลาย ๆ คำที่ อธิบายลักษณะของ XML

2.1.1 ลักษณะโครงสร้างของ XML

XML เป็นการใช้ข้อความเพื่อบ่งบอกโครงสร้างของเอกสาร พิจารณาตัวอย่างรูปแบบโครงสร้างของหนังสือ เมื่อหนังสือประกอบด้วยจำนวนบท 2 บท ในแต่ละบทประกอบด้วยเนื้อความ

```
Begin Book
        Begin Chapter 1
                Text for Chapter 1
        End Chapter 1
        Begin Chapter 2
                Text for Chapter 2
        End Chapter 2
End Book
```

หนังสือที่มีอยู่ในปัจจุบัน จะมีโครงสร้างที่มีรายละเอียดที่ซับซ้อนมากกว่านี้ เช่น บทนำ, สารบัญ, เป็นต้น เช่นเดียวกัน ภายในส่วนเนื้อความ ยังประกอบด้วยโครงสร้างย่อย คือ ย่อหน้า (paragraph) แต่ละย่อหน้ายังประกอบขึ้นจาก ประโยค คำ และตัวอักษรด้วย

ลักษณะของเอกสาร XML นั้น สามารถอธิบายโดยใช้ตัวอย่างที่ 1 ได้ ดังนี้

ตัวอย่างที่ 1

```
<?xmlversion="1.0"encoding="windows-874"?>
<mali>
        <malisorn>ขึ้นต้นด้วยมะลิซ้อน</malisorn>
        <malila>พอแตกใบอ่อนเป็นมะลิลา</malila>
</mail>
<?xml-stylesheet type="text/xsl" href=""?>
```

บรรทัดที่ 1 นั้นหมายความว่าเราประกาศเอกสารนี้เป็นเอกสาร XML และมีการเข้ารหัสอักขระเป็น windows-874 เพื่อให้ใช้ภาษาไทยได้ จากนั้นเราจะเห็น สิ่งที่เราคุ้นเคยที่เราเรียกว่า tag แต่จริงๆแล้ว ในภาษา XML จะแบ่ง โครงสร้างเป็น 2 ส่วนใหญ่ ๆ คือ tag และ element สามารถอธิบายเพิ่มเติมได้จาก ตัวอย่างที่ 2

ตัวอย่างที่ 2

```
<root>
        <element>
                <tag></tag>
        </element>
</root>
```

ความหมายของ Tag กับ Element

2.1.1.1 Tag

สำหรับใน XML แล้ว tag มีความหมายในลักษณะเดียวกับที่ใช้ใน HTML tag คือข้อความที่อยู่ระหว่างสัญลักษณ์<และ>

Tagเปิด(Start tag)

<book>

จากตัวอย่างที่แสดง ด้านบนถูกเรียกว่า tag เปิด ดังนั้น tag เปิดจึงมีสัญลักษณ์คือ <...>

Tagปิด(End Tag)

</book>

Tag ที่ถูกเรียกว่า tag ปิด ต่อเมื่อใน tag มีเครื่องหมาย / อยู่หลังสัญลักษณ์ < ดังนั้นลักษณะของ tag ปิดจึง มีรูปแบบคือ </...> หากพิจารณาระหว่าง tag เปิดกับ Tag ปิดแล้ว ข้อแตกต่างอีกข้อหนึ่งคือ tag เปิด เป็น tag ที่สามารถใส่ข้อมูล attribute ลงไปภายใน tag ได้ แต่ tag ปิดจะไม่ทำกัน

2.1.1.2 Element

ในที่นี้คือโครงสร้างหลักของ XML ซึ่งอยู่ในรูปของ tag เช่นเดียวกัน ตามตัวอย่างข้างบนelement คือ

```
<root>
        <element>
        </element>
</root>
```

จะมีลักษณะซ้อนกันเป็นชั้นๆ โดย root element เป็น element แรกสุดของเอกสาร XML

Element ใช้เป็นส่วนประกอบของเนื้อหาของเอกสาร XML และ attribute เพื่อให้ผู้อ่านได้เข้าใจความหมายของศัพท์หลายๆ คำศัพท์ ก่อนที่จะนำไปใช้ในการสร้าง XML ซึ่งจำเป็นอย่างมาก พิจารณา คำจำกัดความของ element

<chap number="1">Text for Chapter 1</chap>

สังเกตจากลักษณะ รูปทางด้านบน ทั้งหมดตั้งแต่ <chap...จนถึง </chap> ถูกเรียกว่า element หรือถ้ามองง่ายๆคือ element เริ่มต้นที่ tag เปิด และสิ้นสุดที่ tag ปิดใน tag คำสั่งเดียวกัน

2.1.1.3 Content

เนื้อความ หรือ content ถือได้ว่าเป็นข้อมูลเพื่อใช้ในการแสดง ให้ผู้อ่านเอกสารได้เห็น หรือกล่าวอีกนัยหนึ่งคือ content อยู่หลัง tag เปิด และจบที่ก่อนถึง tag ปิดนั่นเอง

2.1.1.4 Attribute

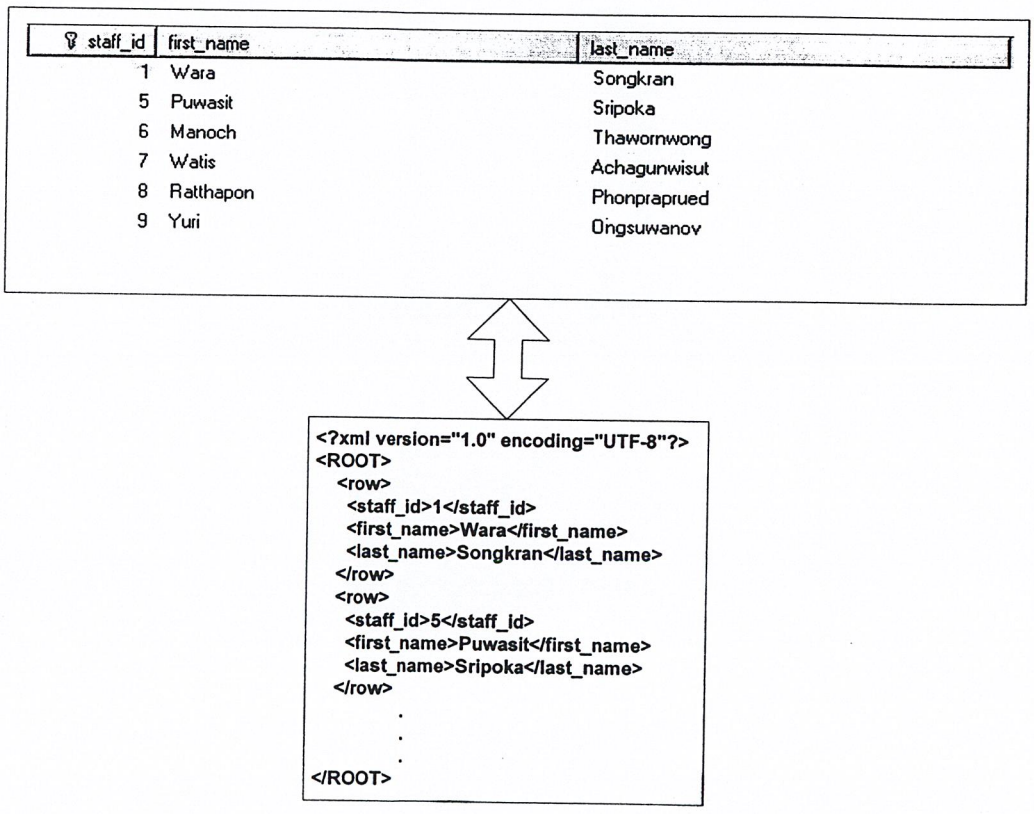
Attribute คือข้อมูลความหมายเพิ่มเติม แน่นอน ถ้ากล่าวถึงสิ่งของ หากเราไม่ได้ให้ความหมาย เพิ่มเติมเราก็จะไม่รู้ และแยกแยะสิ่งที่เหมือนกันออกจากกันได้ เช่น ถ้าพูดถึง บท ในหนังสือ แค่นี้เป็นความหมายโดยรวม แต่ถ้าบอกว่า บทที่ 1 ในหนังสือ เลข 1 ในที่นี้คือความหมายเพิ่มเติมให้กับบท attribute จะ ถูกบรรจุอยู่ภายใน tag เปิด และ attribute นี้ในบางครั้ง อาจจะมีหรือไม่มีก็ได้ รวมถึงถ้ามีอาจมีได้มากกว่าหนึ่งตัวใน tag เปิด

2.2 Mapping XML to Relational Databases

XML เป็นมาตรฐานที่แพร่หลายในการแลกเปลี่ยนข้อมูลระหว่าง application โดยเฉพาะอย่างยิ่ง internet-based application ทำให้มีความต้องการที่จะ publish ข้อมูลที่แต่ละองค์กรมีอยู่แล้วเป็น XML แต่เนื่ององค์กรธุรกิจส่วนใหญ่เก็บข้อมูลไว้บน relational database ดังนั้นจึงมีความจำเป็นในการ map ข้อมูลระหว่าง XML และ relational database

2.2.1 Table-based Mapping

เป็นวิธีการที่ใช้กันอย่างแพร่หลายโดยเฉพาะอย่างยิ่งใน database administrator tool table-based mapping เป็นการ map physical structure ของ relational table กับ XML structure โดยอาจจะ map กับ relational table 1 table หรือ set ของ relational table ก็ได้ วิธีนี้เป็นวิธีที่ง่ายเนื่องจากเป็นการเปรียบเทียบโครงสร้างของ relational table และ result set ของ table แล้วนำมาสร้างเป็น XML document โดยตรง



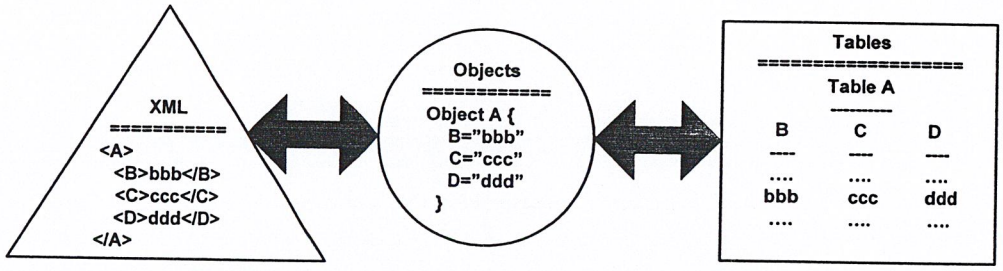
รูปที่ 2-1 รูปแบบของ Table-based Mapping

โดยวิธีนี้อาจจะกำหนดได้ว่า data ใน table column จะถูก map เป็น element หรือ attribute รวมทั้งชื่อของแต่ละ element หรือ attribute ด้วยและอาจจะมีการเพิ่ม table และ column metadata (ชื่อ table , column data type และ constraints ต่างๆ) โดยอาจจะใส่ไว้ในส่วนบนของ XML document หรือเป็น attribute ของแต่ละ element ของ table หรือ column

ข้อดีของ table-based mapping คือความง่ายในการ implement และ code ที่เขียนขึ้นมาก็สามารถทำงานได้เร็วและค่อนข้างมีประโยชน์ในการโอนย้ายข้อมูลระหว่าง database ทีละ table ข้อเสียของ table-based mapping คือมันไม่สามารถ map relational table เป็น XML ในรูปแบบอื่นได้ นอกจากนี้ table-based mapping ยังไม่สามาถ map relationship ระหว่าง table ได้

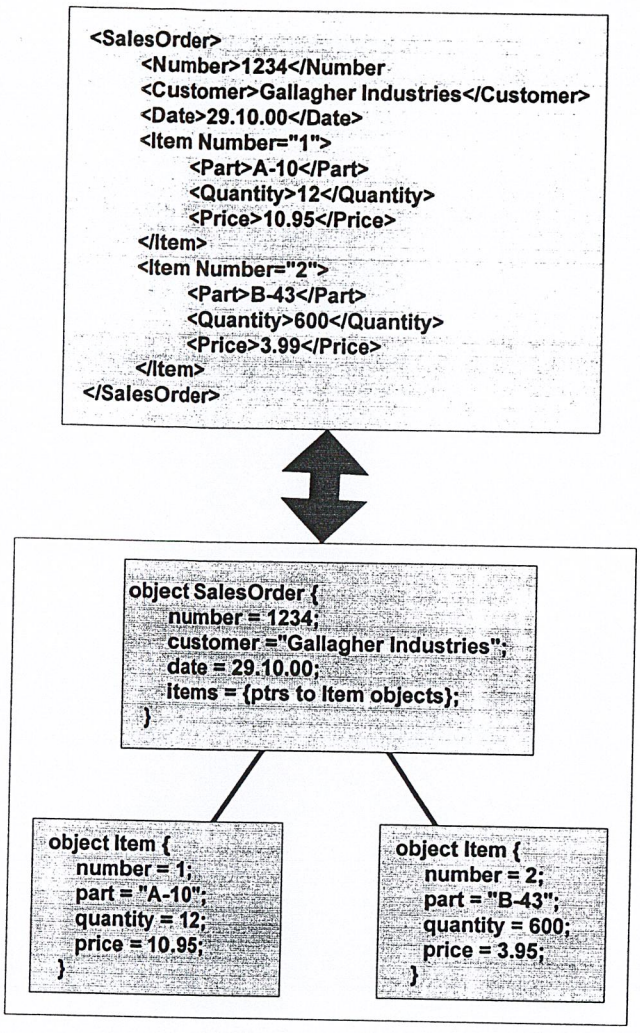
2.2.2 Object-based Mapping

เป็นการ map 2 ขั้นตอนคือ map ระหว่าง XML และ object แล้วจึง map ระหว่าง object และ relational-table อีกขั้นตอนหนึ่ง



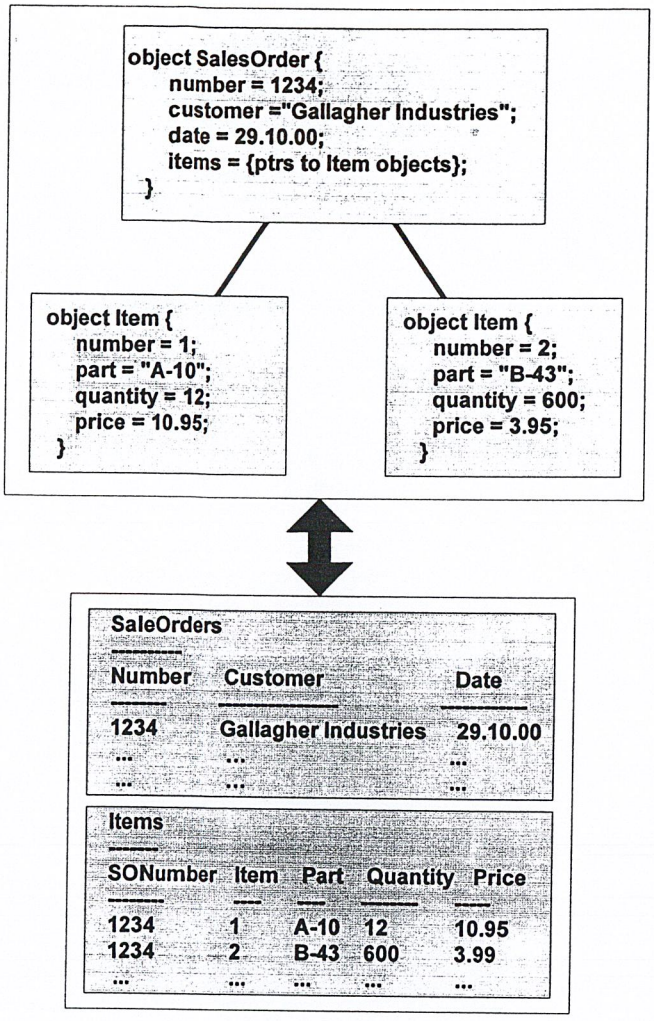
Object-based mapping จะกระทำโดยผ่าน XML-object mapping และ Object-relational mapping

โดยในขั้นตอนของ XML-object mapping จะมอง XML document เป็น tree ของ object ซึ่งมีข้อมูลตรงกับ structure ของ XML document โดยจะ map XML child element หรือ attribute เข้ากับ object properties



รูปที่ 2-2 รูปแบบของ XML-object mapping

ในขั้นตอนของ object-relational mapping จะเป็นการ มอง class เป็น relational table แล้วจึง map class properties เข้ากับ table column และ map object-value properties เข้ากับ foreign key/primary key



รูปที่ 2-3 รูปแบบของ Object-Relational Mapping

Object-based mapping เป็นวิธีการ map XML กับ relational table ที่มีความยืดหยุ่นมากกว่า table-based mapping เนื่องจากสามารถ map relational rable เป็น XML ได้หลายแบบ ทำให้องค์กรสามารถแลกเปลี่ยนข้อมูลกันได้โดยไม่จำเป็นต้องทราบ database schema ของแต่ละฝ่ายแต่ต้องเข้าใจโครงสร้างของ XML ที่จะแลกเปลี่ยนระหว่างกัน

2.3 ลักษณะการใช้งาน Mobile Application

ความก้าวหน้าทางเทคโนโลยีทำให้ mobile device ในปัจจุบันมีประสิทธิภาพที่ดีขึ้นมีหน่วยความจำมากขึ้นสามารถติดต่อกับ network ได้หลากหลายประเภททั้ง Bluetooth, GPRS และ WIFI แต่ในด้านพื้นที่แสดงผลเนื่องจากข้อจำกัดในด้านการพกพา หากทำให้พื้นที่แสดงผลของ mobile device มีขนาดใหญ่มากก็จะทำให้ขนาดและน้ำหนักเพิ่มขึ้นด้วย ทำให้พื้นที่แสดงผลของ mobile device ในปัจจุบันยังคงมีขนาดเล็กและทำให้ application ที่จะใช้กับ mobile device ยังคงมีข้อจำกัดอยู่

จากข้อจำกัดข้างต้นบวกกับความจำเป็นของพนักงานในบางหน้าที่เช่น บุรุษไปรษณีย์, ช่างซ่อมบำรุง หรือพนักงานขับรถรับส่งของ ที่ต้องทำงานนอกสถานที่เกือบตลอดเวลาทำให้ business application ที่ใช้กับ mobile device จะเป็น application ที่ใช้ในการบันทึกและเก็บข้อมูลเป็นส่วนใหญ่

การใช้ mobile device ในการบันทึกข้อมูลแทนการใช้กระดาษทำให้เกิดความคล่องตัวในการทำงานและยังเป็นโครงการที่คืนทุนในระยะสั้น เพราะไม่เพียงแต่จะประหยัดกระดาษและเวลาเท่านั้นแต่ยังประหยัดค่าแรงของพนักงานด้วย เนื่องจากใช้คนน้อยลงในการบันทึกข้อมูลเพราะโดยปกติ หลังจากที่พนักงานที่ทำงานนอก office บันทึกข้อมูลลงกระดาษแล้ว จะต้องมีพนักงานใน office ป้อนข้อมูลจากแบบฟอร์มลง computer อีกครั้งหนึ่ง และทำให้ได้ข้อมูลล่าช้าด้วย

โดยปกติข้อมูลที่ได้จากการ update จะไม่ได้ถูกใช้ในทันทีแต่จะถูกประมวลผลเมื่อสิ้นสุดวันทำงาน ดังนั้นหากมีการนำข้อมูลจาก mobile device เก็บลง database ข้อมูลนั้นจะยังไม่ถูกใช้จนกว่าจะสิ้นสุดวันทำงาน

mobile application ที่ใช้จัดเก็บและบันทึกข้อมูลบน PDA นี้แม้จะมีความแตกต่างกันในแง่ของรูปแบบข้อมูลที่จะจัดเก็บแต่มีกระบวนการในการทำงานเหมือนกัน คือผู้ใช้ download ข้อมูลที่ต้องใช้ในการทำงานลง PDA จากนั้นจึงเปลี่ยนแปลงหรือเพิ่มข้อมูลใน PDA แล้วจึงส่งข้อมูลที่ได้เปลี่ยนแปลง upload คืนให้ server ดังนั้นหากมี framework ที่จะสามารถลดเวลาในการพัฒนา mobile application ประเภทนี้จะทำให้องค์กรสามารถลดเวลาและทรัพยากรที่ใช้ในการพัฒนา application ได้มาก

เนื่องจากข้อมูลในองค์กรส่วนใหญ่มักเก็บไว้ใน relational database ซึ่งทำงานบน computer ที่มีประสิทธิภาพสูง แต่เนื่องจากข้อมูลที่จะใช้บน PDA มีขนาดเล็กและไม่ต้องการคุณสมบัติในการควบคุม constraints ของข้อมูลมากเท่า relational database จึงเลือกใช้ XML มาใช้ในการเก็บข้อมูลชั่วคราวบน PDA และเนื่องจาก XML มีขนาดเล็กและ library ที่ใช้ในการ process XML ก็มีขนาดเล็กทำให้ประหยัดหน่วยความจำ นอกจากนี้ XML ยังมักถูกใช้เป็นตัวกลางในการแลกเปลี่ยนข้อมูลระหว่างระบบหรือ application ดังนั้นการเลือกใช้ XML จึงทำให้การแลกเปลี่ยนข้อมูลระหว่าง application บน PDA กับระบบอื่นในอนาคตทำได้ง่ายขึ้น

บทที่ 3
เครื่องมือที่ใช้ในการพัฒนา

3.1 Python Programming Language 2.3.4

เนื่องจาก framework มีหลายส่วนที่ต้องพัฒนา programming language ที่ใช้จึงต้องสามารถใช้พัฒนา application ได้อย่างรวดเร็ว Python เป็น ภาษาที่เป็น object-oriented เต็มตัว และเนื่องจากมี application code ที่ทำงานทั้งบน PC และ Pocket PC programming language ที่ใช้จึงต้องมีคุณสมบัติ cross-platform ด้วย

นอก Python จะเป็น programming language ที่มีคุณสมบัติดังกล่าวแล้วยังมี library ที่ช่วยให้การพัฒนา enterprise platform ง่ายและเร็วขึ้นเช่น Pyro, XMLObject และ SQLObject

3.2 Python 2.3.4 for Window CE

Python for Window CE เป็น Python interpreter บน platform Pocket PC ของ Microsoft ซึ่ง มีคุณสมบัติทุกอย่างเหมือน Python บน PC ยกเว้น library บางตัวที่ขาดหายไปเนื่องจาก Python library บางตัวเป็นการเขียน wrap C library เอาไว้ และยังไม่มีการ port ไปบน platform Pocket PC

Python เป็น platform ที่เหมาะสำหรับพัฒนา mobile application เนื่องจาก

1. Non-proprietary Python เป็น open source programming language และสามารถ redistribute ได้โดยไม่ต้องใช้ commercial license (Java CDC virtual machine ที่ถูกที่สุดบน Pocket PC คือ IBM J9 คิดค่า license 6 dollar per installed device ราคาเมื่อ มีนาคม 2005)
2. Cross-platform เนื่องจากบน mobile platform อื่นเช่น Palm, Linux mobile และ Nokia Series 60 ต่างก็มี Python interpreter
3. รองรับ native user interface บน Pocket PC ผ่าน library pywin32 ทำให้ user interface ของ application ที่พัฒนาด้วย Python บน Pocket PC มี look and feel เหมือน Win32 application ทั่วไปและทำงานได้เร็ว

3.3 Pyro

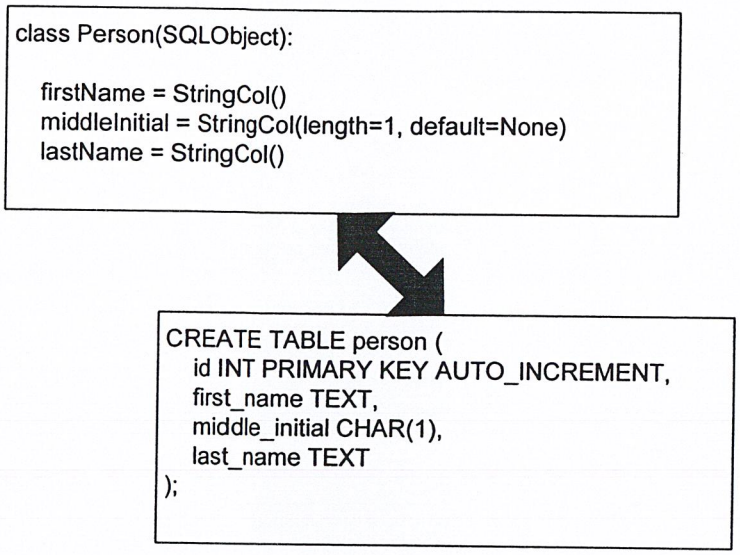
Python remote object (Pyro) เป็น distribute object library ซึ่งพัฒนาโดยใช้ Python ทั้งหมดทำให้สามารถทำงานได้บน Python interpreter บนทุก platform จึงสามารถใช้งานได้บน Python for Window CE ด้วย

Pyro ทำให้นักพัฒนาสามารถพัฒนา distribute system หรือระบบที่ต้องใช้ remote procedural call ได้อย่างรวดเร็วเนื่องจาก remote object ที่พัฒนาด้วย Pyro จะเหมือนกับ Python object ทั่วไปและสามารถ call method ได้เหมือน Python object ทั่วไป นอกจากนี้ Pyro ยังรองรับ authentication และ SSL encryption ด้วย

ผู้พัฒนาโครงการใช้ Pyro ในการติดต่อระหว่าง client และ server รวมทั้งใช้ระหว่าง management interface กับ server ด้วย

3.4 SQLObject

เป็น Python object-relation mapping library ที่ใช้งานได้ง่ายและรองรับ one-to-many mapping และ transparent many-to-many mapping ด้วย SQLObject รองรับ RDBMS สำคัญๆ หลายตัวผ่าน Python DB:API เช่น Oracle, MySQL, Sybase, MS-SQL, PosgreSQL และ Firebird

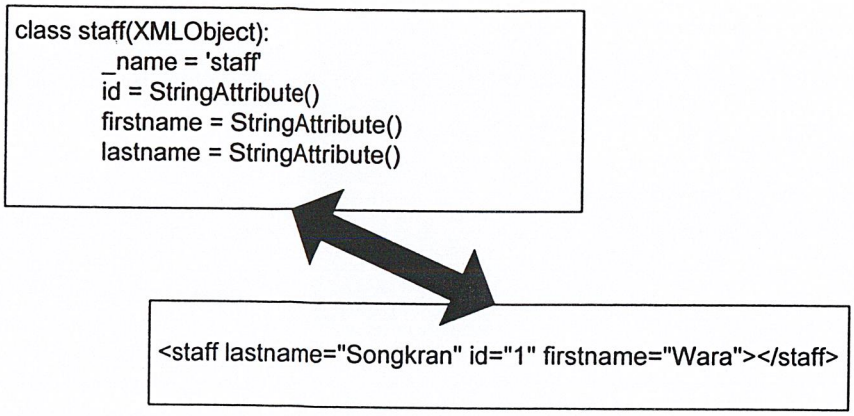


รูปที่ 3-1 ตัวอย่างของ SQLObject Class ที่สัมพันธ์กับ Relational Table

ในการพัฒนาโครงการนี้ผู้พัฒนาได้ใช้ SQLObject เป็น library ช่วยในการพัฒนา XML-relational mapping library ของตนเอง

3.5 XMLObject

เป็น Python XML-object mapping library ทำให้ผู้ใช้สามารถจัดการกับ XML document ได้เหมือนกับจัดการกับ Python object ธรรมดา โดย XMLObject จะ map XML element เข้ากับ Python class และ map class properties เป็น child node หรือ XML attribute

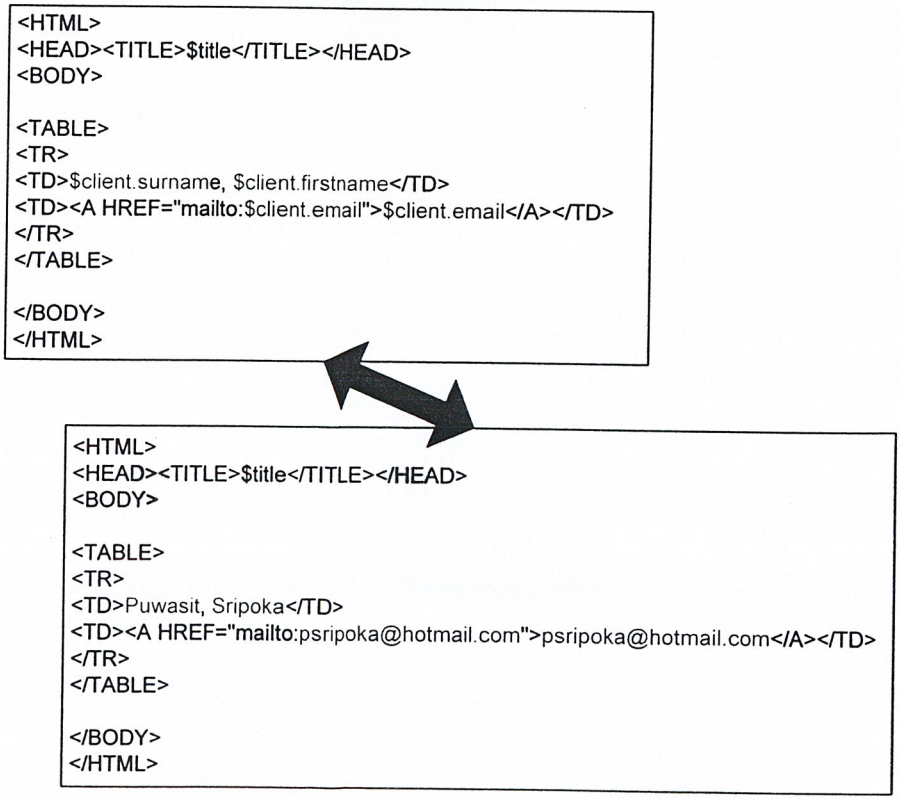


รูปที่ 3-2 ตัวอย่างของ XMLObject ที่สัมพันธ์กับ XML Element

ในการพัฒนาโครงการนี้ผู้พัฒนาได้ใช้ XMLObject เป็น library ช่วยในการพัฒนา XML-relational mapping library ของตนเอง

3.6 Cheetah Template Engine

เป็น template engine และ code generator ที่เขียนด้วย Python Cheetah เป็น template engine ที่ใช้งานง่ายเนื่องจาก syntax คล้ายกับ syntax ของ Python



รูปที่ 3-3 ตัวอย่างของ Template ที่ใช้ในการสร้าง HTML และผลลัพธ์ที่ได้

Cheetah เป็น library สำคัญที่ใช้ในการพัฒนาโครงการนี้เนื่องจาก module สำคัญ 2 ตัวคือ Data Exchanger และ Authenticator ใช้ Cheetah เป็น code generator

3.7 PyXML

เป็น Python XML library ซึ่งรองรับ W3C DOM, minidom, SAX, Pull DOM, XML Marshaller และ XPath

ในโครงการนี้ PyXML เป็น library สำหรับ process XML บนฝั่ง Server

3.8 elementtree

เนื่องจาก PyXML เป็น XML library ที่มี feature ครบถ้วนแต่มีขนาดใหญ่ถึง 5.14 MB และยังทำงานได้ช้าบน Pocket PC ผู้พัฒนาโครงการจึงเลือกใช้ elementtree ซึ่งเป็น Python XML library อีกตัวหนึ่งที่มี feature น้อยกว่า PyXML แต่ทำงานได้เร็วกว่าเป็น XML library ที่ใช้งานบน Pocket PC

3.9 pywin32

เป็น Python library ที่เขียน wrap Win32 API ของ Microsoft Window ทำให้ Python programmer สามารถสร้าง ฟpplication ที่ใช้ Win32 user interface ได้และยังสามารถเรียกใช้ function และ class ใน Win32 API ได้ทั้งหมดรวมทั้งการเรียกใช้และสร้าง COM object ด้วย

pywin32 เป็น library ที่ใช้สร้าง user interface บน Pocket PC ในโครงการนี้

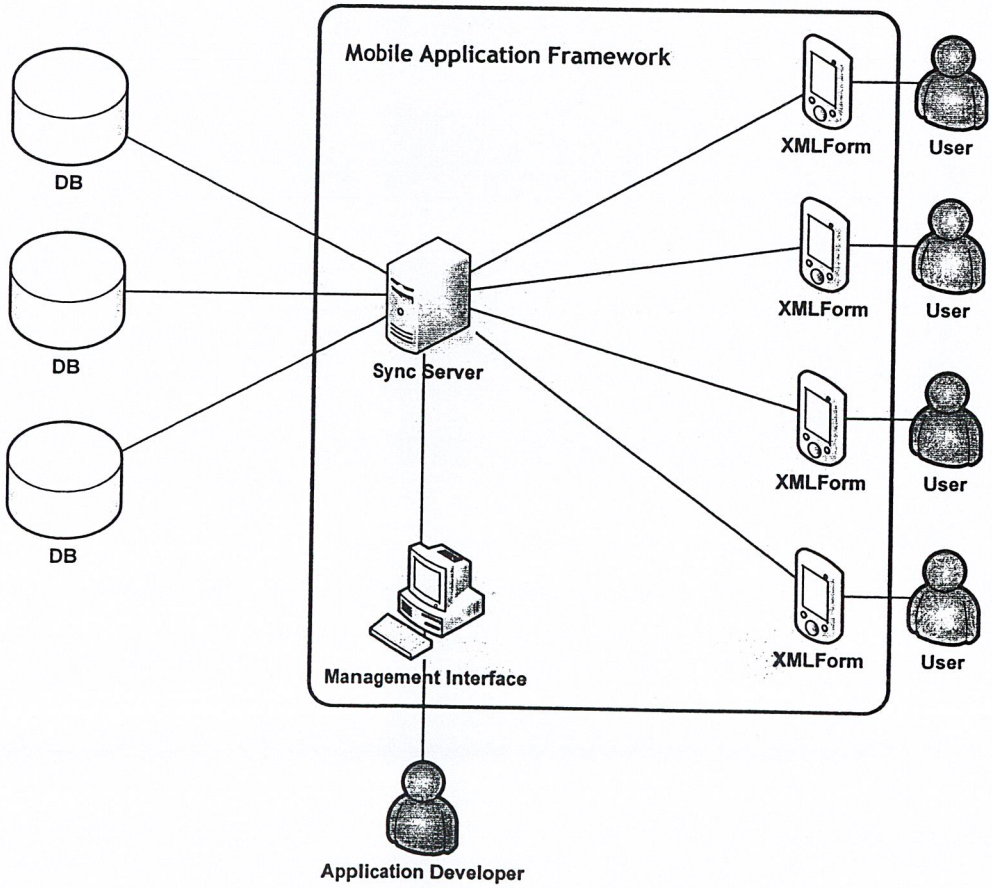
บทที่ 4

การออกแบบระบบ

4.1 Mobile Application Framework Requirement

1. สามารถให้นักพัฒนากำหนดแบบฟอร์มสำหรับให้ผู้ใช้กรอกข้อมูลได้ง่าย
2. สามารถกำหนดรูปแบบการเชื่อมโยงข้อมูลระหว่าง XML บน Pocket PC และ relational database ขององค์กร โดยใช้ XML configuration file ได้
3. Server ที่ใช้ในการแลกเปลี่ยนข้อมูลต้องสามารถรองรับการเชื่อมต่อจาก user หลายราย พร้อมกันได้
4. ระบบต้องสามารถรองรับการเชื่อมโยงข้อมูลระหว่าง database และ XML ได้หลายแบบ พร้อมกัน
5. ต้องมี interface ให้นักพัฒนาสามารถกำหนดและเพิ่มรูปแบบการเชื่อมโยงข้อมูลระหว่าง XML และ database ได้โดยต้องสามารถทำได้ผ่านระบบ network

4.2 โครงสร้างและการทำงานของระบบ



รูปที่ 4-1 ส่วนประกอบหลักของ Mobile Application Framework และผู้ใช้งาน

ระบบประกอบด้วย

1. XMLForm เป็น application บน PDA ซึ่งจะสร้าง UI จาก XMLForm configuration file (Form.xml) และ XML lookup file (Lookup.xml) และให้ user สามารถแก้ไขข้อมูลใน XML file ได้ รวมทั้งทำการส่งข้อมูลที่ผู้ใช้แก้ไขให้ server เมื่อผู้ใช้กดปุ่ม "Update"
2. Server เป็น application พัฒนาโดยใช้ Pyro ทำหน้าที่คอยรับ connection จาก user (XMLForm) พิสูจน์สิทธิ์ของผู้ใช้ และเรียก module data exchanger ขึ้นมาทำงานในการแลกเปลี่ยนข้อมูลระหว่าง XML file และ relational database
3. Management Interface เป็น application เพื่อให้ผู้ใช้ควบคุมการทำงานของ Server ผ่าน network ได้

4.3 Server

Server เป็นหัวใจในการทำงานของ framework โดยจะมี module หลักคือ

1. Data Exchanger (generated code) เป็น XML-relational mapper ทำหน้าที่ map data ใน relational database เป็น XML แล้วส่งให้ client และ map XML จาก client update ลง relational database
2. Authenticator (generated code) ทำหน้าที่พิสูจน์สิทธิ์ของผู้ใช้โดยจะเปรียบเทียบ username และ password จากที่เก็บไว้ใน relational database โดยผ่านตัว Data Exchanger โดยผู้ใช้จะต้องระบุ column ใน database ที่เก็บ username และ password ผ่าน XML configuration file
3. Application Module Generator เนื่องจาก Module Data Exchanger และ Authenticator เป็น Python code ที่เกิดจากการ generate จาก template ที่สร้างไว้ Application Module Generator จะรับผิดชอบการสร้าง module เหล่านี้ โดย parameter ของ template จะได้มากจาก XML-relational mapping configuration XML file
4. Remote Connector เป็น Pyro remote object ทำหน้าที่คอยรับ connection จาก client และพิสูจน์สิทธิ์โดยใช้ Authenticator ที่ตรงกับ Data Exchanger ที่ client ใช้
5. Remote Management เป็น Pyro remote object ทำหน้าที่คอยรับ connection จาก Management Interface เพื่อควบคุมการทำงานของ Server

4.3.1 Data Exchanger

เป็น module หลักของ Server โดยหลักการแล้วทำหน้าที่เป็น XML-relational mapper โดย retrieve data จาก database มาสร้างเป็น XML file ตาม condition ที่กำหนดไว้ให้ client และรับ XML จาก Client มา update บน relational database

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

Data Exchanger จะ map XML กับ relational table ตาม XML-relational mapping configuration XML file ที่ application developer เป็นผู้กำหนด โดย Application Module Generator จะอ่าน XML file แล้วสร้างเป็น Python code 3 file ตั้งชื่อตามชื่อของ model ซึ่งทั้ง 3 file ประกอบด้วย

- Python code ที่ map ระหว่าง XML กับ object เก็บไว้ใน “xmlobj” directory
- Python code ที่ map ระหว่าง object กับ relational เก็บไว้ใน “sqlobj” directory
- Python code ที่ map ระหว่าง XML กับ relational โดยการเรียกใช้ Python code 2 file ข้างต้น เก็บไว้ใน “xchobj” directory

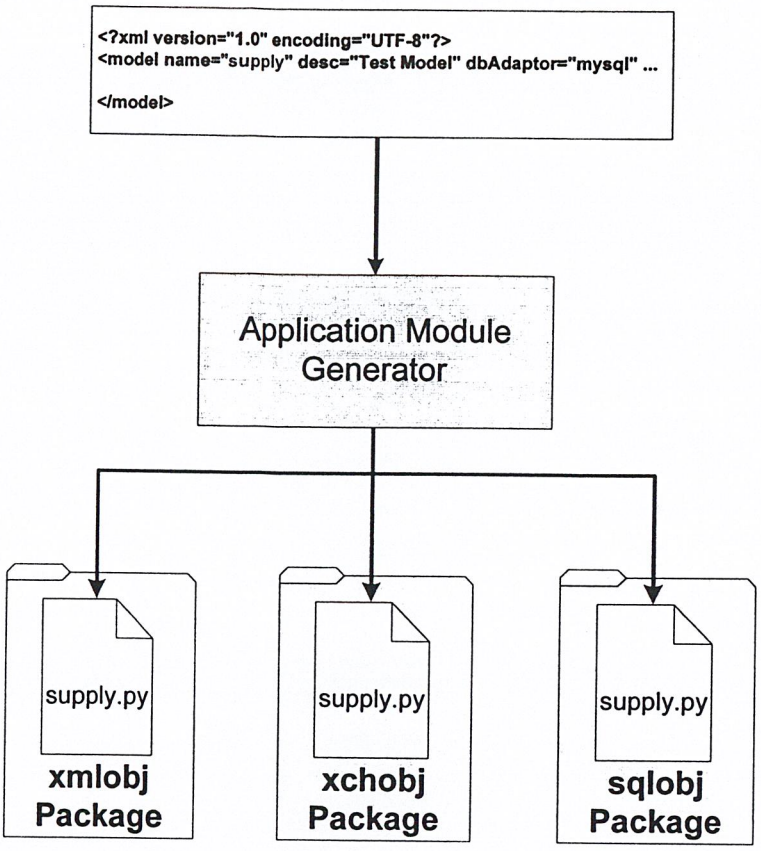
เมื่อ client ติดต่อกับ server Remote Connector จะ import module ซึ่งชื่อเหมือนกับ Data Exchanger ที่ client ร้องขอซึ่งจะอยู่ใน package “xchobj”

การ load Python module ทั้ง 3 ขึ้นมาทำงานจะอาศัยหลักการ load module ของ Python โดยเมื่อ module ถูก import Python interpreter จะค้นหา module ใน PYTHONPATH ซึ่งหลักการคล้าย JAVAPATH ดังนั้น directory ทั้ง 3 คือ xmlobj, sqlobj และ xchobj จะต้องใส่ไว้ใน directory ที่มีอยู่ใน PYTHONPATH ด้วย และจะถูกมองเป็น Python package โดยการ set PYTHONPATH นั้นสามารถทำได้โดยการเพิ่ม environment variable ของ operating system หรือ เพิ่ม PYTHONPATH ในขณะ run-time ก็ได้โดยใช้คำสั่ง

```
import sys
sys.path.append('C:\\ApplicationDir\\')
```

ซึ่งวิธีหลังเป็นวิธีที่ Server ใช้

61376



รูปที่ 4-2 การจัดเก็บ Python Module ที่สร้างจาก Application Module Generator และการตั้งชื่อ

4.3.1.1 คุณสมบัติและข้อจำกัดของ XML-Relational Mapping

เนื่องจาก XML-relational mapping ที่ใช้ใน Server ใช้ในการแลกเปลี่ยนข้อมูลระหว่าง relational database และ ฟpplication XMLForm ที่ทำงานบน Pocket PC โดยใช้ในการบันทึกข้อมูลเท่านั้นดังนั้น XML-relational mapping ที่ใช้ใน Server อาจมีคุณสมบัติต่างจาก XML-DB middleware ทั่วไปคือ

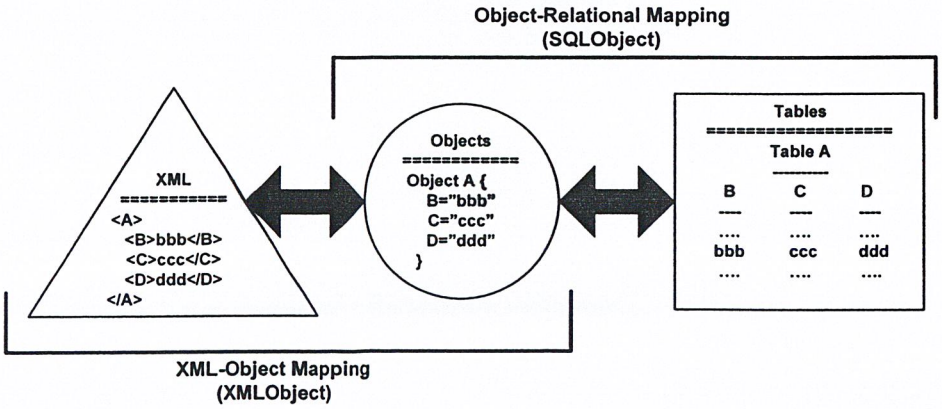
1. เป็น XML-relational mapping แบบ object-based mapping เพราะต้องทำให้สามารถ map เป็น XML ได้หลายรูปแบบและเป็นการ map ในเชิงของ logical Data ไม่ใช้ physical data เหมือนกับ table-based mapping
2. รองรับ SQL data type 4 type คือ Sring(VARCHAR, TEXT), INTEGER, FLOAT(DOUBLE) และ DATETIME
3. Root node ของ output XML จะต้อง map เข้ากับ row ของ table ใด table หนึ่ง
4. Map table เป็น XML element และ column เป็น XML attribute เท่านั้นไม่สามารถ map column เป็น child element ได้
5. Child table จะถูก map เป็น child node
6. ต้องสามารถดึง XML ได้ตามเงื่อนไขที่กำหนดไว้

7. เนื่องจากต้องสามารถ Update ข้อมูลคืนให้ relational database ได้ดังนั้นแต่ละ XML element ที่ map มาจาก relational table จะต้องมี attribute ID ทุก element
8. การแก้ไข XML document ที่ได้จากการ map อาจสามารถเปลี่ยนแปลงข้อมูลใน attribute ได้ทุก attribute ยกเว้น attribute ID เพิ่ม child node ได้แต่จะต้องไม่ทำให้โครงสร้างของ XML เปลี่ยนไปจากที่กำหนดไว้
9. ในการ Update table Data Exchanger จะพิจารณา แต่ละ XML element ว่ามี attribute ID หรือไม่โดยหากมี attribute ID แสดงว่ามี row อยู่ใน table แล้วให้ทำการ update ค่าของ row นั้นตามข้อมูลใน attribute แต่หากไม่มี attribute ID แสดงว่าเป็นข้อมูลใหม่ทั้ง element ให้ทำการสร้าง row ใหม่ใน table

4.3.1.2 การทำงานของ XML-Relational Mapping

Data Exchanger ทำงานโดยอาศัยหลักการ XML-relation mapping แบบ object-based mapping ในการพัฒนาผู้พัฒนาโครงการได้เลือกใช้ library SQLObject และ XMLObject มาช่วยในการพัฒนา การใช้ library 2 ตัวนี้มีข้อดีคือ

1. การพัฒนา XML-relational mapping ด้วยวิธีการ map XML เป็น object ก่อนแล้วจึง map จาก object เป็น relational table เป็นวิธีที่สะดวกกว่า map จาก XML เป็น relational table เลย
2. การใช้ library ที่พัฒนาไว้แล้วและใช้งานอย่างแพร่หลายจะทำให้ application ที่ได้ stable มากขึ้น
3. ทำให้ไม่ต้องเขียนคำสั่งในการสร้าง SQL query เอง
4. สามารถ process XML ได้ง่ายกว่าการใช้ DOM หรือ SAX
5. ลดเวลาในการพัฒนาเทียบกับการพัฒนาเองทั้งหมด



รูปที่ 4-3การนำ XMLObject และ SQLObject นำมาใช้ในการพัฒนา XML-relational mapping

SQLObject และ XMLObject จะถูกเรียกใช้โดย Exchanger class object ซึ่งทั้ง class ทั้ง 3 จะถูกสร้างโดย Application Module Generator

4.3.1.3 Condition ในการ Retrieve ข้อมูลจาก Database

Data Exchanger สามารถกำหนด condition ในการ retrieve ข้อมูลจาก database ได้โดยสามารถกำหนดเป็น condition XML file คือ

```
<condition>
    <Entity name="Staff" conditionString="Staff.id==1 and Staff.firstName=='Wara' and
Staff.lastName=='Songkran'"/>
    <Entity name="Task" conditionString="DATE(Task.assignedTime)==TODAY()"/>
</condition>
```

ในการกำหนด condition ผู้ใช้สามารถใช้ function ของ Python กับแต่ละ property ของ entity ได้ นอกจากนี้ยังมี function ของ variable ประเภท DateTime ที่ให้ใช้ได้คือ

1. DATE(datetime), DATE(string) : return ค่าของวันจาก String (ตัวอย่าง DATE("2005-03-06")) หรือค่าของวันจาก DateTime variable (ตัวอย่าง DATE(Task.assignedTime))
2. TIME(datetime), TIME(string) : return ค่าของเวลาจาก String (ตัวอย่าง TIME("12:00:00")) หรือค่าของวันจาก DateTime variable (ตัวอย่าง TIME(Task.assignedTime))
3. DATETIME(string) : return ค่าของ DateTime จาก String (ตัวอย่าง DATETIME("2005-03-06") หรือ DATETIME("2005-03-06 12:00:00"))

4.4 Application Module Generator

เป็น Module ที่ใช้ Cheetah Template Engine ในการสร้าง Python module โดย parameter ในการสร้างจะได้มาจาก XML configuration file

4.4.1 XML-Relational Mapping Configuration File

โครงสร้างของ XML-relational mapping configurations file ประกอบไปด้วยส่วนต่างๆ ดังนี้

```
<?xml version="1.0" encoding="UTF-8"?>
<model name="supply" desc="Test Model" dbAdaptor="mysql" host="localhost" database="supply"
user="root" password="CL2a7537">
```

model element เป็นส่วนที่ประกาศข้อมูลของ database ที่จะติดต่อด้วย ประกอบด้วย attributes

- name : เป็นชื่อของ model ซึ่งจะนำไปสร้างเป็นชื่อของ Python module ที่ถูก generate ขึ้นมาด้วย
- desc : คำอธิบายของ model
- dbaAdaptor : ชื่อของ DBMS ที่จะติดต่อด้วย ซึ่งอาจจะเป็น Oracle, MySQL, Sybase, MS-SQL, PosgreSQL หรือ Firebird ซึ่งเป็น DBMS ที่ SQLObject รองรับ
- host : ชื่อ hostname หรือ IP address ของ computer ที่ DBMS ทำงานอยู่
- database : ชื่อ database บน DBMS
- user : ชื่อ username ที่ใช้ในการ log-in เข้าใช้ DBMS
- password : คือ password ที่ใช้ในการ log-in เข้าใช้ DBMS

```
<entity name="Supplier" tableName="supplier">
        <id columnName="supplier_id"/>
        <property name="name" columnName="name" type="String"/>
        <property name="address" columnName="address" type="String"/>
        <one-to-many name="tasks" refEntity="Task" joinColumn="supplier_id"/>
        <many-to-many name="products" refEntity="Product" otherColumn="product_id"
intermediateTable="supply_product"/>
</entity>
```

entity element เป็น child element ของ model element อธิบายแต่ละ entity ใน model โดยมี attributes

- name : ชื่อของ entity ซึ่งจะต้องนำไปใช้ในการ map เป็น XML ภายหลัง
- tableName : เป็น physical name ของ relational Table ที่ entity นี้อ้างอิงถึง

นอกจากนี้ยังประกอบด้วย child element ดังนี้

id บอก primary key ของ Table โดยมี attributes

- columnName : ชื่อ primary key column

property อธิบาย property หรือ attribute ซึ่ง map เข้ากับ physical column ใน table ประกอบด้วย attributes

- name : ชื่อของ property
- columnName : ชื่อของ physical column
- type : data type ของ property

one-to-many อธิบายความสัมพันธ์แบบ one-to-many ระหว่าง entity มี attributes

- name : ชื่อของ relationship ซึ่งใช้ในการ reference ไปยัง entity อื่น
- refEntity : ชื่อของ entity ที่เป็น child entity
- join-column : ชื่อ physical column ซึ่งเป็น foreign key บน child table

many-to-many อธิบายความสัมพันธ์แบบ many-to-may ระหว่าง entity มี attributes

- name : ชื่อของ relationship ซึ่งใช้ในการ reference ไปยัง entity อื่น
- refEntity : ชื่อของ entity ที่มีความสัมพันธ์แบบ many-to-many
- intermediateTable : ชื่อของ physical table ที่เป็น intermediate table ระหว่าง table ทั้ง 2 ที่มีความสัมพันธ์แบบ many-to-many กัน
- otherColumn : ชื่อ physical column ซึ่งเป็น foreign key ที่อยู่บน intermediate table

```
<xmlroot name="staff" entity="Staff" clientusername="Staff.username",
clientpassword="Staff.password">
        <attribute property="id" name="id"/>
        <attribute property="firstName" name="firstname"/>
        <attribute property="lastName" name="lastname"/>
        <element name="task" entity="Task">
                <attribute property="id" name="id"/>
                <attribute property="assignedTime" name="assignedtime"/>
                <attribute property="finishedTime" name="finishedtime"/>
                <attribute property="supplier.address" name="address"/>
                <element name="item" entity="Item">
                        <attribute property="id" name="id"/>
                        <attribute property="quantity" name="quantity"/>
                        <attribute property="ppu" name="priceperunit"/>
                        <attribute property="product.name"
                        name="productname"/>
                </element>
        </element>
</xmlroot>
```

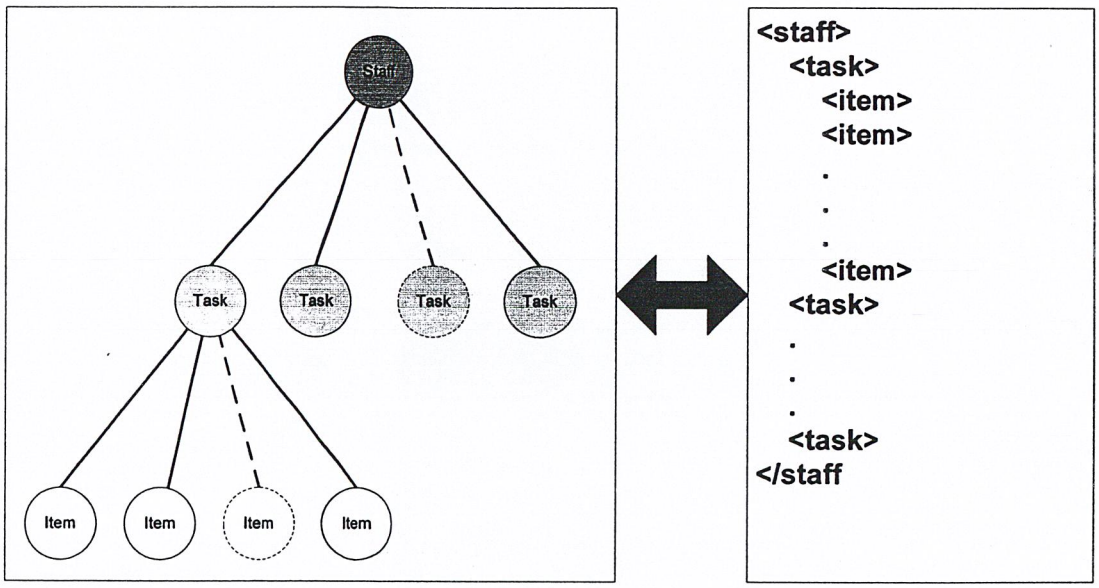
xmlroot และ **element** เป็น child element ของ model element ที่อธิบายการ map ระหว่าง XML document และ ER model ที่กำหนดไว้แล้วข้างต้น ประกอบด้วย attributes

- name : ชื่อของ root node ของ XML ที่จะ map กับ ER entity
- entity : ชื่อของ entity จาก ER model ที่จะ map เข้ากับ XML element

นอกจากนี้ยังมี child node คือ

attribute element อธิบายการ map ระหว่าง attribute ของ element กับ property ของ entity มี attributes

- property : ชื่อของ property ใน entity
- name : ชื่อของ XML attribute



รูปที่ 4-4 การ Map ระหว่าง XML และ Object จะมอง XML เป็น Graph ของ Object

Element xmlroot จะมี attribute อีก 2 ตัวคือ

- clientusername : บอกถึง property และ entity ที่เก็บ username ซึ่งใช้สำหรับการ authenticate เข้าสู่ Server
- clientpassword : บอกถึง property และ entity ที่เก็บ password ซึ่งใช้สำหรับการ authenticate เข้าสู่ Server

4.4.2 SQLObject Class Generation

เป็น template ซึ่งจะรับ element model จาก configuration file มาสร้างเป็น Python code โดยจะได้ SQLObject class 1 class ต่อ 1 ER entity

```
#def sqlObjectClass($entity)
class ${entity.name}(SQLObject):
    #if $entity.tableName
    _table = "${entity.tableName}"
    #end if
    #if $entity.id and $entity.id.columnName
    _idName = "${entity.id.columnName}"
    #end if
    #for $prop in $entity.properties
    #if $prop.type == "String"
    #set $constructString = "StringCol("
    #elif $prop.type == "Integer"
    #set $constructString = "IntCol("
    #elif $prop.type == "Float"
    #set $constructString = "FloatCol("
    #elif $prop.type == "DateTime"
    #set $constructString = "DateTimeCol("
    #end if
    #if $prop.columnName
    #set $constructString = $constructString + "dbName='" + $prop.columnName + "', "
    #end if
    #if $prop.default
    #set $constructString = $constructString + "default='" + $prop.default + "', "
    #elif not $prop.required == "True"
    #set $constructString = $constructString + "default=None, "
    #end if
    #if $prop.alternateID
    #set $constructString = $constructString + "alternateID=True, "
    #end if
    ${prop.name} = ${constructString})
    #end for
    #for $foreignKey in $entity.foreignKeys
    #if $foreignKey.columnName
    #set $constructString = "ForeignKey('" + $foreignKey.refEntity + "', " + "dbName='" + $foreignKey.columnName + "'"
    #else
    #set $constructString = "ForeignKey('" + $foreignKey.refEntity + "'"
    #end if
    ${foreignKey.name} = ${constructString} ,default=None)
    #end for
    #for $one in $entity.oneToManys
    #set $constructString = "MultipleJoin('" + $one.refEntity + "', "
    #if $one.joinColumn
    #set $constructString = $constructString + "joinColumn='" + $one.joinColumn + "', "
    #end if
    ${one.name} = ${constructString})
    #end for
    #for $many in $entity.manyToManys
    #set $constructString = "RelatedJoin('" + $many.refEntity + "', "
    #if $many.intermediateTable
    #set $constructString = $constructString + "intermediateTable='" + $many.intermediateTable + "', "
    #end if
    #if $many.otherColumn
    #set $constructString = $constructString + "otherColumn='" + $many.otherColumn + "', "
    #end if
    ${many.name} = $constructString)
    #end for
#end def
```

Table Name and ID

Property

Foreign Key

One-To-Many

Many-To-Many

ตรวจสอบ Type ของ Property

สร้าง Property และกำหนด Column Name

สร้าง Foreign Key กำหนด Column Name และ Referenced Entity

สร้าง One-To-Many กำหนด Join Column และ Referenced Entity

สร้าง Many-To-Many กำหนด Other Column, Referenced Entity และ Intermediate Table

รูปที่ 4-5 Template ที่ใช้ในการสร้าง SQLObject Class

ตัวอย่าง Python code ที่สร้างออกมาคือ

```
class Task(SQLObject):
        _table = "task"
        _idName = "task_id"
        assignedTime = DateTimeCol(dbName='assigned_time', )
        finishedTime = DateTimeCol(dbName='finished_time', default=None, )
        description = StringCol(dbName='description', default=None, )
        staff = ForeignKey('Staff', dbName='staff_id' ,default=None)
        supplier = ForeignKey('Supplier', dbName='supplier_id' ,default=None)
        items = MultipleJoin('Item', joinColumn='task_id', )
```

4.4.3 XMLObject Class Generation

เป็น Template ที่ใช้ในการสร้าง class ของ XMLObject โดยจะสร้างจาก element xmlrool ใน XML mapping configuration file

XMLObject Class

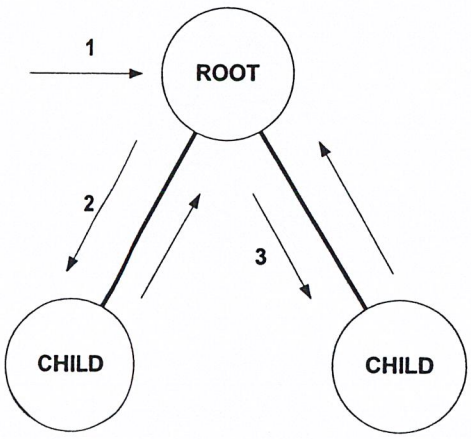
```
#def xmlObj($element)
class ${element.name}(XMLObject):
        _name = '${element.name}'          ประกาศชื่อของ Element
        #for $attr in $element.attributes
        ${attr.name} = StringAttribute(optional=True)      ประกาศ Attribute
        #end for
        #for $element in $element.elements
        ${element.name} = ListNode('${element.name}', optional=True)
        #end for                                          ประกาศ Child Element
#end def
```

Recursive Function

```
#def createXMLObject($node)
$xmlObj($node)
#if $node.elements
#for $element in $node.elements
$createXMLObject($element)
#end for
#end if
#end def
```

รูปที่ 4-6 Template ที่ในการสร้าง XMLObject class

Recursive function จะใช้ในการ parse tree ของ XML ที่กำหนดไว้ใน element xmlroot โดยใช้หลักการของ infix tree parsing



รูปที่ 4-7 Infix Tree Parsing

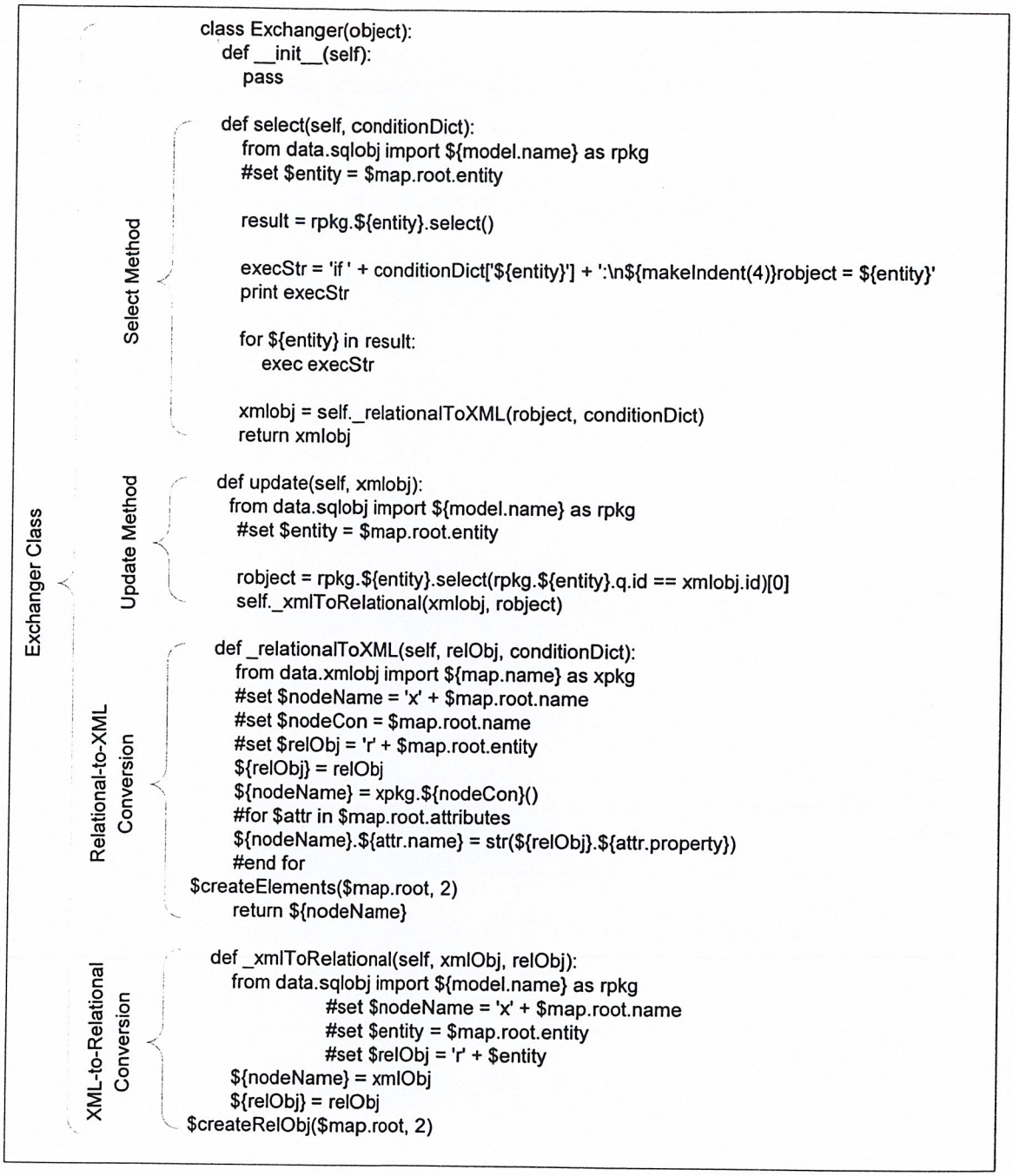
ตัวอย่าง Python code ที่สร้างออกมาคือ

```
class staff(XMLObject):
        _name = 'staff'
        id = StringAttribute(optional=True)
        firstname = StringAttribute(optional=True)
        lastname = StringAttribute(optional=True)
        task = ListNode('task', optional=True)

class task(XMLObject):
        _name = 'task'
        id = StringAttribute(optional=True)
        assignedtime = StringAttribute(optional=True)
        finishedtime = StringAttribute(optional=True)
        address = StringAttribute(optional=True)
        item = ListNode('item', optional=True)

class item(XMLObject):
        _name = 'item'
        id = StringAttribute(optional=True)
        quantity = StringAttribute(optional=True)
        priceperunit = StringAttribute(optional=True)
        productname = StringAttribute(optional=True)
```

4.4.4 Exchanger Class Generation

เป็น Class ที่เรียกใช้ XMLObject class และ SQLObject class และ XMLObject class โดยจะมี method

- select(self, conditionDict) : ดึง XML file ตาม condition ที่กำหนด
- update(self, xmlobj) : update table ตามข้อมูลใน xmlobj
- _relationalToXML(self, relObj, conditionDict) : utility method ใช้ในการถ่ายโอนข้อมูลจาก object ของ SQLObject class ไปยัง object ของ XMLObject class
- _xmlToRelational(self, xmlObj, relObj) : utility method ใช้ในการถ่ายโอนข้อมูลจาก object ของ XMLObject class ไปยัง object ของ SQLObject class



รูปที่ 4-8 Template ที่ใช้ในการสร้าง Data Exchanger Class

4.4.4.1 Relational To XML Function Generator

เป็น Function ที่ใช้โอนข้อมูลจาก object ของ SQLObject class ไปยัง Object ของ XMLObject class

Recursive Function

```
#def createElements($parent, $i)
#for $element in $parent.elements
$initNodes($parent, $element, $i)
#if $element.elements
$createElements($element, $i+1)
#end if
#end for
#end def
```

XML Node Creation

```
#def initNodes($parentNode, $childNode, $i)
#set $parentNodeName = 'x' + $parentNode.name
#set $childNodeName = 'x' + $childNode.name
#set $childNodeCon = $childNode.name
#set $relObj = 'r' + $parentNode.entity
#set $initRObj = $findReferrenceName($parentNode.entity, $childNode.entity)
#if $initRObj[-1:] == 's'
#set $rItemName = $initRObj[0].upper() + initRObj[1:-1]
#else
#set $rItemName = $initRObj[0].upper() + initRObj[1:]
#end if
```

Iterate Child Object Property and Assigned to XML

```
${makeIndent(i)}for ${rItemName} in ${relObj}.${initRObj}:
${makeIndent(i+1)}match = False
${makeIndent(i+1)}try:
${makeIndent(i+2)}execStr = 'if ' + conditionDict['${childNode.entity}'] + ': match = True'
${makeIndent(i+2)}exec execStr
${makeIndent(i+1)}except KeyError:
${makeIndent(i+2)}match = True
${makeIndent(i+1)}except AttributeError:
${makeIndent(i+2)}match = False
${makeIndent(i+1)}if match:
${makeIndent(i+2)}${childNodeName} = xpkg.${childNodeCon}()
#for $attr in $childNode.attributes
${makeIndent(i+2)}try:
${makeIndent(i+3)}if ${rItemName}.${attr.property}:
${makeIndent(i+4)}${childNodeName}.${attr.name} = str(${rItemName}.${attr.property})
${makeIndent(i+2)}except:
${makeIndent(i+3)}print 'attribute access error'
#end for
${makeIndent(i+2)}${parentNodeName}.${childNode.name}.append(${childNodeName})
#if $childNode.elements
${makeIndent(i+2)}r${childNode.entity} = ${rItemName}
${makeIndent(i+1)}if not match:
${makeIndent(i+2)}continue
#end if
#end def
```

รูปที่ 4-9 Template ที่ใช้ในการสร้าง Data Exchanger Class

ตัวอย่างส่วนหนึ่งของ routine ที่ใช้ในการสร้าง XML จาก relational table คือ

```
for Task in rStaff.tasks:
        match = False
        try:
            execStr = 'if ' + conditionDict['Task'] + ': match = True'
            print execStr
            exec execStr
            print Task
            print match
        except KeyError:
            match = True
        except AttributeError:
            match = False
        if match:
            print 'match'
            print Task
            xtask = xpkg.task()
            try:
                if Task.id:
                    xtask.id = str(Task.id)
                    print str(Task.id)
            except:
                print 'attribute access error'
            try:
                if Task.assignedTime:
                    xtask.assignedtime = str(Task.assignedTime)
                    print str(Task.assignedTime)
            except:
                print 'attribute access error'
            try:
                if Task.finishedTime:
                    xtask.finishedtime = str(Task.finishedTime)
                    print str(Task.finishedTime)
            except:
                print 'attribute access error'
            try:
                if Task.supplier.address:
                    xtask.address = str(Task.supplier.address)
                    print str(Task.supplier.address)
            except:
                print 'attribute access error'
            xstaff.task.append(xtask)
            rTask = Task
        if not match:
            continue
```

4.4.4.2 XML to Relational Function Generator

เป็น function ที่ใช้ในการถ่ายโอนข้อมูลจาก object ของ XMLObject class ไปยัง object ของ SQLObject class

```
#def createRelObj($parent, $i)
#for $element in $parent.elements
$initObjects($parent, $element, $i)
#if $element.elements
$createRelObj($element, $i+1)
#end if
#end for
#end def
```

รูปที่ 4-10 Fucntion ที่ใช้ในการสร้าง SQLObject จาก XMLObject

```
#def initObjects($parentNode, $childNode, $i)
#set $parentElementName = 'x' + $parentNode.name
#set $childElementName = 'x' + $childNode.name
#set $parentElement = $parentNode.name
#set $childElement = $childNode.name
#set $childEntity = $childNode.entity
#set $parentEntity = $parentNode.entity
#set $childEntityName = 'r' + $childNode.entity
#set $parentEntityName = 'r' + $parentNode.entity
#set $parentIDName = $findParentForeignKey($parentEntity, $childEntity) + 'ID'
```

รูปที่ 4-11 ส่วนที่เป็นการ Set ค่าของ Function InitObjects

```
${makeIndent(i)}for ${childElementName} in ${parentElementName}.${childElement}:
${makeIndent(i+1)}if ${childElementName}.id:
${makeIndent(i+2)}${childEntityName} = rpkg.${childEntity}.select(rpkg.${childEntity}.q.id ==
int(${childElementName}.id))[0]
#for $attribute in $childNode.attributes
#if not $attribute.name == 'id'
${makeIndent(i+2)}if ${childElementName}.${attribute.name}:
#if $propertyType($childEntity, $attribute.property) == "String"
${makeIndent(i+3)}${childEntityName}.${attribute.property} = ${childElementName}.${attribute.name}
#elif $propertyType($childEntity, $attribute.property) == "Integer"
${makeIndent(i+3)}${childEntityName}.${attribute.property} = int(${childElementName}.${attribute.name})
#elif $propertyType($childEntity, $attribute.property) == "Float"
${makeIndent(i+3)}${childEntityName}.${attribute.property} = float(${childElementName}.${attribute.name})
#elif $propertyType($childEntity, $attribute.property) == "DateTime"
${makeIndent(i+3)}${childEntityName}.${attribute.property} =
DateTimeFrom(${childElementName}.${attribute.name})
#end if
#end if
#end for
```

รูปที่ 4-12 ส่วนที่ใช้ในการสร้าง SQLObject ในกรณีที่มี Row ที่สัมพันธ์กับ XML Element อยู่ใน Table แล้ว

```
${makeIndent(i+1)}else:
${makeIndent(i+2)}${childEntityName} = rpkg.${childEntity}()
${makeIndent(i+2)}
#for $attribute in $childNode.attributes
#if not $attribute.name == 'id'
${makeIndent(i+2)}if ${childElementName}.${attribute.name}:
#if $propertyType($childEntity, $attribute.property) == "String"
${makeIndent(i+3)}${childEntityName}.${attribute.property} = ${childElementName}.${attribute.name}
#elif $propertyType($childEntity, $attribute.property) == "Integer"
${makeIndent(i+3)}${childEntityName}.${attribute.property} = int(${childElementName}.${attribute.name})
#elif $propertyType($childEntity, $attribute.property) == "Float"
${makeIndent(i+3)}${childEntityName}.${attribute.property} = float(${childElementName}.${attribute.name})
#elif $propertyType($childEntity, $attribute.property) == "DateTime"
${makeIndent(i+3)}${childEntityName}.${attribute.property} =
DateTimeFrom(${childElementName}.${attribute.name})
#end if
#end if
#end for
${makeIndent(i+2)}${childEntityName}.${parentIDName} = ${parentEntityName}.id
#end def
```

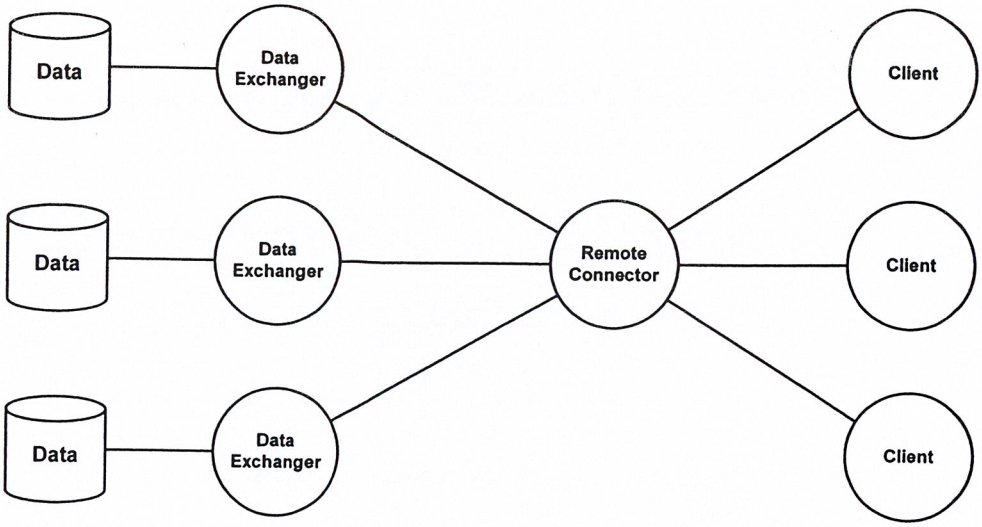
รูปที่ 4-13 ส่วนที่ใช้ในการสร้าง SQLObject ในกรณีที่ไม่มี Row ที่สัมพันธ์กับ XML Element อยู่ใน Table แล้ว

ตัวอย่างส่วนหนึ่งของ routine ที่ใช้ในการ update ข้อมูลจาก XML file ลง relational database คือ

```
for xtask in xstaff.task:
        if xtask.id:
            rTask = rpkg.Task.select(rpkg.Task.q.id == int(xtask.id))[0]
            if xtask.assignedtime:
                rTask.assignedTime = DateTimeFrom(xtask.assignedtime)
            if xtask.finishedtime:
                rTask.finishedTime = DateTimeFrom(xtask.finishedtime)
            if xtask.address:
                rTask.supplier.address = xtask.address
        else:
            rTask = rpkg.Task()

            if xtask.assignedtime:
                rTask.assignedTime = DateTimeFrom(xtask.assignedtime)
            if xtask.finishedtime:
                rTask.finishedTime = DateTimeFrom(xtask.finishedtime)
            if xtask.address:
                rTask.supplier.address = xtask.address
            rTask.staffID = rStaff.id
```

4.5 Remote Connector

เป็น remote object โดยเมื่อทำงานจะคอยอยู่ใน loop เมื่อมี client connect เข้ามา Pyro จะสร้าง instance ของ remote connector ขึ้นมาโดยจะสร้างขึ้นมาทำงานอยู่บน thread ของตัวเอง ทำให้ Server พร้อมรับ connection อีกครั้ง Remote Connector จะสร้าง Data Exchanger instance เพื่อใช้แลกเปลี่ยนข้อมูลกันระหว่าง client กับ database ดังนั้นในขณะหนึ่งๆ จะมีจำนวน client ที่ติดต่อเข้ามาเท่ากับจำนวน Data Exchanger object



รูปที่ 4-14 การเชื่อมโยงระหว่าง Client, Remote Connector และ Data Exchanger

โดย Remote Connector จะมี 2 method คือ

- select(modelName)

  ใช้ในการดึง XML จาก DB ผ่าน Data Exchanger ที่ชื่อตรงกับ model name

- update(modelName, xmlContent)

  ใช้ในการส่ง XML กลับมา update ผ่าน Data Exchanger

4.6 Authenticator

ใช้ feature authentication ของ Pyro โดยจะใช้ code generation สร้าง Class authenticator ที่ inherit มาจาก class Pyro.protocol DefaultConnValidator โดยจะต้อง override method ดังนี้

- acceptHost(self,daemon,connection)

  ถูกเรียกเพื่อตรวจสอบ host ที่จะอนุญาติให้ติดต่อกับ server ได้โดยอาจจะตรวจสอบ IP address หรือ Hostname จาก parameter connection

- acceptIdentification(self, daemon, connection, token, challenge)

  ถูกเรียกจาก server เพื่อตรวจสอบ pass phrase จาก client ว่าถูกต้องหรือไม่

- createAuthToken(self, authid, challenge, peeraddr, URI, daemon)

  ถูกเรียกใช้ทั้งจาก server และ client เพื่อใช้สร้าง token ในการ validate connection

- createAuthChallenge(self, tcpserver, conn)

  ถูกเรียกใช้จาก server เพียงอย่างเดียวเมื่อมี connection ใหม่เข้ามา method นี้จะ return challenge code ที่จะต้องส่งให้ client

- mungeIdent(self, ident)

เป็น utility method ที่ใช้ในการเปลี่ยน identification string (plain text) ให้เป็น cipher text โดยอาจจะใช้ hash algorithm หรือ encryption method ก็ได้

- setAllowedIdentifications(self, ids)

  บอก Pyro deamon ว่า identification string ใดบ้างที่ valid

4.7 Remote Management

เป็น remote object ที่ใช้ในการควมคุมการทำงานของ Server มี method ดังนี้

- addDataExchanger(xmlModelContent, conditionDict)

  เพิ่ม Data Exchanger โดยมี parameters คือ

  - xmlModelContent : string ของ XML configuration file
  - conditionDict : condition ในการ retrieve ข้อมูลจาก database

- removeDataExchanger(exchangerName)

  ลบ Data Exchanger ออกจาก Server มี parameter คือ

  - exchangerName : ชื่อของ data exchanger ที่จะลบออก

- changeRetrieveCondition(exchangerName, conditionDict)

  เปลี่ยน condition ในการ retrieve ข้อมูลจาก database มี parameter คือ

  - exchangerName : ชื่อของ data exchanger ที่จะลบออก
  - conditionDict : condition ใหม่ ในการ retrieve ข้อมูลจาก database

4.8 XMLForm

เป็น application ที่ทำงานบน Pocket PC ทำงานโดยเมื่อเปิดขึ้นจะทำการติดต่อกับ Remote Connector เพื่อดึง XML file และเมื่อผู้ใช้กดปุ่ม update ก็จะทำการส่ง XML กลับไป update ผ่าน Server โดย user interface จะถูกสร้างขึ้นโดย XMLForm configuration file และจะมี lookup file ซึ่งเป็น XML file ที่ทำหน้าที่เป็น lookup table

4.9 Management Interface

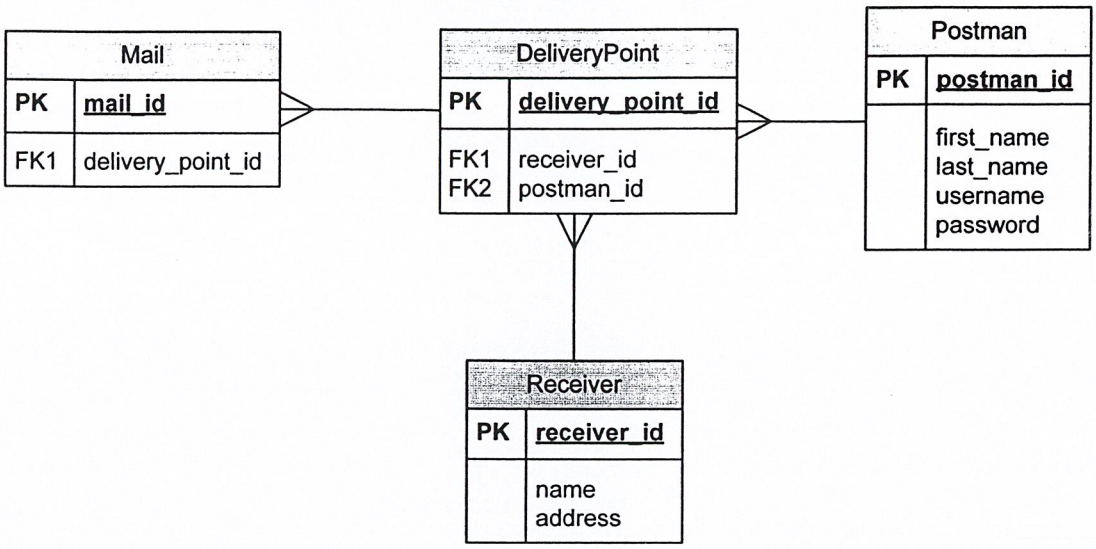
Management Interface เป็น GUI application ที่ติดต่อกับ Server ผ่าน Remote Management module มีหน้าที่เพิ่มและควบคุม Data Exchanger

Management Interface function

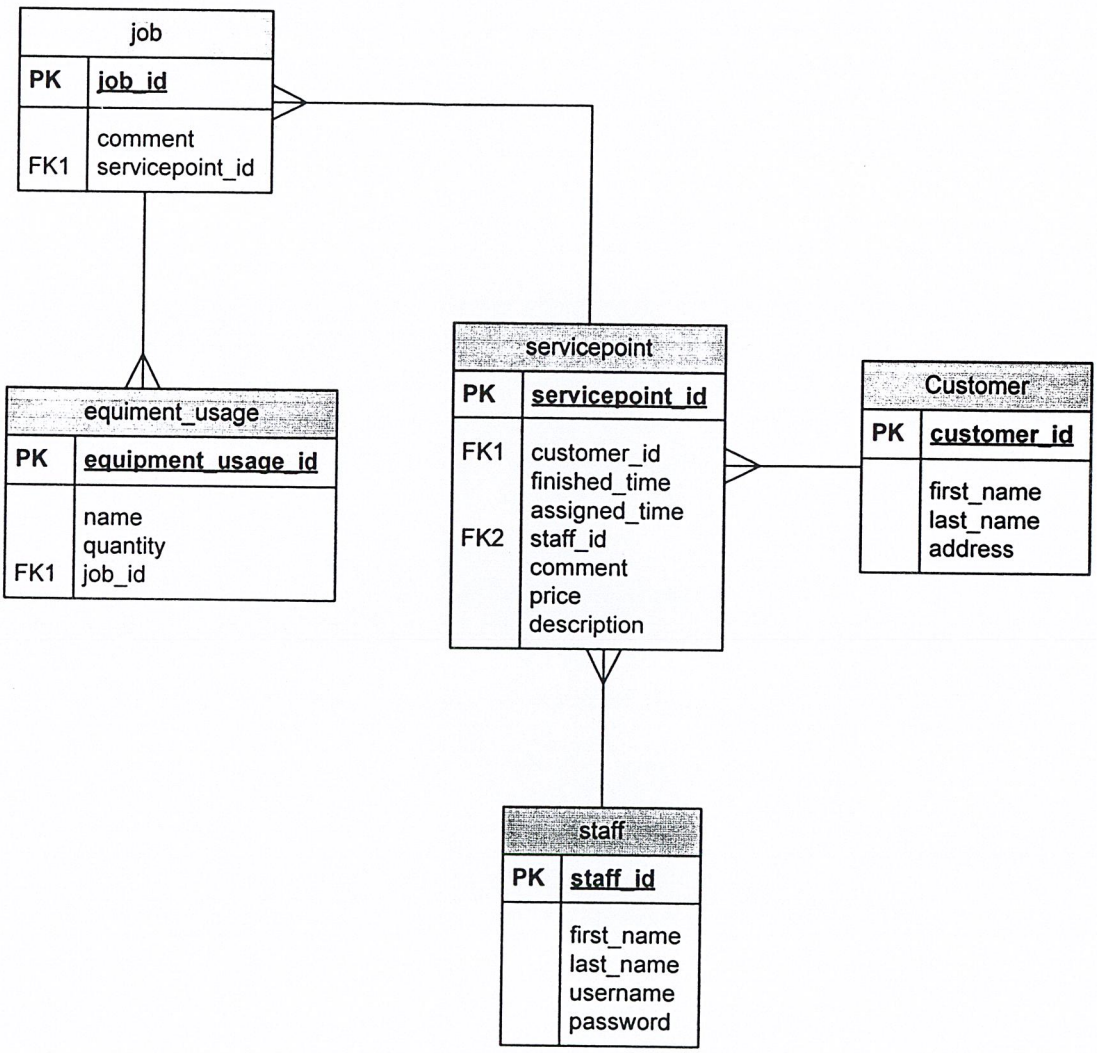
1. เพิ่ม Data Exchanger
2. เปลี่ยน condition ในการ retrieve data ของ Data Exchanger
3. ดูรายชื่อของ Data Exchanger ทั้งหมดและ condition ในการ retrieve data ของแต่ละตัว

4.10 รูปแบบของ Database schema ที่ framework รองรับ

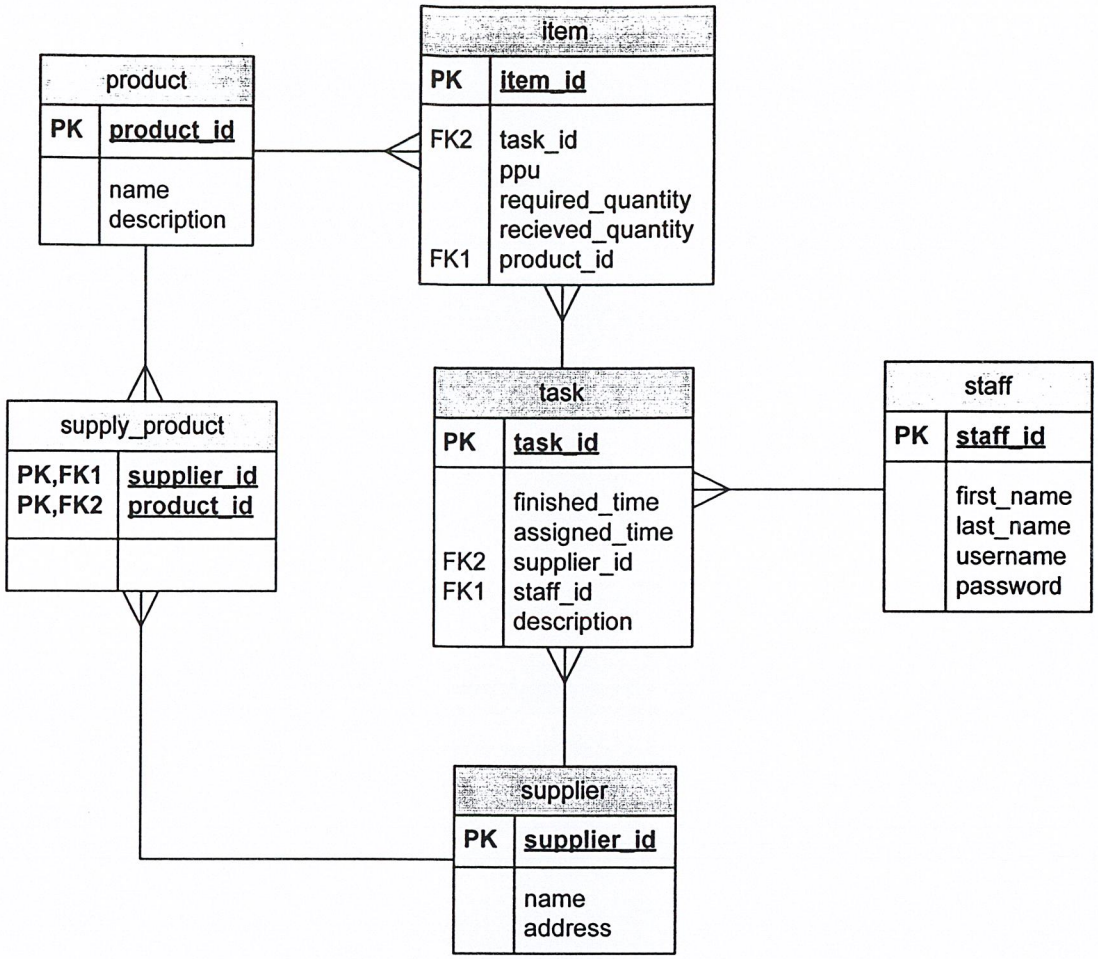
Database schema ของ application ที่ใช้ในการเก็บข้อมูลจะมีรูปแบบใดก็ได้แต่จะต้องมี entity ที่แทนตัว user ดังตัวอย่างเช่น



รูปที่ 4-15 ER Diagram ของ post office database



รูปที่ 4-16 ER Diagram ของ service database



รูปที่ 4-17 ER Diagram ของ supply database

จะเห็นได้ว่าทุก schema จะมี entity ที่แทนตัว user เช่นใน schema เช่น ใน post office database จะมี postman ซึ่งเป็น entity ที่แทนตัวบุรษไปรษณีย์ และใน supply และ service database ก็จะมี staff ซึ่งเป็น entity ที่แทนตัวพนักงาน

บทที่ 5

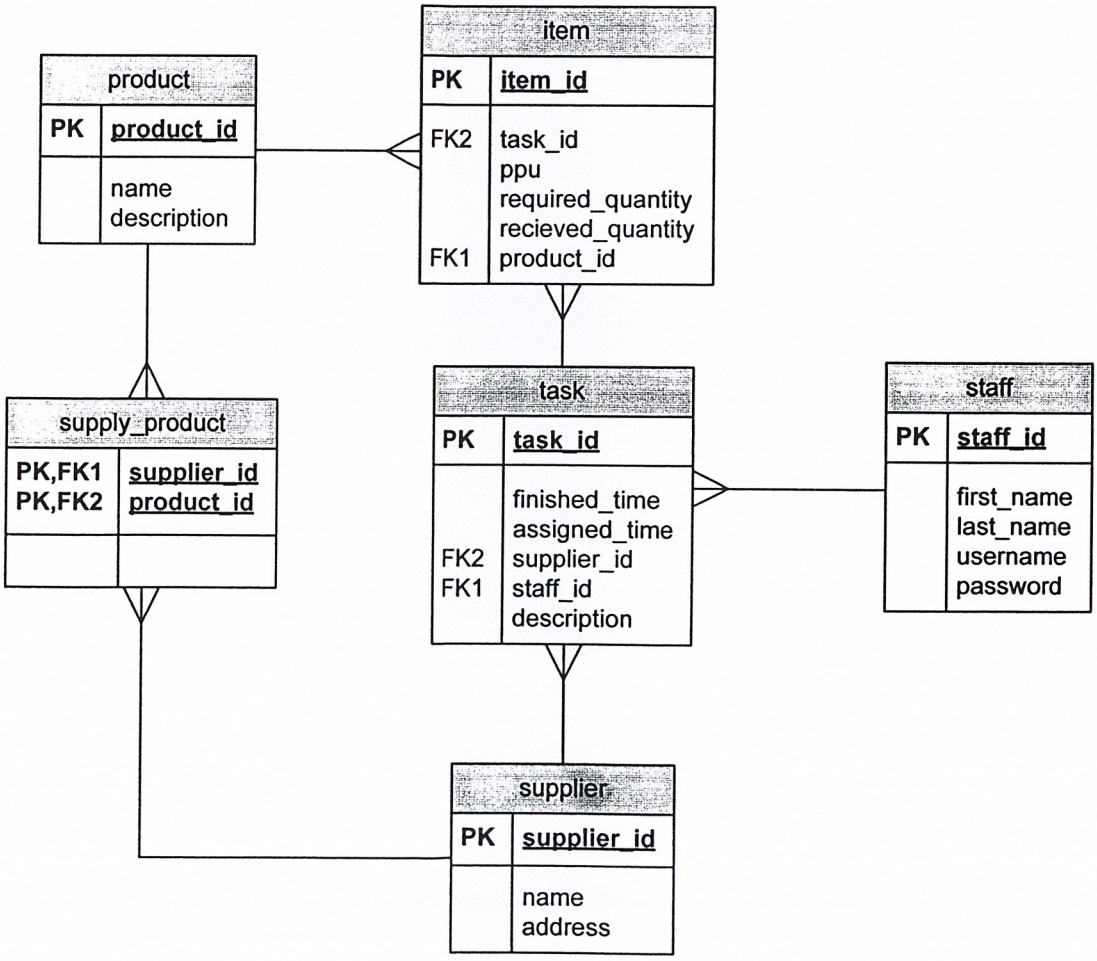
ผลการทดลอง

5.1 การทดสอบใช้งาน XML-Relational Mapping

เพื่อทำสอบการทำงานของ Data Exchanger ผู้ทำโครงงานได้ทดลองสร้าง database ซึ่งจำลองการทำงานจริง และทดลองใช้ Data Exchanger ในการ select และ update database ผ่าน XML

5.2 Sample ER Diagram

ในการทดสอบใช้งานผู้พัฒนาได้ทดลองสร้าง database ทดสอบ 2 ตัวคือ "supply" และ "ubc" โดยมี schema ดังนี้



รูปที่ 5-1 ER Diagram ของ Supply Database

Database supply เก็บข้อมูลการรับของของ mega store แห่งหนึ่ง โดยในแต่ละวัน supervisor จะมอบหมายงานให้รถรับส่งของแต่ละคันไปรับของจาก supplier หลายรายและนำของที่รับมากลับมายังโกดังเก็บของรวมเพื่อกระจายสินค้าออกไปยังสาขาต่างๆ ต่อไป โดยพนักงานขับรถจะต้องระบุจำนวนของที่รับมาจริงจาก supplier แต่ละราย และลงเวลาในการปฏิบัติงานเสร็จในแต่ละ supplier

Table staff : เป็น Table ที่เก็บรายละเอียดของพนักงานรับของ

Column	คำอธิบาย
staff_id	primary key ของ staff
first_name	ชื่อของพนักงาน
last_name	นามสกุลของพนักงาน
username	username ที่ใช้ในการ log-in
password	password ที่ใช้ในการ log-in

Table task : เป็น Table ที่เก็บรายละเอียดของงานที่พนักงานต้องทำในแต่ละวัน

Column	คำอธิบาย
task_id	primary key ของ task
assigned_time	เวลาที่มอบหมายงานให้ลงเวลาโดยผู้มอบหมายงาน
finished_time	เวลาที่ทำงานเสร็จจะลงเวลาโดยผู้ปฏิบัติงาน
supplier_id	foreign key ของ supplier ที่จะไปรับสิ้นค้า
staff_id	foreign key ของ staff ที่มอบหมายงานให้
description	คำอธิบายงานใส่โดยผู้มอบหมายงาน

Table item : เป็น Table ที่เก็บรายละเอียดของสินค้าที่ต้องไปรับมาจากแต่ละ Supplier

Column	คำอธิบาย
item_id	primary key ของ item
task_id	foreign key ของ task
ppu	Price per unit ราคาต่อหน่วยของสินค้า
required_quantity	จำนวนของสินค้าที่ต้องการ
receive_quantity	จำนวนของสินค้าที่รับมาจริง
product_id	foreign key ของ product

Table product : เป็น Table ที่เก็บรายละเอียดของสินค้า

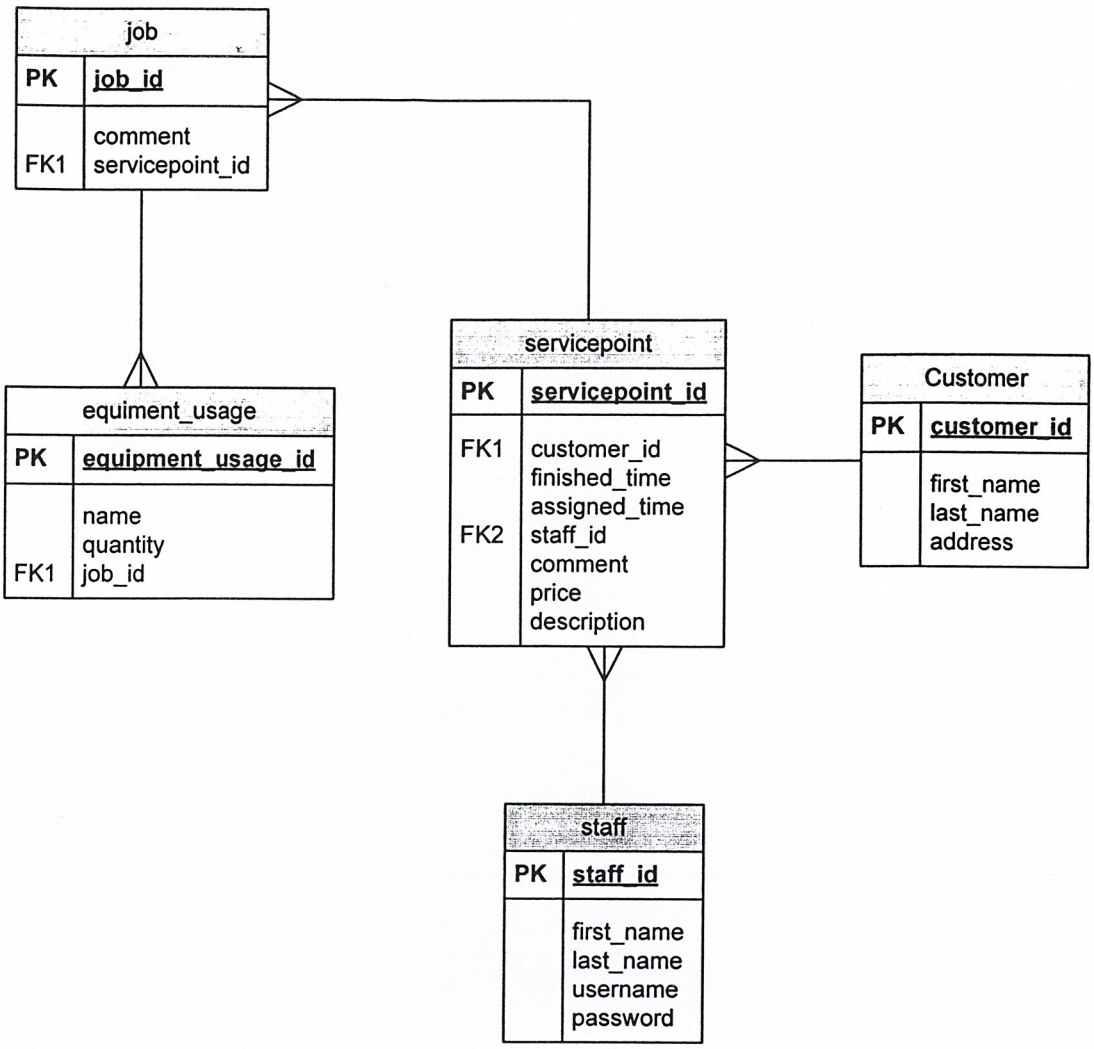
Column	คำอธิบาย
product_id	primary key ของ product
name	ชื่อของผลิตภัณฑ์
description	คำอธิบายของผลิตภัณฑ์

Table supplier : เป็น Table ที่เก็บรายละเอียดของ Supplier

Column	คำอธิบาย
supplier_id	primary key ของ supplier
name	ชื่อของ supplier
address	ที่อยู่ของ supplier

Table supply_product : เป็น Intermediate Table ระหว่าง

Column	คำอธิบาย
supplier_id	foreign key ของ supplier
product_id	foreign key ของ product



รูปที่ 5-2 ER Diagram ของ UBC Service Database

Database ubc เก็บข้อมูลการให้บริการของช่างในระบบ Cable TV รายหนึ่ง โดยในแต่ละวัน supervisor จะจัดสรรงานของลูกค้าแต่ละรายให้รถให้บริการแต่ละคัน โดยในแต่ละจุดที่ไปให้บริการ พนักงานจะต้องระบุงานแต่ละอย่างที่ได้ทำและอุปกรณ์ที่ใช้ในแต่ละงานรวมทั้งต้องประเมินราคาของการให้บริการทั้งหมดและเวลาที่ให้บริการเสร็จสมบูรณ์ในแต่ละจุดที่ไปให้บริการบันทึกลง PDA ด้วย

Table staff : เป็น Table ที่เก็บรายละเอียดของพนักงานให้บริการ

Column	คำอธิบาย
staff_id	primary key ของ staff
first_name	ชื่อของพนักงาน
last_name	นามสกุลของพนักงาน
username	username ที่ใช้ในการ log-in
password	password ที่ใช้ในการ log-in

Table servicepoint : เป็น Table ที่เก็บรายละเอียดของจุดที่พนักงานจะต้องไปให้บริการ

Column	คำอธิบาย
servicepoint_id	primary key ของ service point
assigned_time	เวลาที่มอบหมายงานให้ ลงเวลาโดยผู้มอบหมายงาน
finished_time	เวลาที่ทำงานเสร็จจะลงเวลาโดยผู้ปฏิบัติงาน
customer_id	foreign key ของ customer ที่จะไปให้บริการ
staff_id	foreign key ของ staff ที่มอบหมายงานให้
description	คำอธิบายงานใส่โดยผู้มอบหมายงาน
price	ราคาของการให้บริการทั้งหมดประเมินโดยพนักงานให้บริการ
comment	comment ซึ่งใส่โดยพนักงานให้บริการ

Table job : เป็น Table ที่เก็บรายละเอียดของงานแต่ละอย่างที่พนักงานให้บริการทำในแต่ละจุดที่ไปให้บริการ

Column	คำอธิบาย
job_id	primary key ของ job
description	คำอธิบายงานใส่โดยพนักงานให้บริการ
servicepoint_id	foreign key ของ service point

5.3XML Relational-Mapping Configuration File

เมื่อสร้าง database ขึ้นมาทดลองใช้งานแล้ว จากนั้นเป็นการสร้าง XML-relational mapping configuration file สำหรับทั้ง 2 database

```
<?xml version="1.0" encoding="UTF-8"?>
<model name="supply" desc="Supply Model" dbAdaptor="mysql" host="localhost"
database="supply" user="root" password="CL2a7537">
    <entity name="Staff" tableName="staff">
        <id columnName="staff_id"/>
        <property name="firstName" columnName="first_name" type="String"/>
        <property name="lastName" columnName="last_name" type="String"/>
        <property name="username" columnName="username" type="String"/>
        <property name="password" columnName="password" type="String"/>
        <one-to-many name="tasks" refEntity="Task" joinColumn="staff_id"/>
    </entity>
    <entity name="Task" tableName="task">
        <id columnName="task_id"/>
        <property name="assignedTime" columnName="assigned_time"
        type="DateTime"/>
        <property name="finishedTime" columnName="finished_time"
        type="DateTime"/>
        <property name="description" columnName="description" type="String"/>
        <foreignkey name="staff" refEntity="Staff" columnName="staff_id"/>
        <foreignkey name="supplier" refEntity="Supplier"
        columnName="supplier_id"/>
        <one-to-many name="items" refEntity="Item" joinColumn="task_id"/>
    </entity>
    <entity name="Item" tableName="item">
        <id columnName="item_id"/>
        <property name="requiredQuantity" columnName="required_quantity"
        type="Integer"/>
        <property name="receivedQuantity" columnName="received_quantity"
        type="Integer"/>
        <property name="ppu" columnName="ppu" type="Float"/>
        <foreignkey name="task" refEntity="Task" columnName="task_id"/>
        <foreignkey name="product" refEntity="Product"
columnName="product_id"/>
    </entity>
    <entity name="Product" tableName="product">
        <id columnName="product_id"/>
        <property name="name" columnName="name" type="String"/>
        <property name="desc" columnName="description" type="String"/>
        <one-to-many name="items" refEntity="Item" joinColumn="product_id"/>
        <many-to-many name="suppliers" refEntity="Supplier"
otherColumn="supplier_id" intermediateTable="supply_product"/>
    </entity>
    <entity name="Supplier" tableName="supplier">
        <id columnName="supplier_id"/>
        <property name="name" columnName="name" type="String"/>
        <property name="address" columnName="address" type="String"/>
        <one-to-many name="tasks" refEntity="Task" joinColumn="supplier_id"/>
        <many-to-many name="products" refEntity="Product"
        otherColumn="product_id" intermediateTable="supply_product"/>
    </entity>
</model>
```

รูปที่ 5-3 XML File ส่วนที่อธิบาย ER ของ Supply Database

```
<?xml version="1.0" encoding="UTF-8"?>
<model name="ubcservice" desc="UBC Service Model" dbAdaptor="mysql"
host="localhost" database="ubc" user="root" password="CL2a7537">
    <entity name="Worker" tableName="worker">
        <id columnName="worker_id"/>
        <property name="firstName" columnName="first_name" type="String"/>
        <property name="lastName" columnName="last_name" type="String"/>
        <property name="username" columnName="username" type="String"/>
        <property name="password" columnName="password" type="String"/>
        <one-to-many name="servicepoints" refEntity="ServicePoint"
        joinColumn="worker_id"/>
    </entity>
    <entity name="ServicePoint" tableName="servicepoint">
        <id columnName="servicepoint_id"/>
        <property name="assignedTime" columnName="assigned_time"
        type="DateTime"/>
        <property name="finishedTime" columnName="finished_time"
        type="DateTime"/>
        <property name="description" columnName="description" type="String"/>
        <property name="comment" columnName="comment" type="String"/>
        <property name="price" columnName="price" type="Float"/>
        <foreignkey name="worker" refEntity="Worker" columnName="worker_id"/>
        <foreignkey name="customer" refEntity="Customer"
        columnName="customer_id"/>
        <one-to-many name="jobs" refEntity="Job"
        joinColumn="servicepoint_id"/>
    </entity>
    <entity name="Job" tableName="job">
        <id columnName="job_id"/>
        <property name="comment" columnName="comment" type="String"/>
        <foreignkey name="servicePoint" refEntity="ServicePoint"
        columnName="servicepoint_id"/>
        <one-to-many name="usedEquipments" refEntity="UsedEquipment"
        joinColumn="job_id"/>
    </entity>
    <entity name="UsedEquipment" tablename="used_equipment">
        <id columnName="used_equipment_id"/>
        <property name="name" columnName="name" type="String"/>
        <property name="quantity" columnName="quantity" type="Integer"/>
        <foreignkey name="job" refEntity="Job" columnName="job_id"/>
    </entity>
    <entity name="Customer" tableName="customer">
        <id columnName="customer_id"/>
        <property name="firstName" columnName="first_name" type="String"/>
        <property name="lastName" columnName="last_name" type="String"/>
        <property name="address" columnName="address" type="String"/>
        <one-to-many name="servicePoint" refEntity="ServicePoint"
        joinColumn="customer_id"/>
    </entity>
</model>
```

รูปที่ 5-4 XML File ส่วนที่อธิบาย ER ของ UBC Service Database

```
<xmlroot name="staff" entity="Staff" clientusername="Staff.username"
clientpassword="Staff.password">
    <attribute property="id" name="id"/>
    <attribute property="firstName" name="firstname"/>
    <attribute property="lastName" name="lastname"/>
    <element name="task" entity="Task">
        <attribute property="id" name="id"/>
        <attribute property="finishedTime" name="finishedtime"/>
        <attribute property="supplier.address" name="supplieraddress"/>
        <attribute property="supplier.name" name="suppliername"/>
        <element name="item" entity="Item">
                <attribute property="id" name="id"/>
                <attribute property="requiredQuantity"
                name="requiredquantity"/>
                <attribute property="receivedQuantity"
                name="receivedquantity"/>
                <attribute property="ppu" name="priceperunit"/>
                <attribute property="product.name" name="productname"/>
        </element>
    </element>
</xmlroot>
```

รูปที่ 5-5 XML File ส่วนที่อธิบาย XML-Relational Mapping ของ Supply Database

```
<xmlroot name="worker" entity="Worker" clientusername="Worker.username"
clientpassword="Worker.password">
    <attribute property="id" name="id"/>
    <attribute property="firstName" name="firstname"/>
    <attribute property="lastName" name="lastname"/>
    <element name="servicepoint" entity="ServicePoint">
        <attribute property="id" name="id"/>
        <attribute property="customer.firstName" name="customerfirstname"/>
        <attribute property="customer.lastName" name="customerlastname"/>
        <attribute property="customer.address" name="customeraddress"/>
        <attribute property="assignedTime" name="assignedtime"/>
        <attribute property="finishedTime" name="finishedtime"/>
        <attribute property="description" name="description"/>
        <attribute property="comment" name="comment"/>
        <attribute property="price" name="price"/>
        <element name="job" entity="Job">
                <attribute property="id" name="id"/>
                <attribute property="comment" name="comment"/>
                <element name="usedequipment" entity="UsedEquipment">
                        <attribute property="id" name="id"/>
                        <attribute property="name" name="name"/>
                        <attribute property="quantity" name="quantity"/>
                </element>
        </element>
    </element>
</xmlroot>
```

รูปที่ 5-6 XML File ส่วนที่อธิบาย XML-Relational Mapping ของ UBC Service Database

5.4 XMLForm Configuration File

เป็นการทดลองสร้าง user interface สำหรับ XMLForm สำหรับทั้ง 2 database

```
<?xml version="1.0" encoding="UTF-8"?>
<startpage name="Welcome" label="Please Select Supplier" match="/staff/task"
combobox="@suppliername" username="Staff.username"
password="Staff.password">
    <description row="2" label="Name" match="@suppliername"/>
    <description row="4" label="Address" match="@supplieraddress"/>
    <form name="Supplier" match="/staff/task">
        <static row="2" label="Name" match="@suppliername"/>
        <static row="4" label="Address" match="@supplieraddress"/>
        <form label="Items" name="Items" match="item" combobox="@productname">
              <static label="Name" match="@productname"/>
              <static label="PPU" match="@priceperunit"/>
              <static label="Required Qty" match="@requiredquantity"/>
              <edit label="Received Qty" match="@receivedquantity"/>
        </form>
    </form>
</startpage>
```

รูปที่ 5-7 XML ที่อธิบาย User Interface ของ Supply Database

```
<?xml version="1.0" encoding="UTF-8"?>
<startpage name="UBC Services" label="Please Select Service Point"
match="/staff/servicepoint" combobox="@customerfirstname"
username="Worker.username" password="Worker.password">
    <description label="Firstname" match="@customerfirstname"/>
    <description label="Lastname" match="@customerlastname"/>
    <description row='1' label="Address" match="@customeraddress"/>
    <description row='2' label="Description" match="@description"/>
    <form name="Service Point" match="/staff/servicepoint">
        <static label="Firstname" match="@customerfirstname"/>
        <static label="Lastname" match="@customerlastname"/>
        <static row='1' label="Address" match="@customeraddress"/>
        <static row='2' label="Description" match="@description"/>
        <form label="Click to show/add jobs" name="Jobs" match="job"
        combobox="@comment" addtext="Add a job">
              <form label="Click to show/add Equipments" name="Equipments"
              match="usedequipment" combobox="@name" addtext="Add a
              equipment">
                    <lookup label="Name" match="@name"
                    from="/equipments/equipment"/>
                    <edit label="Quantity" match="@quantity"/>
              </form>
              <edit row='3' label="Comment" match="@comment"/>
        </form>
        <edit label="Price" match="@price"/>
        <edit row='3' label="Comment" match="@comment"/>
    </form>
</startpage>
```

รูปที่ 5-8 XML ที่อธิบาย User Interface ของ UBC Service Database

5.5 ทดสอบสร้าง Data Exchange Class

ขั้นตอนนี้เป็นการสร้าง Data exchanger ของ supply database และ ubc database ผ่าน Interactive Shell โดยเป็นการสร้างโดยใช้ Remote Management แต่เป็นการทดลองใช้ Remote Management โดยตรงโดยไม่ผ่านกลไกของ Pyro

```
>>> from xmf.runtime import remotemanagement
>>> manage = remotemanagement.RemoteManagement()
>>> f = file('E:\\work\\project\\samples\\xml\\supplymodel.xml')
>>> xmlContent = f.read()
>>> manage.addDataExchanger(xmlContent, '')
begin add data exchanger
write to file E:\work\xmf\userdir\sqlobj\supply.py complete
write to file E:\work\xmf\userdir\xmlobj\supply.py complete
write to file E:\work\xmf\userdir\xchobj\supply.py complete
write to file E:\work\xmf\userdir\upsobj\supply.py complete
finish add data exchanger
>>>
>>> f = file('E:\\work\\project\\samples\\xml\\ubcmodel.xml')
>>> xmlContent = f.read()
>>> manage.addDataExchanger(xmlContent, '')
begin add data exchanger
write to file E:\work\xmf\userdir\sqlobj\ubcservice.py complete
write to file E:\work\xmf\userdir\xmlobj\ubcservice.py complete
write to file E:\work\xmf\userdir\xchobj\ubcservice.py complete
write to file E:\work\xmf\userdir\upsobj\ubcservice.py complete
finish add data exchanger
>>>
```

จาก Output แสดงว่าสามารถสร้าง Data Exchanger class, SQLObject class และ XMLObject class ได้ถูกตำแหน่งของ directory

5.6 ทดสอบการ Retrieve XML จาก Database

เป็นการทดสอบการ retrieve ข้อมูลจาก database โดยสร้าง instance ของ class Exchanger กำหนด condition แบบง่ายคือให้ select staff ที่มี ID = 1

```
>>> from xchobj import ubcservice
>>> exchanger = ubcservice.Exchanger()
>>> conditionDict = {"Worker":"Worker.id==1"}
>>> xml = exchanger.select(conditionDict)
>>> print xml
<worker lastname="Songkran" firstname="Wara" id="1">
  <servicepoint description="router failed" customerfirstname="Tim" customeraddress="MUFC
America" customerlastname="Howard" assignedtime="2005-03-22 09:00:00.00" id="1"/>
  <servicepoint description="speed of network slow" customerfirstname="Carlo"
customeraddress="Chelsea FC Italy" customerlastname="Cudicini" assignedtime="2005-03-22
11:45:00.00" id="5"/>
  <servicepoint description="router failed" customerfirstname="Juseppe"
customeraddress="MUFC Italy" customerlastname="Rossi" assignedtime="2005-03-22
14:00:00.00" id="10"/>
</worker>
>>>
```

จาก output แสดงว่า Data Exchanger สามารถดึงข้อมูลได้อย่างถูกต้อง แสดงว่า Application Module Generator สามารถทำงานได้ถูกต้อง

5.7 ทดสอบการ Retrieve XML จาก Database โดยใช้ Condition ที่มีความซับซ้อน

โดยกำหนดให้ condition ต่างๆดังนี้ Staff.id = 1 และ ServicePoint.assigedTime = DATE('2005-03-06') คือให้ Staff Id = 1 และ วันที่มอบหมายงานให้คือวันที่ 02-03-2005

```
>>> conditionDict = {"Worker":"Worker.id==1", "ServicePoint":
"DATE(ServicePoint.assignedTime)>DATE('2005-03-06')"}
>>> xml = exchanger.select(conditionDict)
>>> print xml
<worker lastname="Songkran" firstname="Wara" id="1">
  <servicepoint description="router failed" customerfirstname="Tim"
customeraddress="MUFC America" customerlastname="Howard" assignedtime="2005-03-08
09:00:00.00" id="1"/>
  <servicepoint description="speed of network slow" customerfirstname="Carlo"
customeraddress="Chelsea FC Italy" customerlastname="Cudicini" assignedtime="2005-03-09
11:45:00.00" id="5"/>
</worker>
>>>
```

จาก output Data Exchanger สามารถ select โดยใช้ condition มากกว่า 1 entity โดยในตัวอย่างสามารถกำหนด แondition ทั้ง Staff entity และ ServicePoint entity

5.8 ทดสอบการ Retrieve XML จาก Database โดยใช้ Condition ที่อ้างอิงถึง Entity ที่ผ่าน Foreign key

จากตัวอย่างเป็นการกำหนด condition โดยให้ select ServicePoint ที่ Address ของ Customer เริ่มต้นด้วย "MUFC"

Condition ที่กำหนดคือ Staff.id = 1และ ServicePoint.customer.address.startswith('MUFC')

```
>>> conditionDict = {"Worker":"Worker.id==1", "ServicePoint":
"ServicePoint.customer.address.startswith('MUFC')"}
>>> xml = exchanger.select(conditionDict)
>>> print xml
<worker lastname="Songkran" firstname="Wara" id="1">
  <servicepoint description="router failed" customerfirstname="Tim"
customeraddress="MUFC America" customerlastname="Howard" assignedtime="2005-03-08
09:00:00.00" id="1"/>
  <servicepoint description="router failed" customerfirstname="Juseppe"
customeraddress="MUFC Italy" customerlastname="Rossi" assignedtime="2005-03-06
14:00:00.00" id="10"/>
</worker>
>>>
```

จาก Output Data Exchanger สามารถ select ข้อมูลโดย condition ที่อ้างอิงผ่าน entity อื่นได้

5.9 ทดสอบการ Update ข้อมูลผ่าน XML

โดยการเพิ่ม child element job ให้กับ element servicepoint ที่ select มาจากหัวข้อที่แล้วโดยมีข้อมูลดังนี้

```
<job comment="Very hard work :(" id="44">
  <usedequipment quantity="10" name="Equipment1" id="3"/>
  <usedequipment quantity="20" name="Equipment2" id="4"/>
</job>
```

ผลลัพธ์ที่ได้คือ

```
>>> print xml
<worker lastname="Songkran" firstname="Wara" id="1">
  <servicepoint description="router failed" customerfirstname="Tim"
customeraddress="MUFC America" customerlastname="Howard" assignedtime="2005-03-08
09:00:00.00" id="1">
    <job comment="Very hard work :(">
      <usedequipment quantity="10" name="Equipment1"/>
      <usedequipment quantity="20" name="Equipment2"/>
    </job>
  </servicepoint>
  <servicepoint description="router failed" customerfirstname="Juseppe"
customeraddress="MUFC Italy" customerlastname="Rossi" assignedtime="2005-03-06
14:00:00.00" id="10"/>
</worker>
>>> newxml = exchanger.update(xml)
>>> print newxml
<worker lastname="Songkran" firstname="Wara" id="1">
  <servicepoint description="router failed" customerfirstname="Tim"
customeraddress="MUFC America" customerlastname="Howard" assignedtime="2005-03-08
09:00:00.00" id="1">
    <job comment="Very hard work :(" id="44">
      <usedequipment quantity="10" name="Equipment1" id="3"/>
      <usedequipment quantity="20" name="Equipment2" id="4"/>
    </job>
  </servicepoint>
  <servicepoint description="router failed" customerfirstname="Juseppe"
customeraddress="MUFC Italy" customerlastname="Rossi" assignedtime="2005-03-06
14:00:00.00" id="10"/>
</worker>
>>>
```

job_id	servicepoint_id	comment
44	1	Very hard work :(

used_e...	name	quantity	job_id
3	Equipment1	10	44
4	Equipment2	20	44

รูปที่ 5-9 ข้อมูลใน Database หลังจาก update

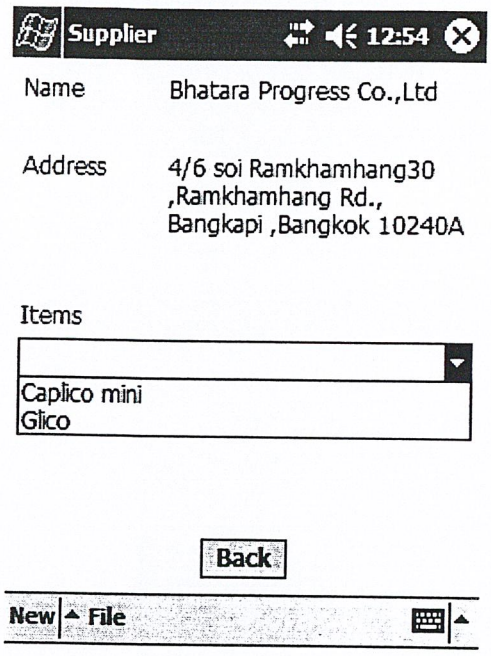
จากการทดสอบ Data Exchanger สามารถ update ข้อมูลบน database โดยใช้ XML ได้อย่างถูกต้อง โดยสังเกตุว่า XML element ที่ return กลับมาจะมี attribute ID เพื่อบ่งบอกว่าเป็น element ที่สัมพันธ์กับ row ที่มีอยู่ใน Database table แล้ว

5.10 ทดสอบการสร้าง user interface จาก XMLForm configuration file

ผลลัพธ์ user interface ที่ได้จาก XMLForm configurations file ซึ่งใช้แสดงข้อมูลของ supply database ในรูป 5.7

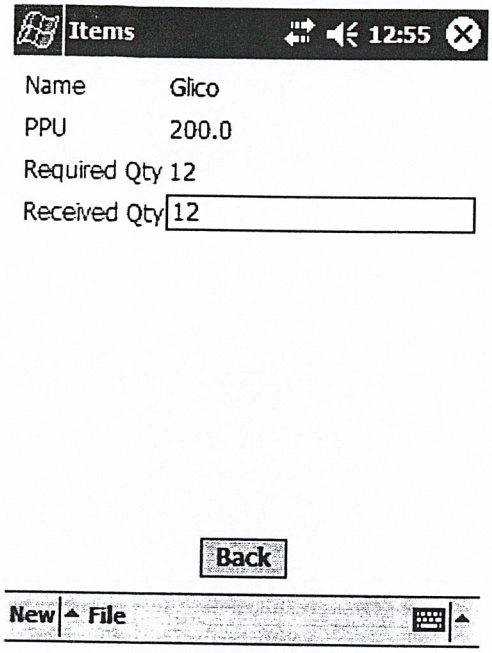

```
<startpage name="Welcome" label="Please Select Supplier" match="/staff/task"
combobox="@suppliername" username="Staff.username"
password="Staff.password">
    <description row="2" label="Name" match="@suppliername"/>
    <description row="4" label="Address" match="@supplieraddress"/>
    ...
</startpage>
```

รูปที่ 5-10 User interface และ XForm ในหน้าแรกของ supply application




```
<form name="Supplier" match="/staff/task">
    <static row="2" label="Name" match="@suppliername"/>
    <static row="4" label="Address" match="@supplieraddress"/>
    <form label="Items" name="Items" match="item" combobox="@productname">
        ...
    </form>
</form>
```

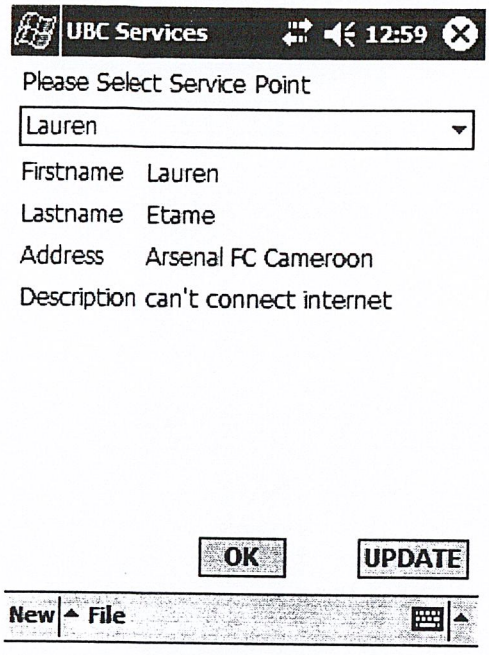
รูปที่ 5-11 User interface และ XForm ในหน้าที่สองของ supply application




```
<form label="Items" name="Items" match="item" combobox="@productname">
    <static label="Name" match="@productname"/>
    <static label="PPU" match="@priceperunit"/>
    <static label="Required Qty" match="@requiredquantity"/>
    <edit label="Received Qty" match="@receivedquantity"/>
</form>
```

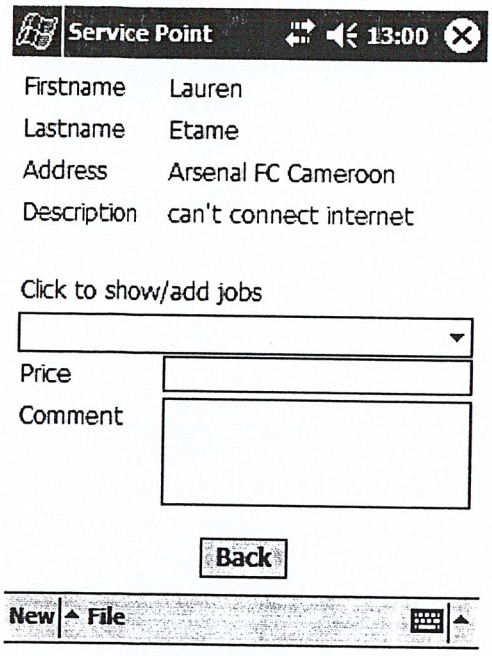
รูปที่ 5-12 User interface และ XForm ในหน้าที่สามของ supply application

ผลลัพธ์ user interface ที่ได้จาก XMLForm configurations file ซึ่งใช้แสดงข้อมูลของ UBC service database ในรูป 5.8



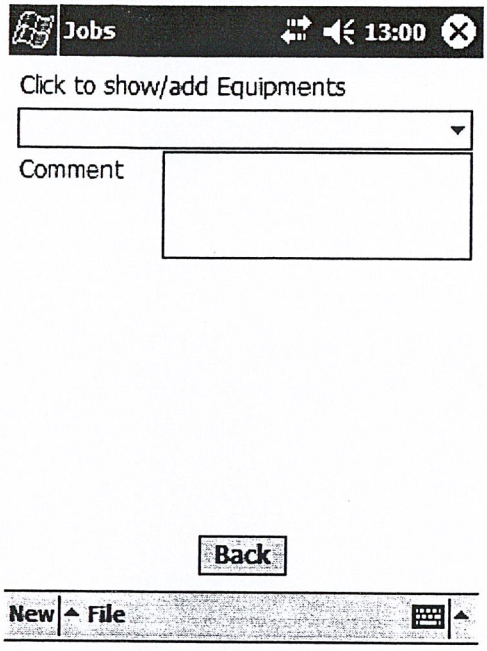

```
<startpage name="UBC Services" label="Please Select Service Point"
match="/staff/servicepoint" combobox="@customerfirstname"
username="Worker.username" password="Worker.password">
    <description label="Firstname" match="@customerfirstname"/>
    <description label="Lastname" match="@customerlastname"/>
    <description row='1' label="Address" match="@customeraddress"/>
    <description row='2' label="Description" match="@description"/>
    …
</startpage>
```

รูปที่ 5-13 User interface และ XForm ในหน้าแรกของ UBC service application



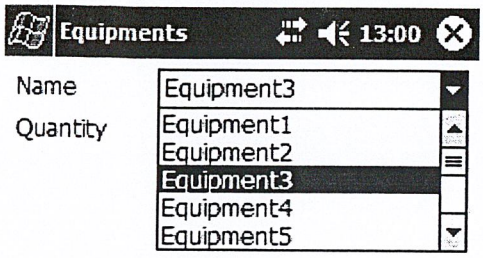

```
<form name="Service Point" match="/staff/servicepoint">
    <static label="Firstname" match="@customerfirstname"/>
    <static label="Lastname" match="@customerlastname"/>
    <static row='1' label="Address" match="@customeraddress"/>
    <static row='2' label="Description" match="@description"/>
    <form label="Click to show/add jobs" name="Jobs" match="job"
combobox="@comment"        addtext="Add a job">
    …
    </form>
    <edit label="Price" match="@price"/>
    <edit row='3' label="Comment" match="@comment"/>
</form>
```

รูปที่ 5-14 User interface และ XForm ในหน้าที่สองของ UBC service application




```
<form label="Click to show/add jobs" name="Jobs" match="job"
combobox="@comment" addtext="Add a job">
    <form label="Click to show/add Equipments" name="Equipments"
match="usedequipment"      combobox="@name" addtext="Add a equipment">
        …
    </form>
    <edit row='3' label="Comment" match="@comment"/>
</form>
```

รูปที่ 5-15 User interface และ XForm ในหน้าที่สามของ UBC service application




```
<form label="Click to show/add Equipments" name="Equipments"
match="usedequipment" combobox="@name" addtext="Add a equipment">
    <lookup label="Name" match="@name" from="/equipments/equipment"/>
    <edit label="Quantity" match="@quantity"/>
</form>
```

รูปที่ 5-16 User interface และ XForm ในหน้าที่สี่ของ UBC service application

บทที่ 6
บทวิจารณ์และสรุป

6.1 บทสรุปในการทดลองใช้ framework พัฒนา application

จากการทดลองสร้าง supply application และ UBC service application ในบทที่แล้วพบว่า framework สามารถตอบสนองการพัฒนา application ได้เป็นอย่างดีโดย

1. สามารถพัฒนาแอพลิเคชั่นบันทึกข้อมูลบนอุปกรณ์พกพาได้ง่ายและรวดเร็วเทียบกับวิธีการเขียนโปรแกรมเองทั้งหมด
2. สามารถเปลี่ยนรูปแบบของฟอร์มในการบันทึกข้อมูลได้อย่างรวดเร็วด้วยการเปลี่ยน XML configuration file
3. framework สามารถรองรับการพัฒนาแอพลิเคชั่นบันทึกข้อมูลบนอุปกรณ์พกพาได้หลากหลายรูปแบบทั้ง application ที่ใช้ในการบันทึกการจัดส่งไปรณีย์ บันทึกการรับ-ส่งของ และบันทึกข้อมูลในการให้บริการลูกค้าของช่างซ่อมบำรุง
4. สามารถเชื่อมโยงข้อมูลกับ relational database ได้อย่างถูกต้องโดยไม่จำกัดรูปแบบของ database schema
5. server สามารถรองรับการทำงานของ application หลายตัวพร้อมกันได้

6.2 แนวทางในการพัฒนาขั้นต่อไป

1. framework ปัจจุบันทำงานกับข้อมูลใน relational table ที่ไม่มีการเปลี่ยนแปลงตลอดระยะเวลาตั้งแต่ดึง XML data จาก server จนกระทั่ง update ข้อมูลกลับคืน แม้ว่าวิธีนี้จะเพียงพอต่อ application ส่วนใหญ่ แต่ในกรณีที่ต้องการข้อมูลแบบ real-time และข้อมูลมีการเปลี่ยนแปลงตลอดเวลา จะต้องมีการใช้ transaction เข้ามาช่วยและ server จะต้องเข้ามาควบคุม transaction ระหว่าง client ด้วย
2. user interface ที่สร้างได้ในปัจจุบันไม่สลับซับซ้อนมากพอยังขาด control หลายตัวเช่น spinner และ date-time chooser เป็นต้น รวมทั้งการวางตำแหน่งของ control ในแบบต่างๆด้วย
3. user interface ยังไม่รองรับการตรวจสอบขอบเขตของข้อมูลที่ใส่ได้ในแต่ละ field และ XML lookup file ยังเป็น static file ซึ่งต้องปรับปรุงให้สามารถสร้างจากข้อมูลบน database ได้
4. การติดต่อระหว่าง client และ server ยังไม่รองรับ data encryption และการรองรับ authentication โดยใช้ directory service
5. การ map ข้อมูลระหว่าง XML และ relational database ยังสามารถทำได้เพียง 4 data type คือ string, integer, float และ datetime ซึ่งยังต้องเพิ่ม data type อื่นอีกเช่น BLOB เป็นต้น

บรรณานุกรม

[1] Mapping DTDs to Databases http://www.rpbourret.com/xml/DTDToDatabase.htm

[2] XML and Databases http://www.rpbourret.com/xml/XMLAndDatabases.htm

[3] Mapping W3C Schemas to Object Schemas
to Relational Schemas http://www.rpbourret.com/xml/SchemaMap.htm

[4] Foundations of
Object-Relational Mapping http://www.chimu.com/publications/objectRelational/

[5] Elliotte Rusty Harold "Effective XML: 50 Specific Ways to Improve Your XML" by Addison-Wesley Professional; 1st edition (September 12, 2003)

[6] Python SQLObject http://sqlobject.org/

[7] Python XML Object http://xmlobject.base-art.net/

[8] Cheetah Template http://www.cheetahtemplate.org/

[9] Modeling
Object-Relational Bridge for python http://modeling.sourceforge.net/

[10] Validus Medical Systems Python 2.3.4 for Windows/CE http://fore.validus.com/~kashtan/

[11] Pyro Python Remote Objects http://pyro.sourceforge.net/

[12] PyXML XML package for Python http://pyxml.sourceforge.net/

[13] elementtree http://effbot.org/zone/element-index.htm