บันทึกวิจัย

การพัฒนาและออกแบบ MDL Analyzer

นัยนา สหเวชชภัณฑ์ และคำรณ อรุณเรื่อ

โครงการกริดสารสนเทศ

หน่วยปฏิบัติการวิจัยการจำลองขนาดใหญ่
ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ

กันยายน 2551

สารบัญ

บทคัดย่อ.....5
1. บทนำ.....6
2. ความต้องการในการพัฒนา MDL Analyzer.....9
2.1 รายละเอียดความต้องการของ MDL Analyzer.....9
2.2 Use case Diagram สำหรับ MDL Analyzer.....10
3. การออกแบบ MDL Analyzer.....13
3.1 สถาปัตยกรรมของ MDL Analyzer.....13
3.2 Package Diagram และ Class Diagram ของ MDL Analyzer.....14
3.4 Sequence Diagram.....18
4 รูปแบบของ input และ output ของ MDL Analyzer.....20
4.1 ตัวอย่างที่ 1.....20
4.2 ตัวอย่างที่ 2.....21
4.3 ตัวอย่างที่ 3.....24
4.4 ตัวอย่างที่ 4.....26
4.5 ตัวอย่างที่ 5.....28
5 การใช้งาน MDL Analyzer.....32
6 สรุป.....33
ภาคผนวก ก.....35
ก.1 Class MdlExtraction.....35
ก.2 Class MdlExtractionString.....37
ก.3 Class AbstractType.....39
ก.4 Class TreeMdl.....40
ก.5 Class SimpleType.....42
ก.6 Class XMLElement.....44
ก.7 Class XMLSchema.....45
ก.8 Class ObjectAdapterJaxb.....46
ก.9 Class XmlSchemaObject.....47
ภาคผนวก ข การใช้งานของ MDL Analyzer ร่วมกับ GiSTool.....50

สารบัญรูป

รูปที่ 1-1 คำสั่งการสืบค้นข้อมูลนักวิจัยที่เชี่ยวชาญ “ไข้หวัดนก”..... 6
รูปที่ 1-2 ความสัมพันธ์ระหว่างผู้ใช้งานกริดสารสนเทศ และกริดสารสนเทศ 6
รูปที่ 2-1 ความสัมพันธ์ระหว่าง MDL Analyzer ต่อ เครื่องมือ และแอพลิเคชั่นที่สนับสนุนการใช้งานกริดสารสนเทศ..... 9
รูปที่ 2-2 Use case diagram สำหรับ MDL Analyzer..... 10
รูปที่ 3-1 สถาปัตยกรรมของ MDL Analyzer 13
รูปที่ 3-2 Package diagram ของ MDL Analyzer..... 14
รูปที่ 3.3 Class diagram ของ Package MdlExtraction..... 15
รูปที่ 3.4 Class diagram ของ Package TreeMdl 16
รูปที่ 3.6 Class diagram ของ Package AdapterPattern..... 16
รูปที่ 3.5 Class diagram ของ Package ObjectMdl..... 17
รูปที่ 3-6 Sequence Diagram สำหรับ use case “transform metadata schema to tree”..... 18
รูปที่ 3-7 Sequence Diagram สำหรับ use case “transform metadata schema to object model”..... 19
รูปที่ 4-1 เอกสาร MDL Project บนพื้นฐานของ XML Schema แบบ 1 ลำดับชั้นโดยไม่มีการอ้างอิงถึงชนิดของเอลิเมนต์20
รูปที่ 4-2 Tree Object สำหรับเอกสาร MDL Project ในรูปที่ 4-1..... 21
รูปที่ 4-3 Object Model สำหรับเอกสาร MDL Project ในรูปที่ 4-1..... 21
รูปที่ 4-4 เอกสาร MDL Project บนพื้นฐานของ XML Schema แบบ 2 ลำดับชั้นโดยไม่มีการอ้างอิงถึงชนิดของเอลิเมนต์22
รูปที่ 4-5Tree Object สำหรับเอกสาร MDL Project ในรูปที่ 4-4..... 23
รูปที่ 4-6 Object Model สำหรับเอกสาร MDL Project ในรูปที่ 4-4..... 23
รูปที่ 4-7 เอกสาร MDL Project บนพื้นฐานของ XML Schema แบบ 3 ลำดับชั้นโดยไม่มีการอ้างอิงถึงชนิดของเอลิเมนต์24
รูปที่ 4-7 เอกสาร MDL Project บนพื้นฐานของ XML Schema แบบ 3 ลำดับชั้นโดยไม่มีการอ้างอิงถึงชนิดของเอลิเมนต์ (ต่อ) 25
รูปที่ 4-8 Tree Object สำหรับเอกสาร MDL Project ในรูปที่ 4-7..... 25
รูปที่ 4-9 Object Model สำหรับเอกสาร MDL Project ในรูปที่ 4-7..... 26
รูปที่ 4-10 เอกสาร MDL Project บนพื้นฐานของ XML Schema แบบ 2 ลำดับชั้นโดยมีการอ้างอิงถึงชนิดของเอลิเมนต์ . 27
รูปที่ 4-11 Tree Object สำหรับเอกสาร MDL Project ในรูปที่ 4-10..... 28
รูปที่ 4-12 Object Model สำหรับเอกสาร MDL Project ในรูปที่ 4-10..... 28
รูปที่ 4-13 เอกสาร MDL Project บนพื้นฐานของ XML Schema แบบ 3 ลำดับชั้นโดยมีการอ้างอิงถึงชนิดของเอลิเมนต์ . 29
รูปที่ 4-13 เอกสาร MDL Project บนพื้นฐานของ XML Schema แบบ 3 ลำดับชั้นโดยมีการอ้างอิงถึงชนิดของเอลิเมนต์ (ต่อ) 30
รูปที่ 4-14 Tree Object สำหรับเอกสาร MDL Project ในรูปที่ 4-13..... 31

สารบัญตาราง

ตารางที่ 1-1 Use case “extract metadata schema”…….……..11

ตารางที่ 2-2 Use case “transform metadata schema to tree” ……..11

ตารางที่ 2-3 Use case “transform metadata schema to object model” **…….……...12**

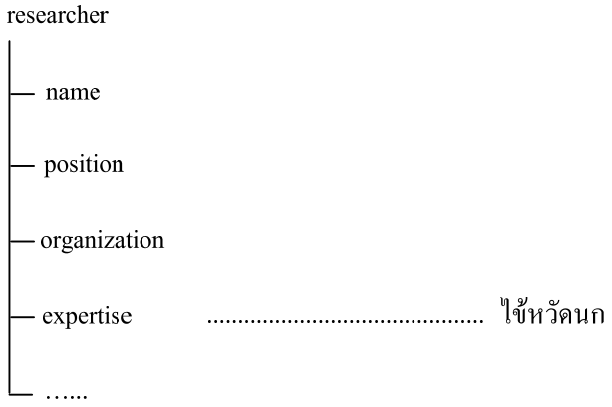
ตารางที่ 6-1 ความสามารถของ MDL Analyzer**……...33**

บทคัดย่อ

MDL Analyzer version 1.0 เป็นต้นแบบระดับห้องปฏิบัติการ ทำหน้าที่ในการสกัด และแปลงโครงสร้างข้อมูล ใน Application Metadata ของเอกสาร MDL ใดๆ ให้อยู่ในรูปของ Tree และ Object Model ทั้งนี้เพื่อให้สามารถรองรับกับการใช้งานของ Information Grid Web Application, Information Grid Client API และ Generic Information Service Tool (GiSTool) ให้มีความคล่องตัวมากขึ้น โดยไม่ยึดติดกับชุดของเอกสาร MDL หนึ่งๆ ในปัจจุบัน MDL Analyzer version 1.0 Beta นี้สามารถใช้งานกับโครงสร้างข้อมูลที่อยู่ในรูปของ XML schema ใน 2 ลักษณะ คือ แบบลำดับชั้นที่ไม่มีการอ้างอิงถึงชนิดของเอลิเมนต์ และแบบลำดับชั้นที่มีการอ้างอิงถึงชนิดของเอลิเมนต์ (ที่อยู่ภายใต้โครงสร้างเดียวกัน) ยกเว้นการแปลงเป็น Object Model นั้น ยังคงจำกัดอยู่ที่แบบ 1 ลำดับชั้นที่มีการอ้างอิงถึงชนิดของเอลิเมนต์

1. บทนำ

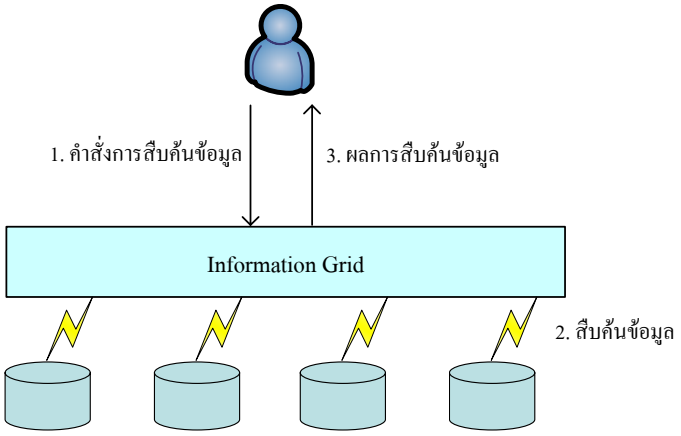
กริดสารสนเทศ (Information Grid) [1] เป็นโครงสร้างพื้นฐานที่สนับสนุนการสืบค้น และบูรณาการข้อมูลใดๆ จากแหล่งข้อมูลต่างๆที่เข้ามาเชื่อมต่อกับกริดสารสนเทศ โดยการสืบค้นและบูรณาการข้อมูลหนึ่งๆนั้น จะอยู่บนพื้นฐานของมาตรฐานโครงสร้างข้อมูล (metadata schema หรือ data structure) ที่ใช้สำหรับอธิบายข้อมูลนั้นๆ เช่น ในการสืบค้นข้อมูลนักวิจัยที่มีความเชี่ยวชาญ "ไข้หวัดนก" นั้น สามารถทำได้โดยการกำหนดเงื่อนไข "ไข้หวัดนก" ในฟิลด์ข้อมูล "expertise" ของโครงสร้างข้อมูลนักวิจัย (researcher) ดังแสดงในรูปที่ 1-1



(ก) โครงสร้างข้อมูลนักวิจัย (ข) เงื่อนไขการสืบค้นข้อมูล

รูปที่ 1-1 คำสั่งการสืบค้นข้อมูลนักวิจัยที่เชี่ยวชาญ "ไข้หวัดนก"

รูปที่ 1-2 ได้แสดงความสัมพันธ์ระหว่างผู้ใช้งานกริดสารสนเทศ และกริดสารสนเทศ ซึ่งมีกระบวนการที่สำคัญ ดังนี้ (1) ผู้ใช้ส่งคำสั่งการสืบค้นข้อมูล ไปยังกริดสารสนเทศ โดยคำสั่งการสืบค้นข้อมูล (ดูรูปที่ 1-1) ประกอบด้วย โครงสร้างข้อมูลที่อธิบายข้อมูลที่จะทำการสืบค้น และเงื่อนไขที่ต้องการจะสืบค้น (2) กริดสารสนเทศทำการสืบค้นข้อมูลไปยังแหล่งข้อมูลต่างๆ เพื่อให้ได้มาซึ่งข้อมูลที่ผู้ใช้ต้องการ และ (3) กริดสนเทศทำการบูรณาการข้อมูลจากแหล่งข้อมูลต่างๆ และส่งผลลัพธ์กลับไปยังผู้ใช้นั้นๆ โดยผลลัพธ์นั้นจะมีรูปแบบตามโครงสร้างข้อมูลที่กำหนดไว้ในคำสั่งการสืบค้นข้อมูล



รูปที่ 2-2 ความสัมพันธ์ระหว่างผู้ใช้งานกริดสารสนเทศ และกริดสารสนเทศ

ภายใต้กริดสารสนเทศ มาตรฐานโครงสร้างข้อมูล จะถูกสร้างขึ้นมาโดยผู้เชี่ยวชาญทางด้านข้อมูลนั้นๆ ทั้งนี้มาตรฐานโครงสร้างข้อมูลนี้ ถือเป็นส่วนหนึ่งของภาษา MDL (Metadata Description Language) [2] ซึ่งเป็นภาษาที่กำหนดอยู่บนพื้นฐานของภาษา XML โดยมีวัตถุประสงค์เพื่อใช้ในการอธิบายรายละเอียดของโครงสร้างข้อมูลหนึ่งๆ นอกเหนือไปจากโครงสร้างข้อมูลนั้นๆ เช่น ชื่อโครงสร้างข้อมูล คำอธิบายโครงสร้างข้อมูล องค์กรที่เป็นผู้กำหนดโครงสร้างข้อมูล และวันที่กำหนด เป็นต้น โดยสรุปแล้วเอกสาร MDL หนึ่งๆ จะใช้ในการอธิบายโครงสร้างข้อมูลหนึ่งๆ และมีรหัสประจำเอกสาร MDL ที่ไม่ซ้ำกับเอกสาร MDL อื่นๆ ทั้งนี้ ภายใต้การทำงานของกริดสารสนเทศนั้น เอกสาร MDL นี้จะถูกใช้เป็นส่วนหนึ่งของ “คำสั่งการสืบค้นข้อมูล” เนื่องจากทำหน้าที่ในการแสดงถึงโครงสร้างข้อมูล

จากข้างต้น จะเห็นว่าโครงสร้างข้อมูล ถือเป็นพื้นฐานการทำงานขององค์ประกอบต่างๆของกริดสารเทศ ตลอดจนเครื่องมือ และแอพลิเคชั่น ที่สนับสนุนการใช้งานกริดสารสนเทศ ได้แก่

1. Information Grid Web Application
   แอพพลิเคชั่นที่อำนวยความสะดวกให้แก่ผู้ใช้ทั่วไปทำการสืบค้นข้อมูลใดๆ ผ่านกริดสารสนเทศ โดยแอพลิเคชั่นนี้จะแสดง Graphical User Interface (GUI) เพื่อให้ผู้ใช้สามารถเลือกมาตรฐานโครงสร้างข้อมูลที่จะทำการสืบค้น และอำนวยความสะดวกให้แก่ผู้ใช้ในการบันทึกเงื่อนไขตามฟิลด์ข้อมูลต่างๆ ที่กำหนดไว้ในโครงสร้างข้อมูลนั้นๆ
2. Information Grid Client API (Application Programming Interface)
   API ที่อำนวยความสะดวกให้แก่ผู้พัฒนาระบบ (system developer) ในการพัฒนาโปรแกรมเพื่อติดต่อกับกริดสารสนเทศ ผ่านทาง Information Broker [] (เป็นองค์ประกอบหนึ่งของกริดสารสนเทศ และเป็น “นายหน้า” ของแหล่งข้อมูลต่างๆ) ในการส่งคำสั่งการสืบค้นข้อมูล และรับผลการสืบค้นข้อมูล ทั้งนี้ ผลการสืบค้นควรอยู่ในรูปแบบที่เอื้อให้ผู้พัฒนาทำการพัฒนาโปรแกรมเพื่อจัดการกับข้อมูลได้ทันที
3. Generic Information Service Tool (GiSTool)
   เครื่องมือที่ใช้ในการสร้าง Information Service [] (เป็นองค์ประกอบหนึ่งของกริดสารสนเทศ และเป็น “ตัวแทน” ของแหล่งข้อมูลหนึ่งๆ) ที่ทำหน้าที่ในการแปลงคำสั่งการสืบค้นข้อมูลในกริดสารสนเทศ[1] ให้เป็นคำสั่งการสืบค้นข้อมูลที่แหล่งข้อมูลนั้นๆเข้าใจ[2] ดังนั้น GiSTool จึงจำเป็นที่จะต้องให้ผู้ดูแลแหล่งข้อมูลนั้นๆทำการจับคู่ความสัมพันธ์ระหว่างฟิลด์ข้อมูลที่กำหนดไว้ในมาตรฐานโครงสร้างข้อมูล และฟิลด์ข้อมูลที่กำหนดไว้ในแหล่งข้อมูลนั้น เพื่อเป็นพื้นฐานในการสร้าง Information Service

ทั้งนี้ เพื่ออำนวยความสะดวกในการพัฒนาเครื่องมือ และแอพลิเคชั่น ดังข้างต้นนั้น จำเป็นที่จะต้องมีเครื่องมือที่ทำการสกัดโครงสร้างข้อมูลที่กำหนดขึ้นบนพื้นฐานของภาษา MDL นั้น และแปลงโครงสร้างข้อมูลนั้นๆ ให้อยู่ในรูปแบบที่เหมาะสมต่อการนำไปใช้งาน ดังนั้น รายงานฉบับนี้ จึงได้นำเสนอถึงการออกแบบ และ

---

[1] คำสั่งการสืบค้นข้อมูล บนกริดสารเทศ ได้อ้างอิง CQL (Common Query Language) เพื่อเป็นภาษาคำสั่งการสืบค้นข้อมูล (Query Language) และมาตรฐานโครงสร้างข้อมูล

[2] คำสั่งการสืบค้นข้อมูล บนแหล่งข้อมูลหนึ่งๆ จะต้องอ้างอิงภาษาคำสั่งการสืบค้นข้อมูล และโครงสร้างข้อมูล ที่กำหนดไว้เฉพาะสำหรับแหล่งข้อมูลนั้นๆ เช่น แหล่งข้อมูลที่เป็น RDBMS ใช้ SQL เป็นภาษาคำสั่งการสืบค้นข้อมูล

พัฒนาเครื่องมือดังกล่าวที่เรียกว่า “MDL Analyzer” เพื่อใช้ในการสกัด และแปลงโครงสร้างข้อมูลในเอกสาร MDL หนึ่งๆ ให้อยู่ในรูปแบบที่เหมาะสม สำหรับเครื่องมือ และแอพลิเคชั่นหนึ่งๆ ของกริดสารสนเทศ

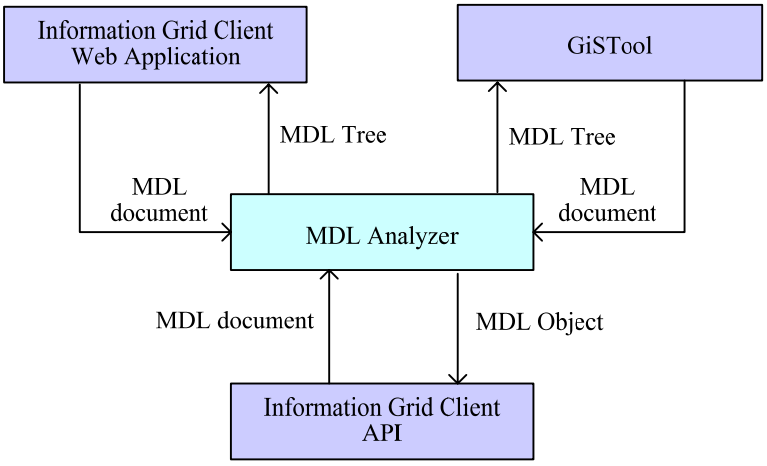
2. ความต้องการในการพัฒนา MDL Analyzer

2.1 รายละเอียดความต้องการของ MDL Analyzer

ความต้องการของ MDL Analyzer ต่อเครื่องมือ และแอพลิเคชั่น ที่สนับสนุนการใช้งานกริดสารสนเทศ มีดังนี้

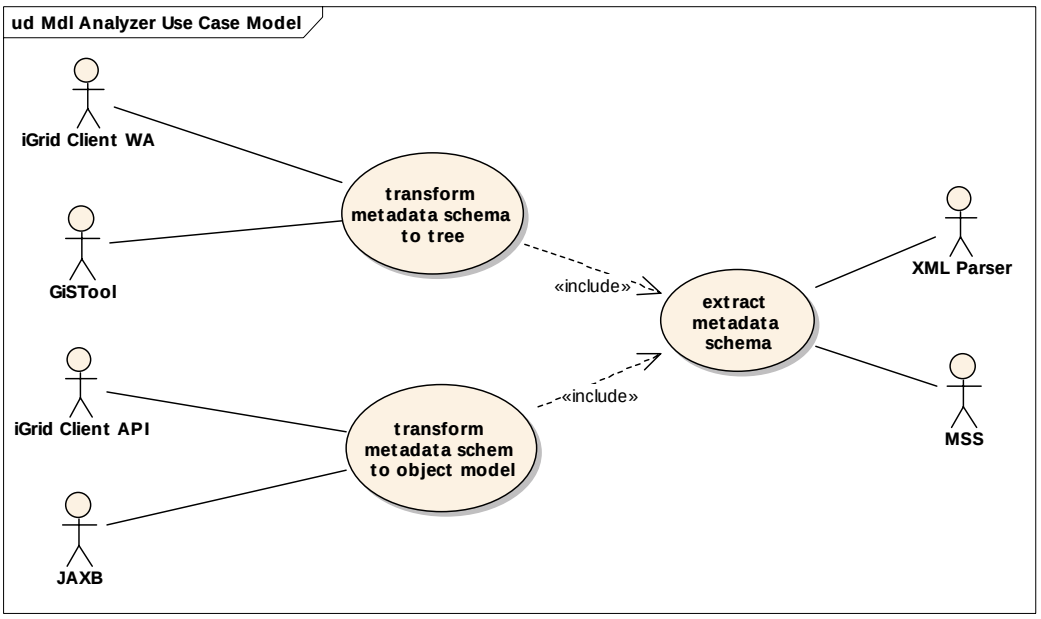
1. สกัดโครงสร้างข้อมูลในเอกสาร MDL หนึ่งๆ ซึ่งถูกอ้างอิงโดยรหัสประจำเอกสาร MDL นั้นๆ ทั้งนี้ โครงสร้างข้อมูลได้ถูกสร้างขึ้นบนพื้นฐานของ XML schema
2. แปลงโครงสร้างข้อมูลให้อยู่ในรูปของ
   - โครงสร้างต้นไม้ (tree data structure) ทั้งนี้เพื่อสนับสนุนการพัฒนา และการใช้งานของ Information Grid Client Web Application และ Generic Information Service Tool และ
   - Jar file ซึ่งบรรจุ object model ของโครงสร้างข้อมูล ทั้งนี้เพื่อสนับสนุนการพัฒนา และการใช้งานของ Information Grid Client API

โดยความสัมพันธ์ของ MDL Analyzer ต่อเครื่องมือ และแอพลิเคชั่น ที่สนับสนุนการใช้งานกริดสารสนเทศ สามารถแสดงได้ดังรูปที่ 2-1



รูปที่ 2-1 ความสัมพันธ์ระหว่าง MDL Analyzer ต่อ เครื่องมือ และแอพลิเคชั่นที่สนับสนุนการใช้งานกริดสารสนเทศ

2.2 Use case Diagram สำหรับ MDL Analyzer



รูปที่ 2-2 Use case diagram สำหรับ MDL Analyzer

รูปที่ 2-2 แสดง Use case diagram สำหรับ MDL Analyzer ซึ่งประกอบด้วย 3 use cases และ 5 actors ดังนี้

Use Cases

1. use case “extract metadata schema”
   สกัดโครงสร้างข้อมูล (metadata schema) ในเอกสาร MDL หนึ่งๆ
2. use case “transform metadata schema to tree”
   แปลงโครงสร้างข้อมูล (metadata schema) ให้อยู่ในรูปของโครงสร้างต้นไม้
3. use case “transform metadata schema to object model”
   แปลงโครงสร้างข้อมูล (metadata schema) ให้อยู่ในรูปของ Jar file ซึ่งบรรจุ object model ของโครงสร้างข้อมูล

Actors

1. iGrid Client WA (Information Grid Client Web Application)
   Web Application ที่พัฒนาบนพื้นฐานของกริดสารสนเทศ (ดูรายละเอียดในบทที่ 1)
2. iGrid Client API (Information Grid Client API)
   API ที่อำนวยความสะดวกให้นักพัฒนาในการเขียนโปรแกรมติดต่อกับกริดสารสนเทศ (ดูรายละเอียดในบทที่ 1)
3. GiSTool (Generic Information Service Tool)
   เครื่องในการสร้าง Information Service (ดูรายละเอียดในบทที่ 1)

4. MSS (Metadata Schema Service)

   เซอร์วิสในการสืบค้นเอกสาร MDL ตามรหัสประจำตัวเอกสาร MDL

5. JAXB

   เครื่องมือในการแปลงโครงสร้างข้อมูล เป็น Java object model สำหรับโครงสร้างข้อมูลหนึ่งๆ ที่ถูกสร้างขึ้นบนพื้นฐานของ XML schema

6. XML Parser

   เครื่องมือในการสกัดโครงสร้างข้อมูล ที่ถูกสร้างขึ้นบนพื้นฐานของ XML schema

Use case name	extract metadata schema
Participating actors	- ถูกเรียกใช้งาน iGrid WA, iGrid API, GiSTool - ติดต่อกับ MSS เพื่อให้ได้มาซึ่งเอกสาร MDL - ติดต่อกับ XML Parser เพื่อสกัด XML schema
Entry Condition	1. ได้รับรหัสประจำตัวเอกสาร MDL จาก use case “transform metadata schema to tree” และ “transform metadata schema to object model”
Flow of events	2. ติดต่อกับ MSS เพื่อให้ได้มาซึ่งเอกสาร MDL นั้นๆ 3. สกัดโครงสร้างข้อมูลที่อยู่ในเอลิเมนต์ของ “Application Metadata” ของเอกสาร MDL
Exit Condition	4. ส่งโครงสร้างข้อมูลที่สกัดแล้วให้ use case “transform metadata schema to tree” และ “transform metadata schema to object model”

ตารางที่ 2-1 Use case “extract metadata schema”

Use case name	transform metadata schema to tree
Participating actors	- ถูกเรียกใช้งาน iGrid WA และ GiSTool
Entry Condition	1. ได้รับรหัสประจำตัวเอกสาร MDL จาก iGrid WA และ GiSTool
Flow of events	2. ติดต่อกับ use case “extract metadata schema” เพื่อให้ได้โครงสร้างข้อมูลที่ได้รับการสกัด 3. แปลงโครงสร้างข้อมูลที่สกัดแล้ว ให้อยู่ในรูปแบบโครงสร้างต้นไม้ ซึ่งนำเสนอโดย DefaultMutableTreeNode Java object
Exit Condition	4. ส่ง DefaultMutableTreeNode Java object ให้ iGrid WA และ GiSTool

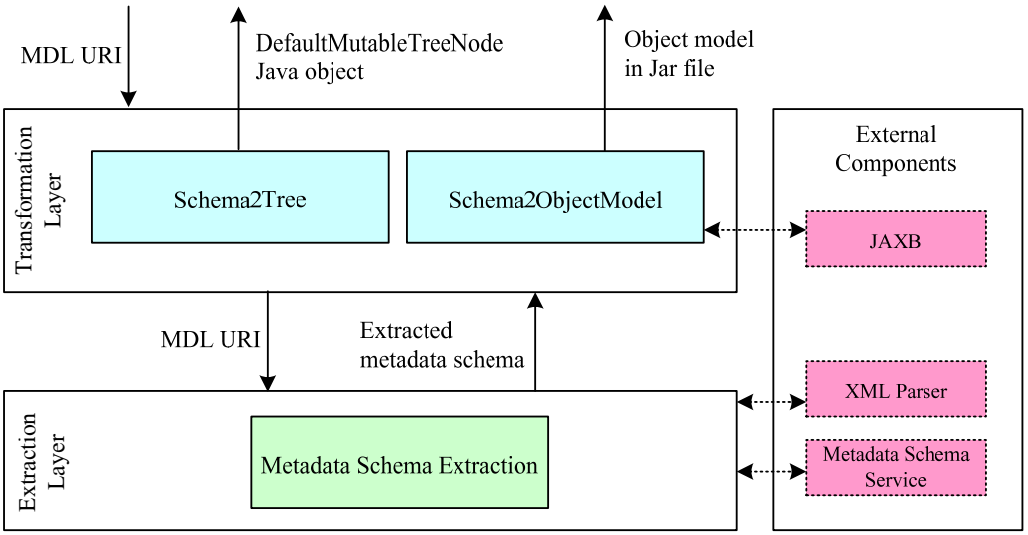
ตารางที่ 2-2 Use case “transform metadata schema to tree”

Use case name	transform metadata schema to object model
Participating actors	- ถูกเรียกใช้งาน iGrid Client API
Entry Condition	1. ได้รับรหัสประจำตัวเอกสาร MDL จาก iGrid Client API
Flow of events	2. ติดต่อกับ use case “extract metadata schema” เพื่อให้ได้โครงสร้างข้อมูลที่ได้รับการสกัด 3. ติดต่อกับ JAXB เพื่อแปลงโครงสร้างข้อมูลที่สกัดแล้ว ให้อยู่ในรูปแบบ object model
	4. บีบอัด object model ให้อยู่ในรูปของ Jar file
Exit Condition	5. ส่ง Jar file ให้ iGrid Client API

ตารางที่ 2-3 Use case “transform metadata schema to object model”

3. การออกแบบ MDL Analyzer

3.1 สถาปัตยกรรมของ MDL Analyzer



รูปที่ 3-1 สถาปัตยกรรมของ MDL Analyzer

รูปที่ 3-1 แสดงสถาปัตยกรรมของ MDL Analyzer ซึ่งประกอบด้วย 2 เลเย่อร์ ดังนี้

1. Extraction Layer

   เป็นเลเย่อร์ที่รับผิดชอบในการสกัด Application Metadata ซึ่งแสดงถึงโครงสร้างข้อมูลที่อยู่ในรูปแบบของ XML schema ในเอกสาร MDL เลเย่อร์นี้ประกอบด้วย 1 โมดูลที่ทำหน้าที่ดังกล่าว ซึ่งจะ (1) รับอินพุต "MDL URI (รหัสประจำเอกสาร MDL)" จากโมดูล Schema2Tree และ Schema2ObjectModel ที่อยู่ใน Transformation Layer (2) ติดต่อกับ Metadata Schema Service เพื่อให้ได้มาซึ่งเอกสาร MDL (3) ติดต่อกับ XML parser เพื่อทำการสกัด Application Metadata และ (4) จัดส่ง Application Metadata ที่ได้สกัดไว้ ให้โมดูลดังกล่าว

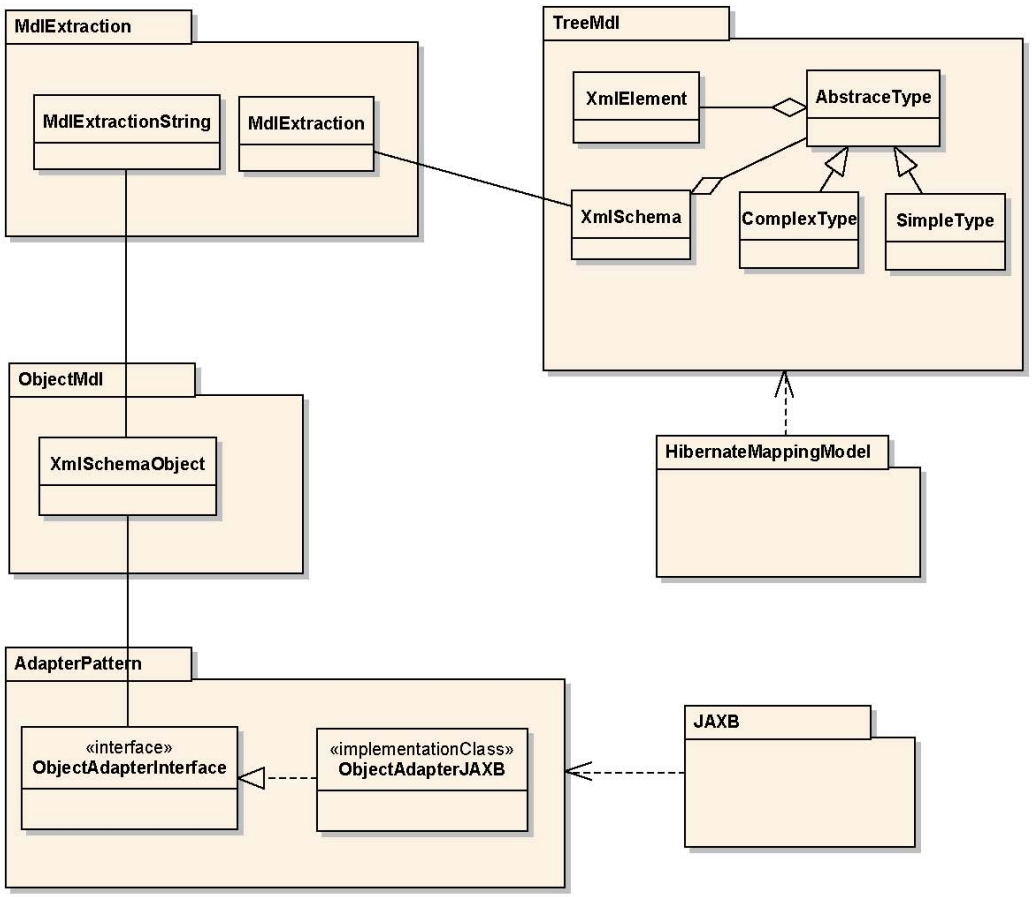
2. Transformation Layer

   เป็นเลเย่อร์ที่รับผิดชอบในการแปลงโครงสร้างข้อมูลที่ได้รับการสกัดแล้วให้อยู่ในรูปแบบโครงสร้างต้นไม้ หรือ object model เลเย่อร์นี้ประกอบด้วย 2 โมดูล คือ Schema2Tree และ Schema2ObjectModel โดย

   - Schema2Tree ทำหน้าที่แปลงโครงสร้างข้อมูลที่ได้รับการสกัดแล้วให้อยู่ในรูปแบบโครงสร้างต้นไม้ ซึ่งจะ (1) รับอินพุต "MDL URI (รหัสประจำเอกสาร MDL)" จาก iGrid Web Application หรือ GiSTool และ Metadata Schema ที่ได้รับการสกัดแล้วจากโมดูล Metadata Schema Extract ที่อยู่ใน Extraction Layter (2) แปลง Application Metadata ให้อยู่ในรูปของโครงสร้างต้นไม้ที่นำเสนอโดย DefaultMutableTreeNode Java object และ (3) จัดส่ง DefaultMutableTreeNode Java object ให้ iGrid Web Application หรือ GiSTool

- Schema2ObjectModel ทำหน้าที่แปลงโครงสร้างข้อมูลที่ได้รับการสกัดแล้วให้อยู่ในรูปแบบ object model ซึ่งจะ (1) รับอินพุต “MDL URI (รหัสประจำเอกสาร MDL)” จาก iGrid Client API และ Metadata Schema ที่ได้รับการสกัดแล้วจากโมดูล Metadata Schema Extract ที่อยู่ใน Extraction Layter (2) แปลง Application Metadata ให้อยู่ในรูปของ object model โดยเรียกใช้ JAXB และ (3) บีบอัด object model ให้อยู่ในรูปของ jar file และ (4) จัดส่ง jar file ให้ iGrid Client API

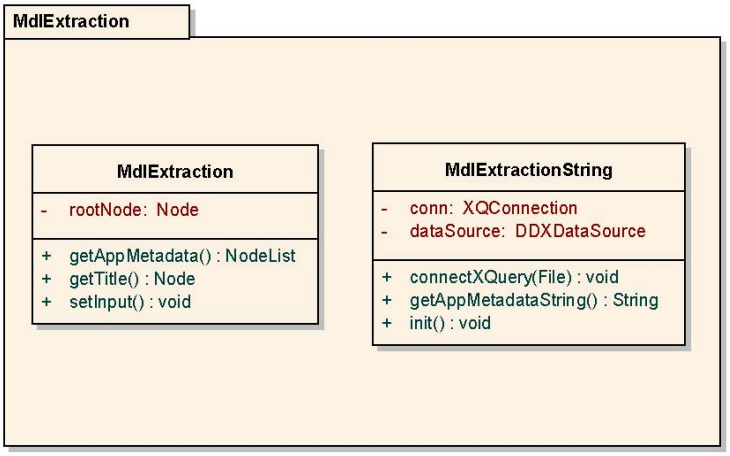
3.2 Package Diagram และ Class Diagram ของ MDL Analyzer



รูปที่ 3-2 Package diagram ของ MDL Analyzer

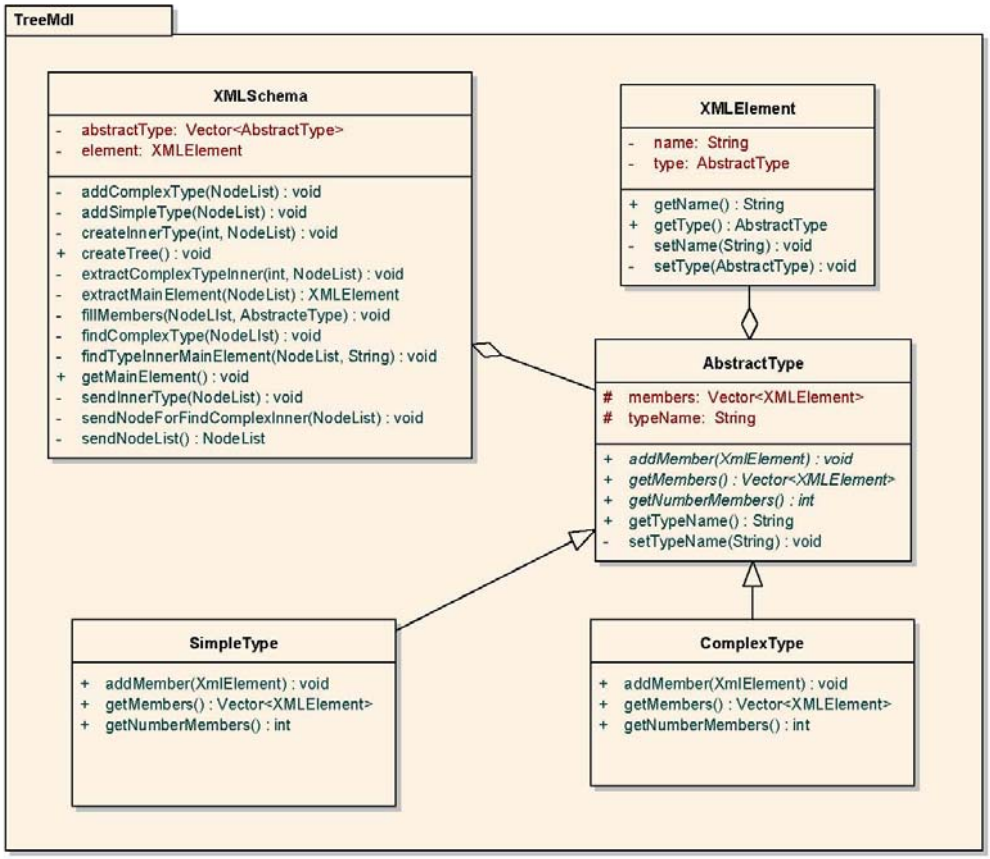
รูปที่ 3-2 แสดง Package diagram ของ MDL Analyzer ซึ่งประกอบด้วย 4 packages: MdlExtraction, TreeMdl , ObjectMdl และ AdapterPattern **(JAXB เป็น Package จากภายนอก)**

1. MdlExtraction Package ทำหน้าที่สกัด Application Metadata ซึ่งประกอบด้วย 2 classes หลักๆดังนี้
   - MdlExtraction (ดูรายละเอียดในภาคผนวก ก)
   - MdlExtractionString (ดูรายละเอียดในภาคผนวก ก)



รูปที่ 3.3 Class diagram ของ Package MdlExtraction

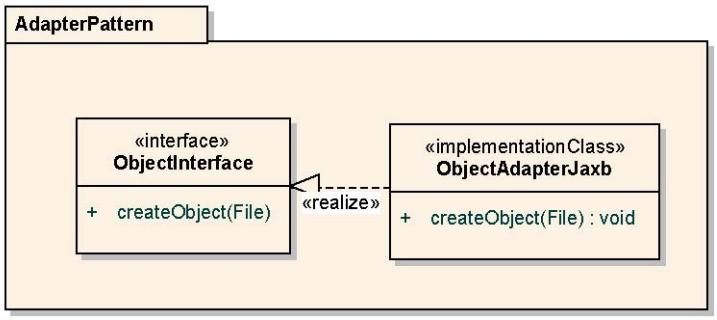
2. TreeMdl Package ทำหน้าที่ในการแปลงโครงสร้างข้อมูลที่อยู่ในรูปแบบของ XML schema ให้อยู่ในรูปแบบของโครงสร้างต้นไม้ ซึ่งประกอบด้วย 5 classes ดังนี้
   - XMLSchema (ดูรายละเอียดในภาคผนวก ก)
   - XMLElement (ดูรายละเอียดในภาคผนวก ก)
   - AbstractType (ดูรายละเอียดในภาคผนวก ก)
   - SimpleType (ดูรายละเอียดในภาคผนวก ก)
   - ComplexType (ดูรายละเอียดในภาคผนวก ก)



รูปที่ 3.4 Class diagram ของ Package TreeMdl

3. AdapterPattern Package ทำหน้าที่เป็นตัวกลางในการเชื่อมระหว่างเครื่องมือที่ใช้แปลงโครงสร้าง XML ให้เป็น Object กับ ObjectMdl package ในที่นี้ได้นำ JAXB มาใช้เป็นเครื่องมือในการแปลงโครงสร้างข้อมูลในรูปแบบของ XML schema ให้เป็น object model ในอนาคตสามารถเพิ่มเครื่องมืออื่นๆเข้ามาทดแทน JAXB ได้ โดยไม่กระทบกับการทำงานของ ObjectMdl package ทั้งนี้ AdapterPattern package ประกอบด้วย 2 classes ดังนี้

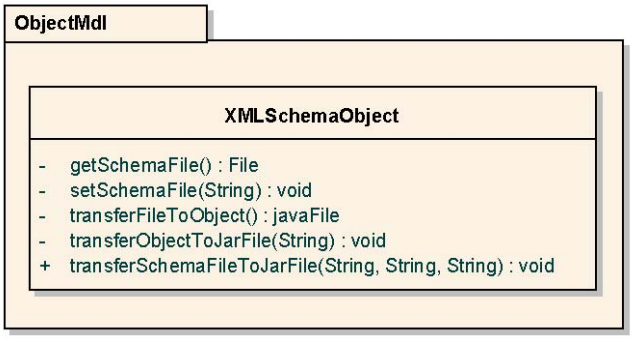
- ObjectInterface (ดูรายละเอียดในภาคผนวก ก)
- ObjectAdapterJaxb (ดูรายละเอียดในภาคผนวก ก)



รูปที่ 3.6 Class diagram ของ Package AdapterPattern

4. ObjectMdl Package ทำหน้าที่ในการบริหารจัดการการแปลงโครงสร้างข้อมูลที่อยู่ในรูปแบบของ XML schema ให้อยู่ในรูปของ object model และบีบอัด object model นี้ให้อยู่ในรูปของ Jar File ซึ่งประกอบด้วย 1 class ดังนี้

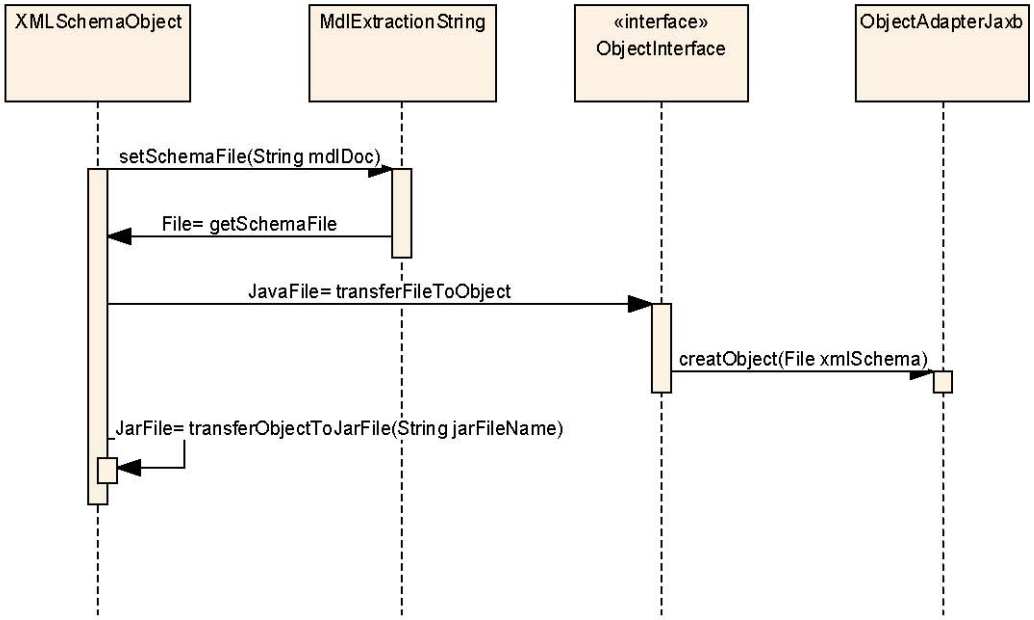
- XMLSchmaObject (ดูรายละเอียดในภาคผนวก ก)



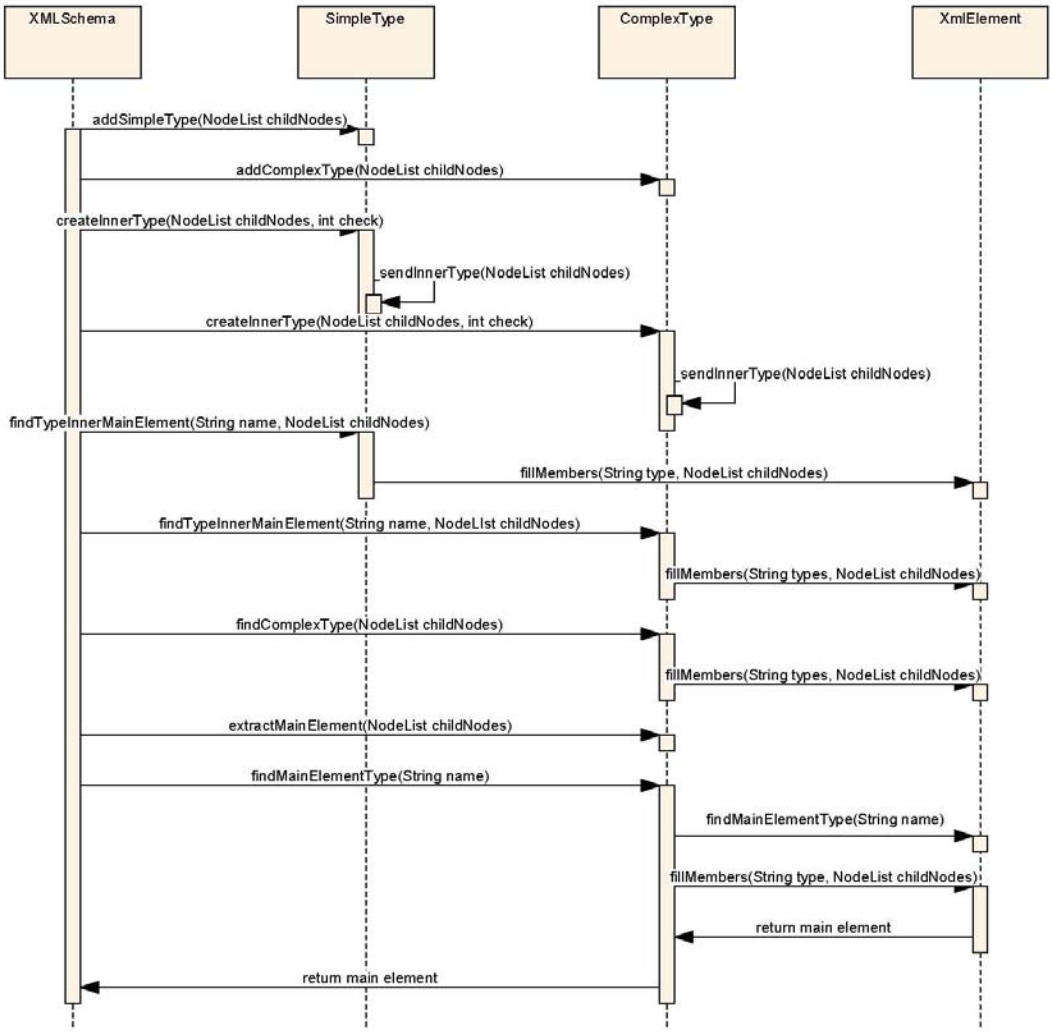
รูปที่ 3.5 Class diagram ของ Package ObjectMdl

3.4 Sequence Diagram

รูปที่ 3-6 และ 3-7 แสดง sequence diagram สำหรับ user case “transform metadata schema to file” และ use case “transform metadata schema to object model” ตามลำดับ



รูปที่ 3-6 Sequence Diagram สำหรับ use case “transform metadata schema to tree”



รูปที่ 3-7 Sequence Diagram สำหรับ use case "transform metadata schema to object model"

4 รูปแบบของ input และ output ของ MDL Analyzer

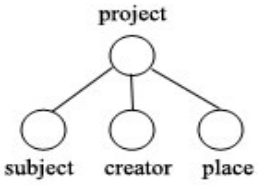
โครงสร้างข้อมูล ที่ถูกสร้างขึ้นบนพื้นฐานของ XML schema นั้นสามารถเขียนได้ใน 4 ลักษณะ ดังนี้ (1) แบบลำดับชั้นโดยไม่มีการอ้างอิงถึงชนิดของเอลิเมนต์ (2) แบบลำดับชั้นที่มีการอ้างอิงถึงชนิดของเอลิเมนต์ที่อยู่ภายใต้ XML schema เดียวกัน (3) แบบลำดับชั้นที่มีการอ้างอิงถึงชนิดของเอลิเมนต์ที่อยู่ข้าม XML schema และ (4) แบบลำดับชั้นที่มีการผสมผสานของแบบที่ 1-3 ทั้งนี้ MDL Analyzer จะต้องสามารถทำการสกัด และแปลงโครงสร้างข้อมูลที่อยู่ในรูปแบบดังกล่าว ให้เป็นโครงสร้างต้นไม้ หรือ object model ที่สอดคล้องกับโครงสร้างข้อมูลนั้นๆ

4.1 ตัวอย่างที่ 1

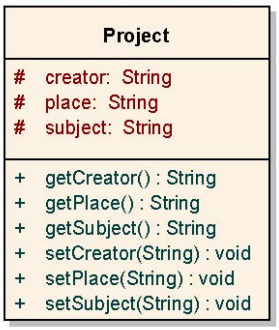
รูปที่ 4-1 แสดงตัวอย่างของเอกสาร MDL Project ที่แสดงโครงสร้างข้อมูลโครงงานวิจัย (project) บนพื้นฐานของ XML schema แบบ 1 ลำดับชั้นโดยไม่มีการอ้างอิงถึงชนิดของเอลิเมนต์ ทั้งนี้เมื่อ MDL Analyzer ทำการแปลงโครงสร้างข้อมูลในลักษณะนี้ จะได้ผลดังรูปที่ 4-2 และ 4-3

```
<?xml version="1.0" encoding="UTF-8"?>
<mdl xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
……..
  <applicationMetadata>
      <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
            attributeFormDefault="unqualified">
            <xs:element name="project">
              <xs:annotation>
                <xs:documentation>Comment describing your root element</xs:documentation>
              </xs:annotation>
              <xs:complexType>
                <xs:sequence>
                    <xs:element name="subject" type="xs:string"/>
                    <xs:element name="creator" type="xs:string"/>
                    <xs:element name="place" type="xs:string"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
      </xs:schema>
  </applicationMetadata>
</mdl>
```

รูปที่ 4-1 เอกสาร MDL Project บนพื้นฐานของ XML Schema แบบ 1 ลำดับชั้นโดยไม่มีการอ้างอิงถึงชนิดของเอลิเมนต์



รูปที่ 4-2 Tree Object สำหรับเอกสาร MDL Project ในรูปที่ 4-1



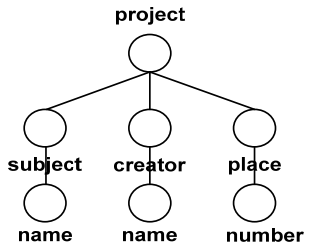
รูปที่ 4-3 Object Model สำหรับเอกสาร MDL Project ในรูปที่ 4-1

4.2 ตัวอย่างที่ 2

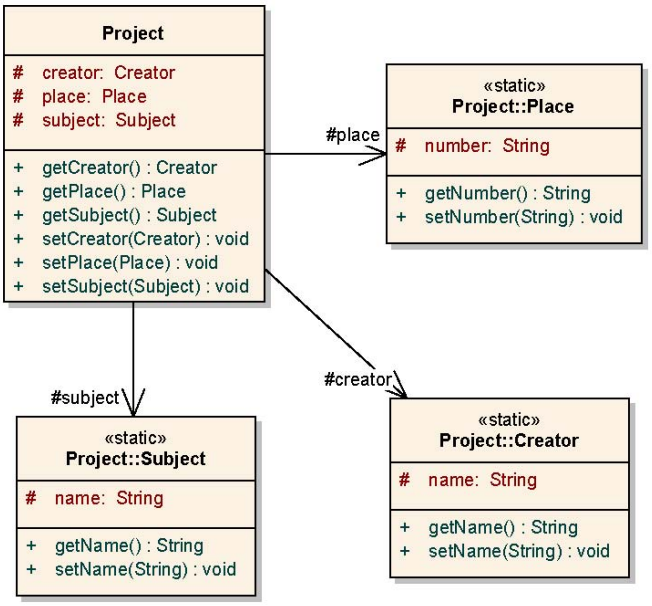
รูปที่ 4-4 แสดงตัวอย่างของเอกสาร MDL Project ที่แสดงโครงสร้างข้อมูลโครงงานวิจัย (project) บนพื้นฐานของ XML schema แบบ 2 ลำดับชั้นโดยไม่มีการอ้างอิงถึงชนิดของเอลิเมนต์ ทั้งนี้เมื่อ MDL Analyzer ทำการแปลงโครงสร้างข้อมูลในลักษณะนี้ จะได้ผลดังรูปที่ 4-5 และ 4-6

```
<?xml version="1.0" encoding="UTF-8"?>
<mdl xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  ……..
  <applicationMetadata>
    <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
      attributeFormDefault="unqualified">
      <xs:element name="project">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="subject">
                <xs:complexType>
                  <xs:sequence>
                      <xs:element name="name" type="xs:string"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="creator">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="name" type="xs:string"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
             <xs:element name="place">
                <xs:complexType>
                  <xs:annotation>
                    <xs:documentation>the details of the where  live or office </xs:documentation>
                  </xs:annotation>
                  <xs:sequence>
                    <xs:element name="number" type="xs:string"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          <xs:complexType>
        </xs:element>     </xs:schema> </applicationMetadata>
</mdl>
```

รูปที่ 4-4 เอกสาร MDL Project บนพื้นฐานของ XML Schema แบบ 2 ลำดับชั้นโดยไม่มีการอ้างอิงถึงชนิดของเอลิเมนต์



รูปที่ 4-5Tree Object สำหรับเอกสาร MDL Project ในรูปที่ 4-4



รูปที่ 4-6 Object Model สำหรับเอกสาร MDL Project ในรูปที่ 4-4

4.3 ตัวอย่างที่ 3

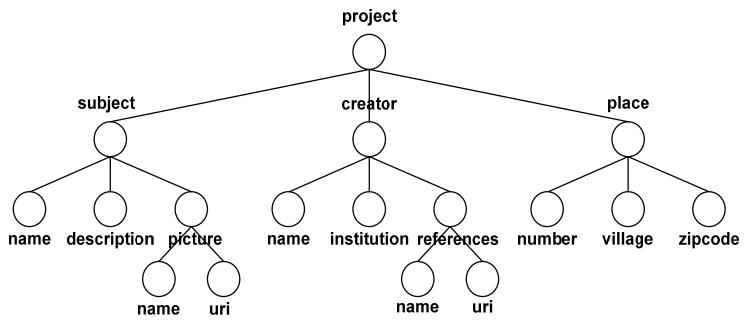
รูปที่ 4-7 แสดงตัวอย่างของเอกสาร MDL Project ที่แสดงโครงสร้างข้อมูลโครงงานวิจัย (project) บนพื้นฐานของ XML schema แบบ 3 ลำดับชั้นโดยไม่มีการอ้างอิงถึงชนิดของเอลิเมนต์ ทั้งนี้เมื่อ MDL Analyzer ทำการแปลงโครงสร้างข้อมูลในลักษณะนี้ จะได้ผลดังรูปที่ 4-8 และ 4-9

```
<?xml version="1.0" encoding="UTF-8"?>
<mdl xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  ……..
  <applicationMetadata>
    <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
        attributeFormDefault="unqualified">
        <xs:element name="project">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="subject">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="name" type="xs:string"/>
                  <xs:element name="description" type="xs:string"/>
                  <xs:element name="picture">
                  <xs:complexType>
                    <xs:sequence>
                      <xs:element name="name" type="xs:string"/>
                      <xs:element name="uri" type="xs:string"/>
                    </xs:sequence>
                  </xs:complexType>
                  </xs:element>
                </xs:sequence>
              </xs:complexType>
              </xs:element>
```

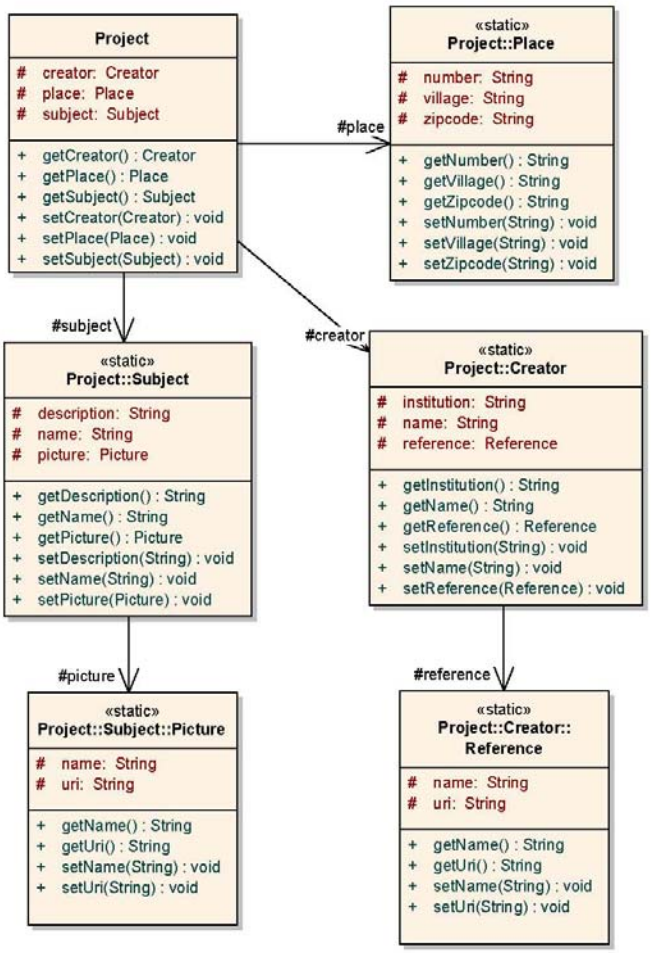
รูปที่ 4-7 เอกสาร MDL Project บนพื้นฐานของ XML Schema แบบ 3 ลำดับชั้นโดยไม่มีการอ้างอิงถึงชนิดของเอลิเมนต์

```
            <xs:element name="creator">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="name" type="xs:string"/>
                <xs:element name="institution" type="xs:string"/>
                <xs:element name="references">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="name" type="xs:string"/>
                    <xs:element name="uri" type="xs:string"/>
                  </xs:sequence>
                </xs:complexType>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
            </xs:element>
            <xs:element name="place">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="number" type="xs:string"/>
                <xs:element name="village" type="xs:string"/>
                <xs:element name="zipcode" type="xs:string"/>
              </xs:sequence>
            </xs:complexType>
            </xs:element>
           </xs:sequence>
          <xs:complexType>
        </xs:element>
      </xs:schema>
  </applicationMetadata>
```

รูปที่ 4-7 เอกสาร MDL Project บนพื้นฐานของ XML Schema แบบ 3 ลำดับชั้นโดยไม่มีการอ้างอิงถึงชนิดของเอลิเมนต์ (ต่อ)



รูปที่ 4-8 Tree Object สำหรับเอกสาร MDL Project ในรูปที่ 4-7



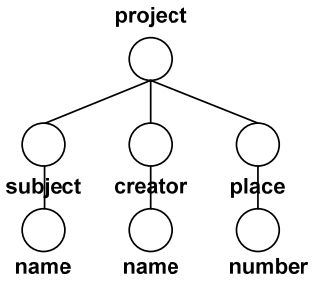
รูปที่ 4-9 Object Model สำหรับเอกสาร MDL Project ในรูปที่ 4-7

4.4 ตัวอย่างที่ 4

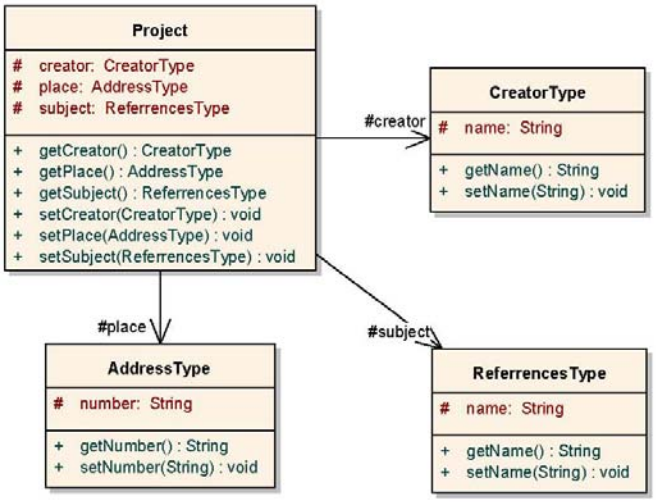
รูปที่ 4-10 แสดงตัวอย่างของเอกสาร MDL Project ที่แสดงโครงสร้างข้อมูลโครงงานวิจัย (project) บนพื้นฐานของ XML schema แบบ 2 ลำดับชั้นโดยมีการอ้างอิงถึงชนิดของเอลิเมนต์ ทั้งนี้เมื่อ MDL Analyzer ทำการแปลงโครงสร้างข้อมูลในลักษณะนี้ จะได้ผลดังรูปที่ 4-11 และ 4-12

```
<?xml version="1.0" encoding="UTF-8"?>
<mdl xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <applicationMetadata>
    <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
     attributeFormDefault="unqualified">
     <xs:complexType name="addressType">
        <xs:annotation>
          <xs:documentation>the details of the place  where  live or office </xs:documentation>
        </xs:annotation>
        <xs:sequence>
          <xs:element name="number" type="xs:string"/>
        </xs:sequence>
     </xs:complexType>
     <xs:complexType name=" referrencesType ">
        <xs:sequence>
          <xs:element name="name" type="xs:string"/>
        </xs:sequence>
     </xs:complexType>
     <xs:element name="project">
        <xs:annotation>
          <xs:documentation> Comment describing your root element </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="subject" type="referrencesType"/>
            <xs:element name="creator" type="creatorType"/>
            <xs:element name="place" type="addressType"/>
          </xs:sequence>
        </xs:complexType>
     </xs:element>
     <xs:complexType name=" creatorType ">
        <xs:sequence>
          <xs:element name="name" type="xs:string"/>
        </xs:sequence>
     </xs:complexType>
    </xs:schema>
  </applicationMetadata>
</mdl>
```

รูปที่ 4-10 เอกสาร MDL Project บนพื้นฐานของ XML Schema แบบ 2 ลำดับชั้นโดยมีการอ้างอิงถึงชนิดของเอลิเมนต์



รูปที่ 4-11 Tree Object สำหรับเอกสาร MDL Project ในรูปที่ 4-10



รูปที่ 4-12 Object Model สำหรับเอกสาร MDL Project ในรูปที่ 4-10

4.5 ตัวอย่างที่ 5

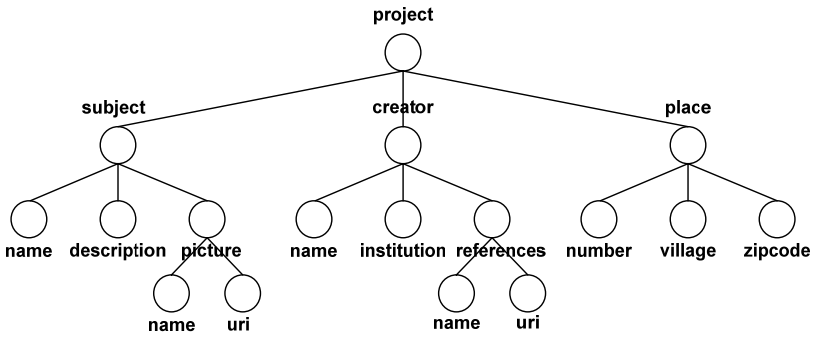
รูปที่ 4-13 แสดงตัวอย่างของเอกสาร MDL Project ที่แสดงโครงสร้างข้อมูลโครงงานวิจัย (project) บนพื้นฐานของ XML schema แบบ 3 ลำดับชั้นโดยมีการอ้างอิงถึงชนิดของเอลิเมนต์ ทั้งนี้เมื่อ MDL Analyzer ทำการแปลงโครงสร้างข้อมูลในลักษณะนี้ จะได้ผลดังรูปที่ 4-14 (ยังไม่สามารถทำการแปลงเป็น object model ได้)

```
<?xml version="1.0" encoding="UTF-8"?>
<mdl xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <applicationMetadata>
    <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
    attributeFormDefault="unqualified">
      <xs:complexType name="addressType">
        <xs:annotation>
          <xs:documentation>the details of the place  where  live or office </xs:documentation>
        </xs:annotation>
        <xs:sequence>
          <xs:element name="number" type="xs:string"/>
          <xs:element name="village" type="xs:string"/>
          <xs:element name="zipcode" type="zipcodeType"/>
        </xs:sequence>
      </xs:complexType>
      <xs: simpleType name=" zipcodeType">
        <xs:restriction base="xs:int">
          <xs:minInclusive value="5"/>
          <xs:maxInclusive value="5"/>
        </xs:restriction>
      </xs: simpleType >
      <xs:complexType name=" referrencesType ">
        <xs:sequence>
          <xs:element name="name" type="xs:string"/>
          <xs:element name="description" type="xs:string" minOccurs="0"/>
          <xs:element name="picture" type="zipcodeType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
      <xs:complexType name=" creatorType ">
        <xs:sequence>
          <xs:element name="name" type="xs:string"/>
          <xs:element name=" institution " type="xs:string" minOccurs="0"/>
          <xs:element name="references" type="fileType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
```

รูปที่ 4-13 เอกสาร MDL Project บนพื้นฐานของ XML Schema แบบ 3 ลำดับชั้นโดยมีการอ้างอิงถึงชนิดของเอลิเมนต์

```
        <xs:complexType name=" fileType ">
          <xs:sequence>
            <xs:element name="name" type="xs:string"/>
            <xs:element name="uri" type="xs:string"/>
            <xs:complexType>
              <xs:simpleContent>
               <xs:extension base="xs:anyURI">
                <xs:attribute name="type" type="xs:NMTOKEN">
                <xs:simpleType>
                  <xs:restriction base="xs:NMTOKEN">
                    <xs:enumeration value="file"/>
                    <xs:enumeration value="url"/>
                  </xs:restriction>
                </xs:simpleType>
                </xs:attribute>
              </xs:extension>
              </xs:simpleContent>
            </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
        <xs:element name="project">
          <xs:annotation>
            <xs:documentation>Comment describing your root element</xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:sequence>
              <xs:element name="subject" type="referrencesType"/>
              <xs:element name="creator" type="creatorType"/>
              <xs:element name="place" type="addressType"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:schema>
    </applicationMetadata>
  </mdl>
```

รูปที่ 4-13 เอกสาร MDL Project บนพื้นฐานของ XML Schema แบบ 3 ลำดับชั้นโดยมีการอ้างอิงถึงชนิดของเอลิเมนต์ (ต่อ)



รูปที่ 4-14 Tree Object สำหรับเอกสาร MDL Project ในรูปที่ 4-13

5 การใช้งาน MDL Analyzer

- เมื่อรันโปรแกรมด้วยตัว GisTool ต้องให้ input file อยู่ที่ C:\MdlDoc
- เมื่อรันโปรแกรมด้วย Object ต้องให้ input file อยู่ที่เดียวกับไดเร็คทอรี่ของ Mdl Analyzer
- Batch File ในการแปลง Object ถูกเก็บไว้ที่ C:\temp
- Jar File ถูกเก็บไว้ที่ C:\temp1
- หลังจากรันโปรแกรมด้วย Object ต้องลบไดเร็คทอรี่ c:\temp1\generated ทุกครั้ง

- รองรับโครงสร้าง XML 2 ลักษณะเท่านั้น คือ แบบลำดับชั้น และแบบอ้างอิงภายใน
- ในการสร้าง Object ไม่รองรับโครงสร้าง XML ที่เป็นแบบอ้างอิงภายใน ที่มีการอ้างอิง 2 ลำดับชั้นขึ้นไป

(3) แบบลำดับชั้นที่มีการอ้างอิงถึงชนิดของเอลิเมนต์ที่อยู่ข้าม XML schema และ

6 สรุป

Mdl Analyzer เป็นต้นแบบระดับห้องปฏิบัติการ ซึ่งมีหน้าที่ในการสกัดและแปลงโครงสร้างข้อมูลที่อยู่ในเอกสาร MDL ใดๆ ให้อยู่ในรูปแบบของ Tree หรือ Object Model และนำไปใช้งานร่วมกับ Information Grid Web Application, Information Grid Client API และ Generic Information Service Tool เพื่อให้การใช้งาน แอพลิเคชั่น หรือเครื่องมือเหล่านี้ มีความคล่องตัวขึ้น (ไม่ยึดติดกับชุดเอกสาร MDL หนึ่งๆ) ทั้งนี้ Mdl Analyzer นี้ได้รับการพัฒนาขึ้นโดยภาษาจาวา และได้รับการทดสอบกับเอกสาร MDL ที่สร้างขึ้นมาเองภายในหน่วยปฏิบัติการฯ ซึ่งในปัจจุบันสามารถใช้งานได้กับโครงสร้างข้อมูลที่อยู่ในรูปแบบของ XML schema ใน 2 ลักษณะ คือ แบบลำดับชั้นโดยไม่มีการอิงถึงชนิดของเอลิเมนต์ และแบบลำดับชั้นที่มีการอ้างอิงถึงชนิดของเอลิเมนต์ที่อยู่ภายใต้ XML schema เดียวกัน โดยมีรายละเอียดความสามารถในการทำงานดังแสดงในตารางที่ 6-1 กล่าวคือ MDL Analyzer (1) สามารถทำการ แปลงโครงสร้างข้อมูลใดๆใน 2 ลักษณะนี้ ให้อยู่ในรูป Tree (2) สามารถ ทำการแปลงโครงสร้างข้อมูลที่ไม่มีการ อ้างอิงชนิดของเอลิเมนต์ที่อยู่ภายใต้ XML schema เดียวกัน ให้อยู่ในรูป Object Model และ (3) สามารถทำการ แปลงโครงสร้างข้อมูลที่มีการอ้างอิงชนิดของเอลิเมนต์ที่อยู่ภายใต้ XML schema เดียวกัน ให้อยู่ในรูป Object Model โดยโครงสร้างข้อมูลนั้น จะต้องมีเพียง 1 ลำดับชั้น เท่านั้น ทั้งนี้ มีสาเหตุมาจากความสามารถของ JAXB ซึ่งเป็นเครื่องมือที่ใช้ในการแปลงโครงสร้างข้อมูลในรูปแบบ XML schema ให้เป็น Object Model

โครงสร้างข้อมูล	**Tree**	**Object Model**
แบบลำดับชั้นโดยไม่มีการอิงถึงชนิดของเอลิเมนต์	√	√
แบบลำดับชั้นโดยมีการอิงถึงชนิดของเอลิเมนต์ ที่อยู่ภายใต้ XML schema	√	-

ตารางที่ 6-1 ความสามารถของ MDL Analyzer

ทั้งนี้ MDL Analyzer นี้จะต้องได้รับการปรับปรุงต่อไป เพื่อให้สามารถรองรับกับโครงสร้างข้อมูล ในลักษณะต่างๆ และที่มีจำนวนลำดับชั้น.ใดๆ

ค. การเลือกใช้เครื่องมือในการแปลงโครงสร้าง XML ให้เป็น Object

ค1. JAXB

JAXB [] สามารถแปลงโครงสร้าง XML ให้เป็น Object และสามารถแปลงกลับจาก Object ให้เป็นโครงสร้าง XML ได้ เมื่อนำไปใช้งานร่วมกับ Mdl Analyzer และ Information Grid Client ไลบรารี่ที่ใช้มีจำนวนน้อยทำให้ไม่เปลืองเนื้อที่ในการจัดเก็บ และไม่ยุ่งยากในการแอดไลบรารี่เข้าไปยัง Mdl Analyzer และ Information Grid Client ส่วนข้อเสียคือรองรับลักษณะของ Object ได้ไม่เต็ม 100 เปอร์เซ็นต์ เห็นได้จากการทดสอบ คือ ไม่มี Collection Type และไม่สามารถสร้าง Object กับโครงสร้าง XML ที่เป็นแบบอ้างอิงภายใน ที่มีการอ้างอิง 2 ลำดับชั้นขึ้นไป

ค2. Globus Toolkit

ภาคผนวก ก

ก.1 Class MdlExtraction

java.lang.Object

└ **MdlExtraction.MdlExtraction**

---

public class **MdlExtraction**

extends java.lang.Object

Constructor Summary
MdlExtraction()

Method Summary	
static org.w3c.dom.NodeList	**getAppMetaData**() ใช้ DOM เข้าถึง tag ที่ชื่อว่า applicationMetadata จากนั้น คืนค่าโหนดทั้งหมดที่อยู่ใน element นี้ออกมาเป็น ModeList
static org.w3c.dom.Node	**getTitle**() ใช้ DOM เข้าถึง tag ที่ชื่อว่า title แล้วคืนค่าโหนดที่เก็บ text ออกมา
static void	**setInput**() ทำหน้าที่รับไฟล์ XML document ข้ามาโดยใช้ DOM ในการอ่านไฟล์
static void	**setInput**(java.lang.String fileName) ทำหน้าที่รับไฟล์ XML document ข้ามาโดยใช้ DOM ในการอ่านไฟล์

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

MdlExtraction

public **MdlExtraction**()

Method Detail

setInput

public static void **setInput**(java.lang.String fileName)

ทำหน้าที่รับไฟล์ XML document ข้ามาโดยใช้ DOM ในการอ่านไฟล์

Parameters:

fileName - ชื่อเอกสาร MDL

---

setInput

public static void **setInput**()

ทำหน้าที่รับไฟล์ XML document ข้ามาโดยใช้ DOM ในการอ่านไฟล์

---

getTitle

public static org.w3c.dom.Node **getTitle**()

throws javax.xml.parsers.ParserConfigurationException,

org.xml.sax.SAXException,

java.io.IOException

ใช้ DOM เข้าถึง tag ที่ชื่อว่า title แล้วคืนค่าโหนดที่เก็บ text ออกมา

Returns:

คืนค่า Node ที่เป็น title ในรูปแบบของ text

Throws:

javax.xml.parsers.ParserConfigurationException

org.xml.sax.SAXException

java.io.IOException

---

getAppMetaData

public static org.w3c.dom.NodeList **getAppMetaData**()

ใช้ DOM เข้าถึง tag ที่ชื่อว่า applicationMetadata จากนั้น คืนค่าโหนดทั้งหมดที่อยู่ใน element นี้ออกมาเป็น ModeList

Returns:

applicationMetadata ในรูปแบบของ NodeList

ก.2 Class MdlExtractionString

java.lang.Object

└ **MdlExtraction.MdlExtractionString**

---

public class **MdlExtractionString**

extends java.lang.Object

Constructor Summary
MdlExtractionString(java.lang.String filename)

Method Summary	
void	**connectXQuery**(java.lang.String xmlFilename) ทำหน้าที่สร้างการติดต่อกับ XQuery และใช้กับไฟล์ XML ที่ระบุไว้ในพารามิเตอร์
java.lang.String	**getApplicationMetadataString**() ทำหน้าที่เข้าถึง tag ที่ชื่อว่า applicationMetadata จากนั้น คืนค่าโหนดทั้งหมดที่อยู่ใน element นี้ ออกมาเป็น String
void	**init**() ทำหน้าที่สร้างการติดต่อกับ XQuery
java.lang.String	**query**(java.lang.String queryString) ทำหน้าที่การสืบค้น tag ตามที่ระบุไว้ในพารามิเตอร์

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

MdlExtractionString

public **MdlExtractionString**(java.lang.String filename)

Method Detail

init

public void **init**()

throws com.ddtek.xquery3.XQException

ทำหน้าที่สร้างการติดต่อกับ XQuery

Throws:

com.ddtek.xquery3.XQException

---

query

public java.lang.String **query**(java.lang.String queryString)

throws com.ddtek.xquery3.XQException

ทำหน้าที่การสืบค้น tag ตามที่ระบุไว้ในพารามิเตอร์

Parameters:

queryString - เป็นเงื่อนไขตาม FLOWER ที่ใช้ในการสืบค้น XML

Returns:

คืนค่าตามเงื่อนไขในการสืบค้น อยู่ในรูปแบบของ String

Throws:

com.ddtek.xquery3.XQException

---

connectXQuery

public void **connectXQuery**(java.lang.String xmlFilename)

ทำหน้าที่สร้างการติดต่อกับ XQuery และใช้กับไฟล์ XML ที่ระบุไว้ในพารามิเตอร์

Parameters:

xmlFilename - เป็นชื่อของเอกสาร MDL

---

getApplicationMetadataString

public java.lang.String **getApplicationMetadataString**()

ทำหน้าที่เข้าถึง tag ที่ชื่อว่า applicationMetadata จากนั้น คืนค่าโหนดทั้งหมดที่อยู่ใน element นี้ ออกมาเป็น String

Returns:

คืนค่า Application Metadata มาให้ อยู่ในรูปแบบของ String

ก.3 Class AbstractType

java.lang.Object

└ **TreeMdl.AbstractType**

Direct Known Subclasses:

ComplexType, SimpleType

---

public abstract class **AbstractType**

extends java.lang.Object

Constructor Summary
AbstractType()

Method Summary	
abstract void	**addMember**(XMLElement member) Implement ที่ class ComplexType, SimpleType
abstract java.util.Vector<XMLElement>	**getMembers**() Implement ที่ class ComplexType, SimpleType
abstract int	**getNumberMemebers**() Implement ที่ class ComplexType, SimpleType
java.lang.String	**getTypeName**() ให้ค่าเป็นชื่อ type ออกมา

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

AbstractType

public **AbstractType**()

Method Detail

getTypeName

public java.lang.String **getTypeName**()

ให้ค่าเป็นชื่อ type ออกมา

Returns:

คืนค่าชื่อ type ในรูปแบบของ String

---

addMember

public abstract void **addMember**(XMLElement member)

Implement ที่ class ComplexType, SimpleType

Parameters:

member - ชื่อสมาชิก

---

getMembers

public abstract java.util.Vector<XMLElement> **getMembers**()

Implement ที่ class ComplexType, SimpleType

Returns:

คืนค่าสมาชิกที่อยู่ในเวคเตอร์

---

getNumberMemebers

public abstract int **getNumberMemebers**()

Implement ที่ class ComplexType, SimpleType

Returns:

จำนวนสมาชิก

ก.4 Class TreeMdl

java.lang.Object

└ TreeMdl.AbstractType

  └ **TreeMdl.ComplexType**

---

public class **ComplexType**

extends AbstractType

Constructor Summary
ComplexType(java.lang.String typeName)

Method Summary	
void	**addMember**(XMLElement member) ใช้ add สมาชิก ของ element นั้นๆเข้าไป
java.util.Vector<XMLElement>	**getMembers**() คืนค่าที่เป็นสมาชิกของ element นั้นๆออกมา
int	**getNumberMemebers**() คืนค่าเป็นจำนวนสมาชิกของ element นั้นๆ

Methods inherited from class TreeMdl.AbstractType
getTypeName

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ComplexType

public **ComplexType**(java.lang.String typeName)

Method Detail

getMembers

public java.util.Vector<XMLElement> **getMembers**()

คืนค่าที่เป็นสมาชิกของ element นั้นๆออกมา

Specified by:

getMembers in class AbstractType

Returns:

คืนค่าสมาชิกที่อยู่ในเวคเตอร์

---

addMember

public void **addMember**(XMLElement member)

ใช้ add สมาชิก ของ element นั้นๆเข้าไป

Specified by:

addMember in class AbstractType

Parameters:

member - ชื่อสมาชิก

---

getNumberMemebers

public int **getNumberMemebers**()

คืนค่าเป็นจำนวนสมาชิกของ element นั้นๆ

Specified by:

getNumberMemebers in class AbstractType

Returns:

จำนวนสมาชิก

ก.5 Class SimpleType

java.lang.Object

└ TreeMdl.AbstractType

└ **TreeMdl.SimpleType**

---

public class **SimpleType**

extends AbstractType

Constructor Summary
SimpleType(java.lang.String typeName)

Method Summary	
void	**addMember**(XMLElement member) ไม่ได้ใช้ทำอะไรในคลาสนี้ เพียงแต่ override มาเฉยๆ เพราะ element ที่เป็น object ของ class SimpleType ไม่มีสมาชิก

java.util.Vector<XMLElement>	**getMembers**() ไม่ได้ใช้ทำอะไรในคลาสนี้ เพียงแต่ override มาเฉยๆ เพราะ element ที่เป็น object ของ class SimpleType ไม่มีสมาชิก
int	**getNumberMemebers**() คืนค่าจำนวนสมาชิกออกมา ซึ่ง ก็คือ 0 element นั้นเป็น object ของ class SimpleType

Methods inherited from class TreeMdl.AbstractType
getTypeName

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

SimpleType

public **SimpleType**(java.lang.String typeName)

Method Detail

addMember

public void **addMember**(XMLElement member)

ไม่ได้ใช้ทำอะไรในคลาสนี้ เพียงแต่ override มาเฉยๆ เพราะ element ที่เป็น object ของ class SimpleType ไม่มีสมาชิก

Specified by:

addMember in class AbstractType

Parameters:

member - ชื่อสมาชิก

---

getMembers

public java.util.Vector<XMLElement> **getMembers**()

ไม่ได้ใช้ทำอะไรในคลาสนี้ เพียงแต่ override มาเฉยๆ เพราะ element ที่เป็น object ของ class SimpleType ไม่มีสมาชิก

Specified by:

getMembers in class AbstractType

Returns:

คืนค่าสมาชิกที่อยู่ในเวคเตอร์

---

getNumberMemebers

public int **getNumberMemebers**()

คืนค่าจำนวนสมาชิกออกมา ซึ่ง ก็คือ 0 element นั้นเป็น object ของ class SimpleType

Specified by:

getNumberMemebers in class AbstractType

Returns:

จำนวนสมาชิก

ก.6 Class XMLElement

java.lang.Object

└ **TreeMdl.XMLElement**

---

public class **XMLElement**

extends java.lang.Object

Constructor Summary
XMLElement(java.lang.String name, AbstractType type)

Method Summary	
java.lang.String	**getName**() ใช้คืนค่าชื่อ element
AbstractType	**getType**() ใช้คืนค่าชื่อ type

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

XMLElement

public **XMLElement**(java.lang.String name,

AbstractType type)

Method Detail

getName

public java.lang.String **getName**()

ใช้คืนค่าชื่อ element

Returns:

คืนค่าชื่อ element ในรูปแบบ String

---

getType

public AbstractType **getType**()

ใช้คืนค่าชื่อ type

Returns:

คืนค่าชื่อ type

ก.7 Class XMLSchema

java.lang.Object

└ **TreeMdl.XMLSchema**

---

public class **XMLSchema**

extends java.lang.Object

Constructor Summary
XMLSchema()

Method Summary	
void	**createTree**() ใช้สำหรับสร้าง tree form
void	**createTree**(java.lang.String dbType) ใช้สำหรับสร้าง tree form
XMLElement	**getMainElement**()

	คืนค่าเป็น element หลัก ซึ่งก็คือ root

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

XMLSchema

public **XMLSchema**()

Method Detail

createTree

public void **createTree**(java.lang.String dbType)

ใช้สำหรับสร้าง tree form

Parameters:

dbType - ชื่อเอกสาร MDL

---

createTree

public void **createTree**()

ใช้สำหรับสร้าง tree form

---

getMainElement

public XMLElement **getMainElement**()

คืนค่าเป็น element หลัก ซึ่งก็คือ root

Returns:

คืนค่าเป็น element หลัก

ก.8 Class ObjectAdapterJaxb

java.lang.Object

└ **AdapterPattern.ObjectAdapterJaxb**

All Implemented Interfaces:

ObjectInterface

---

public class **ObjectAdapterJaxb**

extends java.lang.Object

implements ObjectInterface

Constructor Summary
ObjectAdapterJaxb()

Method Summary	
void	**createObject**(java.io.File xmlSchema) ทำหน้าที่สร้าง object ที่อยู่ในรูป java file โดยใช้ JAXB

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ObjectAdapterJaxb

public **ObjectAdapterJaxb**()

Method Detail

createObject

public void **createObject**(java.io.File xmlSchema)

ทำหน้าที่สร้าง object ที่อยู่ในรูป java file โดยใช้ JAXB

Specified by:

createObject in interface ObjectInterface

Parameters:

xmlSchema - ชื่อไฟล์ที่มีนามสกุลเป็น xsd

ก.9 Class XmlSchemaObject

java.lang.Object

└ **ObjectMdl.XmlSchemaObject**

public class **XmlSchemaObject**

extends java.lang.Object

Constructor Summary
XmlSchemaObject()

Method Summary	
void	**tranferSchemaFileToJarFile**(java.lang.String mdlDoc, java.lang.String xmlSchemaFileName, java.lang.String jarFileName) ทำหน้าที่แปลงเอกสาร Mdl ให้เป็น jar file
void	**transferFileToObject**(java.io.File xmlSchema) ทำหน้าที่แปลงไฟล์ที่ใช้เก็บ applicationMetadata ให้อยู่ในรูปของ object ที่เป็น java file

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

XmlSchemaObject

public **XmlSchemaObject**()

Method Detail

tranferSchemaFileToJarFile

public void **tranferSchemaFileToJarFile**(java.lang.String mdlDoc,
java.lang.String xmlSchemaFileName,
java.lang.String jarFileName)

ทำหน้าที่แปลงเอกสาร Mdl ให้เป็น jar file

Parameters:

mdlDoc - ชื่อเอกสาร MDL

xmlSchemaFileName - ชื่อไฟล์ที่มีนามสกุลเป็น xsd

jarFileName - ชื่อ Jar File

---

transferFileToObject

public void **transferFileToObject**(java.io.File xmlSchema)

ทำหน้าที่แปลงไฟล์ที่ใช้เก็บ applicationMetadata ให้อยู่ในรูปของ object ที่เป็น java file

Parameters:

xmlSchema - ชื่อไฟล์ที่มีนามสกุลเป็น xsd

ภาคผนวก ข การใช้งานของ MDL Analyzer ร่วมกับ GiSTool

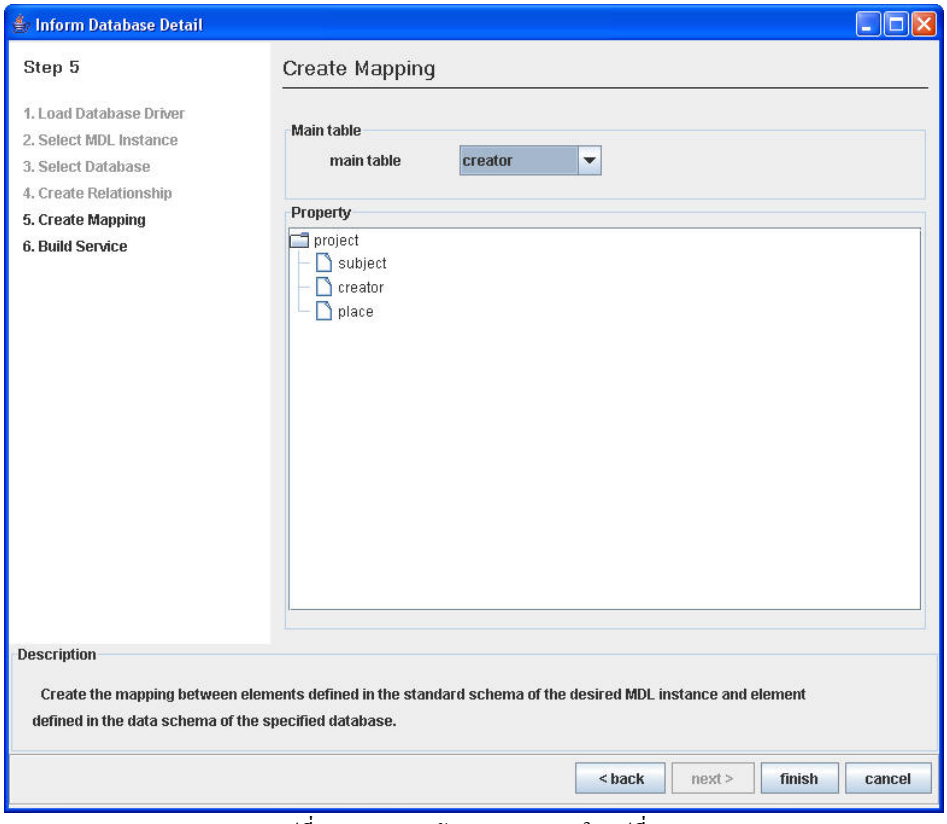
ในปัจจุบัน GiSTool 1.0 Beta ได้รับการพัฒนาขึ้นมาเพื่อใช้ในการสร้าง Information Service สำหรับแหล่งข้อมูลที่อยู่ในรูปของฐานข้อมูลใดๆ บนพื้นฐานของโครงสร้างข้อมูล 3 ประเภท คือ โครงสร้างข้อมูลโครงงานวิจัย (project) โครงสร้างข้อมูลบทความงานวิจัย (publication) และโครงสร้างข้อมูลนักวิจัย (researcher) และเพื่อให้ GiSTool สามารถทำงานร่วมกับโครงสร้างข้อมูลที่กำหนดไว้ในเอกสาร MDL ใดๆนั้น จึงจำเป็นต้องทำการปรับแก้ GiSTool เพื่อให้สามารถทำการติดต่อกับ MDL Analyzer ได้ โดยมีรายละเอียด ดังนี้

1. แก้ไข buildhibernatemapping\SelectDatabasePage.java (ซึ่งอยู่ใน <GiSTool-src>\GiSTool-build CreateHibernate\src) ดังนี้
   - เพิ่ม method “File[] showDir (File file)” เพื่ออ่านไฟล์ในไดเรกเทอรี่ที่กำหนดไว้ในพารามิเตอร์ file
   - ปรับแก้ method “void initComponents ()” เพื่อแสดงเอกสาร MDL (*.xml) ที่อยู่ใน ไดเรกเทอรี่ c:\MdlDoc
   - ปรับแก้ method “void nextButtonActionPerformed ()” เพื่อรองรับการเลือกเอกสาร MDL หนึ่งๆ

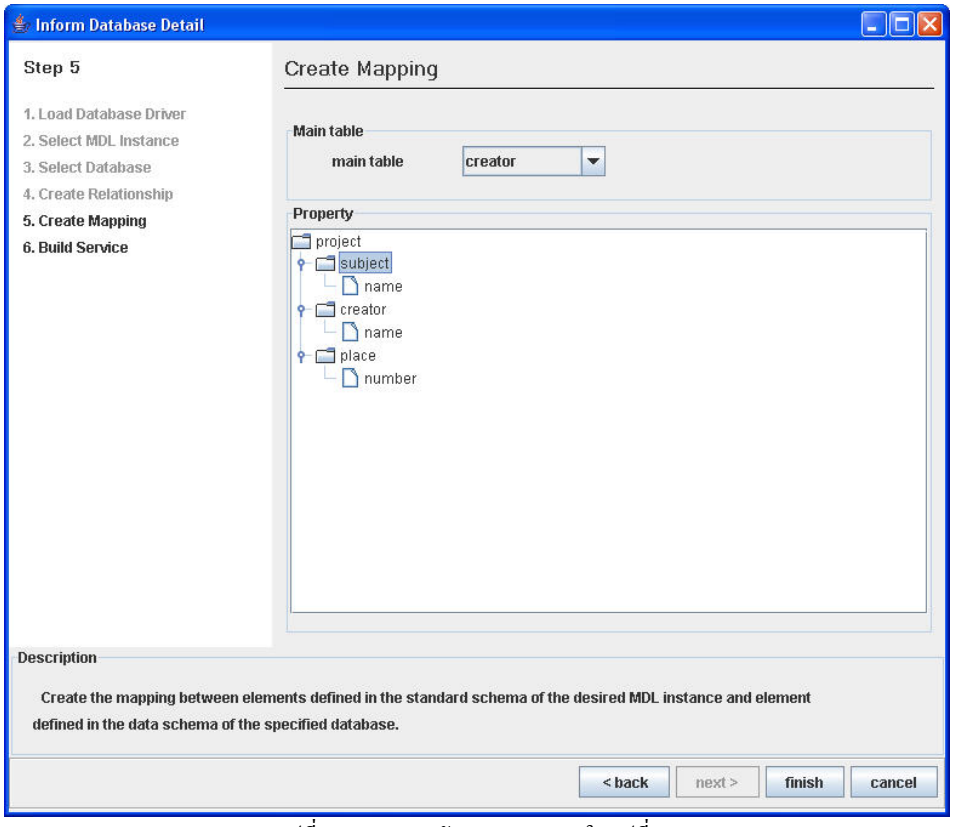
2. แก้ไข HibernateMappingModel\SampleResearchTree.java (ซึ่งอยู่ใน <GiSTool-src>\GiSTool-build CreateHibernate\src) ทั้งหมด เพื่อทำการติดต่อกับ MDL Analyzer และแสดงโครงสร้างข้อมูลในเอกสาร MDL ที่ได้เลือกไว้ให้อยู่ในรูปแบบของต้นไม้ ดังแสดงในรูปที่ ข-1 – ข-3

3. สำเนา <GiStool-src>\GiSTool-build CreateHibernate\bin* ไปยัง <GiSTool>\buildCreate Hibernate\classes

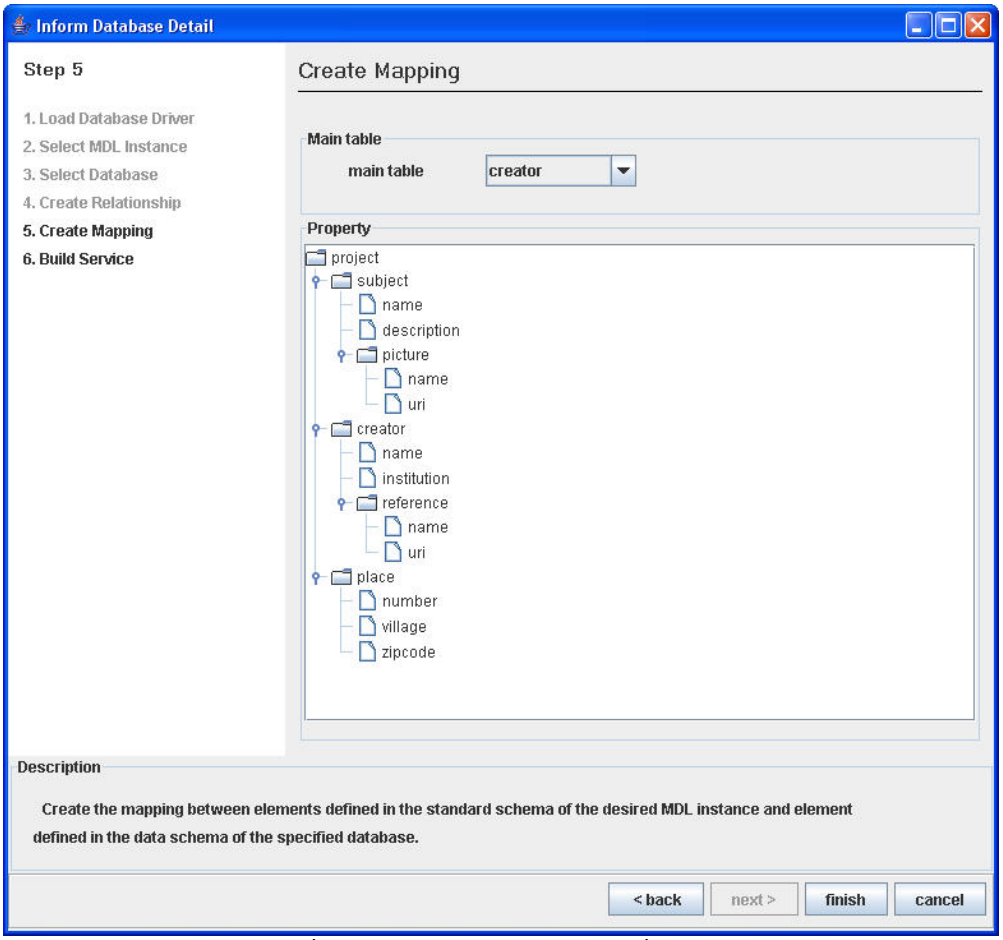
4. เรียกใช้งาน GiSTool ผ่านทาง <GiSTool>\GiSTool.jar



รูปที่ ข-1 GiSTool กับเอกสาร MDL ในรูปที่ …



รูปที่ ข-2 GiSTool กับเอกสาร MDL ในรูปที่ …



รูปที่ ข-3 GiSTool กับเอกสาร MDL ในรูปที่ …

เอกสารอ้างอิง

[1] http://www.informationgrid.org

[2] รายงานบันทึกวิจัย การออกแบบมาตรฐานโครงสร้างข้อมูลภายใต้ระบบกริดสารสนเทศ เวอร์ชั่น 1.0 (Marker Description Language (MDL) Specification Version 1.0), เจษฎา เพ็งสุวรรณ เสกสิทธิ์ สุวรรณ นัยนา สหเวชชภัณฑ์ ศรเทพ วรรณรัตน์

[3] รายงานบันทึกวิจัยกรอบการพัฒนา Information Source เวอร์ชั่น 1, เจษฎา เพ็งสุวรรณ เสกสิทธิ์ สุวรรณ นัยนา สหเวชชภัณฑ์ ศรเทพ วรรณรัตน์

[4] คู่มือการใช้งาน GiSTool เวอร์ชั่น 1.0, นัยนา สหเวชชภัณฑ์ คำรณ อรุณเรื่อ พิทักษ์ แท่นแก้ว