รายงานวิจัยฉบับสมบูรณ์

โครงการ การจัดการข้อมูลเอกสาร XML
(XML Document Management)

โดย ดร. ชุติพร อนุตริยะ และคณะ

มิถุนายน 2550

สัญญาเลขที่ MRG4780192

รายงานวิจัยฉบับสมบูรณ์

โครงการ การจัดการข้อมูลเอกสาร XML
(XML Document Management)

คณะผู้วิจัย	สังกัด
1. ผศ. ดร. ชุติพร อนุตริยะ	มหาวิทยาลัยชินวัตร
2. ศาสตราจารย์วิลาศ วูวงศ์	สถาบันเทคโนโลยีแห่งเอเชีย

สนับสนุนโดยสำนักงานคณะกรรมการการอุดมศึกษา และสำนักงานกองทุนสนับสนุนการวิจัย
(ความเห็นในรายงานนี้เป็นของผู้วิจัย สกอ. และ สกว. ไม่จำเป็นต้องเห็นด้วยเสมอไป)

กิตติกรรมประกาศ

คณะผู้วิจัยขอขอบคุณสำนักงานคณะกรรมการการอุดมศึกษา และสำนักงานกองทุนสนับสนุนการวิจัย ที่อนุมัติทุนวิจัยสำหรับโครงการวิจัยนี้ และขอขอบคุณมหาวิทยาลัยชินวัตรและสถาบันเทคโนโลยีแห่งเอเชียที่สนับสนุนการดำเนินโครงการวิจัยนี้เป็นอย่างดี

หัวหน้าโครงการวิจัยขอขอบคุณ ศาสตราจารย์ ดร. วิลาศ วูวงศ์ ที่ได้เป็นที่ปรึกษาโครงการวิจัย ตลอดจนให้ข้อคิดเห็นและข้อเสนอแนะที่เป็นประโยชน์ยิ่งต่อการวิจัย และช่วยให้งานวิจัยนี้สำเร็จได้ด้วยดี

บทคัดย่อ

รหัสโครงการ MRG4780192

ชื่อโครงการ การจัดการข้อมูลเอกสาร XML

ชื่อนักวิจัย ผศ. ดร. ชุติพร อนุตริย
ภาควิชาวิทยาการคอมพิวเตอร์
มหาวิทยาลัยชินวัตร

E-mail Address chutiporn@shinawatra.ac.th

ระยะเวลาโครงการ 3 ปี

ในปัจจุบันเนื่องจากภาษา XML ได้รับการยอมรับและถูกนำมาใช้งานอย่างกว้างขวาง จึงทำให้เกิดความต้องการในการพัฒนาวิธีการในการจัดการข้อมูลเอกสาร XML อย่างมีประสิทธิภาพ งานวิจัยนี้จึงมุ่งเน้นที่การศึกษาและวิเคราะห์ถึงความต้องการและความสัมพันธ์ระหว่าง การควบคุมและจัดการการเปลี่ยนแปลงรุ่นเอกสาร และการกำหนดสิทธิและการควบคุมการเข้าถึงข้อมูลเอกสารของกลุ่มผู้ใช้ระดับต่างๆ รวมถึงการวิเคราะห์ถึงข้อดีที่เป็นผลลัพธ์จากการรวมหน้าที่ทั้งสองดังกล่าวเข้าด้วยกัน จากการศึกษาดังกล่าวส่งผลให้งานวิจัยนี้นำเสนอและพัฒนาต้นแบบและวิธีการในการจัดภารเอกสาร XML ที่เหมาะสม โดยสามารถเชื่อมโยงหน้าที่การทำงานทั้งสองเข้าด้วยกันอย่างเป็นระบบ

นอกจากนี้ เมื่อพิจารณาถึงงานประยุกต์ที่สำคัญของ XML ในการแสดงความรู้ประเภทออนโทโลยี (ontology) ซึ่งเป็นวิธีบรรยายความรู้แบบหนึ่ง พบว่า ในปัจจุบันจำนวนเอกสารออนโทโลยีบนเว็บมีจำนวนมากขึ้นเป็นลำดับ ในขณะที่งานวิจัยในปัจจุบันที่นำเสนอต้นแบบในการจัดการเอกสาร XML เพื่อเก็บข้อมูลดังกล่าว ยังขาดความสามารถที่ช่วยให้ผู้ใช้สามารถค้นหาออนโทโลยีที่ตรงตามความต้องการได้อย่างถูกต้อง ดังนั้นเพื่อแก้ปัญหาดังกล่าว งานวิจัยนี้จึงพัฒนาต้นแบบ SQORE (Semantic Query based Ontology Retrieval) สำหรับจัดการเอกสาร XML สำหรับเก็บความรู้ประเภทออนโทโลยี โดยนำกลไกการอนุมานและการใช้เหตุผลมาใช้ เพื่อให้การสืบค้นมีประสิทธิภาพและมีความถูกต้องมากยิ่งขึ้น

คำหลัก การควบคุมการเข้าถึงข้อมูลเอกสาร XML, การจัดการการเปลี่ยนแปลงรุ่นเอกสาร XML, การจัดการเอกสาร XML, การจัดการเอกสารออนโทโลยี, การสืบค้นเอกสารออนโทโลยี

Abstract

Project Code : MRG4780192

Project Title : XML Document Management

Investigator : Assistant Professor Dr. Chutiporn Anutariya

E-mail Address : chutiporn@shinawatra.ac.th

Project Period : 3 years

Due to an increasing popularity of employing XML documents in various application domains, there arises a demand for an efficient XML document database management. Thus, development of effective mechanisms for manipulating access control and version control has become a major research area. However, well-developed access control and version control mechanisms alone do not suffice because they are closely interrelated. This research analyzes important features and requirements of both access control and version control mechanisms, and gives an insight into the problems when they are handled separately. Furthermore, to overcome these problems, an approach which integrates these two essential mechanisms into a single framework is proposed.

In addition, this research also considers an important application of XML technology to Web ontologies—a formal structure that explicitly represents semantics of information and provides a concise, uniform and declarative description. To date, the number of freely accessible Web ontologies grows, and thus promoting ontology reusability in various application domains. Therefore, a framework for Semantic Query based Ontology Retrieval, namely SQORE, is developed. It enables precise formulation of a semantic query in order to best capture a user' s ontology requirements, which include not only the desired class and property names, but also their relations and restrictions.

Keywords : Access Control, Version Control, Ontology Management, Ontology Retrieval, XML Document Management

Executive Summary

ภาษา XML (Extensible Markup Language) เป็นภาษากำกับข้อความที่ได้พัฒนาขึ้น เพื่อให้เป็นภาษามาตรฐานสำหรับการแสดง และการสื่อสารแลกเปลี่ยนข้อมูลและเอกสารแบบมีโครงสร้างระหว่างระบบงานประยุกต์ต่างๆ โดยในปัจจุบันภาษา XML ได้รับการยอมรับและถูกนำมาใช้งานอย่างกว้างขวาง โดยได้มีการกำหนดไวยากรณ์มาตรฐานของเอกสาร XML สำหรับแต่ละสาขางานประยุกต์ เพื่อเป็นมาตรฐานในการสื่อสารแลกเปลี่ยนข้อมูลระหว่างผู้ใช้จากต่างองค์กร ดังนั้น จำนวนเอกสาร XML ในองค์กรหนึ่งๆ รวมไปถึงจำนวนเอกสาร XML บนเว็บมีปริมาณเพิ่มมากขึ้นเป็นลำดับ จึงทำให้เกิดความต้องการในการพัฒนาวิธีการในการจัดการข้อมูลเอกสาร XML อย่างมีประสิทธิภาพ งานวิจัยนี้จึงมุ่งเน้นที่การศึกษาและวิเคราะห์ถึงความต้องการและความสัมพันธ์ระหว่างการควบคุมและจัดการการเปลี่ยนแปลงรุ่นเอกสาร และการกำหนดสิทธิและการควบคุมการเข้าถึงข้อมูลเอกสารของกลุ่มผู้ใช้ระดับต่างๆ รวมถึงการวิเคราะห์ถึงข้อดีที่เป็นผลลัพธ์จากการรวมหน้าที่ทั้งสองดังกล่าวเข้าด้วยกัน จากการศึกษาดังกล่าวพบว่า การทำงานในหน้าที่ทั้งสองดังกล่าวแยกกันอย่างเป็นอิสระ และไม่มีการเชื่อมโยงกันนั้น ทำให้ผู้ดูแลระบบต้องกำหนดนโยบายการควบคุมการเข้าถึงข้อมูลที่ซับซ้อนขึ้น เพื่อให้สามารถควบคุมเอกสารในรุ่นต่างๆ ที่แตกต่างกันได้อย่างครบถ้วน ซึ่งทำให้อาจเกิดความผิดพลาดในการกำหนดนโยบายดังกล่าวได้ และส่งผลกระทบถึงความปลอดภัยของข้อมูลได้อีกด้วย ดังนั้น งานวิจัยนี้จึงนำเสนอและพัฒนาต้นแบบและวิธีการในการจัดการเอกสาร XML ที่เหมาะสม มีประสิทธิภาพและความปลอดภัย สามารถเชื่อมโยงหน้าที่การทำงานทั้งสองเข้าด้วยกันอย่างเป็นระบบ โดยอนุญาตให้ผู้ดูแลระบบการจัดการข้อมูลเอกสาร XML สามารถกำหนดกฎความสัมพันธ์ของหน้าที่ทั้งสองดังกล่าว โดยเมื่อมีการเปลี่ยนแปลงข้อมูลในเอกสาร XML นั่นคือมีการเปลี่ยนแปลงรุ่นของเอกสารตามรูปแบบที่กำหนด จะนำไปสู่การเปลี่ยนแปลงสิทธิในการเข้าถึงข้อมูลสำหรับแต่ละกลุ่มผู้ใช้ได้ เช่น เปลี่ยนสถานะจากข้อมูลที่เปิดเผยต่อผู้ใช้บางกลุ่มได้ไปเป็นข้อมูลที่เป็นความลับ และจำกัดกลุ่มผู้ใช้ที่เข้าถึงข้อมูลได้

นอกจากนี้ เมื่อพิจารณาถึงงานประยุกต์ที่สำคัญของ XML ในการแสดงความรู้ประเภทออนโทโลยี (ontology) ซึ่งเป็นวิธีบรรยายความรู้แบบหนึ่ง พบว่า ในปัจจุบันจำนวนเอกสารออนโทโลยีบนเว็บมีจำนวนมากขึ้นเป็นลำดับ ในขณะที่งานวิจัยในปัจจุบันที่นำเสนอต้นแบบในการจัดการเอกสาร XML เพื่อเก็บข้อมูลดังกล่าว ยังขาดความสามารถที่ช่วยให้ผู้ใช้สามารถค้นหาออนโทโลยีที่ตรงตามความต้องการได้อย่างถูกต้อง โดยระบบสืบค้นออนโทโลยีที่มีอยู่ เพียงแค่สนับสนุนการใช้คำค้นสำหรับการค้นหา โดยไม่อนุญาตให้ผู้ใช้อธิบายโครงสร้าง ชื่อคุณสมบัติ หรือรายละเอียดอื่นๆ ที่ถูกกำหนดไว้ในออนโทโลยีที่ต้องการได้ ส่งผลให้ผลลัพธ์จากการค้นหามีจำนวนมากแต่ไม่ตรงตาม

ความต้องการของผู้ใช้ ดังนั้นเพื่อแก้ปัญหาดังกล่าว งานวิจัยนี้จึงได้พัฒนาต้นแบบ SQORE (Semantic Query based Ontology Retrieval) สำหรับจัดการเอกสาร XML สำหรับเก็บความรู้ประเภทออนโทโลยี โดยอนุญาตให้ผู้ใช้กำหนดคำสืบค้นที่มีความหมาย ซึ่งอธิบายรายละเอียดของออนโทโลยีที่ต้องการมายังระบบ จากนั้นจึงนำกลไกการอนุมานและการใช้เหตุผลมาใช้ เพื่อให้การสืบค้นมีประสิทธิภาพและมีความถูกต้องมากยิ่งขึ้น นอกจากนี้ SQORE ยังได้นำฐานข้อมูลคำศัพท์และความสัมพันธ์ระหว่างคำศัพท์มาใช้ เพื่อให้สามารถค้นหาความสัมพันธ์ระหว่างคำสืบค้นและคำในออนโทโลยีได้ถูกต้องมากขึ้น ซึ่งในกรณีที่คำทั้งสองไม่ใช่คำเดียวกัน นั่นหมายความว่ามีความเป็นไปได้ 4 แบบ คือ คำทั้งสองนั้นมีความหมายเหมือนกัน คำที่อยู่ในคำสืบค้นมีความหมายกว้างกว่า คำที่อยู่ในคำสืบค้นมีความหมายแคบกว่า หรือ ไม่สามารถหาความสัมพันธ์ของคำทั้งสองได้ จากกลไกการทำงานดังที่กล่าวนี้ ทำให้ผู้ใช้สามารถระบุความต้องการออนโทโลยีได้อย่างครบถ้วนและทำให้การสืบค้นมีความถูกต้อง สามารถค้นหาออนโทโลยีที่มีความใกล้เคียงกับคำสืบค้นที่มีความหมาย และส่งคืนกลับให้ผู้ใช้เป็นคำตอบได้

สารบัญ

สารบัญรูปภาพ

1 บทนำ

1.1 ความสำคัญและที่มาของปัญหาที่ทำการวิจัย

ภาษา XML (Extensible Markup Language) [12] เป็นภาษากำกับข้อความที่ได้พัฒนาขึ้นในปี พ.ศ. 2541 และได้รับการสนับสนุนจากองค์กร World Wide Web Consortium (W3C) ซึ่งเป็นองค์กรที่ทำหน้าที่เผยแพร่และให้ข้อมูลทางด้านเทคนิคและวิชาการที่เกี่ยวกับเว็บ เพื่อให้เป็นภาษามาตรฐานสำหรับการแสดงและการสื่อสารแลกเปลี่ยนข้อมูลและเอกสารแบบมีโครงสร้างระหว่างระบบงานประยุกต์ต่างๆ โดยที่ภาษา XML อนุญาตให้ผู้ใช้กำหนดคำสำหรับกำกับข้อความ (tag) รวมทั้งโครงสร้าง และไวยากรณ์ของเอกสารสำหรับแต่ละระบบงานได้เอง เพื่อให้เอกสารสามารถสื่อความหมายได้ชัดเจนขึ้น และสามารถนำไปประมวลผลโดยเครื่องคอมพิวเตอร์ได้โดยตรง โดยไม่จำเป็นต้องมีการแปลหรือแปลงรูปไปอยู่ในรูปแบบภาษาอื่น

จากคุณสมบัติดังที่ได้กล่าวมาแล้ว ปัจจุบันภาษา XML ได้รับการยอมรับและถูกนำมาใช้งานอย่างกว้างขวาง โดยได้มีการกำหนดไวยากรณ์มาตรฐานของเอกสาร XML สำหรับแต่ละสาขางานประยุกต์ เพื่อเป็นมาตรฐานในการสื่อสารแลกเปลี่ยนข้อมูลระหว่างผู้ใช้จากต่างองค์กร เช่น ภาษา ebXML (Electronic Business eXtensible Markup Language) [20] ซึ่งเป็นมาตรฐานในการแลกเปลี่ยนข้อมูลและบริการระหว่างผู้ค้าในระบบพาณิชย์อิเล็คทรอนิกส์, ภาษา WSDL (Web Service Description Language) [11] สำหรับการอธิบายคุณลักษณะของบริการต่างๆ ในระบบเว็บเซอร์วิส, ภาษา RDF (Resource Description Framework) [23] สำหรับการอธิบายข้อมูลและทรัพยากรบนเว็บที่มีความหมาย (Semantic Web) และในห้องสมุดดิจิตอล (Digital Library) เพื่อให้การแสดงและการค้นหาข้อมูลมีประสิทธิภาพมากขึ้น รวมไปถึงการกำหนดงานประยุกต์ XML อื่นๆ สำหรับใช้ในระบบ E-Government

จำนวนเอกสาร XML ในองค์กรหนึ่งๆ รวมไปถึงจำนวนเอกสาร XML บนเว็บมีปริมาณเพิ่มมากขึ้นเป็นลำดับ อย่างไรก็ตามเนื่องจาก XML เป็นภาษาแบบกึ่งโครงสร้าง (Semi-Structured Data) ที่มีโครงสร้างแบบต้นไม้ (tree) อีกทั้งยังอนุญาตให้ผู้ใช้กำหนดการเชื่อมโยงและความสัมพันธ์ระหว่างข้อมูลในเอกสารเดียวกันหรือต่างเอกสารกันได้ ด้วยเหตุนี้ทฤษฎีการจัดการฐานข้อมูลเชิงสัมพันธ์ (Relational Database Theory) ซึ่งเป็นที่แพร่หลายในปัจจุบันจึงไม่สามารถนำมาประยุกต์ใช้ได้โดยตรงทั้งหมด จำเป็นต้องมีการพัฒนาเทคนิคและวิธีการเฉพาะสำหรับการจัดการเอกสารในภาษา XML

นอกเหนือจากความสามารถในการจัดเก็บข้อมูลเอกสาร XML, การสืบค้น การควบคุมความถูกต้องของไวยากรณ์และความถูกต้องตรงกันของข้อมูล ทฤษฎีและวิธีการที่พัฒนาขึ้นใหม่จำเป็นต้องมีความสามารถที่สำคัญ 2 ประการ ดังนี้

(1) การควบคุมและจัดการการเปลี่ยนแปลงรุ่นเอกสาร (Version Control) ซึ่งมีหน้าที่ที่สำคัญในการควบคุมและดูแลรุ่นของเอกสารในระบบที่มีการเปลี่ยนแปลงอยู่เสมอ โดยต้องสนับสนุนการย้อนกลับไปดูเนื้อหาของเอกสารรุ่นก่อนหน้า รวมทั้งต้องรองรับการเปรียบเทียบรุ่นของเอกสารและอธิบายถึงความแตกต่างระหว่างรุ่นทั้งสอง นั่นคือจำเป็นต้องเก็บประวัติการเปลี่ยนแปลงรุ่นของเอกสารทั้งหมดอย่างถูกต้อง และสนับสนุนการค้นหาการเปลี่ยนแปลงต่างๆ ดังกล่าว เช่น ค้นหาความเปลี่ยนแปลงทั้งหมดที่ผู้ใช้ ก กระทำต่อเอกสารรุ่น 1.0 ถึงรุ่น 3.5 หรือค้นหาว่าใครเป็นผู้ลบข้อมูลในเอกสารรุ่น 2.0

(2) การกำหนดสิทธิและการควบคุมการเข้าถึงข้อมูลเอกสาร XML ของกลุ่มผู้ใช้ระดับต่างๆ (Access Control) ทั้งนี้ผู้ใช้แต่ละกลุ่มอาจมีสิทธิที่แตกต่างกัน เช่น กลุ่ม ก สามารถอ่าน/แก้ไขข้อมูลในเอกสารได้ทั้งหมด ในขณะที่กลุ่ม ข สามารถอ่านข้อมูลได้เพียงบางส่วนและไม่สามารถแก้ไขข้อมูลได้ นอกจากนี้สิทธิในการเข้าถึงข้อมูลยังอาจมีความขัดแย้งกันได้ เช่น มีการระบุว่าผู้ใช้ ค สามารถแก้ไขและไม่สามารถแก้ไขข้อมูลได้ ในกรณีนี้จำเป็นต้องมีการกำหนดนโยบายการแก้ไขความขัดแย้ง (Conflict Resolution Policy) เช่น อาจจะกำหนดว่าในกรณีที่มีความขัดแย้งเกิดขึ้น ให้ถือว่าผู้ใช้ไม่มีสิทธิในการเข้าถึงข้อมูลนั้นๆ

เมื่อพิจารณาโดยละเอียด จะเห็นได้ว่าหน้าที่ (function) ในการควบคุมการเปลี่ยนแปลงรุ่นเอกสารและการเข้าถึงข้อมูลในเอกสารต่างมีความสัมพันธ์สอดคล้องซึ่งกันและกัน โดยที่การเปลี่ยนแปลงรุ่นของเอกสารสามารถนำไปสู่การเปลี่ยนแปลงสิทธิในการเข้าถึงข้อมูลสำหรับแต่ละกลุ่มผู้ใช้ได้ ทั้งนี้รูปแบบความสัมพันธ์ของทั้งสองหน้าที่ดังกล่าวอาจแตกต่างกันไปขึ้นอยู่กับสาขางานประยุกต์ ตัวอย่างเช่น ในระบบงานโรงพยาบาล โดยทั่วไปผู้ใช้กลุ่มพยาบาลสามารถอ่านข้อมูลการวินิจฉัยโรคของผู้ป่วยได้ แต่ภายหลังเมื่อแพทย์พบว่าผู้ป่วยหนึ่งๆ ป่วยเป็นโรคร้ายแรง ผู้ใช้กลุ่มพยาบาลจะไม่มีสิทธิในการเข้าถึงข้อมูลการวินิจฉัยโรคของผู้ป่วยดังกล่าวโดยอัตโนมัติ ตัวอย่างนี้ชี้ให้เห็นว่า การเปลี่ยนแปลงข้อมูลในเอกสาร XML มีส่วนสัมพันธ์ทำให้ข้อมูลเปลี่ยนสถานะจากข้อมูลที่เปิดเผยต่อผู้ใช้บางกลุ่มได้ ไปเป็นข้อมูลที่เป็นความลับ และมีการจำกัดกลุ่มผู้ใช้ที่เข้าถึงข้อมูลได้

จากความสัมพันธ์ดังกล่าวทำให้เกิดความต้องการในการรวมหน้าที่ทั้งสองหน้าที่นี้เข้าด้วยกัน เพื่อให้การจัดการเอกสาร XML มีประสิทธิภาพ มีความปลอดภัย และง่ายต่อการประยุกต์ใช้มากยิ่งขึ้น อย่างไรก็ดีในปัจจุบันยังไม่มีงานวิจัยใดที่เสนอการรวมหน้าที่ทั้งสองดังกล่าวเข้าด้วยกัน โดยงานวิจัยส่วนใหญ่ต่างเสนอเทคนิคและวิธีการในการจัดการหน้าที่ใดหน้าที่หนึ่งเพียงอย่างเดียว อีกทั้งยังมีการประยุกต์ใช้ภาษาและเทคโนโลยีที่แตกต่างกันไป ก่อให้เกิดความยุ่งยากซับซ้อนในการแลกเปลี่ยนและการรวมข้อมูลเอกสาร XML จากหลายๆ หน่วยงานที่มีสิทธิการเข้าถึงข้อมูลของกลุ่มผู้ใช้ที่แตกต่างกัน ซึ่งจะมีผลกระทบโยงไปถึงความปลอดภัยของข้อมูลได้อีกด้วย

นอกจากนี้ เมื่อพิจารณาถึงงานประยุกต์ของ XML ในการแสดงความรู้ประเภทออนโทโลยี (ontology) เช่น RDF (Resource Description Framework) [23] และ OWL (Web Ontology Language) [5] พบว่างานวิจัยในปัจจุบันที่นำเสนอต้นแบบในการจัดการเอกสาร XML เพื่อเก็บข้อมูลดังกล่าว ยังขาดความสามารถที่ช่วยให้ผู้ใช้สามารถค้นหาออนโทโลยีที่ตรงตามความต้องการได้อย่างถูกต้อง โดยนำกลไกการอนุมานและการใช้เหตุผลมาใช้ ดังนั้นจึงเกิดความต้องการในการพัฒนาระบบต้นแบบที่อนุญาตให้ผู้ใช้สามารถสืบค้นออนโทโลยีที่ต้องการได้อย่างมีประสิทธิภาพ

1.2 วัตถุประสงค์ของโครงการ

(1) ศึกษาและวิเคราะห์ถึงหน้าที่ความต้องการ และความสัมพันธ์ระหว่าง
- การควบคุมและจัดการการเปลี่ยนแปลงรุ่นของเอกสาร XML (version control of XML documents)
- การกำหนดสิทธิและการควบคุมการเข้าถึงข้อมูลเอกสาร XML (access control of XML documents)

รวมถึงการวิเคราะห์ข้อดีที่เป็นผลลัพธ์จากการรวม (integrate) หน้าที่ทั้งสองเข้าด้วยกัน

(2) พัฒนาต้นแบบ (model) และวิธีการในการจัดการเอกสาร XML ที่เหมาะสม มีประสิทธิภาพ สามารถรองรับการทำงานและความต้องการที่ได้วิเคราะห์ในวัตถุประสงค์ข้อที่ 1 ได้อย่างครบถ้วน อีกทั้งยังสามารถเชื่อมโยงหน้าที่การทำงานทั้งสองเข้าด้วยกันได้อย่างเป็นเอกภาพ

(3) นำต้นแบบและวิธีการที่เสนอไปประยุกต์ใช้กับงานประยุกต์ของ XML (XML applications) เพื่อเป็นกรณีศึกษา และสาธิตให้เห็นถึงความสำคัญ ศักยภาพและความเป็นไปได้ในการนำทฤษฎีนี้ไปใช้งานจริง

(4) พัฒนาต้นแบบ (model) และวิธีการในการจัดการเอกสาร XML ที่ใช้จัดเก็บความรู้ประเภทออนโทโลยี

1.2.1 โครงสร้างรายงาน

รายงานวิจัยฉบับสมบูรณ์นี้ประกอบด้วย

บทที่ 2 นำเสนอผลงานวิจัยที่เกี่ยวข้อง

บทที่ 3 วิเคราะห์ความต้องการและความสัมพันธ์ระหว่างการควบคุมการเข้าถึงข้อมูล และการเปลี่ยนแปลงรุ่นของเอกสาร XML รวมถึงศึกษาข้อดีจากการรวมหน้าที่ทั้งสองเข้าด้วยกัน

บทที่ 4 พัฒนาระบบควบคุมการเข้าถึงข้อมูล และการเปลี่ยนแปลงรุ่นของเอกสาร XML

บทที่ 5 พัฒนาระบบการจัดการเอกสาร XML สำหรับเก็บความรู้ประเภทออนโทโลยี

บทที่ 6 สรุปผลการวิจัยและนำเสนอทิศทางงานวิจัยในอนาคต

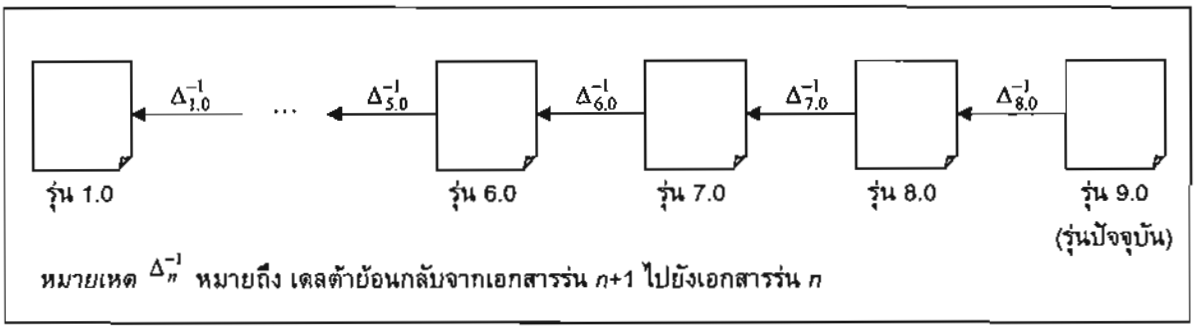
2 ผลงานวิจัยที่เกี่ยวข้อง

2.1 การจัดการการเปลี่ยนแปลงรุ่นของเอกสาร (Version Control)

ระบบการจัดการการเปลี่ยนแปลงรุ่นของเอกสาร มีหน้าที่ที่สำคัญในการควบคุมและดูแลรุ่นของเอกสารในระบบที่มีการเปลี่ยนแปลงอยู่เสมอ โดยระบบดังกล่าวต้องมีความสามารถในการย้อนกลับไปดูเนื้อหาของเอกสารรุ่นก่อนหน้า รวมถึงสามารถเปรียบเทียบรุ่นของเอกสารและอธิบายถึงความแตกต่างระหว่างรุ่นทั้งสอง นั่นคือระบบการจัดการการเปลี่ยนแปลงรุ่นของเอกสารจำเป็นต้องเก็บประวัติการเปลี่ยนแปลงรุ่นของเอกสารทั้งหมดอย่างถูกต้อง และสนับสนุนการค้นหาการเปลี่ยนแปลงต่างๆ ดังกล่าว เช่น ค้นหาความเปลี่ยนแปลงทั้งหมดที่ผู้ใช้ n กระทำต่อเอกสารรุ่น 1.0 ถึงรุ่น 3.5 หรือค้นหาว่าใครเป็นผู้ลบข้อมูลในเอกสารรุ่น 2.0
งานวิจัยที่สำคัญที่เสนอเทคนิคและวิธีการสำหรับการควบคุมการเปลี่ยนแปลงรุ่นของเอกสารสามารถแบ่งได้เป็น 3 กลุ่มหลักๆ ดังนี้

2.1.1 กลุ่ม Text-Based หรือกลุ่มที่ใช้ข้อความตัวหนังสือเป็นหลัก

วิธีการแบบ text-based เป็นวิธีการในยุคแรกๆ ที่นำเสนอการจัดการการเปลี่ยนแปลงรุ่นของเอกสาร วิธีการที่สำคัญในกลุ่มนี้ได้แก่ RCS [27] และ SCCS (Source Code Control System) [29] โดยวิธีแบบ RCS มองเอกสารเป็นเพียงลำดับของบรรทัดข้อความตัวหนังสือ และใช้เดลตา (delta) ในการแสดงความแตกต่างของรุ่น (version differences) วิธีการนี้จัดเก็บรุ่นปัจจุบัน(รุ่นล่าสุด)ไว้ในระบบเพียงรุ่นเดียว โดยรุ่นของเอกสารอื่นๆ ถูกจัดเก็บในรูปของเดลตาย้อนกลับ (reverse delta) ซึ่งอธิบายถึงการย้อนกลับไปยังรุ่นของเอกสารก่อนหน้า ดังนั้นการค้นหารุ่นของเอกสารที่ไม่ใช่รุ่นปัจจุบันจำเป็นต้องนำเดลต้าย้อนกลับมาประมวลผลกับเอกสารรุ่นปัจจุบันอย่างต่อเนื่อง เพื่อให้ได้รุ่นของเอกสารที่ต้องการ ดังรูปที่ 1 ส่วนวิธีการแบบ SCCS นั้นจะแทรกข้อมูลการเปลี่ยนแปลงรุ่นของเอกสารในแต่ละช่วงข้อมูลเข้าไปในเอกสารรุ่นแรกสุด โดยระบุช่วงเวลา (timestamp) ที่การเปลี่ยนแปลงดังกล่าวมีความถูกต้อง (valid) เอกสารที่มีการแทรกข้อมูลดังกล่าวจะเรียกว่า ไฟล์ SCCS ดังนั้นการสร้างเอกสารรุ่นหนึ่งๆ จะทำได้โดยอ่านไฟล์ SCCS แล้วเลือกช่วงข้อมูลที่อยู่ในช่วงเวลาที่ต้องการ



รูปที่ 2.1: วิธีการจัดการการเปลี่ยนแปลงรุ่นของเอกสารแบบ RCS

แม้ว่าวิธีการในกลุ่ม text-based นี้สามารถนำมาประยุกต์ใช้กับข้อมูลเอกสาร XML ได้ เนื่องจากเอกสาร XML จัดว่าเป็นเอกสารที่ประกอบด้วยข้อความตัวหนังสือ แต่ทว่าวิธีการดังกล่าวต่างมีข้อจำกัดเมื่อนำมาใช้กับเอกสาร XML เพราะจะทำให้เกิดการบันทึกความเปลี่ยนแปลงของเอกสารมากเกินความจำเป็น เนื่องจากเอกสาร XML 2 เอกสารอาจมีข้อความตัวหนังสือที่แตกต่างกัน แต่ไม่ถือว่ามีความแตกต่างกันหรือแตกต่างอย่างไม่มีนัยสำคัญ ตัวอย่าง

- ลำดับของแอททริบิวต์ (attributes) ของ XML element หนึ่งๆ สามารถปรากฏในลำดับใดก็ได้ เช่น <Person gender="male" age="30"/> และ <Person age="30" gender="male"/> ถือว่าไม่มีความแตกต่างกัน
- การกำหนด namespace prefix ของเอกสารอาจมีความแตกต่างกันได้

จากปัญหาดังที่ได้ยกตัวอย่างมา จึงสรุปได้ว่าวิธีการในกลุ่ม text-based ไม่เหมาะสมกับการนำมาใช้กับเอกสาร XML

2.1.2 กลุ่ม Content-Based หรือกลุ่มที่ใช้เนื้อหาของเอกสารเป็นหลัก

วิธีการในกลุ่มนี้มุ่งเน้นในการแก้ปัญหาที่พบในกลุ่มวิธีการแบบ text-based และบันทึกความเปลี่ยนแปลงของเอกสารโดยพิจารณาที่เนื้อหาเป็นหลัก ตัวอย่างของวิธีการที่สำคัญในกลุ่มนี้ ได้แก่ UBCC (Usefulness Based Copy Control) [13] ซึ่งได้ปรับปรุงวิธีการ RCS และได้เสนอกระบวนการจัดเก็บข้อมูลการเปลี่ยนแปลงแบบใหม่ โดยวิธีการ UBCC นี้สามารถสร้างรุ่นของเอกสารที่ต้องการได้อย่างมีประสิทธิภาพโดยใช้พื้นที่ในการจัดเก็บข้อมูลที่น้อยลง จากคุณสมบัตินี้เองที่ทำให้วิธีการ UBCC นี้มีศักยภาพในการสร้างรุ่นทั้งหมดของเอกสารในระบบได้อย่างมีประสิทธิภาพ อย่างไรก็ตาม เนื่องจากวิธีการ UBCC มีพื้นฐานมาจากวิธีการ RCS ดังนั้นข้อจำกัดของ RCS ในการจัดการโครงสร้างที่ซับซ้อนของเอกสาร XML ก็ยังไม่สามารถแก้ไขได้

วิธีการที่สำคัญแบบ content-based อีกวิธีการหนึ่ง ได้แก่ วิธีที่เสนอโดย Lam และ Wong [24] ซึ่งเก็บเอกสารที่เป็นรุ่นล่าสุดอย่างสมบูรณ์ และบันทึกความเปลี่ยนแปลงในรูปของเดลตาย้อนกลับเช่นเดียวกับวิธี RCS และ UBCC ข้อแตกต่างที่สำคัญก็คือ วิธีการนี้จะเก็บเอกสารรุ่นระหว่างกลางไว้ด้วย เพื่อให้ระบบสามารถสร้างเอกสารรุ่นต่างๆ ที่ต้องการได้สะดวกรวดเร็วยิ่งขึ้น โดยอนุญาตให้สร้างเอกสารบางรุ่นจากเอกสารรุ่นระหว่างกลาง แทนที่จะต้องสร้างเอกสารนั้นจากรุ่นปัจจุบัน

2.1.3 กลุ่ม Reference-Based หรือกลุ่มที่ใช้การอ้างอิงระหว่างวัตถุเป็นหลัก

ในขณะที่กลุ่มวิธีแบบ content-based เน้นที่การแสดงการเปลี่ยนแปลงข้อมูลในเอกสาร กลุ่มวิธีแบบ reference-based เน้นที่การแสดงส่วนของเอกสารที่ไม่มีการเปลี่ยนแปลง นั่นคือ ส่วนของเอกสารที่เหมือนกันระหว่างสองรุ่นที่ต่อเนื่องกัน วิธีการที่สำคัญในกลุ่มนี้ คือ วิธี RBVM (Reference-Based Version Model) [14] ซึ่งมีความสามารถในการคงโครงสร้างของเอกสารที่เปลี่ยนแปลงไปโดยใช้การอ้างอิงระหว่างวัตถุ (object references) ทำให้สามารถรองรับการสืบค้นข้อมูลเกี่ยวกับการเปลี่ยนแปลงรุ่นของเอกสารได้อย่างมีประสิทธิภาพ รวมไปถึงการสนับสนุนการสร้างเอกสาร XML รุ่นต่างๆ

อย่างไรก็ตามงานวิจัยที่ผ่านมาเร็วๆ นี้ พบว่าความสามารถที่มีประสิทธิภาพในการคงโครงสร้างของเอกสารเพียงอย่างเดียวนั้นไม่เพียงพอในการจัดการเปลี่ยนแปลงรุ่นของเอกสาร XML จำเป็นต้องมีเทคนิคที่สามารถรองรับการสืบค้นประวัติการเปลี่ยนแปลงรวมไปถึงการเปลี่ยนแปลงข้อมูลในแต่ละช่วงเวลา (temporal queries) ดังนั้น Wang และ Zaniolo จึงได้เสนอวิธีการใหม่ [30] โดยได้ขยายขีดความสามารถของวิธีการ SCCS เพื่อให้สามารถบันทึกความเปลี่ยนแปลงรุ่นของเอกสารและรองรับการสืบค้นแบบซับซ้อน โดยวิธีการใหม่นี้ได้กำหนดเอกสาร V (V-document) เพื่อแสดงรุ่นของเอกสาร XML โดยสร้างเอกสาร V นี้จากเอกสาร XML รุ่นแรกสุด ในเอกสาร V นี้แต่ละ element จะมีการแทรกแอททริบิวต์ (attributes) เพิ่มเข้าไป 2 แอททริบิวต์คือ vstart และ vend ซึ่งใช้สำหรับระบุช่วงของรุ่นเอกสารที่ element นั้นๆ ใช้ได้ โดยระบุค่าของรุ่นเริ่มต้นและรุ่นสุดท้าย อย่างไรก็ตามข้อจำกัดของวิธีการนี้ก็คือ ข้อมูลการเปลี่ยนแปลงรุ่นทั้งหมด ซึ่งเก็บอยู่ในเอกสาร V จำเป็นต้องถูกอ่านเพื่อใช้สำหรับสร้างรุ่นของเอกสารที่ต้องการ หรือเพื่อค้นหาข้อมูลการเปลี่ยนแปลงของเอกสาร

2.2 การควบคุมการเข้าถึงข้อมูลเอกสาร (Access Control)

การควบคุมการเข้าถึงข้อมูลของเอกสาร XML ใช้เพื่อป้องกันความเสียหายที่อาจเกิดกับข้อมูล และป้องกันการเปิดเผยข้อมูลที่เป็นความลับอันเกิดจากการอ่านหรือการปรับปรุงข้อมูลโดยผู้ที่ไม่ได้รับอนุญาต

2.2.1 คุณสมบัติและความสามารถที่สำคัญสำหรับการควบคุมการเข้าถึงข้อมูล

ทั้งนี้คุณสมบัติและความสามารถที่สำคัญที่วิธีการสำหรับการควบคุมการเข้าถึงข้อมูลที่ดีจำเป็นต้องมี ประกอบด้วยคุณสมบัติ 3 ประการดังนี้

(1) คุณสมบัติด้านการกำหนดสิทธิในการเข้าถึงข้อมูลในระดับ element และระดับแอททริบิวต์ (attribute)

(2) คุณสมบัติด้านการกำหนดนโยบายสำหรับแก้ไขความขัดแย้ง (conflict resolution policies) และนโยบายการเข้าถึงข้อมูลโดยปริยาย (default policies) เนื่องจากสิทธิในการเข้าถึงข้อมูลหนึ่งๆ ในเอกสารอาจมีความขัดแย้งกันได้ หรืออาจไม่ได้มีการกำหนดสิทธิสำหรับการเข้าถึงข้อมูลนั้นๆ ไว้ ซึ่งในกรณีที่มีความขัดแย้งเกิดขึ้น จำเป็นต้องมีการใช้นโยบายที่ได้กำหนดไว้เพื่อมาแก้ไขความขัดแย้งนั้นๆ ส่วนในกรณีที่ไม่ได้มีการระบุสิทธิสำหรับการเข้าถึงข้อมูลที่ต้องการ ระบบจำเป็นต้องนำนโยบายการเข้าถึงข้อมูลโดยปริยายมาใช้ ทั้งนี้นโยบายสำหรับแก้ไขความขัดแย้ง ที่สำคัญสามารถแบ่งออกเป็น 3 ประเภทดังนี้

- Instance-take-precedence นั่นคือ สิทธิที่ระบุสำหรับเอกสาร XML ในระดับเอกสาร (ระดับ document หรือระดับ instance) มีลำดับเหนือกว่าสิทธิที่ระบุสำหรับเอกสาร XML ในระดับ schema เช่น ในระบบข้อมูลเอกสาร XML ของโรงพยาบาล มีการกำหนดสิทธิในระดับ schema ว่าผู้ใช้ในกลุ่มแพทย์ไม่มีสิทธิในการอ่าน Patient-element แต่ในระดับเอกสารมีการระบุว่าผู้ใช้ในกลุ่มแพทย์มีสิทธิในการอ่านข้อมูล Patient-element จะเห็นได้ว่ามีความขัดแย้งในเรื่องสิทธิการเข้าถึงข้อมูลเกิดขึ้น ดังนั้นเมื่อนำนโยบาย Instance-take-precedence มาใช้ จะสรุปได้ว่าผู้ใช้ในกลุ่มแพทย์มีสิทธิในการอ่าน Patient-element
- Descendant-take-precedence เนื่องจากเอกสาร XML มีลักษณะเป็นลำดับชั้นหรือเป็นรูปแบบโครงสร้างต้นไม้ นั่นคือ XML element หนึ่งๆ สามารถอยู่ภายใต้ XML element อื่นได้ ดังนั้นการกำหนดสิทธิให้กับ element ที่อยู่ในลำดับชั้นสูงกว่าสามารถถ่ายทอดลงไปยัง element ที่อยู่ในลำดับชั้นต่ำกว่าได้ เมื่อมีความขัดแย้งเกิดขึ้นจาก

การถ่ายทอดสิทธิลงไปตามลำดับชั้นของโครงสร้างเอกสาร การใช้นโยบาย descendant-take-precedence จะถือว่าสิทธิในการเข้าถึงข้อมูลที่ระบุสำหรับ element ที่อยู่ในระดับต่ำกว่าถือว่ามีลำดับเหนือกว่าสิทธิที่ระบุในระดับสูงกว่า

- Denial-take-precedence ในกรณีที่มีความขัดแย้งเกิดขึ้นสำหรับการเข้าถึงข้อมูลที่ต้องการ ให้ถือว่าผู้ใช้ไม่สามารถเข้าถึงข้อมูลได้

นโยบายการเข้าถึงข้อมูลโดยปริยาย (default policies) แบ่งออกได้เป็น 2 ประเภทหลักๆ คือ

- นโยบายแบบเปิด (open policy) นั่นคือเมื่อไม่มีการกำหนดสิทธิสำหรับการเข้าถึงข้อมูลในเอกสาร XML ที่ต้องการ ให้ถือว่าผู้ใช้มีสิทธิเข้าถึงข้อมูลนั้นๆ ได้
- นโยบายแบบปิด (close policy) นั่นคือเมื่อไม่มีการกำหนดสิทธิสำหรับการเข้าถึงข้อมูลในเอกสาร XML ที่ต้องการ ให้ถือว่าผู้ใช้ไม่มีสิทธิเข้าถึงข้อมูลนั้นๆ

(3) คุณสมบัติด้านการกำหนดความหมายที่ชัดเจน ไม่คลุมเครือ เนื่องจาก เอกสาร XML รวมทั้งเอกสารที่กำหนดสิทธิในการเข้าถึงข้อมูล XML อาจมีการแลกเปลี่ยน สื่อสารระหว่างระบบงานต่างๆ ในอินเตอร์เน็ต ดังนั้นเอกสารที่กำหนดสิทธิในการเข้าถึงข้อมูลจำเป็นต้องมีความหมายที่ชัดเจนไม่คลุมเครือ เพื่อป้องกันการเกิดข้อผิดพลาดจากการติดต่อสื่อสารข้ามระบบงาน

เมื่อพิจารณาถึงโครงสร้างที่สำคัญของระบบควบคุมการเข้าถึงข้อมูลของเอกสาร XML จะเห็นได้ว่าสามารถแบ่งออกได้เป็น 2 ส่วนที่สำคัญ คือ 1) เอกสารที่กำหนดสิทธิในการเข้าถึงข้อมูลในระบบ และ 2) วิธีการในการบังคับใช้สิทธิต่างๆ ที่ได้กำหนดไว้

สำหรับรูปแบบและไวยากรณ์ของเอกสารที่กำหนดสิทธิในการเข้าถึงข้อมูลนั้น Oasis ซึ่งเป็นองค์กรกลางในการกำหนดมาตรฐานต่างๆ ที่เกี่ยวกับธุรกิจอิเล็คทรอนิคส์ (e-business) ได้เสนอภาษาสำหรับการกำหนดสิทธิในการเข้าถึงข้อมูลเอกสาร XML ขึ้น เรียกว่า ภาษา XACML (Extensible Access Control Markup Language) [21] โดยภาษา XACML นี้สามารถใช้ในการอธิบายเงื่อนไขในการเข้าถึงข้อมูล รวมทั้งยังเป็นภาษาสอบถามที่อนุญาตให้ผู้ใช้สอบถามได้ว่าผู้ใช้หนึ่งๆ มีสิทธิในการเข้าถึงข้อมูลที่ต้องการได้หรือไม่

งานวิจัยในปัจจุบันที่เกี่ยวข้องกับการควบคุมการเข้าถึงข้อมูลบนเอกสาร XML สามารถแบ่งออกเป็นกลุ่มใหญ่ๆ ได้ 3 กลุ่ม คือ

(1) Behavioral Access Control Model

(2) Provisional Access Control Model และ

(3) Rule-Based Access Control Model

รายละเอียดของงานวิจัยในแต่ละกลุ่ม จะได้นำเสนอตามลำดับ ดังนี้

2.2.2 Behavioral Access Control Model

งานวิจัยที่สำคัญในกลุ่มนี้ได้แก่งานที่เสนอโดย Damiani et al. [16,17,18] และ Bertino et al. [7,8,9] โดยทั้งสองงานวิจัยนี้ได้เสนอต้นแบบสำหรับการควบคุมการเข้าถึงข้อมูลในเอกสาร XML ในระดับ element และระดับแอททริบิวต์

งานที่เสนอโดยกลุ่มของ Damiani [16,17,18] นั้นกำหนดว่า เอกสาร XML หรือ DTD (Document Type Definition) ในระบบที่ต้องการควบคุมการเข้าถึง จะต้องแนบเอกสารที่กำหนดสิทธิ (authorization sheet) ในการเข้าถึงข้อมูลนั้นๆ ไว้ด้วยกัน โดยเอกสารที่กำหนดสิทธินี้เป็นเอกสารในภาษา XML เช่นกัน ซึ่งระบุถึงกฎเกณฑ์และเงื่อนไขในการเข้าถึงข้อมูล (authorization rule) ที่ต้องการป้องกันไว้อย่างชัดเจน ทั้งนี้การกำหนดกฎเกณฑ์และเงื่อนไขดังกล่าวผู้ใช้ต้องกำหนดข้อมูล 5 ชนิด (5-tuple) คือ ⟨subject, object, action, sign, type⟩ โดย

- Subject ใช้สำหรับกำหนดผู้ใช้หรือกลุ่มผู้ใช้ที่ต้องการเข้าถึงข้อมูล
- Object ใช้สำหรับอ้างอิงถึงข้อมูลที่ต้องการป้องกันการเข้าถึง โดยข้อมูลนี้อาจเป็นข้อมูลในเอกสาร XML ที่ผู้ใช้ต้องการป้องกันโดยตรง หรือเป็นเอกสาร DTD ซึ่งใช้กำหนดไวยากรณ์ของเอกสาร XML นั่นคือ การป้องกันจะครอบคลุมถึงเอกสาร XML ทั้งหมดที่มีไวยากรณ์ถูกต้องตามเอกสาร DTD ที่ระบุไว้
- Action ใช้ในการกำหนดรูปแบบการเข้าถึงข้อมูล เช่น การอ่าน การเขียน การปรับปรุง การลบ
- Sign ใช้เพื่อกำหนดว่าอนุญาตให้มีการเข้าถึงข้อมูลนั้นๆ หรือไม่ โดย Sign ที่เป็นค่าบวก (+)หมายถึง อนุญาตให้เข้าถึงข้อมูลที่กำหนดได้ ในขณะที่ Sign ที่เป็นค่าลบ (-) หมายถึงไม่อนุญาตให้เข้าถึงข้อมูล
- Type ใช้ในการกำหนดรูปแบบการถ่ายทอดสิทธิการเข้าถึงข้อมูล โดยมีค่าที่เป็นไปได้คือ local, recursive, local weak, recursive weak ทั้งนี้การกำหนดสิทธิแบบ local หมายถึงสิทธิที่กำหนดขึ้นสำหรับ element หนึ่งๆ จะถ่ายทอดไปยังแอททริบิวต์ของ element นั้น แต่ไม่ถ่ายทอดไปยัง element ที่อยู่ภายใต้ element นั้น ในขณะที่การกำหนดสิทธิแบบ recursive หมายถึงสิทธินั้นๆ สามารถถ่ายทอดไปยังแอททริบิวต์และ element ที่อยู่ภายใต้ element ที่ระบุ สำหรับสิทธิแบบ local weak และ recursive weak หมายถึง สิทธินั้นๆ สามารถถูกลบล้างได้ด้วยสิทธิอื่นที่ถ่ายทอดมาและไม่ได้เป็นเป็นสิทธิแบบ weak

ความหมายของการควบคุมการเข้าถึงข้อมูลของผู้ใช้หนึ่งๆ กำหนดอยู่ในรูปแบบของวิว (view) ของเอกสาร XML ซึ่งเรียกว่า วิวในการเข้าถึงเอกสารของผู้ใช้ (user's accessibility view) ซึ่งประกอบด้วย ส่วนของเอกสารที่ผู้ใช้ได้รับอนุญาตให้เข้าถึงได้ ทั้งนี้วิวของเอกสารของผู้ใช้แต่ละคนอาจแตกต่างกันได้ เนื่องจากผู้ใช้แต่ละคนอาจมีสิทธิในการเข้าถึงข้อมูลที่ต่างกัน นอกจากนี้งานวิจัยของกลุ่ม Damiani ยังได้สร้างอัลกอริธึมสำหรับสร้างวิวดังกล่าวโดยใช้เทคนิคที่เรียกว่า tree labeling และ tree pruning ซึ่งในเทคนิค tree labeling นี้เอกสาร XML ซึ่งมีโครงสร้างเป็นแบบต้นไม้ (tree) จะถูกท่องเที่ยว (traverse) จากส่วนของราก (root) ลงไปยังแต่ละ node ของเอกสาร ทั้งนี้เมื่อมีการท่องเที่ยวผ่านแต่ละ node โปรแกรมจะตรวจสอบว่าผู้ใช้มีสิทธิในการเข้าถึงข้อมูลสำหรับ node นั้นๆ หรือไม่ แล้วจึงให้ป้าย (label) กำกับ node นั้นๆ เพื่อบอกว่าผู้ใช้ได้รับอนุญาตให้เข้าถึงข้อมูลใน node นั้นหรือไม่ ดังนั้นเมื่อเสร็จสิ้นการทำ tree labeling แล้ว node ทุก node ในเอกสารจะมีป้ายคำว่า accessible (เข้าถึงได้) หรือ inaccessible (เข้าถึงไม่ได้) กำกับ หลังจากนั้นจึงถึงขั้นตอนการทำ tree pruning ซึ่งจะทำการตัด node ต่างๆ ในเอกสาร XML ที่มีป้าย inaccessible กำกับออก และเพื่อเป็นการคงไว้ซึ่งโครงสร้างของเอกสาร XML วิธีการนี้จึงกำหนดไว้ว่า node ที่มีป้าย inaccessible จะไม่ถูกตัดออกจากเอกสาร หากพบว่ามี node ซึ่งมีป้าย accessible ซ้อนอยู่ภายใต้ node ที่มีป้าย inaccessible นั้น

จุดเด่นของงานวิจัยนี้ คือ การออกแบบตัวต้นแบบสำหรับการควบคุมการเข้าถึงข้อมูล XML ซึ่งมุ่งเน้นที่การแก้ปัญหาที่เกี่ยวข้องในระดับสูง เช่น การกำหนดกฎเกณฑ์และเงื่อนไขในการเข้าถึงข้อมูล การหาคำตอบว่าผู้ใช้หนึ่งๆ สามารถเข้าถึงข้อมูลในรูปแบบที่ต้องการได้หรือไม่ รวมไปถึงการบังคับใช้กฎเกณฑ์การเข้าถึงข้อมูล

เมื่อเปรียบเทียบงานวิจัยของ Damiani et al. [16,17,18] และ Bertino et al. [7,8,9] โดยละเอียด พบว่า งานวิจัยทั้งสองมีความคล้ายคลึงกันดังมีรายละเอียดต่อไปนี้

- เป็นการขยายความสามารถมาจากต้นแบบสำหรับการควบคุมการเข้าถึงข้อมูลแบบดั้งเดิม (traditional discretionary access control models)
- รองรับการกำหนดสิทธิทั้งในระดับเอกสารและระดับ schema
- มีไวยากรณ์เป็นภาษา XML
- ความหมายของการควบคุมการเข้าถึงข้อมูลถูกกำหนดไว้ในรูปแบบของวิว (view) ของเอกสาร XML
- เทคนิคในการสร้างวิวของทั้งสองงานวิจัยมีพื้นฐานมากจากเทคนิคที่เรียกว่า tree labeling และ tree pruning

นอกจากความคล้ายคลึงกันในด้านโครงสร้างของทั้งสองงานวิจัยดังที่ได้กล่าวมาแล้ว งานทั้งสองมีความแตกต่างกันในส่วนของการออกแบบตัวต้นแบบ ได้แก่

- *การกำหนดผู้ใช้และกลุ่มผู้ใช้*

ในงานของ Damiani การระบุผู้ใช้หรือกลุ่มผู้ใช้ในสิทธิที่กำหนด ทำได้โดยการใช้ค่ารหัสประจำตัวผู้ใช้ (user ID) ชื่อกลุ่มผู้ใช้ หรือสถานที่ที่ผู้ใช้ใช้ในการเข้าถึงข้อมูล เช่น กลุ่มของเลข IP address สำหรับงานของ Bertino นั้นการระบุผู้ใช้สามารถทำได้โดยการอ้างถึงรหัสผู้ใช้เท่านั้น

- *นโยบายการแก้ไขความขัดแย้ง (conflict resolution) และการถ่ายทอดสิทธิในการเข้าถึงข้อมูล*

กลไกที่ใช้ในการแก้ไขความขัดแย้งในงานวิจัยของ Damiani นั้นจำกัดอยู่แค่เพียง กลไกแบบ most-specific-take-precedence และ denial-take-precedence ในขณะที่งานวิจัยของ Bertino รองรับการใช้นโยบาย instance-take-precedence, descendant-take-precedence และ denial-take-precedence

2.2.3 Provisional Access Control Model

ในขณะที่งานวิจัยที่เกี่ยวกับการควบคุมการเข้าถึงข้อมูลโดยส่วนใหญ่มุ่งเน้นเพียงแค่การอนุญาตหรือปฏิเสธการเข้าถึงข้อมูล งานวิจัยของ Hada และ Kudo ได้เสนอต้นแบบการควบคุมการเข้าถึงข้อมูลที่เรียกว่า provisional access control model [22] ซึ่งสามารถรองรับการควบคุมการเข้าถึงข้อมูลแบบซับซ้อนได้ โดยได้ขยายขีดความสามารถจากงานวิจัยแบบดั้งเดิม นั่นคือ นอกเหนือจากการอนุญาตหรือปฏิเสธการเข้าถึงข้อมูลที่ต้องการแล้ว ยังสามารถกำหนดเงื่อนไขเพิ่มเติมสำหรับการเข้าถึงข้อมูลได้ เช่น แจ้งให้ผู้ใช้ทราบว่า คำร้องในการเข้าถึงข้อมูลที่ต้องการนั้นจะได้รับอนุญาตหากผู้ใช้ปฏิบัติตามเงื่อนไขที่เกี่ยวข้องกับความปลอดภัยของข้อมูลนั้นๆ เช่น การลงนามยินยอมในข้อตกลงที่เกี่ยวข้อง จะเห็นได้ว่าคุณสมบัติที่เพิ่มเติมขึ้นมานี้เป็นประโยชน์อย่างยิ่งในหลากหลายงานประยุกต์ในระบบพาณิชย์อิเล็คทรอนิกส์ เช่น การประมูลแบบออนไลน์ การทำสัญญาออนไลน์ ซึ่งต่างก็ต้องใช้รูปแบบการเข้าถึงข้อมูลแบบมีเงื่อนไข

นอกเหนือไปจากนี้ Hada และ Kudo ยังได้เสนอภาษา XACL (XML Access Control Language) สำหรับ provisional access control model ที่ได้สร้างขึ้น โดยภาษา XACL นี้สามารถรองรับการเขียนเงื่อนไขในการเข้าถึงข้อมูลที่ยืดหยุ่นได้

แม้ว่างานวิจัยในกลุ่มของ 2.2.2 Behavioral Access Control Model และ 2.2.3 Provisional Access Control Model จะมีจุดเด่นและคุณสมบัติที่น่าสนใจหลายประการ แต่เมื่อพิจารณาเปรียบเทียบกับคุณสมบัติและความสามารถที่สำคัญที่วิธีการสำหรับการควบคุมการเข้าถึงข้อมูลที่ดี

จำเป็นต้องมี (ดังที่ได้วิเคราะห์ไว้ในข้อ 2.2.1) พบว่า งานวิจัยทั้งสองกลุ่มดังกล่าวสามารถรองรับเพียงคุณสมบัติที่หนึ่งและสอง หากแต่ยังขาดคุณสมบัติข้อสุดท้าย นั่นคือการกำหนดความหมายสำหรับกฎเกณฑ์และเงื่อนไขในการเข้าถึงข้อมูลที่ชัดเจนไม่คลุมเครือ เพื่อให้สามารถแลกเปลี่ยนสื่อสารเอกสารกำหนดสิทธิดังกล่าวระหว่างระบบงานต่างๆ ได้อย่างถูกต้อง

2.2.4 Rule-Based Access Control Model

ด้วยวัตถุประสงค์ที่จะเสนอต้นแบบสำหรับการควบคุมการเข้าถึงข้อมูลเอกสาร XML ที่มีคุณสมบัติครบถ้วนตามที่ได้วิเคราะห์ไว้ในข้อ 2.2.1 งานวิจัยในกลุ่ม Rule-Based Access Control Model จึงได้รับการพัฒนาขึ้น

งานวิจัยที่สำคัญในกลุ่มนี้ได้แก่ งานที่นำทฤษฎี XDD (XML Declarative Description) [3, 31] มาประยุกต์ใช้ [1] โดยลักษณะเด่นที่สำคัญของงานวิจัยนี้นอกเหนือจากการใช้ทฤษฎีการพรรณนาเชิงประกาศในการควบคุมการเข้าถึงข้อมูลแล้ว ก็คือ การนำภาษา RDF (Resource Description Framework) ซึ่งเป็นภาษามาตรฐานสำหรับการอธิบายข้อมูลต่างๆ ในระบบเว็บที่มีความหมาย (Semantic Web) มาใช้ในการอธิบายข้อมูลที่เกี่ยวกับสิทธิในการเข้าถึงเอกสาร XML เช่น ข้อมูลเกี่ยวกับผู้ใช้ กลุ่มผู้ใช้ ลำดับชั้นของกลุ่มผู้ใช้ รวมไปถึงความสัมพันธ์ประเภทอื่นๆ ซึ่งทำให้งานวิจัยนี้สามารถรองรับการนิรนัยข้อมูลที่ไม่ได้อธิบายไว้อย่างชัดแจ้งได้ เช่น เมื่อมีการกำหนดว่าผู้ใช้ ก เป็นผู้ใช้ในกลุ่มอาจารย์พิเศษ และกลุ่มอาจารย์พิเศษเป็น subclass ของกลุ่มอาจารย์ ดังนั้นจึงสามารถอนุมานได้ว่า ผู้ใช้ ก ก็เป็นผู้ใช้ในกลุ่มอาจารย์ด้วย นั่นคือ สิทธิที่กำหนดสำหรับกลุ่มอาจารย์จะถ่ายทอดมาถึงผู้ใช้ ก ด้วยเช่นกัน

ลักษณะเด่นที่สำคัญอีกประการหนึ่งของงานวิจัยนี้ คือ การเสนอวิธีการสำหรับการกำหนดสิทธิในการเข้าถึงข้อมูล นโยบายในการแก้ไขความขัดแย้ง นโยบายการเข้าถึงข้อมูลโดยปริยาย การสอบถามสิทธิในการเข้าถึงข้อมูล ในรูปแบบที่เป็นแบบเดียวกัน คือรูปแบบของกฎ (rules) ซึ่งอนุญาตให้ผู้ใช้สามารถกำหนดเงื่อนไขที่ซับซ้อนและเฉพาะเจาะจงสำหรับแต่ละงานประยุกต์ได้ รวมทั้งยังมีความหมายที่ชัดเจน ไม่คลุมเครือ ทำให้สามารถแลกเปลี่ยนสื่อสารข้ามระบบงานได้

3 ความต้องการสำหรับการควบคุมการเข้าถึงข้อมูล และการเปลี่ยนแปลงรุ่นของเอกสาร XML

3.1 การควบคุมการเข้าถึงข้อมูลเอกสาร XML (Access Control for XML Documents)

ในปัจจุบันภาษา XML ได้ถูกนำมาใช้อย่างแพร่หลายในระบบงานประยุกต์ต่างๆ เพื่อใช้ในการแสดงข้อมูลและการสื่อสารแลกเปลี่ยนข้อมูลระหว่างผู้ใช้จากต่างองค์กร ดังนั้นจำนวนเอกสาร XML ในองค์กรหนึ่งๆ รวมไปถึงจำนวนเอกสาร XML บนเว็บจึงมีปริมาณเพิ่มมากขึ้นตามลำดับ

การกำหนดสิทธิและการควบคุมการเข้าถึงข้อมูลเอกสาร XML ของกลุ่มผู้ใช้ระดับต่างๆ จึงมีความสำคัญเป็นอย่างมาก ทั้งนี้เพื่อป้องกันไม่ให้ผู้ใช้ที่ไม่มีสิทธิสามารถอ่านหรือแก้ไขข้อมูลที่ปกปิดได้ ทั้งนี้ต้นแบบสำหรับควบคุมการเข้าถึงข้อมูลเอกสาร XML ที่ดีจำเป็นต้องสนับสนุนวิธีการกำหนดการเข้าถึงข้อมูลทั้งในระดับ schema (schema level) ระดับเอกสาร (instance level) และในระดับที่ย่อยลงไปนั่นคือในระดับ element (element level) และระดับแอททริบิวต์ (attribute) โดยการกำหนดการเข้าถึงสามารถทำได้โดยการระบุโครงสร้างของเอกสาร XML (structure-base) หรือการระบุลักษณะของข้อมูลในเอกสาร (content-base) นอกจากนี้ เนื่องจาก XML เป็นเอกสารที่มีโครงสร้างแบบต้นไม้ ดังนั้นผู้ใช้ควรสามารถระบุได้ว่าต้องการให้สิทธิการเข้าถึงข้อมูลมีคุณสมบัติการถ่ายทอด (cascade) จาก element ในระดับแม่มาสู่ element ในระดับลูกได้หรือไม่

อนึ่ง เนื่องจากสิทธิในการเข้าถึงข้อมูลเอกสาร XML ของผู้ใช้แต่ละคนอาจมีความแตกต่างกันและมีความซับซ้อน ดังนั้นเพื่อให้การกำหนดและจัดการสิทธิของผู้ใช้ทำได้ง่ายและมีประสิทธิภาพ จึงควรจัดผู้ใช้เป็นกลุ่ม (group) หรือบทบาท (role) และกำหนดความสัมพันธ์ระหว่างกลุ่มหรือบทบาทนั้น ๆ ในรูปแบบลำดับชั้น (hierarchy) จากนั้นจึงกำหนดสิทธิในการเข้าถึงข้อมูลสำหรับแต่ละกลุ่มหรือบทบาทของผู้ใช้ โดยอนุญาตให้สิทธิในการเข้าถึงสามารถถ่ายทอดจากกลุ่มหรือบทบาทในลำดับชั้นที่สูงลงมาลำดับชั้นที่ต่ำได้ เช่น กำหนดให้มีกลุ่มผู้ใช้ 3 กลุ่ม คือ 1) กลุ่มเจ้าหน้าที่รักษาพยาบาล 2) กลุ่มพยาบาล และ 3) กลุ่มแพทย์ โดย กลุ่มเจ้าหน้าที่รักษาพยาบาลเป็นกลุ่มผู้ใช้ที่ครอบคลุมกลุ่มพยาบาล และกลุ่มแพทย์ กล่าวคือ กลุ่มเจ้าหน้าที่รักษาพยาบาลสามารถแบ่งออกได้เป็น 2 กลุ่มย่อย คือ กลุ่มพยาบาล และกลุ่มแพทย์ ดังนั้นสิทธิต่างๆ ที่กำหนดสำหรับผู้ใช้ในกลุ่มเจ้าหน้าที่รักษาพยาบาลสามารถถ่ายทอดลงไปยังผู้ใช้ในกลุ่มพยาบาลและกลุ่มแพทย์ได้

เมื่อพิจารณาโดยละเอียดจะเห็นได้ว่า สิทธิในการเข้าถึงข้อมูลสำหรับผู้ใช้หนึ่งๆ อาจมีความขัดแย้งกันได้ เช่น มีการระบุว่าผู้ใช้ในกลุ่มพยาบาลสามารถอ่านข้อมูล Patient-element ในเอกสาร patient.xml ได้ โดยสิทธิในการอ่านข้อมูลนี้สามารถถ่ายทอดลงไปยัง element ที่เป็นลูกๆหลานๆ ของ

patient-element ได้ เช่น Name, Address, MedicalRecord ในขณะเดียวกันก็มีการระบุอย่างชัดเจน โดยปฏิเสธการยินยอมให้ผู้ใช้ในกลุ่มพยาบาลสามารถอ่านข้อมูล MedicalRecord-element ในเอกสาร patient.xml จะเห็นได้ว่าสิทธิในการอ่านข้อมูล MedicalRecord-element ในเอกสาร patient.xml สำหรับผู้ใช้ในกลุ่มพยาบาลมีความขัดแย้งเกิดขึ้น ในกรณีนี้จำเป็นต้องมีการกำหนดนโยบายการแก้ไขความขัดแย้ง (conflict resolution policy) ทั้งนี้นโยบายสำหรับแก้ไขความขัดแย้ง ที่สำคัญสามารถแบ่งออกเป็น 3 ประเภทดังนี้

- Instance-takes-precedence: หากมีความขัดแย้งระหว่างสิทธิในการเข้าถึงข้อมูลที่กำหนดในระดับ schema และระดับเอกสาร ให้สิทธิที่กำหนดในระดับเอกสารมีลำดับเหนือกว่า เช่น ในระบบข้อมูลเอกสาร XML ของโรงพยาบาล หากผู้ใช้ ก ซึ่งอยู่ในกลุ่มผู้ใช้พยาบาลต้องการอ่านข้อมูลประวัติผู้ป่วยในเอกสาร patient.xml ในขณะที่ระดับ schema ระบุว่าผู้ใช้ในกลุ่มพยาบาลสามารถอ่านข้อมูลดังกล่าวได้ ในขณะที่ในระดับเอกสารระบุว่าผู้ใช้ในกลุ่มพยาบาลไม่สามารถอ่านข้อมูลดังกล่าวได้ ให้ถือว่าสิทธิในการเข้าถึงข้อมูลในระดับเอกสารมีลำดับสูงกว่า นั่นคือ ไม่อนุญาตให้ผู้ใช้ ก อ่านข้อมูลที่ต้องการได้
- Descendent-takes-precedence: ในกรณีที่มีความขัดแย้งระหว่างสิทธิในการเข้าถึงข้อมูลโดยสิทธิที่ขัดแย้งนี้อยู่ในระดับไวยากรณ์หรือระดับเอกสารด้วยกันทั้งคู่ ให้พิจารณาว่าสิทธิที่กำหนดใน element ระดับลูกๆหลานๆ (descendant level) มีลำดับเหนือกว่าสิทธิที่กำหนดในระดับบรรพบุรุษ (ancestor level) ดังนั้น หากพิจารณานำนโยบายแก้ไขความขัดแย้งนี้มาใช้กับตัวอย่างของความขัดแย้งที่เกิดขึ้นสำหรับสิทธิในการอ่านข้อมูล MedicalRecord-element ในเอกสาร patient.xml สำหรับผู้ใช้ในกลุ่มพยาบาล ดังที่ได้แสดงข้างต้น จะสรุปได้ว่า ผู้ใช้ในกลุ่มพยาบาลไม่สามารถอ่านข้อมูลดังกล่าวได้
- Denial-takes-precedence: สำหรับกรณีที่มีความขัดแย้งเกิดขึ้นระหว่างสิทธิในการเข้าถึงข้อมูลใดๆ ให้ปฏิเสธการเข้าถึงข้อมูลนั้นๆ

นอกจากความขัดแย้งที่อาจเกิดขึ้นระหว่างสิทธิในการเข้าถึงข้อมูลหนึ่ง ๆ ในระบบข้อมูลเอกสารแล้วปัญหาที่สามารถพบได้อีกประการหนึ่ง คือ การไม่ได้กำหนดสิทธิสำหรับการเข้าถึงข้อมูลที่ต้องการนั้นๆ ไว้ ซึ่งในกรณีนี้ระบบจำเป็นต้องนำนโยบายการเข้าถึงข้อมูลโดยปริยาย (default policies) มาใช้ โดยนโยบายการเข้าถึงข้อมูลโดยปริยายที่สำคัญ แบ่งออกได้เป็น 2 ประเภทหลัก ๆ คือ

- นโยบายแบบเปิด (open policy) นั่นคือเมื่อไม่มีการกำหนดสิทธิสำหรับการเข้าถึงข้อมูลในเอกสาร XML ที่ต้องการ ให้ถือว่าผู้ใช้มีสิทธิเข้าถึงข้อมูลนั้นๆ ได้

- นโยบายแบบปิด (close policy) นั่นคือเมื่อไม่มีการกำหนดสิทธิสำหรับการเข้าถึงข้อมูลในเอกสาร XML ที่ต้องการ ให้ถือว่าผู้ใช้ไม่มีสิทธิเข้าถึงข้อมูลนั้นๆ

นอกเหนือจากการกำหนดสิทธิในการเข้าถึงข้อมูล และการตรวจสอบว่าผู้ใช้หนึ่งๆ สามารถเข้าถึงข้อมูลที่ต้องการได้หรือไม่ หน้าที่อีกประการหนึ่งที่มีความสำคัญ คือ การสร้างวิว (view) ในการเข้าถึงเอกสารของผู้ใช้แต่ละราย (user's accessibility view) ซึ่งประกอบด้วย ส่วนของเอกสารที่ผู้ใช้ได้รับอนุญาตให้เข้าถึงได้ ทั้งนี้วิวของเอกสารของผู้ใช้แต่ละคนอาจแตกต่างกันได้ เนื่องจากผู้ใช้แต่ละคนอาจมีสิทธิในการเข้าถึงข้อมูลที่ต่างกัน

3.2 การควบคุมการเปลี่ยนแปลงรุ่นของเอกสาร XML (XML Document Version Control)

การควบคุมและจัดการการเปลี่ยนแปลงรุ่นเอกสาร ซึ่งมีหน้าที่ที่สำคัญในการควบคุมและดูแลรุ่นของเอกสารในระบบที่มีการเปลี่ยนแปลงอยู่เสมอ โดยต้องสนับสนุนการย้อนกลับไปดูเนื้อหาของเอกสารรุ่นก่อนหน้า รวมทั้งต้องรองรับการเปรียบเทียบรุ่นของเอกสารและอธิบายถึงความแตกต่างระหว่างรุ่นทั้งสอง นั่นคือจำเป็นต้องเก็บประวัติการเปลี่ยนแปลงรุ่นของเอกสารทั้งหมดอย่างถูกต้อง ทั้งนี้ข้อสำคัญที่ต้องพิจารณาสำหรับเทคนิคการจัดการการเปลี่ยนแปลงรุ่นของเอกสาร ก็คือต้องสามารถจัดการเอกสารในภาษา XML ได้อย่างถูกต้อง เนื่องจาก ถึงแม้ว่า เอกสาร XML จะเป็นเอกสารที่ประกอบด้วยข้อความตัวหนังสือ แต่ XML ก็เป็นภาษาแบบกึ่งโครงสร้าง โดยที่ เอกสาร XML 2 เอกสารอาจมีข้อความตัวหนังสือที่แตกต่างกัน แต่ไม่ถือว่ามีความแตกต่างกันหรือแตกต่างอย่างไม่มีนัยสำคัญ ตัวอย่างเช่น

- ลำดับของแอททริบิวต์ (attributes) ของ XML element หนึ่งๆ สามารถปรากฏในลำดับใดก็ได้ เช่น <Person gender="male" age="30"/> และ <Person age="30" gender="male"/> ถือว่าไม่มีความแตกต่างกัน
- การกำหนด namespace prefix ของเอกสารอาจมีความแตกต่างกันได้

นอกจากการเก็บประวัติการเปลี่ยนแปลงของเอกสารดังที่ได้กล่าวมาแล้ว หน้าที่ที่สำคัญอีกประการหนึ่งคือ ความสามารถในการสืบค้นประวัติการเปลี่ยนแปลง และการเปลี่ยนแปลงข้อมูลในแต่ละช่วงเวลา (temporal queries) เช่น ค้นหาความเปลี่ยนแปลงทั้งหมดที่ผู้ใช้ ก กระทำต่อเอกสารรุ่น 1.0 ถึงรุ่น 3.5 ค้นหาว่าใครเป็นผู้ลบข้อมูลในเอกสารรุ่น 2.0 หรือ ค้นหาความเปลี่ยนแปลงที่เกิดขึ้นทั้งหมดในระหว่างเวลา 12:00 ถึง 13:00 ในวันที่ 1 มกราคม 2548

3.3 ความต้องการและความสัมพันธ์ระหว่างการควบคุมการเข้าถึงข้อมูล และการเปลี่ยนแปลงรุ่นของเอกสาร XML

เมื่อพิจารณาโดยละเอียด จะเห็นได้ว่าหน้าที่ (function) ในการควบคุมการเปลี่ยนแปลงรุ่นเอกสารและการเข้าถึงข้อมูลในเอกสารต่างมีความสัมพันธ์สอดคล้องซึ่งกันและกัน โดยที่การเปลี่ยนแปลงรุ่นของเอกสารสามารถนำไปสู่การเปลี่ยนแปลงสิทธิในการเข้าถึงข้อมูลสำหรับแต่ละกลุ่มผู้ใช้ได้ ทั้งนี้รูปแบบความสัมพันธ์ของทั้งสองหน้าที่ดังกล่าวอาจแตกต่างกันไปขึ้นอยู่กับสาขางานประยุกต์ ตัวอย่างเช่น ในระบบงานโรงพยาบาล โดยทั่วไปผู้ใช้กลุ่มพยาบาลสามารถอ่านข้อมูลการวินิจฉัยโรคของผู้ป่วยได้ แต่ภายหลังเมื่อแพทย์พบว่าผู้ป่วยหนึ่งๆ ป่วยเป็นโรคร้ายแรง ผู้ใช้กลุ่มพยาบาลจะไม่มีสิทธิในการเข้าถึงข้อมูลการวินิจฉัยโรคของผู้ป่วยดังกล่าวโดยอัตโนมัติ ตัวอย่างนี้ชี้ให้เห็นว่า การเปลี่ยนแปลงข้อมูลและรุ่นในเอกสาร XML มีส่วนสัมพันธ์ทำให้ข้อมูลเปลี่ยนสถานะจากข้อมูลที่เปิดเผยต่อผู้ใช้บางกลุ่มได้ ไปเป็นข้อมูลที่เป็นความลับ และมีการจำกัดกลุ่มผู้ใช้ที่เข้าถึงข้อมูลได้

ข้อจำกัดที่เกิดขึ้นหากไม่มีการรวมหน้าที่ทั้งสองเข้าด้วยกันที่สำคัญ คือ ทำให้การออกแบบ schema สำหรับเอกสาร XML รวมไปถึงการกำหนดสิทธิในการเข้าถึงเอกสารมีความซับซ้อนมากขึ้น เนื่องจาก ผู้ออกแบบต้องพิจารณาถึงโครงสร้างและลักษณะข้อมูลของเอกสาร XML ที่ครอบคลุมเอกสารในทุกๆ รุ่น รวมทั้งต้องกำหนดสิทธิในการเข้าถึงข้อมูลของผู้ใช้ทุกๆ คน สำหรับเอกสารที่ซับซ้อนนี้ ซึ่งก็ทำให้สิทธิในการเข้าถึงข้อมูลมีความซับซ้อน ยากต่อการเข้าใจและปรับปรุง

จากความสัมพันธ์และปัญหาดังที่ได้กล่าวมาแล้วทำให้เกิดความต้องการในการรวมหน้าที่ทั้งสองหน้าที่นี้เข้าด้วยกัน เพื่อให้การจัดการเอกสาร XML มีประสิทธิภาพ มีความปลอดภัย และง่ายต่อการประยุกต์ใช้มากยิ่งขึ้น โดยมุ่งเน้นไปที่การประยุกต์ใช้ภาษาและเทคโนโลยีที่เป็นเอกภาพ แทนการใช้ภาษาและเทคโนโลยีที่หลากหลาย เพื่อป้องกันการเกิดความยุ่งยากซับซ้อนในการแลกเปลี่ยนและการรวมข้อมูลเอกสาร XML จากหลายๆ หน่วยงานที่มีสิทธิการเข้าถึงข้อมูลของกลุ่มผู้ใช้ที่แตกต่างกัน ซึ่งจะมีผลกระทบโยงไปถึงความปลอดภัยของข้อมูลได้อีกด้วย

3.4 การรวมการควบคุมการเข้าถึงข้อมูล และการเปลี่ยนแปลงรุ่นของเอกสาร XML เข้าด้วยกัน

กล่าวโดยสรุป การรวมการควบคุมการเข้าถึงข้อมูล และการเปลี่ยนแปลงรุ่นของเอกสาร XML เข้าด้วยกันมีข้อดี ดังต่อไปนี้

- นำไปสู่การสร้างต้นแบบที่สามารถควบคุมและจัดการการเข้าถึงข้อมูล และการเปลี่ยนแปลงรุ่นของเอกสาร XML ได้อย่างมีประสิทธิภาพ และง่ายต่อการเข้าใจและปรับปรุง
- อนุญาตให้ผู้ใช้สามารถกำหนดผลกระทบจากการเปลี่ยนรุ่นของเอกสาร ที่มีต่อการเปลี่ยนแปลงสิทธิในการเข้าถึงข้อมูลเอกสาร XML และในทางกลับกัน ได้อย่างเป็นธรรมชาติ เช่น เมื่อเอกสาร patient.xml ได้มีการเปลี่ยนแปลงรุ่น โดยมีการปรับปรุงสถานะของผู้ป่วยรายหนึ่งๆ จากผู้ป่วยนอกเป็นผู้ป่วยใน ให้ทำการเปลี่ยนแปลงสิทธิในการเข้าถึงข้อมูลผู้ป่วยรายนั้นๆ โดยอัตโนมัติ โดยอนุญาตให้พยาบาลสามารถแก้ไขข้อมูลที่เกี่ยวกับการดูแลผู้ป่วยรายนั้นๆ ได้
- เนื่องจากข้อมูลเกี่ยวกับรุ่นของเอกสาร (เช่น ชื่อผู้ใช้ที่ทำการเปลี่ยนแปลงรุ่นเอกสาร รายละเอียดการเปลี่ยนแปลง วัน/เวลาที่เปลี่ยนแปลง) รวมถึงสิทธิการเข้าถึงข้อมูล สามารถจัดเก็บในรูปแบบของเอกสาร XML ได้ ดังนั้นระบบที่รวมหน้าที่ทั้งสองเข้าด้วยกัน จึงสามารถจัดการและควบคุมการเข้าถึงรวมทั้งควบคุมการเปลี่ยนแปลงรุ่นของข้อมูลที่สำคัญดังกล่าวได้ ส่งผลให้ระบบดังกล่าวสามารถจัดการรุ่นและควบคุมความปลอดภัยของข้อมูลและเอกสารทั้งหมดในระบบได้อย่างมีประสิทธิภาพและเป็นเอกภาพ

อย่างไรก็ตาม ถึงแม้ว่าการรวมหน้าที่ทั้งสองเข้าด้วยกันจะมีข้อดีดังที่ได้กล่าวมาแล้ว ในการพัฒนาต้นแบบและวิธีการที่เหมาะสมจำเป็นต้องคำนึงถึงปัจจัยที่สำคัญดังต่อไปนี้

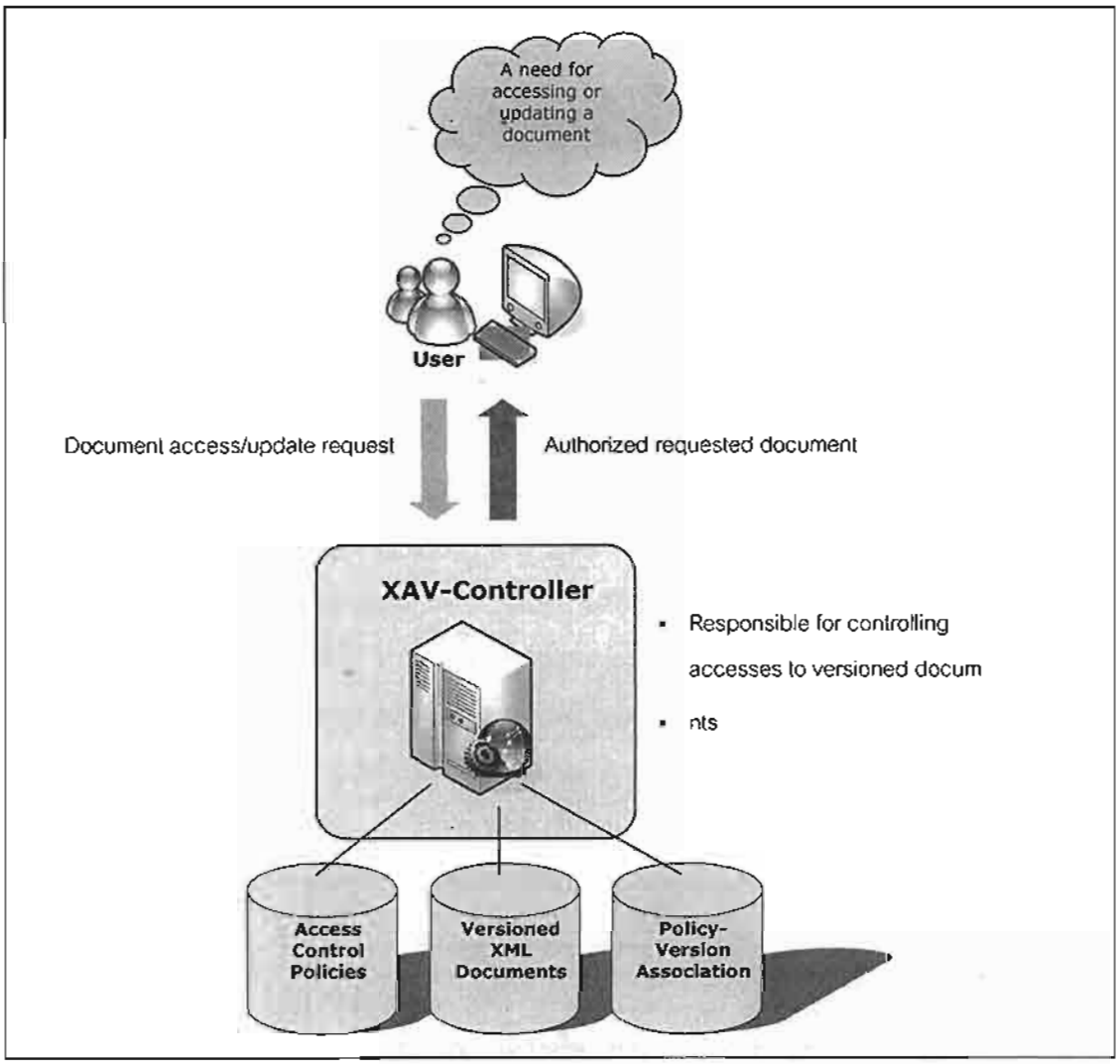
- ปัญหาการอนุมานข้อมูลที่ปกปิด: ผู้ใช้ที่ได้รับการปฏิเสธการเข้าถึงข้อมูลในเอกสารรุ่นหนึ่งๆ ต้องไม่สามารถอนุมานข้อมูลหรือโครงสร้างของข้อมูลที่ปกปิดนั้นๆ ได้จากความพยายามในการเข้าถึงข้อมูลในเอกสารรุ่นอื่น
- ประวัติการเปลี่ยนแปลงรุ่นของเอกสาร: เนื่องจากข้อมูลการเปลี่ยนแปลงรุ่นของเอกสารสามารถนำไปสู่ข้อสรุปบางประการเกี่ยวกับข้อมูลที่ปกปิดได้ ดังนั้นควรบันทึกข้อมูลดังกล่าวในรูปแบบของเอกสาร XML และกำหนดสิทธิการเข้าถึงข้อมูลที่เหมาะสมสำหรับเอกสารนั้นๆ เพื่อเป็นการป้องกันความปลอดภัยให้กับข้อมูลดังกล่าว นอกจากนี้การใช้เลขลำดับ (เช่น 1, 2, 2.1, 2.2, 3) ในการระบุเลขที่รุ่นส่งผลให้ผู้ใช้สามารถคาดเดาถึงความมีอยู่ของเอกสารรุ่นหนึ่งๆ ได้ เช่น ผู้ใช้ ก สามารถอ่านเอกสารรุ่นที่ 1.1 และ 1.4 ได้ ดังนั้นผู้ใช้ ก จะสามารถทราบได้ว่า ในระบบมีเอกสารรุ่น 1.2 และ 1.3 อยู่ แต่ผู้ใช้ ก ไม่สามารถอ่านได้ เพื่อป้องกันปัญหาดังกล่าวจึงควรใช้เวลาที่เอกสารรุ่นนั้นๆ ถูกแก้ไข (timestamp) ในการระบุเลขที่รุ่น
- การจัดการข้อมูล schema: ในระบบการจัดการเอกสาร XML หนึ่งๆ สามารถประกอบด้วยเอกสารจำนวนมาก ซึ่งอาจจะมี schema ที่แตกต่างกันไป นอกจากนี้ schema หนึ่งๆ ยัง

สามารถถูกแก้ไขและเปลี่ยนแปลงโดยผู้ใช้ได้ นั่นคือ แต่ละ schema สามารถมีได้หลายรุ่น และเพื่อให้การจัดการรุ่นของเอกสารทั้งหมดเป็นไปอย่างมีเอกภาพ จึงต้องมีการควบคุมการเข้าถึง รวมไปถึงการจัดการรุ่นของเอกสาร schema ทั้งหลายนี้ อย่างไรก็ดี มีข้อที่ต้องพิจารณาเป็นพิเศษ คือ เอกสาร XML ที่มีบรรพบุรุษเป็นเอกสารรุ่นเดียวกัน ควรจะใช้ schema รุ่นเดียวกันด้วยเพื่อให้ข้อมูลมีความถูกต้องตรงกันและการจัดการเป็นไปโดยง่าย

- ความสามารถในการจัดการเรื่องเวลา (temporal features): เนื่องจากข้อมูลเกี่ยวกับเวลาที่ใช้ในการอธิบายการเปลี่ยนแปลงรุ่นของเอกสาร เช่น เวลาที่เอกสารถูกแก้ไข (editing time) ช่วงเวลาที่ถือว่าเอกสารมีความถูกต้อง (valid time) จัดเป็นข้อมูลที่มีความสำคัญในการกำหนดสิทธิในการเข้าถึงข้อมูล โดยนำมาใช้ในการระบุเงื่อนไขของการเข้าถึงข้อมูล เช่น เมื่อเอกสารมีอายุครบ 10 ปีให้การเข้าถึงข้อมูลในเอกสารนั้นๆ ทำได้โดยสาธารณะ ดังนั้นในการพัฒนาต้นแบบและวิธีการในการควบคุมการเข้าถึงข้อมูลเอกสาร และการเปลี่ยนแปลงรุ่นของเอกสาร XML ที่เหมาะสมจึงจำเป็นต้องพิจารณาถึงความสามารถในการจัดการเรื่องเวลานี้ด้วย ซึ่งรวมไปถึงการรองรับการสืบค้นโดยใช้ข้อมูลเกี่ยวกับเวลา
- การเปิดเผยข้อมูลโดยไม่สมควร (improper disclosure of confidential data): เมื่อการแก้ไขข้อมูลในเอกสาร XML นำไปสู่การเปลี่ยนแปลงในสิทธิการเข้าถึงข้อมูลนั้นๆ ระบบที่ดีควรมีกลไกในการเตือนเพื่อป้องกันการเกิดการเปิดเผยข้อมูลโดยไม่สมควรขึ้น ซึ่งอาจจะเกิดขึ้นเมื่อการแก้ไขข้อมูลเอกสารหนึ่งๆ โดยผู้ใช้หนึ่งๆ ส่งผลให้ผู้ใช้นั้นๆ มีสิทธิในการเข้าถึงข้อมูลที่ถูกปกปิดได้

4 ระบบควบคุมการเปลี่ยนแปลงรุ่น และการเข้าถึงข้อมูลเอกสาร XML

4.1 สถาปัตยกรรมของระบบ



รูปที่ 4.1: สถาปัตยกรรมของระบบควบคุมการเปลี่ยนแปลงรุ่นและการเข้าถึงข้อมูลเอกสาร XML

รูปที่ 4.1 แสดงสถาปัตยกรรมของระบบควบคุมการเปลี่ยนแปลงรุ่นและการเข้าถึงข้อมูลเอกสาร XML หรือเรียกโดยย่อว่า ระบบ XAV-Controller (XML Document Access and Version Controller) ทั้งนี้จะเห็นได้ว่า องค์ประกอบที่สำคัญของระบบดังกล่าว ประกอบด้วย

(1) Access Control Policy Base: ฐานข้อมูลสำหรับเก็บรวบรวมกฎและสิทธิในการเข้าถึงข้อมูลของผู้ใช้ในระบบ รวมถึงนโยบายการแก้ไขความขัดแย้ง

(2) Versioned XML Document Base: ซึ่งเป็นฐานข้อมูลสำหรับเก็บรวบรวมเอกสารและการเปลี่ยนแปลงรุ่นของเอกสาร XML

(3) Policy-Version Association Rule Base: ฐานข้อมูลสำหรับเก็บกฎที่อธิบายความสัมพันธ์ระหว่างการควบคุมการเข้าถึงข้อมูลและการเปลี่ยนแปลงรุ่นของเอกสาร XML

(4) XAV-Controller Engine: ส่วนประมวลผลการควบคุมการเข้าถึงข้อมูลและการเปลี่ยนแปลงรุ่นของเอกสาร XML

อนึ่งรายละเอียดของแต่ละองค์ประกอบจะได้อธิบายในลำดับถัดไป

4.2 Access Control Policy Base

จากการศึกษาเทคโนโลยีที่สำคัญที่เกี่ยวข้องกับการจัดเก็บและประมวลผลข้อมูลกฎและสิทธิในการเข้าถึงข้อมูลเอกสาร XML พบว่า มาตรฐานภาษา RDF (Resource Description Framework) ซึ่งกำหนดโดย World Wide Web Consortium เพื่อเป็นมาตรฐานสำหรับอธิบายข้อมูลและแลกเปลี่ยนข้อมูลบน Semantic Web หรือเว็บที่มีความหมายนั้น สามารถนำมาประยุกต์ใช้เพื่อให้ระบบสามารถรองรับความต้องการในการแลกเปลี่ยนข้อมูลกฎและสิทธิระหว่างระบบต่างๆ รวมถึงสามารถอนุมาน (inference) ข้อมูลที่ไม่ได้อธิบายอย่างชัดแจ้งได้

ดั้งนั้นในการออกแบบฐานข้อมูลดังกล่าว จึงได้นำเทคโนโลยีภาษา RDF มาใช้เป็นสำคัญ โดยโครงสร้างกฎ/สิทธิการเข้าถึงข้อมูลที่ได้กำหนดขึ้น มีส่วนประกอบที่สำคัญดังนี้

- *subject*: ข้อมูลเกี่ยวกับผู้ใช้ กลุ่มของผู้ใช้ หรือข้อมูลอื่นๆ (เช่น อายุ ip address) ที่ใช้ระบุถึงผู้ใช้ที่กฎ/สิทธิดังกล่าวอ้างอิงถึง
- *target*: ระบุเอกสารรุ่นแรกที่กฎนี้นำไปใช้ได้
- *path*: *XPath* expression ที่ระบุ element หรือ attribute ในเอกสาร XML
- *level*: ค่าที่เป็นไปได้คือ "*#instance*" หรือ "*#schema*" ซึ่งกำหนดว่ากฎการเข้าถึงนั้นเป็นกฎในระดับใด
- *privilege*: ค่าที่เป็นไปได้คือ "*#read*" หรือ "*#write*"
- *type*: ค่าที่เป็นไปได้คือ "*#cascade*" หรือ "*#no_propagation*"
- *sign*: ค่าที่เป็นไปได้คือ "*#plus*" หรือ "*#minus*"
- *priority*: ค่าจำนวนเต็มที่ใช้ระบุลำดับความสำคัญของกฎการเข้าถึงข้อมูลนั้นๆ

รูปที่ 4.2 แสดงตัวอย่างการกำหนดกฎการเข้าถึงข้อมูล รวมถึงนโยบายการแก้ไขความขัดแย้ง (conflict resolution policy) และนโยบายการเข้าถึงข้อมูลโดยปริยาย (default policy)

```
<Policy rdf:about="#p1">
   <conflictResolution
      rdf:resource="#denial_take_precedence"/>
   <defaultPolicy rdf:resource="#closed"/>
   <authorization rdf:resource="#a1"/>
   <authorization rdf:resource="#a2"/>
</Policy>

<Authorization rdf:about="#a1">
   <sign rdf:resource="#plus"/>
   <subject rdf:resource="#GeneralCustomer"/>
   <target>catalog.xml</target>
   <path> / </path>
   <privilege rdf:resource="#read"/>
   <type rdf:resource="#cascade"/>
   <sign rdf:resource="#plus"/>
</Authorization>

<Authorization rdf:about="#a2">
   <sign rdf:resource="#plus"/>
   <subject rdf:resource="#GeneralCustomer"/>
   <target>catalog.xml</target> <path>//cost</path>
   <privilege rdf:resource="#read"/>
   <type rdf:resource="#cascade"/>
   <sign rdf:resource="#minus"/>
</Authorization>
```

a. RDF description of access control policy p_1.

```
<Policy rdf:about="#p2">
   <conflictResolution
      rdf:resource="#denial_take_precedence"/>
   <defaultPolicy rdf:resource="#closed"/>
   <authorization rdf:resource="#a3"/>
   <authorization rdf:resource="#a4"/>
</Policy>

<Authorization rdf:about="#a3">
   <sign rdf:resource="#plus"/>
   <subject rdf:resource="#MemberCustomer"/>
   <target>catalog.xml</target> <path> / </path>
   <privilege rdf:resource="#read"/>
   <type rdf:resource="#cascade"/>
   <sign rdf:resource="#plus"/>
</Authorization>

<Authorization rdf:about="#a4">
   <sign rdf:resource="#plus"/>
   <subject rdf:resource="#MemberCustomer"/>
   <target>catalog.xml</target>
<path>//cost</path>
   <privilege rdf:resource="#read"/>
   <type rdf:resource="#cascade"/>
   <sign rdf:resource="#minus"/>
</Authorization>
```

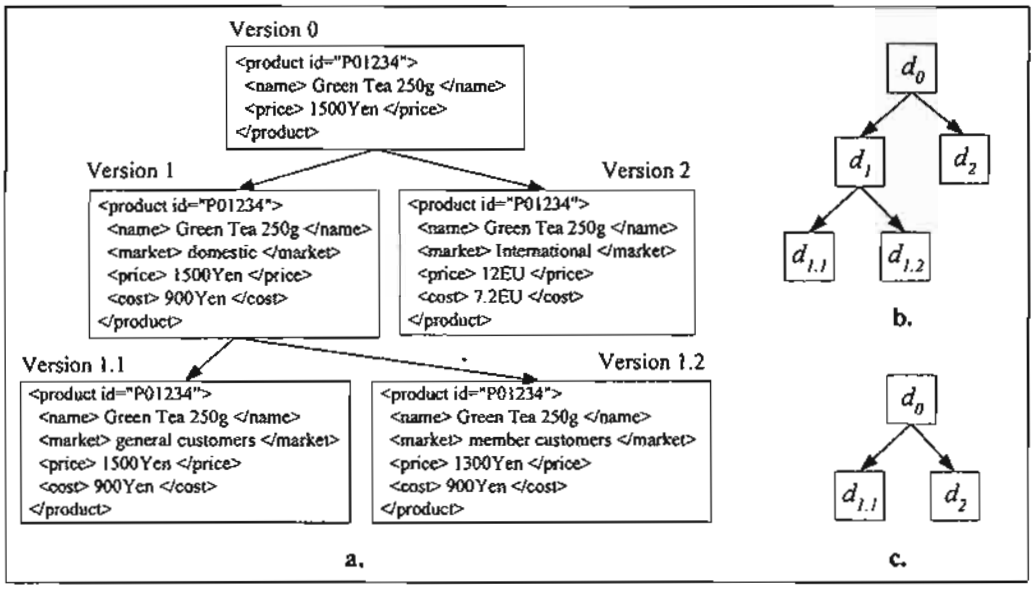
b. RDF description of access control policy p_2.

รูปที่ 4.2: ตัวอย่างกฎและนโยบายการเข้าถึงข้อมูล

4.3 Versioned XML Document Base

จากการศึกษาวิเคราะห์วิธีการจัดเก็บรุ่นของเอกสาร XML พบว่าหากคำนึงถึงเนื้อที่ในการจัดเก็บเอกสารในรุ่นต่างๆ เป็นสำคัญ วิธีการเก็บแบบ Change-Centric Method จะช่วยให้สามารถประหยัดพื้นที่ในการจัดเก็บข้อมูลได้มาก โดยวิธีนี้ จัดเก็บเอกสาร XML รุ่นล่าสุดอย่างสมบูรณ์ และเก็บเฉพาะส่วนของความเปลี่ยนแปลงย้อนกลับไปยังเอกสารก่อนหน้า (Delta $\Delta d_{i,j}$) ดังนั้นหากต้องการสร้างเอกสารรุ่น d_i จะทำได้โดยนำ $\Delta d_{i,j}$ มาประยุกต์เข้ากับเอกสาร d_j

นอกจากนี้จะเห็นได้ว่า เอกสาร XML ในรุ่นต่างๆ กันนั้น ต่างมีความสัมพันธ์กันในรูปแบบต้นไม้ (version tree) ซึ่งทำให้สามารถมองภาพรวมของเอกสารทั้งหมดได้ โดยเอกสารรุ่นหนึ่งๆ สามารถแสดงได้ด้วย node ใน version tree และหากเอกสาร d_i เป็นเอกสารรุ่นที่ได้จากการปรับปรุงเอกสาร d_j ดังนั้น node ซึ่งแสดงเอกสาร d_i ใน version tree จะเป็น child node ของเอกสาร d_j และสามารถสร้างเอกสารรุ่น d_i โดยนำ $\Delta d_{i,j}$ มาประยุกต์เข้ากับเอกสาร d_j ดังแสดงตัวอย่างในรูปที่ 4.3

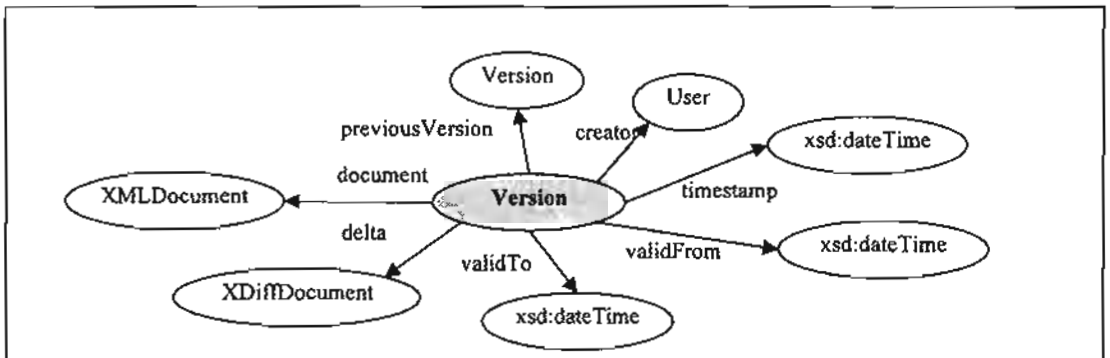


รูปที่ 4.3: ตัวอย่างมุมมองรุ่นของเอกสาร XML แบบต้นไม้

ดังนั้นจะเห็นได้ว่าข้อมูลการเปลี่ยนแปลงรุ่นของเอกสารเป็นข้อมูลที่มีความสำคัญ งานวิจัยนี้จึงได้สร้าง RDF schema เพื่อใช้อธิบายข้อมูลดังกล่าวอย่างมีความหมาย ซึ่งประกอบด้วย

- *document*: ใช้เพื่ออ้างอิงถึงเอกสารรุ่นแรก
- *previousVersion*: ใช้เพื่ออ้างอิงถึงเอกสารรุ่นก่อนหน้า
- *creator*: ผู้ที่สร้างเอกสารรุ่นนี้
- *timeStamp*: เวลาที่เอกสารนี้ถูกสร้าง
- *validFrom*, *validTo*: ใช้อธิบายถึงช่วงเวลาที่เอกสารนี้ถูกต้อง
- *delta*: ใช้อธิบายถึงการเปลี่ยนแปลงที่เกิดขึ้นระหว่างเอกสารนี้และเอกสารก่อนหน้า

รูปที่ 4.4 แสดง schema ดังกล่าว และแสดงตัวอย่างการอธิบายข้อมูลการเปลี่ยนแปลงรุ่นของเอกสาร



a. RDF schema for version description.

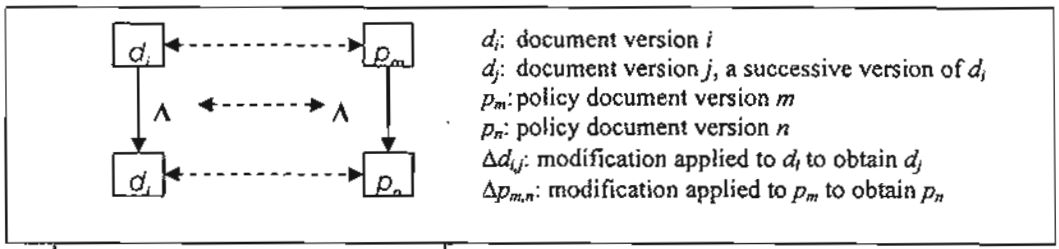
```
<Version rdf:about="#d1.2">
    <document rdf:resource="catalog.xml"/>
    <previousVersion rdf:resource="#d1"/>
    <creator rdf:resource="#john"/>
    <timestamp>2004-02-01T09:00:00</timestamp>
    <validFrom>2004-03-01T00:00:00</validFrom>
    <validTo>2004-04-01T00:00:00</validTo>
    <delta rdf:parseType ="Literal">
        <xd:xmldiff>  <xd:node match="1">
            <xd:change match="2">member customers</xd:change>
            <xd:change match="3">1300Yen</xd:change>
        </xd:node> </xd:xmldiff>
    </delta>
</Version>
```

b. RDF description of product catalog $d_{1.2}$

รูปที่ 4.4: RDF schema และตัวอย่างการอธิบายข้อมูลการเปลี่ยนแปลงรุ่นของเอกสาร

4.4 Policy-Version Association Rule Base

เนื่องจากการเปลี่ยนแปลงรุ่นของเอกสาร XML อาจนำไปสู่การเปลี่ยนแปลงกฎการเข้าถึงเอกสารนั้นๆ รูปที่ 4.5 แสดงความสัมพันธ์ดังกล่าว



รูปที่ 4.5: ความสัมพันธ์ระหว่าง การเปลี่ยนแปลงรุ่นของเอกสาร และสิทธิการเข้าถึงข้อมูล

จากการวิเคราะห์ดังกล่าว ทำให้สามารถนำทฤษฎี RDF Declarative Description (RDD) มาประยุกต์ เพื่อแสดงกฎแห่งความสัมพันธ์ดังกล่าว ซึ่งแสดงได้โดย RDF clause ในรูปแบบดังนี้

$$H \leftarrow B_1, \ldots, B_n.$$

- H คือ RDF expression ซึ่งอธิบายความเชื่อมโยงระหว่างเอกสาร XML และกฎการเข้าถึงข้อมูลสำหรับเอกสารนั้นๆ
- B_i คือ RDF expression หรือ RDF constraint ซึ่งอธิบายเงื่อนไขที่ต้องเป็นจริง เพื่อให้สามารถได้มาซึ่งความเชื่อมโยงดังแสดงใน H ได้

รูปที่ 4.6 แสดงตัวอย่าง clause ดังกล่าว

```
<Policy_Version_Association>
        <policy rdf:resource="#p2"/>
        <version rdf:resource=$S:Y/>
</Policy_Version_Association>
    ←       <Version rdf:about=$S:Y>
                <document rdf:resource="catalog.xml"/>
                <previousVersion rdf:resource=$S:X/>
                <delta>
                    <xd:xmldiff rdf:parseType="Literal">
                        <xd:change match=$S:anynode>
                            member customers </xd:change>
                        $E:otherchanges
                    </xd:xmldiff>
                </delta>
                $E:versionProperties
            </Version>,
            <Policy_Version_Association>
                <policy rdf:resource="#p1"/>
                <version rdf:resource=$S:X/>
            </Policy_Version_Association>
```

a. Policy-version-association clause example.

```
<Policy_Version_Association>
    <policy rdf:resource="#p2"/>
    <version rdf:resource="#d1.2"/>
</Policy_Version_Association>
```

b. Derived Policy_Version_Association element.

รูปที่ 4.6: ตัวอย่างกฎซึ่งแสดงความสัมพันธ์ระหว่างการเปลี่ยนแปลงรุ่นของเอกสารและสิทธิการเข้าถึงข้อมูล

4.5 XAV-Controller Engine

ในส่วนนี้จะกล่าวถึง XAV-Controller Engine หรือ ส่วนประมวลผลการควบคุมการเข้าถึงข้อมูลและการเปลี่ยนแปลงรุ่นของเอกสาร XML ซึ่งจะนำข้อมูลที่จัดเก็บในฐานข้อมูลทั้ง 3 ส่วนดังที่ได้อธิบายโดยละเอียดใน Section 4.2 – 4.4 มาประมวลผล เพื่อใช้สำหรับตรวจสอบสิทธิการเข้าถึงข้อมูลในเอกสาร XML รุ่นต่างๆ โดยผู้ใช้แต่ละคน โดยเทคโนโลยีที่สำคัญที่สามารถรองรับการทำงานของส่วนประมวลผลนี้จำเป็นต้องมีคุณสมบัติดังต่อไปนี้

- สามารถประมวลผลข้อมูลเอกสาร RDF ได้ เนื่องจากกฎการเข้าถึงข้อมูล รวมถึงข้อมูลการเปลี่ยนแปลงรุ่นเอกสาร XML ได้จัดเก็บในรูปแบบเอกสาร RDF
- สามารถประมวลผลกฎซึ่งแสดงความสัมพันธ์ระหว่างการเปลี่ยนแปลงรุ่นของเอกสาร และสิทธิการเข้าถึงข้อมูล ซึ่งจัดเก็บในรูปแบบของ RDF Clause ในทฤษฎี RDD ได้

จากคุณสมบัติทั้ง 2 ข้อ ดังกล่าว ทำให้การวิจัยนี้เลือกใช้ภาษา XET (XML Equivalent Transformation) [4] สำหรับการพัฒนาส่วนประมวลผล XAV-Controller Engine นี้

5 การจัดการเอกสาร XML สำหรับเก็บความรู้ประเภทออนโทโลยี

ออนโทโลยี (ontology) คือ วิธีบรรยายความรู้ (knowledge representation) แบบหนึ่ง โดยมุ่งเน้นที่การอธิบายแนวความคิดตามขอบเขตที่สนใจ (domain conceptualization) อนึ่งภาษาที่เป็นมาตรฐานและสนับสนุนการบรรยายออนทีโลยี โดยได้รับการสนับสนุนจากองค์กร World Wide Web Consortium (W3C) คือ RDF (Resource Description Framework) [23], RDFS (RDF Schema) [10] และ OWL (Web Ontology Language) [5] นอกจากนี้เพื่อให้การเผยแพร่แลกเปลี่ยนออนโทโลยีระหว่างองค์กรเป็นไปอย่างกว้างขวาง ภาษาที่สนับสนุนการบรรยายออนโทโลยีดังกล่าวนี้ จึงได้ใช้ภาษา XML เป็นไวยากรณ์พื้นฐาน

ปัจจุบันจำนวนเอกสารออนโทโลยีบนเว็บมีจำนวนมากขึ้นเป็นลำดับ อย่างไรก็ตามระบบสืบค้นออนโทโลยีที่มีอยู่ เช่น Swoogle [19], OntoKhoj [26] และ OntoSearch [28] เพียงแค่สนับสนุนการใช้คำค้นสำหรับการค้นหา โดยไม่อนุญาตให้ผู้ใช้อธิบายโครงสร้าง ชื่อคลาส (class name) ชื่อคุณสมบัติ (property name) หรือความสัมพันธ์ระหว่างคลาส/คุณสมบัติ ที่ถูกกำหนดไว้ในออนโทโลยีที่ต้องการได้ ส่งผลให้ผลลัพธ์จากการค้นหามีจำนวนมากแต่ไม่ตรงตามความต้องการของผู้ใช้

ดังนั้นเพื่อแก้ปัญหาดังกล่าว งานวิจัยนี้จึงพัฒนาต้นแบบ SQORE (Semantic Query based Ontology Retrieval) [2] สำหรับจัดการเอกสาร XML สำหรับเก็บความรู้ประเภทออนโทโลยี

5.1 สถาปัตยกรรมของ SQORE

รูปที่ 5.1 แสดงสถาปัตยกรรมของ SQORE ซึ่งประกอบด้วยองค์ประกอบที่สำคัญ 4 ส่วนดังนี้

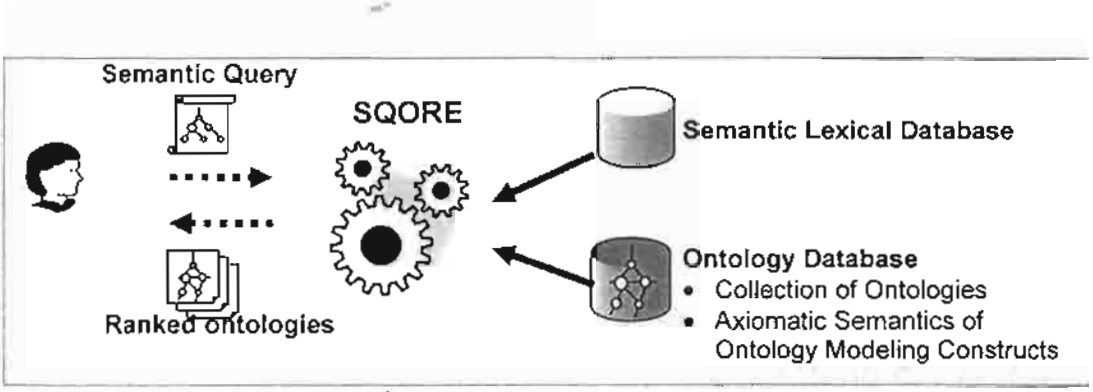
(1) Semantic Query: คำสืบค้นแบบมีความหมาย

(2) SQORE Retrieval Engine: ระบบสืบค้น

(3) Ontology Database: ฐานข้อมูลออนโทโลยีซึ่งประกอบด้วย

- ออนโทโลยีซึ่งบรรยายโดยภาษา OWL
- ความหมายของแต่ละคำสำคัญในการออกแบบออนโทโลยี (ontology modeling construct)

(4) Semantic Lexical Database: ฐานข้อมูลคำศัพท์และความสัมพันธ์ระหว่างคำศัพท์



รูปที่ 5.1: สถาปัตยกรรมของ SQORE

กล่าวโดยสรุป ระบบ SQORE ทำงานดังนี้: ผู้ใช้กำหนดและส่งคำสืบค้นที่มีความหมาย ซึ่งอธิบายรายละเอียดของออนโทโลยีที่ต้องการมายังระบบ SQORE จากนั้นระบบประมวลผลโดยค้นหาออนโทโลยีในฐานข้อมูล user โดยพิจารณาถึงความหมายของแต่ละออนโทโลยีในการประเมินความสอดคล้องของออนโทโลยีและคำสืบค้น ซึ่งส่งผลให้ระบบสามารถนิรนัยข้อมูลและความหมายโดยนัยที่อยู่ในคำสืบค้นและออนโทโลยีได้ นอกจากนี้เมื่อชื่อคลาสหรือชื่อคุณสมบัติที่กำหนดในคำสืบค้นที่มีความหมายและในออนโทโลยีไม่ตรงกัน นั่นหมายความว่า มีความเป็นไปได้ 4 แบบนั่นคือ

- *equivalence* (≡): คำทั้งสองนั้นมีความหมายเหมือนกัน
- *more general* (⊇): คำที่อยู่ในคำสืบค้นมีความหมายกว้างกว่า
- *less general* (⊆): คำที่อยู่ในคำสืบค้นมีความหมายแคบกว่า
- *unknown* (≠): ไม่สามารถหาความสัมพันธ์ของคำทั้งสองได้

ดังนั้นเพื่อให้ ระบบ SQORE สามารถค้นหาความสัมพันธ์ทั้ง 4 แบบดังกล่าวได้ SQORE จึงนำฐานข้อมูลคำศัพท์และความสัมพันธ์ระหว่างคำศัพท์มาใช้ ตัวอย่างของฐานข้อมูลดังกล่าว ได้แก่ WordNet [25] ท้ายที่สุด SQORE คำนวณหาค่าความเหมือน (semantic similarity score) ระหว่างคำสืบค้นที่มีความหมาย และออนโทโลยีที่มีอยู่ในระบบ โดยค่าดังกล่าวมีค่าที่เป็นไปได้ระหว่าง 0 (หมายถึงมีความต่างมากที่สุด) ถึง 1 (หมายถึงมีความเหมือนมากที่สุด) เมื่อพิจารณาจากค่าความเหมือนที่คำนวณได้ SQORE สามารถค้นหาออนโทโลยีที่มีความใกล้เคียงกับคำสืบค้นที่มีความหมาย และส่งคืนกลับให้ผู้ใช้เป็นคำตอบได้

5.2 ฐานความรู้ออนโทโลยี

จากการวิเคราะห์ถึงโครงสร้างและความหมายของออนโทโลยีที่กำหนดโดยภาษา RDF(S) และภาษา OWL โดยละเอียด พบว่าการนำทฤษฎีการพรรณาเชิงประกาศด้วยภาษา XML (XDD: XML Declarative Description Theory) [3, 31] เข้ามาใช้ ช่วยให้ SQORE สามารถโมเดลและ

กำหนดความหมายของออนโทโลยีในฐานข้อมูลได้อย่างถูกต้อง โดยอนุญาตให้ระบบสามารถนิรนัยข้อมูลและความหมายโดยนัยที่อยู่ในคำสืบค้นและออนโทโลยีได้

ตัวอย่างที่ 1 (ออนโทโลยี และความหมายของออนโทโลยี) รูปที่ 5.2 แสดงตัวอย่างออนโทโลยี O ซึ่งประกอบด้วยอิลิเมนต์ E_1 – E_7 ที่บรรยายอย่างชัดแจ้ง ในขณะที่รูปที่ 5.3 เสนอตัวอย่างการกำหนดความหมายของคำสำคัญในการออกแบบออนโทโลยีในรูปของการพรรณนาด้วยทฤษฎี XDD (XDD description) โดยให้ XDD description $D = \{C_1, C_2, C_3\}$ นั่นคือ กฎ C_1 – C_3 กำหนดความหมายของ rdfs:subClassOf owl:DatatypeProperty, rdfs:domain และ rdfs:range ซึ่งทำให้สามารถนิรนัยความหมายของออนโทโลยี O ได้ถูกต้องสมบูรณ์ขึ้น (ดังแสดงโดยรูปที่ 5.4) โดยนอกเหนือจากข้อความ E_1 – E_7 ที่บรรยายอย่างชัดแจ้งโดยออนโทโลยี O แล้ว ยังรวมถึงความหมายที่อธิบายโดยนัยอื่นๆ ด้วย เช่น จากรูปที่ 5.2 ซึ่งอธิบายออนโทโลยี O พบว่าคลาส PhDStudent นั้นได้ถูกกำหนดเป็น subClassOf คลาส Student ในขณะที่ คลาส Student นั้นเป็น subClassOf คลาส Person ดังนั้นโดยกฎ C_1 ซึ่งอธิบายว่า subClassOf นั้นมีคุณสมบัติแบบถ่ายทอด (transitive) ทำให้สามารถนิรนัยได้ว่า คลาส PhDStudent ก็เป็น subClassOf คลาส Person เช่นกัน แม้ว่าความสัมพันธ์นี้จะไม่ได้ถูกบรรยายไว้อย่างชัดแจ้งก็ตาม นั่นหมายความว่า หากมีการสืบค้นหาออนโทโลยีที่มีการกำหนดคลาส PhDStudent ให้เป็น subClassOf คลาส Person ออนโทโลยี O นี้ก็จะเป็นหนึ่งในคำตอบของการสืบค้นดังกล่าว

❑

```
E1: <owl:Class rdf:ID="Person"/>
E2: <owl:Class rdf:ID="Student"/>
E3: <owl:Class rdf:ID="FacultyMember"/>
E4: <owl:Class rdf:ID="PhDStudent"/>
E5: <owl:Class rdf:about="Student">
      <rdfs:subClassOf rdf:resource="#Person"/>
    </owl:Class>
E6: <owl:Class rdf:about="PhDStudent">
      <rdfs:subClassOf rdf:resource="#Student"/>
    </owl:Class>
E7: <owl:DatatypeProperty rdf:about="familyName">
      <rdfs:range rdf:resource="xsd:string"/>
      <rdfs:domain rdf:resource="#Person"/>
    </owl:DatatypeProperty>
```

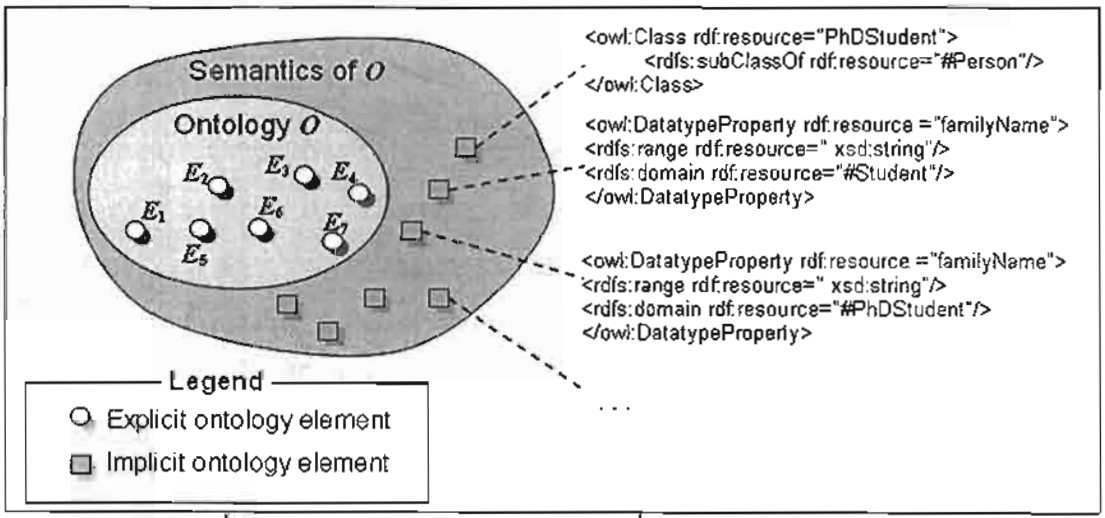
รูปที่ 5.2: ตัวอย่างออนโทโลยี O

```
C1: <owl:Class rdf:about=$S:classA>                    // คลอส C1 อธิบายว่า ถ้า $S:classA เป็น
      <rdfs:subClassOf rdf:resource=$S:classC/>        // subClassOf ของ $S:classB และ
    </owl:Class>                                       // $S:classB เป็น subClassOf ของ
        ← <owl:Class rdf:about=$S:classA>              // $S:classC แล้ว ดังนั้น $S:classA จะ
            <rdfs:subClassOf rdf:resource=$S:classB/>  // เป็น subClassOf ของ $S:classC ด้วย
          </owl:Class>,
          <owl:Class rdf:about=$S:classB>
            <rdfs:subClassOf rdf:resource=$S:classC/>
          </owl:Class>.

C2: <rdf:Property rdf:about=$S:P>                      // คลอส C2 ระบุว่า ถ้า $S:P เป็น
        <rdfs:domain rdf:resource=$S:classA/>          // DataTypeProperty แล้ว $S:P ย่อมเป็น
        <rdfs:range rdf:resource=$S:range/>            // Property ด้วย
    </rdf:Property>
        ← <owl:DataTypeProperty rdf:about=$S:P>
            <rdfs:domain rdf:resource=$S:classA/>
            <rdfs:range rdf:resource=$S:range/>
          </owl:DataTypeProperty>.

C3: <owl:DatatypeProperty rdf:about=$S:P>              // คลอส C3 ระบุว่า ถ้า $S:classA เป็น
        <rdfs:domain rdf:resource=$S:classA/>          // subClassOf ของ $S:classB และ
        <rdfs:range rdf:resource=$S:range/>            // $S:classB มี $S:P เป็น
    </owl:DatatypeProperty>                            // DataTypeProperty แล้ว $S:classA มี
        ← <owl:Class rdf:about=$S:classA>              // $S:P เป็น DataTypeProperty ด้วย
            <rdfs:subClassOf rdf:resource=$S:classB/>  // เช่นกัน นั่นคือ $S:classA ได้รับถ่ายทอด
          </owl:Class>,                                // คุณสมบัติ $S:P จาก $S:classB
          <owl:DataTypeProperty rdf:resource=$S:P>
          <rdfs:domain rdf:resource=$S:classB/>
            <rdfs:range rdf:resource=$S:range/>
          </owl:DataTypeProperty>.
```

รูปที่ 5.3: ตัวอย่างการกำหนดความหมายของคำสำคัญในการออกแบบออนโทโลยี ในรูปของกฎในทฤษฎี XDD



รูปที่ 5.4: ความหมายของออนโทโลยี *O* เมื่อกำหนดความหมายของคำสำคัญในการออกแบบออนโทโลยี (OWL/RDF(S) modeling construct)

จากวิธีการดังที่ยกตัวอย่างนี้ จึงสามารถโมเดลฐานข้อมูลออนโทโลยี *ODB* ได้ด้วยการพรรณนาด้วยทฤษฎี XDD (XDD description) ซึ่งประกอบด้วยส่วนประกอบที่สำคัญ 2 ส่วนคือ

- ODB_C: เซ็ตของออนโทโลยี $O_1, \ldots, O_n$ ซึ่งบรรยายโดยภาษา OWL
- ODB_A: เซ็ตของกฎในทฤษฎี XDD ซึ่งอธิบายความหมายของแต่ละคำสำคัญในการออกแบบออนโทโลยี (ontology modeling construct)

ดังนั้น ความหมายของฐานข้อมูลออนโทโลยี *ODB* หรือ $\mathcal{M}(ODB)$ จึงประกอบด้วย เซ็ตของออนโทโลยีที่บรรยายอย่างชัดแจ้งใน ODB_C และข้อมูลหรือความหมายที่สามารถอนุมานได้โดยอาศัยกฎใน ODB_A

5.3 การสืบค้นที่มีความหมาย

นิยามที่ 1 คำสืบค้นที่มีความหมาย (semantic query) สามารถกำหนดได้ด้วยภาษา XML ซึ่งมีรูปแบบดังนี้

```
<sq:Query>
  <sq:mandatoryConditions> m1 m2 ... mn </sq:mandatoryConditions>
  <sq:optionalConditions> o1 o2 ... or </sq:optionalConditions>
  <sq:semanticLexiconReference> url </sq:semanticLexiconReference>
  <sq:similarityWeightFactors>
    <sq:mandatoryConditionWeight> wM </sq:mandatoryConditionWeight>
    <sq:stringMatching> w≈ </sq:stringMatching>
    <sq:synonymRelation> w= </sq:synonymRelation>
    <sq:moreGeneralRelation> w⊇ </sq:moreGeneralRelation>
    <sq:lessGeneralRelation> w⊆ </sq:lessGeneralRelation>
    <sq:unknownRelation> w* </sq:unknownRelation>
  </sq:similarityWeightFactors>
</sq:Query>
```

โดยที่

- m_i คือ ข้อความหรือ element ในภาษา OWL/RDF(S) ซึ่งอธิบายเงื่อนไขที่จำเป็น (mandatory) ในการสืบค้น
- o_i คือ ข้อความหรือ element ในภาษา OWL/RDF(S) ซึ่งอธิบายเงื่อนไขที่เป็นทางเลือก (optional) ในการสืบค้น
- *url* อ้างถึง URL ของฐานข้อมูลคำศัพท์และความสัมพันธ์ระหว่างคำศัพท์
- $w_M, w_{\approx}, w_{=}, w_{\supseteq}, w_{\subseteq}, w_{*}$ มีค่าระหว่าง 0 ถึง 1 ซึ่งกำหนดค่าน้ำหนักของความสัมพันธ์
- ข้อความ semanticLexiconReference และ similarityWeightFactors เป็นทางเลือก

สำหรับคำสืบค้นที่มีความหมาย Q หนึ่งๆ ให้ $mcond(Q)$ และ $ucond(Q)$ หมายถึง เซ็ตของเงื่อนไขที่จำเป็น และเงื่อนไขทางเลือกทั้งหมดใน Q ตามลำดับ ❑

ตัวอย่างที่ 2 (คำสืบค้นที่มีความหมาย) รูปที่ 5.5 แสดงตัวอย่างการกำหนดคำสืบค้นที่มีความหมาย Q ซึ่งประกอบด้วยเงื่อนไขที่จำเป็น 3 เงื่อนไข คือ $m_1 - m_3$ และเงื่อนไขที่เป็นทางเลือก 4 เงื่อนไข คือ $o_1 - o_4$ ทั้งนี้เมื่อประมวลผลคำสืบค้น Q กับออนโทโลยี O โดยใช้กลไกของ SQORE ซึ่งนำความหมายของคำสำคัญในการออกแบบออนโทโลยี (OWL/RDF(S) modeling construct) และฐานข้อมูลคำศัพท์และความสัมพันธ์ระหว่างคำศัพท์ เช่น WordNet มาใช้แล้ว จะช่วยให้การค้นหามีความถูกต้องมากขึ้น ตัวอย่างเช่น เมื่อพิจารณาเงื่อนที่จำเป็น m_3 พบว่า เมื่ออ้างอิงถึงฐานข้อมูล WordNet ระบบ SQORE สามารถค้นพบได้ว่า คำว่า Professor นั้น มีความหมายที่แคบกว่า คำว่า FacultyMember นั่นหมายความว่า ค่าความเหมือนของคำทั้งสองนั้นเท่ากับ 0.4 (กำหนดโดยน้ำหนัก $w_{\subset}$) นอกจากนี้จากการนำความสามารถในการอนุมานถึงความหมายโดยนัยของออนโทโลยี O ช่วยให้สามารถนิรนัยได้ว่า โดเมนของคุณสมบัติ familyName นั้นครอบคลุมถึงคลาส Student และ คลาส PhDStudent ซึ่งสอดคล้องกับเงื่อนไขทางเลือก o_3

❑

```
<sq:Query>
  <sq:mandatoryConditions>                    //   m1: Ontology has a class named Student
      <owl:Class rdf:ID="Student"/>           //   m2: Ontology has a class named
      <owl:Class rdf:ID="PhDstudent"/>        //   PhDstudent
      <owl:Class rdf:ID="Professor"/>         //   m3: Ontology has a class named
  </sq:mandatoryConditions>                   //   Professor

  <sq:optionalConditions>
    <owl:Class rdf:about="PhDStudent" >       //   o1: SubClassOf of Student class is
      <rdfs:subClassOf rdf:resource ="Student"/> //  PhDstudent
    </owl:Class>                              //   o2: Ontology has a DatatypeProperty
    <rdf:DatatypeProperty rdf:about="familyname"> //  named familyname
      <rdfs:domain rdf:resource ="Student"/>  //   o3: Domain of familyname property is
      <rdfs:range rdf:resource ="xsd:String"/> //   Student
    </rdf:DatatypeProperty>                   //   o4: Range of familyname property is
  </sq:optionalConditions>                    //   xsd:String

  <sq:semanticLexiconReference>               //   WordNet is a semantic lexicon reference
      http://localhost/WordNet                //
  </sq:semanticLexiconReference>              //
  <sq:similarityWeightFactors>                //   Weight factors are defined as follows:
    <sq:mandatoryConditionWeight> 0.6         //   wm=0.6,
      </sq:mandatoryConditionWeight>          //   w  =1,
    <sq:stringMatching> 1 </sq:stringMatching> //   ws=0.8,
    <sq:synonymRelation> 0.8                  //   w≥=0.6,
      </sq:synonymRelation>                   //   w⊂=0.4 and
    <sq:moreGeneralRelation> 0.6              //   w*=0.
      </sq:moreGeneralRelation>               //
    <sq:lessGeneralRelation> 0.4              //
      </sq:lessGeneralRelation>               //
    <sq:unknownRelation> 0 </sq:unknownRelation> //
  </sq:similarityWeightFactors>               //
</sq:Query>
```

รูปที่ 5.5: ตัวอย่างคำสืบค้นที่มีความหมาย

5.4 ค่าความเหมือน

ให้ Σ แทนพยัญชนะสำหรับการสร้างออนโทโลยี ซึ่งประกอบด้วยเซ็ตต่างๆ ดังนี้

- C: ชื่อคลาส (เช่น Person, Student, PhDStudent)
- P: ชื่อคุณสมบัติ (เช่น firstname, lastname, supervises)
- D: ประเภทของข้อมูล (เช่น xsd:string, xsd:nonNegativeInteger)
- M: คำสำคัญในการออกแบบออนโทโลยีในภาษา RDF(S) และ OWL (เช่น rdfs:subClassOf)
- R: ข้อจำกัดสำหรับคลาส หรือคุณสมบัติ โดยมีรูปแบบการกำหนดคือ $m(a, b)$ โดยที่ $m \in M$, $a \in (C \cup P)$, $b \in (C \cup P \cup D)$ (เช่น rdfs:subClassOf(Student, Person))

นิยามที่ 2 (Element Similarity Score: SS_E) ความเหมือนระหว่าง อิลิเมนต์ x และ y สามารถวัดได้โดย

- สำหรับ $x,y \in C$ หรือ $x,y \in P$ หรือ $x,y \in D$:

$$SS_E(x, y) = \begin{cases} w_{=} & : \text{ถ้า } x = y \\ w_{\equiv} & : \text{ถ้า } x \equiv y \\ w_{\supseteq} & : \text{ถ้า } x \supseteq y \\ w_{\subseteq} & : \text{ถ้า } x \subseteq y \\ w_{\neq} & : \text{otherwise} \end{cases}$$

- สำหรับ $x = m(a_1,b_1) \in R$ and $y = m(a_2,b_2) \in R$:

$$SS_E(x,y) = SS_E(a_1,a_2) * SS_E(b_1,b_2)$$

- กรณีอื่น:

$$SS_E(x,y) = \text{unknown}$$

□

จากนิยามที่ 2 นี้ การวัดค่าความเหมือนระหว่าง element x ใน Σ และออนโทโลยี O ทำได้โดยหาค่าความเหมือนที่สูงที่สุดระหว่าง element x และทุก element y ในออนโทโลยี O แนวคิดนี้เองได้นำไปสู่นิยามที่ 3 ดังนี้

นิยามที่ 3 (Best Element Similarity Score: SS_B) ความเหมือนระหว่าง อิลิเมนต์ x และออนโทโลยี O สามารถวัดได้โดย

$$SS_B(x,O) = \max_{y \in M(O)} S(x,y)$$

□

ในลำดับถัดไป เป็นการนำเสนอค่าความพอใจสำหรับเงื่อนไขที่จำเป็นในการสืบค้น (SS_M) และค่าความพอใจสำหรับเงื่อนไขที่เป็นทางเลือกในการสืบค้น (SS_O)

นิยามที่ 4 (Satisfaction Scores of Mandatory conditions: SS_M **และ** Optional conditions: SS_O) สำหรับออนโทโลยี O หนึ่งๆ ค่าความพอใจของคำสืบค้นที่มีความหมาย Q สำหรับเงื่อนไขที่จำเป็นในการสืบค้น (SS_M) และสำหรับเงื่อนไขที่เป็นทางเลือก (SS_O) สามารถวัดได้โดย

$$SS_M(Q,O) = \begin{cases} 1 & : \text{if } mcond(Q) = \varnothing \\ 0 & : \text{if } \exists(c \in mcond(Q))\ SS_B(c,O) \le w_* \\ \dfrac{\sum_{c \in mcond(Q)} SS_B(c,O)}{|mcond(Q)|} & : \text{otherwise} \end{cases}$$

$$SS_O(Q,O) = \begin{cases} 1 & : \text{if } ocond(Q) = \varnothing \\ 0 & : \text{if } SS_M(Q,O) = 0 \\ \dfrac{\sum_{c \in ocond(Q)} SS_B(c,O)}{|ocond(Q)|} & : \text{otherwise} \end{cases}$$

❑

จากนิยามที่ 4 นี้ เมื่อนำค่าความพอใจของคำสืบค้นที่มีความหมาย Q สำหรับเงื่อนไขที่จำเป็นในการสืบค้น (SS_M) และสำหรับเงื่อนไขที่เป็นทางเลือก (SS_O) มาผนวกเข้าด้วยกัน ทำให้สามารถวัดค่าความเหมือนระหว่างคำสืบค้นที่มีความหมาย Q และออนโทโลยี O ได้ดังนี้

นิยามที่ 4 (Query-Ontology Similarity Score: SS) ความเหมือนระหว่างคำสืบค้นที่มีความหมาย Q และออนโทโลยี O สามารถวัดได้โดย

$$SS(Q,O) = w_M SS_M(mcond_Q,O) + (1 - w_M) SS_O(ocond_Q,O)$$

❑

ตัวอย่างที่ 3 (การคำนวณค่าความเหมือน) อ้างอิงถึงคำสืบค้นที่มีความหมาย Q ที่กำหนดดังรูปที่ 5.5 รูปที่ 5.6 แสดงตัวอย่างการคำนวณค่าความเหมือนโดยเทียบกับออนโทโลยี A, B และ C ทั้งนี้จากการพิจารณาเงื่อนไขที่จำเป็นของ Q ซึ่งต้องการค้นหาออนโทโลยีซึ่งประกอบด้วย คลาส Student, PhDStudent และ คลาส Professor นั้น พบว่าออนโทโลยี A สอดคล้องกับเงื่อนไขดังกล่าว นอกจากนี้ เนื่องจากออนโทโลยี C ประกอบด้วยคลาส Faculty_member ซึ่งเป็นคำที่มีความหมายกว้างกว่า Professor ดังนั้นออนโทโลยี C จึงสอดคล้องกับเงื่อนไขดังกล่าวด้วยเช่นกัน อย่างไรก็ตาม เนื่องจากออนโทโลยี B ไม่มีคลาสซึ่งมีความสัมพันธ์กับ คลาส Professor และ คลาส PhDStudent

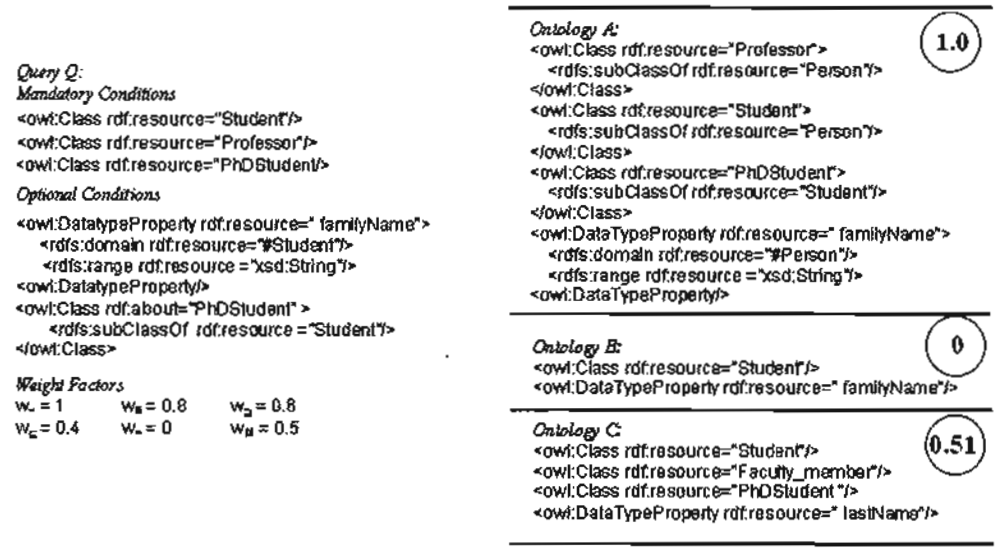
ออนโทโลยี B จึงไม่สามารถเป็นคำตอบของคำสืบค้น Q ได้ นั่นคือ ค่าความเหมือนของออนโทโลยี B มีค่าเทากับ 0

เมื่อพิจารณาเงื่อนไขที่เป็นทางเลือกของ Q ซึ่งกำหนดให้ DatatypeProperty familyName มีโดเมนเป็นคลาส Student และเรนจ์เป็น String พบว่าจากการใช้กลไกการใช้เหตุผล พบว่า ออนโทโลยี A สอดคล้องกับเงื่อนไขดังกล่าว เนื่องจากว่า คุณสมบัติ familyName ในออนโทโลยี A นั้นมีโดเมนเป็นคลาส Person โดยที่ คลาส Student เป็น subClassOf ของคลาส Person นั่นหมายความว่า คุณสมบัติ familyName ในออนโทโลยี A มีโดเมนเป็นคลาส Student ด้วยเช่นกัน และเมื่อพิจารณาออนโทโลยี C พบว่า มีการกำหนด DatatypeProperty lastName ซึ่งเป็นคำเหมือนกับ familyName จึงทำให้ออนโทโลยี C มีความสอดคล้องกับเงื่อนไขดังกล่าวแบบบางส่วน

เงื่อนไขทางเลือกอีกเงื่อนไขหนึ่งใน Q ระบุว่า คลาส PhDStudent ต้องเป็น subClassOf คลาส Student ซึ่งมีเพียงออนโทโลยี A เท่านั้นที่สอดคล้องกับเงื่อนไขนี้

กล่าวโดยสรุปผลลัพธ์ของคำสืบค้นที่มีความหมาย Q ด้วยระบบ SQORE ประกอบด้วยออนโทโลยี A และ C ด้วยค่าความเหมือน 1.0 และ 0.51 ตามลำดับ เนื่องจากออนโทโลยี A มีความหมายที่สอดคล้องกับเงื่อนไขที่จำเป็นและเงื่อนไขทางเลือกของ Q ทั้งหมด ในขณะที่ออนโทโลยี C มีความหมายสอดคล้องกับเงื่อนไขที่จำเป็นทั้งหมด แต่ไม่สอดคล้องกับบางเงื่อนไขที่เป็นทางเลือก

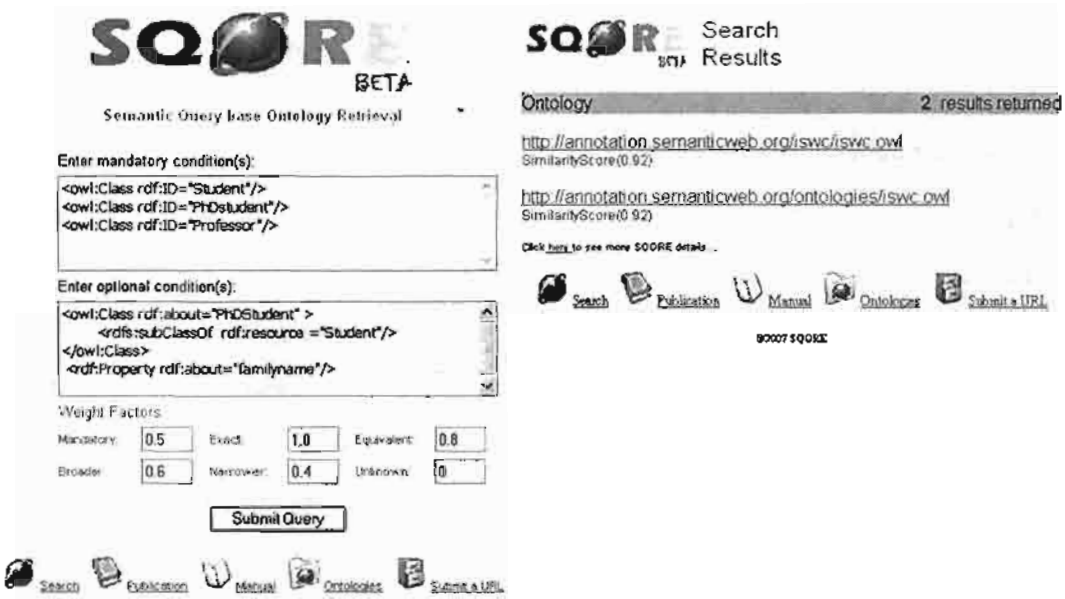
❑



รูปที่ 5.6: ตัวอย่างการคำนวณค่าความเหมือน

5.5 ระบบจำลอง

จากการพัฒนาสถาปัตยกรรม และกลไกการจัดการเอกสาร XML สำหรับเก็บความรู้ประเภทออนโทโลยีดังที่ได้นำเสนอในบทนี้ คณะผู้วิจัยได้พัฒนาระบบจำลอง SQORE ขึ้น ดังรูปที่ 5.7 ซึ่งสามารถทดลองใช้ได้ที่ http://ict.shinawatra.ac.th:8080/sqore



รูปที่ 5.7: ระบบจำลอง SQORE

6 บทวิจารณ์

งานวิจัยนี้ได้ศึกษาและวิเคราะห์ถึงหน้าที่ความต้องการ และความสัมพันธ์ระหว่างการควบคุมจัดการการเปลี่ยนแปลงรุ่นของเอกสาร XML (version control of XML documents) และการควบคุมการเข้าถึงข้อมูลเอกสาร XML (access control of XML documents) โดยพัฒนาระบบควบคุมการเปลี่ยนแปลงรุ่นและการเข้าถึงข้อมูลเอกสาร XML (ระบบ XAV-Controller: XML Document Access and Version Controller) ซึ่งได้รวมหน้าที่ที่สำคัญทั้งสองเข้าด้วยกัน และนำมาตรฐาน RDF (Resource Description Framework) มาใช้ในการอธิบาย กฎ/สิทธิในการเข้าถึงข้อมูลเอกสาร XML นโยบายการแก้ไขความขัดแย้ง และนโยบายการเข้าถึงข้อมูลโดยปริยาย รวมถึงการอธิบายข้อมูลการเปลี่ยนแปลงรุ่นของเอกสาร XML นอกจากนี้ระบบต้นแบบที่ได้พัฒนาขึ้นนี้ยังได้นำทฤษฎี RDF Declarative Description (RDD) มาประยุกต์ใช้เพื่อบรรยายความสัมพันธ์ระหว่างการเปลี่ยนแปลงรุ่นของเอกสารและการเปลี่ยนแปลงกฎการเข้าถึงเอกสารนั้นๆ

นอกจากนี้ งานวิจัยนี้ยังได้พัฒนาระบบ SQORE (Semantic Query based Ontology Retrieval) ขึ้นเพื่อจัดการเอกสาร XML สำหรับเก็บความรู้ประเภทออนโทโลยี โดยอนุญาตให้ผู้ใช้กำหนดคำสืบค้นที่มีความหมาย ซึ่งอธิบายรายละเอียดของออนโทโลยีที่ต้องการมายังระบบ จากนั้นจึงนำกลไกการอนุมานและการใช้เหตุผลมาใช้ เพื่อให้การสืบค้นมีประสิทธิภาพและมีความถูกต้องมากยิ่งขึ้น

สำหรับงานวิจัยต่อยอดจากผลงานที่นำเสนอดังกล่าวนี้ ประกอบด้วย การพัฒนาเทคนิคและวิธีการในการกำหนดสิทธิและควบคุมการเข้าถึงข้อมูลเอกสาร XML ที่ใช้ออนโทโลยีในการอธิบายความหมายข้อมูล เนื่องจากออนโทโลยีเป็นวิธีการบรรยายความรู้ ซึ่งอนุญาตให้นิรนัยข้อมูลใหม่จากความรู้และข้อมูลที่มีอยู่ ดังนั้นเทคนิคและวิธีการในการควบคุมการเข้าถึงข้อมูลเอกสาร XML โดยทั่วไป จึงไม่สามารถนำมาประยุกต์ใช้ได้โดยตรง จำเป็นต้องคำนึงถึงความสามารถในการนิรนัยนี้ด้วย เพื่อให้การควบคุมการเข้าถึงข้อมูลมีความปลอดภัยมากที่สุด

หนังสืออ้างอิง

1. Chutiporn Anutariya, Somchai Chatvichienchai, Mizuho Iwaihara, Vilas Wuwongse, Yahiko Kambayashi. A Rule-Based XML Access Control Model. *Lecture Notes in Computer Science*, Vol. 2876, pp. 35-48, October 2003.
2. Chutiporn Anutariya, Rachanee Ungrangsi, Vilas Wuwongse. SQORE – Semantic Query base Ontology Retrieval. *In Proc. 12th Int. Conf. Database Systems for Advanced Applications (DASFAA07)*, LNCS 4443, pp. 924-929, Bangkok, April 2007.
3. Chutiporn Anutariya, Vilas Wuwongse, and Kiyoshi Akama. XML Declarative Description with First-Order Logical Constraints. *Computational Intelligence*, Vol. 21, No. 2, pp. 130-156, 2005.
4. Chutiporn Anutariya, Vilas Wuwongse, and Vichit Wattanapailin. An Equivalent-Transformation-Based XML Rule Language. *Proc. Int'l Workshop on Rule Markup Languages for Business Rules in the Semantic Web* (CEUR Workshop Proc.), Sardinia, Italy, Vol. 60, 2002.
5. S. Bechhofer et al. OWL Web Ontology Language Reference. *W3C Recommendation*, 10 February 2004. http://www.w3.org/TR/owl-ref/
6. Tim Berners-Lee, James Handler, and Ora Lassila. The Semantic Web, *Scientific American*, May 2001.
7. Elisa Bertino, Silvana Castano, Elena Ferrari and March Mesiti. Specifying and Enforcing Access Control Policies for XML Document Sources. *World Wide Web Journal*, 3, 3, Naltzer Science Publishers, 2000.
8. Elisa Bertino, M. Braun, Silvana Castano, Elena Ferrari, and March Mesiti. Author-X: a Javabased system for XML data protection. *Proceedings 14th IFIP WG 11.3 Working Conference on Database Security*, Netherlands, August 2000.
9. Elisa Bertino, Silvana Castano, and Elena Ferrari. On Specifying Security Policies for Web Documents with an XML-based Language Proc. 6th ACM Symposium on Access control models and technologies, ACM Press, 2001.

10. Dan Brickley, and R.V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. *W3C Recommendation*, 10 February 2004. http://www.w3.org/TR/rdf-schema/
11. David Booth, and Canyang Kevin Liu. Web Services Description Language (WSDL) Version 2.0 Part 0: Primer. *W3C Proposed Recommendation*, 23 May 2007 http://www.w3.org/TR/2007/PR-wsdl20-primer-20070523/
12. Tim Bray et al. Extensible Markup Language (XML) 1.0 (Fourth Edition). *W3C Recommendation*, 16 August 2006. http://www.w3.org/TR/2006/REC-xml-20060816/
13. Shu-Yao Chien, Vassilis J. Tsotras and Carlo Zaniolo. Version Management of XML Documents. *Proceedings of World Wide Web and Databases (WebDB2000), Lecture Notes in Computer Science*, Vol. 1997, Springer Verlag, 2001.
14. Shu-Yao Chien, Vassilis J. Tsotras and Carlo Zaniolo. Efficient Management of Multiversion Documents by Object Referencing. *Proceedings of International Conference on Very Large Databases (VLDB) 2001.*
15. Shu-Yao Chien, Vassilis J. Tsotras, Carlo Zaniolo and Donghui Zhang. Storing and Querying Multiversion XML Documents using Durable Node Numbers. *Proceedings of 2nd International Conference on Web Information Systems Engineering (WISE)*, Kyoto, Japan, 2001.
16. Ernesto Damiani, Sabrina De Capitani di Vimercati, Stefano Paraboschi, Pierangela Samarati, XML Access Control Systems. A Component-Based Approach. *Proceedings IFIP WG11.3 Working Conference on Database Security*, Netherlands, August 21-23, 2000.
17. Ernesto Damiani, Sabrina De Capitani di Vimercati, Stefano Paraboschi, and Pierangela Samarati. Design and implementation of an access control processor for XML documents. *WWW9 / Computer Networks*, 33(1-6):59–75, 2000.
18. Ernesto Damiani, Sabrina De Capitani di Vimercati, Stefano Paraboschi and Pierangela Samarati. A Fine-Grained Access Control System for XML Documents. *ACM Transaction on Information and System Security*, Vol. 5, No. 2, (May 2002) 169–202

19. L. Ding, T. Finin, A. Joshi, R. Pan, R. Scott Cost, Y. Peng, P. Reddivari, V. Doshi, J. Sachs. Swoogle: A Search and Metadata Engine for the Semantic Web. *Proc. 13 ACM Int'l Conf. Information and Knowledge Management*, DC, November 08-13, 2004.
20. Brian Eisenberg, and Duane Nickull. ebXML Technical Architecture Specification v1.0.4. *Final Draft*, 16 February 2001. http://www.ebxml.org/specs/ebTA.pdf
21. S. Godik, and T. Moses. XACML 1.0. *OASIS Standard*, Feb. 2003. http://www.oasis-open.org/committees/download.php/2406/oasis-xacml-1.0.pdf
22. Satoshi Hada and Michiharu Kudo. XML Access Control Language: Provisional Authorization for XML Documents. *IBM Technical Paper*, Oct. 2000. http://www.trl.ibm.com/projects/xml/xacl/xacl-spec.html
23. G. Klyne and J. J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. *W3C Recommendation*, 10 February 2004. http://www.w3.org/TR/rdf-concepts/
24. Nicole Lam and Raymond K. Wong. Efficient Content-Based Version Management for XML Data. 2002.
25. A. Miller. Wordnet: A lexical database for English. *Communications of the ACM*, number 38(11), 1995.
26. C. Patel, K. Supekar, Y. Lee, E. K. Park. OntoKhoj: a semantic web portal for ontology searching, ranking and classification, *Proc. 5th ACM Int'l Workshop on Web Information and Data Management*, Louisiana, November 07-08, 2003.
27. Marc J. Rochkind. The Source Code Control System. *IEEE Transactions on Software Engineering*, SE-1, 4, Dec. 1975.
28. E. Thomas, H. Alani, D. Sleeman, and C. Brewster. Searching and Ranking Ontologies on the Semantic Web. *Proc. Workshop Ontology Management: Searching, Selection, Ranking, and Segmentation*, 3rd K-CAP 2005, pp. 57-60, Banff, Canada, 2005.
29. Walter F. Tichy. RCS – A System for Version Control. *Software Practice and Experience*, 15, 7, July 1985.
30. Fusheng Wang and Carlo Zaniolo. Temporal Queries in XML Document Archives and Web Warehouses. *TIME-ICTL*, 2003.

31. Vilas Wuwongse, Chutipom Anutariya, Kiyoshi Akama and Ekawit Nantajeewarawat. XML Declarative Description (XDD): A Language for the Semantic Web. *IEEE Intelligent Systems*, Vol. 16, No. 3, pp. 54–65, May/June 2001

Output จากโครงการวิจัยที่ได้รับทุนจาก สกอ. และ สกว.

(1) Somchai Chatvichienchai, Chutiporn Anutariya[1], Mizuho Iwaihara, Vilas Wuwongse, and Yahiko Kambayashi: Towards Integration of XML Document Access and Version Control. *Lecture Notes in Computer Science*, Vol. 3180, ISSN 0302-9743, Springer Verlag, pp. 791-800 (September 2004). Impact Factor 0.402

(2) Chutiporn Anutariya, Rachanee Ungrangsi, and Vilas Wuwongse: SQORE: A Framework for Semantic Query Based Ontology Retrieval. *Lecture Notes in Computer Science, Vol. 4443, ISSN 0302-9743, Springer Verlag, pp. 924 – 929* (April 2007). Impact Factor 0.402

[1] Correspondent author

ภาคผนวก

(1) ผลงานวิจัยที่ตีพิมพ์ในวารสารวิชาการระดับนานาชาติ

Somchai Chatvichienchai, Chutiporn Anutariya[†], Mizuho Iwaihara, Vilas Wuwongse, and Yahiko Kambayashi: Towards Integration of XML Document Access and Version Control. *Lecture Notes in Computer Science*, Vol. 3180, ISSN 0302-9743, Springer Verlag, pp. 791-800 (September 2004). Impact Factor 0.402

(2) ผลงานวิจัยที่ตีพิมพ์ในวารสารวิชาการระดับนานาชาติ

Chutiporn Anutariya, Rachanee Ungrangsi, and Vilas Wuwongse: SQORE: A Framework for Semantic Query Based Ontology Retrieval. *Lecture Notes in Computer Science*, Vol. 4443, ISSN 0302-9743, Springer Verlag, pp. 924 – 929 (April 2007). Impact Factor 0.402

(3) การนำเสนอผลงานในการประชุมทางวิชาการระดับนานาชาติ

Somchai Chatvichienchai, Chutiporn Anutariya, Mizuho Iwaihara, Vilas Wuwongse, and Yahiko Kambayashi: Towards Integration of XML Document Access and Version Control. *The 15th International Conference on Database and Expert Systems Applications (DEXA 2004)*, Zaragoza, Spain, September 2004.

(4) การนำเสนอผลงานในการประชุมทางวิชาการระดับนานาชาติ

Chutiporn Anutariya, Rachanee Ungrangsi, and Vilas Wuwongse: SQORE: A Framework for Semantic Query Based Ontology Retrieval. *The 12th International Conference on Database Systems for Advanced Applications (DASFAA 2007)*, Bangkok, Thailand, April 2007.

[†] Correspondent author

(1) ผลงานวิจัยที่ตีพิมพ์ในวารสารวิชาการระดับนานาชาติ

Somchai Chatvichienchai, Chutiporn Anutariya*, Mizuho Iwaihara, Vilas Wuwongse, and Yahiko Kambayashi: Towards Integration of XML Document Access and Version Control. *Lecture Notes in Computer Science*, Vol. 3180, ISSN 0302-9743, Springer Verlag, pp. 791-800 (September 2004). Impact Factor 0.402

Towards Integration of XML Document Access and Version Control

Somchai Chatvichienchai[1], Chutiporn Anutariya[2], Mizuho Iwaihara[3], Vilas Wuwongse[4], Yahiko Kambayashi*

[1] Department of Info-Media, Faculty of Global Communication, Siebold University of Nagasaki, Nagasaki, Japan
[2] Computer Science Program, Shinawatra University, Pathumthani, Thailand
[3] Department of Social Informatics, Graduate School of Informatics, Kyoto University, Kyoto, Japan
[4] Computer Science and Information Management Program, Asian Institute of Technology, Pathumthani, Thailand
[1] somchaic@sun.ac.jp
[2] chutiporn@shinawatra.ac.th
[3] iwaihara@i.kyoto-u.ac.jp
[4] vw@cs.ait.ac.th

Abstract. Due to an increasing popularity of employing XML documents in various application domains, there arises a demand for an efficient XML document database management. Thus, development of effective mechanisms for manipulating access and version control has become a major research area. However, well-developed access and version control mechanisms alone do not suffice because they are closely interrelated. An update to a document (resulting in a new version of that document) may have a side effect to its associated access control policy, and vice versa. Thus, different document versions can be assigned different access authorizations. This paper gives an insight into the problems of access and version control of XML documents and proposes an approach to overcoming them by integrating these two essential mechanisms into a single framework. It employs RDF as a unified representation of access control policies and version information, and formalizes their interrelationships by means of *RDF Declarative Description Theory*.

1 Introduction

With an increasing popularity of employing XML documents in a variety of applications, a demand for efficient XML document database management has hence arisen. Since a single document may be edited and accessed concurrently by many different users, development of effective mechanisms for manipulating versions and access control has become a major research area. Version control keeps track of changes of evolving XML documents and maintains appropriate version information, including the identities of users who modify the documents, date/time and modification details;

* Prof. Kambayashi passed away on February 6th, 2004.

hence enabling the documents to be reverted back to their previous versions. Access control, on the other hand, deals with granting or denying particular accesses to the specified documents, mostly to their latest versions. The development of suitable authorization for XML documents poses new protection requirements with respect to conventional object-oriented databases [11]. These new requirements arise because of the richer structure of XML documents with respect to HTML documents and the possibility of attaching a DTD to XML documents, which has similarity with the notion of type in the object-oriented context.

However, well-developed access and version control mechanisms alone do not suffice because they are closely interrelated. An update to a document (resulting in a new version of that document) may lead to a change of the document's access policy. Thus, different document versions can be assigned different access authorizations. For instance, insertion of certain confidential data may require addition of new access policies for the inserted data. To accommodate such requirements, a management framework with fine-grained access control mechanisms on multi-version XML documents is demanded. This framework will be useful to various applications. However, to the best of our knowledge, there does not exist such a framework

This paper not only analyzes essential features and requirements of both access and version control mechanisms but also explores in depth their interrelationships. Based on the analysis, an approach to integrating them into a single framework is developed.

Section 2 gives insight into access and version control problems, Section 3 discusses their interrelationships and points out the benefits gained from their integration, Section 4 proposes an integrated framework by employment of *RDF* [12] and *RDF Declarative Description (RDD)* theory [1], and Section 5 concludes with a summary and a list of future work.

2 Access and Version Control Problems

2.1 Access Control Problems

Various XML access control models [2,4,15] have been proposed. These models enforce access restrictions directly on the structure and content of XML documents. In this way, information in XML format can be protected at a finer level of granularity (e.g., the element level) rather than the whole document. Each document is associated with a set of authorizations specifying access rights of users on information within the document. By employment of an RDF rule-based declarative language, the approach proposed in [2] has incorporated facilities which allow derivation of implicit, complex authorizations on the basis of other authorizations.

The structure of XML documents changes by various reasons (such as information exchange between organizations that use different document formats). When a document is transformed to a new format (schema), the associated authorizations must also be translated for the transformed document. Algorithms for translating authorizations of source documents into those of transformed documents by using document-node and schema-node mapping have been proposed in [5,6].

2.2 Version Control Problems

Version management of XML documents has recently been discussed in [13,7]. A change-centric method proposed in [13] stores the latest version of a document and the sequence of forward completed deltas. The *Reference-Based Version Model* (*RBVM*) [7] focuses on the storage performance of multi-version documents.

Let d_i and d_j be documents of versions i and j, respectively. Assume that d_j is established by applying a number of changes (insertions, deletions or updates) to d_i. In a typical *RCS* (*Revision Control System*) scheme [16], these changes are stored in a script denoted by *Delta* Δd_{ij}. Such a script could be generated directly from the edit commands of a structured editor or obtained by applying the pair (d_i, d_j) to a structured diff package, such as XyDiff [9] and Microsoft XML Diff [14].

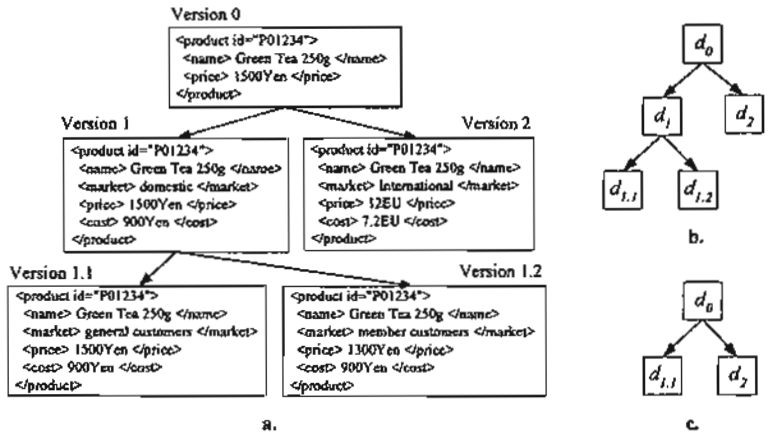


Fig. 1. Version evolution of a product catalog "catalog.xml" and its corresponding version trees.

2.3 Integrating Access and Version Control Requirements

The problems that arise when both access and version controls are not integrated into XML document database management will be now discussed. As a document changes, some information of the changed document should not be disclosed to some users.

As an example, consider Fig. 1-a which illustrates version evolution of a product catalog, and Fig. 1-b which depicts its corresponding *version tree*[1], where d_0, d_1, $d_{1.1}$, $d_{1.2}$, and d_2 denote product catalogs of versions 0, 1, 1.1, 1.2 and 2, respectively. Firstly, d_0 is evolved into d_1 and d_2 for domestic market and international market, respectively, and d_1 is next evolved into $d_{1.1}$ and $d_{1.2}$ for general customers and member customers, respectively. After d_1 is evolved into $d_{1.1}$ and $d_{1.2}$, general customers who are allowed to access d_1 and $d_{1.1}$ should not be allowed to access $d_{1.2}$ which is only accessible by member customers. A lack of proper access control over docu-

[1] For simplicity, version trees are employed but they can be readily generalized to *version graphs*.

ments of new versions results in confidential information disclosure problem. Furthermore, a user may have different access authorizations on different document versions. For instance, a general customer is allowed to read the entire d_0 but not allowed to read some part (e.g., *cost*) of $d_{1.1}$ and the whole part of $d_{1.2}$. Thus, fine-grained access control is essential for defining access rights of users to information inside each version of a document. Assuming that a member customer is not allowed to access d_1, $d_{1.2}$, a version tree of this customer is depicted in Fig.1-c.

On the other hand, a lack of multi-version concept makes schema design and access authorization policy of a document become very complex and error-prone. For instance, integration of $d_{1.1}$, $d_{1.2}$ and d_2 into a single document requires redefining price elements for all types of markets. In addition, since the schema of a combined single-version document tends to be more complicated, authorization control policy of each user on a database of such documents will be more complex. Integration of access and version control can simplify schema and access control policy definitions.

3 Integrating Access and Version Control

The previous section has analyzed the strong interrelationships between access and version control. Thus, their integration not only solves the outlined problems, but also results in a unified framework with facilities to handle these two mechanisms effectively and simultaneously. It allows one to simply and straightforwardly model the side effect of document versions towards access control policies, and vice versa. Moreover, because version information and access control policies themselves can be represented as XML documents, the framework then provides a complete and uniform solution to the manipulation of schemas, version information and access control policies of XML documents. Issues of concern, when developing such an integrated framework are as follows:

- *Inference Problems*: As a rule, unauthorized users should not be aware of the existence of confidential information contained in a document. Hence, proper mechanisms that prevent users from inferring contents and/or structures of protected data should be implemented.
- *Version History*: Public view of version history and version tree could let users yield some conclusions about protected data. Thus, it is preferable to represent version information as an XML document so that fine-grained access control over such version information can be simply defined. Moreover, the use of a number sequence as a version ID may let users deduce the existence of some versions that they are not allowed to access, hence one could simply use a document's timestamp as its version ID.
- *Version Control over Schemas*: Version control mechanisms could be used to manage not only versions of XML documents but also their schemas. To avoid consistency and integrity problems, all versions of a document in a version tree should conform to the same version of schema. That is, when a schema of a document is modified, a new version tree should be created.
- *Temporal Features*: The association of *timestamps* (or *temporal data*) with various attributes of version information, e.g., editing and valid time, is fun-

damental to version management. They play an important role in sophisticated access control models by allowing definition of policies with certain conditions on timestamp attributes, e.g., a policy which states that all documents will be disclosed to public after ten years from the date of document creation. Broad sets of queries related to temporal information are also enabled.

- *Improper Disclosure of Confidential Data*: When a modification of a document results in a change of access policies, the framework should have a warning mechanism to ensure that it does not lead to improper disclosure of confidential data, which usually happens when an update by a particular user inadvertently gives that user an authorization to access protected data.

4 Proposed Framework

The developed framework employs *RDF* [12] as a unified representation of access control policies and version information, and formalizes their interrelationships by means of *RDF Declarative Description (RDD) Theory* [1].

4.1 Authorization Model

Managing access to a protected document is expressed in terms of *access control policies* [2]. A policy is simply described by a set of authorization rules, a conflict resolution and a default policy. An authorization specifies for each user (or group of users) the types of accesses that the user can/cannot exercise on each object. An authorization either positive (granting access) or negative (denying access) on an element can be propagated to all of its sub-elements (by *cascade* option), or to only its attributes (by *no_propagation* option). The *read* privilege allows a subject to view an element and its attributes. The *write* privilege allows a subject to update and delete information in an element. A *priority* is an integer value which specifies the priority level of an authorization. An authorization with a higher priority can prevail over lower ones. The default value of a priority is zero.

The possibility of specifying both positive and negative authorizations introduces potential conflicts among authorizations with the same priority levels. Hence, a conflict resolution policy must be specified. Examples of conflict resolution policies include *instance-*, *descendant-*, *denial-* and *grant-take-precedence*. In addition, there might exist an element without an associated authorization, either defined explicitly or propagated along the document structure by means of the cascade option. In such a case, a default authorization could be applied. Two main types of default policies are *open* and *close*. Formal definitions of policies and authorizations follow.

Definition 1 (Access Control Policy) An access control policy is simply described by a set of *authorizations*, a *conflict resolution* and *default policy*. An authorization consists of the following properties:

- *subject*: a user, a user group, a role or credentials describing properties (e.g., age, gender, domain) of the user to whom the authorization is specified;
- *target*: specifying the baseline version of a document;

- *path*: an *XPath* [8] expression identifying an element or attribute within the target document;
- *level*: possible values of which are "*#instance*" and "*#schema*" indicating whether the defined authorization is applied at an instance or schema level,
- *privilege*: possible values of which are "*#read*" and "*#write*";
- *type*: possible values of which are "*#cascade*" and "*#no_propagation*";
- *sign*: possible values of which are "*#plus*" and "*#minus*".
- *priority*: an integer value specifying the priority of the authorization. □

For more explanation of the developed RDF schema for access control policies and authorizations, please refer to [2]. Fig. 2-a and Fig. 2-b, respectively, give examples of policies p_1 and p_2 together with their corresponding authorizations. Intuitively, p_1 defines that only GeneralCustomer group is granted an access to read the entire document except the cost element, while p_2 specifies similar authorizations for the MemberCustomer group.

```
<Policy rdf:about="#p1">
    <conflictResolution
        rdf:resource="#denial_take_precedence"/>
    <defaultPolicy rdf:resource="#closed"/>
    <authorization rdf:resource="#a1"/>
    <authorization rdf:resource="#a2"/>
</Policy>
<Authorization rdf:about="#a1">
    <sign rdf:resource="#plus"/>
    <subject rdf:resource="#GeneralCustomer"/>
    <target>catalog.xml</target> <path> / </path>
    <privilege rdf:resource="#read"/>
    <type rdf:resource="#cascade"/>
    <sign rdf:resource="#plus"/>
</Authorization>
<Authorization rdf:about="#a2">
   <sign rdf:resource="#plus"/>
   <subject rdf:resource="#GeneralCustomer"/>
   <target>catalog.xml</target> <path>//cost</path>
   <privilege rdf:resource="#read"/>
   <type rdf:resource="#cascade"/>
   <sign rdf:resource="#minus"/>
</Authorization>
```

a. RDF description of access control policy p_1.

```
<Policy rdf:about="#p2">
    <conflictResolution
        rdf:resource="#denial_take_precedence"/>
    <defaultPolicy rdf:resource="#closed"/>
    <authorization rdf:resource="#a3"/>
    <authorization rdf:resource="#a4"/>
</Policy>
<Authorization rdf:about="#a3">
    <sign rdf:resource="#plus"/>
    <subject rdf:resource="#MemberCustomer"/>
    <target>catalog.xml</target> <path> / </path>
    <privilege rdf:resource="#read"/>
    <type rdf:resource="#cascade"/>
    <sign rdf:resource="#plus"/>
</Authorization>
<Authorization rdf:about="#a4">
   <sign rdf:resource="#plus"/>
   <subject rdf:resource="#MemberCustomer"/>
   <target>catalog.xml</target> <path>//cost</path>
   <privilege rdf:resource="#read"/>
   <type rdf:resource="#cascade"/>
   <sign rdf:resource="#minus"/>
</Authorization>
```

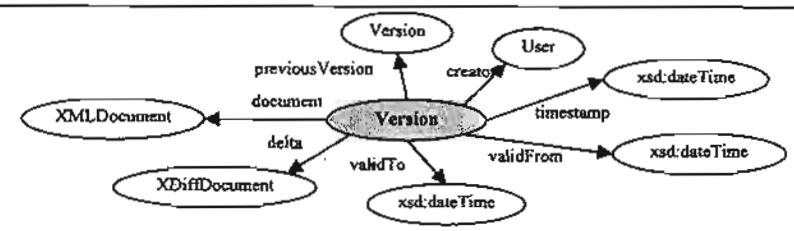
b. RDF description of access control policy p_2.

Fig. 2. Access control policy and authorization examples.

4.2 Version Control Model

As briefly described in Sect. 2.2, a *document version tree* represents relationships between versions of a document such that each version is denoted by a node in a tree and identified by its version ID and if d_j is a successive version of d_i, d_j is represented as a child node of d_i. Hence, the latest (current) versions of a document are those which are the leaf nodes of the tree. In addition, let Delta $\Delta d_{i,j}$ denote the change applied to d_i in order to obtain a successive version d_j. Thus, given d_i and $\Delta d_{i,j}$, one can reconstruct a successive version d_j by applying $\Delta d_{i,j}$ to d_i. Therefore, maintaining only the base version and Deltas is sufficient to retrieve all versions in a tree. However, since the latest versions are usually requested, the framework maintains also the whole contents of these latest versions instead of storing only Deltas. Note that Mi-

crosoft XML Diff and Patch [14] is an example of available tools which can generate $\Delta_{d_i,j}$ from the given documents d_i and d_j, and can also reconstruct d_j from d_i and $\Delta_{d_i,j}$.



a. RDF schema for version description.

```
<Version  rdf:about="#d1.2">
    <document  rdf:resource="catalog.xml"/>
    <previousVersion  rdf:resource="#d1"/>
    <creator  rdf:resource="#john"/>
    <timestamp>2004-02-01T09:00:00</timestamp>
    <validFrom>2004-03-01T00:00:00</validFrom>
    <validTo>2004-04-01T00:00:00</validTo>
    <delta  rdf:parseType ="Literal">
        <xd:xmldiff>  <xd:node  match="1">
            <xd:change  match="2">member customers</xd:change>
            <xd:change  match="3">1300Yen</xd:change>
        </xd:node>  </xd:xmldiff>
    </delta>
</Version>
```

b. RDF description of product catalog $d_{1.2}$ of Fig. 1-a.

Fig. 3. Version description schema and its example.

Definition 2 (Version) Fig. 3-a illustrates the developed RDF schema for semantic description of version information. Each version identified by a version ID (or a resource URI) is described by the following properties:

- *document*: a reference to the baseline version of a document;
- *previousVersion*: a reference to its previous version,
- *creator*: author of the described version,
- *timeStamp*: the version's timestamp,
- *validFrom*, *validTo*: representing valid time interval of the described version,
- *delta*: changes between the previous and current versions specified in XML Diff language. ❑

To prevent confidential information disclosure problem, this version control model does not permit a direct access to Deltas because they can be used to reconstruct documents of particular versions. Referring to the product catalog document of version $d_{1.2}$ of Fig. 1-a, Fig. 3-b gives a corresponding version description with $\Delta_{d_1,1.2}$ specified in delta property in XML Diff Language.

4.3 Authorization and Version Association

An RDF schema describing the association between a particular document version and an access control policy is depicted by Fig. 4-a and an example of associating the policy p_1 of Fig. 3-b with the document d_1 of Fig. 1-a is given by Fig. 4-b. However,

manually assigning an appropriate policy with each document version is, in many cases, a complex and error-prone task. The proposed framework, therefore, incorporates facilities to model interrelationships between authorization and version management and to automate such a complex process.

a. RDF schema for policy and version association.

```
<Policy_Version_Association>
    <policy  rdf:resource="#p1"/>
    <version  rdf:resource="#d1.1"/>
</Policy_Version_Association>
```

b. Example: associating policy p_1 to the document version $d_{1.1}$.

Fig. 4. Policy-version-association schema and example.

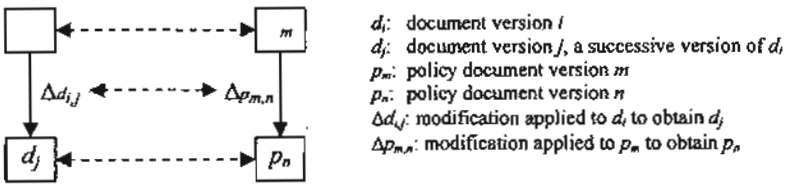


Fig. 5. Access and version control interrelationships.

First, consider the modeled interrelationships of Fig. 5. Assume that authorization of a document version d_i is regulated by a policy p_m. An update $\Delta d_{i,j}$ made to d_i resulting in a newer version d_j may lead to a change of its associated policy, i.e., accesses to d_j will then be regulated by another policy p_n. In particular, there are three possible alternatives to select an appropriate policy for d_j, depending on the type of changes occurred in $\Delta d_{i,j}$:

- A user manually defines an appropriate policy p_n for d_j;
- The system automatically assigns an existing policy p_n, or instantly generates a new policy p_n based on some predefined relationships between $\Delta d_{i,j}$ and $\Delta p_{m,n}$;
- If the above two cases are not applied, by default d_j will be assigned with the same policy as its previous version d_i.

These three methods can be formulated as *RDF clauses* [1], hence allowing the framework to determine the appropriate policy for each version based on some predefined application-specific rules.

Definition 3 (Policy-Version-Association Clause) A policy-version-association clause is represented as an *RDF clause* of the form $H \leftarrow B_1, \ldots, B_n$, where

- H is an RDF expression describing a derived policy-version-association and conforming to the schema of Fig. 4-a,

- B_j is an RDF expression or an RDF constraint restricting a certain condition on the derived association. □

As an example of a policy-version-association clause, consider again the running example of product catalog document. Suppose that in this scenario, the catalog version d_1 is the latest version and is manually associated with the policy p_1 of Fig. 2-a by a security officer. After that d_1 is evolved into $d_{1.1}$ and $d_{1.2}$ with Deltas $\Delta d_{1.1.1}$ and $\Delta d_{1.1.2}$, respectively. The description of version $d_{1.2}$ together with $\Delta d_{1.1.2}$ are given by Fig. 3-b. Since d_1 is evolved into $d_{1.2}$ by updating the value of market-element to "member customers", based on the definition of the policy-version-association clause of Fig. 6-a, the framework can automatically yield an element of Fig. 6-b. That is, the access control of document version $d_{1.2}$ is regulated by the policy p_2.

```
<Policy_Version_Association>
    <policy rdf:resource="#p2"/>
    <version rdf:resource=$S:Y/>
</Policy_Version_Association>
    ← <Version rdf:about=$S:Y>
          <document rdf:resource="catalog.xml"/>
          <previousVersion rdf:resource=$S:X/>
          <delta>
              <xd:xmldiff rdf:parseType="Literal">
                  <xd:change match=$S:anynode>
                      member customers </xd:change>
                  $E:otherchanges
              </xd:xmldiff>
          </delta>
          $E:versionProperties
      </Version>,
      <Policy_Version_Association>
          <policy rdf:resource="#p1"/>
          <version rdf:resource=$S:X/>
      </Policy_Version_Association>
```

a. Policy-version-association clause example.

```
<Policy_Version_Association>
    <policy rdf:resource="#p2"/>
    <version rdf:resource="#d1.2"/>
</Policy_Version_Association>
```

b. Derived Policy_Version_Association element.

```
// This RDF clause formalizes that
// if a document catalog.xml of
// version Y is evolved from ver-
// sion X by changing the value of
// market-element to "member
// customers" and that document
// version X is associated with
// policy p1, then policy p2 will be
// assigned to the new version Y.
//
```

Fig. 6. Policy-version-association clause.

5 Conclusions

This paper has addressed the problems that arise when access and version control of XML documents are handled separately, such as the complexity of designing a single policy for enforcement of authorization over all versions of a document. To deal with these problems, a framework which integrates both access and version control mechanisms has been developed. This unified framework has facilities to handle these two essential mechanisms effectively and simultaneously. Moreover, because access control policies and version information themselves can be represented as XML documents, the framework then provides a complete and uniform solution to easily manipulating schemas, version information and access control policies of XML document databases. Further work includes the following issues:

- incorporation of a mechanism to handle version control over document schemas;

- development of an appropriate technique to solve inference problems (i.e., a situation that allows users to infer certain information about protected data);
- in-depth analysis and refinement of the representation of temporal features to deal with temporal XML document databases and temporal query processing;
- application of the framework to XML data warehouses-repositories of data coming from several sources (e.g., web sites) in which many important issues such as security, consistency and maintenance/querying of history of or changes in document versions are interwoven; and
- implementation of the framework by *RDD's inference engine* [3].

References

1. Anutariya, C., Wuwongse, V., Akama, K. and Nantajeewarawat, E.: RDF Declarative Description: A Language for Metadata. *J. Digital Information*, Vol. 2, Issue 2 (2001).
2. Anutariya, C., Chatvichienchai, S., Iwiahara, M., Wuwongse, V. and Kambayashi, Y.: A Rule-Based XML Access Control Model. *Proc. Rules and Rule Markup Languages for the Semantic Web Workshop*, LNCS #2762, Springer Verlag, pp. 92-103 (October 2003).
3. Anutariya, C., Wuwongse, V. and Wattanapailin, V.: An Equivalent-Transformation-Based XML Rule Language. *Proc. Int'l Workshop Rule Markup Languages for Business Rules in the Semantic Web*, Sardinia, Italy (2002)
4. Bertino, E., Castano, S., Ferrari, S. and Mesiti, M.: Specifying and Enforcing Access Control Policies for XML Document Sources. *World Wide Web*, Vol. 3, No. 3, Baltzer Science Publishers, Netherlands (2000).
5. Chatvichienchai, S., Iwaihara, M. and Kambayashi, Y.: Translating Content-Based Authorizations for XML Documents. *Proc. 4th Int. Conference on Web Information Systems Engineering*, IEEE CS, pp.103–112, Roma, Italy (Dec. 2003).
6. Chatvichienchai, S., Iwaihara, M. and Kambayashi, Y.: Authorization Translation for XML Document Transformation. *J. World Wide Web*, Vol. 7, No. 1, pp. 111-138 (2004).
7. Chien, S. Y., Tsotras, V. J. and Zaniolo, C.: Efficient Management of Multiversion Documents by Object Referencing. *Proc. VLDB 2001* (Sep. 2001).
8. Clark, J. and DeRose, S.: XML Path Language (XPath) Version 1.0 (November 1999). http://www.w3c.org/TR/xpath
9. Cobena, G., Abiteboul, S., Marian, A.: XyDiff Tools Detecting changes in XML Documents. http://wwwrocq.inria.fr/cobena
10. Cuppens, F., Gabillon, A.: Cover story management. *Data Knowledge Engineering*, Vol. 37, No. 2, pp. 177-201 (2001).
11. E. Fernandez, E. Gudes, and H. Song.: A Model for Evaluation and Administration of Security in Object-Oriented Databases. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 6, pp. 275-292 (April 1994).
12. Lassila, O. and Swick, R.R.: Resource Description Framework (RDF) Model and Syntax Specification. *W3C Recommendation* (Feb. 1999) http://www.w3.org/TR/REC-rdf-syntax/
13. Marian, A., Abiteboul, S., Cobna, G. and L. Mignet. Change-centric management of versions in an XML warehouse. *Proc. VLDB 2001* (Sep. 2001).
14. Microsoft XML Diff and Patch 1.0, Microsoft Corporation (2002). http://apps.gotdotnet.com/xmltools/xmldiff/overview.html
15. OASIS XACML Technical Committee: XACML 1.0 Specification Set. *OASIS Standard* (Nov. 2002). http://www.oasis-open.org/committees/xacml
16. Tichy, W. F.: RCS A System for Version Control. *Software Practice and Experience*, Vol. 15, No. 7, pp. 637-654 (July 1985).

(2) ผลงานวิจัยที่ตีพิมพ์ในวารสารวิชาการระดับนานาชาติ

Chutiporn Anutariya, Rachanee Ungrangsi, and Vilas Wuwongse: SQORE: A Framework for Semantic Query Based Ontology Retrieval. *Lecture Notes in Computer Science*, Vol. 4443, ISSN 0302-9743, Springer Verlag, pp. 924 – 929 (April 2007). Impact Factor 0.402

SQORE: A Framework for Semantic Query Based Ontology Retrieval

Chutiporn Anutariya[1], Rachanee Ungrangsi[1], and Vilas Wuwongse[2]

[1] School of Technology, Shinawatra University
99 Moo 10 Bangtoey, Samkok, Pathum Thani, 12160 Thailand
{chutiporn, rachanee}@shinawatra.ac.th
[2] School of Engineering and Technology, Asian Institute of Technology
P.O. Box 4, Klong Luang, Pathum Thani, 12120 Thailand
vw@cs.ait.ac.th

Abstract. Existing approaches to ontology retrieval solely base their search mechanisms on keyword matching while taxonomic structure is solely used for ranking purpose. Users, therefore, are not equipped with expressive means to structurally and semantically describe their ontology needs. To tackle this problem, this paper develops a framework for *Semantic Query based Ontology Retrieval*, namely *SQORE*. It enables precise formulation of a *semantic query* in order to best capture a user' s ontology requirements, which include not only the desired class and property names, but also their relations and restrictions. SQORE employs *XML Declarative Description* (*XDD*) *theory* as its theoretical foundation for modeling ontology databases and evaluating semantic queries. Moreover, *similarity score* is formally defined as a key metric for ranking the resulting ontologies based on their conceptual closeness to the given query.

1 Introduction

The *Semantic Web* aims to expand the World Wide Web by allowing data to be shared and reused across applications and communities via *ontology* [4]. However, existing ontology search engines primarily base their approaches on search terms which cannot sufficiently capture the structural and semantic information about the domain concepts that users want. *Swoogle* [6, 7] and *OntoKhoj* [9] implement the *Google*'s *PageRank*-like algorithm [5] which creates the rankings based on ontology referral network. However, this approach is currently inefficient due to the lack of links among ontologies on the Web. *OntoSearch* [10] provides several criteria for users to evaluate and browse the ontologies and then uses *AKTiveRank* [1] as metrics for ontology ranking based on the taxonomic structure information such as class names, shortest paths, the linking density and the positions of focused classes in the ontology. Although a large number of ontologies are returned as the result in these approaches, they are often un-usable because they do not meet user requirements or otherwise they require tremendous modification efforts. Furthermore, semantic information such as properties and ontological relations (e.g. subClassOf, inverseOf, equivalentClass, and subPropertyOf) is not considered.

R. Kotagiri et al. (Eds.): DASFAA 2007, LNCS 4443, pp. 924 – 929, 2007.

This paper develops *SQORE – Semantic Query based Ontology Retrieval* framework which allows users to structurally and semantically formulate their ontology requirements in terms of *semantic queries*. It employs *XML Declarative Description (XDD) theory* [2, 3, 11] as its theoretical foundation for modeling ontology databases and evaluating semantic queries, which does not only facilitate ontology matching and retrieval, but also support reasoning capability to enhance the matching results. Moreover, it also enables the use of a semantic lexical database, such as *WordNet* [8], for determining semantic relation between two given terms. Thus, the retrieval performance (precision and recall) can be significantly improved when compared to a conventional keyword search. In addition, it employs *similarity score* to rank the resulting ontologies by focusing on their conceptual closeness to the formulated semantic query.

Sect. 2 presents SQORE's overview architecture. Sect. 3 describes ontology database modeling and its semantics. Sect. 4 develops an approach to semantic query formulation and evaluation. Sect. 5 concludes and draws future research directions.

2 SQORE's System Architecture Overview

Fig. 1 illustrates SQORE's system architecture which comprises four major components: i) a *semantic query*, ii) a *retrieval engine*, iii) an *ontology database*, and iv) a *semantic lexical database*. In essence, the system works as follows: First, a user formulates and submits a semantic query which precisely captures his/her ontology requirements. The system then executes such query by semantically evaluating it against the ontology database, which comprises a collection of ontologies and a set of rules defining ontology axiomatic semantics. By incorporating these rules, implicit information about classes/properties in a query and an ontology can be derived, and hence enabling semantic query evaluation. Furthermore, when class/property names defined in a query and an ontology do not *exactly match* (=), four possibilities occur: i) *equivalence* (≡): the two terms are synonym, ii) *more general* (⊇): the query term is broader, iii) *less general* (⊆): the ontology term is broader, and iv) *unknown* (≠): the relation is unknown. To tackle this, a referenced lexical database, such as *WordNet* [8], is employed in order to determine their appropriate semantic relation. Finally, the system computes the *semantic similarity score* between a given query and an ontology in the collection, which ranges from 0 (strong dissimilarity) to 1 (strong similarity), and returns as the answer the list of ranked ontologies.

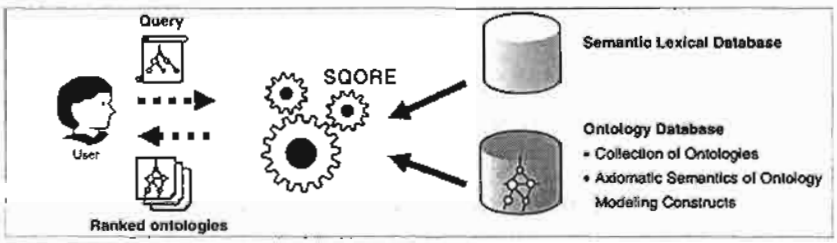


Fig. 1. SQORE System Architecture

3 Ontology Database Model and Its Semantics

By employment of XDD theory, an ontology formalized in RDF(S) or OWL can readily and directly become an XDD description. In order to determine the meaning of a particular ontology, formalization of the axiomatic semantics of each RDF(S)/OWL modeling construct is demanded.

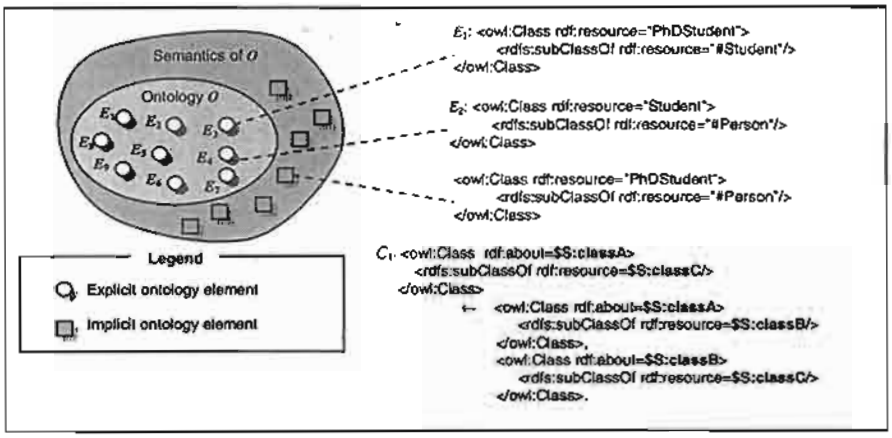


Fig. 2. Semantics of the ontology O wrt. the axiomatic semantics of RDF(S)/OWL constructs

Fig. 2 gives an example of formalizing rdfs:subClassOf in terms of an XDD description C_1. By incorporating this description, the meaning of the ontology O, consequently, yields elements E_1–E_9 which are explicitly defined by O, together with the derived ones. For instance, as depicted by Fig. 2, assume that Query Q searches for ontologies with PhDStudent modeled as a subClassOf Person, while Ontology O defines PhDStudent as a subClassOf Student which is then a subClassOf Person; therefore by modeling the semantics of subClassOf to be transitive, one can derive that in Ontology O, PhDStudent is also a subClassOf Person, and thus satisfying the query requirement.

Founded on this formalism, an ontology database ODB becomes an XDD description with two parts: i) ODB_C: An ontology collection consisting of n ontologies $O_1,\ldots, O_n$, and ii) ODB_A: A set of XML clauses defining the axiomatic semantics of ontology modeling constructs. The database semantics, denoted by $\mathcal{M}(ODB)$, are the set of OWL elements describing classes, properties or instances (individuals) which are directly represented by the database via the ontology collection ODB_C plus those derivable from it via the axiomatic semantics ODB_A.

4 Semantic Query Formulation and Evaluation

Definition 1 (Semantic Query). A *semantic query* is formulated as an XML element of the form:

```
<sq:Query>
  <sq:mandatoryConditions> m1 m2 ... mn </sq:mandatoryConditions>
  <sq:optionalConditions> o1 o2 ... or </sq:optionalConditions>
  <sq:semanticLexiconReference> url </sq:semanticLexiconReference>
  <sq:similarityWeightFactors>
        <sq:mandatoryConditionWeight> wM </sq:mandatoryConditionWeight>
        <sq:stringMatching> w≈ </sq:stringMatching>
        <sq:synonymRelation> w≡ </sq:synonymRelation>
        <sq:moreGeneralRelation> w⊇ </sq:moreGeneralRelation>
        <sq:lessGeneralRelation> w⊆ </sq:lessGeneralRelation>
        <sq:unknownRelation> w∗ </sq:unknownRelation>
  </sq:similarityWeightFactors>
</sq:Query>
```

where

- m_i is an OWL/RDF(S) expression describing a query's mandatory condition,
- o_i is an OWL/RDF(S) expression describing a query's optional condition,
- *url* gives a URL of a semantic lexical database,
- $w_M, w_{\approx}, w_{\equiv}, w_{\supseteq}, w_{\subseteq}, w_{*} \in [0, 1]$ are semantic weight factors,
- semanticLexiconReference- and similarityWeightFactors-elements are optional.

Given a semantic query Q, let $mcond(Q)$ and $ocond(Q)$, respectively, denote the sets of mandatory conditions and optional conditions. □

Formally speaking, an ontology O in ODB satisfies a condition m_i or o_i if such a conditional element is included in the meaning of O. The weight factor w_M allows explicit specification of how important the mandatory conditions are, and hence $1-w_M$ becomes the weight for the optional conditions. The semantic lexicon reference specifies external knowledge used for determining appropriate semantic relations between elements of Q and O. Thus, based on the discovered semantic relation between a query element t_1 and an ontology element t_2 defined as follows, the weight factors $w_{\approx}$, $w_{\equiv}$, $w_{\supseteq}$, $w_{\subseteq}$, w_{*}, respectively, allow the user to quantify how similar the two elements are. In principle, it is recommended that $1 = w_{\approx} \geq w_{\equiv} \geq w_{\supseteq} \geq w_{\subseteq} \geq w_{*} = 0$. In practice, these weights can be configured as default settings of the system or manually defined by a user.

5 Similarity Measures

This section formally defines important similarity measures. First, let Σ be an *ontology alphabet* comprising *ontology elements* in the following sets:

- C: Class names (such as Person, Student, PhDStudent),
- P: Property names (such as firstname, lastname, supervises, isSupervisedBy),
- D: Datatypes (such as xsd:string, xsd:nonNegativeInteger),
- M: Modeling constructs of RDF(S) and OWL (such as rdfs:subClassOf),
- R: Restrictions on classes and properties, having the form $m(a,b)$ where $m \in M$, $a \in (C \cup P)$, $b \in (C \cup P \cup D)$ (such as rdfs:subClassOf(Student, Person)).

The following definition first measures how well two ontology elements match.

Definition 2 (Element Similarity Score: SS_E). The similarity of two ontology elements x and y is measured by:

- For $x,y \in C$ or $x,y \in P$ or $x,y \in D$:

$$SS_E(x,y) = \begin{cases} w_{=} & : \text{if } x = y \\ w_{\equiv} & : \text{if } x \equiv y \\ w_{\supseteq} & : \text{if } x \supseteq y \\ w_{\subseteq} & : \text{if } x \subseteq y \\ w_{\times} & : \text{otherwise} \end{cases} \quad (1)$$

- For $x = m(a_1,b_1) \in R$ and $y = m(a_2,b_2) \in R$:

$$SS_E(x,y) = SS_E(a_1,a_2) * SS_E(b_1,b_2) \quad (2)$$

- Otherwise:

$$SS_E(x,y) = \text{unknown} \quad (3)$$

□

Based on this element similarity score definition, one can measure the similarity between a given element x in Σ and an ontology O in ODB by finding the highest similarity score between x and each element y that is semantically defined by O. This is defined by:

Definition 3 (Best Element Similarity Score: SS_B). The similarity between an element $x \in C \cup P \cup R$ and an ontology $O \in ODB$ is measured by:

$$SS_B(x,O) = \max_{y \in M(O)} S(x,y) \quad (4)$$

□

Next, the satisfaction scores of mandatory conditions (SS_M) and of optional conditions (SS_O) are defined.

Definition 4 (Satisfaction Scores of Mandatory conditions: SS_M and Optional conditions: SS_O). With respect to an ontology O, the satisfaction scores of Q's mandatory conditions (SS_M) and Q's optional conditions (SS_O) are measured by:

$$SS_M(Q,O) = \begin{cases} 1 & : \text{if } mcond(Q) = \varnothing \\ 0 & : \text{if } \exists (c \in mcond(Q))\ SS_B(c,O) \leq w_{\times} \\ \dfrac{\sum_{c \in mcond(Q)} SS_B(c,O)}{|mcond(Q)|} & : \text{otherwise} \end{cases} \quad (5)$$

$$SS_O(Q,O) = \begin{cases} 1 & : \text{if } ocond(Q) = \varnothing \\ 0 & : \text{if } SS_M(Q,O) = 0 \\ \dfrac{\sum_{c \in ocond(Q)} SS_B(c,O)}{|ocond(Q)|} & : \text{otherwise} \end{cases} \quad (6)$$

□

Finally, one can measure the similarity between a semantic query Q and an ontology O by integrating the two components: mandatory condition satisfaction score and optional condition satisfaction score, as follows:

Definition 5 (Query-Ontology Similarity Score). The similarity between a semantic query Q and an ontology O is measured by:

$$SS(Q,O) = w_M SS_M(mcond_Q, O) + (1 - w_M) SS_O(ocond_Q, O) \quad (7)$$

□

6 Conclusions

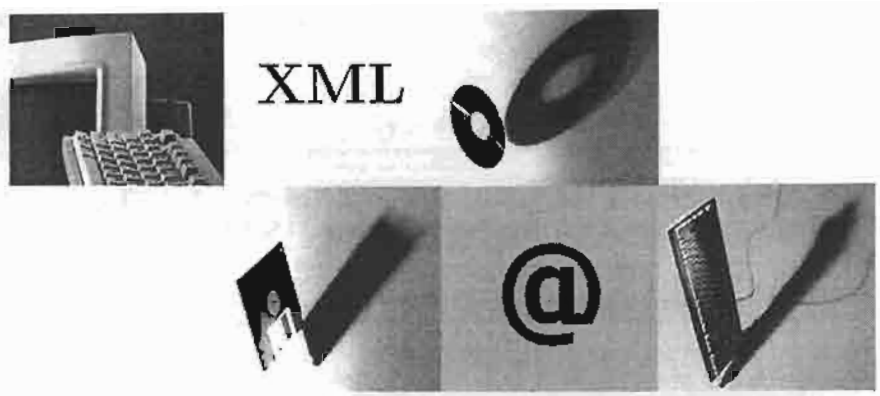
This paper proposes and develops *SQORE* – a novel framework for ontology retrieval system based on *semantic query*. It enables a user to precisely and structurally formulate their ontology requirements, which include not only the desired class and property names, but also their relations and restrictions. Moreover, when evaluating a query, its semantics together with an ontology's semantics are also taken into account in order to correctly and semantically match them.

Acknowledgement

This work was supported by Thailand Research Fund and Commission on Higher Education, Thailand, under grant number MRG4780192.

References

1. Alani, H. and Brewster, C.: Metrics for Ranking Ontologies. In Proceedings of 4th International EON Workshop, 15th Int'l WWW Conference, Edinburgh, 2006.
2. Anutariya, C., Wuwongse, V. and Akama, K.: XML Declarative Description with First-Order Logical Constraints. Computational Intelligence, Vol. 21, No. 2, pp. 130-156 (2005)
3. Anutariya, C., Wuwongse, V., Akama, K., Wattanapailin, V.: Semantic Web Modeling and Programming with XDD. Proc. Semantic Web Working Symposium (SWWS-01), CA (2001), 161–180.
4. Berners-Lee, T., Handler, J., and Lassila, O.: The Semantic Web, Scientific American, May 2001.
5. Brin, S., and Page, L. The anatomy of a large-scale hyper-textual web search engine. In Seventh International WorldWide Web Conference, Brisbane, Australia, 1998
6. Ding, L., Finin, T., Joshi, A., Pan, R., Scott Cost, R., Peng, Y., Reddivari, P., Doshi, V., Sachs, J., Swoogle: a search and metadata engine for the semantic web. Proc. 13 ACM Int'l Conf. Information and Knowledge Management, November 08-13, 2004, DC
7. Finin, T., Mayfield, J., Joshi, A., Scott Cost, R., Fink, C.: Information Retrieval and the Semantic Web, Proc. 38th Annual Hawaii Int'l Conf. System Sciences (HICSS'05) - Track 4, p.113.1, 2005
8. Miller, A., Wordnet: A lexical database for English. In Communications of the ACM, number 38(11), 1995.
9. Patel, C., Supekar, K., Lee, Y., Park, E. K., OntoKhoj: a semantic web portal for ontology searching, ranking and classification, Proc. 5th ACM Int'l Workshop on Web Information and Data Management, November 07-08, 2003, Louisiana.
10. Thomas, E., Alani, H., Sleeman, D. and Brewster, C.: Searching and Ranking Ontologies on the Semantic Web. In Proc. Workshop Ontology Management: Searching, Selection, Ranking, and Segmentation. 3rd K-CAP 2005, pp. 57-60, Banff, Canada
11. Wuwongse, V., Anutariya, C., Akama, K., Nantajeewarawat, E.: XML Declarative Description (XDD): A Language for the Semantic Web. IEEE Intelligent Systems, Vol. 16, No. 3, pp. 54–65 (May/June 2001)



Towards Integration of XML Document Access and Version Control

Authors

- S. Chatvichienchai (Siebold University of Nagasaki, Japan)
- C.Anutariya (Shinawatra U., Thailand),
- V.Wuwongse (Asian Institute of Technology, Thailand),
- M.Iwaihara (Kyoto U., Japan)
- Y.Kambayashi (Kyoto U., Japan) passed away on February 6th, 2004.

Contribution

- Investigate interaction problems of access and version control of XML documents
- Propose a novel framework integrating access and version control to solve the problems
- Formalize their interrelationships by means of RDF Declarative Description (RDD) Theory

Outline of presentation

- Background
  - Access control for XML documents
  - Version control for XML documents
- Requirements and issues of integrating access and version control
- Proposed framework that formalizes the inter-relationships of access and version control by RDF Description
- Conclusion and future work

Access Control on XML Documents

- Application of XML Documents : Electronic Catalogs, Invoices, Contacts, Patient Records etc
- Access control on shared XML documents is essential

Product Catalog

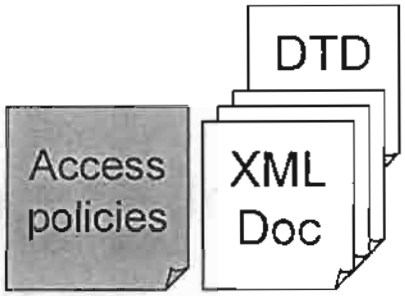
```
<product id="P01234">
  <name> Green Tea 250g </name>
  <price> 1,500 </price>
  <cost> 700 </cost>
</product>
```

Granularity of access control: element level of XML documents

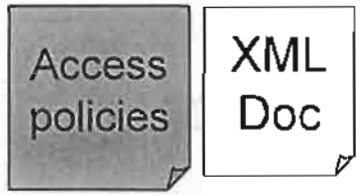
- Sales managers are allowed to read all data of products.
- Customers are not allowed to read costs of products

Related work of XML access control

- Proposed models: AuthorX (E.Bertino et.al [WWW'00]), Rule-Based XML Access Control Model (C.Anutariya et.al [RuleML'03]), and XACML of OASIS etc.
- Common properties
  - Fined-grained access control: elements of document
  - Coarse-grained access control: the whole document



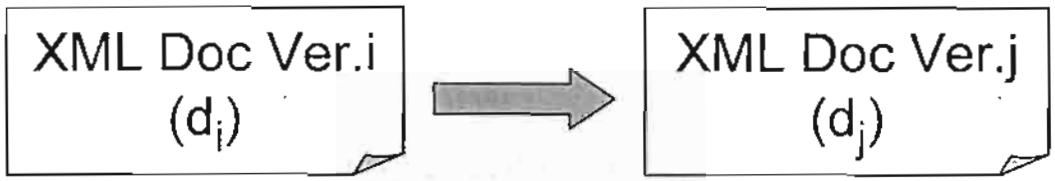
Definition of user access rights for a set of schema instances

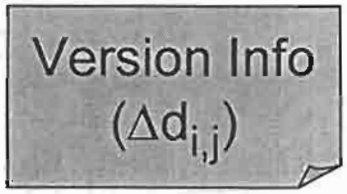


Definition of user access rights for a specific document

Version control for XML documents

- Keep track of changes of evolving XML documents
- Maintain appropriate version information





- users who modify the documents
- Date / time of modification
- modification details

- Enabling the documents to be reverted back to their previous versions

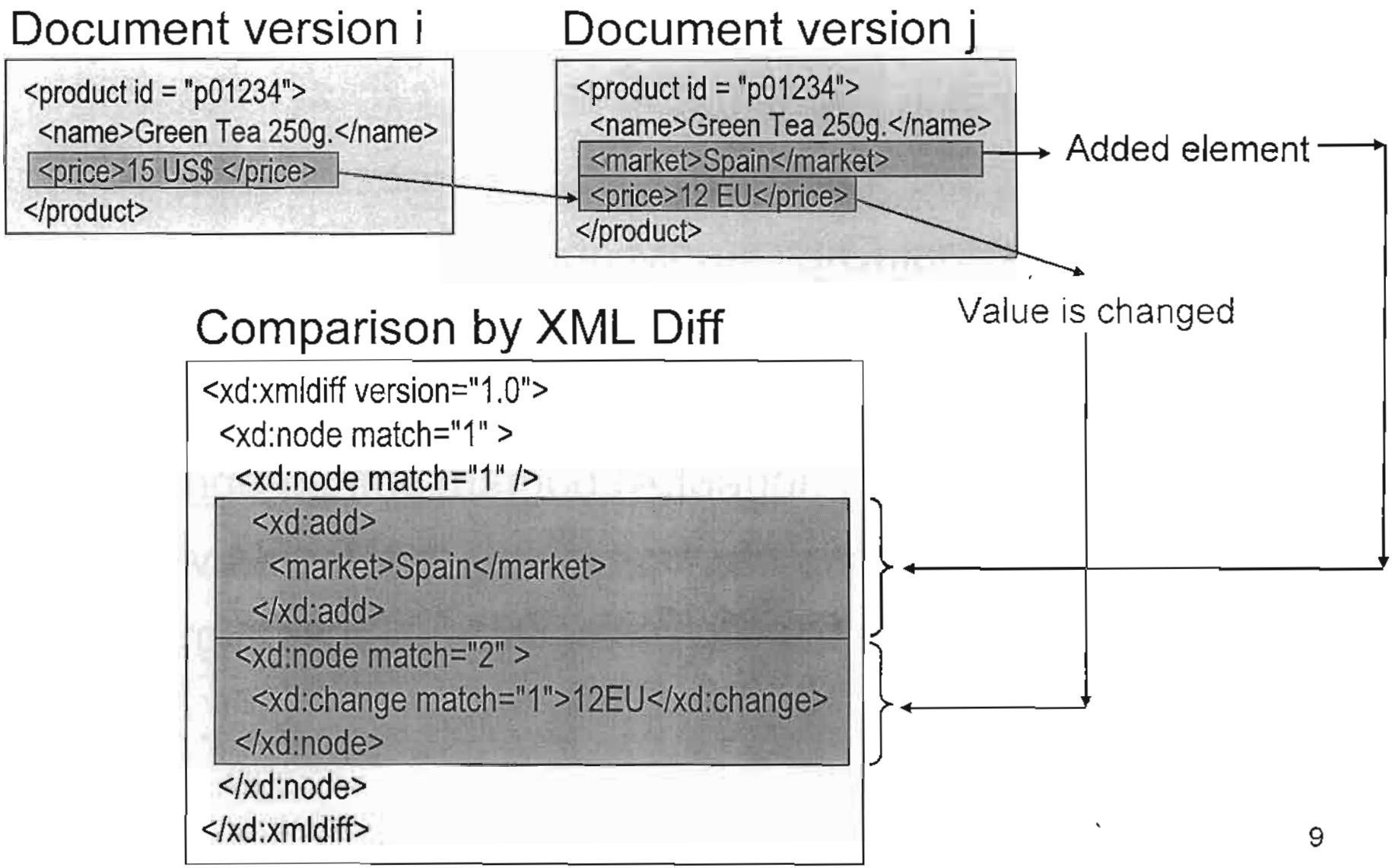
Example: Given d_j and $\Delta d_{i,j}$, we can compute d_i.

How to compute Delta $\Delta d_{i,j}$ of XML document

Given a pair of documents d_i and d_j, delta $\Delta d_{i,j}$ can be computed by one of the following methods:

- a script generated directly from the edit commands of a structured editor.
- Document comparison analyzed by a structured diff package, such as
  - XyDiff Tool (http://wwwrocq.inria.fr/cobena), or
  - Microsoft XML Diff (http://apps.gotdotnet.com/xmltools/xmldiff/overview.html).

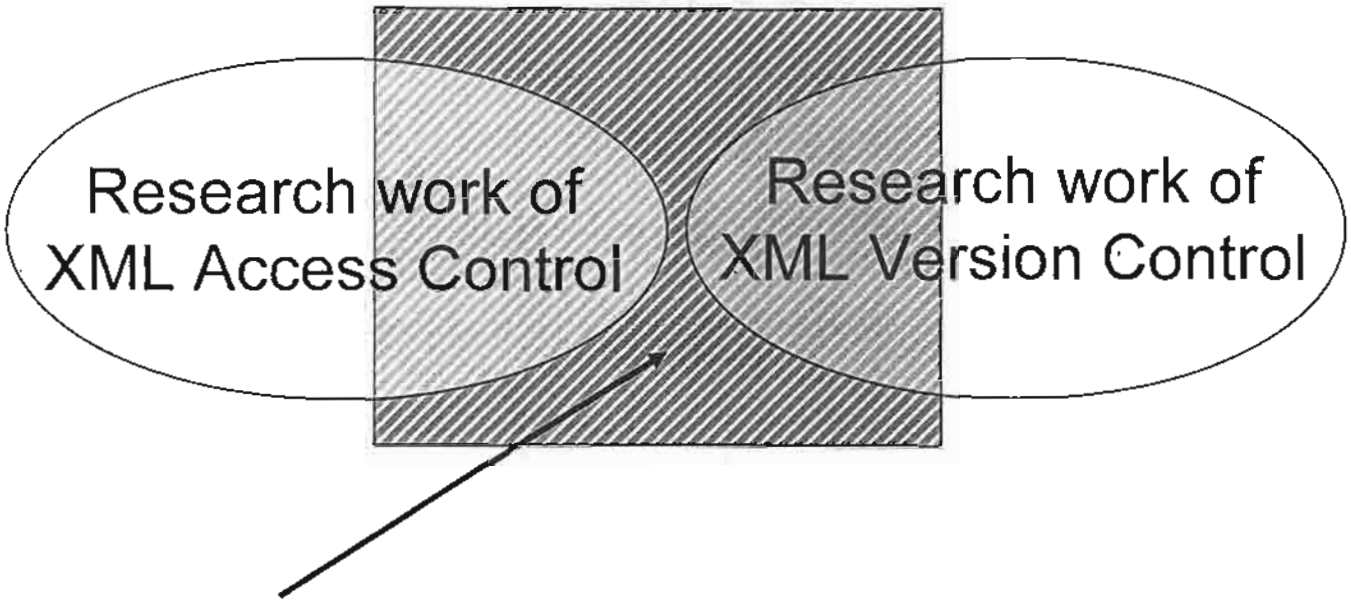
Document comparison by XML Diff



Related work of version control

- In a typical Revision Control System (RCS) scheme (W. F. Tichy [SPE'85]), changes of document d_i to d_j are stored in a script denoted by delta $\Delta d_{i,j}$.
- Reference-Based Version Model (S.Chien et.al [VLDB01]) focuses on the storage performance of multi-version XML documents.
- A change-centric method (A.Marian et.al[VLDB02]) stores the latest version of a XML document and the sequence of forward completed deltas.

Open Problems



There is no work discussing interaction problems between access and version control of XML documents.

Integration need of access & version control (1/2)

1. Version control simplifies schema design and access control policy definition.

Catalog.xml

```
<product id = "p01234">
  <name>Green Tea 250g.</name>
  <price>15 US$ </price>
</product>
```

Catalog.xml

```
<product id = "p01234">
  <market country="USA">
    <name>Green Tea 250g.</name>
    <price>15 US$ </price>
  </market>
  <market country="Spain">
    <name>Té verde 250g.</name>
    <price>12 EU</price>
  </market>
</product>
```

In case without version control

Catalog.xml

Ver.0

```
<product id = "p01234">
  <name>Green Tea 250g.</name>
  <price>15 US$ </price>
</product>
```

Ver.1

```
<product id = "p01234">
  <name>Green Tea 250g.</name>
  <market>USA</market>
  <price>15 US$ </price>
</product>
```

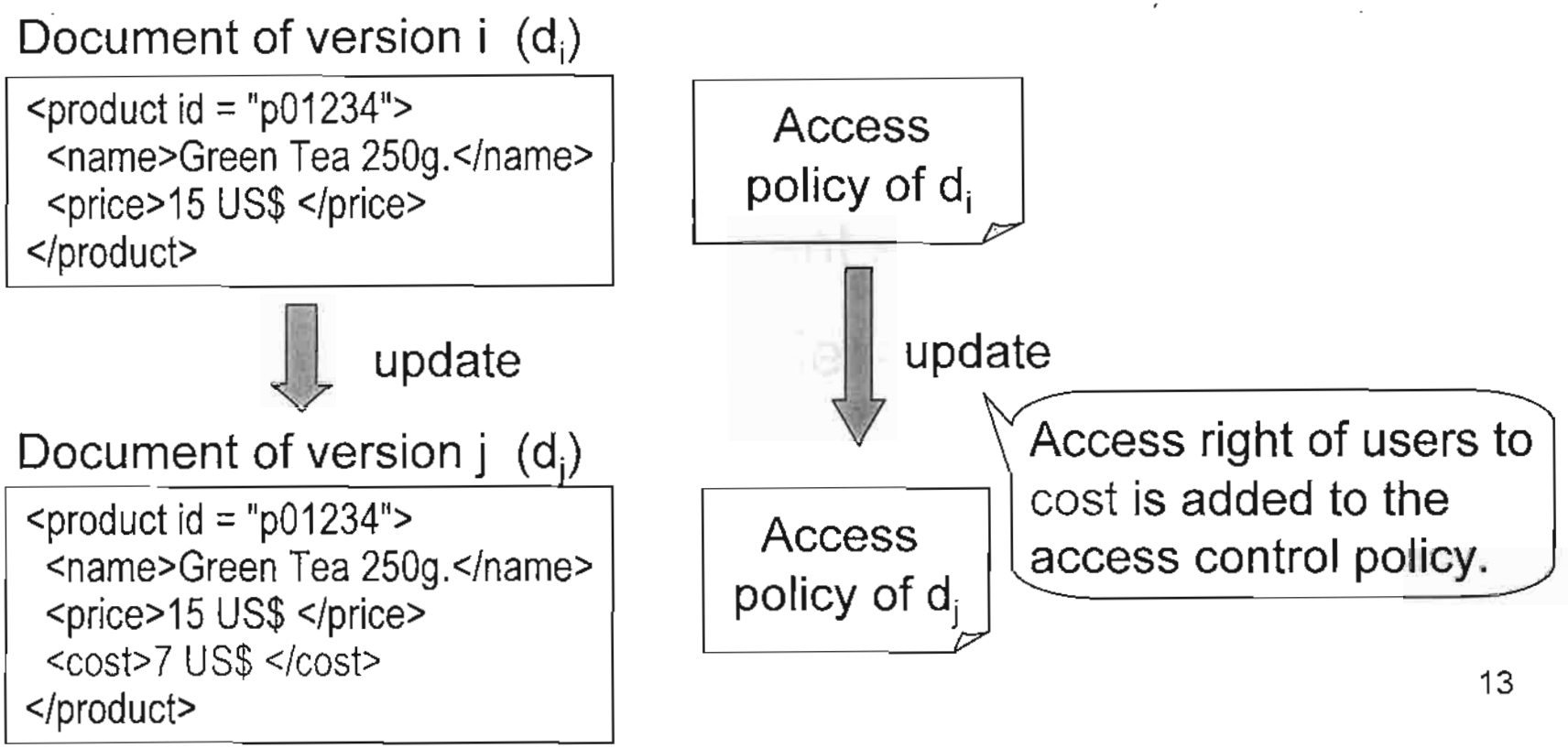
Ver.2

```
<product id = "p01234">
  <name>Té verde 250g.</name>
  <market>España</market>
  <price>12 EU</price>
</product>
```

In case with version control

Integration need of access & version control (2/2)

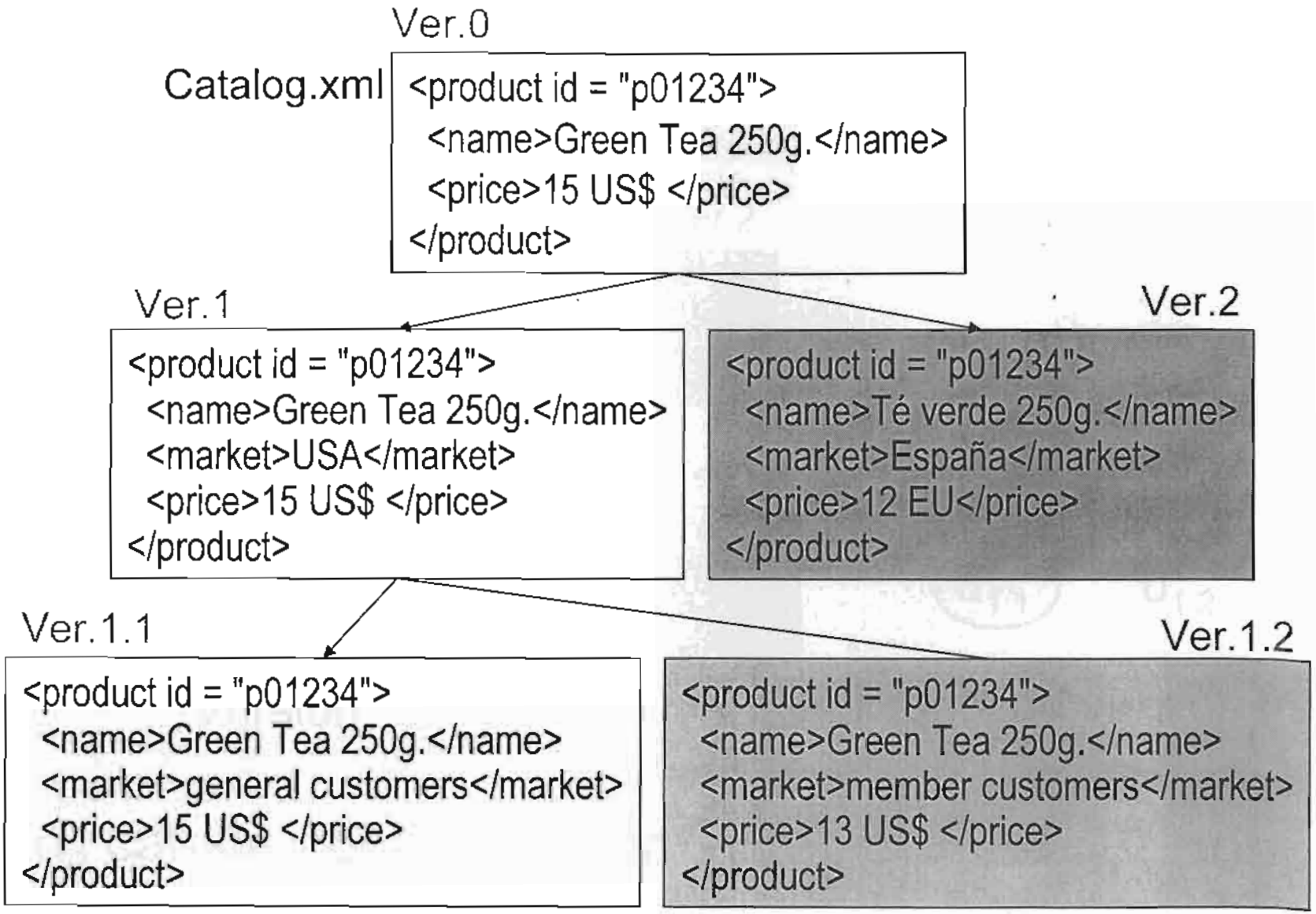
1. Version control simplifies schema design and access control policy definition.
2. Define an access policy of a document of new version based on that of document of current version.



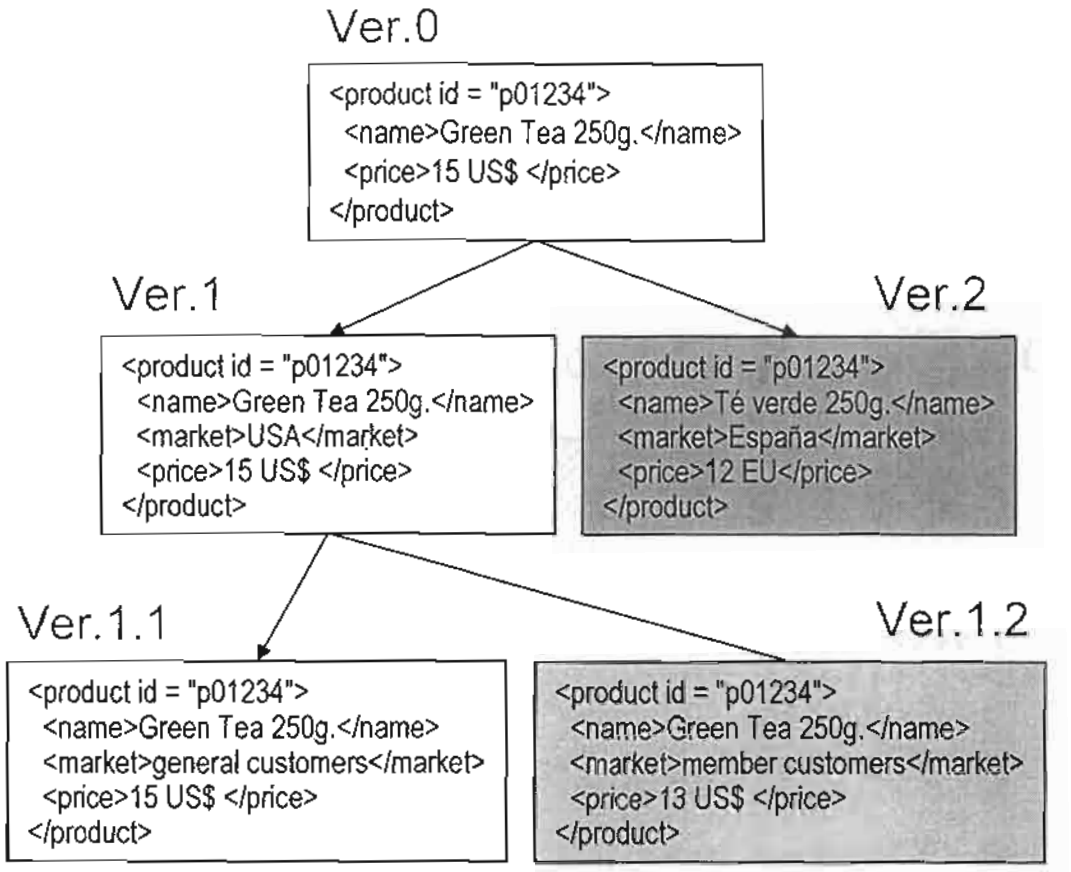
Issues on access and version control integration

- Inference problems
  - Existence of sensitive documents by version history
  - Sensitive data contained in documents of new version
- Improper disclosure of confidential data from updating document
- Version control over document schemas
- Temporal features allowing definition of access control policies with certain conditions on timestamp attributes

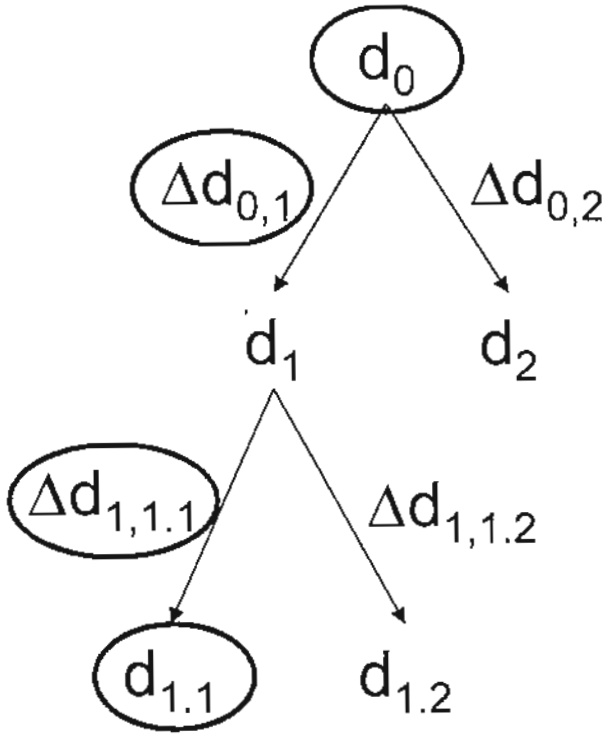
Version history



Representation of version history by version tree



Version history

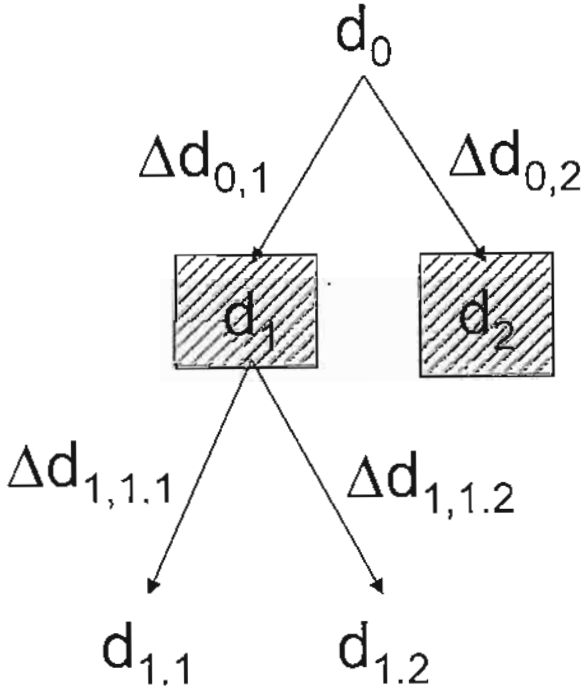


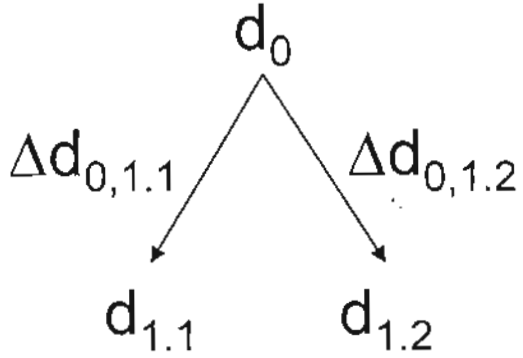
Proposed version tree

d_0 can be computed from given $d_{1.1}$, $\Delta d_{1,1,1}$, $\Delta d_{0,1}$

Inferring existence of sensitive documents by version history

Assume that Bob is not allowed to access documents of ver.1 and 2.

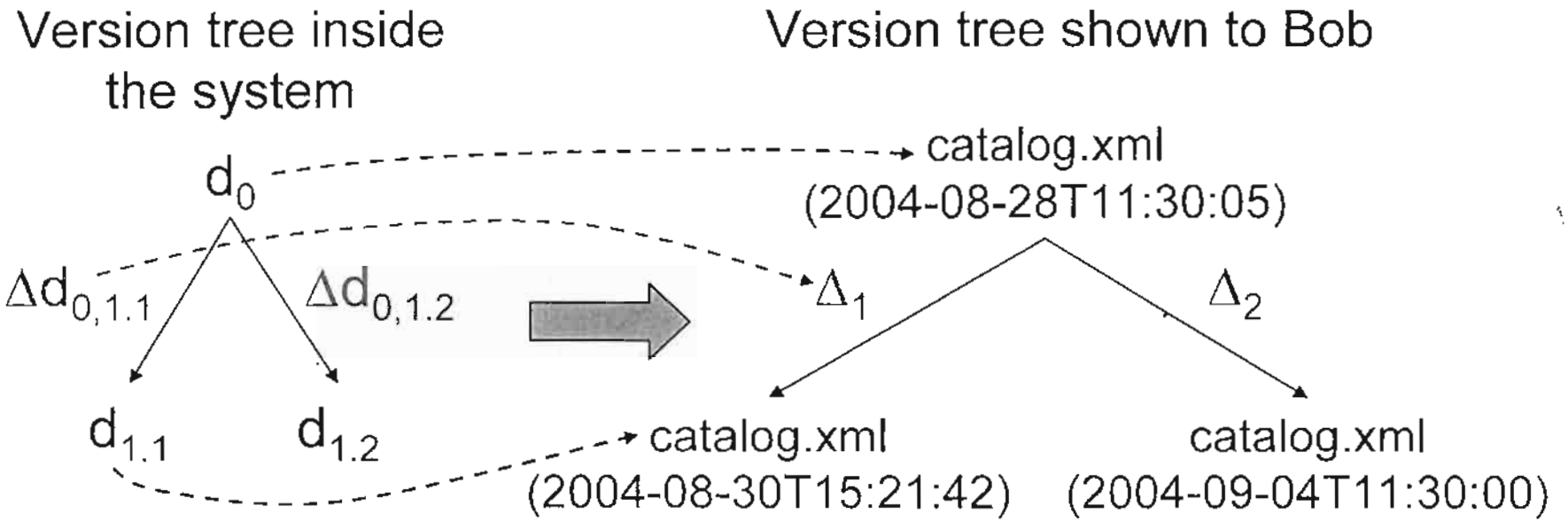




View of version tree seen by Bob.
(d_1 does not appear between d_0 and $d_{1.1}$).

Problem: Bob can aware of d_1 if its filename denotes the disappeared version that is between d_0 and $d_{1.1}$.

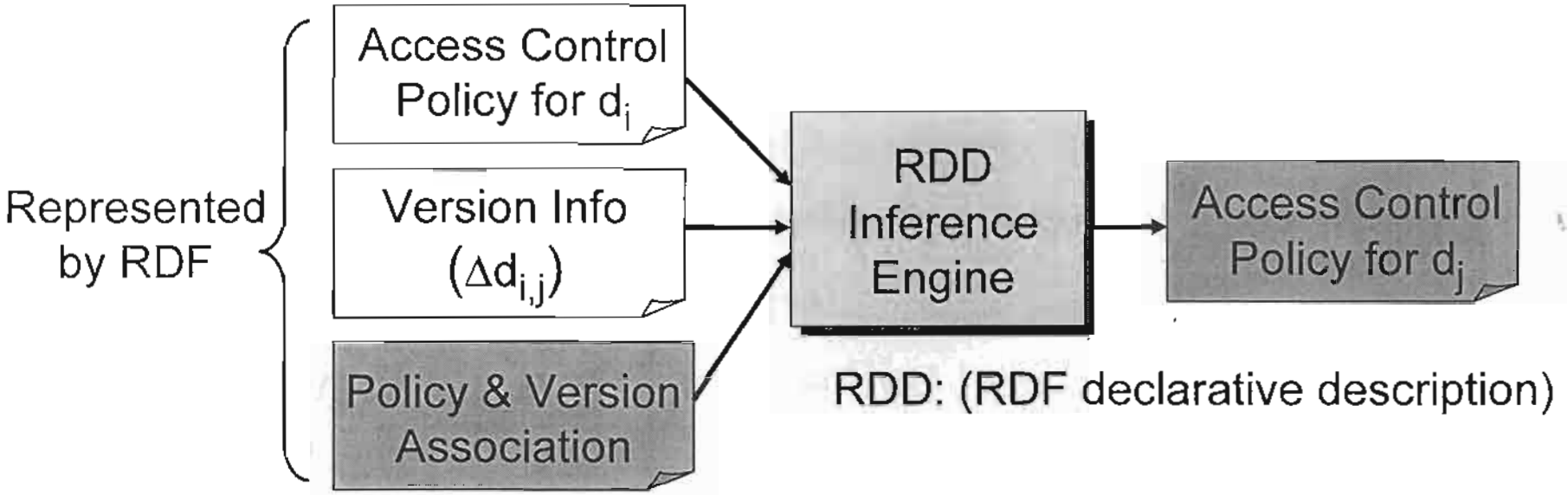
Solution of Inferring existence of sensitive documents



Solution:

1. d_i is represented by its file name that is based on timestamp.
2. $\Delta d_{i,j}$ is represented by Δ_k where k is an integer.

Proposed framework



RDF (Resource Description Framework) is developed by the W3C for Web-based metadata, using XML as an interchange syntax.

RDF allows users defining metadata that has scalability and flexible structure better than by those defined by XML.

RDF (Resource Description Framework)

RDF is built on the following rules:

- a *Resource* is anything that can have a URI; this includes all the world's Web pages, as well as individual elements of an XML document.
- a *PropertyType* is a resource that has a name and can be used as a property, for example Author or Title.
- a *Property* is the combination of a Resource, a PropertyType, and a value.

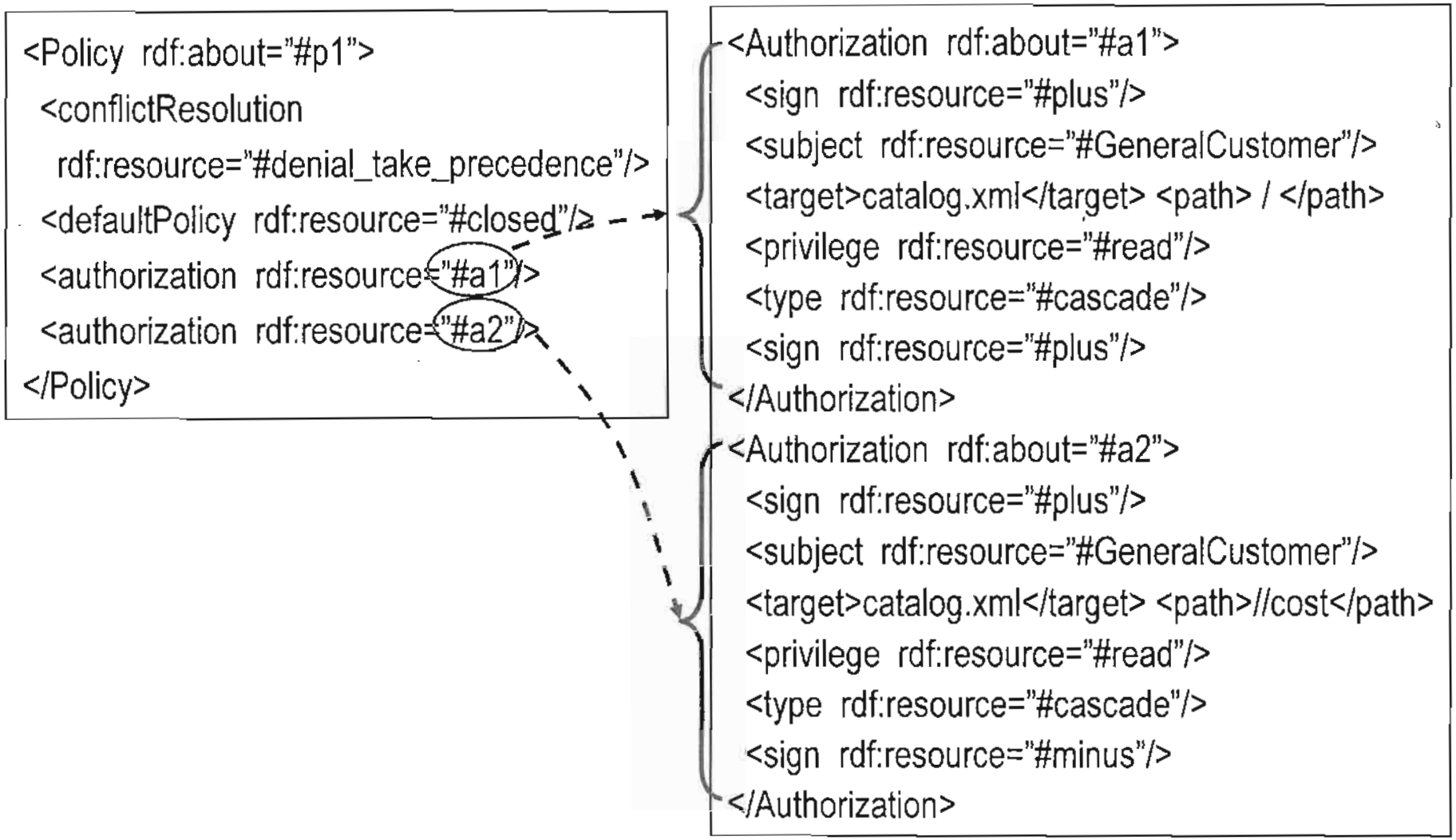
Access control policies

Definition of an *access control policy* consists of the following arguments:

- Subject: a user, user group or a role,
- Target: an XML document,
- Path: an XPath expression identifying data in the document,
- Level: instance level or schema level,
- Privilege: read or write operation,
- Type: cascade or no_propagation,
- Sign: '+'(grant) or '-'(deny),
- Priority: conflict resolution policy (denial_take_precedence, descendant_take_precedence etc),
- Default: open or close policy.

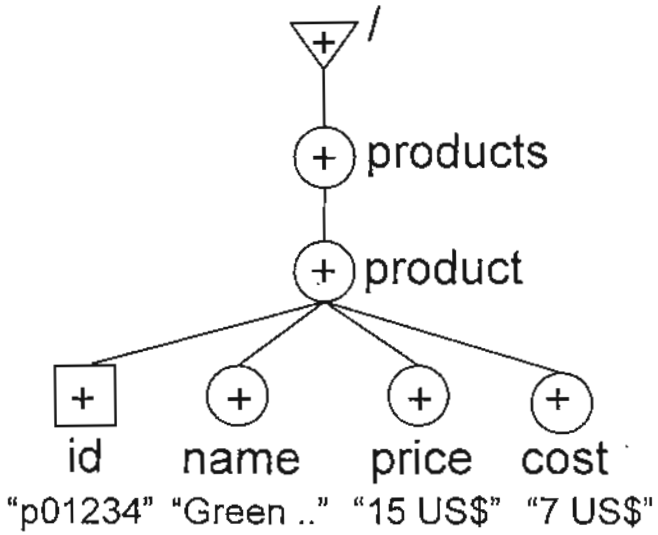
Access control policy defined by RDF (1/4)

An access policy p1 consists of two authorization rules a1 and a2.



Access control policy defined by RDF (2/4)

Based on a1, access right of Bob to each node is granted.

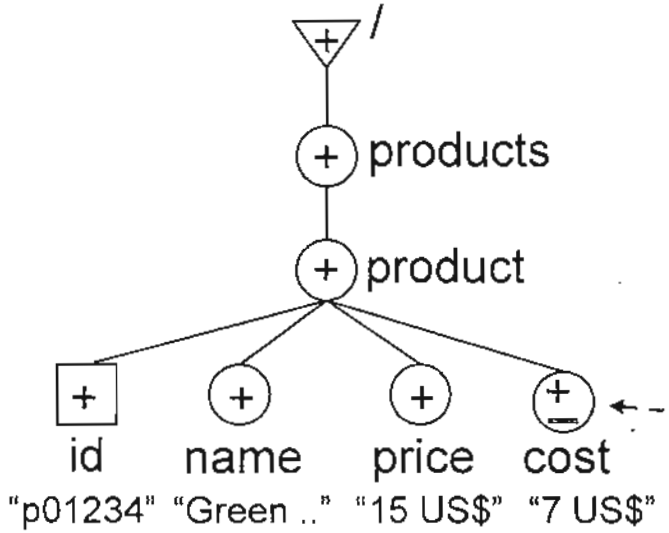


Document tree of catalog.xml

```
<Authorization rdf:about="#a1">
  <sign rdf:resource="#plus"/>
  <subject rdf:resource="#Bob"/>
  <target>catalog.xml</target> <path> / </path>
  <privilege rdf:resource="#read"/>
  <type rdf:resource="#cascade"/>
  <sign rdf:resource="#plus"/>
</Authorization>
<Authorization rdf:about="#a2">
  <sign rdf:resource="#plus"/>
  <subject rdf:resource="#GeneralCustomer"/>
  <target>catalog.xml</target> <path>//cost</path>
  <privilege rdf:resource="#read"/>
  <type rdf:resource="#cascade"/>
  <sign rdf:resource="#minus"/>
</Authorization>
```

Access control policy defined by RDF (3/4)

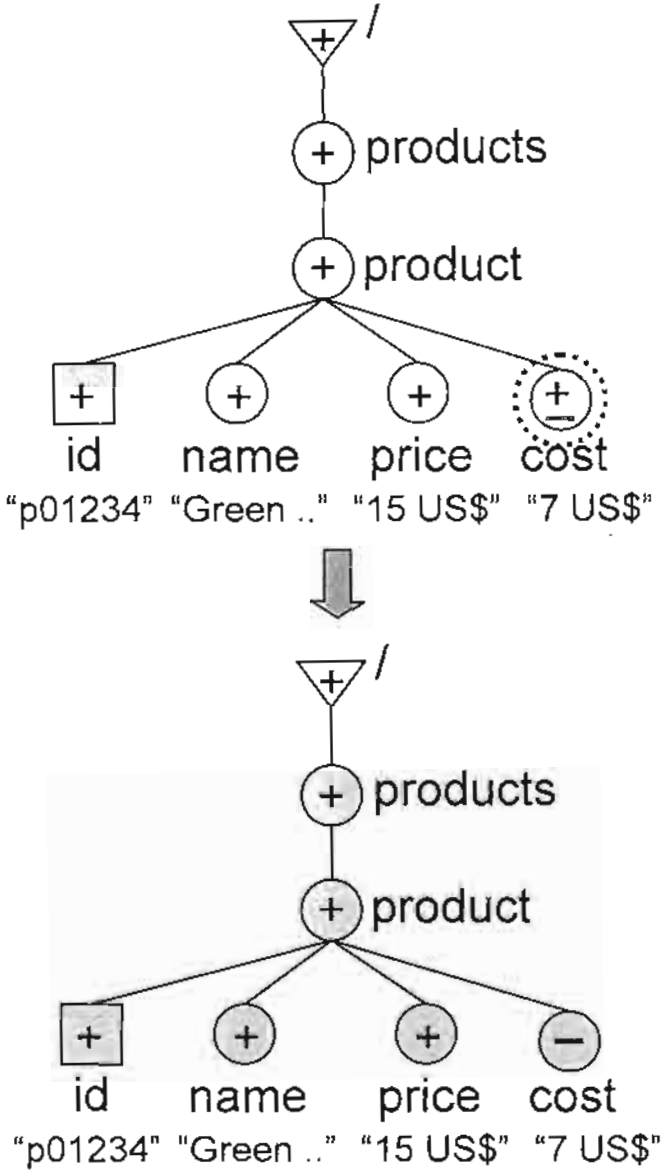
Based on authorization rule a1, Bob is allowed read each node.



Document tree of catalog.xml

```
<Authorization rdf:about="#a1">
  <sign rdf:resource="#plus"/>
  <subject rdf:resource="#Bob"/>
  <target>catalog.xml</target> <path> / </path>
  <privilege rdf:resource="#read"/>
  <type rdf:resource="#cascade"/>
  <sign rdf:resource="#plus"/>
</Authorization>
<Authorization rdf:about="#a2">
  <sign rdf:resource="#plus"/>
  <subject rdf:resource="#Bob"/>
  <target>catalog.xml</target> <path>//cost</path>
  <privilege rdf:resource="#read"/>
  <type rdf:resource="#cascade"/>
  <sign rdf:resource="#minus"/>
</Authorization>
```

Access control policy defined by RDF (4/4)




```
<Policy rdf:about="#p1">
  <conflictResolution
    rdf:resource="#denial_take_precedence"/>
  <defaultPolicy rdf:resource="#closed"/>
  <authorization rdf:resource="#a1"/>
  <authorization rdf:resource="#a2"/>
</Policy>
```

Version information

Definition of version information consists of the following arguments:

- Document: a reference to the baseline version of a document;
- PreviousVersion: a reference to its previous version,
- Creator: author of the described version,
- TimeStamp: the version's timestamp,
- ValidFrom, ValidTo: representing valid time interval of the described version,
- Delta: changes between the previous and current versions specified in XML Diff language.

Version information defined by RDF

```
<Version rdf:about="#d1.2">
 <document rdf:resource="catalog.xml"/>
 <previousVersion rdf:resource="#d1"/>
 <creator rdf:resource="#john"/>
 <timestamp>2004-02-01T09:00:00</timestamp>
 <validFrom>2004-03-01T00:00:00</validFrom>
 <validTo>2004-04-01T00:00:00</validTo>
  <delta rdf:parseType ="Literal">
   <xd:xmldiff>
    <xd:node match="1">
     <xd:change match="2">member customers</xd:change>
     <xd:change match="3">1300Yen</xd:change>
    </xd:node>
   </xd:xmldiff>
  </delta>
</Version>
```

Policy-version-association

A policy-version-association clause is represented as an RDF clause of the following form:

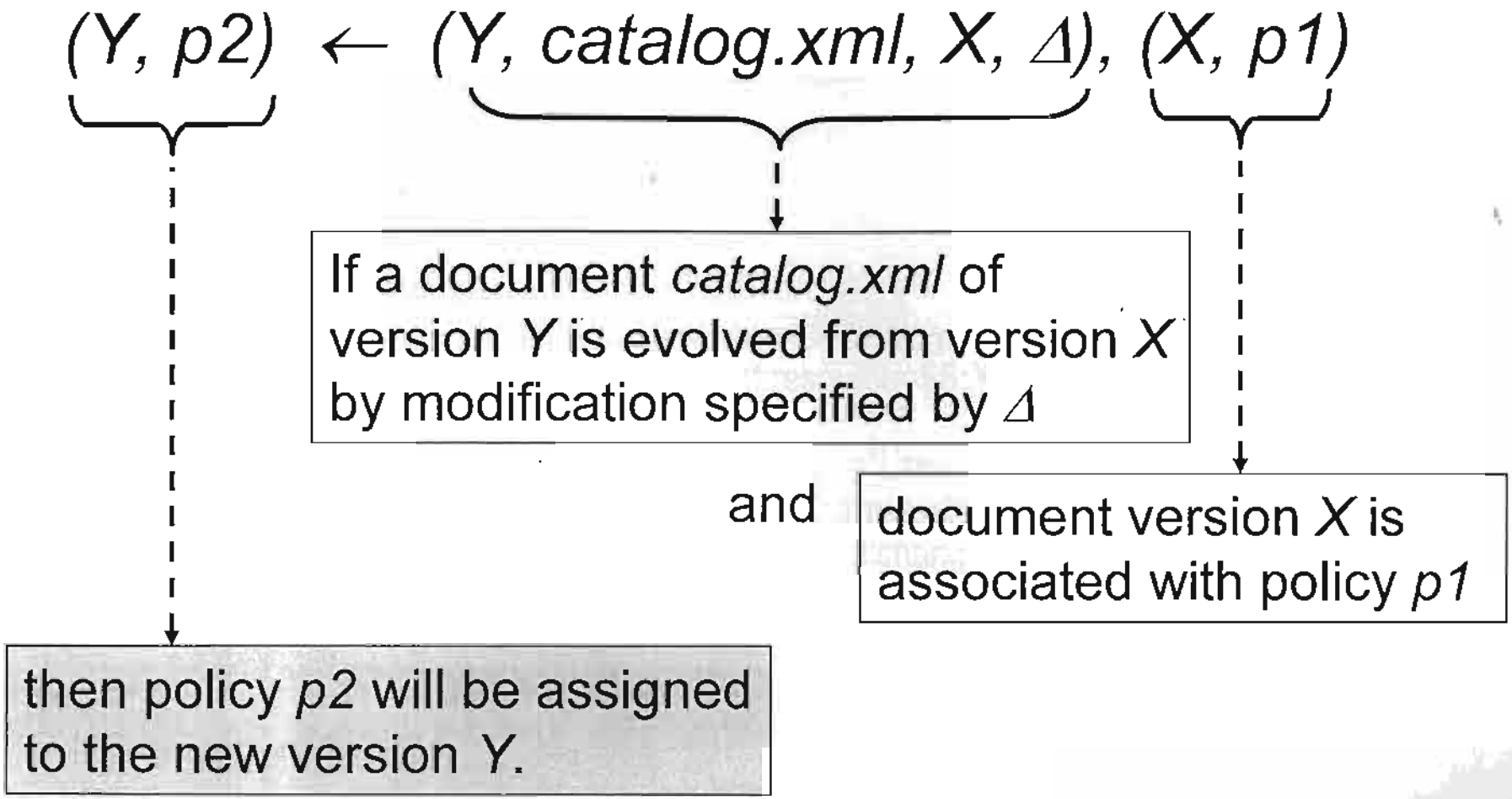
$$H \leftarrow B_1, \ldots, B_n,$$

where

H: an RDF expression describing a derived policy-version-association,

B_i : an RDF expression or an RDF constraint restricting a certain condition on the derived association.

A sample of policy-version-association



Policy-version-association described by RDF

(Y, p2) ← (Y, catalog.xml, X, Δ), (X, p1)

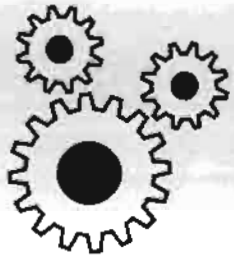
```
<Policy_Version_Association>
 <policy rdf:resource="#p2"/>
 <version rdf:resource=$S:Y/>
</Policy_Version_Association>
  ← <Version rdf:about=$S:Y>
        <document rdf:resource="catalog.xml"/>
        <previousVersion rdf:resource=$S:X/>
        <delta>
         <xd:xmldiff rdf:parseType="Literal">
          <xd:change match=$S:anynode>
            member customers </xd:change>
                $E:otherchanges
         </xd:xmldiff>
        </delta>
        $E:versionProperties
    </Version>,
    <Policy_Version_Association>
     <policy rdf:resource="#p1"/>
        <version rdf:resource=$S:X/>
    </Policy_Version_Association>
```

Conclusions

- Investigated interaction problems of access and version control of XML documents
- Proposed a novel framework integrating access and version control to solve the problems
- Formalized their interrelationships by means of RDF Declarative Description Theory

Further work

1. Implementation of the framework by RDD's inference engine
2. A mechanism that handles version control over document schemas
3. Detection of document modification (Δ) that results in improper disclosure of confidential data
4. Temporal features that deal with temporal XML document DB and temporal query processing



SQORE

A framework for Semantic Query based Ontology REtrieval

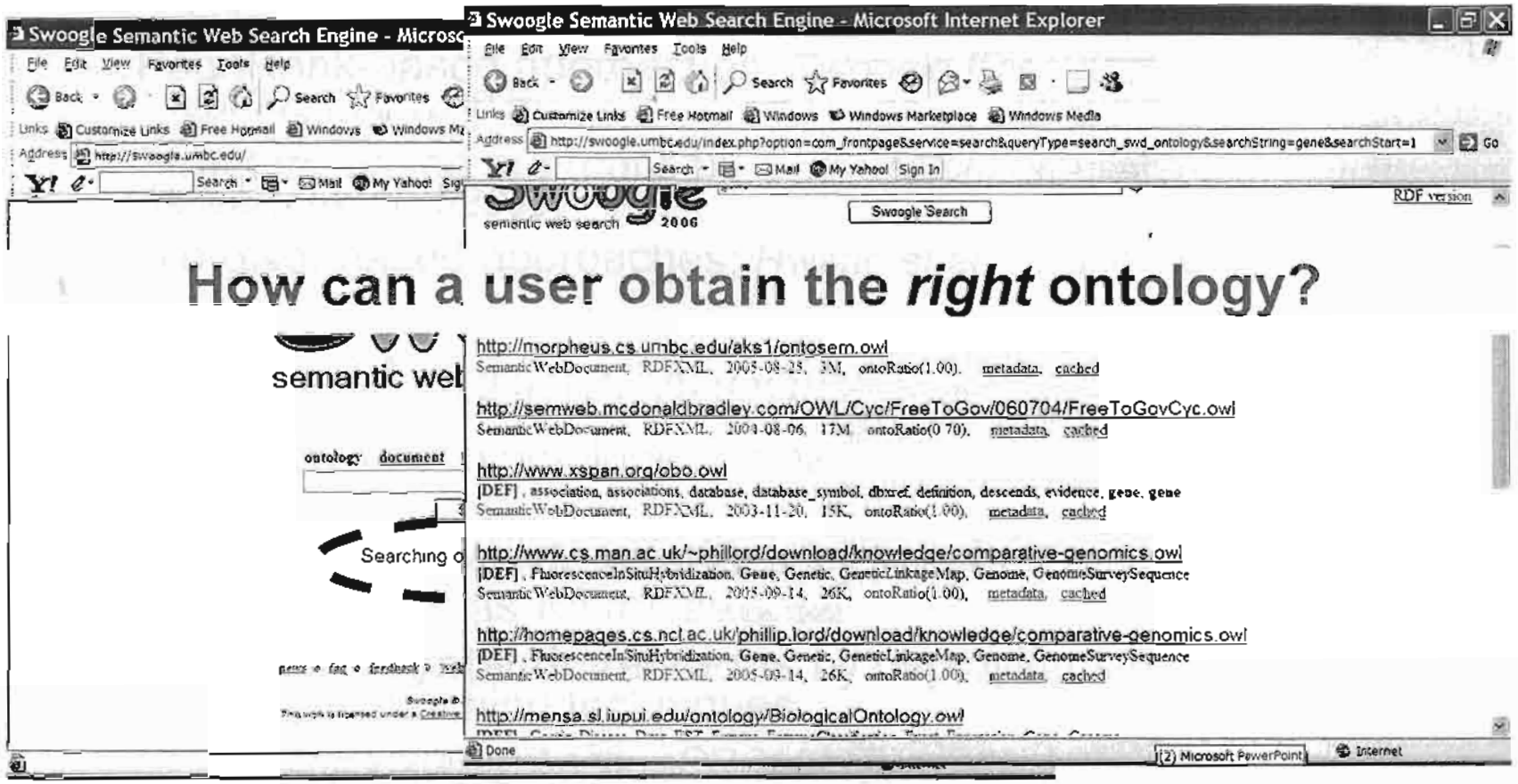
Chutiporn Anutariya[1] , *Rachanee Ungrangsi[1]*, Vilas Wuwongse[2]
Shinawatra University[1], Asian Institute of Technology[2]

This work was supported by Thailand Research Fund and Commission on Higher Education, Thailand, under grant number MRG4780192.

Outline

- Motivation
- Literature Review
- SQORE System Architecture
- Similarity Measures
- An Example
- Conclusions

Motivation: Ontology Reuse Problem



Literature Review

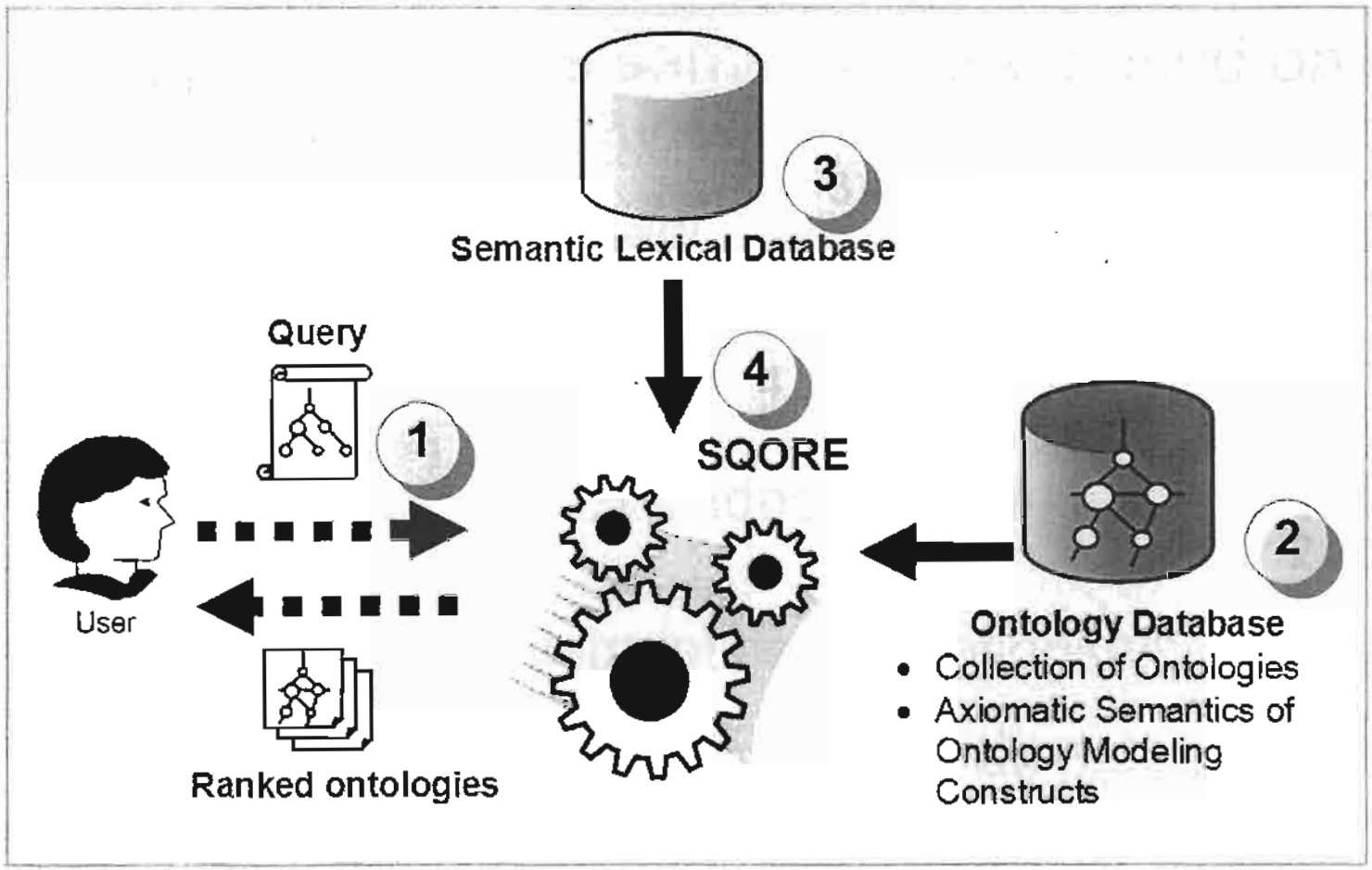
- Ontology Retrieval
  - PageRank-based approaches: *Swoogle [Ding2004], OntoKhoj[Patel2003]*
  - Structure-based approaches: *OntoSearch[Thomas2005], AktiveRank[Alani2006]*
  - WordNet-based approaches: *Hwang et.al [Hwang2006]*

- Ontology Matching/Mapping
  - **Similarity** of two ontologies is based on semantic and syntactic relations
  - determines **semantic relations** by referring to linguistic resources such as *WordNet*[Miller1995]
  - determines **syntactic relations** by deploying *string* matching and *graph* matching techniques
  - *i.e. COMA[Do2002], Anchor-PROMPT[Noy2001], S-Match[Giunchiglia2004]*

Drawbacks

- Keywords cannot capture user's ontology requirements efficiently.
- Existing Ontology Retrieval systems ignore Axiomatic Semantic information such as properties and ontological relations
- Using only WordNet to determine semantics is insufficient

SQORE System Architecture



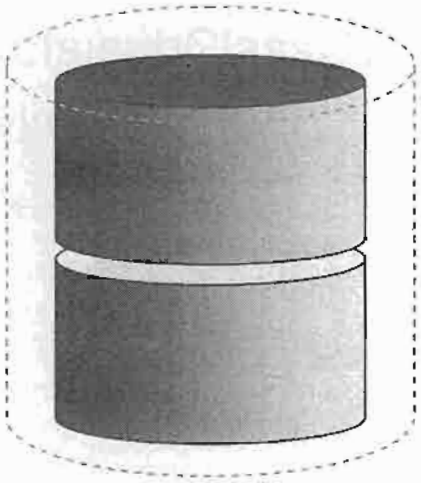
Ontology Database Model

- ODB model and its semantics are based on *XML Declarative Description (XDD) theory* *[Anutariya01, Wuwongse01]*
- A description in XDD is a set of
  - *Ordinary XML elements*
  - *XML expressions*: extended XML elements with variables

    : constraints and relationships
- OWL or RDF/RDFS ontologies directly becomes an XDD description

Ontology Database Model

- An ontology database *ODB* becomes an XDD description with two parts:



$ODB_{Collection\ of\ Ontologies}$

$ODB_{Axiomatic\ Semantics}$

- By incorporating ODB_A, *implicit information* about classes/properties in a query and ontology *can be derived*, and hence enabling semantic query evaluation.

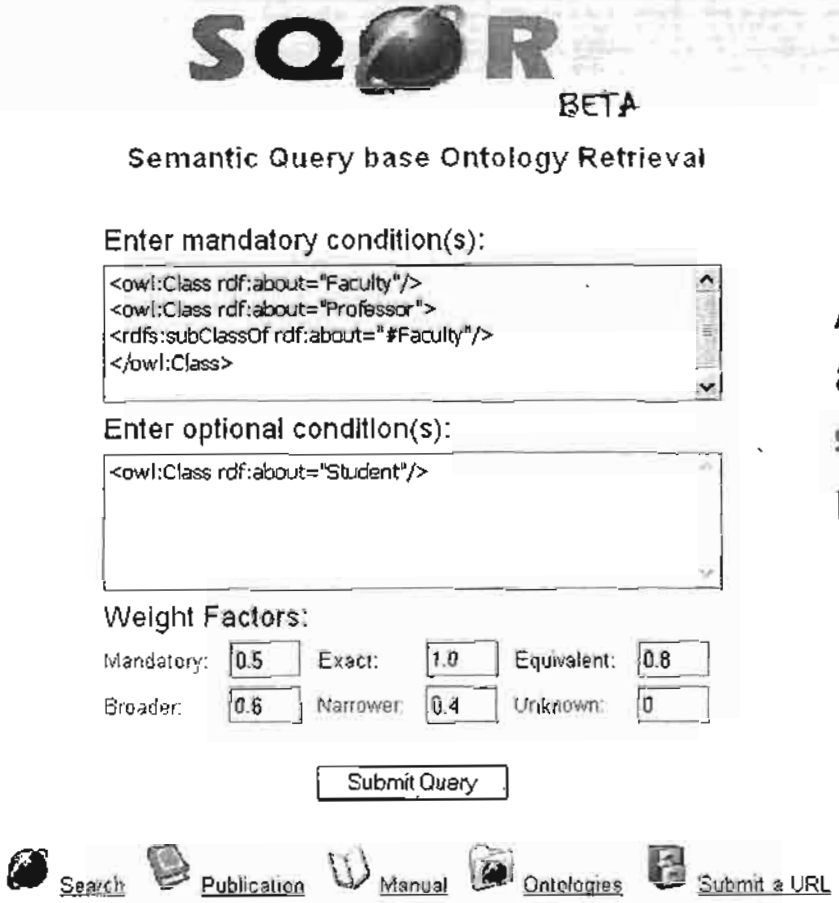
Ontology Database Model: an Example

```
C1: <owl:Class  rdf:about=$S:classA>
        <rdfs:subClassOf rdf:resource=$S:classC/>
    </owl:Class>
        ←   <owl:Class rdf:about=$S:classA>
                <rdfs:subClassOf rdf:resource=$S:classB/>
            </owl:Class>,
            <owl:Class rdf:about=$S:classB>
                <rdfs:subClassOf rdf:resource=$S:classC/>
            </owl:Class>.
```

Semantic Lexical Database

- When class/property names defined in a query and an ontology do not match (=), four possible semantic relations occur:
  - i) : the two terms are synonym,
  - ii) *more general*($\supseteq$): the query term is broader,
  - iii) *less general*($\subseteq$): the ontology term is broader, and
  - iv) *unknown*($\neq$): the relation is unknown.
- Semantic lexical database such as *WordNet*[Miller1995] is employed to determine semantic relations

Semantic Query



A *semantic query* is developed here as means for precisely and **structurally** describing ontology requirements.

Element Similarity Score

represents the similarity of two ontology elements *x* and *y*.

For $x,y \in C$ or $x,y \in P$ or $x,y \in D$:

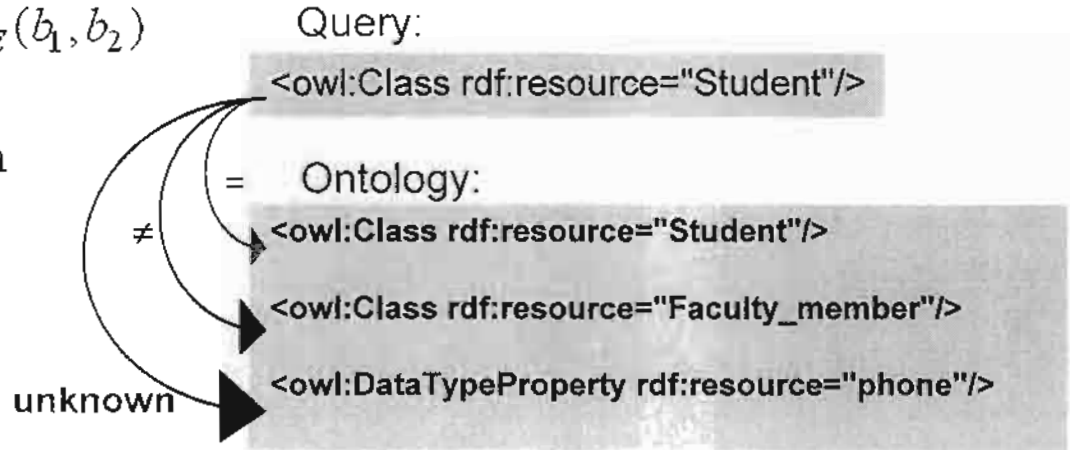
$$SS_E(x,y) = \begin{cases} w_{=} & : \text{if } x = y \\ w_{\equiv} & : \text{if } x \equiv y \\ w_{\supseteq} & : \text{if } x \supseteq y \\ w_{\subseteq} & : \text{if } x \subseteq y \\ w_{\neq} & : \text{otherwise} \end{cases}$$

For $x = m(a_1,b_1) \in R$ and $y = m(a_2,b_2) \in R$:

$$SS_E(x,y) = SS_E(a_1,a_2) * SS_E(b_1,b_2)$$

Otherwise:

$$SS_E(x,y) = \text{unknown}$$



Best Element Similarity Score

measures the similarity between an element $x \in C \cup P \cup R$ and an ontology $O \in ODB$

$$SS_B(x, O) = \max_{y \in M(O)} SS_E(x, y)$$

$SS_B(q,O) = w_=$

Query:

```
<owl:Class rdf:resource="Student"/>
```

Ontology O:

```
<owl:Class rdf:resource="Student"/>

<owl:Class rdf:resource="Faculty_member"/>

<owl:DataTypeProperty rdf:resource="phone"/>
```

Satisfaction Score of Conditions

Mandatory conditions:

$$SS_M(Q,O) = \begin{cases} 1 & : \text{if } mcond(Q) = \varnothing \\ 0 & : \text{if } \exists (c \in mcond(Q))\, SS_B(c, O) \leq w_{\neq} \\ \dfrac{\sum_{c \in mcond(Q)} SS_B(c,O)}{|\, mcond(Q)\,|} & : \text{otherwise} \end{cases}$$

Optional conditions:

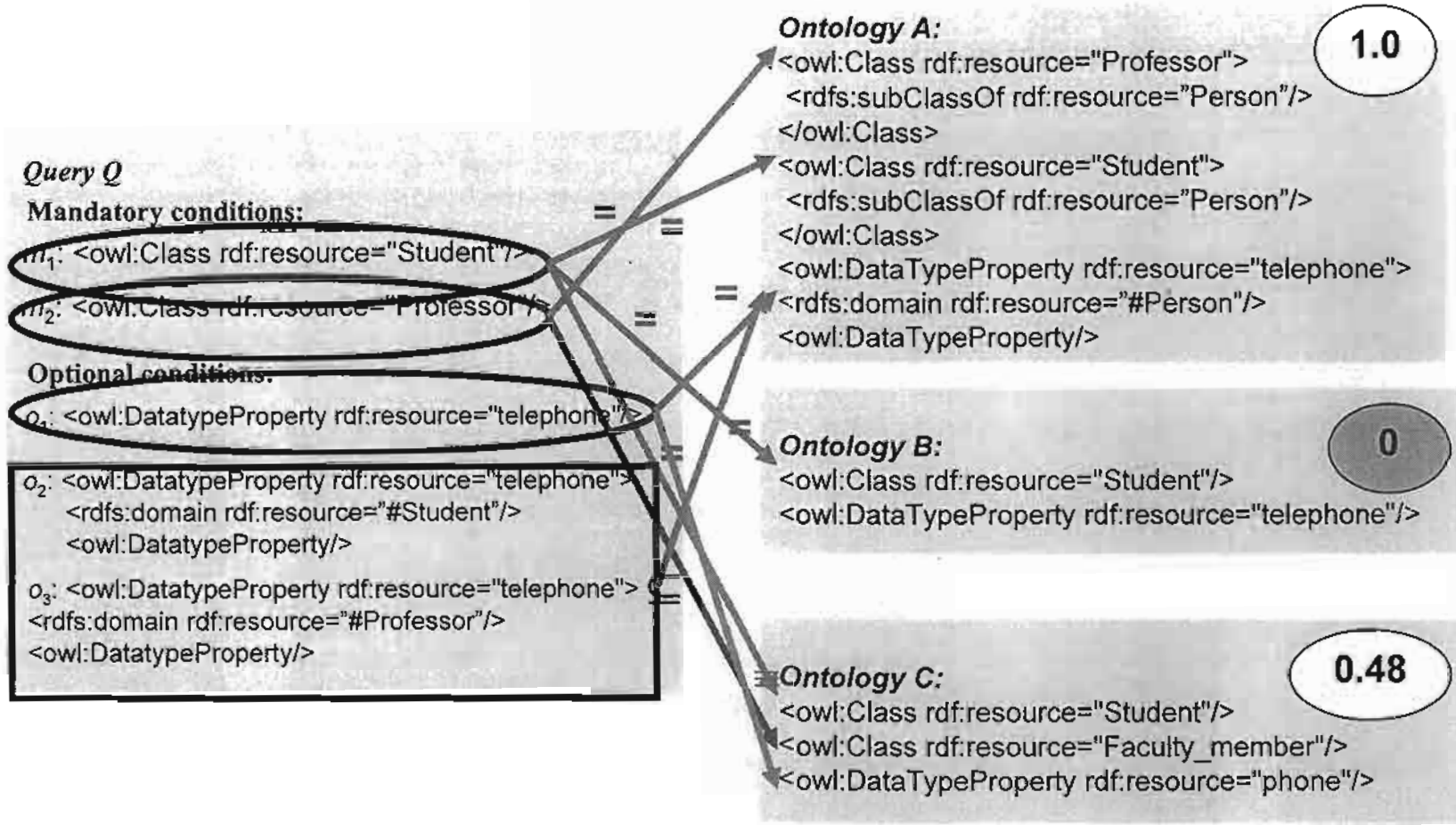
$$SS_O(Q,O) = \begin{cases} 1 & : \text{if } ocond(Q) = \varnothing \\ 0 & : \text{if } SS_M(Q,O) = 0 \\ \dfrac{\sum_{c \in ocond(Q)} SS_B(c,O)}{|\, ocond(Q)\,|} & : \text{otherwise} \end{cases}$$

Query-Ontology Similarity Score

represents the similarity between a semantic query Q and an ontology O

$$SS(Q,O) = w_M SS_M(mcond_Q, O) + (1 - w_M) SS_O(ocond_Q, O)$$

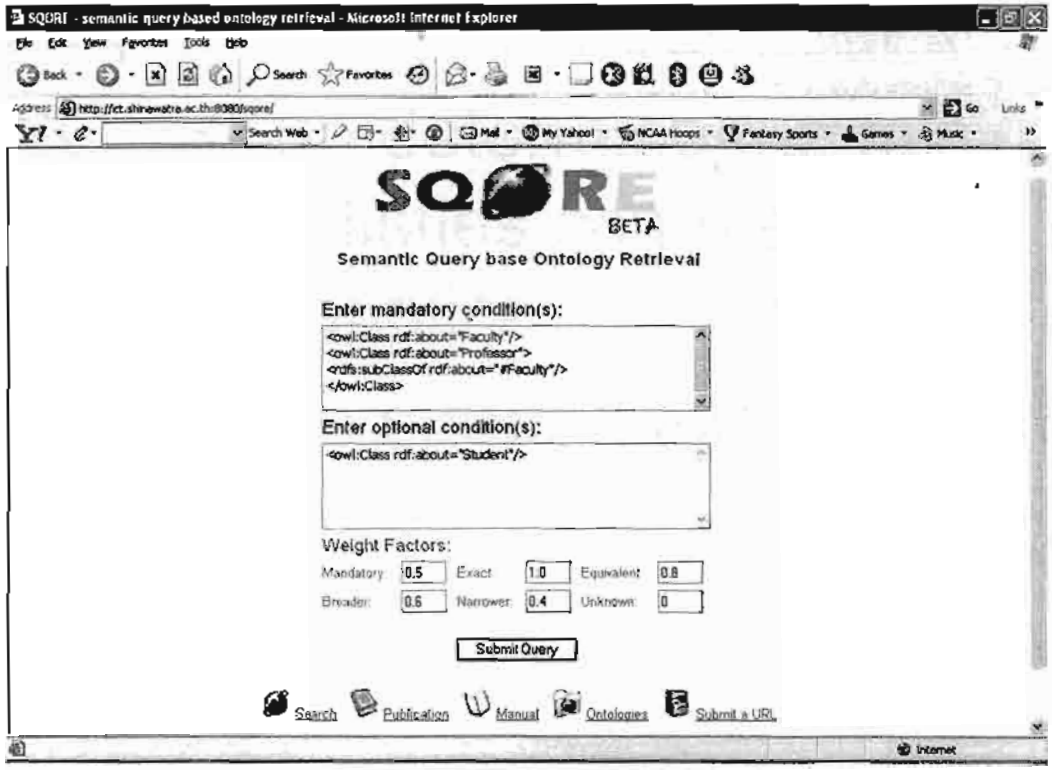
An Example



$W_M = 0.5$ $W_{=}=1.0$ $W_{\equiv}=0.8$ $W_{\supseteq}=0.6$ $W_{\subseteq}=0.4$ $W_{\neq}=0$

SQORE Prototype System

http://ict.shinawatra.ac.th:8080/sqore



Future Work

- *Compare* SQORE with other Ontology Retrieval Techniques
- *Extend* SQORE to determine the minimum set of ontologies that satisfies complex multidisciplinary application requirements
- *Deploy* SQORE to support exploratory search for non-expert users

Q&A