Extracting Knowledge from XML Document Repository: A Semantic Web-Based Approach

Henry M. Kim

Schulich School of Business, York University

Arijit Sengupta

Kelley School of Business, Indiana University

Contact author:

Henry M. Kim

4700 Keele St., Toronto, Ontario Canada M3J 1P3

hkim@schulich.yorku.ca | (416) 736-2100 x77952 | (416) 736-5687 [fax]

This is a pre-print version of article to be published in *Information Technology Management* in 2007.
Posted on http://www.yorku.ca/hmkim/files/krox-jit-207-ver6.pdf with permission

Abstract

XML plays an important role as the standard language for representing structured data for the traditional Web, and hence many Web-based knowledge management repositories store data and documents in XML. If semantics about the data are formally represented in an ontology, then it is possible to extract knowledge: This is done as ontology definitions and axioms are applied to XML data to automatically infer knowledge that is not explicitly represented in the repository. Ontologies also play a central role in realizing the burgeoning vision of the semantic Web, wherein data will be more sharable because their semantics will be represented in Web-accessible ontologies. In this paper, we demonstrate how an ontology can be used to extract knowledge from an exemplar XML repository of Shakespeare's plays. We then implement an architecture for this ontology using *de facto* languages of the semantic Web including OWL and RuleML, thus preparing the ontology for use in data sharing. It has been predicted that the early adopters of the semantic Web will develop ontologies that leverage XML, provide intra-organizational value such as knowledge extraction capabilities that are irrespective of the semantic Web, and have the potential for inter-organizational data sharing over the semantic Web. The contribution of our proof-of-concept application, KROX, is that it serves as a blueprint for other ontology developers who believe that the growth of the semantic Web will unfold in this manner.

Keywords: XML, Ontologies, Knowledge Extraction, Query processing, Semantic Web

Extracting Knowledge from XML Document Repository: A Semantic Web-Based Approach

1 Introduction

Ontologies play a central role in realizing the burgeoning vision of the semantic Web [1]. Formal ontologies with which data are modeled—as opposed to informal ontologies used for information systems design [2]—"consist of a representational vocabulary with precise definitions of the meanings of the terms of this vocabulary plus a set of formal axioms that constrain interpretation and well-formed use of these terms" [3]. Ontology definitions and axioms can be codified and applied to structured data for automated inference— i.e. query answering. Ontologies thus can be used internally by the organization to support decision-making. In the vision of the semantic Web, a software agent will refer to ontologies associated with data it needs to process, irrespective of where that data reside. Even if the data are beyond the span of control of the agent—i.e. the data are external or "foreign" to the agent—it will still be possible to make automated inferences about the data; ontology definitions and constraints are represented in a formal language, and hence minimize the uncertainty of whether the inferences are valid vis-à-vis the assumptions and intentions of the data creator. Ontologies thus can be used *internal* to the organization for automated inference or *external* to the organization to support data sharing over the semantic Web.

The eXtensible Markup Language (XML) is a language designed for data sharing over the Web. As a standardized Web language, XML is very popular and many organizations have not only converted their data into XML for inter-organizational data sharing, but also as a commonly recognized data representational scheme for intra-organizational applications.

Usefulness of XML data can be embellished by applying ontologies to them: Ontologies can be used to answer queries on data structured in XML that would not otherwise be possible solely with XML-based query mechanisms [4]. There then is a research opportunity to investigate ontologies that:

(i) can primarily be used for knowledge extraction from a repository of XML documents

(ii) can secondarily be used for sharing XML data over the semantic Web

It is important to consider these two issues in *tandem*. Glushko, *et al.* [5] posit limitations of using ontologies in lieu of or even with XML, namely that it is impractical to develop formal ontology representations given that business needs change rapidly. Their criticism is that ontologies are complex, and hence require constant maintenance and more highly trained workers to develop than, say, XML-based systems. So developing ontologies for (i) may not be considered worthwhile. Ontologies' inherent complexity is also cited as a key inhibitor to the adoption of the semantic Web vision [6]. Who is going to build and maintain the numerous ontologies useable by industry—not "toy" ontologies developed for experimentation—that are needed to fulfil the vision if ontologies are so complicated? Therefore not enough useable ontologies may be developed for (ii).

It is stated that under certain situations complexity of ontology use and development is reduced, thus increasing the possibility that a semantic Web of ontologies will be adopted [6]: "Ontology developed by the knowledge worker is of use to the knowledge worker irrespective of whether it is used for data sharing; and there are ontology development tools that can be practically used by knowledge workers, not necessarily ontologists." A starting point is to realize that XML and ontology uses are not mutually exclusive." Smith and Coulter [7], for example,

discuss the use of ontologies with XML data for e-business applications. In particular, what if ontologies are used to answer queries on data structured in XML that would not otherwise be possible solely with XML-based query mechanisms? This way the ontology is of value in and of itself; value added by its use for data sharing can be considered a side-effect. Furthermore if tools to develop these XML-leveraging ontologies are used, they would reduce the complexity of representing ontologies. Finally if the XML data are more archival data, not transaction or real-time data, then the limitations of ontology use for fast-changing business are not greatly relevant. Under this scenario, XML-based ontologies have a good chance of being developed and posted for use on the semantic Web for data sharing in a similar vein as how KMS data are now shared with value-added partners over an extranet. Of course, a system based on such ontologies would ensure that company confidential data cannot be accessed from outside the organization. We believe that archival or historical data that are stored in an organization's knowledge management system (KMS) and structured in XML characterize a good example of such a scenario. There is a synergy between these two capabilities [(i) knowledge extraction from an XML repository, and (ii) data sharing over the semantic Web] under such a scenario; it may be worthwhile to build a system capable of both, but not worthwhile to build a system capable of only one.

While presenting the two synergistic, ontology based capabilities in this paper, we especially emphasize the ontology development methodology by presenting a thorough worked-through example of how to develop ontologies that are applied to extract knowledge from a Knowledge Management system's (KMS) XML repository. To supplement this emphasis, we also present our proof of concept application, *KROX* (Knowledge Retrieval using Ontologies and XML). The ontology and the application extract knowledge from a repository of Shakespearian

plays [8], which are represented in XML and available in the public domain. As the literature review in §2 details, arguably, none of the works that address both capabilities systematically present a repeatable ontological engineering methodology that others desiring to develop knowledge extracting ontologies could use. Addressing this deficiency is the aim of this paper: To explicate a repeatable methodology following a tutorial approach and using KROX as the example application. It is important to note that our work focuses on describing a methodology for manually building an ontology, which then can be used to automatically extract knowledge from a KMS. Though there is a natural language parsing technique employed to automatically structure some ontology representations in our work, we concentrate primarily on *ontology-based knowledge extraction*, not *automated ontology extraction* from a knowledge base.

The rest of the paper is organized as follows. In §3, the development of the Shakespeare ontology is presented, and in §4, the KROX application is detailed. In §5, we provide results of the computational performance of KROX and discuss achievements and limitations of our work. Finally in §5, concluding remarks and future work are stated.

2 Literature Review

Data sharing requires that there exists common understanding about semantics of the structured data. Common understanding can be enforced via use of an off-the-shelf library, e.g. xCBL™ [9], or standardized industry-based languages, e.g. Financial products Markup Language (FpML) [10]. These are examples of XML-based languages. Though many organizations have converted their KMS documents for representation in XML or one of these XML-based languages, these may not be fully utilized. These situations arise when definitions, rules, and assumptions about these documents are not represented with the documents; there is "money left on the table" as organizational knowledge is not sufficiently exploited. Using XML-

based languages requires that informal standards—i.e. standards that guide human developers, not enforce machine-encoded constraints—are applied to ensure proper interpretation of structured data. Contrast this way of enforcing common understanding to use of ontologies as explicit, formal representations of shared understanding [11].

As stated already, ontologies and XML can be used together. Ontologies about data structured using XML in existing documents can be organized and applied to support knowledge extraction and discovery. Domain-specific ontologies useful for focused intra-organizational knowledge management tasks and directly relevant to the data can be developed, and domain independent, generalizable ontology representations can be organized over time [12]. It would be these generalizable ontologies that can potentially be posted on the semantic Web and used for inter-organizational data sharing. For use of ontologies vis-à-vis XML, there are works that use XML as the language for representing or porting ontologies. Some examples are languages such as XOL (eXtensible Ontology Language) and to a small extent, OWL (Ontology Web Language) [13], and tools that support porting such as Protégé [14] and WebODE [15]. However a discussion of XML mediation to share ontologies is outside the scope of our research.

There are works that add semantics in the form of ontologies to XML documents. Erdmann and Studer [4] take an existing ontology and convert it into XML Data Type Definitions (DTD's), which then can be used to populate XML documents. Applying the same approach, ontologies can be converted into structural definitions to populate documents encoded in other languages such as HTML and SGML. Ontologies serve to mediate between disparate documents, say for documents posted on community Web portals [16]. There are other examples of ontology-based mediation to share XML data [17-19]. All these works fall in the broad area of mediated data integration [20-22]. However our scenario requires ontologies to be developed

from structures of existing XML documents, primarily for the task of extracting knowledge from an XML repository, *then secondarily* for data sharing.

There are many works related to ontology based knowledge extraction. In the Artequakt project [23], a natural language processing tool parses unstructured data from the Web, guided by an ontology that details what type of knowledge to harvest. The ontology enables describing names, places, and artefacts such that biographies of artists like Rembrandt can be constituted from different Web sources. S-CREAM [24] and MnM [25] projects also develop semi-automatic annotation tools to structure Web text, and use ontologies for this purpose. In many of these works, XML is used as an intermediary language to structure free text so that the ontology can be applied to the structured data. In these works, the systems architecture and exemplar extractions are laid out well, but the emphasis is not placed on how the ontology is developed. There are many ontological engineering methodologies [26, 27], but they do not provide detailed examples for developing ontologies for knowledge extraction.

Though some work in ontology-based knowledge extraction with emphasis on development methodology has been done by Kim [28], his work does not adequately address the secondary capability for data sharing. Conversely experiences of implementing a mediating ontology in OWL, the standardized ontology representation language for the semantic Web, to share XML data have been outlined [29].

Therefore a way to address the research opportunity is to elaborate on Kim's work [28] in development methodology of ontologies for knowledge extraction and extend it to encompass semantic Web data sharing capability by implementing Kim's ontology in the *de facto* semantic Web ontology languages for classes and objects (OWL) and definitions and constraints

(RuleML) [30]. This is the direction of this paper, and in the next section, the development of the ontology is detailed.

3 KROX Shakespearian ontology

This ontology is developed and presented using the methodology shown below [31].

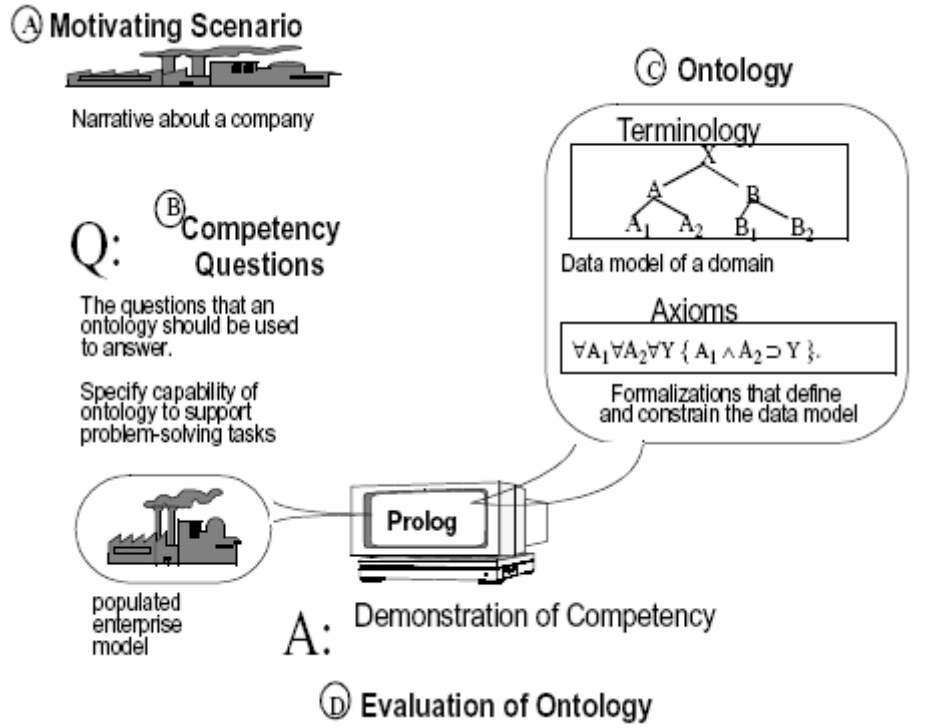

Figure 1: Overview of the TOVE Ontological Engineering Methodology

3.1 Motivating Scenario

The motivating scenario is a detailed narrative about problems faced or tasks performed by a user for which an ontology-based IS application is constructed. Here is *KROX*'s motivating scenario:

An organization with an IS-based KMS is studying the feasibility of migrating the system to an XML-based platform. This study entails prototyping an application for knowledge

extraction/query from existing XML documents to surmise development effort and cost/benefit before converting all documents to XML. A focus group is selected to provide requirements for, and ultimately test, the prototype application. Documents existing in public domain, specifically Shakespeare's plays, are chosen for the test. The reasons for this are the following: The focus group has read many plays, so can ably provide requirements; consistent XML element definitions exist, so an application broadly querying all Shakespeare's plays can be designed; and extending the application to query about other domains—e.g. plays in general or historical writings— may be possible. The focus group then provides requirements for an application more sophisticated than their current KM system, and general query and search engines. Specifically, they want this application, KROX to answer non-trivial queries, the kind that a student studying Shakespeare may ask.

3.2 Competency Questions - Informal

Users' requirements expressed as queries of an ontology-based application are competency questions. Since terms to pose formal queries in the ontology's language are not yet developed, these questions are inherently informal, asked in English using vocabulary and semantics familiar to users. The following are some competency questions for the Shakespearian play ontology used for *KROX*.

Who is Montague's son?

CQ 2. Who said "Et tu, brute!"?

CQ 3. Where does 'King Lear' take place?

CQ 4. Who are all the characters in 'Taming of the Shrew'?

3.3 Ontology

Terminology

The terminology of the ontology is comprised minimally of all terms required to formally express, but not answer, the competency questions. In turn, the expression that defines a given term is expressed using other ontology terms. Ultimately, a primitive ontology term is not defined, but sourced and mapped from a data repository. In presenting the terminology, the data scheme of the Shakespearian XML documents is presented pictorially, as a hierarchical model, then terminologically, as the ontology's primitive terms expressed as predicates. Next key terms and relationships in the informal competency questions are identified and integrated into pictorial (ER) and terminological (predicate) models.

Primitive Terms - Hierarchical Model.

```
        <TITLE>The Tragedy of Julius Caesar</TITLE>
```

```
<PLAY>
     <PERSONAE>
          <TITLE>Dramatis Personae</TITLE>
          <PERSONA>JULIUS CAESAR</PERSONA>
          ....
          <PGROUP>
                   <PERSONA>FLAVIUS</PERSONA>
                   <PERSONA>MARULLUS</PERSONA>
                   <GRPDESCR>tribunes.</GRPDESCR>
          ....
     </PERSONAE>
     <SCNDESCR>SCENE Rome: the neighbourhood of Sardis: the neighbourhood of
     Philippi.</SCNDESCR>
     ...
     <ACT>
          <TITLE>ACT I</TITLE>
          <SCENE>
                   <TITLE>SCENE I. Rome. A street.</TITLE>
                   ...
                   <SPEECH>
                            <SPEAKER>FLAVIUS</SPEAKER>
                            <LINE>Hence! home, you idle creatures get you home:</LINE>
                            <LINE>Is this a holiday? what! know you not,</LINE>
                            ...
                   </SPEECH>
     ...
     <PLAYSUBT>JULIUS CAESAR</PLAYSUBT>
...
</PLAY>
```

Figure 2: Use of Defined Elements within a Shakespeare's Plays in XML

Figure 2 shows an excerpt from the one of the XML documents that is in the collection. Figure 3 shows a representation of the hierarchy of the structural components of this collection. Given this hierarchical structure, a relationship structure between the components of plays can be

constructed. All relationships can be read top-to-bottom as *has*—i.e. a play *has* title, or play *has* act, which *has* title. Parent entities have children entities. Terminal nodes in the diagram correspond to elements that mark-up document content; i.e. they are attributes of an entity. Some terminal nodes are attributes that uniquely identify an entity. This model can then be expressed using predicates, which either relate an entity's unique identifier to its own attributes or child's unique identifier. In an ontology, predicates that are not formally defined are called primitive terms because they are populated through assertions, not by inference. So relationships between terminal nodes in the XML data structure correspond to primitive terms of the ontology.

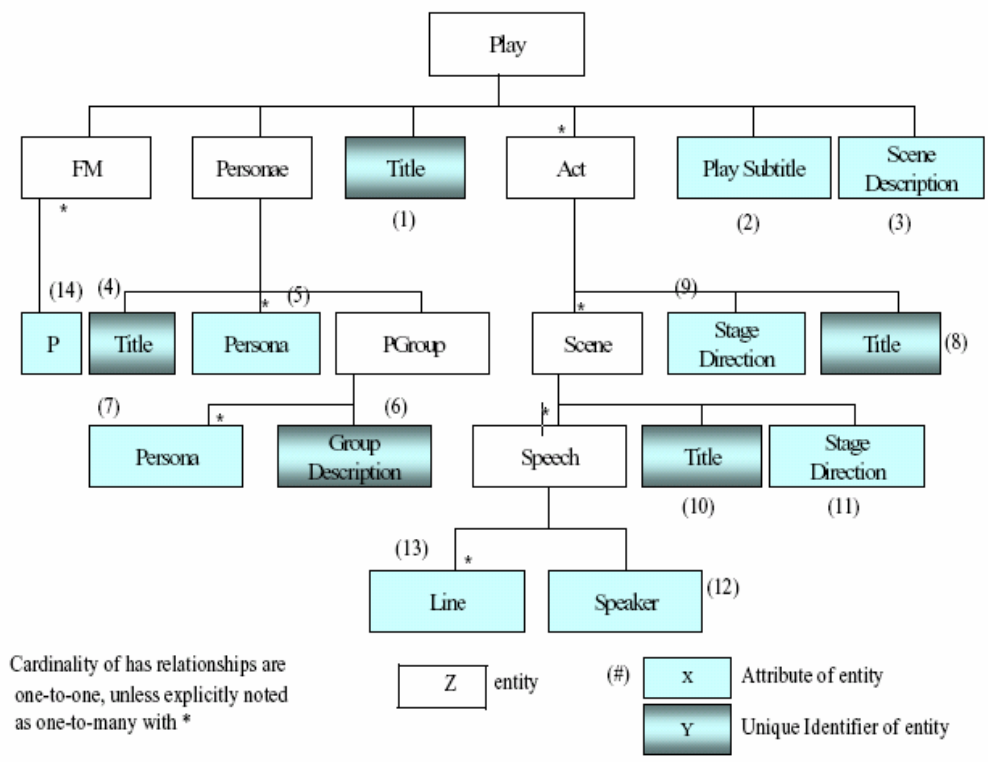

Figure 3: Hierarchical Structure of the Shakespearian Plays in XML

Primitive Terms - Predicate Model. Here is a description of primitive term variables. Variable numbers match those in Fig. 3[1].

(1) P: Name of the play; value of Play[2] → Title

(2) Pa: A character list, one descriptive name for section wherein all characters are introduced; value of Play → Personae →Title

Pe_1: Character description set of all characters individually introduced; value of Play → Personae → Persona

(3) Gd: Character description for each grouping of characters; value of Play → Personae →PGroup → Group Description

Pe_2: Character description set of all characters introduced within a given group; value of Play → Personae → PGroup → Persona

PT-1. **play_has_act(P,A)**

e.g. play_has_act('The Tragedy of Julius Caesar', 'ACT I').

play_has_subtitle(P,S)

e.g. play_has_subtitle('The Tragedy of Julius Caesar', 'JULIUS CAESAR').

play_has_character_list(P,Pa)

e.g. play_has_character_list('The Tragedy of Julius Caesar', 'Dramatis Personae').

character_list_has_character_description(Pa,Pe_1)

e.g. character_list_has_character_description('Dramatis Personae', 'JULIUS CAESAR').

character_list_has_group(Pa,Gd)

e.g. character_list_has_group('Dramatis Personae', 'tribunes.').

group_has_character_description(Gd,Pe_2)

e.g. group_has_character_description('tribunes.', 'FLAVIUS.').

Each primitive term can be populated by a query on XML documents. Take the question, "Given that 'Tragedy of Julius Caesar' is the title of the play, what is the play's subtitle?". Using First-Order Logic, the question is expressed using primitive terms as the following query:

{st | play_has_subtitle('The Tragedy of Julius Caesar', st)}.

[1] Greater detail to these ontology models is shown in
[28] H. M. Kim, "XML-hoo! A prototype application for intelligent query of XML documents using domain-specific ontologies," presented at 35th Annual Hawaii International Conference on Systems Science (HICSS-35), Hawaii, HI, 2002.

2 → denotes parent-of

Using XQuery [32], the question is expressed as follows, and returns the value, <playsubt>JULIUS CAESAR</playsubt>

```
for $p in doc("Julius_Caeser.xml")/play
where $p.title = "Tragedy of Julius Caesar"
return $p.playsubt
```

Though not explicitly structured using XML elements, there is an observed format for introducing characters, which applies with few exceptions. For instance, the value of the <PERSONA> element always starts with the character's name, and may be preceded by combinations of pseudonym, qualifiers, and statements of relationship with other characters.

```
<PERSONA>[1] ::= <description set>

<description set> ::=
    <character name>
    [ (<character psuedonym>) ]
    [ { , <qualification of character> }
      { , <relationship of character> }
      ; <description set> | <primitive description set>
    |
      , <description set> | <primitive description set>
    ] [.]

<GRPDESCR>[1] ::=
    <qualfication of character> |
    <relationship of character>
    [...unstructured text ].

<primitive description set> ::=
    <character name>[2]
    [ (<character psuedonym>) ]

<qualification of character> ::=
    [                 { <adjective> | <adverb> } ]
    <character title>
    [ {               <preposition> | <article> }
                      <location qualifier>
    ]

<relationship of character> ::=
    [                 <conjunction> ]
    <relation title>
    [ {               <preposition> | <article> } ]
                      <character related to>
    [
                      {   <conjunction>
                          <character related to> }
    ]

<relation title> ::=
<relation noun> <relation preposition>
```

```
e.g. in <PERSONA>REYNALDO,
servant to Polonius.</ PERSONA>
-   <character name> = REYNALDO
-   <relation noun> = servant
-   <relation preposition> = to
-   <characer related to> =
    Polonius

e.g. in <PERSONA>PARIS, a young
    nobleman,kinsman to the
    prince.
</PERSONA>
-   <character title> = nobleman
-   <relation noun> = kinsman
-   <relation preposition> = to
-   <characer related to> =
    prince

e.g. in <GRPDESCR>friends to
    Brutus and Cassius.</
    GRPDESCR>
-   <relation noun> = friends
-   <relation preposition> = to
-   <characer related to> =
    Brutus
-   <characer related to> =
    Cassius
```

Figure 4: Format for Persona and Group Descriptions, expressed in BNF notation

Obviously, the implementation to parse values within an element is not trivial and this parsing capability has been developed to a great, but not full, extent in the implementation. Following are the primitive terms, which express relationships between <PERSONA> or <GRPDESCR> and the primitive formats such as <character name> and <relation noun>.

Pe: Pe_1 (individual character description) or Pe_2 (description of individual characters described in a group description); value for <PERSONA> element

Pd: Description for one character; value for <primitive description set> format

D: Pd or Gd (group description: value for <GRPDESCR> element)

C: Just the name of the character; value for <character name> format

Ps: Pseudonym of the character; value for <character pseudonym> format

Qt: A qualifying title of a character, e.g. 'King'; value for <character title> format

Lq: A location that qualifies a character's title, e.g. 'King of Denmark'; value for <location qualifier> format

Cr: A character who is referenced when describing another character; value for <character related to> format

Rn: The noun describing the nature of the relation between a character and Cr, e.g. father; value for <relation noun> format

Rp: Relation preposition that qualifies Rn, e.g. 'of' in 'father-of'; value for <relation preposition> format

character_description_has_primitive_description_set(Pe,Pd)

e.g1. character_description_has_primitive_description_set('Senators, Citizens, Guards, Attendants' , 'Senators').

e.g2. character_description_has_primitive_description_set('CLAUDIUS, king of Denmark' , 'CLAUDIUS, king of Denmark').

PT-13. **primitive_description_set_has_character(Pd,C)**

e.g. primitive_description_set_has_character('CLAUDIUS, king of Denmark' , 'CLAUDIUS').

PT-14. **primitive_description_set_has_pseudonym(Pe,Ps)**

e.g. primitive_description_set_has_pseudonym('MARCUS ANTONIUS (ANTONY)' , 'ANTONY').

PT-15. **description_has_qualifying_title(D,Qt)**

e.g. description_has_qualifying_title('CLAUDIUS, king of Denmark' , 'king').

PT-16. **description_has_location_qualifier(D,Qt,Lq)**

e.g. description_has_location_qualifier('CLAUDIUS, king of Denmark' , 'king' , 'Denmark').

PT-17. **description_has_relationship(D,Rn,Rp,Cr)**

e.g1. description_has_relationship('REYNALDO, servant to Polonius.' , 'servant' , 'to' , 'Polonius').

e.g2. description_has_relationship('friends to Brutus and Cassius.' , 'friends' , 'to' , 'Brutus').

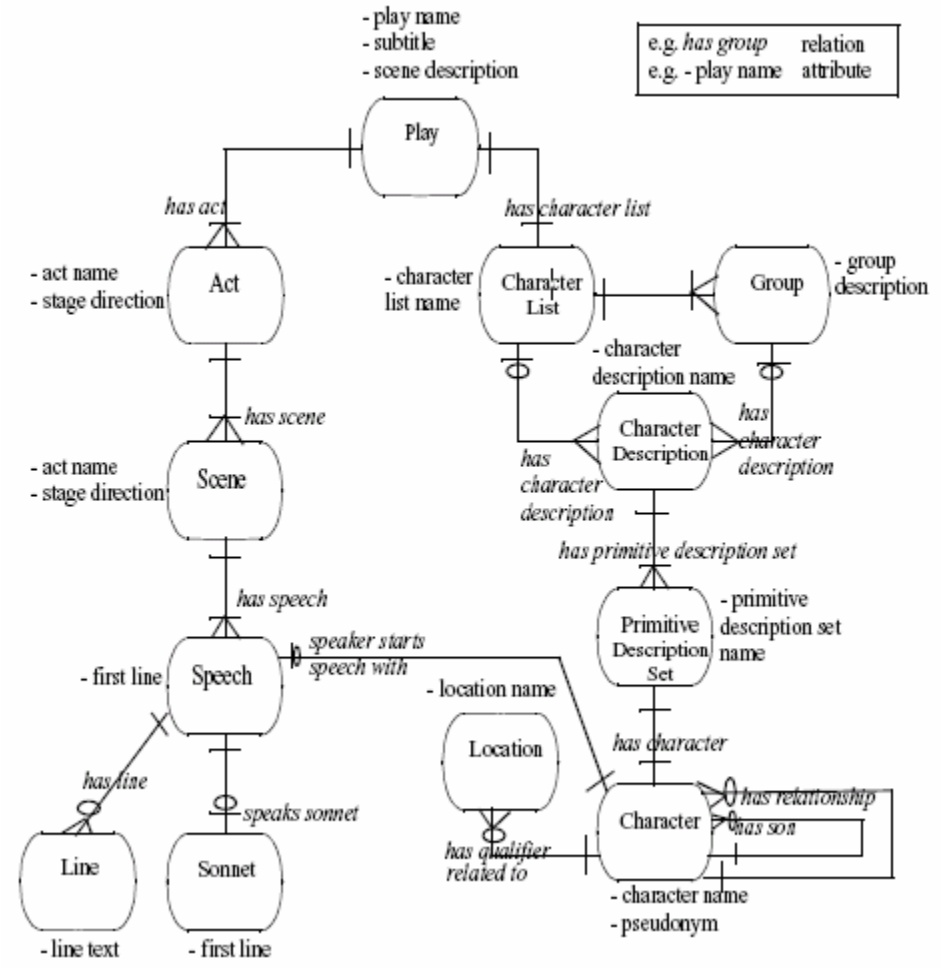

Figure 5: Revised Data Model (Represented using Entity-Relationships)

Ontology Data and Predicate Models. From the informal competency questions, the following key words are isolated: *who*, *son*, *said/utter*, *where... take place*, *characters*, *sonnet,* and *what... say*. Obviously, some of these words can be easily defined using the primitive terms as:

Pred-1. **character(C)**

Pred-2. **location(Lo)**

Pred-3. **speaker_starts_speech_with(Sp,L_1)**

These in turn are used to define ontology terms such as the following:

Pred-4. **play_has_character(P,C)**

Pred-5. **play_has_location(P,Lo)**

Pred-6. **has_son(C_1,C_2)**

Pred-7. **speaker_says(Sp,L)**

The data model presented in Figure 5 is an overall, graphical view presented as an ER model of Figures 3 and 4, and the primitive terms and predicates that have been so far presented.

Formal Competency Questions. Competency questions can now be posed, and formally expressed in First-Order Logic queries.Which character is the son of the Montague character?

{C | has_son('Montague',C) }Which speaker says the line, "Et tu, brute!"?

{S | speaker_says(S,'Et tu, brute!') }

CQ 2. What is the location for the play 'King Lear'?

{Lo | play_has_location('King Lear',Lo) }

CQ 3. Which are all the characters in the play 'Taming of the Shrew'?

{C | play_has_character('Taming of the Shrew',C) }

Each question corresponds to an axiom. To prove it, ontology axioms defining and constraining use of terms comprising the question axiom must exist. Such ontology axioms are presented next.

Ontology - Axioms[3]

To answer **CQ-1**, the following predicates are formally defined.

Defn-1. **character(C)**

The first part of a primitive description set for one character is the character's name.

character(C) = {C | ∃ *Pd* [primitive_description_set_has_character(*Pd*,C)}

Defn-2. **play_has_character_description(P,Pe)**

A character description Pe is either in a list of individual character descriptions, or contained within a list of group descriptions.

play_has_character_description(P,Pe) =
{*P, Pe* | ∃*Pa* [play_has_character_list(*P,Pa*) ∧
character_list_has_character_description(*Pa,Pe*) ∨
∃*Gd* (character_list_has_group(*Pa,Gd*) ∧
group_has_character_description(*Gd,Pe*)) }.

Defn-3. **play_has_character(P,C)**

A character name C is the first part of a primitive description set describing one character, which is part of a list of character descriptions

play_has_character(P,C) =
{*P, C* | ∃*Pe* ∃*Pd* play_has_character_description(*P,Pe*)
∧ character_description_has_primitive_description_set(*Pe,Pd*) ∧
primitive_description_set_has_character(Pd,C)

4 KROX Implementation

KROX is implemented using fully XML and semantic Web-standards compliant technologies to determine the feasibility of our ontology design, and not for the purpose of

[3] Only some of the axioms need to answer this competency question is shown in this section. The remaining axioms as well a walk-through of the answering of the competency question is shown in the appendix

processing efficiency. The experiments with this system aptly demonstrate the areas where technology needs to be improved in order to provide a consistent business solution for generalized processing of knowledge extracting ontologies.

4.1 Ontology-based implementation

We first demonstrate a standards-based implementation of the ontology and a processor for this application. The application is designed with potential data sharing in mind. Instead of building an application specifically for the Shakespeare ontology, this application is designed around an architecture that can easily adapt to different ontologies and data schema.

The system is built around a series of transformations. Because of the XML-oriented nature of the application, the focus of the architecture is to stay within the XML standards domain. We make extensive use of OWL, XSLT [33] and RuleML for this purpose. OWL is used primarily for representing the ontology, while XSLT is used to introduce the transformation logic that enables the proper translation of the XML data into appropriate RuleML markups for processing. RuleML is used for actual deduction logic to answer the queries.

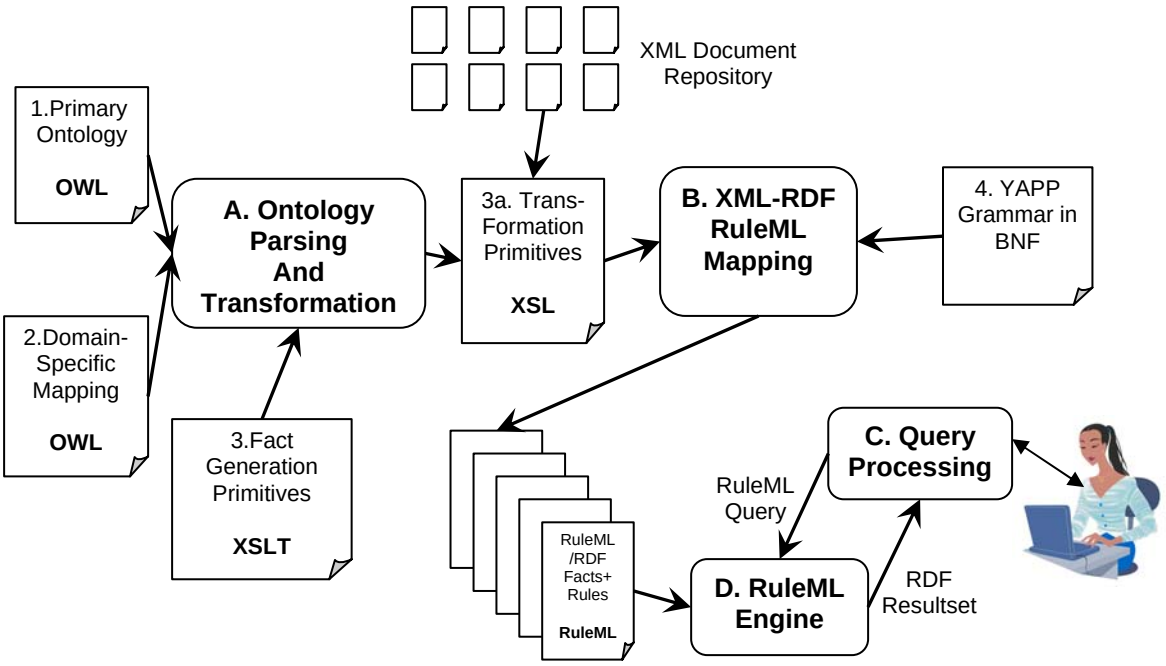

Figure 6. Architecture of KROX Ontology-based querying

Architecture

Figure 6 shows the architecture of the system. The input to the system is the set of XML files containing the data that needs to be searched. In addition, the system accepts two OWL resources for bootstrapping purposes, and RuleML queries to search the repository. The OWL resources can be created using a tool such as Ontobuilder [34]. The architecture is developed around four process components (A-D) and four data components (1-4) as shown in Figure 6. A description of each of these components is given below:

Data Components

1. The primary ontology. The primary ontology, in this case the Shakespeare ontology, is represented in OWL and is used as the primary meta-data description resource. In the current implementation, all the primitive terms and axioms of the ontology are implemented in OWL.

An excerpt from this document is given below, in which the basic header of the Shakespeare ontology and four predicates are declared.

```
<?xml version="1.0"?>
<rdf:RDF xmlns="http://www.joshsimerman.com/ontology/shakespeare-facts#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#">
     <owl:Ontology rdf:about="">
            <owl:versionInfo>Shakespeare 1.0</owl:versionInfo>
            <rdfs:label>Shakespeare Facts Ontology</rdfs:label>
            <rdfs:comment>This file defines the rules for generating facts from the
shakespeare xml documents.</rdfs:comment>
     </owl:Ontology>
     <owl:ObjectProperty rdf:ID="character">
            <rdfs:domain rdf:resource="#Character"/>
     </owl:ObjectProperty>
     <owl:ObjectProperty rdf:ID="hasCharacter">
            <rdfs:domain rdf:resource="#Play"/>
            <rdfs:range rdf:resource="#Character"/>
     </owl:ObjectProperty>
     <owl:ObjectProperty rdf:ID="hasAct">
            <rdfs:domain rdf:resource="#Play"/>
            <rdfs:range rdf:resource="#Act"/>
     </owl:ObjectProperty>
     <owl:ObjectProperty rdf:ID="hasSubtitle">
            <rdfs:domain rdf:resource="#Play"/>
            <rdfs:range rdf:resource="#SubTitle"/>
     </owl:ObjectProperty>
   ...
  </rdf:RDF>
```

2. Domain-specific mapping. Once the ontology is described, the process of mapping the XML documents into the Ontology must also be created. This allows the translation of the XML documents into corresponding OWL instances or another format suitable for searching. This is necessary because the XML data in the native format are not suitable for ontology-based searches. The mapping format is simple – for each ontology term, a corresponding XPath query is specified for extracting the corresponding elements from the XML documents. Often XPath cannot be used to completely define the term, and we intend to allow XQuery in the mapping process in our future work. However, for all the tested predicates in the system, XPath seemed to suffice. An excerpt from this mapping document is shown below.

```
<?xml version="1.0"?>
<rdf:RDF xml:lang="en" xmlns="http://www.joshsimerman.com/ontology/shakespeare-
structure#" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
xmlns:jms="http://www.joshsimerman.com/ontology/shakespeare-rdf#">
```

```
    <owl:Ontology rdf:about="">
           <owl:versionInfo>Shakespeare 1.0</owl:versionInfo>
           <rdfs:label>Shakespeare Structure Ontology</rdfs:label>
           <rdfs:comment>This file defines the hierarchy structure of the
shakespeare xml documents.</rdfs:comment>
    </owl:Ontology>
    <owl:Class rdf:ID="Play">
           <jms:term entity="PLAY">P</jms:term>
           <rdfs:comment>Name of the play</rdfs:comment>
           <rdfs:label xml:lang="en">play</rdfs:label>
           <jms:path useBaseDoc="true">PLAY/TITLE</jms:path>
    </owl:Class>
    <owl:Class rdf:ID="SubTitle">
           <jms:term>S</jms:term>
           <rdfs:comment>Subtitle of play</rdfs:comment>
           <jms:path useBaseDoc="true">PLAY/PLAYSUBT</jms:path>
    </owl:Class>
...
</rdf:RDF>
```

The above XML representations show how a general OWL ontology is augmented by domain-specific path constructs that aid in the translation of XML data into processable RuleML.

3. Fact generation primitives. The fact generation primitives consist of a set of XSLT templates which can be used for the purpose of generating the final transformation stylesheet. The fact-generation primitives use the general ontology and the mapping ontology to generate a transformation process using newly generated XSLT primitives that assist in translating the source XML documents into parsable XML-structured ontology form. This XSLT is a generic XSLT, and does not change for different ontologies. An excerpt from this XSLT document, demonstrating how the OWL ontology and the corresponding mapping primitives are used to generate the relation structures, is shown below:

```
<xsl:stylesheet xml:lang="en" xmlns:xjs="urn:out" version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
syntax-ns#" xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:jms="http://www.joshsimerman.com/ontology/shakespeare-rdf#">
  <xsl:namespace-alias stylesheet-prefix="xjs" result-prefix="xsl"/>
  <xsl:output method="xml" indent="yes" encoding="UTF-8"/>
  <xsl:param name="ClassID" select="document('ontStructure.owl')"/>
  <xsl:template match="/">
    <xjs:stylesheet version="1.0">
      <xjs:output method="xml" indent="yes" encoding="UTF-8"/>
      <xjs:param name="fileList" select="document('file-list.xml')"/>
      <xjs:template match="/">
        <prologs>
          <xsl:apply-templates/>
        </prologs>
      </xjs:template>
      <xjs:template match="node()" mode="multiple">
```

```
        <xjs:param name="subject"/>
        <xjs:param name="term"/>
        <xjs:for-each select=".">
          <prolog>
              <relation>
                <xjs:value-of select="$subject"/>
              </relation>
              <argument>
                <xjs:value-of select="$term"/>
              </argument>
              <xjs:if test="string(.) != $term">
                <argument>
                  <xjs:value-of select="normalize-space(.)"/>
                </argument>
              </xjs:if>
          </prolog>
        </xjs:for-each>
      </xjs:template>
    </xjs:stylesheet>
  </xsl:template>
```

3a. Transformation primitives. The transformation primitives component is simply an automatically generated XSLT stylesheet which can be applied directly to the XML repository to generate the XML structure. This stylesheet is generated by the Ontology Parsing and Transformation process (A), which will be discussed shortly.

4. YAPP Grammar. Although most of the primary translation can be performed using direct XSLT, some of the more advanced textual transformation and parsing cannot be performed by XSLT because of the lack of an advanced string processing and adequate computational capabilities of XSLT. For example, the derivation of the relationship structure in the Shakespeare ontology from the specific structure of the persona component cannot be performed in XSLT, so more advanced XML parser generators, such as YAPP, are necessary. The BNF for the Shakespeare ontology has been discussed earlier in Section 2.

Process Components

A. Ontology Parsing and Transformation. The ontology parsing and transformation stage performs two primary transformations. First, the domain-specific mapping is processed using the fact-generation primitives to generate a set of XSLT transformation templates – which are then

used to transform the primary ontology into an XML representation of relations. The result of this step is a set of simplified XML documents containing the normalized XML data represented as relations. An excerpt of this temporary XML structure is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<prologs                                  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"     xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:jms="http://www.joshsimerman.com/ontology/shakespeare-rdf#">
<prolog>
<relation>character</relation>
<argument>Another Poet</argument>
</prolog>
<prolog>
<relation>character</relation>
<argument>ARTEMIDORUS Of Cnidos</argument>
</prolog>
…
<relation>hasCharacter</relation>
<argument>The Tragedy of Julius Caesar</argument>
<argument>ARTEMIDORUS Of Cnidos</argument>
</prolog>
<prolog>
<relation>hasCharacter</relation>
<argument>The Tragedy of Julius Caesar</argument>
<argument>A Soothsayer</argument>
</prolog>
…
<relation>hasSceneDescription</relation>
<argument>The Tragedy of Julius Caesar</argument>
<argument>SCENE Rome: the neighbourhood of Sardis: the neighbourhood of Philippi.</argument>
</prolog>
<prolog>
<relation>hasSceneDescription</relation>
<argument>The Tragedy of Romeo and Juliet</argument>
<argument>SCENE Verona: Mantua.</argument>
</prolog>
…
<prolog>
<relation>hasCharacterDescription</relation>
<argument>Dramatis Personae</argument>
<argument>JULIUS CAESAR</argument>
</prolog>
<prolog>
<relation>hasCharacterDescription</relation>
<argument>Dramatis Personae</argument>
<argument>ARTEMIDORUS Of Cnidos, a teacher of rhetoric.</argument>
</prolog>
…
```

B. XML/RDF to RuleML Mapping. In this stage, the temporary XML relation structure is processed by the RuleML translation XSLT, to generate the final RuleML equivalent of the facts represented in the original documents. The output of this process is a set of RuleML documents which include all the rules in the ontology in RuleML format, as well as the facts generated from the source XML document through the process of parsing.

C and D. Query Processing and RuleML Engine. The RuleML generated by the mapping step can be used for processing with a suitable RuleML engine. In our experiments, we used a local installation of Michael Sintek's RuleML engine [35]. In the current implementation, we use a command line interface where user queries are specified using the Competency question predicates (see CQ-1 through CQ-5), and the system responds by providing all matches for the free variables in the query.

To summarize the architecture as shown in Figure 6, the basic flow of information in the system follows a one-time mapping process that translates the XML documents into RuleML/RDF for rule-based processing. The whole mapping process can be sub-divided into two phases. In the first phase, the OWL ontology and the mapping resources are used along with a fact-generation stylesheet (written in XLST) for the ontology parsing and transformation. The output of this stage is another XSLT, which is domain specific with the knowledge of specific document structures. This XSLT, along with a set of BNF grammar rules using the YAPP XML parser generator, is used in the second stage of transformation, which takes the input XML documents and generates the RuleML/RDF facts and rules for rule-based query processing. These RuleML facts and rules are then used by the RuleML engine for answering queries. User queries are translated into RuleML queries and fed to the RuleML engine, and the RDF result set obtained from the RuleML engine is then transformed back for the user.

5 Results & Discussion

5.1 Experimental Results

The KROX implementation is tested by generating the RuleML maps for all the XML documents in the Shakespeare collection. Here we provide some results of timing the different mapping and querying processes. The data are generated and queried using a Pentium M 1.4Ghz laptop with 512MB RAM, so the times could be significantly improved using a faster machine. Note that the architecture is prone to several levels of optimization for fast processing, which is intended to be a future work in this project.

File	*Size*	*Stage 1*	*Stage 2*	*CQ-1*	*CQ-2*	*CQ-3*	*CQ-4*
j_ceaser.xml	190k	23s	8s	0.14s	0.13s	0.15s	0.12s
r_and_j.xml	225k	23s	9s	0.15s	0.14s	0.14s	0.11s
hamlet.xml	290k	23s	11s	0.18s	0.15s	0.21s	0.14s
j_ceaser+ r_and_j	415k	23s	15s	0.21s	0.12s	0.24s	0.13s
All XML	8MB	23s	97s	4.48s	3.14s	0.85s	2.35s

Table 1. Experimental results (Timing)

The experimental results show that for moderately sized document repositories, this architecture is stable. The querying time is fast for small files, but the XSB-based RuleML engine used in the experiments does not have any caching capabilities, so queries perform slowly for the full repository. Note, however, that the mapping process need only be performed once, and queries can be repeatedly executed once the RuleML facts are generated. However the system can certainly be improved by using a better rule engine, and optimizing the performance using a database system for processing queries that does not require any special axiom processing. Answering CQ-1 consistently takes more time than the rest of the queries, likely because CQ-1 needs the use of the relation extraction primitives. However, from the perspective of a proof-of-concept system, the results are quite satisfactory.

5.2 Discussion

We have achieved the following technical milestones in this paper, thus making a contribution to practice by presenting a worked-through and repeatable example.

- An ontological engineering methodology is followed to state the motivating scenario and competency questions [36]. The hierarchical structure for a common set of XML documents, namely Shakespearian plays, is translated to develop primitive terms—ontology predicates that are populated by look-ups into an XML document, rather than inferred using formal definitions.

- Additional structure within an element is discerned; e.g. there is a fomat for character introductions that holds with few exceptions, which applies to the <PERSONA> element. This structure is translated to develop more primitive terms. We employ a simple natural language parsing technique to discern primitive terms; this discernment constitutes the ontology extraction facet of our otherwise knowledge extraction research.

- Ontology predicates are identified from competency questions and ensured for consistency with primitive terms. This is sufficient to express the competency questions formally in the language of the ontology using predicates.

- Axioms that define meanings of predicates or constrain their interpretation are developed. By applying ontology axioms to populated primitive terms, answers to competency questions are inferred.

- We demonstrate a standards-based implementation of the ontology and a processor for this application, which is designed with potential data sharing in

mind using an architecture that can easily adapt to different ontologies and data schema. We develop a system that is built on a series of transformations. Because of the XML-oriented nature of the application, the focus of the architecture is to stay within the XML standards domain using standard languages like OWL, XSLT, and RuleML.

- The primitive terms and axioms of the ontology are represented in OWL, so the ontology is used as the primary meta-data description resource. OWL instances are then transformed into XSLT process logic, which is necessary because the XML data in the native format are not suitable for ontology-based searches. When this logic is applied to XML data, facts and rules are now all represented in RuleML/RDF, whence query processing (inference) are performed using a RuleML engine.

- The computational performance of the KROX implementation is tested by generating the RuleML maps for all the XML documents in the Shakespeare collection. For example, the 'has son' query returns a value in on average 0.15 s. Other similar experimental results show that for moderately sized document repositories, this architecture is stable.

Admittedly, our work provides a methodology for manually developing high-quality ontologies used to automate the extraction of knowledge, not the automated extraction of the ontology itself. We do however employ some natural language parsing techniques to parse through free text to instantiate, for example, familial relationships that are buried within the free text so there are facets to automated ontology extraction in our work. There are good works that

do present techniques for semi-automatic ontology extraction (also sometimes called ontology learning) from corporate intranet documents [37-39], and we believe that our research is very complementary to such works.

One of the compelling arguments for semi-automatic ontology extraction is that it is generalizable; that is, the technique is capable of producing initial ontologies that can then be refined by knowledge workers largely irrespective of the knowledge domain. This approach is semi-automatic rather than automatic, because such initial ontologies as is would seldom meet the needs of the knowledge workers. Therefore manual ontological engineering will always be necessary, and a blueprint for doing this will always be helpful. Especially if the task for ontology use is knowledge extraction from an XML data repository where the data are more unstructured and dynamic than the exemplar dataset used in this paper, employing an ontology extraction technique and then refining the ontology and adding semantics using the methodology and example-based guidance detailed herein would be superior to employing either method alone. Ontology extraction is essential because of generalizability of application and flexibility to deal with unstructured and dynamic data; a manual ontological engineering methodology is essential to give guidance to the knowledge worker to refine and organize initial representations, some of which are likely to be non-sense, to some purposeful set of representations. Therefore the contribution of our work can be presented juxtaposed to ontology extraction using Fox's criteria for evaluating ontologies [40]: Inasmuch as we believe that automation characteristic of ontology extraction techniques leads to *efficiency* of ontology engineering and *scalability* of ontology-based applications, manual intervention guided by vetted guidelines as presented in this paper leads to *competence* ("How well does an ontology support problem solving?) of the ontology representations. According to Fox, Uschold and Gruninger [41][42], ontological

engineering methodologies driven by formal competency questions—as our work is—is re-useable, capable of generalizably being applied for manual ontology construction of any domain. Though the example presented in this paper is not of an automatically extracted ontology, it is very reasonable that the methodology can be used to refine an automatically extracted initial ontology.

Another way to juxtapose the contributions of our work relative to the contributions of the automated extraction paradigm is as follows. Doan's work [43] is a good example of knowledge sharing as ability to extract from and integrate with various data sources. In the same vein as the ability to add semantics and guidance for manual ontological engineering complements ontology extraction for knowledge extraction, so can the work presented here complement Doan's automated approach by facilitating manual organization of the knowledge that is automatically extracted from disparate sources for the intent of sharing.

When the intent is to have knowledge extraction *and* knowledge sharing of more structured and static archival XML data in tandem, then the KROX approach can be exclusively employed by knowledge workers of the organization which owns the data, primarily for extraction and secondarily, as a by-product, for facilitating sharing. However having extraction and sharing in tandem is not essential. When the focus is solely knowledge extraction of unstructured and dynamic data, then the KROX approach can be used by knowledge workers within the organization where data originated to refine an automatically extracted ontology. When the focus is solely knowledge sharing of XML data from different sources, then the KROX approach can be used by knowledge workers doing the integration—i.e. not the organization where data originated—to refine an automatically integrated ontology. Therefore there are three distinctly different scenarios to which the KROX approach can add value.

We also recognize that the presentation of the methodology may not be prescriptive enough for some. For instance, why did we choose to include a predicate called “character has location,” or draw the relationship between speech and character. Rest assured careful attention went into making design choices like “What is a good name for the predicate that answers the first competency question, what should be its arguments, and why is one formal definition for it better than another?” However in the interest of presenting a worked-through example from the motivating scenario to experiments on the computational performance and touching upon every step in between, we do not address such important, yet very domain specific system design questions. Once again it is because making these design choices requires extensive domain knowledge that manual engineering of ontologies is useful whether or not preceded by automatic ontology extraction.

Though manually developed, there is still efficiency gained from ontology use. As long as there is a sufficient volume of XML data to which ontology representations can be applied for query answering, the time and effort to manually develop ontologies can be justified. The Shakespeare Ontology, for example, can be applied to not only Shakespeare plays but potentially other plays. Similarly a domain ontology developed for a KMS may have more than a very narrow scope. Granted, an ontology developed using generalizable procedures outlined for automatic ontology extraction would have a wider scope and hence be more scalable. However this enhanced scalability must be weighed against the greater precision and recall, characteristic of an ontology specifically developed for a domain that has been vetted by a domain expert and/or an ontologist. Admittedly, we do not have experimental data on scalability versus recall and precision however since our experimentation was specifically for computational performance for data sharing rather than usability of system use. This is a purposeful decision

that we as systems designers made because the question of whether multitude of Web-based languages and tools could be combined to work to demonstrate the feasibility of our work in an implementation was the technical question to which we turned our energies rather than the question of evaluating usability. We do recognize that this is a limitation of work and hope to address it in the enhanced version of KROX.

6 Concluding Remarks and Future Work

The capability of the ontology to support inference of facts not explicitly structured in XML demonstrates that an ontology-based approach to query answering is a natural complementary function for an XML data repository. For the Shakespeare example, familiar relationships are not structured in XML. Plus, nowhere is it explicitly nor parametrically represented that Montague is Romeo's father. Yet, this can be reasoned in KROX. So by developing KROX, we demonstrate how an ontology that can be used to extract knowledge from an XML repository can be developed. By implementing the ontology in an architecture using XML and semantic Web compliant standards technologies, we show how such an ontology can be prepared for use for data sharing over the semantic Web. *So our work makes a technical contribution to practice*. There are few works that address both these capabilities. And none of these works are able to do the following: By systematically demonstrating a development methodology from the initial description of the motivating scenario to a final evaluation of the computational performance of the ontology for use in data sharing, we provide a blueprint for others desiring to do something similar. *Since we address this deficiency, our work can be characterized as a novel contribution as well.*

It has been predicted that firms that will be early adopters of the semantic Web will develop ontologies that leverage XML, provide intra-organizational value such as knowledge extraction capabilities that are irrespective of the semantic Web, and have the potential for inter-organizational data sharing over the semantic Web [6]. We hope that KROX will serve as a blueprint for other ontology developers who believe that the growth of the semantic Web will unfold in this manner.

In keeping with our desire for KROX to serve as a blueprint for this prediction, we concentration our future work towards the one aspect of the prediction that we do not address thoroughly. The prediction states that early adoption semantic Web ontologies will be developed by knowledge workers, not necessarily just ontologists. Yet, a tool for such development by users is not really a part of *KROX*. There are popular tools like Protégé [14] out there, but we will explore the research opportunity for specialized tools to develop knowledge extraction and data sharing ontologies that work with XML repositories. Concomitant to this are future work related to architecture and implementation already identified such as more accurate routines for parsing within XML markups, XPath processing capability, and query optimization.

References

[1] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific American*, vol. 284, pp. 34-43, 2001.

[2] Y. Wand and R. Weber, "Towards a Theory of Deep Structure of Information Systems," *Journal of Information Systems*, pp. 203-23, 1995.

[3] A. E. Campbell and S. C. Shapiro, "Ontological Mediation: An Overview," *Proceedings of IJCAI Workshop on Basic Ontological Issues in Knowledge Sharing*, Menlo Park, CA, 1995.

[4] M. Erdmann and R. Studer, "How to structure and access XML documents with ontologies," *Data and Knowledge Engineering*, vol. 36, pp. 317-335, 2001.

[5] R. J. Glushko, J. M. Tenenbaum, and B. Meltzer, "An XML Framework for Agent-based E-commerce," *Communications of the ACM*, vol. 42, pp. 106+, 1999.

[6] H. M. Kim, "Predicting how the semantic Web will evolve," *Communications of the ACM*, vol. 45, pp. 48-54, 2002.

[7] H. Smith and K. Poulter, "Share the Ontology in XML-based Trading Architectures," *Communications of the ACM*, vol. 42, pp. 1999.

[8] J. Bosak, "The plays of Shakespeare," http:// http://www.oasis-open.org/cover/bosakShakespeare200.html, Open Oasis.org, 1999 (Last Updated: July).

[9] CommerceOne, "xCBL.org: XML Common Business Library," Commerce One, Inc.,

Pleasanton, CA, 2003.

[10] ISDA, "FpML™: The XML Standard for Swaps, Derivatives, and Structured Products," http://www.fpml.org, International Swaps and Derivatives Association, 2004 (Last Updated: November 19).

[11] T. R. Gruber, "Towards Principles for the Design of Ontologies Used for Knowledge Sharing," *Proceedings of the International Workshop on Formal Ontology*, Padova, Italy, 1993.

[12] H. M. Kim, "Integrating Business Process-Oriented and Data-Driven Approaches for Ontology Development," *Proceedings of the AAAI Spring Symposium Series 2000 - Bringing Knowledge to Business Processes*, Stanford, CA, 2000.

[13] I. Horrocks, P. F. Patel-Schneider, and F. v. Harmelen, "From SHIQ and RDF to OWL: The making of a web ontology language," *Journal of Web Semantics*, vol. 1, pp. 7-26, 2003.

[14] N. F. Noy, M. S. Decker, M. Crubezy, R. W. Fergerson, and M. A. Musen, "Creating Semantic Web Contents with Protégé-2000," *IEEE Intelligent Systems*, vol. 16, pp. 60-71, 2001.

[15] J. C. Arpírez, O. Corcho, M. Fernández-López, and A. Gómez-Pérez, "WebODE in a nutshell," *AI Magazine*, vol. 24, pp. 37-47, 2003.

[16] A. Maedche, S. Staab, R. Studer, Y. Sure, and R. Volz, "SEAL - Tying Up Information Integration and Web Site Management by Ontologies," *IEEE Computer Society Data*

Engineering Bulletin, vol. 25, pp. 10-17, 2002.

[17] S. Philippi and J. Kohler, "Using XML technology for the ontology-based semantic integration of life science databases," *IEEE Transactions on Information Technology in Biomedicine*, vol. 8, pp. 154-160, 2004.

[18] B. Amann, C. Beeri, I. Fundulaki, and M. Scholl, "Querying XML Sources Using an Ontology-Based Mediator," *Lecture Notes in Computer Science*, vol. 2519, pp. 429 - 448, 2002.

[19] V. Christophides, G. Karvounarakis, I. Koffina, G. Kokkinidis, A. Magkanaraki, D. Plexousakis, G. Serfiotis, and V. Tannen, "The ICS-FORTH SWIM: APowerful Semantic Web Integration Middleware," *Proceedings of the First International Workshop onSemantic Web and Databases (SWDB)*, Humboldt-Universitat, Berlin, Germany, 2003.

[20] A. Tomasic, L. Raschid, and P. Valduriez, "Scaling Access to Heterogeneous Data Sources with DISCO," *IEEE Transactions on Knowledge and Data Engineering*, vol. 10, pp. 808-823, 1998.

[21] G. Wiederhold, "Mediation in information systems," *ACM Computing Surveys*, vol. 27, pp. 265-267, 1995.

[22] C. H. Goh, S. Bressan, S. Madnick, and M. Siegel, "Context interchange: New features and formalisms for the intelligent integration of information," *ACM Transactions on Information Systems*, vol. 17, pp. 270-293, 1999.

[23] H. Alani, S. Kim, D. E. Millard, M. J. Weal, W. Hall, P. H. Lewis, and N. R. Shadbolt, "Automatic Ontology-Based Knowledge Extraction from Web Documents," *IEEE Intelligent Systems*, vol. 18, pp. 14-21, 2003.

[24] S. Handschuh, S. Staab, and F. Ciravegna, "S-CREAM - Semi-Automatic Creation of Metadata," *Lecture Notes in Computer Science*, vol. 2473, pp. 358-372, 2002.

[25] M. Vargas-Vera, E. Motta, J. Domingue, S. B. Shum, and M. Lanzoni, "Knowledge Extraction by using an Ontology-based Annotation Tool," *Proceedings of First International Conference on Knowledge Capture, (K-CAP'01)*, Victoria, B.C.,Canada, 2001.

[26] A. Gómez-Pérez, M. Fernández-López, and O. Corcho, *Ontological Engineering with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*: Springer, 2004.

[27] Y. Sure, M. Erdmann, J. Angele, R. Studer, S. Staab, and D. Wenke, "OntoEdit: Collaborative Ontology Development for the Semantic Web," *Lecture Notes in Computer Science*, vol. 2342, 2002.

[28] H. M. Kim, "XML-hoo! A prototype application for intelligent query of XML documents using domain-specific ontologies," *Proceedings of 35th Annual Hawaii International Conference on Systems Science (HICSS-35)*, Hawaii, HI, 2002.

[29] P. Lehti and P. Fankhauser, "XML data integration with OWL: experiences and challenges," *Proceedings of International Symposium on Applications and the Internet*, Fraunhofer Inst., Darmstadt, Germany, 2004.

[30] H. Boley, S. Tabet, and G. Wagner, "Design Rationale of RuleML: A Markup Language for Semantic Web Rules," *Proceedings of First Semantic Web Working Symposium (SWWS'01)*, Stanford, CA, 2001.

[31] H. M. Kim, M. S. Fox, and M. Grüninger, "An Ontology for Quality Management: Enabling Quality Problem Identification and Tracing," *BT Technology Journal*, vol. 17, pp. 131-9, 1999.

[32] S. Boag, D. Chamberlin, M. Fernandez, D. Florescu, J. Robie, and J. Simeon, "XQuery 1.0: An XML Query Language - W3C Working Draft 29 October 2004," http://www.w3.org/tr/xquery, W3C, 2004 (Updated: October 29).

[33] J. Clark, "XSL Transformations (XSLT) Version 1.0," http://www.w3.org/tr/xslt, W3C, 1999 (Updated: November 16).

[34] G. Modica, A. Gal, and H. Jamil, "The use of machine-generated ontologies in dynamic information seeking," *Proceedings of Cooperative Information Systems (CoopIS '01)*, Trento, Italy, 2001.

[35] M. Sintek, M. Junker, L. Elst, and A. Abecker, "Using Information Extration Rules for Extending Domain Ontologies," Proceedings of IJCAI-2001 Workshop on Ontology Learning, Seattle, 2001.

[36] L. Narens, *Abstract Measurement Theory*. Cambridge, MA: MIT Press, 1985.

[37] J.-U. Kietz, A. Maedche, and R. Volz, "A Method for Semi-Automatic Ontology Acquisition from a Corporate Intranet," *Proceedings of EKAW'00 Workshop on*

Ontologies and Text, Juan-Les-Pins, France, 2000.

[38] D. Faure and C. Nedellec, "Knowledge acquisition of predicate-argument structures from technical texts using machine learning," presented at EKAW, Dagstuhl Castle, Germany, 1999.

[39] C. Brewster, F. Ciravegna, and Y. Wilks, "User-Centred Ontology Learning for Knowledge Management," *Lecture Notes in Computer Science*, vol. 2553, pp. 203-7, 2002.

[40] M. Fox, F. Fadel, and J. Chionglo, "A Common-Sense Model of the Enterprise", *Proceedings of the Industrial Engineering Research Conference*, Atlanta, GA, 1993.

[41] M. Gruninger and M.S. Fox, “The Role of Competency Questions in Enterprise Engineering”, *Proceedings of the IFIP WG5.7 Workshop on Benchmarking - Theory and Practice*, Trondheim, Norway, June 1994.

[42]

[43] W. Shen, X. Li, and A. Doan, "Constraint-Based Entity Matching," *Proceedings of the American AI Conference (AAAI-05)*, Pittsburgh, PA, July 2005.

Appendix

Defn-4. **character_has_pseudonym(C,Ps)**

A primitive description set describing one character can have both the character's name and pseudonym used.

character_has_pseudonym(C,Ps) =
{ *C, Ps* I∃ *Pd* [primitive_description_set_has_character(*Pd,C*) ∧
primitive_description_set_has_psuedonym(Pd,Ps) }

Defn-5. **a) related_characters(C_1,Rn,Rp,C_2)**

C_1 has a relationship, expressed as relation noun(Rn)+preposition(Rp), with C_2, if:

- C_1 and C_2 are characters in the same play.

- C_1 is explicitly stated as related to C_2 or C_2's pseudonym, and

- C_1 is a character introduced individually, or is any of the characters in a group that has a relationship to C_2, and

related_characters(C_1,Rn,Rp,C_2) =
{ C_1, *Rn, Rp,* C_2 |
(∃P (play_has_character(P,C_1) ∧ play_has_character(P,C_2)) ∧
(∃*D* (description_has_relationship(*D,Rn,Rp,*C_2) ∨
∃*Cr* (description_has_relationship(*D,Rn,Rp,Cr*) ∧
character_has_pseudonym(C_2,Cr))) ∧
(primitive_description_set_has_character(D,C_1) ∨
∃*Pe*∃*Pd* (group_has_character_description(*D,Pe*) ∧
character_description_has_primitive_description_set(Pe,Pd) ∧
primitive_description_set_has_character(Pd,C_1))) }

Defn-6. **b) related_characters(C_1,Rn,Rp,C_2)**

C_1 has a relationship, expressed as relation noun(Rn)+preposition(Rp), with C_2 if:

- C_1 is a pseudonym for a character whose relationship with C_2 can be inferred, or

- C_2 is a pseudonym for a character whose relationship with C_1 can be inferred, or

- C_1 and C_2 are pseudonyms for characters whose relationship with each other can be inferred.

related_characters(C_1,Rn,Rp,C_2) =
{ C_1, Rn, Rp, C_2 | $\exists C_a \exists C_b$
(related_characters(C_1,Rn,Rp,C_b) $\wedge$ character_has_pseudonym(C_b,C_2)) $\vee$
(related_characters(C_a,Rn,Rp,C_2) $\wedge$ character_has_pseudonym(C_a,C_1)) $\vee$
(related_characters(C_a,Rn,Rp,C_b) $\wedge$ character_has_pseudonym(C_a,C_1) $\wedge$
character_has_pseudonym(C_b,C_2) }

Defn-7. **a) may_be_related_characters(C_1,Rn,Rp,C_2)**

C_1 *may have* a relationship, expressed as relation noun(Rn)+preposition(Rp), with C_2, if:

- C_1 and C_2's relationship (Rn+Rp) cannot be inferred for sure, and

- C_1 and C_2 are characters in the same play, and

- C_1 is explicitly stated as related to C_2's qualifying title or location qualifier, and

- C_2 is a character introduced individually, or is any of the characters in a group, and

- C_1 is a character introduced individually, or is any of the characters in a group that has a relationship to C_2.

may_be_related_characters(C_1,Rn,Rp,C_2) =
{ C_1, Rn, Rp, C_2 | $\neg$ related_characters(C_1,Rn,Rp,C_2) $\wedge$
($\exists$P play_has_character(P,C_1) _ play_has_character(P,C_2)) $\wedge$
($\exists$C $\exists$D $\exists D_2$
(description_has_relationship(D,Rn,Rp,C) $\wedge$
(description_has_qualifying_title(D_2,C) $\vee$ description_has_location_qualifier(D_2,C)) $\wedge$
(primitive_description_set_has_character(D_2,C_2) $\vee$
($\exists Pe_2 \exists Pd_2$ (group_has_character_description(D_2,Pe_2) $\wedge$
character_description_has_primitive_description_set(Pe_2,Pd_2) $\wedge$
primitive_description_set_has_character(Pd_2,C_2))) $\wedge$
(primitive_description_set_has_character(D,C_1) $\vee$
($\exists$Pe$\exists$Pd (group_has_character_description(D,Pe) $\wedge$
character_description_has_primitive_description_set(Pe,Pd) $\wedge$
primitive_description_set_has_character(Pd,C_1))) }

Defn-8. **has_son(C_1,C_2) =**

{C_1,C_2 | related_characters(C_2,'son' , 'of',C_1) $\vee$ related_characters(C_2,'son' , 'to',C_1) }

Defn-9. **has_father(C_1,C_2) =**

{C_1, C_2 | related_characters(C_2,'father' , 'of',C_1) $\vee$related_characters(C_2,'father' , 'to',C_1) }

Defn-10. **male(C) =**

{ C | $\exists C_1$ has_son(C_1,C) $\vee$ has_father(C_1,C) }

Defn-11. **has_child(C_1,C_2) =**

$$\{C_1, C_2 \mid has\_son(C_1,C_2) \vee has\_father(C_2,C_1) \}$$

Obviously, many such relationship terms can be defined, e.g. *daughter of, mother of*, an additional definition *of parent of*, *uncle of,* etc. Also possible familial relationships can be defined using may_be_related_characters.

Definitions for answering CQ-2 and CQ-3 are straightforward, so are not presented. The predicate play_has_character has been defined, so CQ-4 can be answered.

In the next section, these axioms are applied to answer competency questions.

```
<?xml version="1.0"?>
<PLAY>
<TITLE>The Tragedy of Romeo and Juliet</TITLE>
<fm>
<p>Text placed in the public domain by Moby Lexical Tools, 1992.</p>
<p>SGML markup by Jon Bosak, 1992-1994.</p>
<p>XML version by Jon Bosak, 1996-1997.</p>
<p>This work may be freely copied and distributed worldwide.</p>
</fm>
<PERSONAE>
<TITLE>Dramatis Personae</TITLE>
<PERSONA>ESCALUS, prince of Verona. </PERSONA>
<PERSONA>PARIS, a young nobleman, kinsman to the prince.</PERSONA>
<PGROUP>
<PERSONA>MONTAGUE</PERSONA>
<PERSONA>CAPULET</PERSONA>
<GRPDESCR>heads of two houses at variance with each other.</GRPDESCR>
</PGROUP>
<PERSONA>An old man, cousin to Capulet. </PERSONA>
<PERSONA>ROMEO, son to Montague.</PERSONA>
<PERSONA>MERCUTIO, kinsman to the prince, and friend to Romeo.</PERSONA>
<PERSONA>BENVOLIO, nephew to Montague, and friend to Romeo.</PERSONA>
<PERSONA>TYBALT, nephew to Lady Capulet.</PERSONA>
```

Figure 6: Excerpt from XML document of 'Romeo and Juliet'

Demonstration of Competency

Following are some primitive terms.

(i) play_has_character_list('The Tragedy of Romeo and Juliet' , 'Dramatis Personae').
(ii) character_list_has_character_description('Dramatis Personae' , 'ROMEO, son to Montague.').
(iii) character_description_has_primitive_description_set('ROMEO, son to Montague.' , 'ROMEO, son to Montague.').
(iv) primitive_description_set_has_character('ROMEO, son to Montague.' , 'ROMEO').
(v) description_has_relationship('ROMEO, son to Montague.' , 'son' , 'to' , 'Montague').
(vi) character_list_has_group('Dramatis Personae','heads of two houses at variance with each other.').
(vii) group_has_character_description('heads of two houses at variance with each other.' , 'MONTAGUE')
(viii) character_description_has_primitive_description_set('MONTAGUE' , 'MONTAGUE').
(ix) primitive_description_set_has_character('MONTAGUE' , 'MONTAGUE').

Relevant Primitive Term Instances

With that, the following competency question can be answered

CQ-1. Which character is the son of the Montague character?

∃C has_son('Montague',C). —> returns —> has_son('Montague' , 'Romeo').

- applying Defn-2 to (i) & (ii), infer
  (x) play_has_character_description('The Tragedy of Romeo and Juliet' , 'ROMEO, son to Montague.')
- applying Defn-2 to (i), (vi) & (vii), infer
  (xi) play_has_character_description('The Tragedy of Romeo and Juliet' , 'MONTAGUE')
- applying Defn-3 to (x), (iii) & (iv), infer
  (xii) play_has_character('The Tragedy of Romeo and Juliet' , 'ROMEO')
- applying Defn-3 to (xi), (viii) & (ix), infer
  (xiii) play_has_character('The Tragedy of Romeo and Juliet' , ''MONTAGUE')
- applying Defn-5 to (iii), (iv), (xii) & (xiii), infer
  (xiv) related_characters('ROMEO' , 'son' , 'to' , 'Montague')
- applying Defn-8 to (xiv), infer
  (xv) has_son(,'Montague' , 'Romeo').

Answering CQ-1