Reducing Complexity of Diagnostic Message Pattern Specification and Recognition with Semantic Techniques

DPS Dissertation

by

Gilbert Alipui, B.S., M.S.

Submitted in partial fulfillment
of the requirements for the degree of
Doctor of Professional Studies
in Computing

at

School of Computer Science and Information Systems

Pace University

May 2016

We hereby certify that this dissertation, submitted by **Gilbert Alipui** satisfies the dissertation requirements for the degree of Doctor of Professional Studies in Computing and has been approved.

Lixin Tao

Dr. Lixin Tao
Chairperson of Dissertation Committee

5/27/16
Date

[illegible]

Dr. Ronald Frank
Dissertation Committee Member

5/27/16
Date

Meikang Qiu

Dr. Meikang Qiu
Dissertation Committee Member

5/27/2016
Date

Seidenberg School of Computer Science and Information Systems

Pace University

Abstract

Reducing Complexity of Diagnostic Message Pattern Specification and Recognition with Semantic Techniques

by

Gilbert Alipui, B.S., M.S.

Submitted in partial fulfillment

of the requirements for the degree of

Doctor of Professional Studies

in Computing

May 2016

Different companies in the same line of business can have similar computer systems with built-in diagnostic routines, and the ability to regularly send error-driven or event-driven environmental diagnostic messages in XML back to the system manufacturer. The system manufacturer typically uses these to determine faults in the system. The outcome of this troubleshooting can also assist end-users and clients in solving problems, and provide the production team valuable information that can be used to improve future versions of the product. A company merger could lead to the same team processing diagnostic messages from similar but different products, in different syntax, leading to the complexity of specifying and maintaining diagnostic message pattern specification and recognition for many different syntaxes.

This research reduces the above complexity by extending ISO Schematron, the industry standard language for XML semantic constraints specification and validation, with conceptual rules. Pace University Knowledge Graphs are used to describe the concepts or classes relevant to the diagnostic messages of a system, and the new conceptual Schematron rules are introduced to specify diagnostic patterns on these concepts. Such conceptual diagnostic patterns are then converted automatically into concrete Schematron rules based on the syntax of the specific diagnostic messages. A complete prototype was designed and implemented to validate this new methodology.

Acknowledgements

Tackling the challenge of researching my topic, and writing my DPS dissertation, trying to keep up with a demanding job, and completing all the requirements for the DPS program have been one of the most glorious, difficult, and exhausting challenges I have ever undertaken in my life. I liken the experience to a very intense roller coaster ride on steroids, for the very exciting, dizzying heights, thinking that I had finally found a dissertation topic, to very depressing lows, upon realizing that my topic fell short of Dr. Grossman's "So What?" question.

I would like to thank My parents for their endless love and prayers, and for inspiring me to strive for more knowledge, and my entire family for praying for me, and I thank God for hearing those prayers, guiding, and helping me to stay on track to complete this very challenging program.

Special Thanks to my wonderful twin teenage sons Miles and Milan for their patience, companionship, encouragement, and putting up with Daddy being always buried in piles of papers and books, and not being able to take them to Breezy water slides park as often as they would have liked.

I would also like to thank my great brother Dr. Nicholas Alipui, his wife Sista Mabel and their great children, my nephews Andrew and Mathew for welcoming me into their home, supporting and encouraging me even in the bleakest of moments.

Thank you to my dearest friend Hirut Kassaye whom I affectionately call "Tweety Bird". She is a great mom, and a great financial advisor. Thank you, Hirut for helping me re-organize my budget to enable me to keep up with my crushing financial obligations, whilst at the same time improving my credit rating. Thank you also for your constant intellectual support, willingness to engage in meandering exploratory discussions, your unwavering encouragement, and for patiently listening to me moan ad-nauseam about not being able to find a dissertation topic. Thank you for your love, and for helping me extricate myself from the prison of bitterness and disappointment.

Thank you to my advisor Dr. Tao whose invaluable expert knowledge and guidance I have benefited from greatly.

Thanks also to Dr. Frank for expertly guiding me to identify my research topic, and to Dr. Qiu for all the great advice, and for being on my dissertation committee.

Thanks to Christian Martinez, and Steven Golikov for giving freely of their time to explain their dissertations and conduct code reviews with me.

Last but not least, I would like to acknowledge my great friends and colleagues from the 2016 cohort. I shall long cherish the magic and great memories you have given me in the DPS program.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

XML is used in many applications to capture and store diagnostic information. Built-in diagnostic routines in computer systems regularly send error-driven or event-driven diagnostic messages encoded in XML back to the system manufacturer for fault conditions analysis to pinpoint what is wrong. The findings can also be used to help end-users and clients to solve problems, and provide the production team information with which to improve future versions of the product.

The XML file transfers are typically done through dial-up modem transport, Secure File Transfer Protocol (SFTP) or over the Internet using secure HTTP transport. Over time, a greater volume of diagnostic message files are sent home in a flood of files being transmitted to report systems' health status. Some of this incoming traffic contains spurious, or trivial errors. Others contain critical errors that if ignored, could cause catastrophic system failure and damage. If this growing volume of error reports is not controlled, these files could overwhelm front-end servers, which process the incoming XML files.

Another very important consideration is that the files largely contain similar information about the health status of systems and their components. Crucially, these data are stored in a myriad different ways. This is because "XML standards and technologies do not provide an adequate layer for coping with dialects". [95] Worse yet, in the absence of a corporate XML standard, there is a proliferation of vast amounts of similar diagnostic data being transmitted in a nightmarish potpourri of different dialects and formats. [95]

This presents a serious challenge to define criteria for efficiently and accurately recognizing the different system problems amidst the different dialects contained in the XML documents.

1.2 Hardware and Software Diagnostic Challenges

1.2.1 Motivating Example

The crucial problem is how to manage the complexity of maintaining business rules on similar but inexact XML dialects.

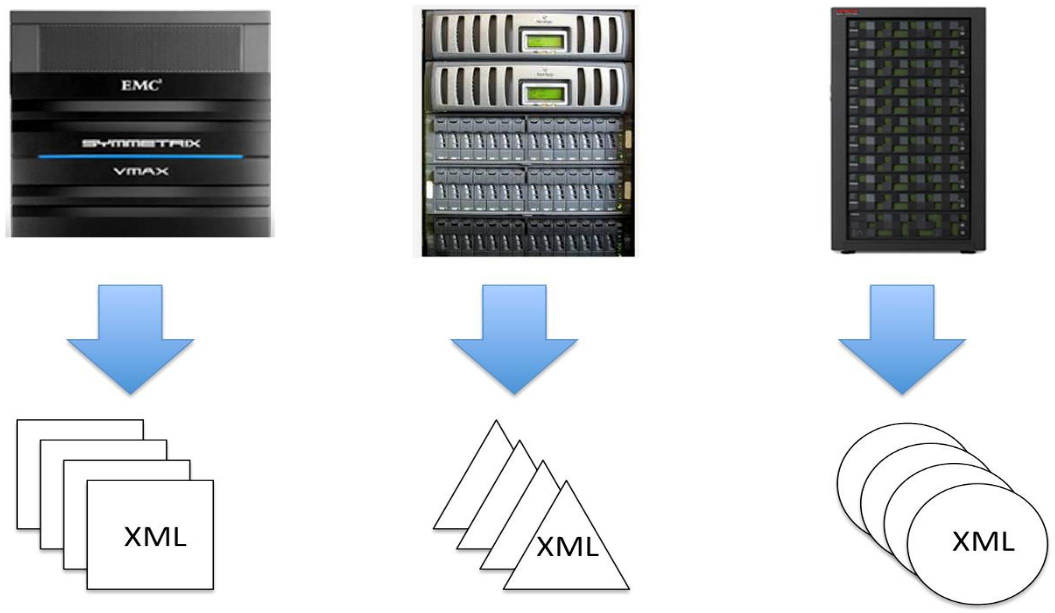
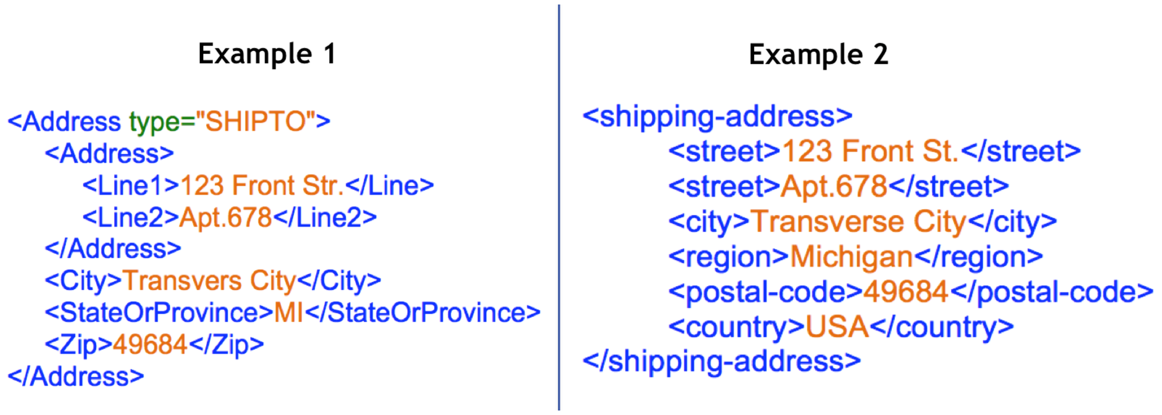


Figure 1 Computer Systems Diagnostic Files with Different XML Dialects

The following is a motivating example of a mailing address to illustrate the severity of the problem of encoding similar data in different dialects. A mailing address is a good starting point for illustrating the severity of the Semantic Rule Complexity and Heterogeneity (SERCH) problem.

1.2.2 Mailing Address Use-Case



Example 1

```
<Address type="SHIPTO">
    <Address>
        <Line1>123 Front Str.</Line>
        <Line2>Apt.678</Line2>
    </Address>
    <City>Transvers City</City>
    <StateOrProvince>MI</StateOrProvince>
    <Zip>49684</Zip>
</Address>
```

Example 2

```
<shipping-address>
        <street>123 Front St.</street>
        <street>Apt.678</street>
        <city>Transverse City</city>
        <region>Michigan</region>
        <postal-code>49684</postal-code>
        <country>USA</country>
</shipping-address>
```

Figure 2 Example of Similar Information Encoded in Different Dialects

The examples shown in Figure 2 above are both syntactically valid and well-formed XML representing the same information but because the vocabulary is different, they are not the same.

- "The type of shipping address is reflected in the data (as an attribute value SHIPTO), in Example 1, but is indicated in the tag (shipping address) in Example 2.

- The street address and apartment number are encoded in nested tags in Example 1, but not in Example 2. The tag names are also different.

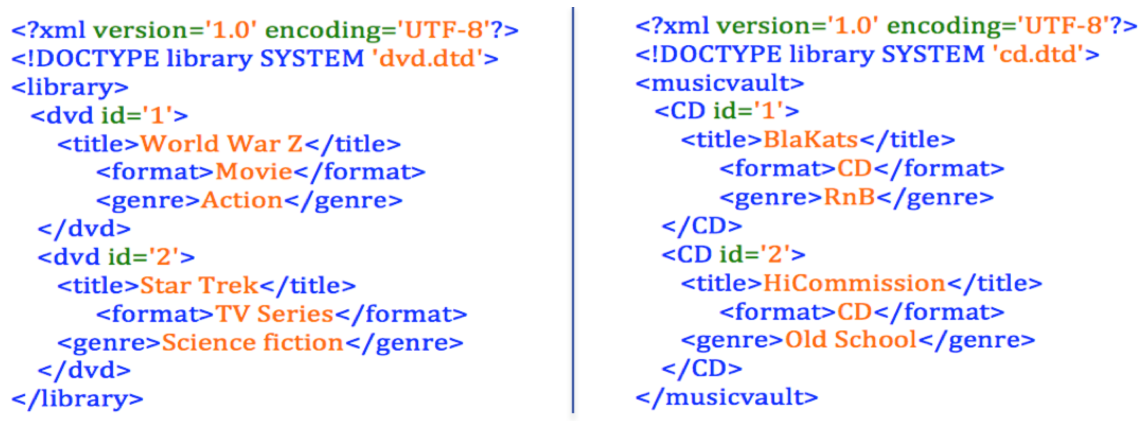
- Zip is used in Example 1 whereas postal-code is used in Example 2. Also, the capitalization is different.

- Special characters are used in Example 2, and the structure is different.

- Lastly, the format of the data is different. Example 1 uses two-letter abbreviations, and Example 2 contains the name of the state". [95]

What this means is that validation rules that worked for the first instance document (Address) will fail for the second document simply because the Address tag has been changed to street.

1.2.3 DVD and CD-ROM Use-Case



```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE library SYSTEM 'dvd.dtd'>
<library>
  <dvd id='1'>
    <title>World War Z</title>
        <format>Movie</format>
        <genre>Action</genre>
   </dvd>
   <dvd id='2'>
    <title>Star Trek</title>
        <format>TV Series</format>
    <genre>Science fiction</genre>
   </dvd>
</library>
```

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE library SYSTEM 'cd.dtd'>
<musicvault>
  <CD id='1'>
    <title>BlaKats</title>
        <format>CD</format>
        <genre>RnB</genre>
   </CD>
   <CD id='2'>
    <title>HiCommission</title>
        <format>CD</format>
    <genre>Old School</genre>
   </CD>
</musicvault>
```

Figure 3 Another Depiction of the XML Dialect Proliferation and Rule Complexity Problem

1.3 **Complexity Caused by Multiple Dialects**

1.3.1 Inconsistent Tag Names

As shown in the Figure 3, the challenge presented is that the tag names are not constant. Worse yet, for similar information, they may differ, and may continue to evolve with different versions or with other products. The severity of this fact will be explained in more detail in Chapter 3.

1.3.2 Semantic Validation for Data Integration

The previous section describes important validation considerations for sound data integration. Schematron schemas are the most expressive for semantic validation and are typically used for semantic validation of XML data.[1] The following use-cases show how Schematron can be used to specify patterns for constraints validation.

1.3.3 E-Mail Address Use-Case

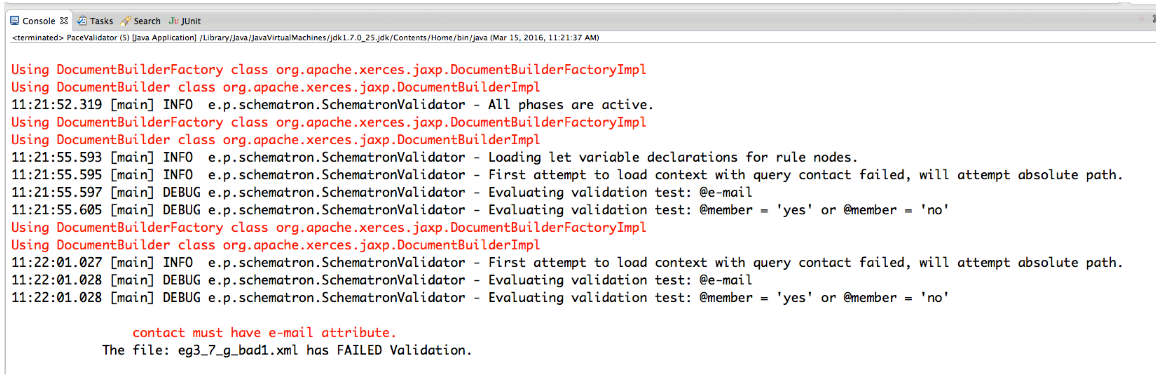
The listing in figure 4 below depicts an XML document missing the required contact e-mail address. Figure 5 on page 6 shows the listing for a Schematron document used to make sure that an e-mail address for the contact is present in the XML file. Figure 6 on page 6 depicts the Schematron validation session correctly enforcing the rule that contact e-mail must have an e-mail address.

```xml
<doc>
  <prologue>
    <title>Technical Document</title>
      <subtitle>System Health Status</subtitle>
        <contact member='yes'>John Doe.com</contact>
  </prologue>
  <section/>
</doc>
```

Figure 4 XML Listing with Missing E-mail Attribute

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns=http://purl.oclc.org/dsdl/schematron
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:schemaLocation="http://purl.oclc.org/dsdl/schematron ../schematron-iso.xsd">
  <title>Technical document schema</title>
  <pattern id="Contact_attributes">
    <rule context="contact">
      <assert test="@e-mail">
          contact must have e-mail attribute.
       </assert>
       <assert test="@member = 'yes' or @member = 'no'">
          contact must have member attribute with 'yes' or 'no' value.
        </assert>
      </rule>
   </pattern>
</schema>
```

Figure 5 Listing of XML and Schematron Rule Where E-mail is Required



```
<terminated> PaceValidator (5) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.7.0_25.jdk/Contents/Home/bin/java (Mar 15, 2016, 11:21:37 AM)

Using DocumentBuilderFactory class org.apache.xerces.jaxp.DocumentBuilderFactoryImpl
Using DocumentBuilder class org.apache.xerces.jaxp.DocumentBuilderImpl
11:21:52.319 [main] INFO  e.p.schematron.SchematronValidator - All phases are active.
Using DocumentBuilderFactory class org.apache.xerces.jaxp.DocumentBuilderFactoryImpl
Using DocumentBuilder class org.apache.xerces.jaxp.DocumentBuilderImpl
11:21:55.593 [main] INFO  e.p.schematron.SchematronValidator - Loading let variable declarations for rule nodes.
11:21:55.595 [main] INFO  e.p.schematron.SchematronValidator - First attempt to load context with query contact failed, will attempt absolute path.
11:21:55.597 [main] DEBUG e.p.schematron.SchematronValidator - Evaluating validation test: @e-mail
11:21:55.605 [main] DEBUG e.p.schematron.SchematronValidator - Evaluating validation test: @member = 'yes' or @member = 'no'
Using DocumentBuilderFactory class org.apache.xerces.jaxp.DocumentBuilderFactoryImpl
Using DocumentBuilder class org.apache.xerces.jaxp.DocumentBuilderImpl
11:22:01.027 [main] INFO  e.p.schematron.SchematronValidator - First attempt to load context with query contact failed, will attempt absolute path.
11:22:01.028 [main] DEBUG e.p.schematron.SchematronValidator - Evaluating validation test: @e-mail
11:22:01.028 [main] DEBUG e.p.schematron.SchematronValidator - Evaluating validation test: @member = 'yes' or @member = 'no'

            contact must have e-mail attribute.
        The file: eg3_7_g_bad1.xml has FAILED Validation.
```

Figure 6 Schematron Validation Failure on Missing E-mail Attribute

1.3.4 Reducing Complexity

The complexity of maintaining multiple Schematron files for the different dialects is resource-intensive, error-prone, and inefficient. This research seeks an adaptive means of recognizing and classifying the plurality of dialects to reduce the complexity of maintaining custom rules.

1.4 **Current Approaches to the Problem**

The following use-cases describe current approaches to the problem.

1.4.1 Computer Code Use-Case

Computer programmers can use code and code-like rules to identify different patterns and target actionable errors or conditions in XML diagnostic message files. These programmed instructions typically take the form of if-else condition testing, and other program logic and can take the form of data scrapers, and or parsers. The advantage of using code is that code is accessible, often readily available, and there are many competent programmers. This approach however, is brittle. Hard-coding if-else-logic is not desirable as it is not agile, it is very hard to adjust to the different syntaxes, and it does not scale easily.

1.4.2 Schema Use-Case

Schema Languages are often employed to enforce certain rules to ensure XML instance documents conform to a certain format. The most common Schema languages in widespread use are Document Type Definitions (DTDs), W3C XSD (XML Schema Definition), Relax-NG, and Schematron.

1.4.3 Schematron Use-Case

As explained in section 1.3.2, Schematron, a powerful and flexible rule-based XML schema language is another alternative for constraint validation[1]. Schematron is more flexible, and more powerful than W3C XML schema, Relax-NG, and DTDs. This alternative however, is still limited in its ability to handle the problem of the complexity of many different dialects, and versions, as it is still necessary to specify multiple schema files for the respective XML diagnostic message files.

1.4.4 A Better Approach

An ontology is a rich, formal logic-based model for using semantics to describe knowledge about objects in a domain. Using a pre-defined, reserved vocabulary of terms to define concepts and relationships between them, an ontology can refer to either a vocabulary, taxonomy or something more. "The OWL Web Ontology Language provides an expressive language for defining ontologies that capture the semantics of domain knowledge".[3] To overcome Schematron's limitations, instead of restricting the validation scope to syntactic and semantic validations, we propose leveraging additional technologies to simplify the handling of multiple copies of pattern specifications by adopting the inheritance relations for knowledge representation that is supported by the Web Ontology Language (OWL), commonly referred to as the is_a relation.

1.4.5 An Even Better Approach

As explained in the previous section, an ontology is capable of referring either to a vocabulary, taxonomy or something more, as in the case of this research. It is possible to go further in overcoming additional challenges in knowledge representation. OWL primarily uses inheritance, the aforementioned is_a relation for knowledge representation. Custom relations with their increased expressiveness are required for knowledge representation in the diagnostic pattern specification domain. This research goes further, and uses the composition relation (*part_of*, or *partOf*) and other custom relations in novel ways to define custom relations.

1.5 **Problem Statement**

Computing devices have built-in diagnostic routines that automatically send error-driven or event-driven diagnostic messages encoded in XML back to the system manufacturers. Each company may use different syntaxes for their diagnostic information. In the event of a merger or acquisition, there is often a need to handle diagnostic pattern classification for multiple dialects. This research aims at reducing the complexity of maintaining patterns of XML system diagnostic information based on different syntaxes.

This research expects to make the following contributions:

- Provision of a knowledge graph to describe the diagnostic patterns in a more declarative way, allowing domain experts to easily understand and use the framework without needing extensive XML knowledge.

- Reduce complexity by introducing an ontology, and going further by using a knowledge graph to support more economical specifications of diagnostic classification patterns for XML semantic constraints validation.

- Reduction of the complexity of maintaining multiple versions of such semantic patterns by unique use of knowledge graphs.

1.6 **Solution Methodology**

Domain experts can use any text editor to represent domain knowledge in OWL files. For convenience, the Protégé IDE can also be used to create the OWL files, and to visually to represent domain knowledge. Coupling Schematron with a knowledge graph to use semantic constraints to define diagnostic classification patterns and reducing the complexity of maintaining multiple versions of such semantic patterns are major contributions of this research, providing an adaptive framework that brings more scalable and reusable semantic error explication to the semantic constraints validation process.

Different systems sending messages similar in semantics but not in the same dialects is a significant and important data integration problem. To avoid the duplicated effort of maintaining many versions of the Schematron rules, this research proposes a framework to:

(a) Apply Schematron to define exception patterns and automatically launch reactions to actionable error conditions.

(b) Introduce a knowledge graph to define extra concepts and their relationships and extend Schematron so that exception patterns are defined at a conceptual level so that only one copy of the pattern needs to be maintained, and the pattern can be used to automatically validate diagnostic information in different syntaxes, or variations of the constraints or patterns, resulting in significantly reduced complexity of maintain many versions of similar constraints or patterns for different dialects.

(c) Introduce the implementation of a prototype to validate the research idea.

1.7 **Roadmap**

The rest of this dissertation is structured as follows:

Chapter 2 entitled 'Current Status of the XML Semantic Rule Complexity Problem' provides background information on the XML data integration problem, reviews achievements so far, key methodologies and tools relevant to the problem. A motivating example is introduced to explain the current status of the Heterogeneous XML Rule Complexity Problem and describe the major schema languages and related technologies. Following that is an outline of Schematron's XSLT-based design goals and intended uses. This chapter demonstrates semantic constraints with examples.

Chapter 3 entitled 'Solution Methodology' expands on the problem, and covers the research in more detail and how it overcomes the limitations of XML data integration and XSLT-based Schematron Validation.

Chapter 4 entitled 'Additional Solution Methodology Details' provides highlights in the implementation of the research solution methodology.

Chapter 5 entitled 'Interpretation and Evaluation of Solution Methodology ' describes the experimental validation of the research solution.

Chapter 6 concludes this dissertation, outlines the advantages of the knowledge-driven Data Integration Solution and lays out future work that may extend this research.

Chapter 2

Current Status of the XML Semantic Rule Complexity Problem

2.1 Current Methods

The primary thrust of this research is to propose a cost-effective data integration framework that targets the semantic rule complexity and heterogeneity problem, enabling the processing of the explosion of content in the form of inbound XML diagnostic files traffic from field-resident systems running the gamut from enterprise servers, storage arrays, software agents, other internet-connected devices and IoT devices. In addition, this research also seeks to promote convenient post-validation failure explication of symptom codes, and reduce the complexity of maintaining many versions of similar constraints or patterns for different dialects by providing a framework for pattern identification and classification that facilitates data integration and improves the identification and explication of actionable errors. The complexity that this research targets is known by different terms. "A good example of semantic heterogeneity is the use of synonyms, where different terms are used to refer to the same concept. There are many more types of semantic heterogeneity and they have been classified in [Visser et al.,1998]."[120] Variously called the "Integration Problem" or "Interoperability Problem", "XML Dialect Proliferation Problem", "Semantic Heterogeneity Problem" or the "Semantic

Rule Complexity Problem". This research uses **Semantic Rule Complexity and Heterogeneity Problem**. (Henceforth interchangeably "Semantic Rule Complexity and Heterogeneity Problem" (**SERCH**) or the "**XML Semantic Dialect Proliferation Problem**").

Prior recent research in this field employ many approaches to achieve efficient interoperability between heterogeneous information systems. They can be classified into three main ontology-based approaches: single, multiple and hybrid.[22]

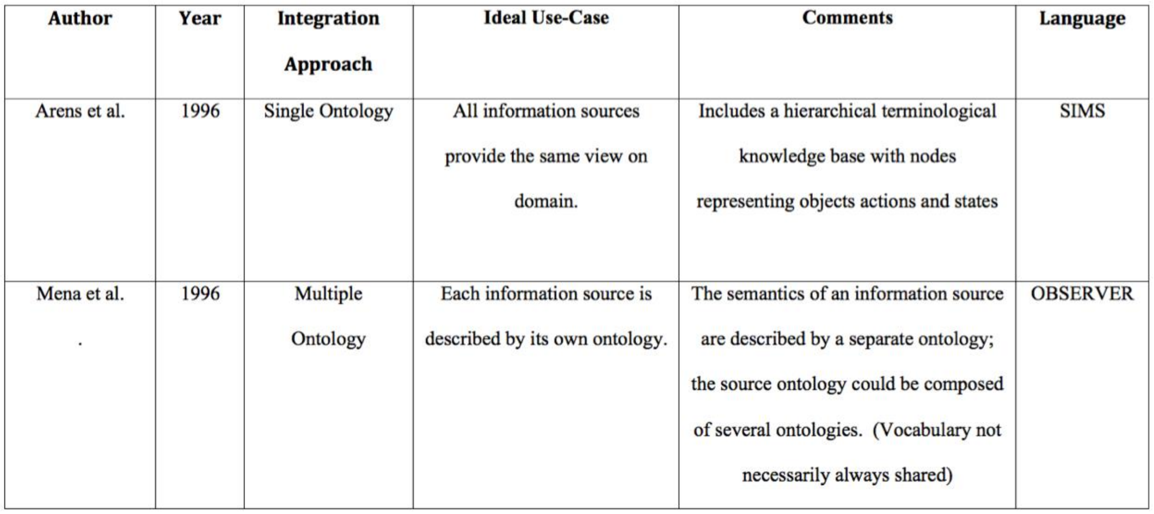
1. Single ontology approaches use one global ontology that links all information sources by relations expressed via mappings that identify the correspondence between each information source and the ontology.
2. Multiple ontologies approaches describe each information source by its own ontology and inter-ontology mappings are used to express the relationships between the ontologies.
3. The hybrid approaches combine the two previous approaches whereby each information source has its own ontology and the semantic of the domain of interest as a whole is described by a global reference ontology.

In these approaches there are two types of mappings:

1. Mappings between an information source and its local ontology.
2. Mappings between local ontologies and the global ontology.[95]

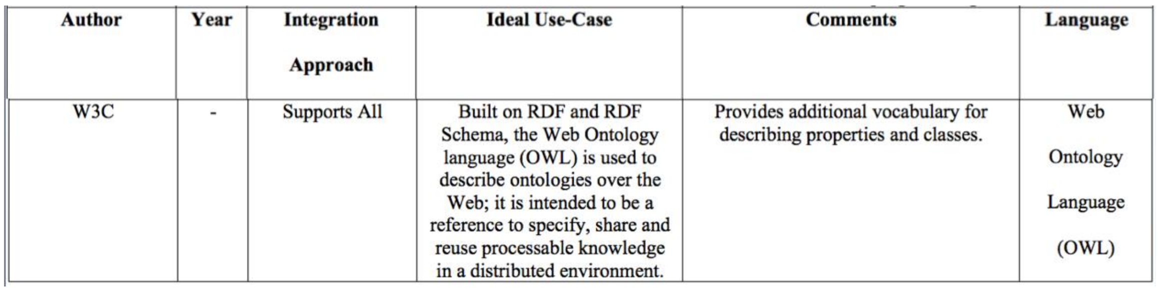
Another approach to overcome the issues posed by data heterogeneity to achieving data interoperability also draws on ontologies. Based on architecture, a central data integration system or a peer-to-peer data integration system is employed, where the central data integration system typically has a global schema, and the peer-to-peer data integration system does not.[94] Table 1 in next section (pages 14 - 16) summarizes ontology-based data integration approaches drawn from the existing literature.

Table 1 Summary of Ontology-Based Data Integration Approaches - 1



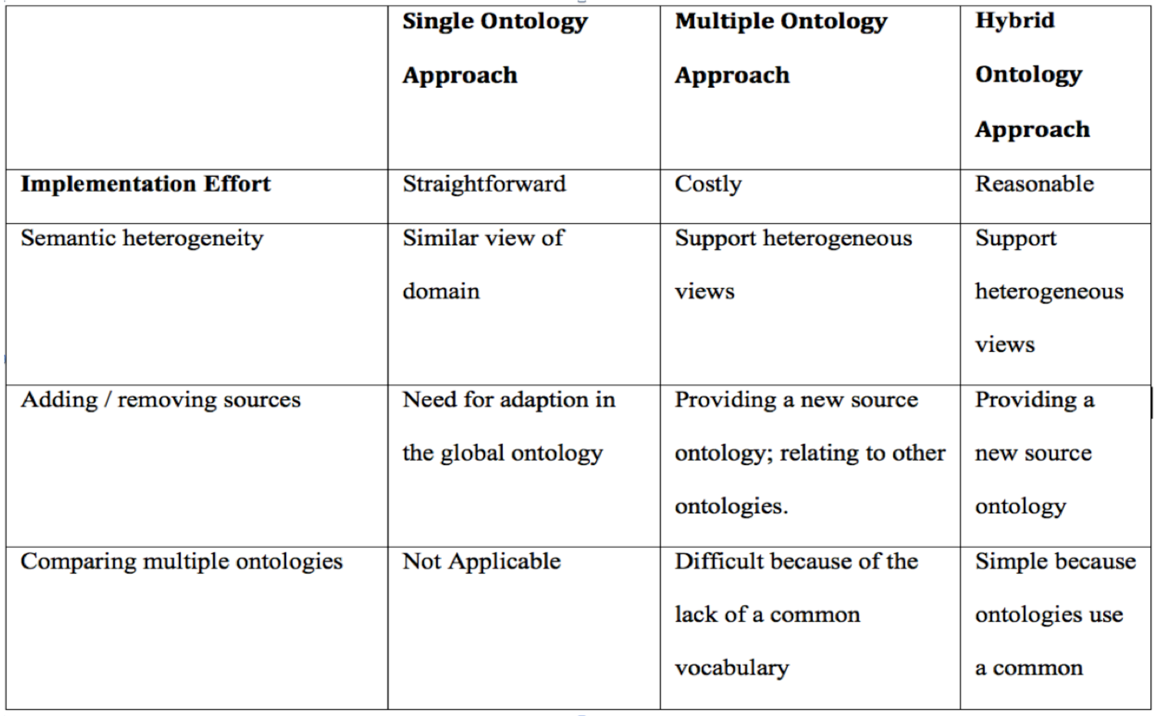
Author	**Year**	**Integration Approach**	**Ideal Use-Case**	**Comments**	**Language**
Arens et al.	1996	Single Ontology	All information sources provide the same view on domain.	Includes a hierarchical terminological knowledge base with nodes representing objects actions and states	SIMS
Mena et al. .	1996	Multiple Ontology	Each information source is described by its own ontology.	The semantics of an information source are described by a separate ontology; the source ontology could be composed of several ontologies. (Vocabulary not necessarily always shared)	OBSERVER

Author	Year	Integration Approach	Ideal Use-Case	Comments	Language
Goh et al.	1997	Hybrid Ontology	Similar to multiple ontology, each source semantic is described by its own ontology, and built on one global share ontology to make the source ontology comparable.	Overcomes the drawbacks of single and multiple ontologies. The shared ontology contains basic terms (primitives) of the domain. Shared vocabulary makes the source ontologies comparable and overcomes the drawbacks of multiple ontologies. Ontology re-use however is difficult.	COIN
Cycorp	-	Supports All	A declarative language based on first-order logic (FOL)	Used to express common sense knowledge and to represent the knowledge stored in the Cyc Knowledge Base.	Cycl
Calvanese et al	2001	Multiple Ontology	Each information source is described by its own ontology.	(Vocabulary not necessarily always shared)	DWQ
Karlsruhe Univ Germany	1995	Supports All	A formalism to represent knowledge.	F-logic stands in the same relationship to object-oriented programming as classical predicate calculus stands to relational database programming.	F-Logic
Author	Year	Integration Approach	Ideal Use-Case	Comments	Language
U. So California	1986	Supports All	Under DARPA sponsorship. Loom is a language and environment for constructing intelligent applications	Knowledge representation system that is used to provide deductive support for the declarative portion of the Loom language.	Loom
Michael Genesereth	-	Supports All	Designed for use in the interchange of knowledge among disparate computer systems (created by different programmers, at different times, in different languages, and so forth)	Created to serve as a syntax for first-order logic that is easy for computers to process.	KIF Knowledge Interchange Format
W3C	-	Supports All	Graphical language used for representing information about resources on the web thus constituting a basic ontology language.	Resources are described in terms of properties and property values usingRDF statements. Statements are representated as triples consisting of subject, predicate, object (S, P, O). RDF is written in XML and uses Unique Resource Identifiers URIs to to identify resources.	RDF (S) Resource Description Framework (Schema)
W3C	-	Supports All	Latest standard in ontology languages from the Worldwide Web Consortioum (W3C). Based on its predecessor DAML+OIL, extends RDF (S).	Employing classes and relations as building blocks, OWL has a rich set of modeling constructors. OWL has are three sublanguages with increasing levels of expressivity, namely: OWL-Lite, OWL-DL, OWL-Full.	OWL



Author	Year	Integration Approach	Ideal Use-Case	Comments	Language
W3C	-	Supports All	Built on RDF and RDF Schema, the Web Ontology language (OWL) is used to describe ontologies over the Web; it is intended to be a reference to specify, share and reuse processable knowledge in a distributed environment.	Provides additional vocabulary for describing properties and classes.	Web Ontology Language (OWL)

Table 2 Comparison of Ontology-Based Data Integration Approaches



	Single Ontology Approach	**Multiple Ontology Approach**	**Hybrid Ontology Approach**
Implementation Effort	Straightforward	Costly	Reasonable
Semantic heterogeneity	Similar view of domain	Support heterogeneous views	Support heterogeneous views
Adding / removing sources	Need for adaption in the global ontology	Providing a new source ontology; relating to other ontologies.	Providing a new source ontology
Comparing multiple ontologies	Not Applicable	Difficult because of the lack of a common vocabulary	Simple because ontologies use a common

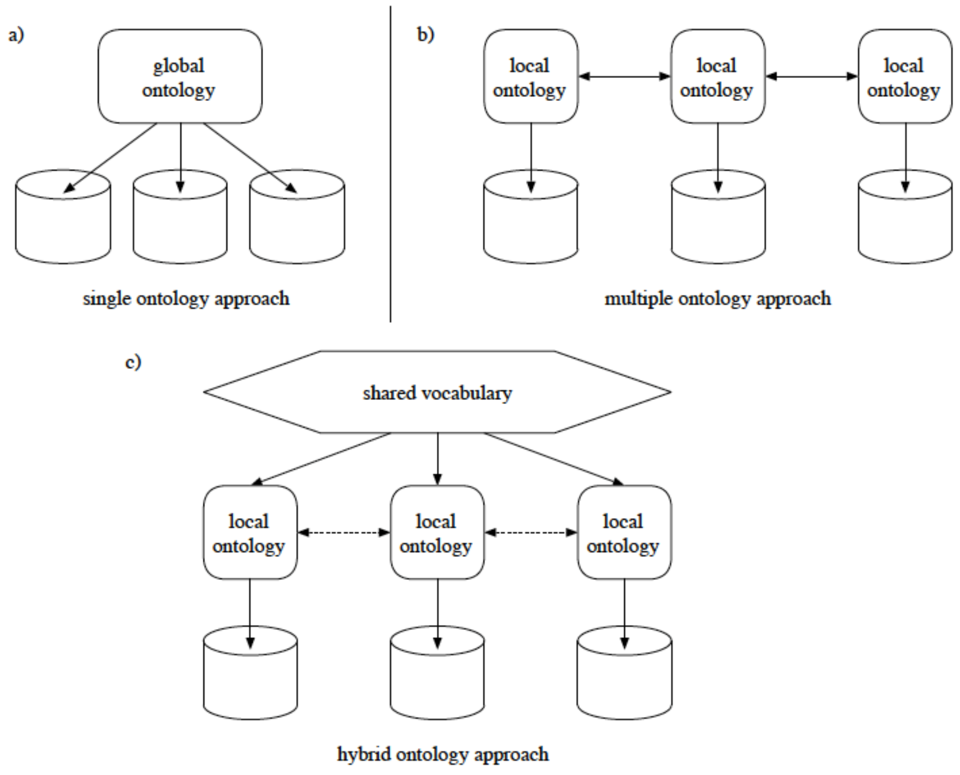


Figure 7 The Three Possible Ways for Using Ontologies for Content Explication

There are many interesting initiatives such as Health Level 7 (HL7), the Clinical Data Architecture (CDA), and Learning Source Description (LSD) in the healthcare arena, but these are primarily based around the use of thesauri in automatic document transformation. Though the thesauri play a similar role to that of the mapping file in this research, fundamentally, automatic document transformation is not the objective of this research [23], as transformations can come with their own set of complexity and complications.

2.2 **World Wide Web Background in Brief**

World Wide Web Consortium W3C

“The World Wide Web Consortium (W3C) is the main international standards organization for the World Wide Web (abbreviated WWW or W3).”[110] "where member organizations, a full-time staff, and the public work together to develop Web standards which are documents published by W3C that define Web technologies through a process designed to promote consensus, fairness, public accountability, and quality. At the end of this process, W3C then publishes Recommendations, which are considered Web standards.” Some W3C standards include CSS, SVG, WOFF, the Semantic Web stack, XML, and a variety of APIs.[112] XML is the de-facto standard for data interchange.[114] It is ubiquitous in its use as an information exchange platform in many business domains.[81][82] XML provides a surface syntax for structured documents, but imposes no semantic constraints on the meaning of these documents.[76] “An XML Schema is a language for expressing constraints about XML documents.” XML Schemas also extend XML with datatypes. [76] The main schema languages in widespread use are Document Type Definitions (DTDs), Relax-NG, Schematron and W3C XSD (XML Schema Definitions).[112]

Founded with the support of the U.S. federal government, and subsequent to 1993, operated as a standards development function under the auspices of the Internet Society, the Internet Engineering Task Force (IETF) is an open standards organization that develops and promotes voluntary Internet standards, in particular the standards that comprise the Internet protocol suite (TCP/IP).[118]. Its stated mission “is to make the Internet work better by producing high quality, relevant technical documents that influence the way people design, use, and manage the Internet.”[119]

2.3 XPointer

XPointer is "a syntax for URI fragment identifiers that selects particular parts of the XML document referred to by the URI — often used in conjunction with an XLink."[119] URIs play a critical role in the Semantic Web. They make resources uniquely identifiable; provide the basis for the graph-like RDF data model, and enable distributed metadata creation.[4]

URI: A Uniform Resource Identifier defines a unique name for statements across the entire Internet. Each component of a statement; the subject, predicate, and object contains a URI affirming its identity throughout the entire WWW, eliminating naming conflicts, and also providing a path to additional information. A URI may include a Uniform Resource Locator (URL), which may be dereferenced for useful additional information, or an abstract Uniform Resource Name (URN). A URI can also extend to Internationalized Resource Identifiers (IRIs).[4]

IRI: The internationalized resource identifier (IRI) was defined by the Internet Engineering Task Force (IETF) in 2005 as an extension to the uniform resource identifier (URI) with support for the Universal Character Set "whereas URIs are limited to a subset of the ASCII character set. IRIs may contain characters from the Universal Character Set (Unicode/ISO 10646), including Chinese or Japanese kanji, Korean, Cyrillic characters, and so forth."[2]

2.4 XPath

ISO/IEC 19757 uses XPath to identify information items in Schematron schemas.[13] XPath is "a non-XML syntax used by both XPointer and XSLT for identifying particular pieces of XML documents. For example, XPath can locate the third address element in the document or all elements with an email attribute whose value is elharo@metalab.unc.edu."[5]

2.5 XQuery

XQuery is a W3C Recommendation, supported by all major databases, "XQuery is the language for querying XML data. XQuery for XML is like SQL for databases. XQuery is built on XPath expressions." [103]

2.6 XPattern

XPattern, a subset of Xpath is the way to identify templates in XSLT.

2.7 Document Object Model (DOM) and SAX

The Document Object Model, a language-neutral, tree-oriented API that treats an XML document as a set of nested objects with various properties.[6] It is the only XML Document parsing model that is officially recommended for XML document parsing by the W3C. The W3C DOM can be used to create XML documents, navigate DOM structures, and add, modify, or delete DOM nodes. DOM creates a representation of the entire document as nodes, regardless of how large the document is. DOM parsing can be slower than SAX parsing, but can be handy for retrieving all the data from a document, or retrieving a piece of data several times.

The DOM stays resident in memory as long as the code that created the DOM representation is running. [104]

"The SAX API is event-based. XML parsers that implement the SAX API generate events that correspond to different features found in the parsed XML document. By responding to this stream of SAX events in Java code, you can write programs driven by XML-based data." Though slightly more complicated, SAX parsing is faster than DOM parsing. Also, XML document representations in SAX follows a different type of directory and file structure to that of DOM documents. SAX parsing is more appropriately compared to an index search as opposed to DOM parsers extracting the same information by reformatting the entire book into a DOM format in memory. [104]

2.8 **XML Parsers**

XML parsers are off-the-shelf components that perform some compiler-related tasks like handling all lexical analysis and parsing. Many currently available Java-based XML parsers support two popular parsing standards: the SAX and DOM APIs.

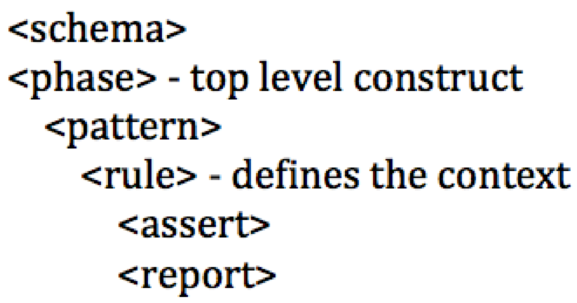
2.9 **Schematron**

Schematron is for specifying patterns. The Schematron assertion language provides a mechanism for making assertions about the validity of an XML document using XPath expressions. There are six commonly used elements in a Schematron document: schema, ns, pattern, rule, assert, and report. The namespace URI for the elements used by the Schematron assertion language is http://purl.oclc.org/dsdl/schematron.

Designed and implemented by Rick Jeliffe at the Academia Sinica Computing Centre, Taiwan, Schematron is an easy-to-use highly customizable rule-based language that validates and reports on XML documents. Schematron is used by the likes of the National EMS Information System (NEMESIS.org), OASIS Universal Business Language (UBL), OASIS Election Markup Language, UK government's e-government interoperability framework, IBM's Business Information Conformance Statements, and the National Institute of Standards (B2B Interoperability test-bed and Security Content Automation Protocol)[1][108]. Even the office of the Director of National Intelligence of the United States (dni.gov) utilizes Schematron for data validation for the XML data encoding specification for intelligence publications.[1][109] Using XPath, which is a node-walking language used to look into and walk through the nodes of an XML document, it can report on well formedness and the validity of XML documents and run multiple stages of a workflow. It reports on the structure or content of the XML document whilst leaving the document structure intact. Languages can be very complicated and awkward in relation to real-world business rules and often, it is difficult for domain experts to map business rules written in English to the technical schema specifications, whereas Schematron expressions are more declarative and closer to English grammar. Schematron is expressive enough to document and assert business rules. Capable of working well with XML schemas, it can assert complex requirements for constraints that are beyond the capability of schemas. Schematron also uses a multi-phase approach to validating XML, which is very handy with larger XML files. This way, groups of assertions and constraints on related components are processed together in separate phases.

2.9.1 Components of a Schematron file.

The Schematron file is relatively simple, and is comprised of six basic elements that make up the following structure[12]:



```
<schema>
<phase> - top level construct
  <pattern>
    <rule> - defines the context
      <assert>
      <report>
```

Figure 8 Basic Structure of a Schematron File

The core elements of a Schematron file are the root element bearing the name schema, which can be represented in various ways provided by XML namespaces, and has a number of rule elements. It is recommended that the schema element be given a descriptive title. Each rule uses either an Assert or Report element to perform validation. Rules are built using pattern elements that contain related rules.[11]

In the above Schematron document structure, the context can be any valid XPattern.

2.9.1.1 Schema

Schema is the root element of the Schematron document. ISO Schematron must use the namespace: http://purl.oclc.org/dsdl/schematron

2.9.2 Phase

Schematron Phases are used to organize a large number of rules into manageable modular collections called phases.

2.9.3 Pattern

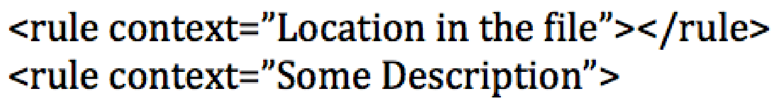
A pattern is a collection of related rules. Rules work independently of each other, and do not need to work on the same elements. The name of the pattern will be displayed in the output and is helpful in identifying which section of the document is failing the validation within the pattern.

2.9.3.1 Assert

Assert is the fundamental element in the Schematron file that is used to test for a given condition. Assert supports XSLT and XPath expression, Boolean logic and complicated formulas.

2.9.3.2 Rule

Rule is another basic building block of a Schematron file.



```
<rule context=”Location in the file”></rule>
<rule context=”Some Description”>
```

Figure 9 The Schematron Rule Element

The context can be any valid XPattern. “All the asserts statements for particular rule context are grouped together”. [11]

Schematron can inform the user about whether or not an element or attribute is present, how many times an elemental attribute is present, about the content of an elemental attribute and about the sequence of the elements in the XML document.

Once the desired location in the document is reached, assertions can be made or questions asked through reports. Assertions allow the user to "declare that some condition must be true, or return a message if it is not.” Report "asks if a condition is true, and returns a message if it is."[1]

Schematron differs from other schema languages in its use of tree patterns to operate on the parsed document. Other schema languages, which are grammar-based, lack the convenience and ease of representing many different kinds of structures and rules. It does not have the limitations of DTDs. Schematron has additional capabilities. It can:

- Detect whitespace in an e-mail address
- Check character lengths of titles
- Detect empty elements[105]
- Check that an ISSN has 8 characters with a hyphen in the middle

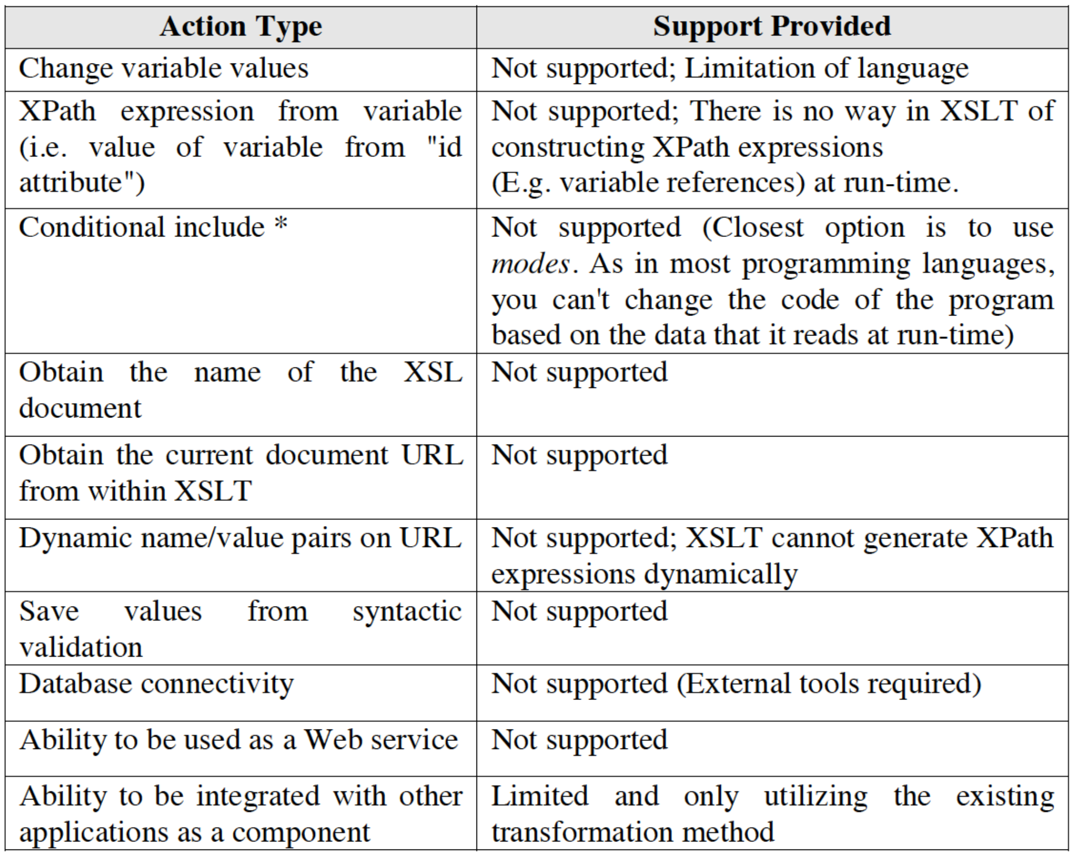
ISSN stands for International Standard Serial Number. It is an eight-digit serial number used to uniquely identify a serial publication. [106]

- Ensure that DOI syntax matches business rules

DOI stands for Digital Object Identifier. It is a serial code used to uniquely identify content of various types on electronic networks.[107]

2.10 **XSLT Validation**

In addition to Xpath, Schematron also uses XSLT in its validation rules. However, since XSLT's main purpose is document transformation, the XSLT-based Schematron implementation has many limitations, as listed in figure 10 below.



Action Type	Support Provided
Change variable values	Not supported; Limitation of language
XPath expression from variable (i.e. value of variable from "id attribute")	Not supported; There is no way in XSLT of constructing XPath expressions (E.g. variable references) at run-time.
Conditional include *	Not supported (Closest option is to use *modes*. As in most programming languages, you can't change the code of the program based on the data that it reads at run-time)
Obtain the name of the XSL document	Not supported
Obtain the current document URL from within XSLT	Not supported
Dynamic name/value pairs on URL	Not supported; XSLT cannot generate XPath expressions dynamically
Save values from syntactic validation	Not supported
Database connectivity	Not supported (External tools required)
Ability to be used as a Web service	Not supported
Ability to be integrated with other applications as a component	Limited and only utilizing the existing transformation method

Figure 10 XSLT-Based Schematron Limitations

2.11 **Semantic Web**

"The Semantic Web is a vision for the future of the Web in which information is given explicit meaning, making it easier for machines to automatically process and integrate information available on the Web. The Semantic Web will build on XML's ability to define customized tagging schemes and the Resource description Framework (RDF)'s flexible approach to representing data. Semantic web information representation is done through a set of assertions called statements comprised of subject predicate and the object, sometimes referred to or also known as triples. These three elements are analogous in meanings to normal English grammar. "The subject of the statement is the thing that statement describes, the predicate describes the relationship between the subject and the object."[7]

```
Andrew knows Matt.
Andrew's surname is Perez-Lopez.
Matt knows John.
Ryan works with John.
```

Figure 11 Information about some people

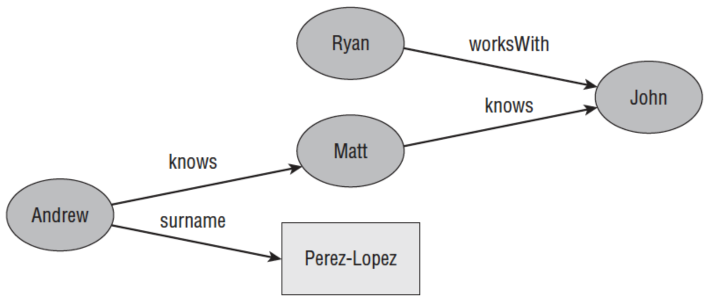


Figure 12 Graph representation of the information about some people

Table 3 An RDF Triple Expressed in Table Format

Subject	Predicate	Object
Ryan	Works With	John

Table 4 An RDF Triple Expressed in Table Format

Subject	Predicate	Object
Matt	Knows	John

Table 5 An RDF Triple Expressed in Table Format

Subject	Predicate	Object
Andrew	Knows	Matt

For example, Figure 11 shows a small dataset of information, which is graphically depicted in Figure 12 is a graphical representation of the small dataset depicted in figure 11. The first level above RDF required for the Semantic Web is an ontology language that can formally describe the meaning of terminology used in Web documents. If machines are expected to perform useful reasoning tasks on these documents, the language must go beyond the basic semantics of RDF Schema."[76] "Key to the implementation of the Semantic Web is that data be structured"[96] into a common vocabulary called an ontology.

2.12 **Web Ontology Language (OWL)**

The W3C OWL 2 Web Ontology Language is a semantic web language designed for the representation of rich and complex knowledge about things, groups of things and relations between things. A computational logic-based language, it allows knowledge expressed in OWL to be reasoned through by computer programs either for the verification of consistency of the knowledge being represented or to make implicit knowledge explicit. It is designed for use by applications that need to process the content of information instead of just presenting information to humans. OWL facilitates greater machine interpretability of Web content than that supported by XML, RDF, and RDF Schema (RDFS).[116]

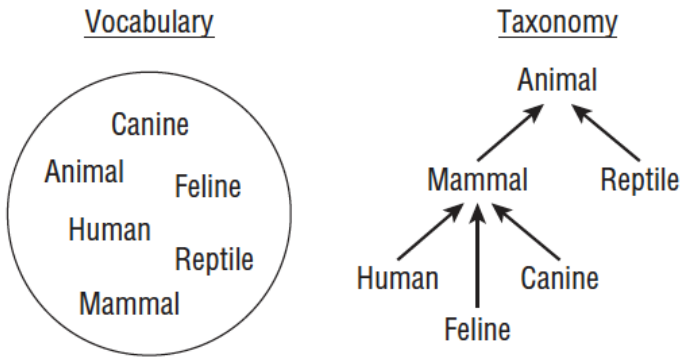


Figure 13 A Vocabulary and a Taxonomy

An ontology is a rich, formal logic-based model for using semantics to describe knowledge about objects in a domain. Using a pre-defined, reserved vocabulary of terms to define concepts and relationships between them, an ontology can refer to either a vocabulary, taxonomy or something more. “The OWL Web Ontology Language provides an expressive language for defining ontologies that capture the semantics of domain knowledge.”[3]

OWL documents are also known as ontologies, and can be published in the World Wide Web and may be referred to from other OWL ontologies. OWL is part of the W3C semantic web technology stack, which also includes RDF and SPARQL.[117] OWL is built on RDF Schema, and is one of the most popular languages for creating ontologies today. RDF is limited. OWL is more expressive than RDF, and with it, much more complex and richer relationships may be defined. OWL = RDF schema + new constructs for OWL and RDF schema have the same purpose; to define classes, properties, and their relationships. All the classes and properties of RDF schema can be used in creating an OWL document providing the ability to construct agents or tools with greatly enhanced reasoning ability.[97]

The disadvantages of RDF and OWL are a lack of support for procedural functions like arithmetic and string manipulation functions that are needed in real-world applications.[31]

2.13 **PaceProtégé**

Another component of this research is to take advantage of Pace's extension to the open source Stanford university Protégé. It is a large open source project which can be compiled using a Maven script that was provided with the project. Once it is compiled and run, it launches Protégé.

The extension that Pace contributed to this open source project is the relations' tab. Version 5 of protégé that is available for download from the Git website, does not have the relations tab.

This extended feature boasts the capability of representing new custom relations. For example, the partOf composition construct. This is explained in greater detail in section 2.15.3.

2.14 **PaceJena**

The open-source Jena has been simplified and extended to support Pace computing research and is called PaceJena. As an additional benefit of this research, PaceJena is invoked upon validation failure, and the error code in the failing instance document is passed to it. PaceJena parses the OWL knowledge graph file to extract semantics and relations pertinent to other errors, other classes and objects, if any. As shown in figure 22 (chapter 3), this dynamic error code explication enables semantics, and object relations to be packaged with actionable errors that must be escalated to technical support staff.

2.15 **Knowledge Representation**

2.15.1 Association

Association is a (*a*) relationship between two classes, which allows one object instance to cause another to perform an action on its behalf. It defines a *has-a* relationship between two classes where there is no particular ownership in place. Association is the more general term that defines the relationship between two classes, where as aggregation and composition are relatively special. Aggregation is a weak type of Association with partial ownership. The term used for an Aggregation relationship is *uses* to imply a weak *has-a* relationship. This is weak compared to Composition, meaning the linked components of the aggregator may survive the aggregations life-cycle without the existence of their parent objects. “For example, a school department *uses* teachers. Any teacher may belong to more than one department. And so, if a department ceases to exist, the teacher will still exist.”[70]

2.15.2 is_a relationship

One of the most important relationships among objects in the real world is specialization. Specialization can be described as the “*is-a*” relationship. The statement, “A dog is a mammal”, means that the dog is a specialized kind of mammal. Having all the characteristics of any mammal, (the fact that it bears live young, nurses with milk, has hair etc.), it specializes these characteristics to the familiar characteristics of canis domesticus.

“The specialization and generalization relationships are both reciprocal and hierarchical. Specialization is just the other side of the generalization coin: Mammal generalizes what is common between dogs and cats, and dogs and cats specialize mammals to their own specific subtypes.”[70]

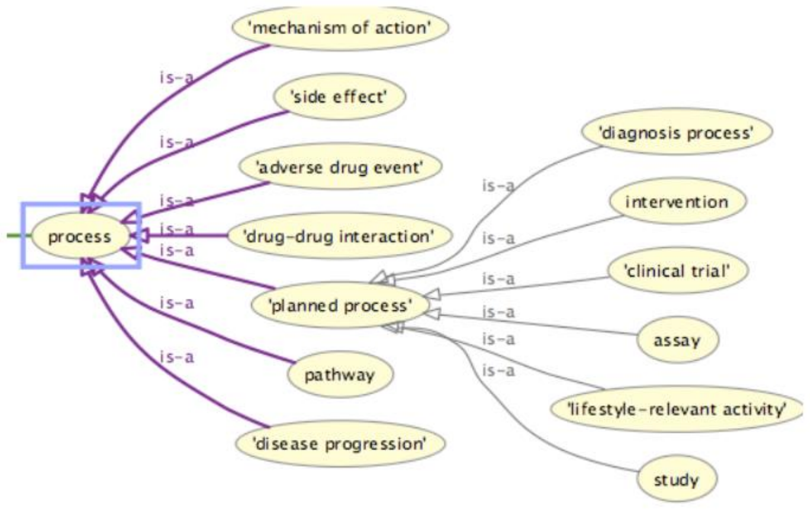


Figure 14 Knowledge Representation Solely With the is_a Relation

Figure 14 illustrates a protégé class view of knowledge being represented solely with the less expressive is_a inheritance structure.

2.15.3 part_of relationship

The *part_of* relationship is more formally known as composition.[70] Composition is a strong type of Association with full ownership. The term used for a Composition relationship, is **owns** or **part_of* *to imply a strong **has-a** relationship. For example, a department **owns** courses, which means that any course's life-cycle depends on the department's life-cycle. Hence, if a department ceases to exist, the underlying courses will cease to exist as well." Relationships with no ownership in place are regarded as just an Association and the term used is **has-a**, or sometimes the verb describing the relationship. "For example, a teacher **has-a** or **teaches** a student. There is no ownership between the teacher and the student, and each has their own life-cycle".[70]

Protégé has emerged as one of the most popular interfaces for creating knowledge representation. The standard open source protégé primarily uses the inheritance relationship, *is_a* to represent knowledge. Even though current OWL files can emulate the custom relations by using this approach, relying solely on the is_a inheritance relationship construct to describe knowledge is rather restrictive and some domain experts find it awkward to understand and to use it to validate their knowledge representations.

In the quest for a more expressive way of representing knowledge, this research dispenses with these complexities and adds to *is_a* constructs for custom relations using a Knowledge Graph, which is a more natural and flexible way to represent knowledge, and can be used to overcome the limitations and restrictiveness of the sole *is_a* relation.

2.16 **Knowledge Graph**

The Knowledge Graph is not an actual graph. It is a type of XML document. OWL is extended by defining custom relations and applications; the result is called "Knowledge Graph". Knowledge Graphs can be created with any text editor. It can also be created in Protégé, an optional IDE. Using custom relations enables the use of the *part_of* or *partOf* composition relation, and based on the domain, any additional custom relations can be introduced and defined in a straightforward, flexible and expressive way by Subject Matter Experts (SMEs) or other domain experts.

2.17 **Conclusion**

This chapter covered all the technologies relevant to support this research, XPointer, XPath, XQuery, XPattern RDF, Schematron, Semantic Web, PaceProtégé, PaceJena, OWL and knowledge graphs were described in detail. All these technologies support the framework that enables the cost-effective integration of data from diverse sources containing similar information encoded in different dialects. The need for knowledge graphs was explained. As will be shown in the next chapter, the ability to define custom relations is very valuable for the sort of data integration, and error explication, through the local knowledge graph approach that is addressed by this research. This research solution methodology offers an alternative to costly full-blown integration.

Chapter 3

Solution Methodology

3.1 Solution Strategy for the XML Semantic Dialect Proliferation Problem

The problem is finding a cost-effective means of integrating data from disparate sources by finding an approach to successfully reduce the complexity of files caused by multiple dialects and providing a convenient semantic validation symptom code explication.

This research seeks to provide a way of handling the Semantic Rule Complexity and Heterogeneity Problem (SERCH) in a uniform way by using semantics pattern description for the classification of diagnostic information coupled with knowledge graphs for explication of actionable error data. After determining the classification of the diagnostic information and its matching semantic design pattern, Schematron is used to perform semantic constraint validation, and a knowledge graph is introduced to define extra concepts and their relationships, explain the meaning of the symptom code, and reduce the complexity of maintaining various versions of semantic patterns.

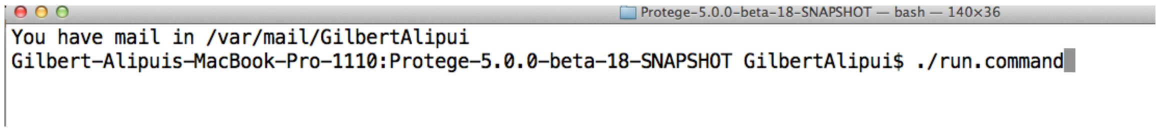
By overcoming the limitations of one inheritance relation, this research framework:

1. Is adaptive, and capable of classifying different kinds of system diagnostic information for different products.

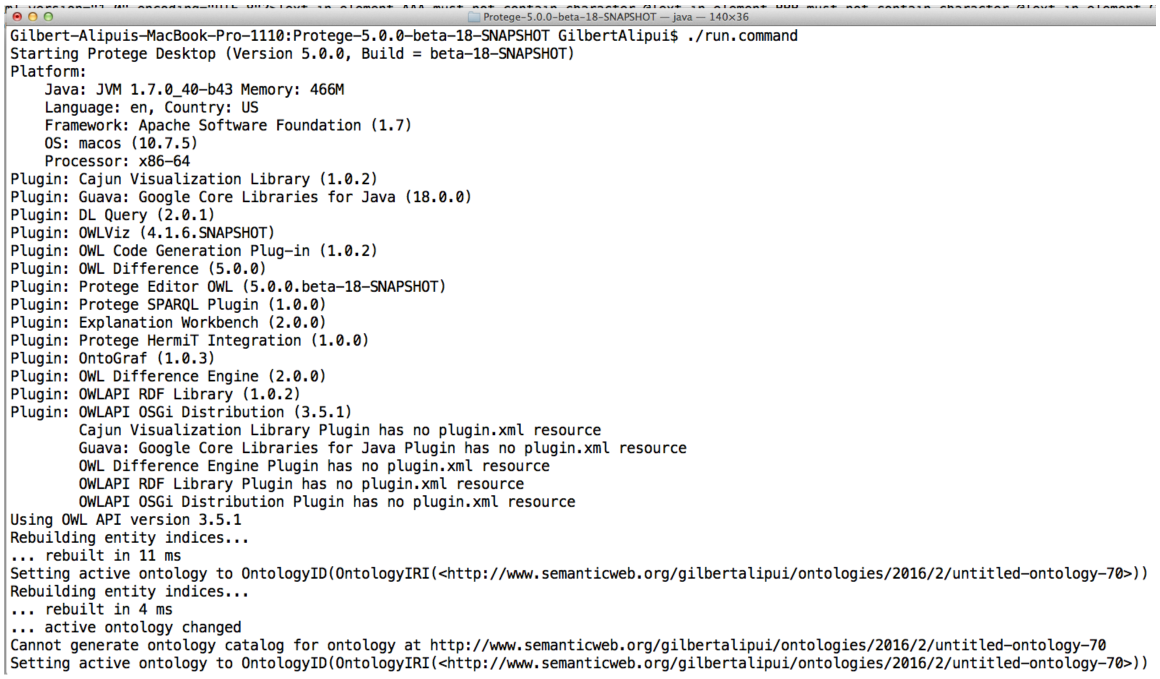
2. Uses semantic constraints to define patterns for classifying the various diagnostic information.
3. Uses a knowledge graph to reduce the complexity of maintaining multiple versions of similar patterns.

3.2 Using a Knowledge Graph to Specify Relations and Explications

Web Ontology Language (OWL) knowledge graph is for specifying custom relations and explications. Figure 15 demonstrates how to launch PaceProtégé on the command line.

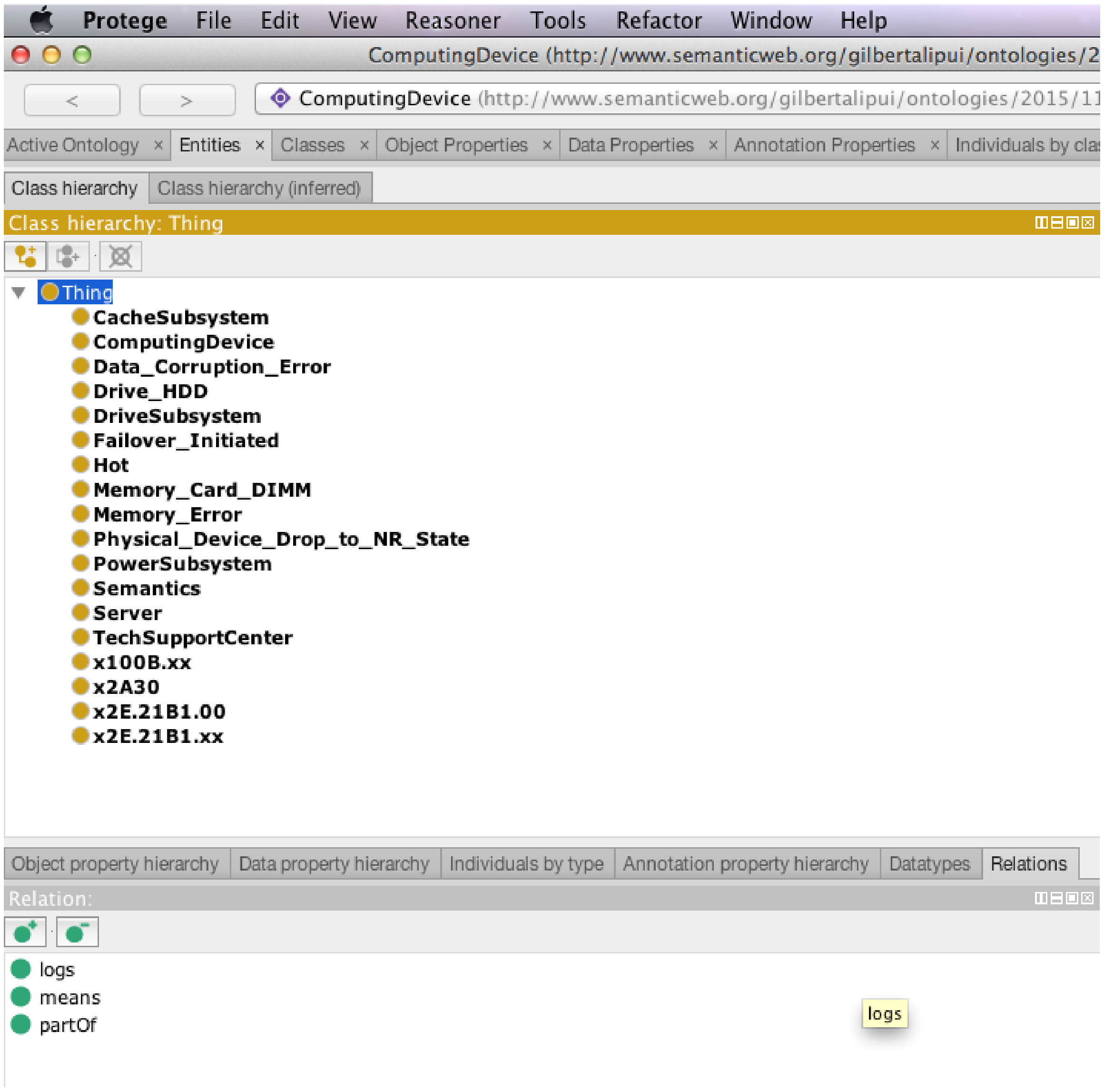


```
You have mail in /var/mail/GilbertAlipui
Gilbert-Alipuis-MacBook-Pro-1110:Protege-5.0.0-beta-18-SNAPSHOT GilbertAlipui$ ./run.command
```



```
Gilbert-Alipuis-MacBook-Pro-1110:Protege-5.0.0-beta-18-SNAPSHOT GilbertAlipui$ ./run.command
Starting Protege Desktop (Version 5.0.0, Build = beta-18-SNAPSHOT)
Platform:
    Java: JVM 1.7.0_40-b43 Memory: 466M
    Language: en, Country: US
    Framework: Apache Software Foundation (1.7)
    OS: macos (10.7.5)
    Processor: x86-64
Plugin: Cajun Visualization Library (1.0.2)
Plugin: Guava: Google Core Libraries for Java (18.0.0)
Plugin: DL Query (2.0.1)
Plugin: OWLViz (4.1.6.SNAPSHOT)
Plugin: OWL Code Generation Plug-in (1.0.2)
Plugin: OWL Difference (5.0.0)
Plugin: Protege Editor OWL (5.0.0.beta-18-SNAPSHOT)
Plugin: Protege SPARQL Plugin (1.0.0)
Plugin: Explanation Workbench (2.0.0)
Plugin: Protege HermiT Integration (1.0.0)
Plugin: OntoGraf (1.0.3)
Plugin: OWL Difference Engine (2.0.0)
Plugin: OWLAPI RDF Library (1.0.2)
Plugin: OWLAPI OSGi Distribution (3.5.1)
        Cajun Visualization Library Plugin has no plugin.xml resource
        Guava: Google Core Libraries for Java Plugin has no plugin.xml resource
        OWL Difference Engine Plugin has no plugin.xml resource
        OWLAPI RDF Library Plugin has no plugin.xml resource
        OWLAPI OSGi Distribution Plugin has no plugin.xml resource
Using OWL API version 3.5.1
Rebuilding entity indices...
... rebuilt in 11 ms
Setting active ontology to OntologyID(OntologyIRI(<http://www.semanticweb.org/gilbertalipui/ontologies/2016/2/untitled-ontology-70>))
Rebuilding entity indices...
... rebuilt in 4 ms
... active ontology changed
Cannot generate ontology catalog for ontology at http://www.semanticweb.org/gilbertalipui/ontologies/2016/2/untitled-ontology-70
Setting active ontology to OntologyID(OntologyIRI(<http://www.semanticweb.org/gilbertalipui/ontologies/2016/2/untitled-ontology-70>))
```

Figure 15 Launching PaceProtégé on the Command Line

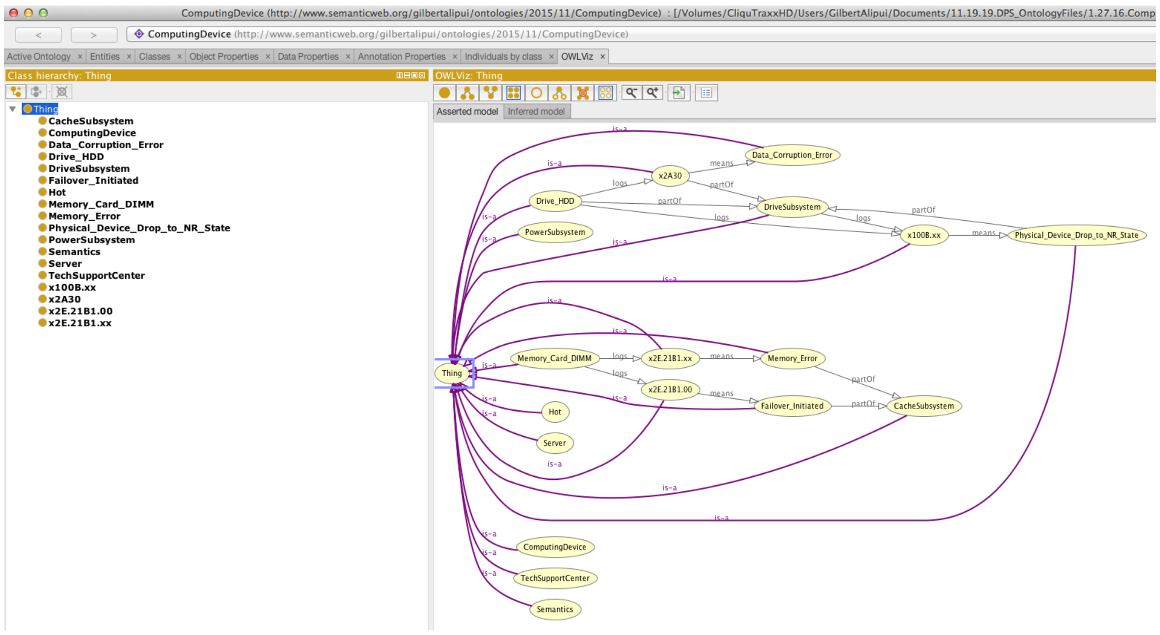

Protege File Edit View Reasoner Tools Refactor Window Help
ComputingDevice (http://www.semanticweb.org/gilbertalipui/ontologies/2
ComputingDevice (http://www.semanticweb.org/gilbertalipui/ontologies/2015/11
Active Ontology
Entities
Classes
Object Properties
Data Properties
Annotation Properties
Individuals by clas
Class hierarchy
Class hierarchy (inferred)
Class hierarchy: Thing

- Thing
  - CacheSubsystem
  - ComputingDevice
  - Data_Corruption_Error
  - Drive_HDD
  - DriveSubsystem
  - Failover_Initiated
  - Hot
  - Memory_Card_DIMM
  - Memory_Error
  - Physical_Device_Drop_to_NR_State
  - PowerSubsystem
  - Semantics
  - Server
  - TechSupportCenter
  - x100B.xx
  - x2A30
  - x2E.21B1.00
  - x2E.21B1.xx

Object property hierarchy
Data property hierarchy
Individuals by type
Annotation property hierarchy
Datatypes
Relations
Relation:

- logs
- means
- partOf

Figure 16 PaceProtégé Class View Showing Custom Relations


ComputingDevice (http://www.semanticweb.org/gilbertalipui/ontologies/2015/11/ComputingDevice) : [/Volumes/CliquTraxxHD/Users/GilbertAlipui/Documents/11.19.19.DPS_OntologyFiles/1.27.16.Comp

ComputingDevice (http://www.semanticweb.org/gilbertalipui/ontologies/2015/11/ComputingDevice)

Class hierarchy: Thing

- Thing
  - CacheSubsystem
  - ComputingDevice
  - Data_Corruption_Error
  - Drive_HDD
  - DriveSubsystem
  - Failover_Initiated
  - Hot
  - Memory_Card_DIMM
  - Memory_Error
  - Physical_Device_Drop_to_NR_State
  - PowerSubsystem
  - Semantics
  - Server
  - TechSupportCenter
  - x100B.xx
  - x2A30
  - x2E.21B1.00
  - x2E.21B1.xx

Figure 17 PaceProtégé Class View Showing Linked Data

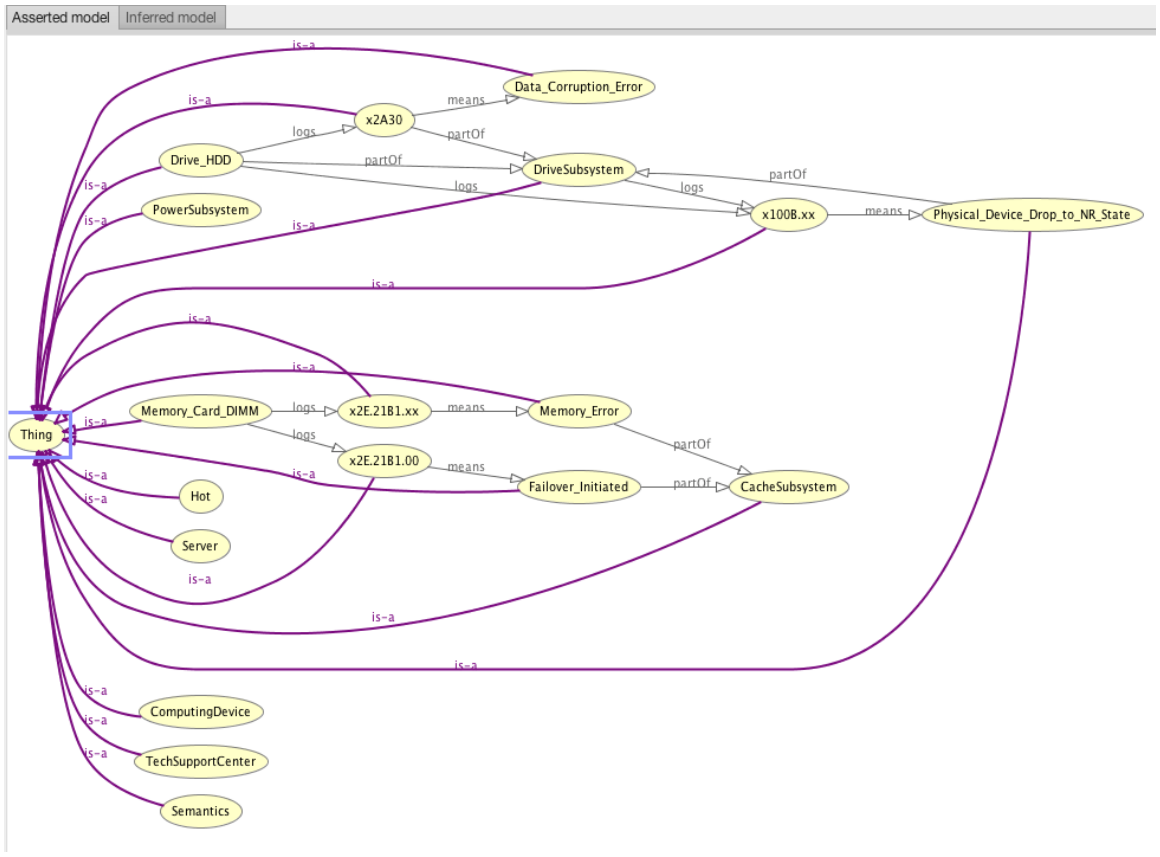


Figure 18 Complex and Rich Relationships Being Modeled Between Domain Objects, Classes and Data

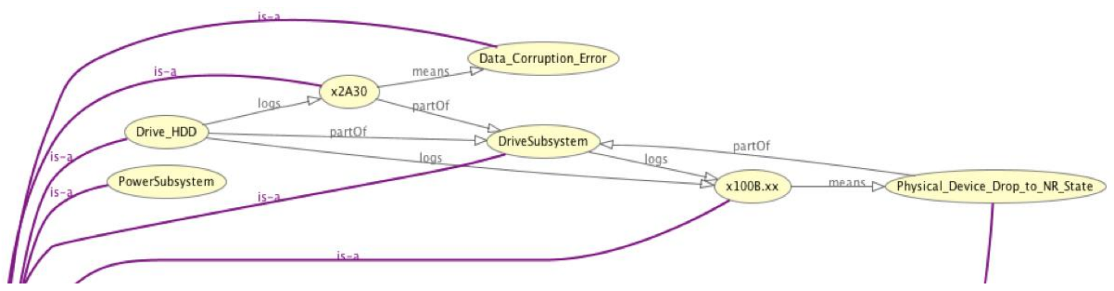


Figure 19 Knowledge Graph Using Extended Custom Relations to Describe a Hard Drive

The custom relations extend the ontology to a knowledge graph. The custom relations in this example are:

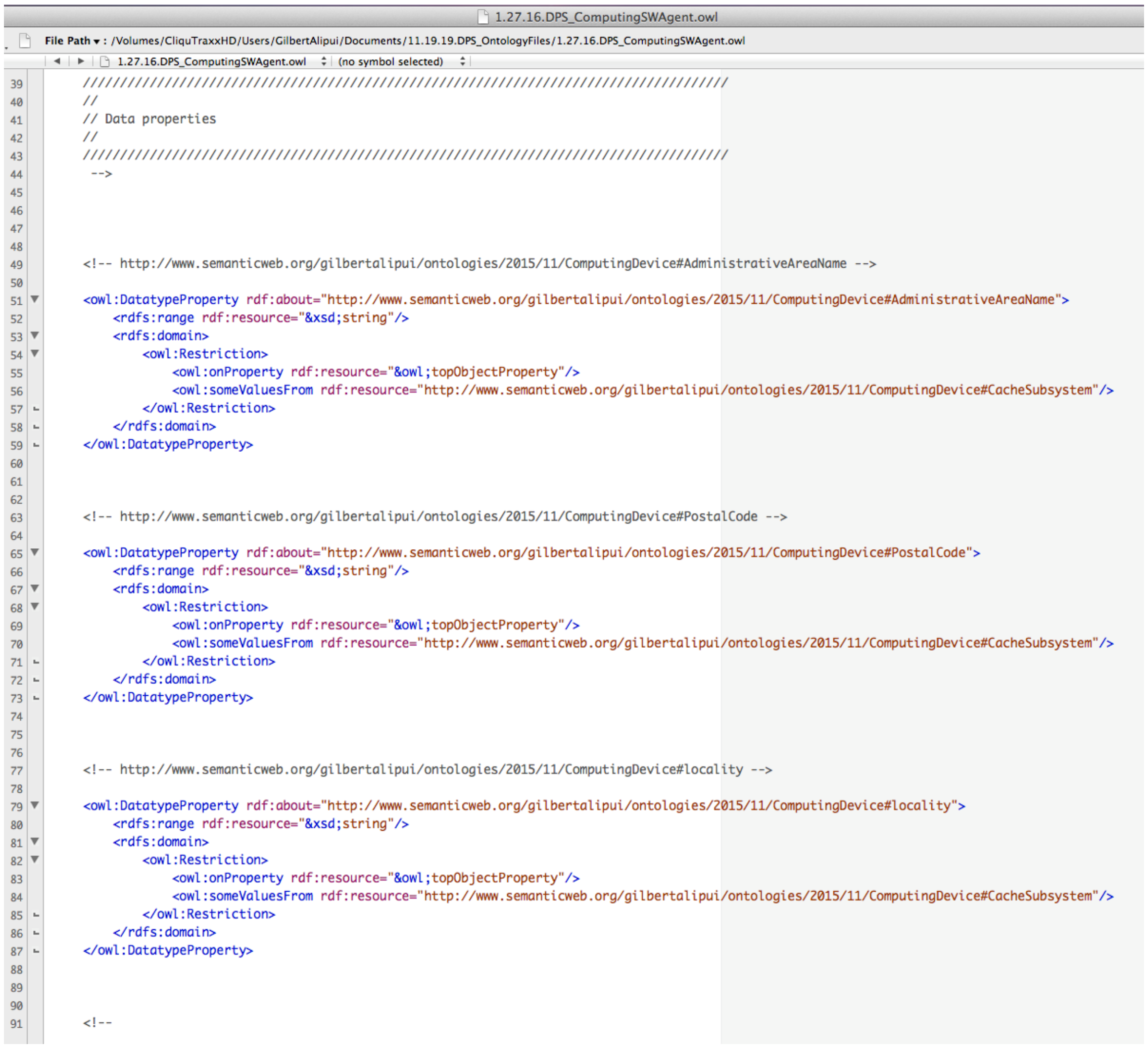
1. Logs
2. Means
3. Part of

These custom relations provide a more convenient and flexible means of describing elements in a given problem domain. For example in figure 20 above, the knowledge graph shows that:

1. A hard drive is part of the drive subsystem.

2. A hard drive logs the 2A30 error, which means data corruption.

3. A hard drive also logs the 100B error, which means that the physical device has dropped into a Not-Ready State.



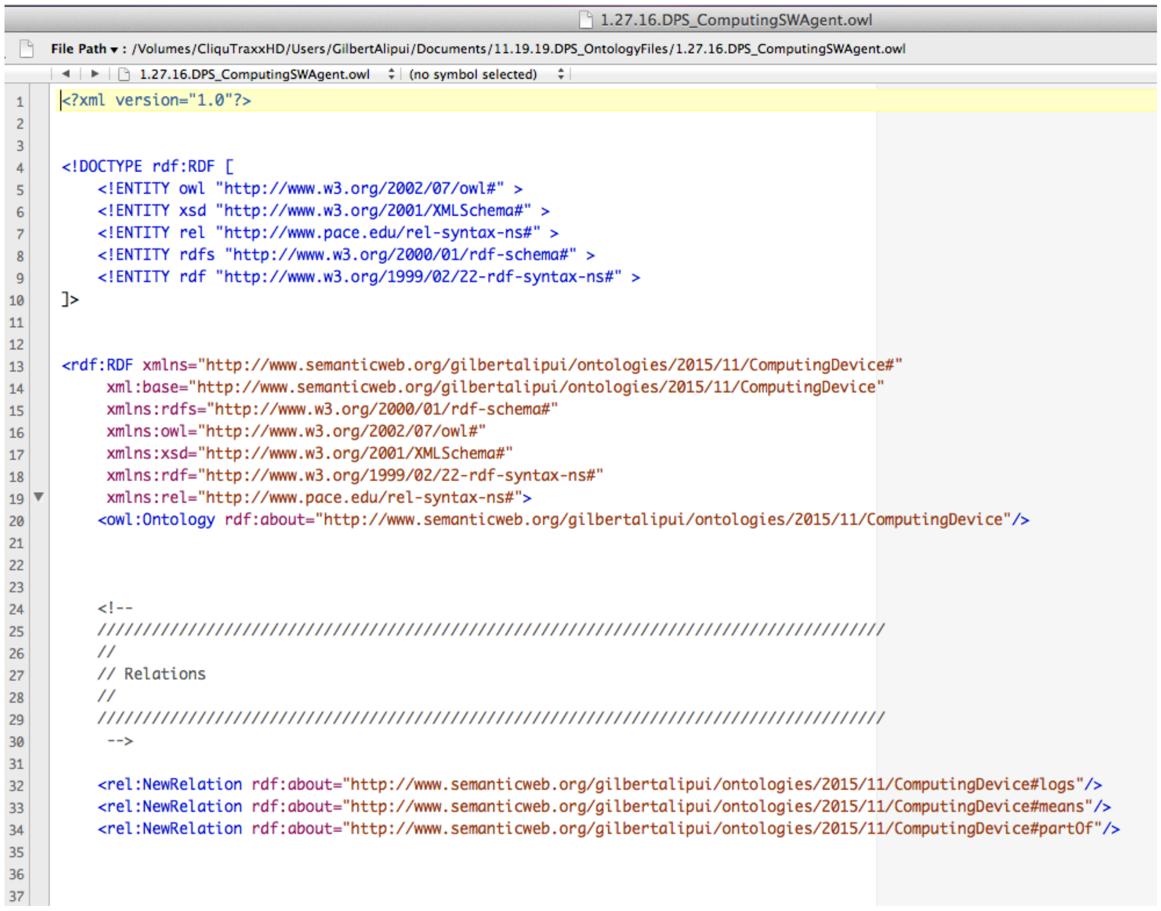
1.27.16.DPS_ComputingSWAgent.owl

File Path ▾ : /Volumes/CliquTraxxHD/Users/GilbertAlipui/Documents/11.19.19.DPS_OntologyFiles/1.27.16.DPS_ComputingSWAgent.owl

1.27.16.DPS_ComputingSWAgent.owl (no symbol selected)

```
    ///////////////////////////////////////////////////////////////////////////////////////
    //
    // Data properties
    //
    ///////////////////////////////////////////////////////////////////////////////////////
     -->




    <!-- http://www.semanticweb.org/gilbertalipui/ontologies/2015/11/ComputingDevice#AdministrativeAreaName -->

    <owl:DatatypeProperty rdf:about="http://www.semanticweb.org/gilbertalipui/ontologies/2015/11/ComputingDevice#AdministrativeAreaName">
        <rdfs:range rdf:resource="&xsd;string"/>
        <rdfs:domain>
            <owl:Restriction>
                <owl:onProperty rdf:resource="&owl;topObjectProperty"/>
                <owl:someValuesFrom rdf:resource="http://www.semanticweb.org/gilbertalipui/ontologies/2015/11/ComputingDevice#CacheSubsystem"/>
            </owl:Restriction>
        </rdfs:domain>
    </owl:DatatypeProperty>



    <!-- http://www.semanticweb.org/gilbertalipui/ontologies/2015/11/ComputingDevice#PostalCode -->

    <owl:DatatypeProperty rdf:about="http://www.semanticweb.org/gilbertalipui/ontologies/2015/11/ComputingDevice#PostalCode">
        <rdfs:range rdf:resource="&xsd;string"/>
        <rdfs:domain>
            <owl:Restriction>
                <owl:onProperty rdf:resource="&owl;topObjectProperty"/>
                <owl:someValuesFrom rdf:resource="http://www.semanticweb.org/gilbertalipui/ontologies/2015/11/ComputingDevice#CacheSubsystem"/>
            </owl:Restriction>
        </rdfs:domain>
    </owl:DatatypeProperty>



    <!-- http://www.semanticweb.org/gilbertalipui/ontologies/2015/11/ComputingDevice#locality -->

    <owl:DatatypeProperty rdf:about="http://www.semanticweb.org/gilbertalipui/ontologies/2015/11/ComputingDevice#locality">
        <rdfs:range rdf:resource="&xsd;string"/>
        <rdfs:domain>
            <owl:Restriction>
                <owl:onProperty rdf:resource="&owl;topObjectProperty"/>
                <owl:someValuesFrom rdf:resource="http://www.semanticweb.org/gilbertalipui/ontologies/2015/11/ComputingDevice#CacheSubsystem"/>
            </owl:Restriction>
        </rdfs:domain>
    </owl:DatatypeProperty>



    <!--
```

Figure 20 Example of the OWL File


1.27.16.DPS_ComputingSWAgent.owl

File Path ▾ : /Volumes/CliquTraxxHD/Users/GilbertAlipui/Documents/11.19.19.DPS_OntologyFiles/1.27.16.DPS_ComputingSWAgent.owl

1.27.16.DPS_ComputingSWAgent.owl (no symbol selected)

```
<?xml version="1.0"?>


<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY rel "http://www.pace.edu/rel-syntax-ns#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>


<rdf:RDF xmlns="http://www.semanticweb.org/gilbertalipui/ontologies/2015/11/ComputingDevice#"
     xml:base="http://www.semanticweb.org/gilbertalipui/ontologies/2015/11/ComputingDevice"
     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:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:rel="http://www.pace.edu/rel-syntax-ns#">
    <owl:Ontology rdf:about="http://www.semanticweb.org/gilbertalipui/ontologies/2015/11/ComputingDevice"/>



    <!--
    ///////////////////////////////////////////////////////////////////////////////////////
    //
    // Relations
    //
    ///////////////////////////////////////////////////////////////////////////////////////
     -->

    <rel:NewRelation rdf:about="http://www.semanticweb.org/gilbertalipui/ontologies/2015/11/ComputingDevice#logs"/>
    <rel:NewRelation rdf:about="http://www.semanticweb.org/gilbertalipui/ontologies/2015/11/ComputingDevice#means"/>
    <rel:NewRelation rdf:about="http://www.semanticweb.org/gilbertalipui/ontologies/2015/11/ComputingDevice#partOf"/>
```

Figure 21 Custom Relations in the OWL File

As stated in Chapter 2, the disadvantages of RDF and OWL are a lack of support for procedural functions like arithmetic and string manipulation functions that are needed in real-world applications.[31] This research targets this gap in this area of data integration. The challenges is how to encode diagnostic knowledge in a way that enables computers to automatically apply them to diagnostic messages, and how to manage the complexity of similar but different diagnostic semantics based on different terms.

We propose to use Schematron to specify the semantic constraints, and develop knowledge graphs in Pace-extended OWL for capturing various dependencies among system components with customized relations to support specification of semantic constraints with ontology concepts instead of XML tag and attribute names. We modify Pace Schematron Validator to support the resulting language extensions.

3.3 Using Schematron to Specify Diagnostic Constraints Patterns

Schematron is used for constraint validation, to specify XML sematic patterns validation. This research proposes to extend Schematron for XML diagnostic message processing and error explication by coupling Schematron with knowledge graphs. This coupling takes the form of Schematron validations, used in tandem with OWL ontologies, to promote data integration whist, identifying and explaining actionable system problems based on product diagnostic symptom codes.

3.4 Detailed Solution Workflow

The flowchart in Figure 23 below, illustrates the general flow of the solution methodology. The process starts when a file arrives. The research solution program inspects the received file to try to identify the dialect. If the dialect is recognized, it proceeds to the abstract level to extract the abstract rule, performs the abstract-to-concrete rule generation, populates the place-holder variables with actual values, and proceeds on to validate the received diagnostic message file. If the validation is successful and it is able to identify an actionable error, that has the potential to cause a critical problem, it emits an alert. PaceJena is then invoked to extract the semantics and relations relevant to that error, and the information is escalated to the technical support staff to prevent catastrophic damage to business-critical systems. If the error is spurious and deemed trivial, then it just discards the file (Figure 22) and goes back to the beginning to check for additional files, and if additional files exist, it goes through the entire process again for the new files.

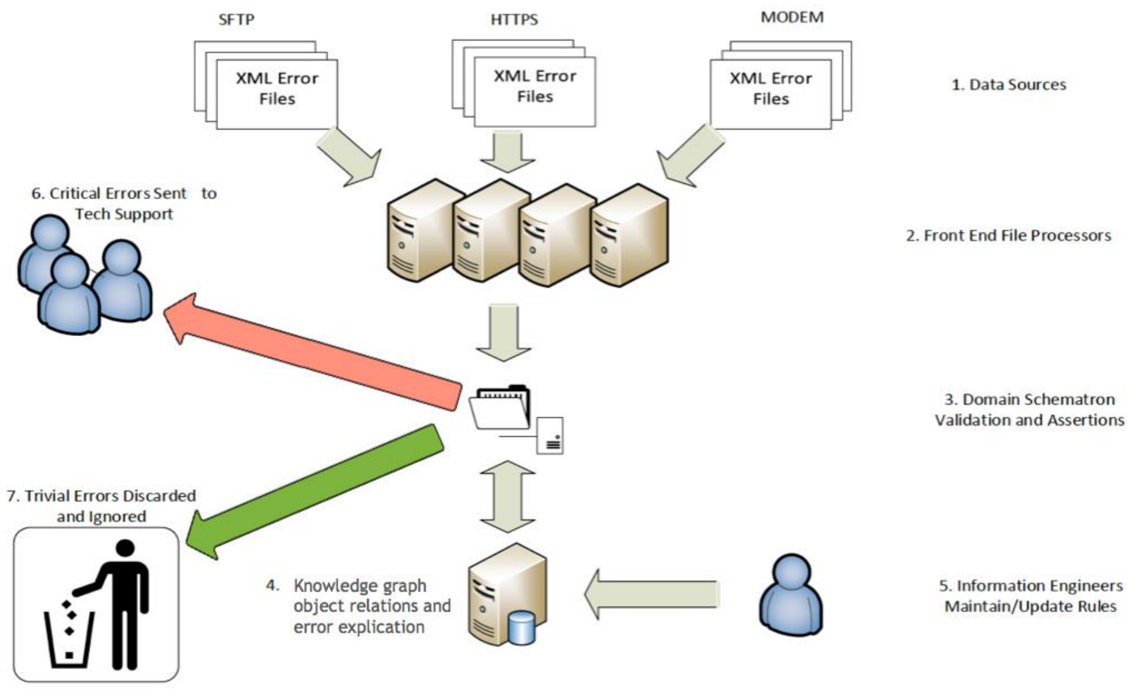

SFTP
HTTPS
MODEM
XML Error Files
XML Error Files
XML Error Files
1. Data Sources
6. Critical Errors Sent to Tech Support
2. Front End File Processors
3. Domain Schematron Validation and Assertions
7. Trivial Errors Discarded and Ignored
4. Knowledge graph object relations and error explication
5. Information Engineers Maintain/Update Rules

Figure 22 System Architecture for Data Integration and Knowledge-driven Error Classification

When all the files are processed and there are no more files to process, the process terminates. This process can be automated with a cron job or scheduled task.

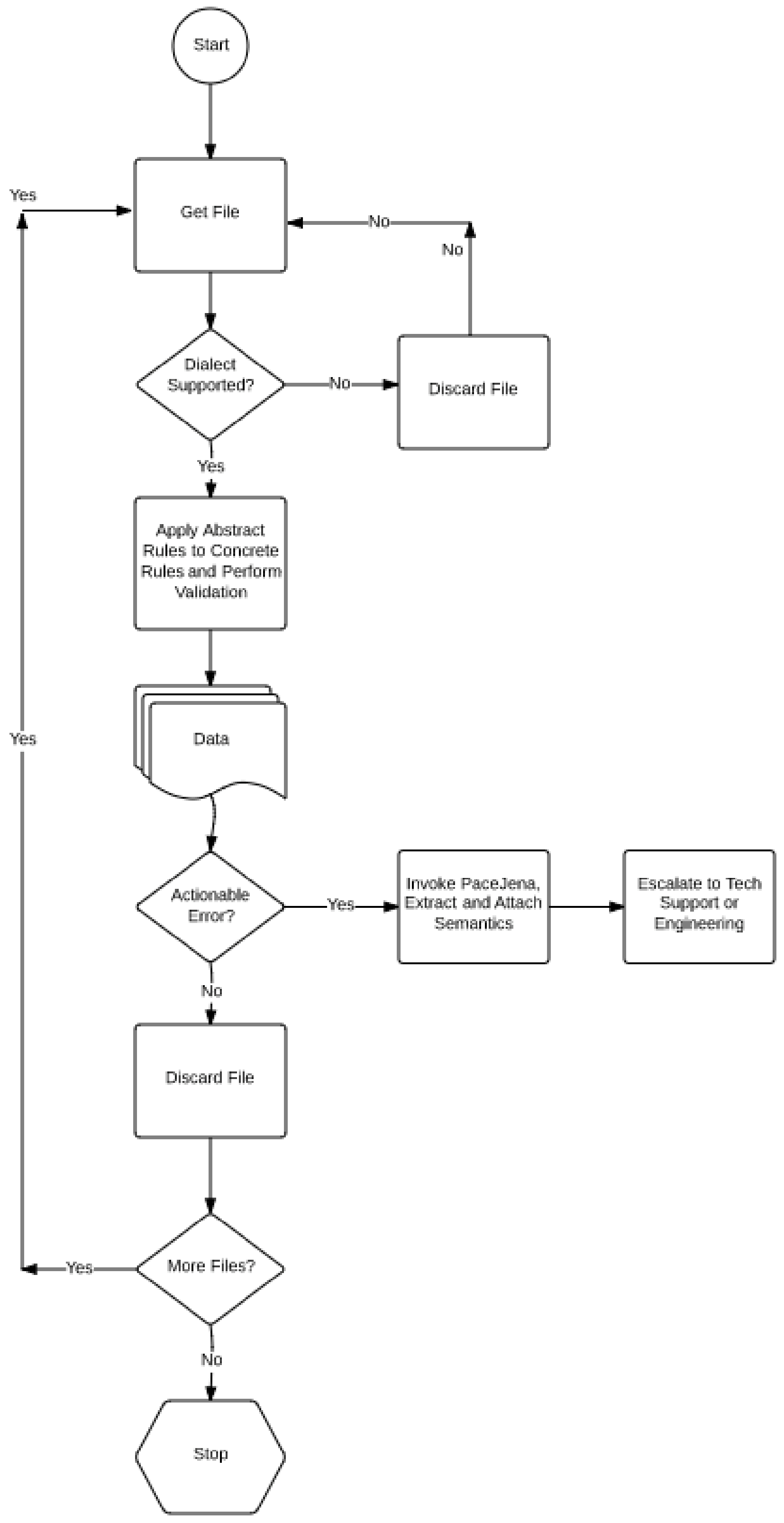
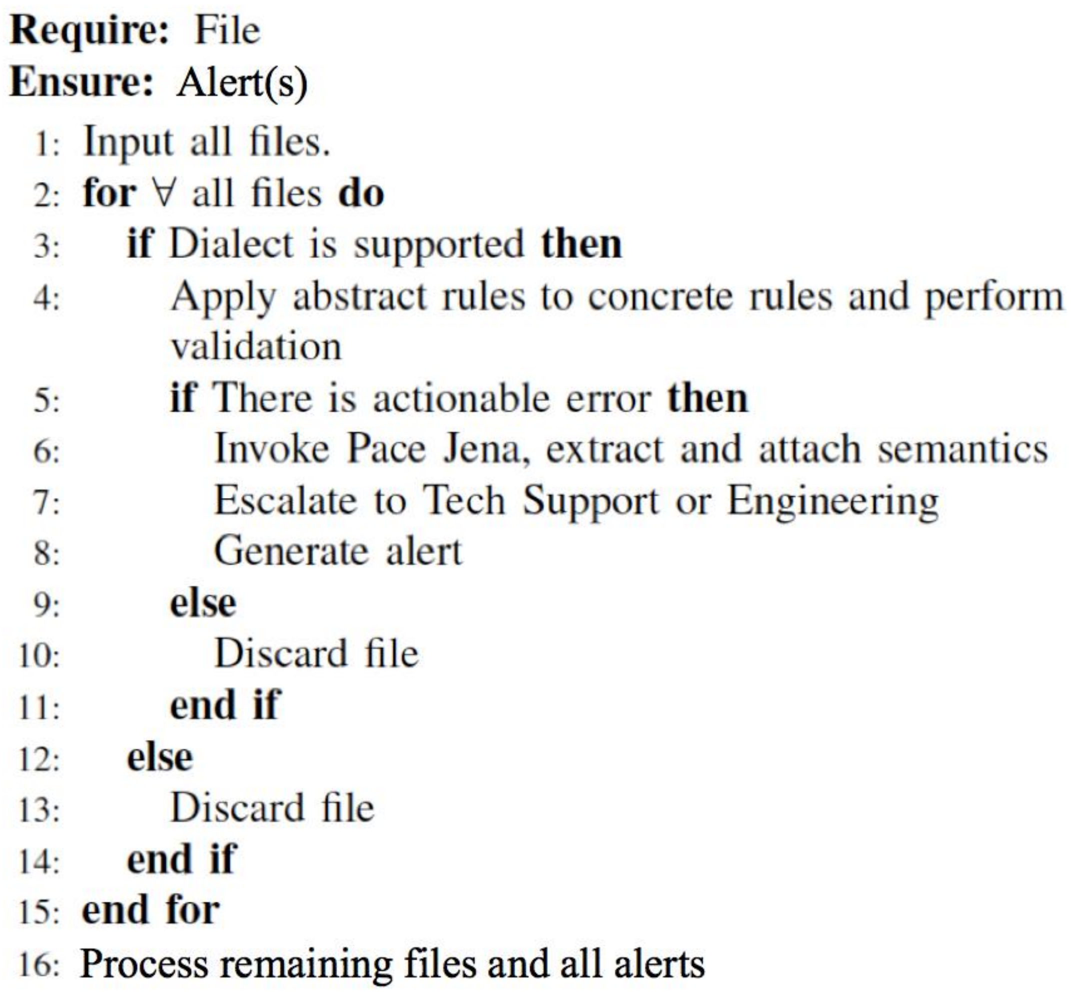

Start
Get File
Yes
No
No
Dialect Supported?
No
Discard File
Yes
Apply Abstract Rules to Concrete Rules and Perform Validation
Data
Yes
Actionable Error?
Yes
Invoke PaceJena, Extract and Attach Semantics
Escalate to Tech Support or Engineering
No
Discard File
Yes
More Files?
No
Stop

Figure 23 Solution Process Flowchart

3.4.1 Algorithm. Dialect Identification (DI) Algorithm

To address the complexity caused by multiple dialects, we propose a Dialect Identification (DI) algorithm. The pseudocode of the DI algorithm is shown in figure 24 below.



```
Require: File
Ensure: Alert(s)
 1: Input all files.
 2: for ∀ all files do
 3:     if Dialect is supported then
 4:         Apply abstract rules to concrete rules and perform
            validation
 5:         if There is actionable error then
 6:             Invoke Pace Jena, extract and attach semantics
 7:             Escalate to Tech Support or Engineering
 8:             Generate alert
 9:         else
10:             Discard file
11:         end if
12:     else
13:         Discard file
14:     end if
15: end for
16: Process remaining files and all alerts
```

Figure 24 Pseudocode for the Dialect Identification Algorithm

3.4.2 Diagnostic Message File Use-Case

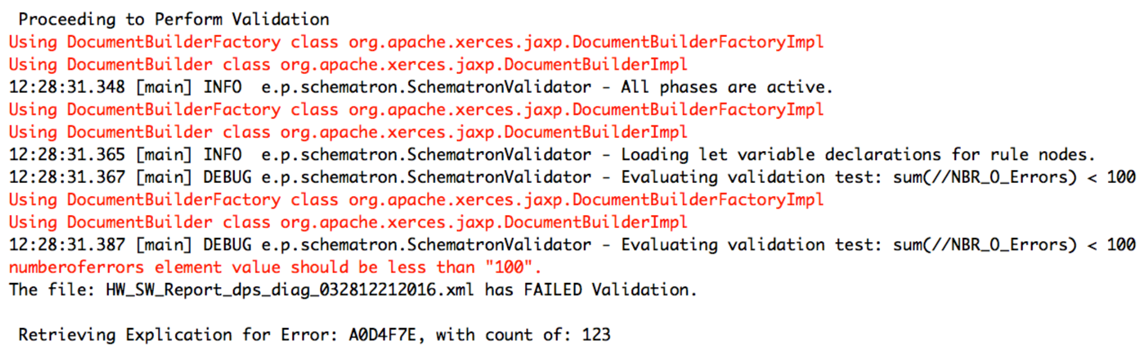
Chapter 1 used a mailing address to describe the SERCH problem. Now, consider the same scenario, except this time with XML files containing diagnostic system health status information. As shown in figure 24 below, the XML System Health Report file from a system or software agent contains identifying information along with diagnostic error code and other pertinent information. Figure 25 shows the listing for a Schematron document used to make sure that the pre-set safety threshold of 100 occurences is not exceeded for the error in the given XML instance file.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<dpsrule type="System Health Report">
    <siteinfo id="1">
    <siteid>100099</siteid>
    <company>ACME</company>
    <model>Viper IO</model>
    <error>07D4FE2</error>
    <xpath>//numberoferrors</xpath>
    <numberoferrors>105</numberoferrors>
  </siteinfo>
</dpsrule>
```

Figure 25 XML Listing with Customer Information and Error Code

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron">
  <pattern name="Sum Less Than 100">
    <rule context="/">
      <assert test=""sum(//numberoferrors) &lt; 100">numberoferrors element
              value should less than "100".
      </assert>
     </rule>
  </pattern>
</schema>
```

Figure 26 Listing of PaceSchematron Rule to Enforce Error Safety Threshold



```
 Proceeding to Perform Validation
Using DocumentBuilderFactory class org.apache.xerces.jaxp.DocumentBuilderFactoryImpl
Using DocumentBuilder class org.apache.xerces.jaxp.DocumentBuilderImpl
12:28:31.348 [main] INFO  e.p.schematron.SchematronValidator - All phases are active.
Using DocumentBuilderFactory class org.apache.xerces.jaxp.DocumentBuilderFactoryImpl
Using DocumentBuilder class org.apache.xerces.jaxp.DocumentBuilderImpl
12:28:31.365 [main] INFO  e.p.schematron.SchematronValidator - Loading let variable declarations for rule nodes.
12:28:31.367 [main] DEBUG e.p.schematron.SchematronValidator - Evaluating validation test: sum(//NBR_O_Errors) < 100
Using DocumentBuilderFactory class org.apache.xerces.jaxp.DocumentBuilderFactoryImpl
Using DocumentBuilder class org.apache.xerces.jaxp.DocumentBuilderImpl
12:28:31.387 [main] DEBUG e.p.schematron.SchematronValidator - Evaluating validation test: sum(//NBR_O_Errors) < 100
numberoferrors element value should be less than "100".
The file: HW_SW_Report_dps_diag_032812212016.xml has FAILED Validation.

 Retrieving Explication for Error: A0D4F7E, with count of: 123
```

Figure 27 PaceSchematron Validation Failure on Safety Threshold Exceeded

A typical use case is to determine if a preset safety threshold has been breached or exceeded. In this case the total count of errors received clearly exceeds the hundred safety threshold so the validation has Schematron triggers, and emits an error message to alert to the failure.

3.5 Pace-Extended Schematron

This research employs the console application version of PaceSchematron. When run from the command line this extended version performs semantic constraints validation on the XML file specified on the command line, and upon failure invokes PaceJena for sematic explanation of the error code.

3.6 The Abstract Concept

There are three levels of knowledge abstraction that must be considered for proper, formalized knowledge representation. As shown in figure 27 below, knowledge representation flows from the more abstract at the very top to the more concrete on the bottom. The very top level is the methodological knowledge where modeling of the data is done using OWL. The middle level is the conceptual level where the conceptual model is built, and then below that is the factual knowledge level where specific, concrete description of the data is handled.

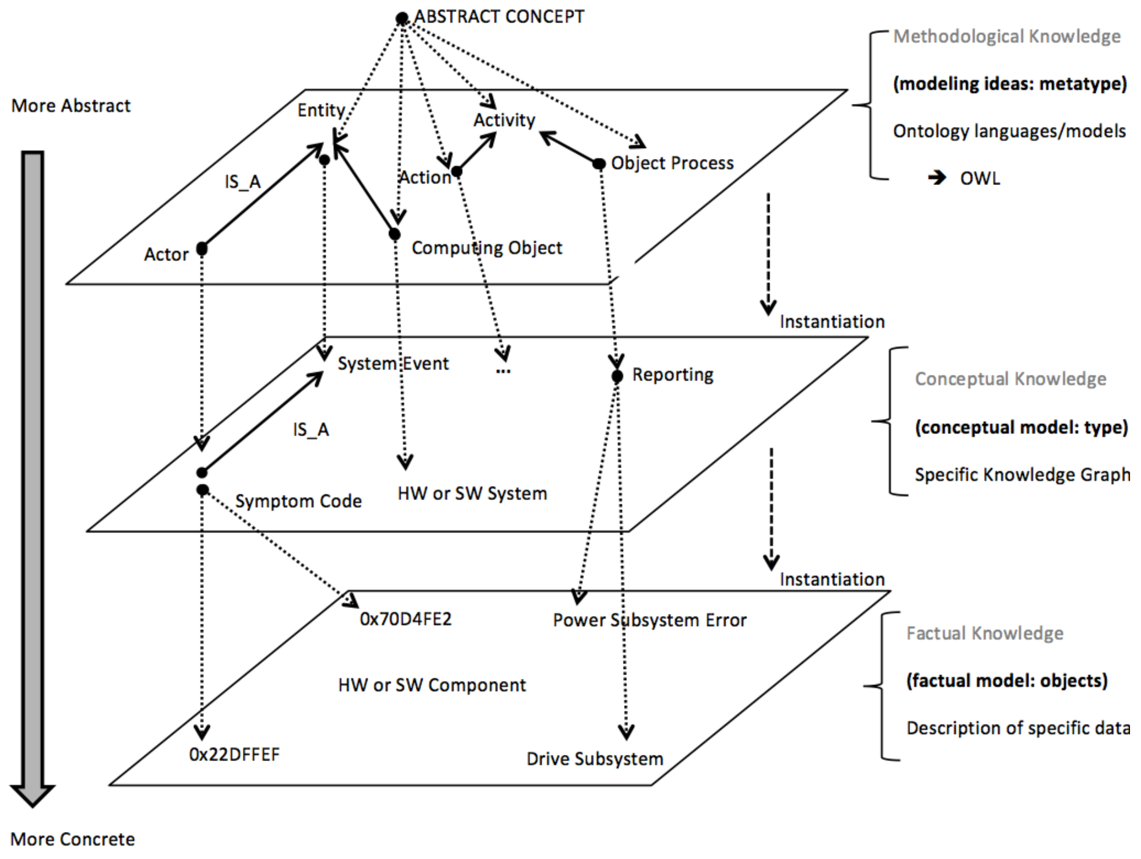


Figure 28 The Three Levels of Knowledge Representation

Table 6 The Three Levels of Knowledge Representation

#	Level of Abstraction	Knowledge Level	Example Data
1	More Abstract	Methodological Knowledge	System Fault

2	Intermediate	Conceptual Knowledge	Power Subsystem Fault
3	Concrete	Factual Knowledge	0x0D4F7E2

At the methodological level (top), Actor *is_a* (n) Entity, Computing Object *is_a* (n) Entity, Action *is_a* (n) activity and Business Process *is_a* (an) Activity.

At the conceptual level (middle), Symptom Code *is_a* System Event, HW or SW System *is_a* (n) Entity, and Reporting *is_a* (n) Object Process.

At the factual level (bottom) the data are now concrete and more specific. Actual symptom hexadecimal symptom codes are used. For example, 0x22DFFEF and 0x70D4FE2 are actual error codes and the HW or SW System is a HW or SW Component and the actions (Reporting) are Power Subsystem Error and Drive Subsystem Error.

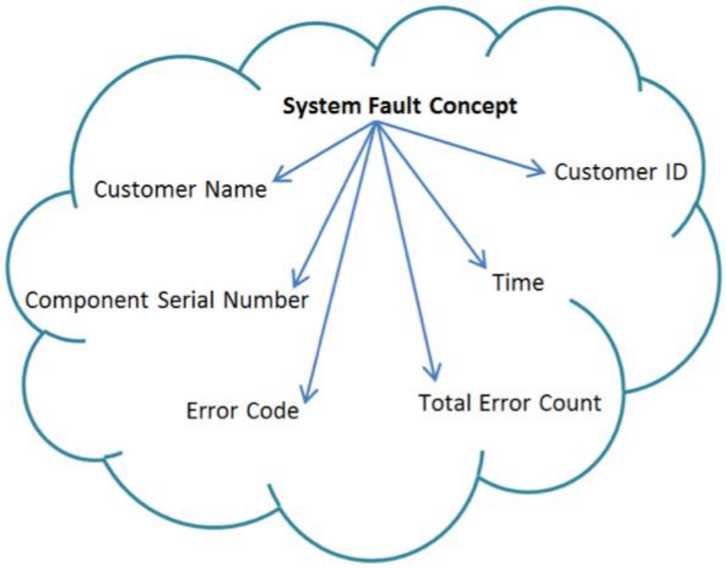


Figure 29 The Abstract System Fault Concept

Figure 29 represents the system fault concept in an abstract way, where the most common elements in the various XML files that are of interest are represented here in the abstract concept containing pertinent data such as the *Customer Name, Customer ID, System Component, Serial Number, Time, the Error Code* and *Total Error Count* logged.

To implement this, mappings play a major role as the glue that tie information together from various sources to enable integration of such information to build a bridge between the more abstract level and the factual concrete level.

As we saw in the address examples from chapter 1, the constraint validation rule that works with the Address tag in one instance document (Example 1), will fail once the structure changes, and the address tag is changed to street (Example 2). The semantics have not changed but the structure has. So the Schematron rule to validate the previous instance document will not work on the new instance document that is using street for to represent address because the rule looking for the address tag will instead find the street tag and fail validation. An entirely new Schematron rule would need to be created to validate this new instance document.

Revisiting the address example seen previously, when one considers that additional instance documents with such differences in semantics will be received, it becomes clear that more Schematron rules would have to be created and maintained to handle these changes. From the foregoing observations it can be said that in this problem domain there is a need for N number of rules for N number of XML dialects.

$$Ns = ? \qquad [15] \qquad (1)$$

Where Ns is the number of Schematron rules needed for complete validation coverage of the address concept.

To determine exactly what the exact number would be for complete coverage of rules to safely handle the dialect variations, one would need to consider the following from the XML 1.1 Specification:

“A Name is a token beginning with a letter or one of a few punctuation characters, and continuing with letters, digits, hyphens, underscores, colons, or full stops, together known as name characters”.[21]

This means that Name:

- can contain letters, numbers and a myriad of other characters.
- cannot start with a number, a punctuation character, nor xml
- cannot contain spaces.[15]

This suggests a grave problem, with a raft of character/digit combinations in a multiplicity of written languages for the address tag name. Indeed, this suggests the value of N to be infinite.

$$Ns = \infty \qquad [15] \qquad (2)$$

As shown in the previous section rules are growing linearly correspondingly with the dialect variations. However, assuming that the address concept doesn't change then the semantics of the address concept remains the same, meaning that if the address concept is expressed in some formalized abstract way, then semantic validation rules based on the abstract concept can be used instead, enabling N to gravitate from infinity, closer to one.[15]

$$Ns = 1 \qquad [15] \qquad (3)$$

In the case of the problem domain targeted by this research, the semantic target does not exactly remain the same, but the semantics are similar enough for the foregoing arguments to be applicable.

Schematron supports abstract rules for declaring and re-using Schematron rules, but they are limited to html tables. Therefore another type of abstract rule is required.

The abstract rules used in this research leverage the flexibility, and rich expressivity of semantic web technologies and knowledge graph structures.

3.6.1 OWL Knowledge representation

The scientific literature is replete with information on OWL's superiority over XML Schemas or DTDs for concept representation. Following the process laid out in Amer and Tao[15], the XPath is transformed as explained in the following section.

The "sum" in test="sum" tests the XPath //numberoferrors.

Substituting the XPath with place-holder variables corresponding to the ontology class and attributes, the Schematron rule can be re-written as an OWL-based abstract rule.

```
<rule context="$xpath">
```

```
<rule context="$error">
```

These changes replace hard-coded XPath expression with place-holder variables. Then, with known mappings between the knowledge graph class and XPath representations of corresponding concepts in a given syntax, it is possible, with some normalization, to create a concrete rule, which can then be used to validate the XML instances of that particular dialect.

3.7 **Framework Validation Workflow**

Step 1: Create and publish a knowledge graph of the concept

Step 2: Create an abstract rule based on the knowledge graph

Step 3: Obtain the following artifacts related to the XML document to be validated:

a. Schema (XSD/DTD) file

b. Mappings between ontology class and XPath representations. Typically a text file containing name-value pairs of the published ontology

c. One or more XML Instance documents to be validated

Step 4: Values in the mapping are normalized and substituted for the values that make up the concrete rule.

Step 5: The newly created concrete rule is used to validate all the instances of the provided dialect by using a Schematron Validator such as the Pace SchematronValidator.

3.8 **Solution Methodology Files**

3.8.1 Abstract Rule

```
<rule context="$systemfaultdetails">
            <assert test="$custidentity">Must be valid</assert>
            <assert test="$location">Must be valid</assert>
            <assert test="$errorcode">Must be valid</assert>
            <assert test="$time">Must be valid</assert>
<assert test="$counts">Must be valid</assert>
</rule>
```

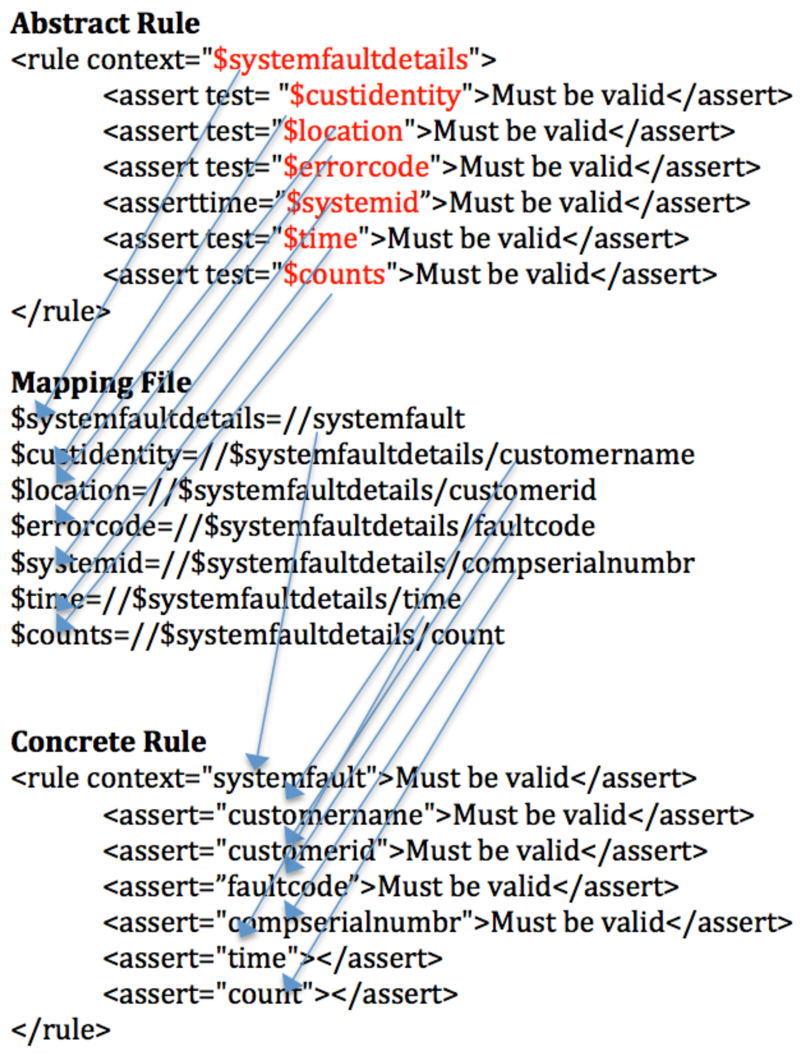
3.8.2 Mapping File

```
$systemfaultdetails=//systemfault
$custidentity=//$systemfaultdetails/customername
$location=//$systemfaultdetails/customerid
$errorcode=//$systemfaultdetails/faultcode
$location=//$systemfaultdetails/compserialnumbr
$time=//$systemfaultdetails/time
$count=//$syetemdefaultdetails/count
```

3.8.3 Concrete Rule

```
<rule context="systemfault">Must be valid</assert>
            <assert="customername">Must be valid</assert>
            <assert="customerid">Must be valid</assert>
            <assert=”faultcode”>Must be valid</assert>
            <assert="compserialnumbr">Must be valid</assert>
            <assert="time">Must be valid</assert>
<assert="count">Must be valid</assert>
</rule>
```

3.9 Mapping Used to Create the Concrete Rule



Abstract Rule

```
<rule context="$systemfaultdetails">
        <assert test= "$custidentity">Must be valid</assert>
        <assert test="$location">Must be valid</assert>
        <assert test="$errorcode">Must be valid</assert>
        <asserttime="$systemid">Must be valid</assert>
        <assert test="$time">Must be valid</assert>
        <assert test="$counts">Must be valid</assert>
</rule>
```

Mapping File

```
$systemfaultdetails=//systemfault
$custidentity=//$systemfaultdetails/customername
$location=//$systemfaultdetails/customerid
$errorcode=//$systemfaultdetails/faultcode
$systemid=//$systemfaultdetails/compserialnumbr
$time=//$systemfaultdetails/time
$counts=//$systemfaultdetails/count
```

Concrete Rule

```
<rule context="systemfault">Must be valid</assert>
        <assert="customername">Must be valid</assert>
        <assert="customerid">Must be valid</assert>
        <assert="faultcode">Must be valid</assert>
        <assert="compserialnumbr">Must be valid</assert>
        <assert="time"></assert>
        <assert="count"></assert>
</rule>
```

Figure 30 Mapping Used to Create the Concrete Rule

With placeholders replacing hard-coded XPath expressions; having knowledge of the mapping between Knowledge Graph classes/attributes and XPath representations corresponding to concepts in a given dialect; plus some normalization, concrete rules can be created to validate the XML instances of the varying dialect.[15]

3.10 **Mapping File Format**

The mapping file is typically a simple text file containing name value pairs separated by an equal sign. The name to the left of equal sign is the value of OWL Class/Property rdf:ID. The value on the hand side is the XPath expression that resolves to the XML Schema entity equivalent to the knowledge graph entity.[15] There are many approaches to data integration. Chapter 2 described studies of existing approaches such as the single, multiple and hybrid ontology, and related work in the healthcare domain like Health Level 7 (HL7), the Clinical Document Architecture (CDA) and the Learning Source Description (LSD). Chapter 2 also outlined the inflexibility of representing knowledge solely with the *is_a* inheritance construct, and explained the need to expand to more flexible, richly expressive custom relations that are possible with knowledge graph constructs.

By describing information in a more abstract conceptual manner, one can simplify the information and reduce the complexity of the data being represented. There are three levels of knowledge abstraction that must be considered. As can be seen in figure 27, knowledge representation flows from the more abstract at the very top to the more concrete on the bottom. The very top level is the methodological knowledge, where modeling of the data is done using OWL. The middle level is a conceptual level where the conceptual model is built, and then below that is the factual knowledge level where specific, concrete description of the data is handled.

As shown in Chapter 2, existing approaches to the data integration challenge fall into three categories the single ontology approach, the multiple ontology approach and the hybrid ontology approach.

Mappings are useful in linking one set of terms with another to create mutually understood vocabulary and explicate the data.

Chapter 2 also explained the process of using the inheritance relationship *is_a* to describe more complex relationships. This technique falls short when more expressiveness is required as it can prove to be difficult, complicated and labor-intensive.

However, through the extension of protégé this research goes beyond ontologies and simple explication, by being able to define multiple custom relations such as the composition *part_of* and other relations defined by subject matter experts (SMEs) and having the flexibility of describing many of the multiple relations specific to the problem domain of this research extends. Thus, this work extends beyond mere ontologies to use knowledge graphs.

Though they have some common elements, this research can be seen in contrast to these other approaches which are restricted to using database interoperability whereas this research couples knowledge graphs with Schematron validation to ensure accuracy of data integration, to enable a more powerful level of recognition, when two pieces of seemingly unrelated data are describing the same thing.

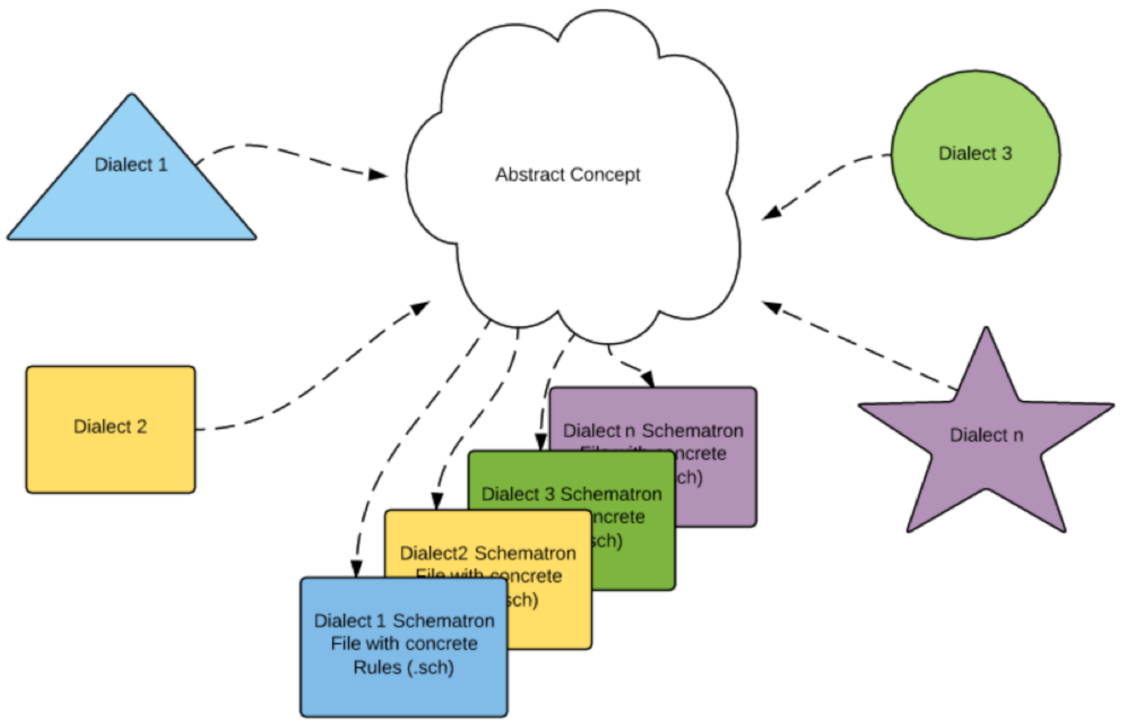


Figure 31 Multiple Dialects Represented in an Abstract Concept

Figure 33 above, represents a high level overview of the solution methodology. The abstract concept in the middle contains an abstract representation of the common elements of all the data from the different dialects. The different dialects are represented as different shapes, triangle, rectangle, circle, and a star etc. These are different dialects coming from different systems, machines and software agents. The common elements are represented in this abstract concept. This drives the auto-population and generation of concrete rules, which are used, for the actual validation of the received XML message files. Benefits of this approach include promoting a more rapid response to actionable events, increased flexibility, reducing complexity, lowering the total cost of management, and avoiding having to maintain multiple Schematron files.

This research framework solution classifies Schematron rules by dialect. IT staff, domain experts and subject matter experts (SMEs) use the same pattern, but write different Schematron rules for different products.

Figure 34 below depicts the multiple dialect-handling framework with the same abstract-to-concrete rules pattern for reduced complexity, increased flexibility, lower cost of management, enhanced semantic explication, and promoting more rapid response to actionable errors.

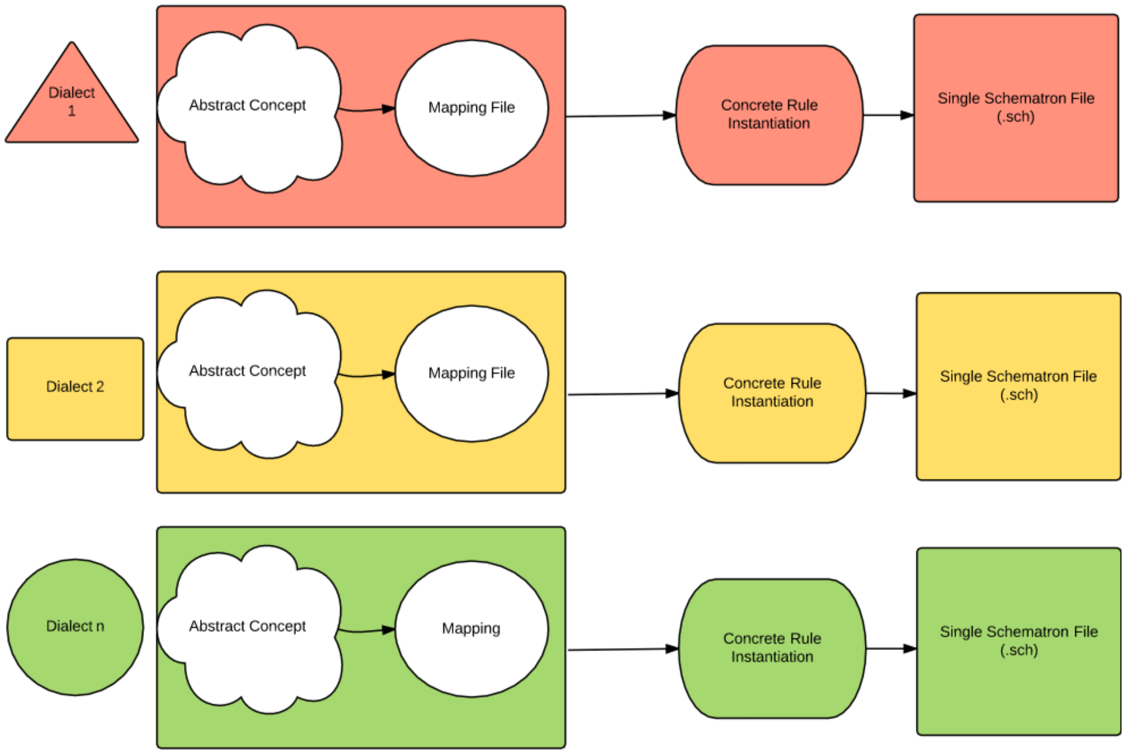
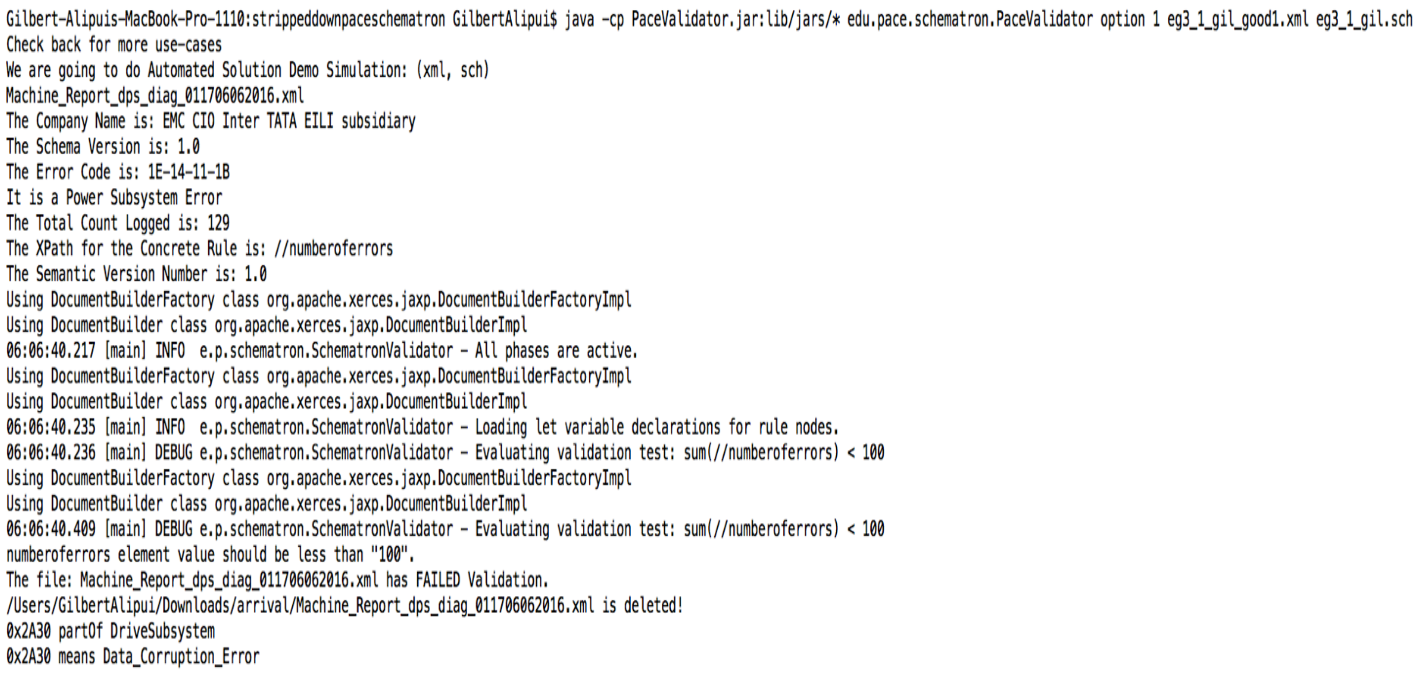

Dialect 1
Abstract Concept
Mapping File
Concrete Rule Instantiation
Single Schematron File (.sch)
Dialect 2
Abstract Concept
Mapping File
Concrete Rule Instantiation
Single Schematron File (.sch)
Dialect n
Abstract Concept
Mapping
Concrete Rule Instantiation
Single Schematron File (.sch)

Figure 32 Multiple Dialect-Handling Framework with Abstract to Concrete Rules

Figure 35 is a screenshot of running the PaceValidator on the command line with required arguments. The first parameter, 1 is for the standard run option, the second parameter is the XML filename, and the third option is the Schematron file that is used to perform the validation. An error code is identified, and the validation tests to see if a predetermined safety threshold has been breached. (100). In this case, the threshold is 101, which is more than the allowable limit. The safety threshold has been exceeded, so the file has failed validation. An error to that effect is emitted; PaceJena is invoked to obtain additional relations and semantics from an associated OWL file for this error and other associated errors. These are extracted, packaged together, and escalated to tech support.

This solution enables one to avoid having to maintain multiple Schematron files. There is reduced complexity because multiple diagnostic message files are processed with the afore-mentioned framework, to validate the array of arriving XML diagnostics.



```
Gilbert-Alipuis-MacBook-Pro-1110:strippeddownpaceschematron GilbertAlipui$ java -cp PaceValidator.jar:lib/jars/* edu.pace.schematron.PaceValidator option 1 eg3_1_gil_good1.xml eg3_1_gil.sch
Check back for more use-cases
We are going to do Automated Solution Demo Simulation: (xml, sch)
Machine_Report_dps_diag_011706062016.xml
The Company Name is: EMC CIO Inter TATA EILI subsidiary
The Schema Version is: 1.0
The Error Code is: 1E-14-11-1B
It is a Power Subsystem Error
The Total Count Logged is: 129
The XPath for the Concrete Rule is: //numberoferrors
The Semantic Version Number is: 1.0
Using DocumentBuilderFactory class org.apache.xerces.jaxp.DocumentBuilderFactoryImpl
Using DocumentBuilder class org.apache.xerces.jaxp.DocumentBuilderImpl
06:06:40.217 [main] INFO  e.p.schematron.SchematronValidator - All phases are active.
Using DocumentBuilderFactory class org.apache.xerces.jaxp.DocumentBuilderFactoryImpl
Using DocumentBuilder class org.apache.xerces.jaxp.DocumentBuilderImpl
06:06:40.235 [main] INFO  e.p.schematron.SchematronValidator - Loading let variable declarations for rule nodes.
06:06:40.236 [main] DEBUG e.p.schematron.SchematronValidator - Evaluating validation test: sum(//numberoferrors) < 100
Using DocumentBuilderFactory class org.apache.xerces.jaxp.DocumentBuilderFactoryImpl
Using DocumentBuilder class org.apache.xerces.jaxp.DocumentBuilderImpl
06:06:40.409 [main] DEBUG e.p.schematron.SchematronValidator - Evaluating validation test: sum(//numberoferrors) < 100
numberoferrors element value should be less than "100".
The file: Machine_Report_dps_diag_011706062016.xml has FAILED Validation.
/Users/GilbertAlipui/Downloads/arrival/Machine_Report_dps_diag_011706062016.xml is deleted!
0x2A30 partOf DriveSubsystem
0x2A30 means Data_Corruption_Error
```

Figure 33 Running Schematron PaceValidator on the Command Line

This research avoids translation, and seeks to directly check the semantics of the XML file that is being processed.

3.11 **Solution Benefits**

We propose a framework solution that:

1. Is adaptive, and capable of classifying different kinds of system diagnostic information for different products.
2. Uses semantic constraints to define patterns for classifying the various diagnostic information.
3. Uses a knowledge graph to reduce the complexity of maintaining multiple versions of similar patterns.

The proposed solution is a framework that facilitates the integration of data from disparate sources containing similar information, in different dialects, automatically identifies system problems based on product diagnostic messages, uses semantic constraints to define patterns for classifying the various diagnostic information, and introduces a knowledge graph to define extra concepts and their relationships, and explain the meaning of symptom codes. Schematron is used to specify business rules on abstract concepts so that such abstract rules are automatically applied to diagnostic messages based on concrete incarnations of these concepts, achieving efficient maintenance of diagnostic rules.[15]

3.12 **Conclusion**

This chapter provided a detailed specification of the XML Dialect proliferation problem, provided a comprehensive overview of the proposed solution strategy, outlined the use of knowledge graphs to specify patterns and explained using Schematron to specify diagnostic constraints patterns. We saw the solution workflow, the abstract concept, the framework validation workflow, and mapping file. Chapter 4 will provide more detail. The rational for extending Schematron to specify business rules on ontology concepts, and automatically apply abstract rules to specific tag names used in the diagnostic messages was explained. This chapter also explained the use of the same abstract-to-concrete rules pattern for reduced complexity, flexibility, lower cost of management, enhanced semantic explication, and also the promotion of more rapid response to actionable errors.

As far as I am aware, this research solution to the SERCH problem is unique in coupling Schematron validations with OWL knowledge graphs, to enable data integration, automatically identifying system problems based on product diagnostic messages.

Chapter 4

Implementation Highlights

4.1 Introduction

Chapter 3 described the research solution methodology in detail and covered PaceSchematron, PaceProtégé, and PaceJena. This chapter provides additional details, which will describe the Eclipse IDE for the development of the Java program, used to couple Schematron with the knowledge graph and the process of post-validation failure symptom code explication.

As we saw in Chapter 3, OWL's lack of support for arithmetic and string manipulation functions[31] provides an opportunity for this research to target this gap in this area of data integration. This research overcomes this limitation by extending Schematron to handle OWL's lack of support for arithmetic and string manipulation functions.

Leveraging the extensions mentioned in the previous chapter, this research couples Schematron validation with OWL to give it more expressive power and the capability of handling data from disparate sources and recognizing relations between linked data and providing explication of the data. Rather than manually processing each distinct XML dialect file, the program swaps in concrete rules at run-time. This is accomplished through the use of the $-prefixed variables where the abstract rules are replaced by concrete rules. The abstract rule variables then contain the concrete rules after the run-time swap is done, automatically producing the concrete rule-instantiated Schematron instance file.

The previous chapter described the process by which an ontology-based knowledge graph is used to map the conceptual level Schematron rules to concrete Schematron rules. The next sections provide more detail.

4.2 Deriving the Concrete Rules from the Abstract Rules

Since this research solution can be used for many companies and for many different situations, this section provides the algorithm and the description of the implementation of the algorithm.

4.3 Abstract to Concrete Algorithm

4.3.1 Dialect Recognition

Dialect recognition is an important part of the abstract-to-concrete algorithm generation. There are multiple options for dialect identification.

- embedded xsd containing the version number
- Regex pattern identification
- A combination of both

If there are multiple dialects that fit the same pattern, regular expressions may prove to be unsuitable, and match duplicate patterns. In such a case more work would be needed to distinguish these patterns from each other, or it may be more expedient to switch to the embedded specifying the dialect version in the xsd file referenced in the xml file.

4.3.2 Abstract to Concrete Algorithm (AbsToConc)

To explain the process of deriving concrete rules from abstract concepts, we propose the Abstract to Concrete (AbsToConc) algorithm. The pseudocode of the AbsToConc algorithm is shown in figure 34 below.

```
Input files
For ∀all sessions do
   Get XML, sch abstract, mapping file, owl, xsd files
     Read XML line-by-line
        Apply Regex to identify dialect by pattern for Xpath & symptom
        code
        Read Mapping file (line-by-line)
        Find XPath in mapping file
        Extract Tokens
        Read Schematron.sch file (line-by-line)
        Locate Target XPath, replace $-prefixed variable with actual values
        (tokens) (results in concrete rule . sch. file)
        Use Concrete schematron file to perform validation
        if Validation fails then
           pass Symptom code to PaceJena for extraction of semantics and
           relations
          else Stop
        end if
end
```

Figure 34 Pseudocode for the Abstract to Concrete Algorithm

This research has application utility for other domains like cloud, cyber-security, law enforcement, counter-terrorism (Figure 12, Figure 47) or the FOAF project that "offers tools to relate people through a model that contains typical social attributes such as a name, email address, interests, and the like"[10], and healthcare. When faced with similar pattern definition challenges, information technology staff from other domains, from medical companies for instance, could meet, collaborate and make decisions, then agree on abstract rules for their respective domains. They would state the specific type of XSD they are using, the other companies doing likewise, and they arrive at a common understanding and agreement. Based on agreed-upon concepts and knowledge graphs, a given XPath would be referenced, and matched to a specific pattern and mapping. The system would then read a specific diagnostic message looking for the respective patterns in an element within the XML file to identify the syntax used. The abstract concept, being mapped this way would then be used to implement the specific concrete rules.

The workflow follows that of this research, and is illustrated in the sequence below. For each new dialect, the analyst analyses the XML diagnostic file and provides the knowledge graph, mapping file, and the abstract rule file.

4.3.3 An Example Research Ontology

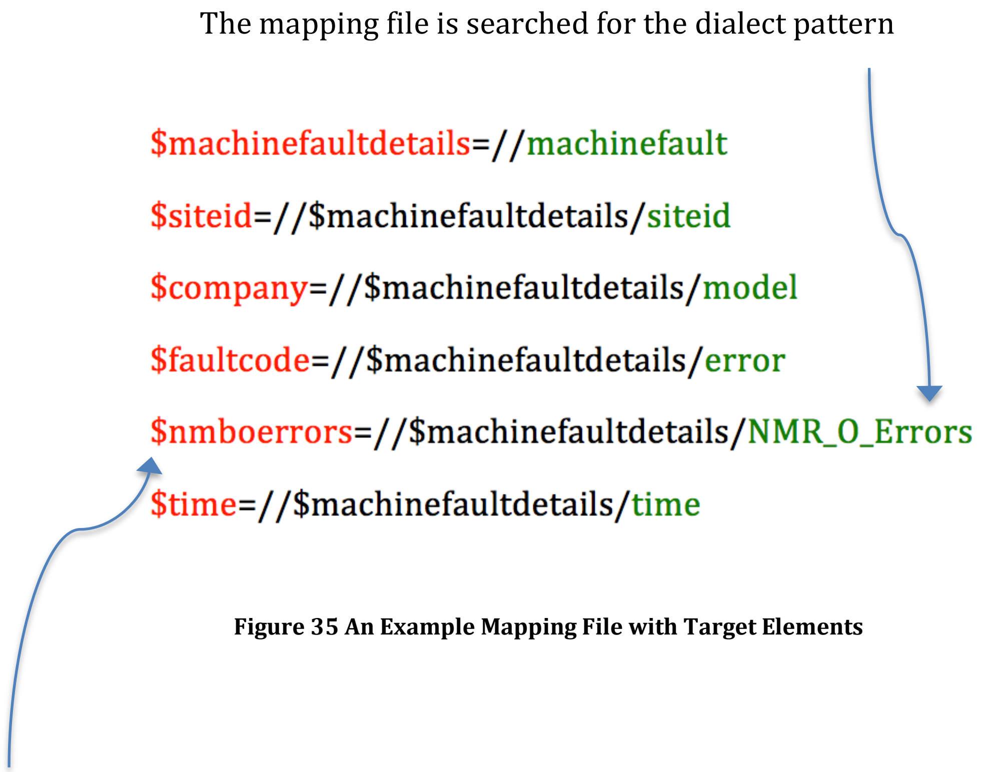
```
<owl:Class rdf:ID="MachineFaultDetails">
<rdfs:label> MachineFaultDetails</rdf:label>
<rdfs:comment> MachineFaultDetails</rdfs:comment>
</owl:class>


<owl:DatatypePropert rdf:ID="siteid">
<rdf:domain rdf:resource="#MachineFaultDetails"/>
```

```
<rdfs:range rdf:resource="&xsd;string" />
</owl:DatatypeProperty>

<owl:DatatypePropert rdf:ID="error">
<rdf:domain rdf:resource="#MachineFaultDetails"/>
<rdfs:range rdf:resource="&xsd;string" />
</owl:DatatypeProperty>

<owl:DatatypePropert rdf:ID="numberoferrors">
<rdf:domain rdf:resource="#MachineFaultDetails"/>
<rdfs:range rdf:resource="&xsd;string" />
</owl:DatatypeProperty>
```

With the abstract rule mapping file in place, the following steps are taken to perform the validation:

For a given knowledge graph above, the Schematron command line or IDE-based program takes the name of XML file, the Schematron file, the mapping file, and the OWL file as input parameters. After the XML, sch, txt and owl filenames are passed to Schematron, (through the IDE or on the command line), the XML file is parsed and a regular expression (regex) is applied to identify the dialect. For example, a given vocabulary is identified by a pattern similar to XXX_X_XXX. The error or symptom code is also acquired at this time and held in reserve for use in case semantic validation requirements are not met. Figure 34 below illustrates how elements of interest are identified. As previously described, in this case the XXX_X_XXX pattern from the mapping file is used to identify the dialect. Specifically, a pattern like NBR_O_Errors or NMR_O_Errors, which is a tag name or could also be an attribute in the XML file is matched by pattern.

Once it matches that tag name in the XML file, it searches for it in the mapping file. A match for the tag name found in the mapping file points to a corresponding XPath in the Schematron file. In this case, NBR_O_Errors in the mapping file is equal to //nmboferrors and is represented with a $ sign, $numberoferrors to signify that it is a variable of the abstract rule.



```
$machinefaultdetails=//machinefault
$siteid=//$machinefaultdetails/siteid
$company=//$machinefaultdetails/model
$faultcode=//$machinefaultdetails/error
$nmboerrors=//$machinefaultdetails/NMR_O_Errors
$time=//$machinefaultdetails/time
```

Figure 35 An Example Mapping File with Target Elements

The line, $nmboerror=//$machinefaultdetails/NBR_O_Error, matching the dialect identification pattern is further processed to acquire all the necessary tokens. Once the tokens are available, the place-holder variables in the abstract rule are replaced with the appropriate token. For example, in this case $nmboerror is replaced with NBR_O_Errors at run-time to produce the concrete rule as shown in Figure 36 below.

During the validation session, the Schematron program substitutes the XPath found in the mapping file for $numberoferrors ($numberoferrors is replaced with the actual XPath i.e. NBR_O_Errors). Once the substitution is done, then the Schematron file is now a concrete rule that can be used to perform the validation.

4.4 Solution Methodology Files

4.4.1 The XML File

```
<?xml version="1.0" encoding="UTF-8">
<library xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="support.xsd">
<machinefault="System Health Report">
    <siteinfo id="1">
    <siteid>100099</siteid>
    <company>ACME</company>
    <model>Viper IO</model>
    <time>2016-03-23</time>
    <error>07D4FE2</error>
    < NBR_O_Error >105</ NBR_O_Error >
  </siteinfo>
</dpsrule>
```

4.4.2 Mapping File

```
$machinefaultdetails=//machinefault

$siteid=//$machinefaultdetails/siteid

$company=//$machinefaultdetails/model

$faultcode=//$machinefaultdetails/error

$nmboerrors=//$machinefaultdetails/NBR_O_Errors

$time=//$machinefaultdetails/time
```

4.4.3 Abstract Rule

```
<rule context="$machinefaultdetails">

      <assert test="$siteid">siteid may not be empty.</assert>

      <assert test="$company">Company may not be empty.</assert>
```

```
        <assert test="$error">Error may not be empty.</assert>

        <assert test="sum($nmboerror) &lt; 100">numberoferrors element
            value should less than "100".
        </assert>

        <assert test="$time">Time may not be empty.</assert>

</rule>
```

4.4.4 *Concrete Rule*

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron">
  <pattern name="Sum Less Than 100">
    <rule context test="machinefault">Title may not be empty. </assert>
      <assert test="siteid">Siteid may not be empty.</assert>
      <assert test="company">Company may not be empty.</assert>
      <assert test=”error”>Error may not be empty.</assert>
      <assert test="sum(//NMB_O_Error) &lt; 100">numberoferrors element
             value should less than "100".
      </assert>
      <assert test="time">Must be valid</assert>
    </rule>
</schema>
```

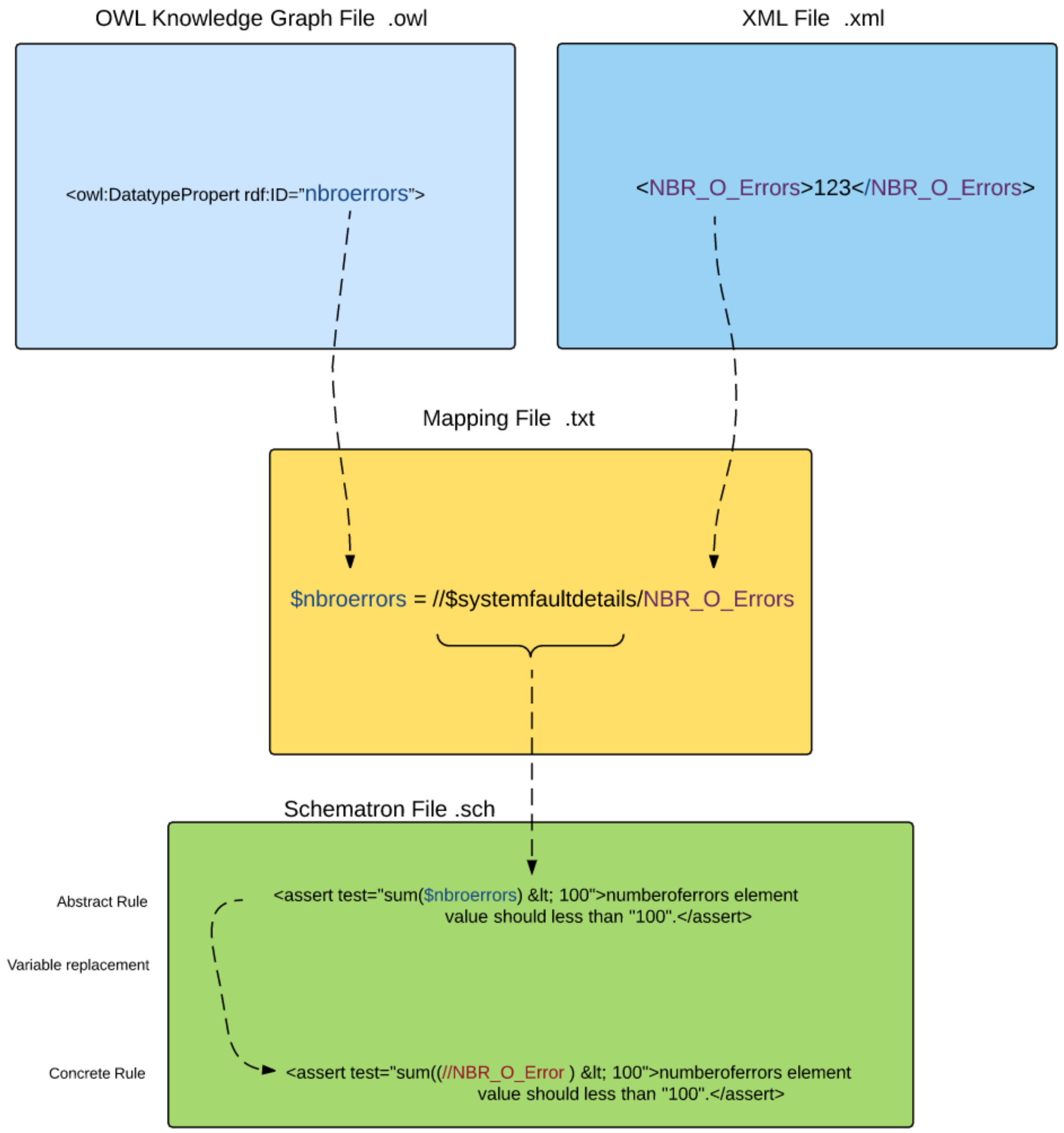
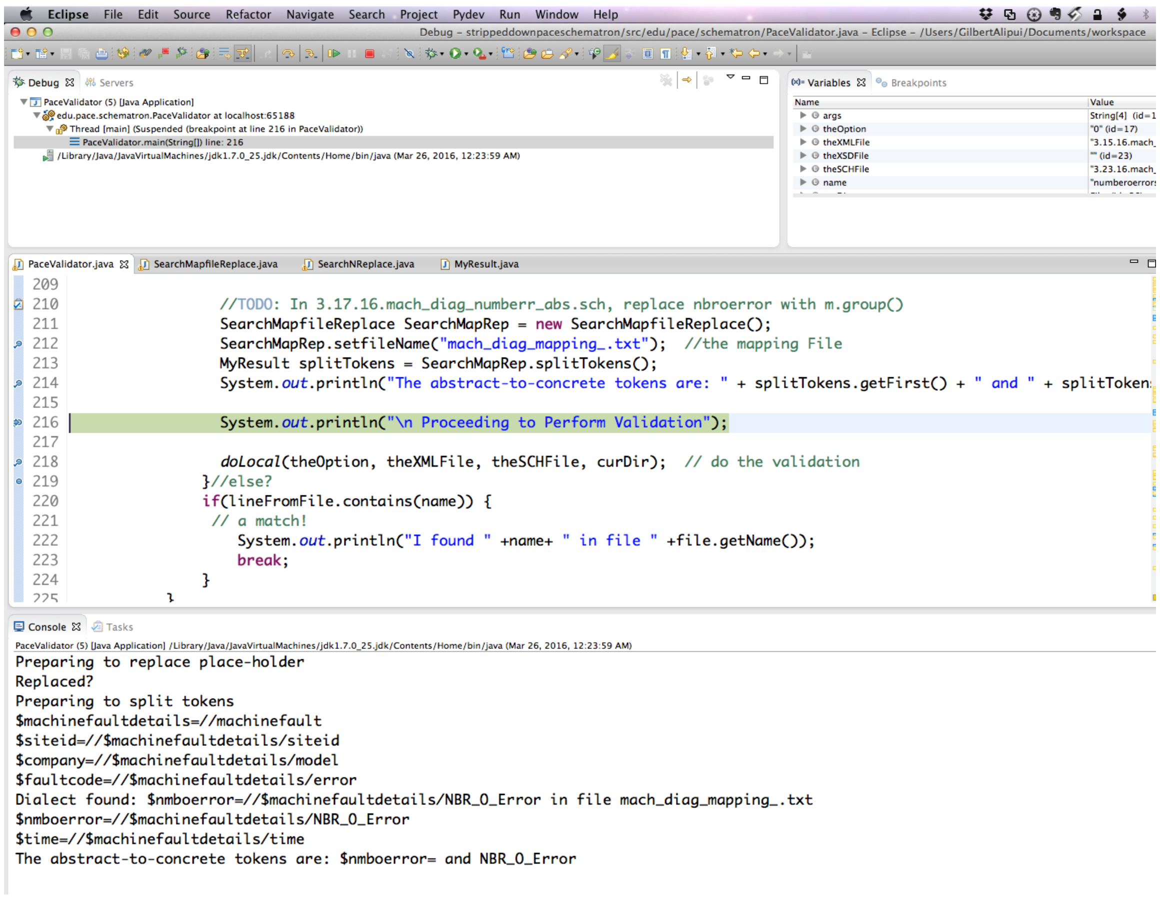

OWL Knowledge Graph File .owl
<owl:DatatypePropert rdf:ID="nbroerrors">
XML File .xml
<NBR_O_Errors>123</NBR_O_Errors>
Mapping File .txt
$nbroerrors = //$systemfaultdetails/NBR_O_Errors
Schematron File .sch
Abstract Rule
<assert test="sum($nbroerrors) &lt; 100">numberoferrors element value should less than "100".</assert>
Variable replacement
Concrete Rule
<assert test="sum((//NBR_O_Error) &lt; 100">numberoferrors element value should less than "100".</assert>

Figure 36 Transition from Abstract Rule to Concrete Rule

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron">
  <pattern name="Sum Less Than 100">
    <rule context="/">
      <assert test="sum(//NBR_O_Error) &lt; 100">numberoferrors element
              value should be less than "100".
      </assert>
    </rule>
  </pattern>
</schema>
```

Figure 37 The Generated Concrete Rule



Debug – strippeddownpaceschematron/src/edu/pace/schematron/PaceValidator.java – Eclipse – /Users/GilbertAlipui/Documents/workspace

```
PaceValidator (5) [Java Application]
  edu.pace.schematron.PaceValidator at localhost:65188
    Thread [main] (Suspended (breakpoint at line 216 in PaceValidator))
      PaceValidator.main(String[]) line: 216
  /Library/Java/JavaVirtualMachines/jdk1.7.0_25.jdk/Contents/Home/bin/java (Mar 26, 2016, 12:23:59 AM)
```

Name	Value
args	String[4] (id=1
theOption	"0" (id=17)
theXMLFile	"3.15.16.mach_
theXSDFile	"" (id=23)
theSCHFile	"3.23.16.mach_
name	"numberoerror:

```java
                    //TODO: In 3.17.16.mach_diag_numberr_abs.sch, replace nbroerror with m.group()
                    SearchMapfileReplace SearchMapRep = new SearchMapfileReplace();
                    SearchMapRep.setfileName("mach_diag_mapping_.txt");  //the mapping File
                    MyResult splitTokens = SearchMapRep.splitTokens();
                    System.out.println("The abstract-to-concrete tokens are: " + splitTokens.getFirst() + " and " + splitToken

                    System.out.println("\n Proceeding to Perform Validation");

                    doLocal(theOption, theXMLFile, theSCHFile, curDir);  // do the validation
                  }//else?
                  if(lineFromFile.contains(name)) {
                   // a match!
                      System.out.println("I found " +name+ " in file " +file.getName());
                      break;
                  }
              }
```

PaceValidator (5) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.7.0_25.jdk/Contents/Home/bin/java (Mar 26, 2016, 12:23:59 AM)

```
Preparing to replace place-holder
Replaced?
Preparing to split tokens
$machinefaultdetails=//machinefault
$siteid=//$machinefaultdetails/siteid
$company=//$machinefaultdetails/model
$faultcode=//$machinefaultdetails/error
Dialect found: $nmboerror=//$machinefaultdetails/NBR_O_Error in file mach_diag_mapping_.txt
$nmboerror=//$machinefaultdetails/NBR_O_Error
$time=//$machinefaultdetails/time
The abstract-to-concrete tokens are: $nmboerror= and NBR_O_Error
```

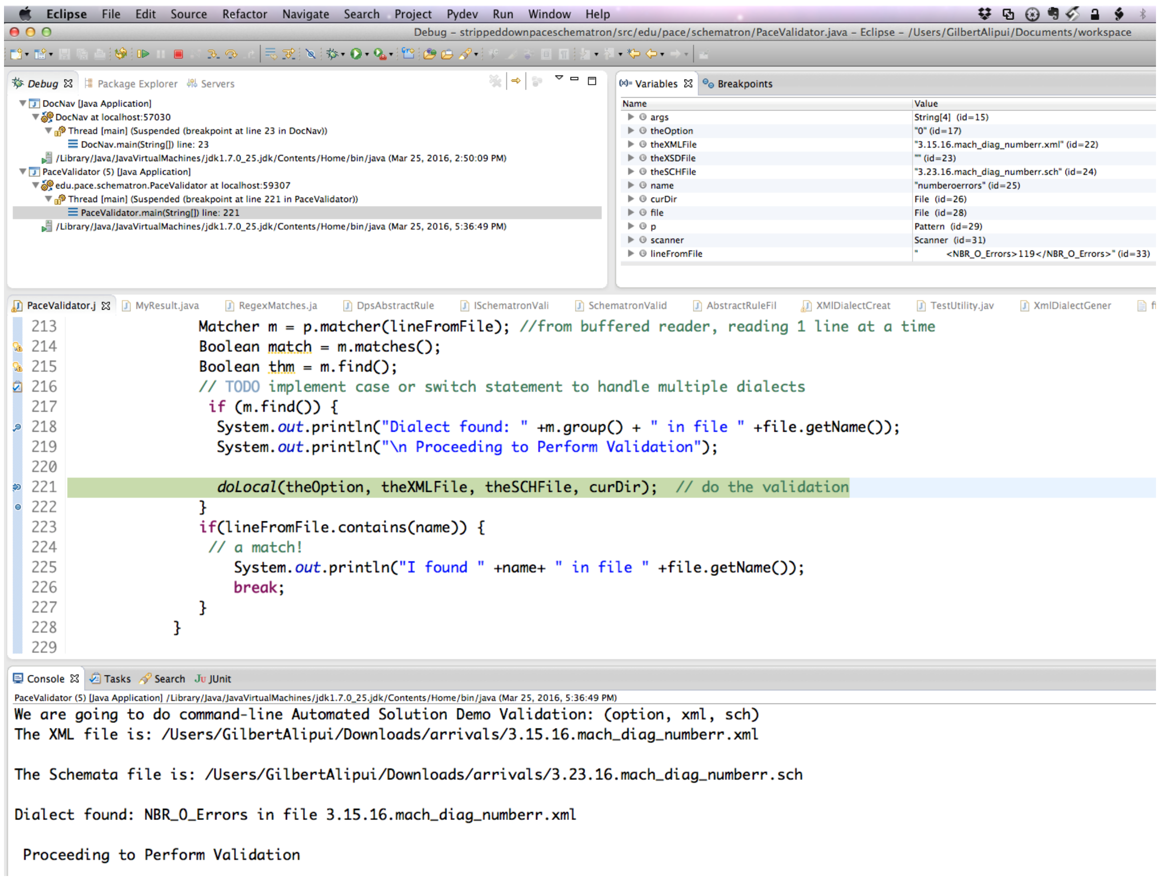
Eclipse File Edit Source Refactor Navigate Search Project Pydev Run Window Help
Debug
Servers
Variables
Breakpoints
PaceValidator.java
SearchMapfileReplace.java
SearchNReplace.java
MyResult.java
Console
Tasks

Figure 38 Retrieving the Abstract-to-Concrete Variables from the Mapping File

With the concrete rule in place, the validation is performed. If the XML file fails the validation, PaceJena is invoked, the symptom or error code (which is already on hand) is passed to PaceJena to retrieve its semantic explication.

4.5 Implementation Examples

This section displays screenshots of excerpts of important solution implementation code examples from the Eclipse IDE like dialect identification, retrieval of relations and explications, and the invocation of PaceJena.



Debug – strippeddownpaceschematron/src/edu/pace/schematron/PaceValidator.java – Eclipse – /Users/GilbertAlipui/Documents/workspace

```
DocNav [Java Application]
  DocNav at localhost:57030
    Thread [main] (Suspended (breakpoint at line 23 in DocNav))
      DocNav.main(String[]) line: 23
  /Library/Java/JavaVirtualMachines/jdk1.7.0_25.jdk/Contents/Home/bin/java (Mar 25, 2016, 2:50:09 PM)
PaceValidator (5) [Java Application]
  edu.pace.schematron.PaceValidator at localhost:59307
    Thread [main] (Suspended (breakpoint at line 221 in PaceValidator))
      PaceValidator.main(String[]) line: 221
  /Library/Java/JavaVirtualMachines/jdk1.7.0_25.jdk/Contents/Home/bin/java (Mar 25, 2016, 5:36:49 PM)
```

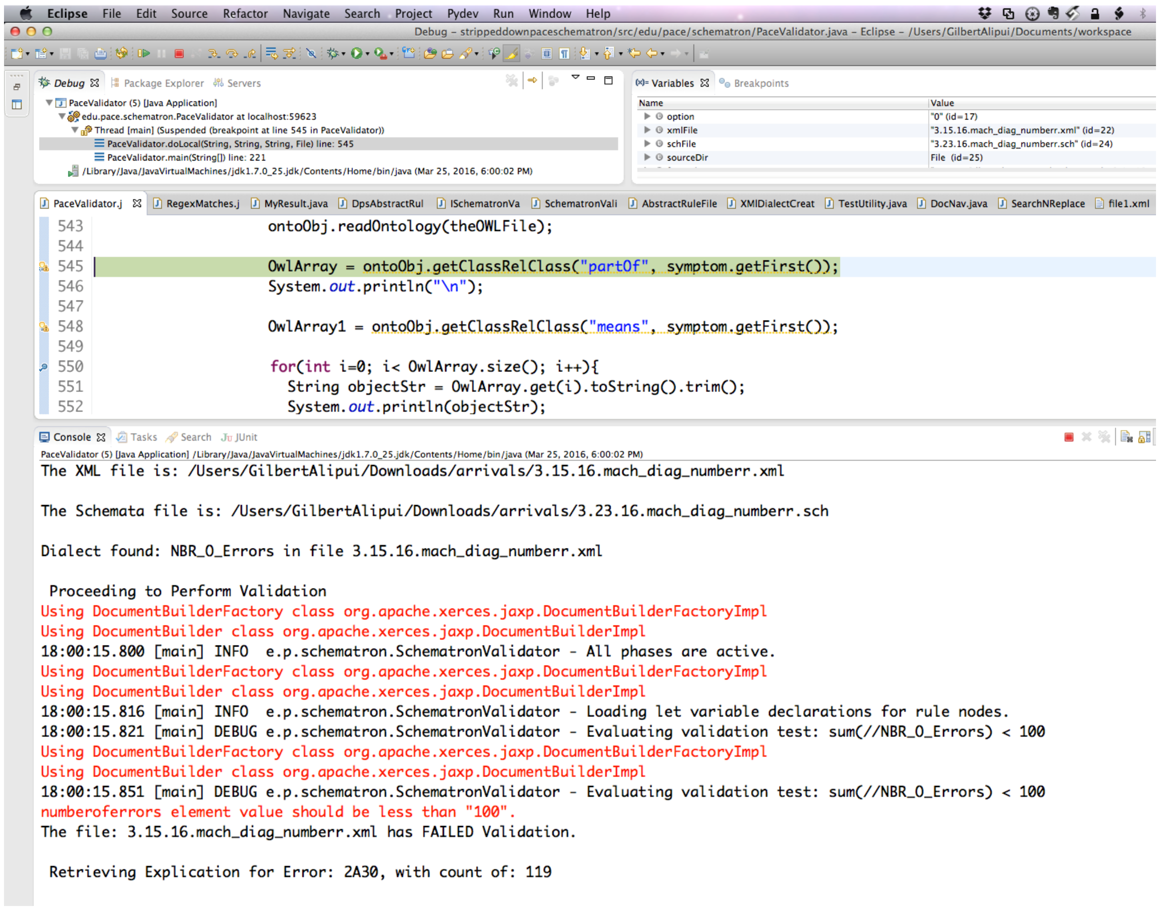
Name	Value
args	String[4] (id=15)
theOption	"0" (id=17)
theXMLFile	"3.15.16.mach_diag_numberr.xml" (id=22)
theXSDFile	"" (id=23)
theSCHFile	"3.23.16.mach_diag_numberr.sch" (id=24)
name	"numberoerrors" (id=25)
curDir	File (id=26)
file	File (id=28)
p	Pattern (id=29)
scanner	Scanner (id=31)
lineFromFile	" <NBR_O_Errors>119</NBR_O_Errors>" (id=33)

```java
            Matcher m = p.matcher(lineFromFile); //from buffered reader, reading 1 line at a time
            Boolean match = m.matches();
            Boolean thm = m.find();
            // TODO implement case or switch statement to handle multiple dialects
             if (m.find()) {
              System.out.println("Dialect found: " +m.group() + " in file " +file.getName());
              System.out.println("\n Proceeding to Perform Validation");

              doLocal(theOption, theXMLFile, theSCHFile, curDir);  // do the validation
            }
            if(lineFromFile.contains(name)) {
             // a match!
                System.out.println("I found " +name+ " in file " +file.getName());
                break;
            }
         }
```

PaceValidator (5) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.7.0_25.jdk/Contents/Home/bin/java (Mar 25, 2016, 5:36:49 PM)

```
We are going to do command-line Automated Solution Demo Validation: (option, xml, sch)
The XML file is: /Users/GilbertAlipui/Downloads/arrivals/3.15.16.mach_diag_numberr.xml

The Schemata file is: /Users/GilbertAlipui/Downloads/arrivals/3.23.16.mach_diag_numberr.sch

Dialect found: NBR_O_Errors in file 3.15.16.mach_diag_numberr.xml

 Proceeding to Perform Validation
```

Figure 39 Dialect Identified Prior to Validation



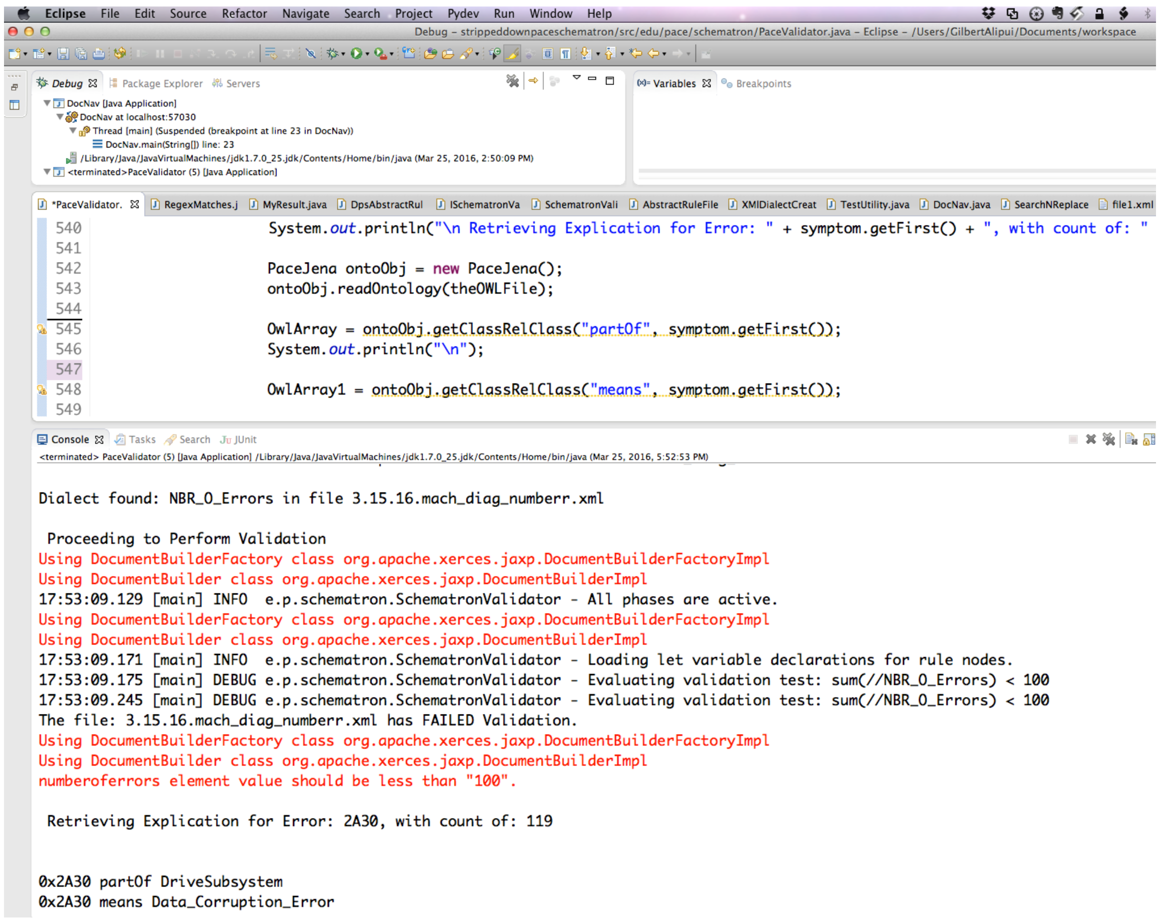
```
PaceValidator (5) [Java Application]
  edu.pace.schematron.PaceValidator at localhost:59623
    Thread [main] (Suspended (breakpoint at line 545 in PaceValidator))
      PaceValidator.doLocal(String, String, String, File) line: 545
      PaceValidator.main(String[]) line: 221
  /Library/Java/JavaVirtualMachines/jdk1.7.0_25.jdk/Contents/Home/bin/java (Mar 25, 2016, 6:00:02 PM)
```

Name	Value
option	"0" (id=17)
xmlFile	"3.15.16.mach_diag_numberr.xml" (id=22)
schFile	"3.23.16.mach_diag_numberr.sch" (id=24)
sourceDir	File (id=25)

```
543                     ontoObj.readOntology(theOWLFile);
544
545                     OwlArray = ontoObj.getClassRelClass("partOf", symptom.getFirst());
546                     System.out.println("\n");
547
548                     OwlArray1 = ontoObj.getClassRelClass("means", symptom.getFirst());
549
550                     for(int i=0; i< OwlArray.size(); i++){
551                       String objectStr = OwlArray.get(i).toString().trim();
552                       System.out.println(objectStr);
```

```
PaceValidator (5) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.7.0_25.jdk/Contents/Home/bin/java (Mar 25, 2016, 6:00:02 PM)
The XML file is: /Users/GilbertAlipui/Downloads/arrivals/3.15.16.mach_diag_numberr.xml

The Schemata file is: /Users/GilbertAlipui/Downloads/arrivals/3.23.16.mach_diag_numberr.sch

Dialect found: NBR_O_Errors in file 3.15.16.mach_diag_numberr.xml

 Proceeding to Perform Validation
Using DocumentBuilderFactory class org.apache.xerces.jaxp.DocumentBuilderFactoryImpl
Using DocumentBuilder class org.apache.xerces.jaxp.DocumentBuilderImpl
18:00:15.800 [main] INFO  e.p.schematron.SchematronValidator - All phases are active.
Using DocumentBuilderFactory class org.apache.xerces.jaxp.DocumentBuilderFactoryImpl
Using DocumentBuilder class org.apache.xerces.jaxp.DocumentBuilderImpl
18:00:15.816 [main] INFO  e.p.schematron.SchematronValidator - Loading let variable declarations for rule nodes.
18:00:15.821 [main] DEBUG e.p.schematron.SchematronValidator - Evaluating validation test: sum(//NBR_O_Errors) < 100
Using DocumentBuilderFactory class org.apache.xerces.jaxp.DocumentBuilderFactoryImpl
Using DocumentBuilder class org.apache.xerces.jaxp.DocumentBuilderImpl
18:00:15.851 [main] DEBUG e.p.schematron.SchematronValidator - Evaluating validation test: sum(//NBR_O_Errors) < 100
numberoferrors element value should be less than "100".
The file: 3.15.16.mach_diag_numberr.xml has FAILED Validation.

 Retrieving Explication for Error: 2A30, with count of: 119
```

Figure 40 Retrieving Semantics and Explications Upon Validation Failure



```
DocNav [Java Application]
  DocNav at localhost:57030
    Thread [main] (Suspended (breakpoint at line 23 in DocNav))
      DocNav.main(String[]) line: 23
  /Library/Java/JavaVirtualMachines/jdk1.7.0_25.jdk/Contents/Home/bin/java (Mar 25, 2016, 2:50:09 PM)
<terminated>PaceValidator (5) [Java Application]
```

```
                    System.out.println("\n Retrieving Explication for Error: " + symptom.getFirst() + ", with count of: "

                    PaceJena ontoObj = new PaceJena();
                    ontoObj.readOntology(theOWLFile);

                    OwlArray = ontoObj.getClassRelClass("partOf", symptom.getFirst());
                    System.out.println("\n");

                    OwlArray1 = ontoObj.getClassRelClass("means", symptom.getFirst());
```

```
<terminated> PaceValidator (5) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.7.0_25.jdk/Contents/Home/bin/java (Mar 25, 2016, 5:52:53 PM)

Dialect found: NBR_O_Errors in file 3.15.16.mach_diag_numberr.xml

 Proceeding to Perform Validation
Using DocumentBuilderFactory class org.apache.xerces.jaxp.DocumentBuilderFactoryImpl
Using DocumentBuilder class org.apache.xerces.jaxp.DocumentBuilderImpl
17:53:09.129 [main] INFO  e.p.schematron.SchematronValidator - All phases are active.
Using DocumentBuilderFactory class org.apache.xerces.jaxp.DocumentBuilderFactoryImpl
Using DocumentBuilder class org.apache.xerces.jaxp.DocumentBuilderImpl
17:53:09.171 [main] INFO  e.p.schematron.SchematronValidator - Loading let variable declarations for rule nodes.
17:53:09.175 [main] DEBUG e.p.schematron.SchematronValidator - Evaluating validation test: sum(//NBR_O_Errors) < 100
17:53:09.245 [main] DEBUG e.p.schematron.SchematronValidator - Evaluating validation test: sum(//NBR_O_Errors) < 100
The file: 3.15.16.mach_diag_numberr.xml has FAILED Validation.
Using DocumentBuilderFactory class org.apache.xerces.jaxp.DocumentBuilderFactoryImpl
Using DocumentBuilder class org.apache.xerces.jaxp.DocumentBuilderImpl
numberoferrors element value should be less than "100".

 Retrieving Explication for Error: 2A30, with count of: 119


0x2A30 partOf DriveSubsystem
0x2A30 means Data_Corruption_Error
```

Figure 41 Invoking PaceJena for Semantics and Explication

4.6 Important Research Considerations

As with all engineering work, tradeoffs are an important consideration. In this research, scalability of the solution methodology is a major consideration. In weighing the advantages and disadvantages of using regular expressions for dialect identification the risk of dialect misidentification was undeniable, and recognized as being high, when processing a large volume of in-bound files because of the likelihood of multiple patterns being similar in structure and falsely matching a specific regex pattern.

4.7 The Quest for a More Scalable Approach

Since IT staff and domain experts are typically involved in integration activities, it is natural to engage them. This research methodology was extended to introduce a process where IT integration staff collaborate with domain experts to leverage existing product and systems' itemized inventories in documenting and pairing the acquired systems and products, with their respective dialect IDs defined either in the document elements or linked XSD file, and generating their corresponding Schematron rule files beforehand, to eliminate the risk of dialect misidentification, and thus enabling the research solution to scale easily to support many more dialects. As the company grows and more products are added, this framework would handle new dialects in the same uniform way as shown in Figure 42 on page 79. PaceJena is still invoked on validation failure to retrieve pertinent relations and symptom code explications.

4.8 A More Scalable Approach for Real-World Applications

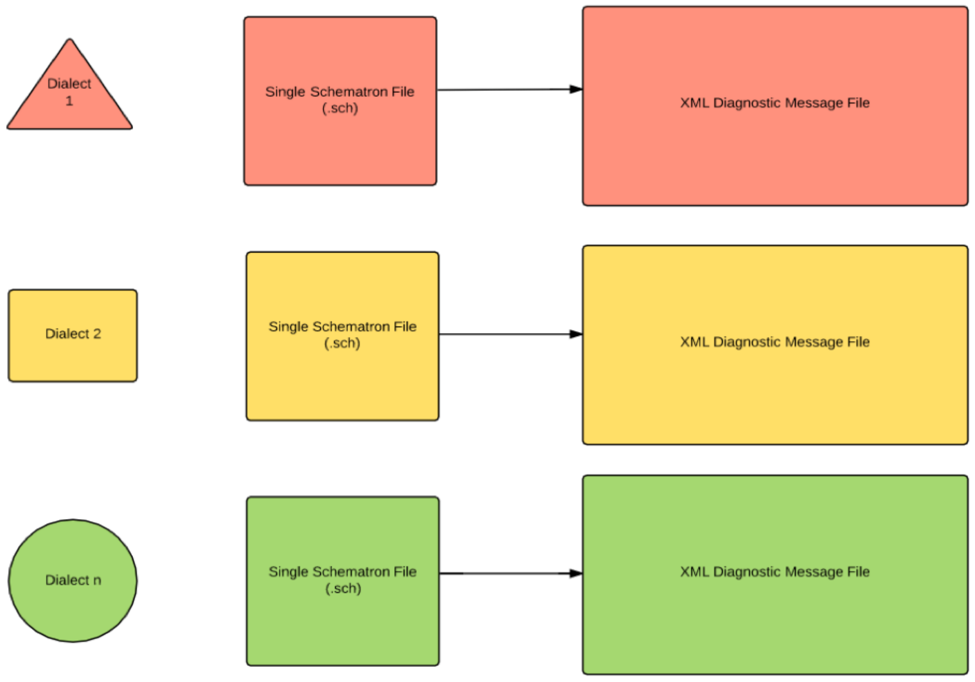

Dialect 1
Single Schematron File (.sch)
XML Diagnostic Message File
Dialect 2
Single Schematron File (.sch)
XML Diagnostic Message File
Dialect n
Single Schematron File (.sch)
XML Diagnostic Message File

Figure 42 XML Diagnostic File with Schematron Rule Files Created Beforehand

4.8.1 Element-based or In-Header Dialect Specification Use-Case

This process would be applied to handle different dialect like the one depicted in Figure 39 on page 76 below where the dialect is specified in the siteinfo element.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<sysdiag type="System Health Report">
    <siteinfo id="1">
    <siteid>100099</siteid>
    <company>ACME</company>
    <model>Extra Maax </model>
    <error_count>07D4FE2> 62</error_count >
    <error_count >07D4FE2> 62</error_count >
    <error_count >22DFEED>24</error_count >
    <error_count >10D0EEB>15</error_count >
  </siteinfo>
</sysdiag>
```

Figure 43 Another XML Diagnostic File Dialect

4.8.2 Research Data Generation

The generated data are as close to industry standard as possible. File and data generated by Java Programs run from cron jobs.

4.9 **Research Platform Equipment**

Hardware:

19" MacBook Pro Running Mac OS X Version 10.7.5 (Lion)

Software:

Eclipse IDE, Keplar version.

Central Processing Power (CPU)

Processor 2.66 GHz Intel Core i7

Memory 8 GB 1067 MHz DDR3

4.10 **Conclusion**

This chapter expanded on the description of the solution methodology information in Chapter 3 by providing additional detail on the novel coupling of Schematron and expressive OWL semantic web structures in this research. This chapter covered this research framework validation methodology in more detail, elaborating on the sematic validation and explication of the data integration workflow step-by-step from beginning to end, to clarify the process. The next chapter will describe how to set up a platform for experimental demonstration to validate this research methodology.

Chapter 5

Interpretation and Evaluation of Solution Methodology

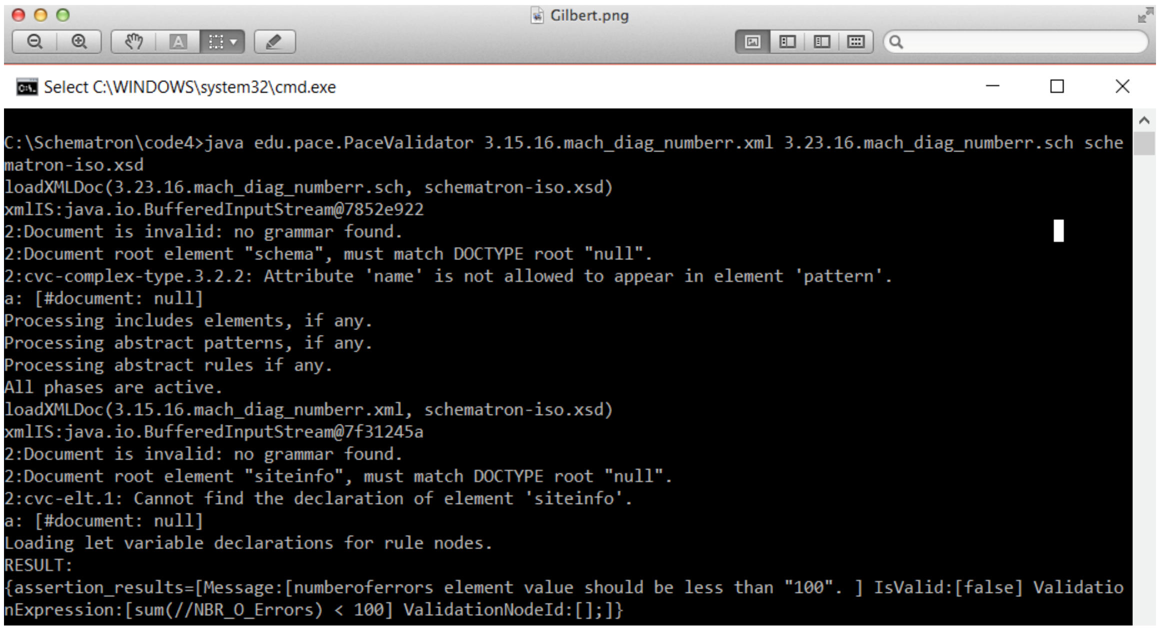
5.1 Evaluating the Solution

This chapter describes how to set up a platform for experimental demonstration to validate this research methodology. PaceSchematron and the research validation extension program were both developed using Java in the Eclipse IDE. Pace-extended Schematron is platform-independent and can run on any major operating system such as Windows Linux OS X Lion (10.7.5) on the Mac.

This research was verified on a MacBook Pro running OS X. As mentioned previously, this research also uses Protégé for creating the knowledge graph. It also uses PaceJena for semantic processing of the knowledge graph as we saw in the previous chapter. All of these technologies converge to address the XML Semantic Rule Complexity Heterogeneity problem (SERCH).

5.2 Research Experiment Demonstration

The methodology validation prototype of this work uses PaceJena, to integrate Schematron validation with a knowledge graph into the research solution. Figures 40, 41, and 42 demonstrate a Schematron semantic constraints validation session, where PaceJena is invoked to retrieve symptom code explications and relations parsed from the OWL file.



```
C:\Schematron\code4>java edu.pace.PaceValidator 3.15.16.mach_diag_numberr.xml 3.23.16.mach_diag_numberr.sch sche
matron-iso.xsd
loadXMLDoc(3.23.16.mach_diag_numberr.sch, schematron-iso.xsd)
xmlIS:java.io.BufferedInputStream@7852e922
2:Document is invalid: no grammar found.
2:Document root element "schema", must match DOCTYPE root "null".
2:cvc-complex-type.3.2.2: Attribute 'name' is not allowed to appear in element 'pattern'.
a: [#document: null]
Processing includes elements, if any.
Processing abstract patterns, if any.
Processing abstract rules if any.
All phases are active.
loadXMLDoc(3.15.16.mach_diag_numberr.xml, schematron-iso.xsd)
xmlIS:java.io.BufferedInputStream@7f31245a
2:Document is invalid: no grammar found.
2:Document root element "siteinfo", must match DOCTYPE root "null".
2:cvc-elt.1: Cannot find the declaration of element 'siteinfo'.
a: [#document: null]
Loading let variable declarations for rule nodes.
RESULT:
{assertion_results=[Message:[numberoferrors element value should be less than "100". ] IsValid:[false] Validatio
nExpression:[sum(//NBR_O_Errors) < 100] ValidationNodeId:[];]}
```

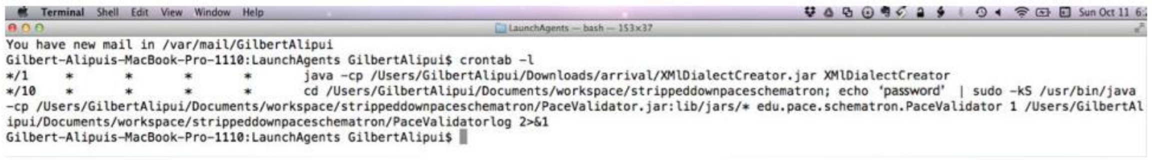
Figure 44 Running PaceValidator from the Windows Command Line

5.3 Research Data Generation

The generated data are as close to industry standard as possible. File and data generated by Java Programs run from cron jobs.

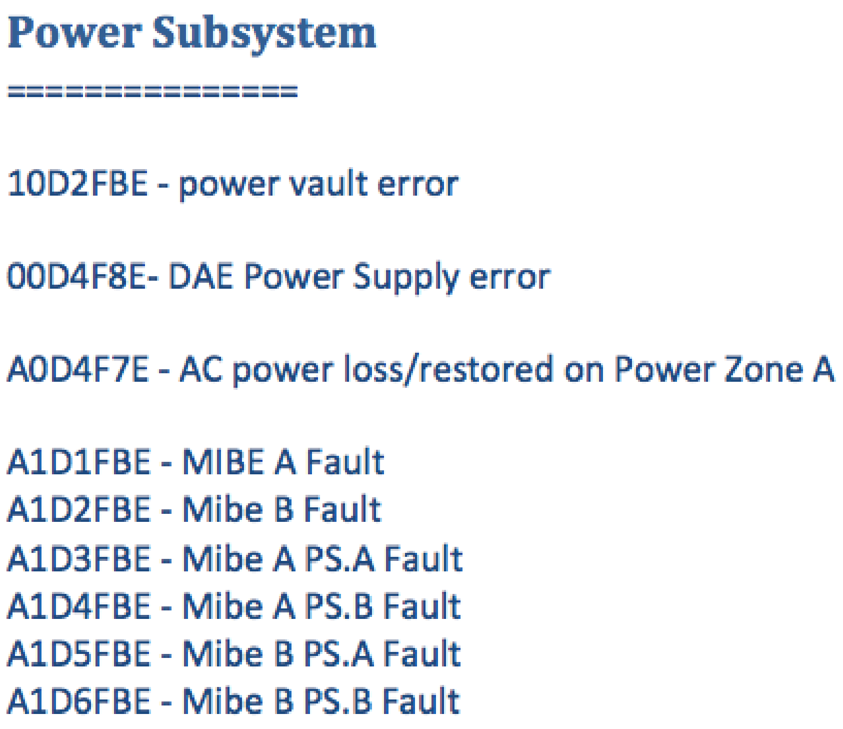
5.4 Crontab File Content

Research data are generated by a cron-driven java program. Figure 43 below depicts an excerpt of the cron file.


Terminal Shell Edit View Window Help
Sun Oct 11
LaunchAgents — bash — 153x37

```
You have new mail in /var/mail/GilbertAlipui
Gilbert-Alipuis-MacBook-Pro-1110:LaunchAgents GilbertAlipui$ crontab -l
*/1     *       *       *       *       java -cp /Users/GilbertAlipui/Downloads/arrival/XMlDialectCreator.jar XMlDialectCreator
*/10    *       *       *       *       cd /Users/GilbertAlipui/Documents/workspace/strippeddownpaceschematron; echo 'password' | sudo -kS /usr/bin/java
-cp /Users/GilbertAlipui/Documents/workspace/strippeddownpaceschematron/PaceValidator.jar:lib/jars/* edu.pace.schematron.PaceValidator 1 /Users/GilbertAl
ipui/Documents/workspace/strippeddownpaceschematron/PaceValidatorlog 2>&1
Gilbert-Alipuis-MacBook-Pro-1110:LaunchAgents GilbertAlipui$
```

Figure 45 Crontab File Content



Power Subsystem

===============

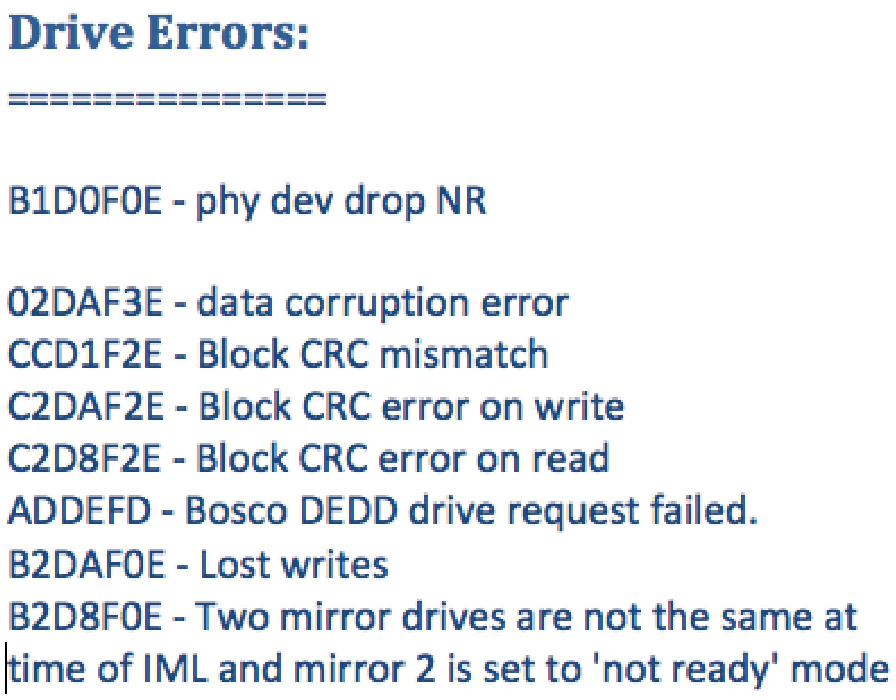
10D2FBE - power vault error

00D4F8E- DAE Power Supply error

A0D4F7E - AC power loss/restored on Power Zone A

A1D1FBE - MIBE A Fault
A1D2FBE - Mibe B Fault
A1D3FBE - Mibe A PS.A Fault
A1D4FBE - Mibe A PS.B Fault
A1D5FBE - Mibe B PS.A Fault
A1D6FBE - Mibe B PS.B Fault

Figure 46 Power Subsystem Diagnostic Research Dataset -1



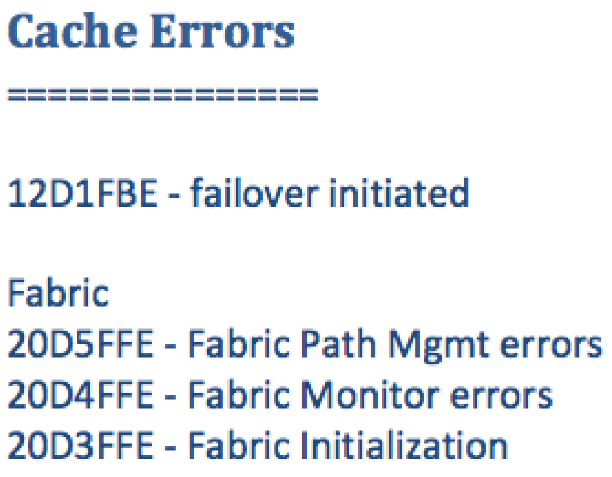
Drive Errors:

===============

B1D0F0E - phy dev drop NR

02DAF3E - data corruption error
CCD1F2E - Block CRC mismatch
C2DAF2E - Block CRC error on write
C2D8F2E - Block CRC error on read
ADDEFD - Bosco DEDD drive request failed.
B2DAF0E - Lost writes
B2D8F0E - Two mirror drives are not the same at time of IML and mirror 2 is set to 'not ready' mode

Figure 47 Drive Subsystem Diagnostics Research Dataset – 2

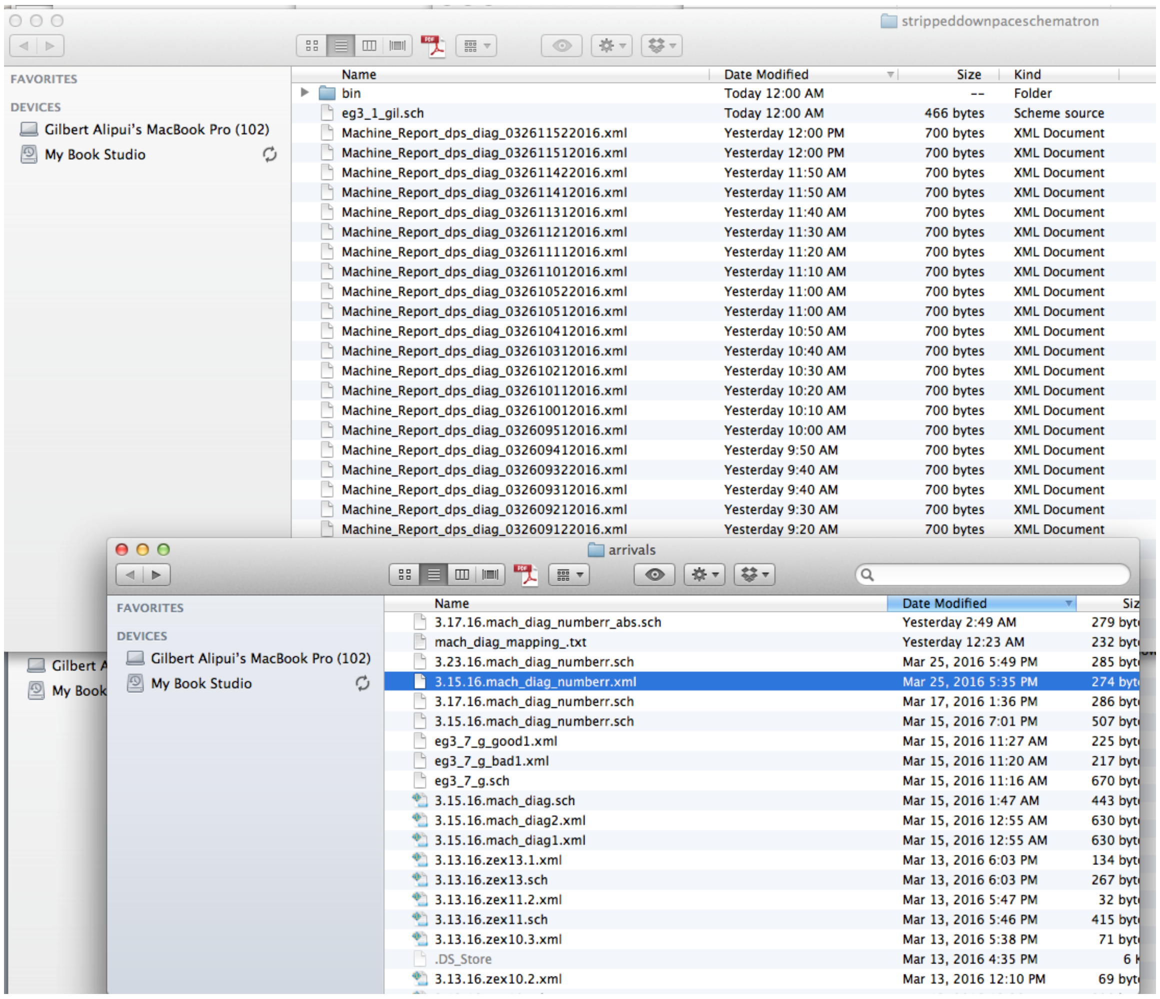


Cache Errors
===============

12D1FBE - failover initiated

Fabric
20D5FFE - Fabric Path Mgmt errors
20D4FFE - Fabric Monitor errors
20D3FFE - Fabric Initialization

Figure 48 Cache Diagnostics Research Dataset - 3

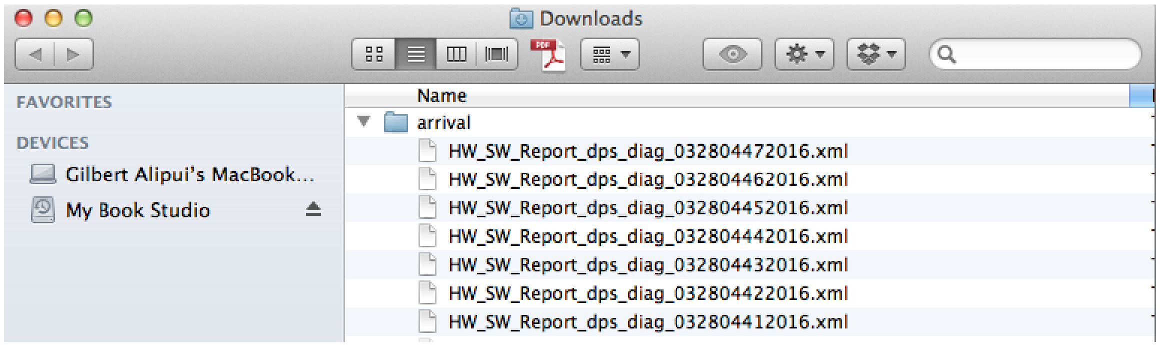


strippeddownpaceschematron

Name	Date Modified	Size	Kind
bin	Today 12:00 AM	--	Folder
eg3_1_gil.sch	Today 12:00 AM	466 bytes	Scheme source
Machine_Report_dps_diag_032611522016.xml	Yesterday 12:00 PM	700 bytes	XML Document
Machine_Report_dps_diag_032611512016.xml	Yesterday 12:00 PM	700 bytes	XML Document
Machine_Report_dps_diag_032611422016.xml	Yesterday 11:50 AM	700 bytes	XML Document
Machine_Report_dps_diag_032611412016.xml	Yesterday 11:50 AM	700 bytes	XML Document
Machine_Report_dps_diag_032611312016.xml	Yesterday 11:40 AM	700 bytes	XML Document
Machine_Report_dps_diag_032611212016.xml	Yesterday 11:30 AM	700 bytes	XML Document
Machine_Report_dps_diag_032611112016.xml	Yesterday 11:20 AM	700 bytes	XML Document
Machine_Report_dps_diag_032611012016.xml	Yesterday 11:10 AM	700 bytes	XML Document
Machine_Report_dps_diag_032610522016.xml	Yesterday 11:00 AM	700 bytes	XML Document
Machine_Report_dps_diag_032610512016.xml	Yesterday 11:00 AM	700 bytes	XML Document
Machine_Report_dps_diag_032610412016.xml	Yesterday 10:50 AM	700 bytes	XML Document
Machine_Report_dps_diag_032610312016.xml	Yesterday 10:40 AM	700 bytes	XML Document
Machine_Report_dps_diag_032610212016.xml	Yesterday 10:30 AM	700 bytes	XML Document
Machine_Report_dps_diag_032610112016.xml	Yesterday 10:20 AM	700 bytes	XML Document
Machine_Report_dps_diag_032610012016.xml	Yesterday 10:10 AM	700 bytes	XML Document
Machine_Report_dps_diag_032609512016.xml	Yesterday 10:00 AM	700 bytes	XML Document
Machine_Report_dps_diag_032609412016.xml	Yesterday 9:50 AM	700 bytes	XML Document
Machine_Report_dps_diag_032609322016.xml	Yesterday 9:40 AM	700 bytes	XML Document
Machine_Report_dps_diag_032609312016.xml	Yesterday 9:40 AM	700 bytes	XML Document
Machine_Report_dps_diag_032609212016.xml	Yesterday 9:30 AM	700 bytes	XML Document
Machine_Report_dps_diag_032609122016.xml	Yesterday 9:20 AM	700 bytes	XML Document

arrivals

Name	Date Modified	Siz
3.17.16.mach_diag_numberr_abs.sch	Yesterday 2:49 AM	279 byt
mach_diag_mapping_.txt	Yesterday 12:23 AM	232 byt
3.23.16.mach_diag_numberr.sch	Mar 25, 2016 5:49 PM	285 byt
3.15.16.mach_diag_numberr.xml	Mar 25, 2016 5:35 PM	274 byt
3.17.16.mach_diag_numberr.sch	Mar 17, 2016 1:36 PM	286 byt
3.15.16.mach_diag_numberr.sch	Mar 15, 2016 7:01 PM	507 byt
eg3_7_g_good1.xml	Mar 15, 2016 11:27 AM	225 byt
eg3_7_g_bad1.xml	Mar 15, 2016 11:20 AM	217 byt
eg3_7_g.sch	Mar 15, 2016 11:16 AM	670 byt
3.15.16.mach_diag.sch	Mar 15, 2016 1:47 AM	443 byt
3.15.16.mach_diag2.xml	Mar 15, 2016 12:55 AM	630 byt
3.15.16.mach_diag1.xml	Mar 15, 2016 12:55 AM	630 byt
3.13.16.zex13.1.xml	Mar 13, 2016 6:03 PM	134 byt
3.13.16.zex13.sch	Mar 13, 2016 6:03 PM	267 byt
3.13.16.zex11.2.xml	Mar 13, 2016 5:47 PM	32 byt
3.13.16.zex11.sch	Mar 13, 2016 5:46 PM	415 byt
3.13.16.zex10.3.xml	Mar 13, 2016 5:38 PM	71 byt
.DS_Store	Mar 13, 2016 4:35 PM	6 k
3.13.16.zex10.2.xml	Mar 13, 2016 12:10 PM	69 byt



arrival
HW_SW_Report_dps_diag_032804472016.xml
HW_SW_Report_dps_diag_032804462016.xml
HW_SW_Report_dps_diag_032804452016.xml
HW_SW_Report_dps_diag_032804442016.xml
HW_SW_Report_dps_diag_032804432016.xml
HW_SW_Report_dps_diag_032804422016.xml
HW_SW_Report_dps_diag_032804412016.xml

Figure 49 XML Dialect Data Integration

5.5 Crucial Components of the Research Solution Methodology

- **XMlDialectCreator.jar**: Generates XML files with different semantics to simulate in-bound XML diagnostic files' transfer. XMlDialectCreator.jar is executed by a cron job running every minute.

- **PaceValidator.jar** identifies the semantic version of the received XML file, generates the concrete rule dynamically from the abstract concept and validates the XML files using the appropriate Schematron schemata, and crucially instantiates a **PaceJena** object, and calls its public methods such as **getClassNames()**, **getClassRelClass(String relation, String ObjectName)** to extract relevant semantics, class names and object relations. Executed by a cron job running every 10 minutes, it logs its execution in **PaceValidatorRunLogtimestamp.txt**

As illustrated in the previous chapter, the research extension to PaceValidator uses regular expressions pattern matching (regex) to identify the dialect found within the received XML file. The dialect can also be explicitly specified in an XSD file, and the file referenced in the XML file. PaceValidator.jar checks to see if the file exists first. If it exists, it identifies the syntax as previously described, then performs some variable substitutions of the relevant tags or elements and normalization to instantiate the concrete rules file.

5.6 Research Validation

Chapter 4 illustrated the data integration and semantic explication process of this research solution. This research was validated by Java extensions to PaceSchematron. Specifically, PaceSchematron was extended with:

1. Multiple dialect classification.
2. Coupling with a knowledge graph for extended semantics and symptom code explication.
3. Reduction of the complexity of integrating XML in different dialects.

5.7 Conclusion

This chapter set up a platform for experimental demonstration to validate this research methodology. Chapter 6 will provide a summary of the major contributions, followed by suggested future work.

Chapter 6

Conclusion

6.1 Coupling Schematron Validation with Knowledge Graph

As far as I am aware, the coupling of Schematron and Knowledge Graph in this research provides a singularly unique and valuable contribution to reduction of complexity of diagnostic message specification and recognition with data relations and explications.

6.2 Summary

After a company merger and acquisition, the acquiring company is often faced with weighty decisions with regard to data integration. In the specific problem domain addressed by this research work, there is added need to improve the identification of actionable errors, fault conditions, and to provide easy to understand semantics of the identified events. Current XML semantic validation methods are syntax-oriented, and fall short when it comes to the validation of concepts. Applying techniques derived from Amer and Tao[15] the validation process of this research employed abstract concept-driven semantic validation, coupled with knowledge graph semantic explication. This process allows the validation rule to be created once, and then mapping is leveraged[15] to create concrete rules that can handle the multiple XML file dialects in the incoming files from the disparate systems.

The major contributions of this research are threefold. This research solution:

1. Is adaptive, and capable of classifying different kinds of system diagnostic information for different products in data integration.
2. Uses knowledge graph-driven to define custom relations, and enhances the meaning of diagnostic events.
3. Reduces the complexity of maintaining multiple versions of similar XML diagnostic file patterns.

6.3 **Future Work**

Future work can automate the entire process, and support more dialects. Extended relations between error codes could be defined to set the foundation for enhanced interpretation, inference, and prediction. With regard to prediction, this research can be extended for predictive modeling, to enable machines to monitor trends, and forecast potential future problems. One such use-case is predicting the risk of a site-wide power event. Future work could also provide support for semantic validation on JSON, and encoding error explication in JSON format. Another use-case is for potential law enforcement or counter terrorism applications. Figure 49 below illustrates one such use-case:

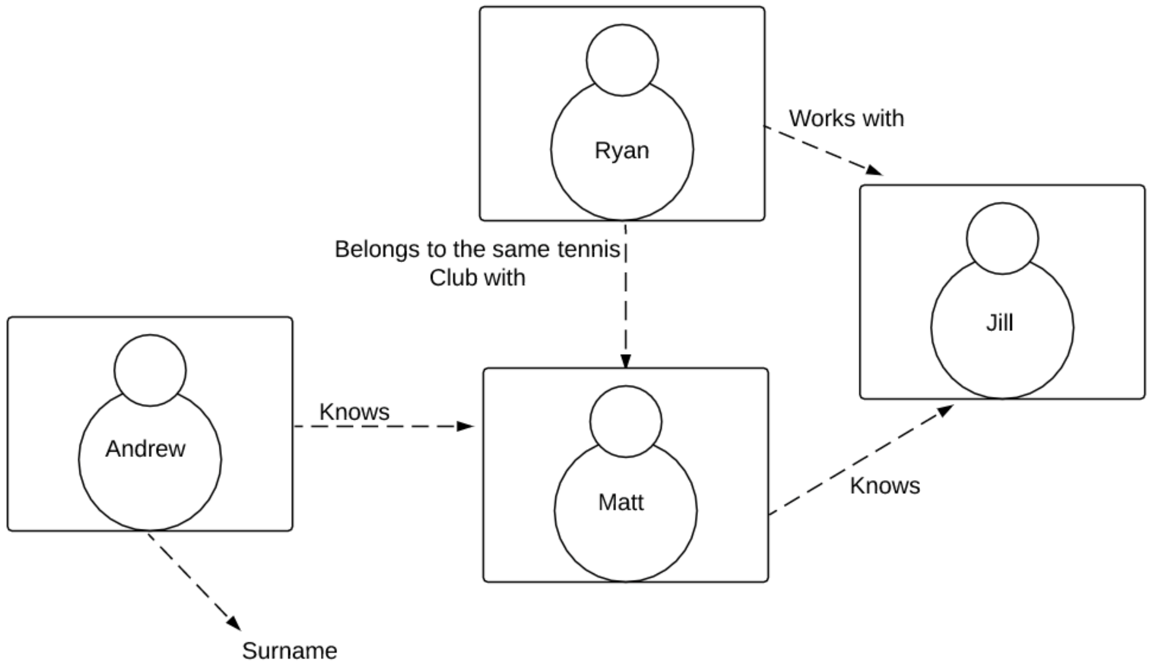
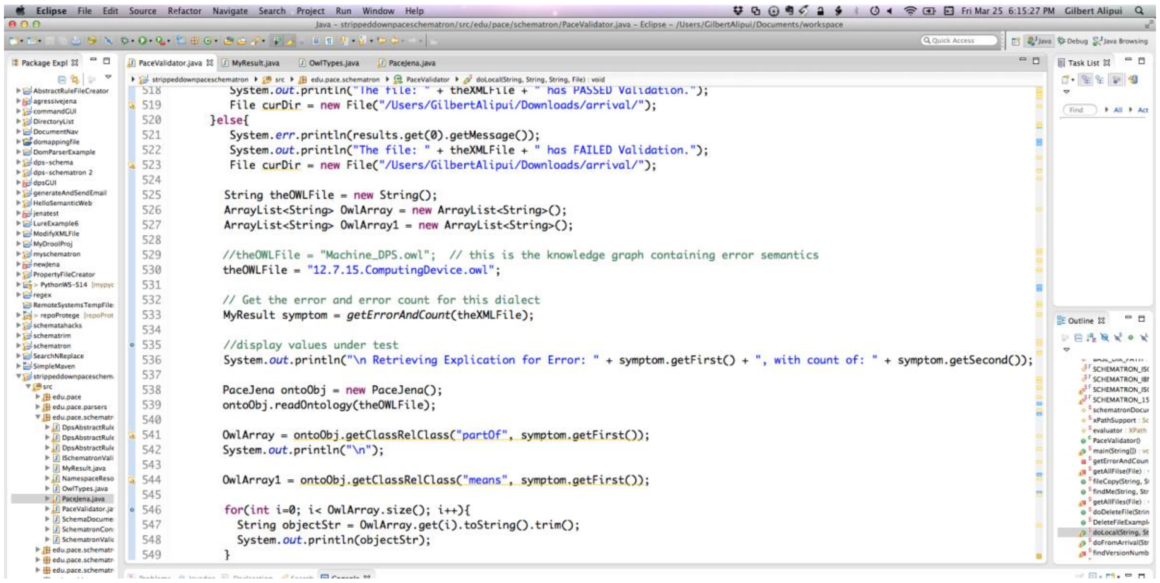


Figure 50 Potential Law Enforcement or Counter-Terrorism Use-Case

Appendix A

Schematron Validation Code

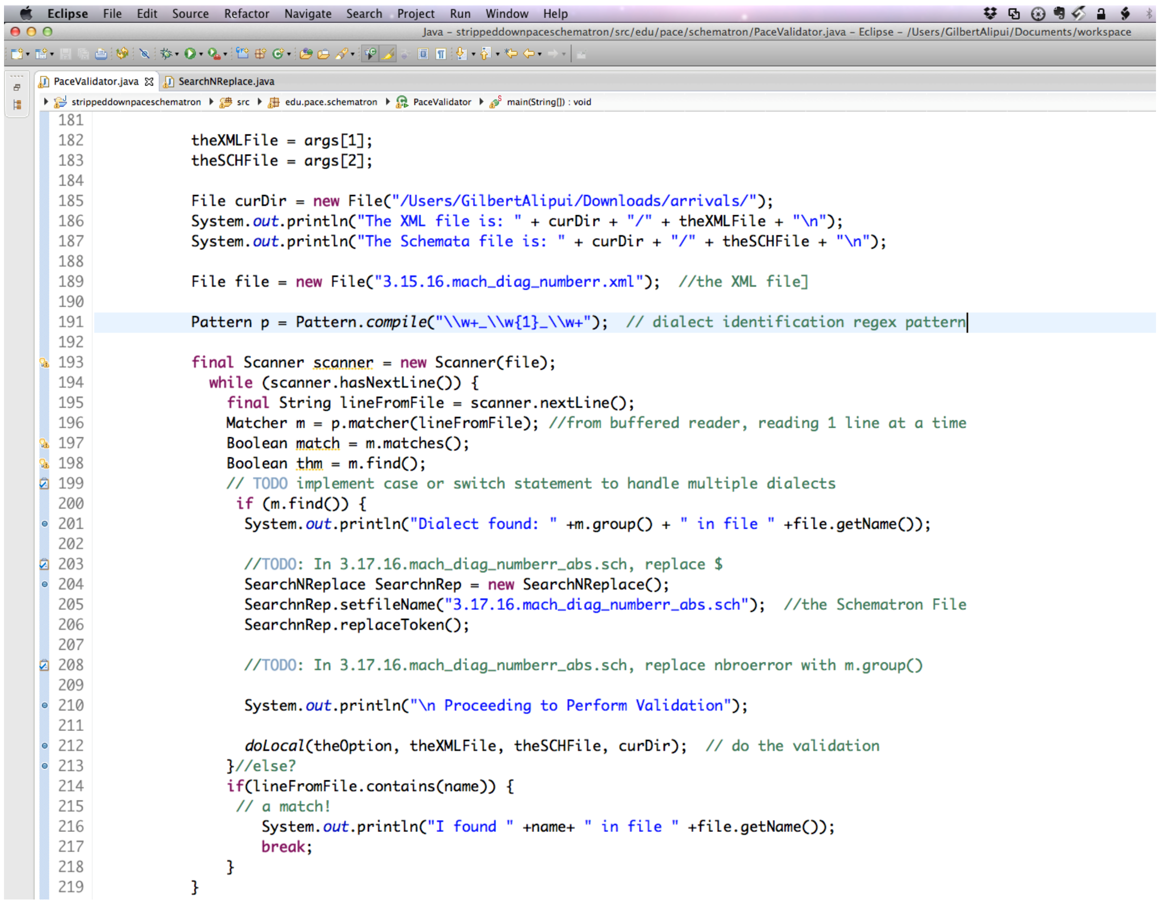


```
            System.out.println("The file: " + theXMLFile + " has PASSED Validation.");
            File curDir = new File("/Users/GilbertAlipui/Downloads/arrival/");
        }else{
            System.err.println(results.get(0).getMessage());
            System.out.println("The file: " + theXMLFile + " has FAILED Validation.");
            File curDir = new File("/Users/GilbertAlipui/Downloads/arrival/");

        String theOWLFile = new String();
        ArrayList<String> OwlArray = new ArrayList<String>();
        ArrayList<String> OwlArray1 = new ArrayList<String>();

        //theOWLFile = "Machine_DPS.owl";  // this is the knowledge graph containing error semantics
        theOWLFile = "12.7.15.ComputingDevice.owl";

        // Get the error and error count for this dialect
        MyResult symptom = getErrorAndCount(theXMLFile);

        //display values under test
        System.out.println("\n Retrieving Explication for Error: " + symptom.getFirst() + ", with count of: " + symptom.getSecond());

        PaceJena ontoObj = new PaceJena();
        ontoObj.readOntology(theOWLFile);

        OwlArray = ontoObj.getClassRelClass("partOf", symptom.getFirst());
        System.out.println("\n");

        OwlArray1 = ontoObj.getClassRelClass("means", symptom.getFirst());

        for(int i=0; i< OwlArray.size(); i++){
          String objectStr = OwlArray.get(i).toString().trim();
          System.out.println(objectStr);
        }
```

Figure 51 Schematron Validation Code Listing

Appendix B

Dialect Identification Code

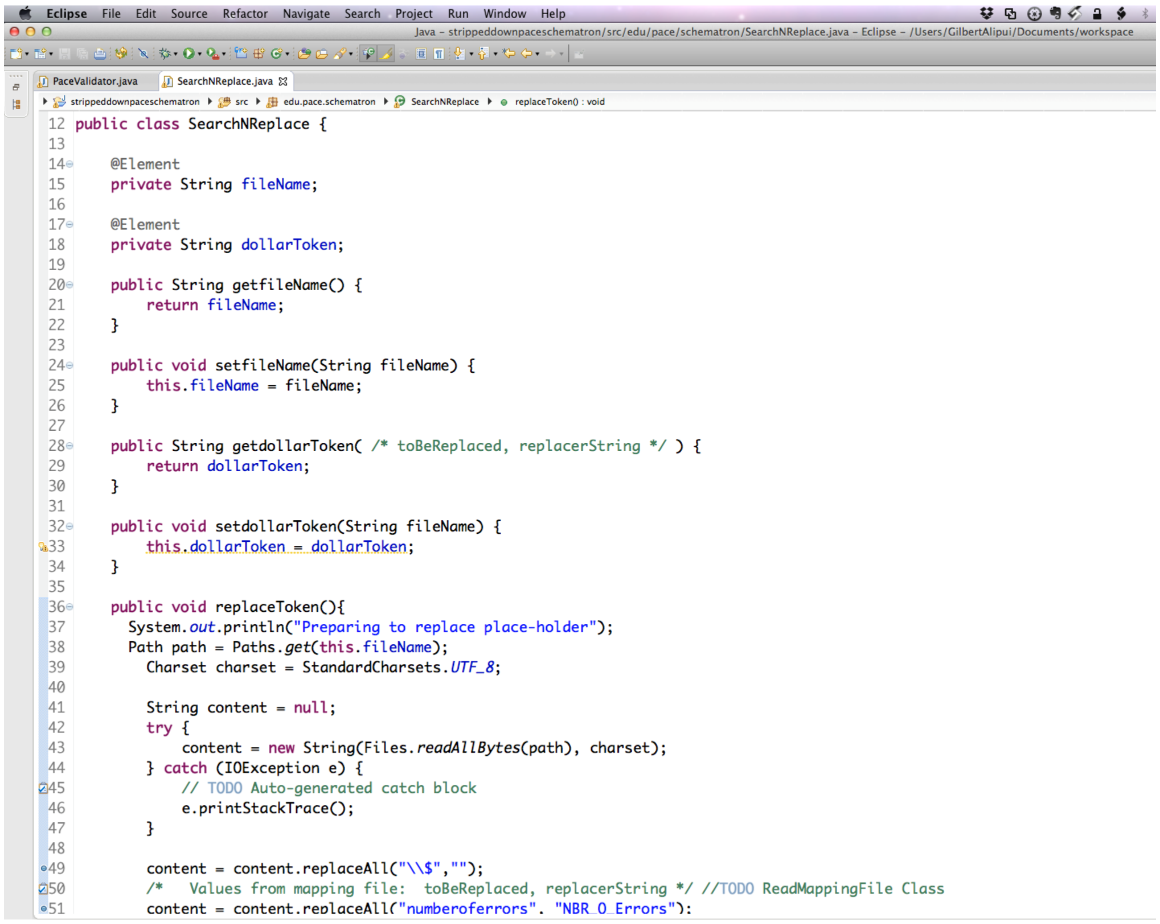

Eclipse File Edit Source Refactor Navigate Search Project Run Window Help
Java – strippeddownpaceschematron/src/edu/pace/schematron/PaceValidator.java – Eclipse – /Users/GilbertAlipui/Documents/workspace
PaceValidator.java
SearchNReplace.java
strippeddownpaceschematron
src
edu.pace.schematron
PaceValidator
main(String[]) : void

```java

            theXMLFile = args[1];
            theSCHFile = args[2];

            File curDir = new File("/Users/GilbertAlipui/Downloads/arrivals/");
            System.out.println("The XML file is: " + curDir + "/" + theXMLFile + "\n");
            System.out.println("The Schemata file is: " + curDir + "/" + theSCHFile + "\n");

            File file = new File("3.15.16.mach_diag_numberr.xml");  //the XML file]

            Pattern p = Pattern.compile("\\w+_\\w{1}_\\w+");  // dialect identification regex pattern

            final Scanner scanner = new Scanner(file);
              while (scanner.hasNextLine()) {
                final String lineFromFile = scanner.nextLine();
                Matcher m = p.matcher(lineFromFile); //from buffered reader, reading 1 line at a time
                Boolean match = m.matches();
                Boolean thm = m.find();
                // TODO implement case or switch statement to handle multiple dialects
                 if (m.find()) {
                  System.out.println("Dialect found: " +m.group() + " in file " +file.getName());

                  //TODO: In 3.17.16.mach_diag_numberr_abs.sch, replace $
                  SearchNReplace SearchnRep = new SearchNReplace();
                  SearchnRep.setfileName("3.17.16.mach_diag_numberr_abs.sch");  //the Schematron File
                  SearchnRep.replaceToken();

                  //TODO: In 3.17.16.mach_diag_numberr_abs.sch, replace nbroerror with m.group()

                  System.out.println("\n Proceeding to Perform Validation");

                  doLocal(theOption, theXMLFile, theSCHFile, curDir);  // do the validation
                }//else?
                if(lineFromFile.contains(name)) {
                 // a match!
                    System.out.println("I found " +name+ " in file " +file.getName());
                    break;
                }
            }
```

Figure 52 Dialect Identification Code Listing

Appendix C

Abstract-to-Concrete Rule Substitution Code

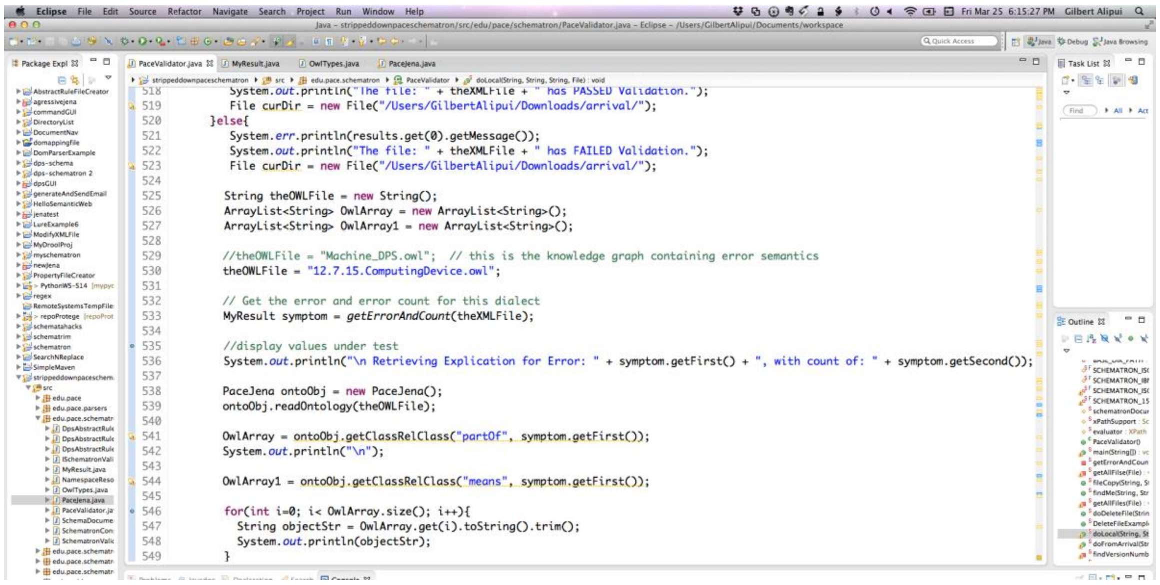

Eclipse File Edit Source Refactor Navigate Search Project Run Window Help
Java – strippeddownpaceschematron/src/edu/pace/schematron/SearchNReplace.java – Eclipse – /Users/GilbertAlipui/Documents/workspace
PaceValidator.java
SearchNReplace.java
strippeddownpaceschematron ▸ src ▸ edu.pace.schematron ▸ SearchNReplace ▸ replaceToken() : void

```
public class SearchNReplace {

    @Element
    private String fileName;

    @Element
    private String dollarToken;

    public String getfileName() {
        return fileName;
    }

    public void setfileName(String fileName) {
        this.fileName = fileName;
    }

    public String getdollarToken( /* toBeReplaced, replacerString */ ) {
        return dollarToken;
    }

    public void setdollarToken(String fileName) {
        this.dollarToken = dollarToken;
    }

    public void replaceToken(){
      System.out.println("Preparing to replace place-holder");
      Path path = Paths.get(this.fileName);
        Charset charset = StandardCharsets.UTF_8;

        String content = null;
        try {
            content = new String(Files.readAllBytes(path), charset);
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        content = content.replaceAll("\\$","");
        /*   Values from mapping file:  toBeReplaced, replacerString */ //TODO ReadMappingFile Class
        content = content.replaceAll("numberoferrors", "NBR_0_Errors");
```

Figure 53 Abstract-to-Concrete Rule Substitution Code

Appendix D

Semantics and Explication Retrieval Code



```
        System.out.println("The file: " + theXMLFile + " has PASSED Validation.");
        File curDir = new File("/Users/GilbertAlipui/Downloads/arrival/");
      }else{
        System.err.println(results.get(0).getMessage());
        System.out.println("The file: " + theXMLFile + " has FAILED Validation.");
        File curDir = new File("/Users/GilbertAlipui/Downloads/arrival/");

       String theOWLFile = new String();
       ArrayList<String> OwlArray = new ArrayList<String>();
       ArrayList<String> OwlArray1 = new ArrayList<String>();

       //theOWLFile = "Machine_DPS.owl";  // this is the knowledge graph containing error semantics
       theOWLFile = "12.7.15.ComputingDevice.owl";

       // Get the error and error count for this dialect
       MyResult symptom = getErrorAndCount(theXMLFile);

       //display values under test
       System.out.println("\n Retrieving Explication for Error: " + symptom.getFirst() + ", with count of: " + symptom.getSecond());

       PaceJena ontoObj = new PaceJena();
       ontoObj.readOntology(theOWLFile);

       OwlArray = ontoObj.getClassRelClass("partOf", symptom.getFirst());
       System.out.println("\n");

       OwlArray1 = ontoObj.getClassRelClass("means", symptom.getFirst());

       for(int i=0; i< OwlArray.size(); i++){
         String objectStr = OwlArray.get(i).toString().trim();
         System.out.println(objectStr);
       }
```

Figure 54 Semantics and Explication Retrieval Code

References

[1] Integrated Syntactic/Semantic XML Data Validation with a Reusable Software Component. Steven Golikov DPS Dissertation. Pace University, White Plains, N.Y. 2012.

[2] http://shop.oreilly.com/product/9780596527716.do. (March, 2016)

[3] J. Hebeler, M. Fisher, R. Blace and A. Perez-Lopez, “Semantic Web Programming,” pp.100, 2009.

[4] J. Hebeler, M. Fisher, R. Blace and A. Perez-Lopez, “Semantic Web Programming,” pp.480, 2009.

[5] J. Hebeler, M. Fisher, R. Blace and A. Perez-Lopez, “Semantic Web Programming,” pp.6, 2009.

[6] J. Hebeler, M. Fisher, R. Blace and A. Perez-Lopez, “Semantic Web Programming,” pp.7, 2009.

[7] J. Hebeler, M. Fisher, R. Blace and A. Perez-Lopez, “Semantic Web Programming,” pp.69, 2009.

[8] J. Hebeler, M. Fisher, R. Blace and A. Perez-Lopez, “Semantic Web Programming,” pp.486, 2009.

[9] J. Hebeler, M. Fisher, R. Blace and A. Perez-Lopez, “Semantic Web Programming,” pp.468, 2009.

[10] J. Hebeler, M. Fisher, R. Blace and A. Perez-Lopez, “Semantic Web Programming,” pp.29, 2009.

[11] IBM Developer Works. A hands-on introduction to Schematron. pp.3 - 6. Uche Ogbuji, Fourthought, 02 Sep 2004

[12] The Bornhold Group. http://www.bornholtz.com. (March, 2016).

[13] This part of ISO/IEC 19757 uses XPath to identify information items in Schematron schemas. (March, 2016).

[14] http://www.cisco.com/c/en/us/td/docs/storage/san_switches/mds9000/sw/rel_3_x/troubleshooting/guide/trblgd/ts_call.pdf (August 4, 2015).

[15] A. Amer and L. Tao, “Syntax Agnostic Semantic Validation of XML Documents Using Schematron” http://csis.pace.edu/~ctappert/srd2014/c3.pdf , pp.5 (July 8, 2015).

[16] http://www.zvon.org/xxl/SchematronTutorial/Examples/Example13/example.html (July 8, 2015).

[17] http://www.storagenewsletter.com/rubriques/systems-raid-nas-san/emc-entry-level-symmetrix-vmax-10k/ (July 17, 2015).

[18] http://www.consultoriaadhoc.com/netapp_oracle.aspx (July 17, 2015).

[19] https://store.emc.com/us/Product-Family/EMC-VMAX-Products/EMC-VMAX-10K/p/EMC-VMAX10K (July 17, 2015).

[20] http://www.information-management.com/issues/20010401/3170-1.html (July 17, 2015).

[21] https://www.w3.org/TR/xml11/ section 2.3. (March 16, 2016).

[22] Ontology-Based Integration of Information — A Survey of Existing Approaches. H.Wache, T. V¨ogele, U. Visser, H. Stuckenschmidt,G. Schuster, H. Neumann and S. H¨ubner

[23] A Feasibility Study of Ontology-Based Automatic Document Transformation. Anne Manette Wright DPS Dissertation. Pace University, White Plains, N.Y. 2009.

[24] http://broadcast.oreilly.com/2012/03/xmls-dialect-problem.html (July 17, 2015).

[25] Tao and Golikov, "Integrated Syntax and Semantic Validation for Services Computing." http://csis.pace.edu/~ctappert/srd2013/d2.pdf (July 29, 2015).

[26] Rodrigues, Rosa, and Cardoso, "Mapping XML to Existing OWL Ontologies." https://eden.dei.uc.pt/~jcardoso/Research/Papers/CP-2006-026-WWW-Internet-Mapping-XML-to-existing-OWL-ontologies.pdf (July 29, 2015)

[27] "Yu, *A Developer's Guide to the Semantic Web*." pp. 72, 2011.

[28] Kumaran and Tao, "Efficient Information Access in Enterprise Content Management Systems Using Semantic Technologies."(August 3, 2015)

[29] http://www.w3.org/TR/2008/WD-rif-ucr-20081218/#Interchanging_Rule_Extensions_to_OWL

[30] http://www.alquier.org//laurent/documents/conferences/www2010-linked-data-framework-final.pdf pp.10, (March 17, 2015)

[31] https://www.w3.org/2001/sw/BestPractices/OEP/SemInt/

[32] "8.4.15.Taming Data Chaos- Simplify Data Integration.pdf". An ENTERPRISE MANAGEMENT ASSOCIATES® (EMA™) White PaperPrepared for DataDirect March 2009

[33] http://cdn.ttgtmedia.com/searchSOA/downloads/17_791199_ch14.pdf. (August 4, 2015) (8.14.15.Integrating and Automating Business Processes_ch14.pdf)

[34] K. Parris, "http://www.technewsworld.com/story/69267.html", Feb 4, 2010 5:00 AM PT

[35] http://www.infoq.com/articles/lawson-casestudy. (August 4, 2015)

[36] http://www.emc.com/collateral/software/white-papers/h10775-emc-it-integration-mergs-acqs-wp.pdf. (8.4.15.h10775-emc-it-integration-mergs-acqs-wp (1).pdf) (August 4, 2015)

[37] Liu, Zhang, and Tang, "Grid-Enabled Fault Diagnostic System for Manufacturing Equipments."(8.6.15.Grid-enabled Fault Diagnostic System for Manufacturing Equipments.pdf)

[38] 8.6.15.Symptoms Ontology for Mapping Diagnostic Knowledge Systems .pdf

[39] http://knoesis.wright.edu/library/download/SemanticEnterpriseContentManagement.pdf

[40] http://www.xml.com/pub/a/2002/11/06/ontologies.html

[41] http://www.xml.com/2002/11/06/Ontology_Editor_Survey.html

[42] Fisher and Sheth, "Semantic Enterprise Content Management." 8.6.15.Semantic Enterprise Content Management.pdf (For OWL Fodder)

[43] http://austria.emc.com/collateral/TechnicalDocument/docu55298.pdf

[44] http://broadcast.oreilly.com/2012/03/xmls-dialect-problem.html. Schematron abstract patterns.

[45] http://www.ibm.com/developerworks/library/x-stron/

[46] http://www.devarticles.com/c/a/XML/Schematron-Patterns-and-Validation/1/

[47] http://www.dpawson.co.uk/schematron/abstract.html

[48] http://www.schematron.com/iso/P8.html (is-a)

[49] http://www.schematron.com/iso/P8.html

[50] http://www.xml.com/pub/a/2003/11/12/schematron.html?page=2 (Nice abstract pattern explanation using $time)

[51] http://web1.see.asso.fr/erts2012/Site/0P2RUC89/1D-1.pdf (August 7, 2015).

[52] http://www.sourcetricks.com/2014/02/parse-xml-file-using-xpath-in-java.html#.VdSV5CxVhBc

[53] file:///Users/GilbertAlipui/Downloads/Schematron.XUG.2012.pdf

[54] http://www.javaworld.com/article/2076171/java-se/xml-document-processing-in-java-using-xpath-and-xslt.html

[55] http://www.schematron.com/

[56] http://protegewiki.stanford.edu/wiki/ProtegeOWL_API_Programmers_Guide

[57] https://www.biostars.org/p/10439/

[58] S. Shanmugan and L. Tao, “An Approach to Eliminate Semantic Heterogenity Using Ontologies in Enterprise Data Integration”.(August, 2015)

[59] https://acc.dau.mil/adl/en-US/19593/file/1045/PM_BusinessProcessUseCase_0705.pdf. (September 3, 2015)

[60] 9.3.15.XML Use Cases.pdf. (September 3, 2015)

[61] https://en.wikibooks.org/wiki/XML_-_Managing_Data_Exchange/acord. Insurance comp

[62] http://pluto.ksi.edu/~cyh/cis501/ch8.html. (September 4, 2015) goo importance of validation when companies exchange XML

[63] http://www.xml.com/pub/a/2001/01/24/rdf.html. (September 4, 2015)

[64] http://www.ibm.com/developerworks/library/x-ind-ublcodel/

[65] http://www.iimahd.ernet.in/~jajoo/ibmpc_faq/part4.htm

[66] http://broadcast.oreilly.com/2010/01/microsoft-patents-schematron.html. (Example of outlining limitations)

[67] Fan, Wenfei, Minos Garofalakis, Ming Xiong, and Xibei Jia. “Composable XML Integration Grammars.” In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, 2–11. ACM, 2004. http://dl.acm.org/citation.cfm?id=1031176.

[68] Schematron INTERNATIONAL STANDARD ISO/IEC 19757-3 First edition 2006-06-01 — Information technology — Document Schema Definition Languages (DSDL) — Part 3: Rule-based validation. pp. 6.

[69] https://www.oxygenxml.com/events/2014/schematronUsecases.pdf. (September 16, 2015)

[70] http://www.codeproject.com/Articles/22769/Introduction-to-Object-Oriented-Programming-Concep. (September 16, 2015)

[71] Woensel, William Van, Newres Al Haider, Patrice C. Roy, Ahmad Marwan Ahmad, and Syed S.R. Abidi. “A Comparison of Mobile Rule Engines for Reasoning on Semantic Web Based Health Data,” 126–33. IEEE, 2014. doi:10.1109/WI-IAT.2014.25.

[72] http://www.w3.org/2005/Incubator/ssn/wiki/XML_dataset_content_validation. (September 16, 2015)

[73] http://www.dpawson.co.uk/schematron/introduction.html#arch. (September 17, 2015)

[74] http://www.w3.org/2001/sw/BestPractices/OEP/SimplePartWhole/simple-part-whole-relations-v1.3.html. (September 18, 2015)

[75] Aitken, Webber, and Bard, “Part-of Relations in Anatomy Ontologies.”

http://psb.stanford.edu/psb-online/proceedings/psb04/aitken.pdf. (September 18, 2015)

[76] http://www.w3.org/TR/owl-features/#s1.2. (September 22, 2015)

[77] http://www.computer.org/csdl/mags/cs/2004/04/c4012.pdf. (September 22, 2015)

[78] http://homepages.inf.ed.ac.uk/wadler/xml/#core. (September 22, 2015)

[79] http://www.techrepublic.com/article/xml-too-much-of-a-good-thing.(September 22, 2015)

[80] http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA420617. (September 24, 2015)

[81] http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4185431/. (September 25, 2015)

[82] http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA420617. Appendix B (September 25, 2015)

[83] http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-47/wache.pdf. (September 28, 2015)

[84] http://ceur-ws.org/Vol-47/cui.pdf. (September 28, 2015)

[85] Cui, Zhan, Dean Jones, and Paul O'brien. "Semantic B2B Integration: Issues in Ontology-Based Approaches." *ACM SIGMOD Record* 31, no. 1 (2002): 43–48. (September 28, 2015)

[86] http://www2.cs.uic.edu/~advis/publications/dataint/eis05j.pdf. (September 28, 2015)

[87] http://web.stanford.edu/~natalya/papers/SigmodRecordReview.pdf. (September 28, 2015)

[88] Grasic, Bostjan, and Vili Podgorelec. "Automating Ontology Based Information Integration Using Service Orientation." *WSEAS Transactions on Computers* 9, no. 6 (2010): 1109–2750

[89] Uschold, Mike, and Michael Gruninger. "Ontologies: Principles, Methods and Applications." *The Knowledge Engineering Review* 11, no. 02 (1996): 93–136.

http://www.upv.es/sma/teoria/sma/onto/96-ker-intro-ontologies.pdf. (October 5, 2015)

[90] http://ubercrawl.com/wp-content/uploads/2011/01/The-Semantic-Web.pdf

10.8.15.The-Semantic-Web.pdf

[91] https://homes.cs.washington.edu/~alon/files/acmq.pdf (October 9, 2015)10.8.15.Why Your Data Won't Mix- Semantic Heterogeneity.pdf

Hakimpour, Farshad, and Andreas Geppert. "Resolving Semantic Heterogeneity in Schema Integration." In *Proceedings of the International Conference on Formal Ontology in Information Systems-Volume 2001*, 297–308. ACM, 2001. http://dl.acm.org/citation.cfm?id=505196.

[92] Hammer, Joachim, and Dennis McLeod. "An Approach to Resolving Semantic Heterogeneity in a Federation of Autonomous, Heterogeneous Database Systems." *International Journal of Intelligent and Cooperative Information Systems* 2, no. 01 (1993): 51–83.

[93] Pinto, Carlos Sousa, Herlina Jayadianti, Lukito Edi Nugroho, and Paulus Insap Santosa. "Solving Problems of Data Heterogeneity, Semantic Heterogeneity and Data Inequality: An Approach Using Ontologies." In *MCIS2012-The 7th Mediterranean Conference on Information Systems, In Knowledge and Technologies in Innovative Information Systems*, 2012. http://repositorium.sdum.uminho.pt/handle/1822/23089.

[94] Kotenko, Igor, Olga Polubelova, Andrey Chechulin, and Igor Saenko. "Design and Implementation of a Hybrid Ontological-Relational Data Repository for SIEM Systems." *Future Internet* 5, no. 3 (July 9, 2013): 355–75. doi:10.3390/fi5030355.

[95] Ghawi, Raji, and Nadine Cullot. "Database-to-Ontology Mapping Generation for Semantic Interoperability." In *VDBL'07 Conference, VLDB Endowment ACM*, 1–8, 2007. http://test2010le2i.u-bourgogne.fr/le2i/IMG/publications/InterDB07-Ghawi.pdf.

[96] Yu, Liyang. *Introduction to Semantic Web and Semantic Web Services*. Boca Raton: Chapman & Hall/CRC, pp.49, 2007.

[97] Yu, Liyang. *Introduction to Semantic Web and Semantic Web Services*. Boca Raton: Chapman & Hall/CRC, pp.95, 2007.

[98] http://www.w3.org/TR/2012/REC-owl2-rdf-based-semantics-20121211/

(October 13, 2015)

[99] Shenoy, Manjula, K. C. Shet, and U. Dinesh Acharya. "OWL Based XML Data Integration." *International Journal of Computer Applications* 68, no. 2 (2013). http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.404.1441&rep=rep1&type=pdf.

[100] 1.20.15.XML in a Nutshell_Tao.pdf, Elliotte Rusty Harold, W. Scott Means pp.473

[101] http://www.javaworld.com/article/2076141/java-se/mapping-xml-to-java--part-1.html. (October 20, 2015)

[102] XML 1.1 Bible, 3rd Edition.pdf. p.53

[103] http://www.w3schools.com/xsl/xquery_intro.asp. (October 20, 2015)

[104] 2.14.15.XML Programming Bible.pdf p.80

[105] Schematron.XUG.2012.pdf. INERA Inc. XUG 2012

[106] https://en.wikipedia.org/wiki/International_Standard_Serial_Number

[107] https://en.wikipedia.org/wiki/Digital_object_identifier

[108] Scap.nist.gov, "Security Content Automation Protocol". 2012. Available at http://scap.nist.gov/revision/1.2/schematron.html.

[109] Scridb.com, "XML Data Encoding Specification for Intelligence Publications Version 6 (PUBS.XML.V6)". 2012. Available at http://www.scribd.com/doc/61419741/Spying-Documents-Specifications.

[110] https://en.wikipedia.org/wiki/World_Wide_Web_Consortium. (October 23, 2015)

[111] http://www.w3.org/Consortium/. (October 23, 2015)

[112] http://www.w3.org/standards/xml/schema. (October 23, 2015)

[113] Lehti, Patrick, and Peter Fankhauser. "XML Data Integration with OWL: Experiences and **Challenges**." In *Applications and the Internet, 2004. Proceedings. 2004 International Symposium on*, 160–67. IEEE, 2004. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1266111.

[114] http://ftp.cse.buffalo.edu/users/azhang/disc/SIGMOD/pdf-files/563/721-nimble.pdf. Draper, Halevy and Weld. The Nimble Integration Engine XML.pdf.

[115] http://www.w3schools.com/xml/xml_rdf.asp. (October 26, 2015)

[116] http://www.w3.org/TR/2004/REC-owl-features-20040210/#s1.3. (October 26, 2015)

[117] http://www.w3.org/TR/owl2-primer/#Introduction. (October 26, 2015)

[118] https://en.wikipedia.org/wiki/Internet_Engineering_Task_Force. (October 27, 2015)

[119] http://ietf.org. (October 27, 2015)

[120] 10.28.15.Assessing Heterogeneity by Classifying Ontology Mismatches.pdf

[121] 9.26.15.Ontology Based Integration of Information.pdf

[122] The IEEE computing edge October, 2015 pp. 22-28. "Trustworthy Processing of Healthcare Big Data in Hybrid Clouds". Blue Skies Department.

[123] Jennings, Nicholas R., Anthony G. Cohn, Maria Fox, Derek Long, Michael Luck, Danius T. Michaelides, Steve Munroe, and Mark J. Weal. "Motivation, Planning and Interaction." *Cognitive Systems: Information Processing Meets Brain Science*, 2006, 163–88.