AUTOMATED SOFTWARE SYSTEM FOR CHECKING THE STRUCTURE AND FORMAT OF ACM SIG DOCUMENTS

A THESIS SUBMITTED TO THE GRADUATE SCHOOL OF APPLIED SCIENCES OF NEAR EAST UNIVERSITY

By
ARSALAN RAHMAN MIRZA

In Partial Fulfillment of the Requirements for The Degree of Master of Science in Software Engineering

NICOSIA, 2015

ACKNOWLEDGEMENTS

This thesis would not have been possible without the help, support and patience of my principal supervisor, my deepest gratitude goes to Assist. Prof. Dr. Melike Şah Direkoglu, for her constant encouragement and guidance. She has walked me through all the stages of my research and writing thesis. Without her consistent and illuminating instruction, this thesis could not have reached its present from.

Above all, my unlimited thanks and heartfelt love would be dedicated to my dearest family for their loyalty and their great confidence in me. I would like to thank my parents for giving me a support, encouragement and constant love have sustained me throughout my life. I would also like to thank the lecturers in software/computer engineering department for giving me the opportunity to be a member in such university and such department. Their help and supervision concerning taking courses were unlimited.

Eventually, I would like to thank a man who showed me a document with wrong format, and told me “it will be very good if we have a program for checking the documents”, however I don’t know his name, but he hired me to start my thesis based on this idea.

To Alan Kurdi

To my Nephews
Sina & Nima

ABSTRACT

Microsoft office (MS) word is one of the most commonly used software tools for creating documents. MS office word 2007 and above are formatted using Extensible Markup Language (XML). Metadata about the documents are automatically created using Office Open XML (OOXML) syntax. A new framework was developed, which is called ADFCS (Automated Document Format Checking System) that takes the advantage of the OOXML metadata, in order to extract semantic information from MS word documents. In particular, a new ontology for ACM SIG documents and representing the structure and format of these documents by using OWL ontology language has been developed. Then, the metadata is extracted automatically in RDF according to this ontology using the developed software. Finally, extensive rules are generated in order to infer whether the documents are formatted according to ACM SIG standards. This thesis, introduces ACM SIG ontology, metadata extraction process, inference engine, ADFCS online user interface, system evaluation and user study evaluations.

***Keywords*:** Semantic Web; Jena; Notation 3; document format checking; Metadata; OOXML

ÖZET

Microsoft office (MS) word belgeleri oluşturmak için en sık kullanılan yazılım araçlarından biridir. MS office word 2007 ve üzeri versiyonları, Genişletilebilir Biçimlendirme Dili (XML) kullanılarak biçimlendirilir. Belgelerle ilgili meta veri otomatik olarak Office Open XML (OOXML) sözdizimi kullanılarak oluşturulur. Bu tezde, MS Word belgelerinin anlamsal bilgilerini ayıklamak için OOXML meta verisinden yararlanılarak, ADFCS (Otomatik Belge Formatı Denetleme Biçimi) isminde yeni bir sistem geliştirildi. Özellikle, ACM SIG belgeleri ve OWL ontoloji dili kullanarak bu belgelerin yapısını ve biçimini temsil etmek için yeni bir ontoloji geliştirilmiştir. Ardından, geliştirilen yazılım ve bu ontoloji kullanılarak, elde edilen meta veri vede otomatik olarak RDF verisine çevrildi. Son olarak, geliştirilen kapsamlı kurallar ile belgelerin ACM SIG standartlarına göre biçimlendirilmiş olup olmadığını anlaması sağlanmıştır. Bu tez, ACM SIG ontolojisi, meta veri çıkarma işlemi, sonuç çıkarma motoru, ADFCS online kullanıcı arayüzü, sistem değerlendirme ve kullanıcı çalışma değerlendirmelerini içermektedir.

Anahtar Kelimeler: Semantik Web; Jena; Notation 3; belge biçimi denetimi; Meta veri; OOXML

TABLE OF CONTENTS

APPENDICES

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

A-BOX	Assertion Box
ACM	Associate Computing Machinery
ADFCS	Automated Document Format Checking System
DOM	Document Object Model
DTD	Document Type Definition
ECMA	European Computer Manufacturer Association
FFM	Full Functionality Mode
IEC	International Electro-technical Commission
ISO	International Standard for Organization
JVM	Java Virtual Machine
MS	Microsoft Office
N3	Notation 3 – a format for representing RDF triples
OASIS	Organization for the Advancement of Structured Information Standards
OOXML	Office Open XML
OWL	Ontology Web Language
REST	Representational State Transfer
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
SIG	Special Interest Groups
SAX	Simple API for XML
SGML	Standard Generalized Markup Language
SOAP	Simple Object Access Protocol
SPARQL	SPARQL Protocol and RDF Query Language
T-BOX	Transitive Box
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UOF	Uniform Office Format
WWW	The World Wide Web
W3C	The World Wide Web Consortium
XSD	Xml Schema Definition
XSLT	eXtensible Style sheet Language Transformations
XML	eXtensible Markup Language

CHAPTER 1

INTRODUCTION

Nowadays, most of the software engineering approaches aim to completely or partially automate the software testing processes since manual testing is tedious, time-consuming and error prone. In addition, through automation, the cost of testing is reduced as well as automated testing is more reliable than manual testing approaches. Automated software engineering approaches have been utilized in many areas of software engineering. These include requisites definition, designation, architecture, design, implementation, modelling, testing and quality assurance, verification and validation. Automated software engineering techniques have additionally been used in a wide range of domains and application areas including industrial software, embedded and authentic-time systems, aerospace, automotive and medical systems, Web-based systems and computer games.

1.1 Thesis Problem

For a conference coordinator who deals with hundreds of documents and submitted papers, an automated software system is the best solution for checking format of the documents for its correctness. An automated software system can be implemented for checking the format of the documents, and it is clear that every document needs to be revised for the correctness of its format. The proofreader needs to check all the format standards manually which will not guarantee that the document will be checked for all format standards by the proofreader. Since manual document format checking is time-consuming, error-prone and not reliable. There may be chance of incorrect format of the document or unseen text formatting in it, regardless of the time spent on checking document formats. Furthermore, when the number of documents increases, this process becomes more difficult.

1.2 The Aim of the Thesis

In this thesis a software framework is proposed, called ADFCS[1], which takes into account the automated software engineering process for automating the process of checking the format and structure of ACM SIG documents. The Association for Computing Machinery (ACM) is the world's largest scientific and educational organization for publishing research

[1] The complete code of ADFCS system in one file is available at http://www.semanticdoc.org/acm_doc.java

in the field of computing. As 2011, it has more than 100,000 not-for-profit professional members. ACM is organized into 171 local chapters and 37 Special Interest Groups (SIGs). In addition, numerous numbers of conferences and journals in the field of computing are sponsored by ACM. All of the sponsored conferences and journals require publishing their content according to ACM SIG document format structure. By developing an automated software system framework for automatically checking the format and structure of ACM SIG documents, we aim to help; (1) authors so that they can validate the format of their research papers before submitting to an ACM conference or journal, (2) conference organizers can check the validity of the format of the submitted papers with ease, (3) proofreaders can be supported with our automated software. For developing such software, it is necessary to obtain and evaluate the metadata of the document. In our framework, we extract the metadata of the document according to ACM SIG ontology. Then using Reasoner and created rules, we can build an automated software system for validating the format of documents.

1.3 The Importance of the Thesis

By utilizing the automated document format checking system, the checking process will save time, will be more robust in terms of finding format errors, and will give the opportunity to the proofreader to focus on content only. The document, which needs to be checked is an ACM SIG word document which is submitted to a journal or conference for publishing. Nevertheless, the automating document format checking system can be applied to other document standards by adapting the data extraction process and inference rules.

1.4 Limitation of the Study

The automated document format checking system might be useful only when we have a stable OOXML Schema of document. Since it is not possible to significantly modify an OOXML file format; because when changing the feature of OOXML file format it becomes very difficult to manage the scripts or modify the contents of the file.

The checking process can be performed if XML Schema of the document is well formed; XML Schema describes the element position and its relationship to other elements as well as specifies the constraints on the element. In recent years, MS office documents are added with more and more information types and quantities, such as sound, image, database and

Web information. This makes, office document format more complex and more inconvenient, when processing it.

For automation of checking process it is urgent to access the metadata of document. Metadata means data about data and shows how the data will be presented. Without metadata, there will be only the possibility for extracting the textual content of the document, which is not useful without the semantic information about documents.

1.5 Overview of the Thesis

This thesis, is divided into 8 chapters and organized as follows.

Chapter 1: Introduces the thesis problem, aim of the thesis and the type of problem it is going to be solve.

Chapter 2: Introduces the related research work by defining its aims and motivations. We discussed some previous work related to document format checking system and metadata extraction. Moreover, we discussed converting XML documents to ontologies and other approaches for metadata extraction.

Chapter 3: Introduces the Semantic Web technologies, RDF, RDFS, Ontology and the structure of SPARQL query.

Chapter 4: Introduces ODF and OOXML document format types, comparing to each other and how we can benefit from metadata extraction in OOXML.

Chapter 5: Introduces the framework of ADFCS and how the data from OOXML is extracted and converted into N3 file for semantic processing by Jena. In particular, the SPARQL queries for retrieving data from Jena Reasoner and converting them into a report.

Chapter 6: Introduces our online user interface and system implementation of ADFCS.

Chapter 7: In this chapter, we evaluate and compare the traditional manual checking process with ADFCS automatic checking system for the assessment of ADFCS.

Chapter 8: In this chapter, we summarize the overall thesis and discuss the future work for next version of ADFCS system.

CHAPTER 2

RELATED RESEARCH

In this chapter, we are discussing related research dealing with document format checking, semantic mapping of XML document to ontologies and OOXML document data extraction.

2.1 Document Format Checking

Xu et al. (2010) present a proposal for checking the format of undergraduate's graduation thesis with technology of using java. The study tries to detect the format of the document as follows; first reading the MS word document format and second investigating and analysis the content of the document. This approach uses the java xml parser package for capturing the metadata of document (e.g. page numbers, headers and footers, margins) and then compares the extracted data with the defined format for document. Finally, a report is generated for document. The test rate for this research was more than 95% for the whole process.

Hou et al. (2010), compares documents that are in both OOXML and ODF formats. According to their paper, many components of word processing documents that are in one format have logical counterpart in other one and some component have no counterpart or corresponding relationship. They divide the degree of difficulty of converting between OOXML and ODF into easy, middle and difficult types. In easy type, the components in OOXML and ODF have direct and obvious relationship, and it is easy to convert from one format to other. For example paragraph and table. In middle type, components of OOXML and ODF cannot find the corresponding part directly or use different XML structures to represent them. However, the most content can find counterpart from logical level, for example page layout. In difficult type, components are very difficult to convert or even cannot be converted at all, because of the different design idea or incapability of descriptions used in OOXML and ODF (like change tracking and collaboration support).

2.2 XML Document Data Extraction

Many methods has been produced for extraction of information from MS word documents that has been created by OOXML format. There are various ways for extracting the metadata from XML documents and all of the methods have their own advantages and disadvantages.

These methods include Java XML parser, XPath Queries, DOM, DTD, SAX, XSD, and XSLT. In Figure 1, the timeline of XML and Semantic Web technology development is explained.

A method is proposed by (Kwok and Nguyen, 2006) for extracting data automatically from an electronic contract composed of a number of documents in PDF format. Their approach comprises of an administrator module, a PDF parser, a pattern recognition engine and a contract data extraction engine. This type of system is useful for extracting contract data using data mining.

He et al. (2013) build a system for evaluating XPath Queries in a user-friendly manner. They developed a prototype system named VXPath, which is a visual XPath query evaluator that allows the user to evaluate an XPath query by clicking the nodes in an expanding tree instead of typing the whole XPath query by hand. Their system supports various XPath axes, including child, descendant, self, parent, ancestor, following-sibling, preceding sibling, predicate and so on, and instead of loading the whole XML document into memory, they extract a concise data synopsis termed structural summary from the original XML document to avoid the loading overhead for of large XML documents.

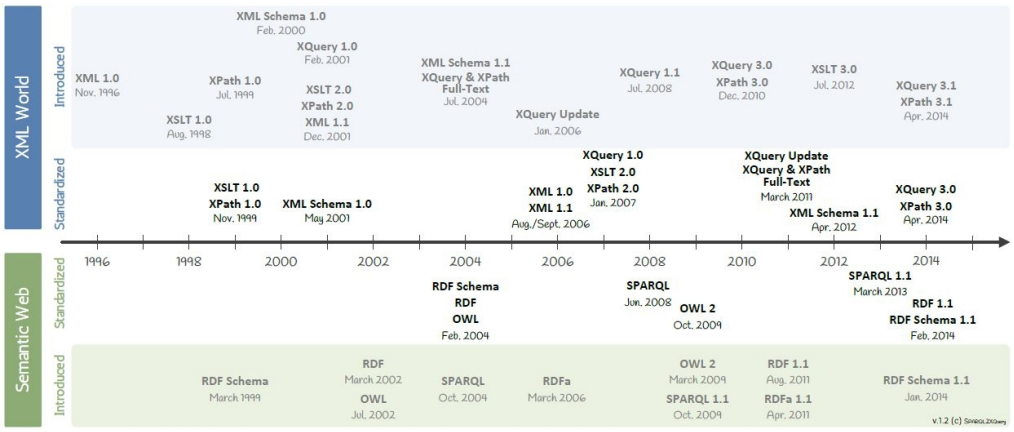


Figure 1: Time line of XML, Semantic Web and W3C standards[2]

Pellet and Chevalier (2014) develop a method for automatic extraction of formal properties of Microsoft Word, Excel, and Power point documents saved in OOXML format for

[2] http://www.dblab.ntua.gr/~bikakis/XMLSemanticWebW3CTimeline.pdf Retrieved 06 Aug, 2015

educational purpose. Their method was developed by Scala programming language for automatically extracting and inspecting XML structure of the document for word-processing-based entrance examination. Then, they report the result of a case study comparing manual and automatic evaluation. The results show that the automatic correction yields equal or more accurate results than the manual evaluation. In their approach they use Scala which is a concise, statically typed language that runs on the Java Virtual Machine (JVM). However Scala has language-level support for XML. And they test on a technical entrance examination, and not the format of document. Their work is mainly for comparing metadata extraction in both manual and automatic ways. Whereas in this thesis, manual and automatic document format checking system are compared and we do not compare manually and automatically extracted metadata.

In this thesis the same idea of (Xu et al., 2010) and (Pellet and Chevalier, 2014) for data extraction was used. However instead of using Java XML parser or Scala, we proposed a new method for data extraction. In particular, in our approach, the depth of hierarchy and inheritance in OOXML file format and the extracted metadata will not be compared with defined format for document. Instead, the extracted metadata in RDF is processed automatically and compared with set of semantic rules by using Semantic Web technologies. The advantages of using RDF metadata and semantic rules in our approach allows: (1) Data interoperability; once the metadata is converted in a common RDF representation, it is easy to incorporate new datasets, new attributes and aggregate disparate data sources. (2) Re-usability; once the extracted data from OOXML is converted to an RDF format using an ontology, any set of rules can be used to support reasoning. Thus, we can apply our system to other domains by changing ontology and semantic inference rules.

DocBook DTD is a unified vocabulary for describing documentations between companies, which is defined by SGML. DocBook was originally designed to enable the interchange of documentation between organizations. Şah and Wade (2010) propose a new framework for extracting metadata from a multilingual enterprise content by utilizing different document parsing algorithms in order to extract rich metadata form multilingual enterprise and using developed ontologies for DocBook. The framework was evaluated on English, German and French version of the Semantic Norton 360 knowledge base with an average precision of

89.39% accuracy on metadata value of document difficulty, document interactivity level and document interactivity type.

2.3 Conversion from XML to RDF

In researches of (Bosch and Mathiak, 2011) and (Jieping and Zhaohua, 2010), transformations from XML to derived ontology are proposed. Bosch and Mathiak (2011) describes a new approach of implementing a general transformation of any XML Schema for generating ontologies automatically by using XLST method. They declare that in most of cases the declaration of terminologies and syntactic structures of domain data model are already described in the form of XML Schema. Jieping and Zhaohua (2010) on the other hand, define a mapping formalism to convert the XML data to the ontology by using XSD and XPath expression method.

Milicka and Burget (2013) tries to describe the modeling of web documents based on semantic ontologies and present four level of document descriptions where all descriptions are based on ontology that represent different level of knowledge. Their proposed model of ontologies are (1) Box Model Ontology, where a Box is defined as a base element and the whole process starts with document rendering. The output of rendering is called a box model of the document and it basically describes the positions of the individual pieces of the document content on the resulting page and their visual features. (2) Segmentation Ontology, where the segmentation ontology represents the individual visually distinguished segments of the document contents in the page. (3) Semantic Ontology; this level of document description defines the parts of content with a specific role in the document. The semantic ontology processing is based on the segmented document (e.g. SALT ontology), and (4) Domain Ontology is defined for a particular application domain of the published information. For the documents from the given domain, the individual parts of the document that are described using rendering, segmentation and semantic ontologies may be assigned to some concepts of the domain ontology (like FOAF Ontology).

In another research proposed by (Bakkas et al., 2014), a semantic mapping from DTD documents to ontologies are proposed. This approach is characterized by its simplicity and generated classes can be instantiated at data level.

Deursen et al. (2008) proposes a generic approach for the transformation of XML data into RDF instances in an ontology dependent way. They try to obtain RDF instances of the OWL ontology, based on the XML data. A generic XMLtoRDF tool was proposed which takes XML data, an OWL ontology, and a mapping document as input. This mapping document describes the link between the XML Schema (describing the structure of the XML data) and the OWL ontology. The results of the XMLtoRDF tool are RDF instances based on the XML data, compliant with the OWL ontology.

Another research has been proposed by (Tian et al., 2009) for solving the office document processing complexity, by analyzing the characteristics of document structure. They present two methods for intelligent processing for MS office documents based on ontology. They declare that the logic content node should be recognized from non-content node, by using DOM and XPath technologies. Finally they describe building an ontology for UOF (Uniform Office Format), of Chinese office document standards, and define Semantic Web Rule Language (SWRL) for this ontology. UOF is a standard file format for Chines Office Document standard but OOXML is standardized format of MS Office by (ECMA-376, 2012) and (ISO/IEC-29500, 2012). And in this thesis the ontology will be built for OOXML not UOF.

In this thesis, the main aim of building the ontology is to capture the structure of ACM SIG document. We are not capturing the structure of OOXML of the document. In our approach, OOXML metadata of the document will be unzipped in a directory unlike (Hu et al., 2012) which proposes a method for querying the XML data in RDBMS. Subsequently in our work data is converted into RDF (in N3 format) based on ACM SIG ontology for semantic processing and reasoning. In case of building ontology based on OOXML for document format checking system, there will be the lack of inconsistency between ontology and instance data (Tian et al., 2009) and (Hou et al., 2010).

CHAPTER 3

THE SEMANTIC WEB

Since the invention of WWW (also known as WEB) by Tim Berners-Lee in 1989 and it becomes the most successful and widely used hypertext system of interconnected documents around the world, it intend for human to share the information. It undertake human friendly data format (HTML) and universal Internet protocol (http, ftp). However the Web lacks from semantics and automated processing; machine cannot understand the meaning of content that is represented by HTML, and HTML cannot be automatically shared among applications. The overcome the limitation of the web Tim Berners-Lee introduced the Semantic Web, which is an extension of the Web to enable such information to be understandable by machines by using Semantic Web technologies (e.g. RDF, Ontologies, SPARQL, Reasoner). By using Semantic Web technologies data can be accessed and processed automatically as well as shared across applications.

3.1 The Semantic Web Architecture

The architecture of semantic web is illustrated in the Figure 2. The first layer, URI and Unicode, follows the important features of the existing WWW. Unicode is a standard of encoding international character sets and it allows that all human languages can be used (written and read) on the web using one standardized form. URI is a string of a standardized form that allows to uniquely identify resources (e.g., documents). A subset of URI is Uniform Resource Locator (URL), which contains access mechanism and a (network) location of a document - such as http://www.semanticdoc.org/. The usage of URI is important for a distributed internet system as it provides understandable identification of all resources. An international variant to URI is Internationalized Resource Identifier (IRI) that allows usage of Unicode characters in identifier and for which a mapping to URI is defined.

The Semantic Web extends the existing Web, adding a multitude of language standards and software components to give humans and machines direct access to data. The Semantic Web is used for data publishing, querying and reasoning. The Semantic Web is rooted in a set of language specifications which represent a common infrastructure upon which applications can be built.

The Semantic Web Stack, also known as Semantic Web Cake or Semantic Web Layer Cake, illustrates the architecture of the Semantic Web.

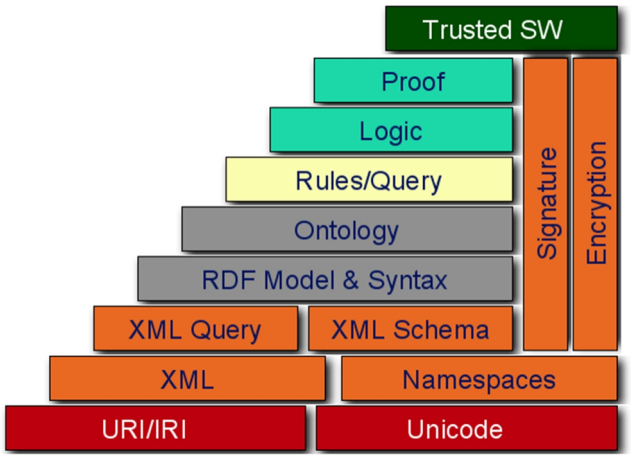


Figure 2: Semantic Web Stack[3]

Given the decentralized nature of the Semantic Web, data publishers require a way to refer to resources unambiguously. Resources on the Internet are identified with Uniform Resource Identifiers (URIs). URIs on both the Web and the Semantic Web typically use identifiers based on HTTP, which allows for piggybacking on the Domain Name System (DNS) to ensure the global uniqueness of domain names and hence URIs. The URL is an implicit mechanism for retrieving the content of document on web. The Namespace of an element, is the scope within which, it is valid. An XML namespace is a collection of names, identified by a URI reference. Names from XML namespaces may appear as qualified names, which contain a single colon, separating the name into a prefix and a local part. The prefix, which is mapped to a URI reference, selects a namespace.

One of the key goals of the Semantic Web technologies is to provide machines with machine-processable data; this allows intelligent understanding and usage of data. To this end, an increasing number of Web sites publish data using Semantic Web standards in standards defined by the World Wide Web Consortium (W3C). Given a wider availability of quality semantic data, applications can leverage this rich data and can provide elaborate services to their users.

[3] http://www.w3.org/2004/Talks/1117-sb-gartnerWS/sw_stack.png Retrieved 22 Aug, 2015

3.2 Extensible Markup Language (XML)

The ability to point to resources unambiguously and dereference them is a first step. Next, a language is required to exchange description of resources. For this purpose The Extensible Markup Language (XML) can be used. Where XML provides means for specifying and serializing structured documents which can be parsed by different software system across various operating systems.

3.3 Resource Description Framework (RDF)

RDF is closely related to semantic networks. Like semantic networks, it is a graph-based data model with labeled nodes and directed, labeled edges. This is a very flexible model for representing data. The fundamental unit of RDF is the statement, which corresponds to an edge in the graph. An RDF statement has three components: a subject, a predicate, and an object. These statements are often referred as triples. Since each statement must be composed of three elements; subject, predicate and object. The subject is the source of the edge and must be a resource. In RDF, a resource can be anything that is uniquely identifiable via a URI. More often than not, this identifier is a URL, which is a special case of URI. However, URIs are more general than URLs. In particular, there is no requirement that a URI can be used to locate a document on the Internet. The object of a statement is the target of the edge. Like the subject, it can be a resource identified by a URI, but it can alternatively be a literal value like a string or a number. The predicate of a statement determines what kind of relationship holds between the subject and the object. It is too identified by a URI. An example RDF graph is shown in Figure 3. For instance, “http://www.semanticdoc.org/ontology/2015/v1.6.owl#Section” is the subject, “http://www.semanticdoc.org/ontology/2015/v1.6.owl#hasSubSection” is the predicate and “http://www.semanticdoc.org/ontology/2015/v1.6.owl#SubSection” is the object.

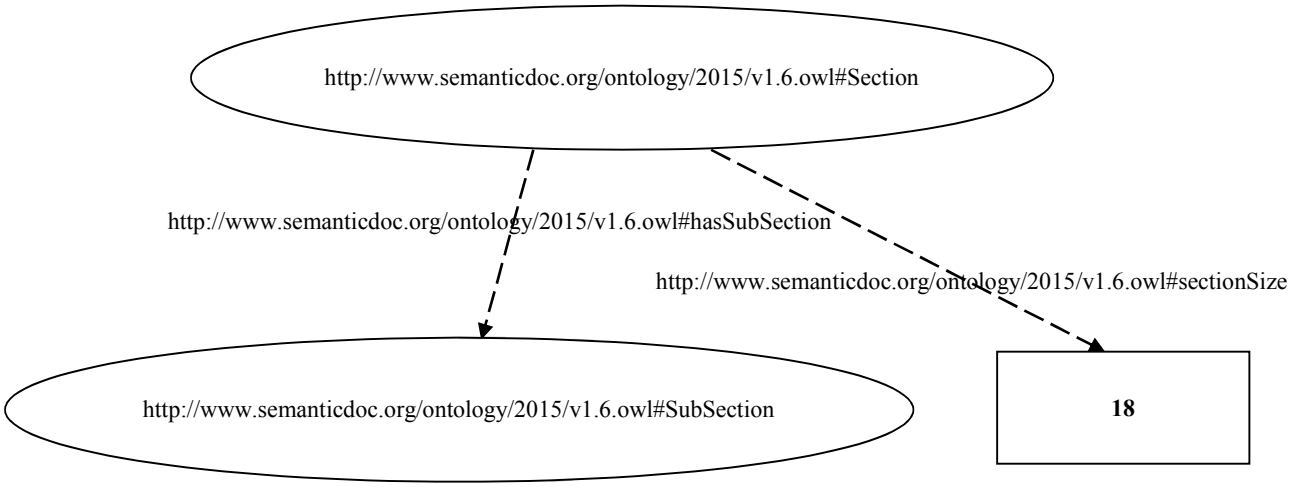


Figure 3: Representing a sample of RDF graph with fully qualified URIs

RDF can be serialized in a number of formats, such as Notation 3, Turtle, N-Triples, RDF/XML and JSON. RDF/XML is the only standardized serialization of RDF. In section 5.3, we explain the Notation 3 format which is used in this work.

3.4 Resource Description Framework (RDFS)

By itself, RDF is just a data model; it does not have any significant semantics. RDF Schema is used to define a vocabulary for use in RDF models. In particular, it allows you to define classes so that resource type can be created and to define properties so that resources can have attributes and relationship to other resources. An important point is that an RDF Schema document is simply a set of RDF statements. However, RDF Schema provides a vocabulary for defining classes and properties. In particular, it includes rdfs:Class, rdf:Property (from the RDF namespace), rdfs:subClassOf, rdfs:subPropertyOf, rdfs:domain, and rdfs:range. It also include properties for documentation, including rdfs:label and rdfs:comment. One problem with RDF Schema is that it has very weak semantic primitives. This is one of the reasons for the development of the Wen Ontology language namely OWL. Each of the important RDF Schema terms are either included directly in OWL or are superseded by new OWL terms.

3.5 Ontology

In the emerging document engineering, there is an urgent need for improving the management and maintenance of documents. Because of the inexistence of a common

understanding of a domain for document sharing, this leads the development of common vocabularies. Thus documents can be shared and communicated across people and application. A formal ontology is a controlled vocabulary expressed in an ontology representation language for describing and representing the area of concern. XML can only describe the structure of the data rather than the meaning of the data, but ontology is distinguished by its power of semantic representation.

Ontologies have been represented in machine-readable format, so that it is possible to manipulate the data and check it's consistency with predefined types of domain. It is the declaration of a classification system with classes, sub-classes, taxonomies, definitions, properties, relationships and axioms that taken together specify a particular ontology.

The Web Ontology Language (OWL) is an international standard for encoding and exchanging ontologies and is designed to support the Semantic Web. The concept of the Semantic Web is that information should be given explicit meaning, so that machines can process it more intelligently. Instead of just creating standard terms for concepts as is done in XML, the Semantic Web also allows users to provide formal definitions for the standard terms they create. Machines can then use inference algorithms to reason about the terms.

A crucial component to the Semantic Web is the definition and use of ontologies. For over a decade, artificial intelligence researchers have studied the use of ontologies for sharing and reusing knowledge. Although there is some disagreement as to what comprises an ontology, most ontologies include a taxonomy of terms (e.g., stating that a Car is a Vehicle), and many ontology languages allow additional definitions using some type of logic. Guarino (1998) has defined an ontology as "a logical theory that accounts for the intended meaning of a formal vocabulary." A common feature in ontology languages is the ability to extend preexisting ontologies. Thus, users can customize ontologies to include domain specific information while retaining the interoperability benefits of sharing terminology where possible. In addition, ontology language allow automated inferences, i.e. drawing conclusions based on existing facts.

OWL is an ontology language for the Web. It became a World Wide Web Consortium (W3C) Recommendation in February 2004. As such, it was designed to be compatible with the eXtensible Markup Language (XML) as well as other W3C standards. In particular, OWL

extends the Resource Description Framework (RDF) and RDF Schema, two early Semantic Web standards endorsed by the W3C. Syntactically, an OWL ontology is a valid RDF document and as such also a well-formed XML document. This allows OWL to be processed by the wide range of XML and RDF tools already available (Mishra and Yagyasen, 2013).

Encoding data as graph covers only parts of the meaning of the data. Often, constructs to model class or property hierarchies provide machines and subsequently humans a more sapient understanding of data. To more comprehensively model a domain of interest, so-called ontology languages can be employed. RDF Schema (RDFS) is an ontology language which can be used to express for example class and property hierarchies as well as domain and range of properties. However, RDFS is not very expressive for representing complex semantics, such as complex cardinality and restriction rules. OWL ontology language facilitates greater machine readability of Web content than that supported by XML, RDF, and RDFS by providing additional vocabulary along with a formal semantics. For example, OWL allows specifying equality of resources or cardinality constraints of properties. The OWL is designed for use by applications that need to process the content of information instead of just presenting information to humans (Harth et al., 2011).

3.6 SPARQL Query

SPARQL Query Language is a declarative query language, similar to SQL in RDBMS, which allows for specifying a mechanism queries against integrated data and graphs in RDF. SPARQL queries are executed against RDF datasets, consisting of RDF graphs.

A SPARQL query comprises, in order:

1. Prefix declarations, for abbreviating URIs
2. Dataset definition, stating what RDF graph(s) are being queried
3. A result clause, identifying what information to return from the query
4. The query pattern, specifying what to query for in the underlying dataset
5. Query modifiers, slicing, ordering, and otherwise rearranging query results

The following example will illustrate the SPARQL query and returned results set.

Dataset:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
_:a     foaf:name       “Johnny Outlaw”.
_:a     foaf:email      <jlow@example.com>.
_:b     foaf:name       “Peter Goodguy”.
_:b     foaf:email      <peter@example.com>.
_:c     foaf:email      <carol@example.com>.
```

SPARQL Query:

```
PREFIX foaf:http://xmlns.com/foaf/0.1/      // “Prefix” keyword is used to define a prefix.
SELECT ?name ?email                         // for projecting results “select” keyword is used
WHERE                                       // clause for identifying what will be returned.
{
        ?x foaf:name ?name.
        ?x foaf:email ?email.
}
ORDER BY ?name                              // Query modifier
```

Query Result:

name	email
Johnny Outlaw	jlow@example.com
Peter Goodguy	peter@example.com

As shown in this example, the SPARQL query is executed against the dataset and retrieve the results based on the defined query pattern in WHERE clause.

CHAPTER 4

DOCUMENT FORMAT

In computer terminology, document file format can be described as a text, or binary data file type that are used to store formatted documents (texts, pictures, clipart, tables, charts, multiple pages, multiple documents etc.). The format of a document belongs to the overall layout of a document. For example, the formatting of text on many English documents is aligned to the left of a page. Today, there is a multitude of incompatible document file formats.

The most known document file extensions are used for documents created by Microsoft Office suite are DOC and DOCX for Microsoft Word document, XLS and XLSX for Microsoft Excel spreadsheets, and finally PPT and PPTX for Microsoft PowerPoint presentations.

By contrast, the default file formats in Office 2007 are based on Extensible Markup Language (XML). To denote the change in format, the filename extensions associated with each format have changed, adding an X at the end of each Word's new default Format. For example .docx instead of .doc. MS Office 2007 programs can still open and save files using the older formats, although some features new to MS Office 2007 will be lost in the conversion. Tables 1 gives a list of different file extensions for MS Office and Open Office suite.

Table 1: List of different file extensions for MS Office and Open Office for documents

Distributor	**Type**	**Extension**
MS Office	Document	.docx (FFM)
MS Office	Macro enabled document	.docm
MS Office	Template	.dotx
MS Office	Macro enabled template	.dotm
Open Office	ODF text document	.odf
Open Office	ODF text document template	.ott
Open Office	XML text document	.sxw
Open Office	XML text document Template	.stw

In addition to these new formats, MS Word will support opening and saving .doc and .dot files for backward compatibility, along with other options such as .htm files. MS Word automatically adds the .docx extension to every file saved in the default format.

Word 2013, Word 2010, Word 2007, and Word 2003 users will continue to experience interoperability. However, Word 2013's, 2010's, and 2007's "native" format is radically different and better than the old format. The new format boasts a number of improvements over the older format as discussed below.

Open format: The basic file is in ZIP format, an open standard, which serves as a container for .docx and .docm files. Additionally, many (but not all) components are in XML format (Extensible Markup Language). Microsoft makes the full specifications available free, and they may be used by anyone royalty-free. In time, this should improve and expand interoperability with products from software publishers other than Microsoft.

Compression: The ZIP format is compressed, resulting in files that are much smaller. Additionally, Word's "binary" format has been mostly abandoned (some components, such as VBA macros, are still written in binary format), resulting in files that ultimately resolve to plain text and that are much smaller.

Robustness: ZIP and XML are industry-standard formats with precise specifications that offer fewer opportunities to introduce document corruption. Hence, the frequency of corrupted Word files should be greatly reduced.

Backward-compatibility: Though MS Word 2013, 2010, and 2007 have slightly different formats, they still fully support the opening and saving of files in legacy formats. A user can opt to save all documents in an earlier format by default. Moreover, Microsoft makes available a Compatibility Pack that enables MS Word 2000–2003 users to open and save in the new format. In fact, MS Word 2000-2003 users can make the .docx format their default, providing considerable interoperability among users of the different versions.

Extensions: MS Word 2013 has four native file formats: .docx (ordinary documents), .docm (macro-enabled documents), .dotx (templates that cannot contain macros), and .dotm (templates that are macro-enabled, such as Normal.dotm).

Calling the x-file format "XML format" actually is a bit of a misnomer which is not in XML format but some of the components of Word's x files, do use XML format. XML is at the heart of Words x format; however, the files saved by Word are not XML files. And it can be verify this by trying to open one using Internet Explorer.

A last look at the .docx file structure reveals clues about why it is different from the older .doc format. As indicated earlier, Words new .docx format does not itself use XML format. Rather, the main body of your document is stored in XML format, but that file is not stored directly on disk. Instead, it is stored inside a ZIP file, which gets a .docx, .docm, .dotm, or .dotx file extension.

To verify this, you can create a simple Word 2013 file, and save and close it. Next, in Windows Explorer (Windows 7) or File Explorer (Windows 8), display file name extensions and change the file's extension to .zip. Finally, the double click the file to display the contents of that ZIP file.

MS Office Word .docx files can contain additional folders as well, such as one named customXml. This folder is used if the document contains content control features that are linked to document properties, an external database or forms server. The main parts of the MS Office Word document are inside the folder named "word". The main text of the document is stored in document.xml. Using an XML editor you could actually make changes to the text in document.xml, replace the original file with the changed one, rename the file so that it has a .docx extension instead of .zip, and open the file in Word, and those changes would appear. More complex Word files contain additional elements, such as clip art, an embedded Excel chart, several pictures, and some SmartArt, as well as custom XML links to document properties.

4.1 What is Metadata?

Metadata is a difficult term to define - it means many things to so many different audiences, and sometimes metainformation which is 'data about data', of any sort in any media." Within any domain, the term metadata can be more usefully defined by describing its agreed use – social sciences research has a well-developed metadata culture, which allows us to be very specific. Researchers understand what data are – the data sets which are collected, processed,

analyzed and used in the conduct of research. Metadata is all the documentation about that data.

4.2 Office Open XML (OOXML) File Format

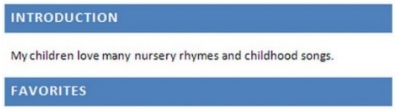
Office Open XML, also known as Open XML or OOXML, is an XML-based format for office documents, including word processing documents, spreadsheets, presentations, as well as charts, diagrams, shapes, and other graphical material. The specification was developed by Microsoft and adopted by ECMA International as ECMA-376 in 2006. A second version was released in December, 2008, and a third version of the standard released in June, 2011, and the fourth version of the standard released in December 2012. The specification has been adopted by ISO and IEC as ISO/IEC 29500. (ECMA-376 and ISO/IEC, 2012)

ECMA-376 includes three different specifications for each of the three main office document types Word processing ML for word processing documents, Spreadsheet ML for spreadsheet documents, and Presentation ML for presentation documents. It also includes some supporting markup languages, most importantly Drawing ML for drawings, shapes and charts.

Although the older binary formats (.doc, xls, and .ppt) continue to be supported by Microsoft, OOXML is now the default format of all Microsoft Office documents (.docx, .xlsx, and .pptx). Example OOXML tag structure are shown in Table2.

Table 2: OOXML close and open tag structure, with typeface view

Tag meaning/typeface view	OOXML File Format
Root element and namespace declarations	<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <w:document xmlns:ve="http://schemas.openxmlformats.org/markup-compatibility/2006" xmlns:o="urn:schemas-microsoft-com:office:office" xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships" xmlns:m="http://schemas.openxmlformats.org/officeDocument/2006/math" xmlns:v="urn:schemas-microsoft-com:vml" xmlns:wp="http://schemas.openxmlformats.org/drawingml/2006/wordprocessingDrawing" xmlns:w10="urn:schemas-microsoft-com:office:word" xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2006/main" xmlns:wne="http://schemas.microsoft.com/office/word/2006/wordml">

	<w:body> <w:p> <w:pPr> <w:pStyle w:val="Heading1"/> </w:pPr> <w:r><w:t>Introduction</w:t></w:r> </w:p> <w:p> <w:r><w:t xml:space="preserve">My children love many nursery rhymes and childhood songs. </w:t></w:r> </w:p> <w:p> <w:pPr> <w:pStyle w:val="Heading1"/> </w:pPr> <w:r><w:t>Favorites</w:t></w:r> </w:p>
Section properties and closing tags	<w:sectPr> <w:footerReference w:type="default" r:id="rId7"/> <w:pgSz w:w="12240" w:h="15840"/> <w:pgMar w:top="1440" w:right="1440" w:bottom="1440" w:left="1440" w:header="720" w:footer="720" w:gutter="0"/> <w:cols w:space="720"/> <w:docGrid w:linePitch="360"/> </w:sectPr> </w:body> </w:document>

4.3 Open Document File Format

Open Document Format (ODF) is an international family of standards that is the successor of commonly used deprecated vendor specific document formats such as .doc, .wpd, .xls and .rtf. ODF is standardized at OASIS (Organization for the Advancement of Structured Information Standards). ODF is not software, but a universal method for storing and processing information that transcends specific applications and providers. ODF is not only more flexible and efficient than its predecessors, but also future proof. Public sector, business and cultural content must not be lost if a supplier decides to no longer support legacy file formats, while other software cannot deal with those files. With ODF you avoid that risk: it is an international standard actively supported by multiple applications, and it can be safely implemented in any type of software, including open source software - such as is common on the majority of mobile phones and tablets these days. The societal importance of the move to ODF is therefore considerable.

In ODF the way for storing documents does not determine the software you work with. Files in the Open Document Format (ODF) are platform independent and do not rely on any

specific piece of software. Every software maker can implement without having to pay royalties. Although technically behind the scenes all Office applications now use the same ISO-standardized format, for the convenience of new users it was chosen to use separate names for the different applications - just like they are used to. You recognize these by their "extensions": .odt (text) .ods (for spreadsheets), .odp (for presentations), and so on. Example ODF tag structures are shown in Table 3.

Table 3: ODF close and open tag structure, with typeface view

Tag meaning/typeface view	ODF File Format
Root element and namespace declarations	<office:document-content xmlns:office="urn:oasis:names:tc:opendocument:xmlns:office:1.0" xmlns:style="urn:oasis:names:tc:opendocument:xmlns:style:1.0" xmlns:text="urn:oasis:names:tc:opendocument:xmlns:text:1.0" xmlns:table="urn:oasis:names:tc:opendocument:xmlns:table:1.0" xmlns:draw="urn:oasis:names:tc:opendocument:xmlns:drawing:1.0" xmlns:fo="urn:oasis:names:tc:opendocument:xmlns:xsl-fo-compatible:1.0" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:meta="urn:oasis:names:tc:opendocument:xmlns:meta:1.0" xmlns:number="urn:oasis:names:tc:opendocument:xmlns:datastyle:1.0" xmlns:svg="urn:oasis:names:tc:opendocument:xmlns:svg-compatible:1.0" xmlns:chart="urn:oasis:names:tc:opendocument:xmlns:chart:1.0" xmlns:dr3d="urn:oasis:names:tc:opendocument:xmlns:dr3d:1.0" xmlns:math="http://www.w3.org/1998/Math/MathML" xmlns:form="urn:oasis:names:tc:opendocument:xmlns:form:1.0" xmlns:script="urn:oasis:names:tc:opendocument:xmlns:script:1.0" xmlns:ooo="http://openoffice.org/2004/office" xmlns:ooow="http://openoffice.org/2004/writer" xmlns:oooc="http://openoffice.org/2004/calc" xmlns:dom="http://www.w3.org/2001/xml-events" xmlns:xforms="http://www.w3.org/2002/xforms" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:rpt="http://openoffice.org/2005/report" xmlns:of="urn:oasis:names:tc:opendocument:xmlns:of:1.2" xmlns:xhtml="http://www.w3.org/1999/xhtml" xmlns:grddl="http://www.w3.org/2003/g/data-view#" xmlns:tableooo="http://openoffice.org/2009/table" xmlns:textooo="http://openoffice.org/2013/office" xmlns:field="urn:openoffice:names:experimental:ooo-ms-interop:xmlns:field:1.0" office:version="1.2"> <office:scripts/>
Introduction My children love many nursery rhymes and childhood songs. **Favorites**	<office:font-face-decls> <style:font-face style:name="Mangal1" svg:font-family="Mangal"/> <style:font-face style:name="Times New Roman" svg:font-family="'Times New Roman'" style:font-family-generic="roman" style:font-pitch="variable"/> <style:font-face style:name="Arial" svg:font-family="Arial" style:font-family-generic="swiss" style:font-pitch="variable"/> <style:font-face style:name="Mangal" svg:font-family="Mangal"

	style:font-family-generic="system" style:font-pitch="variable"/> <style:font-face style:name="Microsoft YaHei" svg:font-family="'Microsoft YaHei'" style:font-family- generic="system" style:font-pitch="variable"/> <style:font-face style:name="SimSun" svg:font-family="SimSun" style:font-family-generic="system" style:font-pitch="variable"/> </office:font-face-decls> <office:automatic-styles/> <office:body> <office:text> <text:sequence-decls> <text:sequence-decl text:display-outline-level="0" text:name="Illustration"/> <text:sequence-decl text:display-outline-level="0" text:name="Table"/> <text:sequence-decl text:display-outline-level="0" text:name="Text"/> <text:sequence-decl text:display-outline-level="0" text:name="Drawing"/> </text:sequence-decls> <text:h text:style-name="Heading_20_1" text:outline- level="1">Introduction</text:h> <text:p text:style-name="Standard"> My children love many nursery rhymes and childhood songs. </text:p> <text:h text:style-name="Heading_20_1" text:outline- level="1">Favorites</text:h> </office:text> </office:body> </office:document-content>

4.4 Discussion of OOXML and ODF

ISO is a worldwide network of national standards institutes from 157 country. It has a present arrangement of more than 17,000 standards for Business, Government and Society. ISO's Standards make up a complete offering for each of the three measurements of sustainable development, economic, environmental and social. Founded on 23 February 1947, the organization promotes worldwide proprietary, Industrial and Commercial standards. ISO has framed joint boards of trustees with the International Electro-technical Commission (IEC) to develop standards and terminology in the areas of Electrical, Electronic and related technologies.

The question is why OOXML and ODF standards for document are important? We probably do not lose anything that our word processor is saving documents in the wrong format. We may have some old files that do not open correctly, or somebody may have sent you a spreadsheet that does not work in anything except than Excel, however we most likely discovered some approach to work around the issue. In any case, when information is vital

and should be utilized in different ways or archived for a long time, the format really does matter. It all comes down to one question, who is the owner of data? If the data can be used in a wide variety of applications, we own it. If it can only be used cleanly with one vendor's applications, that vendor is really the one with control.

The Open Document Format (ISO/IEC-26300, 2006) is an XML format intended to exchange office document data. Initially developed by Sun Microsystems, it has been reviewed and developed by OASIS (Organization for the Advancement of Structured Information Standards) since 2002. ODF was consistently approved as an ISO standard on May 3, 2006. The ODF detail is a bit more than 700 pages long, was made by an open process that included different sellers, and has been implemented in a variety of products, including Open Office, KOffice, GoogleDocs, IBM Lotus Symphony, and Macintosh TextEdit. The ODF standard was the only existing ISO standard for office document data at that time.

Microsoft has been using XML in some file formats since 2000, and they provided full support for exporting office data to XML in Microsoft Office 2003. These XML formats were designed by Microsoft for the exchange of Microsoft Office data. Office Open XML (OOXML) is a further development of the formats used in Microsoft Office 2003. OOXML is not only complex, it cannot be completely implemented without access to inside information. Although its specification is more than 6,000 pages long, it contains various references to things that are defined only in Microsoft's software, not in the specification itself. ODF is a smaller and simpler specification than Microsoft's OOXML. ODF was designed to represent office documents; OOXML was designed to represent Microsoft Office applications.

Microsoft submitted OOXML to ECMA International, in November 2005 in an effort to fast-track, then Microsoft attempts to officially standardize the Office Open XML Format (OOXML) by the ISO-Standards ISO/IEC DIS 29500, the representatives from six countries Brazil, South Africa, Venezuela, Ecuador, Paraguay, and Cuba have written an open letter to the ISO and IEC criticizing the handling of the OOXML appeals. The OOXML fast-track process and subsequent approval vote was riddled with complaints that Microsoft acted deceitfully. Finally The Office Open XML file formats were standardized between December 2006 and November 2008, first by the ECMA International consortium (ECMA-

376, 2012), and subsequently, after a contentious standardization process, by the ISO/IEC (ISO/IEC-29500, 2012).

Now ODF and OOXML both are open document formats that are meant to be used in cross-platform and cross-suite environments. ISO voted in ODF as an international document standard in 2006. ISO also voted in OOXML as an international document standard in 2008.

As the best office software, MS Office Professional has every application that any user will need to create, edit, send, publish, manage and document in one office software suite. However 57.67%[4] of all users for Microsoft operating system use windows 7 which support OOXML file format of MS Office. The new MS office productivity software includes a few new features and a simplified interface allowing users the ability to create documents, spreadsheets and presentations.

For decades, Microsoft Office has been the leader in office software which more than 1.2 billion people use MS Office[5]. MS Office impresses now more than ever. The design update is the largest for the office software giant since the redesign for its 2007 launch, and with the redesign come new features across all of its applications[6]. These features, make MS Office to be unique among all other office suite and most of the people around the world use MS Office for managing their documents. For this reason, we decided to use OOXML in our work.

4.5 Metadata and XML Based Technologies

One of the biggest developments in the growth of the Internet - and for distributed computing generally was the advent of the eXtensible Markup Language (XML), and the suite of related technologies and standards. Derived from a technology standard for marking up print documents the Standard Generalized Markup Language (SGML). The original focus of XML was to better describe documents of all sorts, so they could be used more effectively by applications discovering them on the Internet.

[4] https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=10&qpcustomd=0 Retrieved 9 Sep 2015.
[5] http://news.microsoft.com/bythenumbers/index.HTML retrieved 9 Sep 2015
[6] http://office-software-review.toptenreviews.com/microsoft-office-review.html retrieved 9 Sep 2015

XML is a meta-language used to describe tag-sets, effectively injecting additional information into a document. Unlike HTML (which was also based on SGML), however, there was no fixed list of tags – the whole point is that documents could be designed to carry specific additional information about their contents. Thus, XML document types could be designed to carry any sort of metadata, in-line with the contents of the document.

XML is not only a language but also a collection of technologies available to perform various operations on the underlying data or metadata: XML schema, for describing document structure; XPath and XQuery for querying and searching XML; SOAP (Simple Object Access Protocol) or REST (Representational State Transfer) to facilitate the exchange of information and many others.

CHAPTER 5

DATA EXTRACTION AND DOCUMENT FORMAT CHECKING

In our work, in order to extraction metadata from ACM SIG documents, first we need to access OOXML format of the documents. To achieve this, first the document is unzipped and then the content of document which is in OOXML format with metadata is converted to RDF (N3 format) using the developed ontology. Finally, using a set of reasoning rules, the validity of the document format is automatically checked by the proposed ADFCS framework. In this chapter, we discuss; (1) ACM SIG document structure and OOXML analysis in Section 5.1. (2) Then, we explain the developed ACM SIG Ontology for metadata extraction in Section 5.2. (3) ADFCS and the metadata extraction process is summarized in Section 5.3. Jena reasoning rules and the format checking procedure is discussed in Section 5.4

5.1 ACM SIG Document Structure

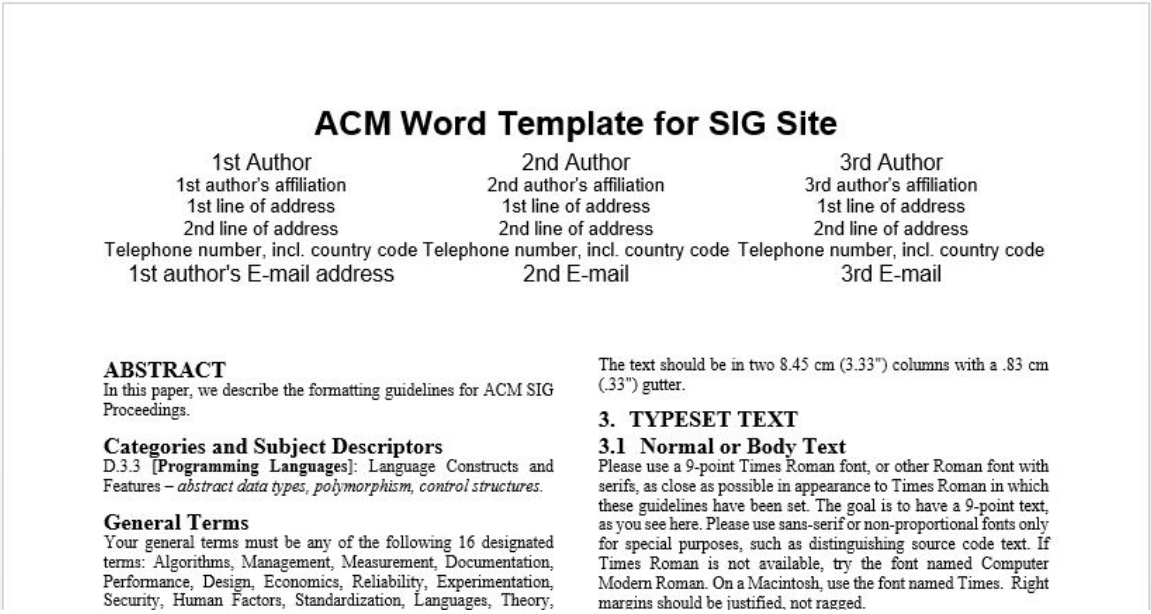
According to the ACM SIG word template from SIG Website[7], any type of ACM SIG document can be categorized by three main parts for data extraction.

- Title (the title of ACM SIG document in one column with style).
- Author (the Author(s) in one, two or three column with style).
- Body (main text of the document in two column with style).

Each of these three parts of the document may contain any type of data but the main structure and format is fixed and cannot be changed. There must be a continuous section break between each part of ACM SIG document in typeface to let the ADFCS to be able to distinct parts between different parts. Any type of ACM SIG document that will be published in ACM sponsored conference or journal has a style similar to Figure 4; researchers just replace their desired text with template text. At the end, the style and structure of all ACM SIG documents are the same, but with different material. The structure of ACM SIG documents comes as sequences, and each part has a specific type of format. For example, the first paragraph in the main body of text which is in two columns, start with Abstract, Category

[7] ACM SIG Website is available at http://www.acm.org

and Subject Descriptor, General Terms and Keywords. Then the sections start with the Introduction and end with the References. Each paragraph[8] in any part of ACM SIG document has, its own format and some paragraphs headlines like Abstract, Keywords, etc., it must be written as same as ACM SIG template with right format. In Figure 4, the standard format for paragraph ABSTRACT are (Times New Roman as font Style, Bold, Font Size 12 pt. and alignment as Left) and for paragraph (In this paper, we describe …) the standards format is (Times New Roman, Font Size 9 pt. and alignment as Justify).



ACM Word Template for SIG Site

1st Author
1st author's affiliation
1st line of address
2nd line of address
Telephone number, incl. country code
1st author's E-mail address

2nd Author
2nd author's affiliation
1st line of address
2nd line of address
Telephone number, incl. country code
2nd E-mail

3rd Author
3rd author's affiliation
1st line of address
2nd line of address
Telephone number, incl. country code
3rd E-mail

ABSTRACT
In this paper, we describe the formatting guidelines for ACM SIG Proceedings.

Categories and Subject Descriptors
D.3.3 [**Programming Languages**]: Language Constructs and Features – *abstract data types, polymorphism, control structures.*

General Terms
Your general terms must be any of the following 16 designated terms: Algorithms, Management, Measurement, Documentation, Performance, Design, Economics, Reliability, Experimentation, Security, Human Factors, Standardization, Languages, Theory,

The text should be in two 8.45 cm (3.33") columns with a .83 cm (.33") gutter.

3. TYPESET TEXT

3.1 Normal or Body Text
Please use a 9-point Times Roman font, or other Roman font with serifs, as close as possible in appearance to Times Roman in which these guidelines have been set. The goal is to have a 9-point text, as you see here. Please use sans-serif or non-proportional fonts only for special purposes, such as distinguishing source code text. If Times Roman is not available, try the font named Computer Modern Roman. On a Macintosh, use the font named Times. Right margins should be justified, not ragged.

Figure 4: ACM SIG word template for SIG site[9]

Each document which has been created by MS word that support OOXML include information about main content, page layout, header, footer, etc. To extract metadata from OOXML format of the ACM SIG documents, first the document is unzipped. Each document if created in MS word 2007 and upper version is a zip file and the content of the document can be extracted easily just by opening in a zip file reader or renaming the extension of the file from .docx to .zip file extension. By extracting the content of MS Office word document, the content will appear as similar to the Figure 5.

[8] The paragraph in MS word in typeface can be selected by triple click on desired text inside document.
[9] https://www.acm.org/sigs/publications/pubform.doc Retrieved 18 Mar, 2015.

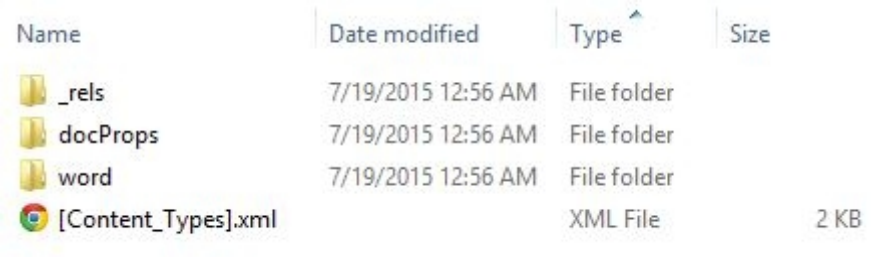


Figure 5: The content of extracted MS word document

In Figure 5, the main root directory of extracted document is shown. The content of this directory is related to the metadata of the document. For example the word folder in Figure 5 contains the original text and the style of the document. The folder docProps contain the properties of the document, like author, date, etc. If the original document contain some figure and clipart's, the figures will be in a folder with the "extra" name.

In OOXML file format, a document is only a logical document or a container which are integrated into a zip package. In ODF there are two ways to compose a document, one is to use one XML file and the second is to use several files. If several files were used to compose a logical document in ODF, four physical file (content.xml, meta.xml, settings.xml and styles.xml) are generated for any type of document, while for MS word document in OOXML, different files and part are used to compose different types of document. Both ODF and OOXML are standards for word processing and the physical content of zip packages for word processing of ODF and OOXML formats and correlation between them has been compared by (Hou et al., 2010).

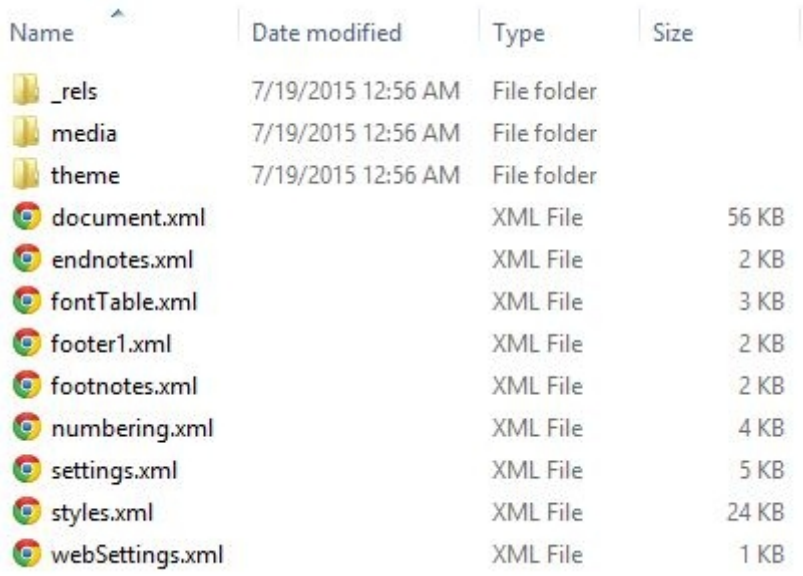


Figure 6: The content of word directory /word/

The content of /word/ directory is shown in the Figure 6. According to the ECMA standards for OOXML, there are different files in this directory and each one represents a specific metadata of the document. The most important metadata documents in this directory are document.xml and styles.xml which include all the text and related metadata about text inside the document. All these metadata documents of OOXML are related to each other. A small part of document.xml file is shown in the Figure 7, which is in OOXML format.

```
▼<w:body>
  ▼<w:p w:rsidR="008B197E" w:rsidRDefault="008B197E">
    ▼<w:pPr>
       <w:pStyle w:val="Paper-Title"/>
       <w:spacing w:after="60"/>
     </w:pPr>
     <w:bookmarkStart w:id="0" w:name="_GoBack"/>
     <w:bookmarkEnd w:id="0"/>
    ▼<w:r>
       <w:t>ACM Word Template for SIG Site</w:t>
     </w:r>
   </w:p>
```

Figure 7: A sample content of document.xml file

As shown in Figure 7 which is the metadata of Figure 4, style related metadata of document.xml is located in another file named style.xml which contains that how the text will appear to the user (see Figure 8). Both document.xml and style.xml files are related to

each other by a resource identifier named “Paper-Title”. We use these unique identifiers to track related information from different OOXML files in order to retrieve other metadata about the text inside the document. The content of the “Paper-Title” part is shown in Figure 7. Relationship in OOXML format is a kind of connection between a source part and a target part in a package. Relationships make the connections between different metadata files directly discoverable without looking at the content in the parts, and without altering the parts themselves.

```
▼<w:style w:type="paragraph" w:customStyle="1" w:styleId="Paper-Title">
   <w:name w:val="Paper-Title"/>
   <w:basedOn w:val="Normal"/>
  ▼<w:pPr>
     <w:spacing w:after="120"/>
     <w:jc w:val="center"/>
   </w:pPr>
  ▼<w:rPr>
     <w:rFonts w:ascii="Helvetica" w:hAnsi="Helvetica"/>
     <w:b/>
     <w:sz w:val="36"/>
   </w:rPr>
 </w:style>
```

Figure 8: A sample content of style.xml file

As shown the style format for the text “ACM Word Template for SIG Site” in Figure 4 have some format (like, font size, font style, alignment and bold). ECMA-376 specifies a family of XML schemas, collectively called Office Open XML, which define the XML vocabularies for word processing, spreadsheet, and presentation documents, as well as the packaging of documents that conform to these schemas. It also specifies requirements for OOXML consumers and producers and the goal is to facilitate extensibility and interoperability by enabling implementations by multiple vendors and on multiple platforms.

Each resource in OOXML is defined in ECMA-376 and relationship between that parts of unzipped document. The resources in OOXML are related by a unique identifier to the other parts. Beside this every element in OOXML has been defined in ECMA, for example <w:b/> declare that the formatting paragraph for the text is bold, <w:sz w:val=”36”> declare the size of formatting paragraph of the text is 18. (ECMA-376, 2012)

Most of the children of an element have a single val attribute that is limited to a specific set of values. For example, the b (bold) element causes the text that follows it to be bold when

the b element has a val attribute with value 1. If the val attribute is not present for the b element, it defaults to "1". Therefore, <w:b/> is equivalent to <w:b w:val="1"/>.

5.2 ACM SIG Ontology

For semantic processing, and checking the format and structure of ACM SIG documents, a new ontology has been developed, by using protégé application. In order to create this ontology, we analyzed the ACM SIG document structure and OOXML metadata files very carefully. After understanding ACM SIG formatting rules and related OOXML elements, we generated the ACM SIG ontology, which corresponds to various parts of ACM SIG document structure as we describe below.

In ACM SIG ontology (Appendix 3), the main aim is to build an ontology based on ACM SIG document structure and standards. Each class in ACM SIG ontology is related to a specific part in ACM SIG document. For example the class Abstract in ACM SIG ontology corresponds to Abstract Part of the ACM SIG document. The Abstract class also contains data/object properties and the asserted values are completely related to Abstract part of the ACM SIG document. We tried to define all of the required data type and object type properties to the all of the created classes in ACM SIG ontology, by carefully investigating ACM SIG documents and OOXML metadata. During the metadata extraction, the ADFCS extracts metadata from OOXML files and convert it to an RDF format. Some features of ACM SIG document are not included in ACM SIG ontology, because of the difficulty of extracting the same equivalent element in ACM SIG document. For example the number of columns, and the equality of columns length on last page cannot be automatically extracted thus we did not model these in the ontology. In addition, Based on ACM SIG document standard, the main text of document must be in two columns. However any figure and table in ACM SIG document, may extend across both columns to a maximum width of page size. In this case the number of column will be one column, and will conflict with ACM SIG ontology. In summary, we analyzed the whole standards while creating ACM SIG ontology.

In total, we created 9 classes in ACM SIG ontology with 7 object property and 67 data type property. All data type property are scattered between classes with their own asserted value.

For example abstractSize as data type property, is a member of Abstract class in ACM SIG ontology with asserted value "18"[10].

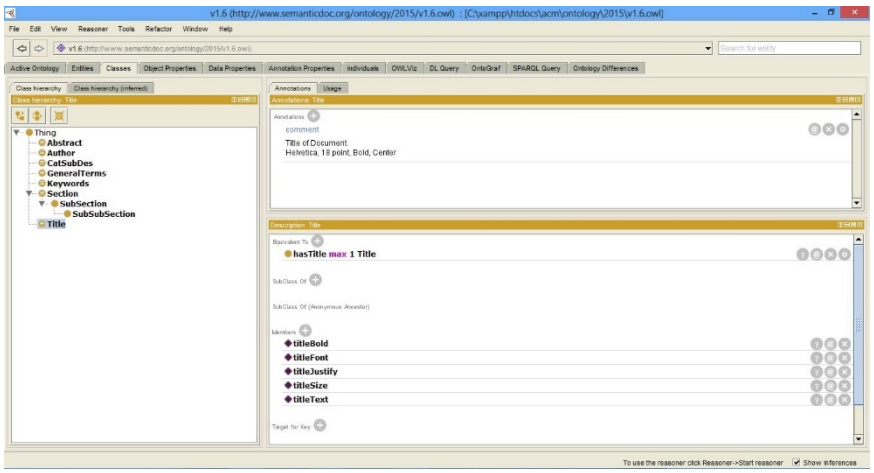


Figure 9: ACM SIG ontology created by protégé

5.3 The Proposed Framework, ADFCS, for Metadata Extraction

According to (ECMA-376, 2012) a Word processing markup language package's main part starts with a word processing root element (<w:body>). That element contains a body, which, in turn, contains one or more paragraphs (as well as tables, pictures, etc.). A paragraph (<w:p>) contains one or more runs, where a run (<w:r>) is a container for one or more pieces of text having the same set of properties. Like many elements that defined a logical piece of a word processing document, each run and paragraph can have associated set of properties. For example, a run might have the property bold, which indicates that the run's text is to be displayed as bold in a typeface.

Each paragraph (<w:p>) has its own close tag (</w:p>) which indicate the start and end of a paragraph tag, and in typeface for selecting the paragraph triple click will be used. In typeface between paragraphs section break can be used when the main subject of text change and also in OOXML, this is represented with *w:sectPr* element. The *w:sectPr* element are used by the ADFCS system to distinct the title, author and main body of ACM SIG documents.

[10] In OOXML a positive measurement, specified in half-point (ECMA-376, 2012).

All the process for viewing the metadata of OOXML document can be done manually, just by opening the document in zip package file reader or renaming the file extension to .zip. In automating process, this process is applied after the document has been uploaded, the ADFCS renames the document file and unzip the content in a new created directory.

In the metadata extraction level, after unzipping the content of the document, the ADFCS starts to capture the text and format of the document. The ADFCS is written in Java programming language which will read the content of /word/document.xml and /word/style.xml files of unzipped documents. These files contain the main text and format (ECMA-376, 2012) belongs to ACM SIG uploaded document. The ADFCS will read the files line by line by using BufferedReader class in Java, and will capture each paragraph start tag to end tag and this process will be continued till the end of the file. The next step is to detect and capture the format. By looking at Figure 7 it shown that the first paragraph starts with *w:p* element and end with the same close tag. The paragraph has text which is inside *w:t* tag and style which is related to this text is available in /word/style.xml with a reference value. Table 4 shows a sample of extracted metadata of document which is the same as typeface. In our work, the list of captured tags of unzipped document by ADFCS are *w:sectPr*, *w:pgSz*, *w:pgMar*, *w:jc*, *w:rFonts*, *w:spacing w:b*, *w:i*, *w:sz*, *w:pStyle*, *w:t*, *w:p*, *w:r*, *w:style*, *w:name*, *w:basedOn*, *w:styleId*, *w:next*, and *w:docDefaults*.

Table 4: OOXML elements definition and its effect in typeface (ISO/IEC-29500, 2012) and (ECMA-376, 2012)

Element	Element definition/typeface effect
w:sectPr	This element defines the section properties for a section of the document.
w:pgSz	This element specifies the properties (size and orientation) for all pages in the current section.
w:pgMar	This element specifies the page margins for all pages in this section.
w:jc	This element specifies the paragraph alignment which shall be applied to text in this paragraph.
w:rFonts	This element specifies the fonts which shall be used to display the text contents of this run. Within a single run, there can be up to four types of content present which shall each be allowed to use a unique font: • ASCII (i.e., the first 128 Unicode code points) • High ANSI

	• Complex Script • East Asian The use of each of these fonts shall be determined by the Unicode character values of the run content, unless manually overridden via use of the cs element.
w:spacing	This element specifies the inter-line and inter-paragraph spacing which shall be applied to the contents of this paragraph when it is displayed by a consumer.
w:b	This element specifies whether the bold property shall be applied to all non-complex script characters in the contents of this run when displayed in a document.
w:i	This element specifies whether the italic property should be applied to all non-complex script characters in the contents of this run when displayed in a document.
w:sz	This element specifies the font size which shall be applied to all non-complex script characters in the contents of this run when displayed. The font sizes specified by this element's val attribute are expressed as half-point values.
w:pStyle	This element specifies the style ID of the paragraph style which shall be used to format the contents of this paragraph.
w:t	This element specifies that this run contains literal text which shall be displayed in the document.
w:p	This element specifies a paragraph of content in the document.
w:r	A paragraph contains one or more runs, where a *run* is a container for one or more pieces of *text* having the same set of properties. This element specifies a run of content in the parent field, hyperlink, custom XML element, structured document tag, smart tag, or paragraph.
w:style	One relationship from the document part specifies the document's styles. A *style* defines a text display format. A style can have properties, which can be applied to individual paragraphs or runs. Styles make runs more compact by reducing the number of repeated definitions and properties, and the amount of work required to make changes to the document's appearance. With styles, the appearance of all the pieces of text that share a common style can be changed in one place, in that style's definition. This element specifies the definition of a single style within a WordprocessingML document. A *style* is a predefined set of table, numbering, paragraph, and/or character properties which can be applied to regions.
w:name	This element specifies the primary name for the current style in the document. This name can be used in an application's user interface as desired. The actual primary name for this style is stored in its val attribute.
w:basedOn	This element specifies the style ID of the parent style from which this style inherits in the style inheritance. The *style inheritance*

	refers to a set of styles which inherit from one another to produce the resulting set of properties for a single style. The val attribute of this element specifies the styleId attribute for the parent style in the style inheritance.
w:styleId	Specifies a unique identifier for the parent style definition. This identifier shall be used in multiple contexts to uniquely reference this style definition within the document.
w:next	This element specifies the style which shall automatically be applied to a new paragraph created following a paragraph with the parent paragraph style applied.
w:docDefaults	This element specifies the set of default paragraph and run properties which shall be applied to every paragraph and run in the current WordprocessingML document. These properties are applied first in the style hierarchy; therefore they are superseded by any further conflicting formatting, but apply if no further formatting is present.

Each of these tags end with the same close tag and has been defined in OOXML which represents a specific format of text in the document. As it is clear that the first paragraph in Figure 4, and its equivalent metadata in Figure 7 and 8, the first output of ADFCS after reading the metadata of documents is shown below.

```
<w:spacing w:after=”120”/>
<w:jc w:val=”center”/>
<w:rFont w:ascii=”Helvetica” w:hAnsi=”Helvetica”/>
<w:b/>
<w:sz w:val=”36”/>
<w:spacing w:after=”60”/>
<w:t>ACM Word Template for SIG Site</w:t>
```

Figure 10: ADFCS output after reading the first w:p tag

As shown in ADFCS first output, only the last two elements is available in *w:p* tag in document.xml file, and the other elements have been extracted from style.xml related sections. However the section style in style.xml Figure 8, also inherit some elements from Normal section style which are not listed in ADFCS output because, all the elements has been overridden. If the element in sections is not listed the children will inherit the element from parent section. The final operation of ADFCS for first paragraph is arranging and replacing the element with new one. Table 5 shows the last properties for first paragraph in ACM SIG document.

Table 5: A sample of ADFCS extracted metadata of document

Element	Value
Paragraph Text	ACM Word Template for Sig Site
Paragraph space after	60
Paragraph Justify	Center
Paragraph Font	Helvetica
Paragraph Bold	Enabled
Paragraph Size	36

The paragraph size is 36 which is specifies a positive measurement specified in half-points will be 18 (half-point 36) which is equals to the size of text in typeface (ECMA-376, p307, 2012).

5.4 Notation 3 File Format

After the whole metadata that has been extracted it needs to be formalized in way to be an input for processing semantically and this cannot be done unless by converting into RDF model. Notation 3 (N3) is a non-XML and with a human readability in mind, Notation 3 also known for serialization of Resource Description Framework (RDF). The ADFCS will write the extracted metadata in a N3 file format in a new text file.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:<http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix sig: <http://www.semanticdoc.org/ontology/2015/v1.6.owl#> .
# Title
<acm_9648456_title> <sig:titleText> "ACM Word Template for SIG Site".
<acm_9648456_title> <sig:titleFont> "Helvetica".
<acm_9648456_title> <sig:titleJust> "Center".
<acm_9648456_title> <sig:titleSize> "36".
<acm_9648456_title> <sig:titleBold> "bold".
```

Figure 11: A sample of Notation 3 file of extracted document, rewrite again by ADFCS

The N3 file contain the metadata of document in RDF model. Each document has a separate N3 file which contain all the metadata that has been extracted from document. The next process of ADFCS will be populating the N3 file to ACM SIG ontology for its consistency.

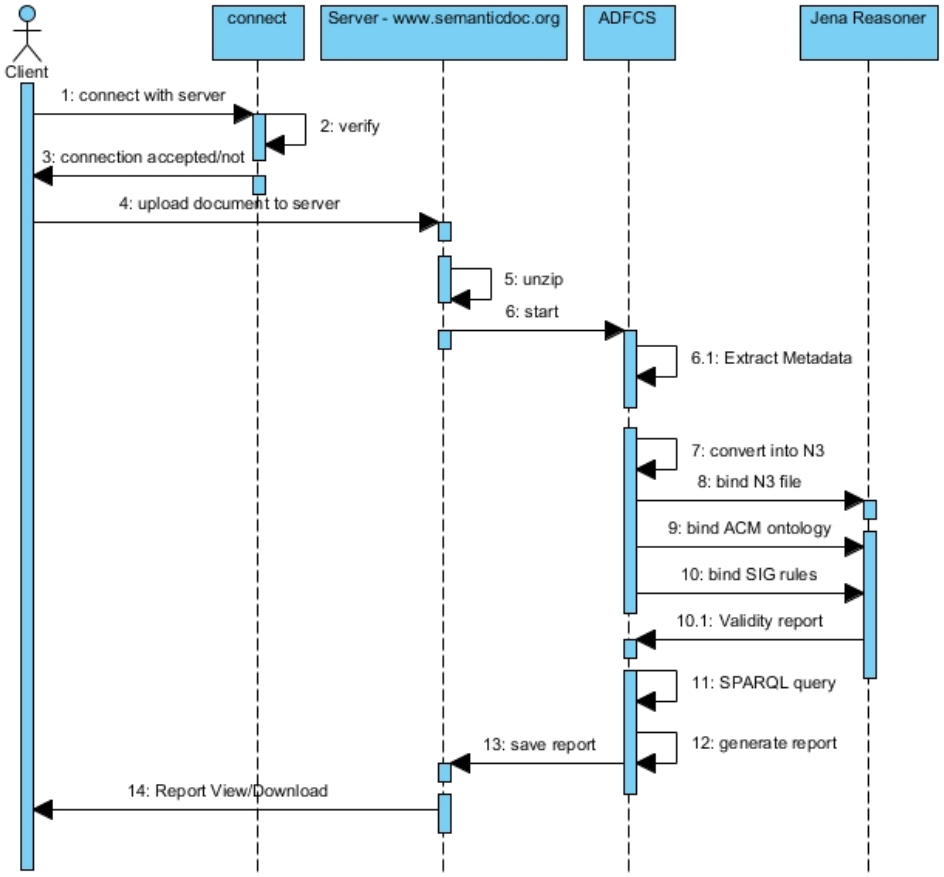


Figure 12: Client server sequence diagram for ADFCS System

The Figure 12 illustrate the sequence of action which will be taken for processing the ACM SIG document format checking. The diagram is useful when the system fail for checking the ACM SIG document. For example if a client upload a MS word version 2003, the system will fail at unzipping level, and the system will not generate the report.

5.5 Reasoning and Rules

Reasoning in ontologies and knowledge bases are one of the reasons why a specification needs to be formal. By reasoning we mean deriving facts that are not expressed in ontology or in knowledge base explicitly. Reasoning is required when a program must determine some information or some action that has not been explicitly told about. It must be figured out what it needs to know from what it already known.

A Reasoner is a key component for working with OWL ontologies. In fact, virtually all querying of an OWL ontology should be done using a reasoner. This is because knowledge

in an ontology might not be explicit and a Reasoner is required to deduce implicit knowledge so that the correct query results are obtained. The OWL API includes various interfaces for accessing OWL Reasoners. In order to access a Reasoner via the API a Reasoner implementation is needed. There following Reasoners provide implementations of the OWL API: FaCT++, JFact, Jena, HermiT, Pellet, RacerPro.

5.5.1 Jena reasoning

Jena is a java based frameworks for building semantic web application. The Jena API support Reasoner to be plugged into the Jena for performing the reasoning task and the process of deriving additional information. The Jena support the meaning of specialized Reasoner by binding it to an ontology, beside this the specialized reasoned can be attached into a set of instance data.

The intent of choosing the Jena Reasoner for ADFCS between all the available Reasoner is that, the Jena is an open source improved by Apache Software Foundation (ASF) and it is written in java environment, which will make no requirements for integrating between ADFCS and Jena (Appendix 6). Jena support OWL reasoning and rule based Reasoner that will make the working with RDF and semantic data easier. Beside this Jena can be used for converting RDF data to another format of data type. The available format of semantic RDF which Jena support are: Turtle, N3, RDF, RDF/XML,

Reasoning in ontologies are divided into two main part inferencing and querying. Inference also is consist of two part, inference and inference rule. In inference implicit data will be inferred to the Reasoner based on inference rule and the explicit data will be produced. Finally the produced new fact can be queried for retrieving the data from Reasoner.

Inference is an act or process of constructing new expressions from existing expressions, or the result of such an act or process. In RDF, inferences corresponding to entailments that are described as correct or valid. Inference rule is a formal description of a type of inference; inference system, organized system of inference rules; also, software which generates inferences or checks inferences for validity.

5.6 Jena Rules for Document Checking

For ACM SIG document checking, the T-Box (Schema Model) is ACM SIG ontology and the A-Box (Instance Data) is N3 file of the document. Moreover the Jena support a rule based Reasoner with user defined rules. Figure 13 shows some Jena rule for ACM SIG document, the complete Jena rule for ACM SIG document is available at (appendix 2). By this rule the new fact can be produced. For example the titleFont of document which exist in N3 file if the value equal to *Helvetica* so that the new fact will be validTitleFont. All the new data will be added to Jena Reasoner and its need to query the Jena Reasoner for new triples. Beside this the Reasoner will add these new triple based on ACM SIG ontology if the triple passes all the constraint and condition that has been defined.

```
@prefix sig: <http://www.semanticdoc.org/ontology/2015/v1.6.owl#> .
# Rules
[rule1: (?f sig:titleText ?d) -> (?p sig:validTitleText "true")]
[rule2: (?d sig:titleFont "Helvetica") -> (?p sig:validTitleFont "true")]
[rule3: (?f sig:titleSize "36") -> (?p sig:validTitleSize "true")]
[rule4: (?f sig:titleBold "bold") -> (?p sig:validTitleBold "true")]
```

Figure 13: Jena rules for ACM SIG site documents

In Jena, the number of defined rules for ACM SIG document must be balanced between the system performance and the requirements for satisfying the standards of ACM SIG document. Each statement in Jena rule indicates a specific standard for ACM SIG documents. Moreover some standards needs more than one rule. In the experiment of defining the rules for ACM SIG documents, we realized that the number of defined rules is a key factor for ADFCS to successfully check the format of an ACM SIG document. By defining more rules and with respect to the requirements of ACM SIG the system will lose its performance.

As shown in Figure 14, by increasing the number of rules, the time that is needed for running rules by Jena will increase considerably. In particular the minimum amount of rules, cannot satisfy all of the required standards for checking the format of ACM SIG document. In our work, the ADFCS has been tested with different number of rules, and finally we reach to number of 112 rules for stability between the number of Jena rules and the performance of ADFCS.

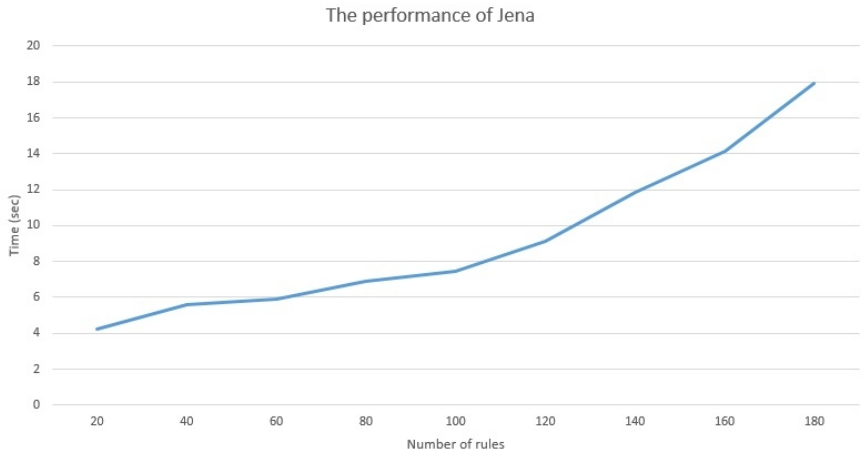


Figure 14: The average performance of Jena with respect to the number of defined rules for ADFCS System

In our Jena model, all rules for ACM SIG document checking has been defined as forward chaining rules using the forward chaining engine. In forward chaining rule, the new fact is derived from the existing facts. However, first the correctness of the left-hand-side of the rule is checked. As an example, whenever the size of a document title is 18 point, so we can derive that the font size of document title adhere the ACM SIG document standard for size of title.

In Jena API, an ontology model is an extension of the Jena RDF model, providing extra capabilities for handling ontologies. Ontology models are created through the Jena ModelFactory. In Jena, a graph is called a model and is represented by the Model interface.

Model model = ModelFactory.createDefaultModel();

It creates an empty Model, by using the ModelFactory method createDefaultModel() to create a memory-based model. Jena contains other implementations of the Model interface. An important feature of Jena is support for different kinds of inference over RDF-based models (RDFS and OWL). Inference models are constructed by applying Reasoners to base models.

In our work, we build three models (1) data model which handles the N3 file, (2) ontology model which handle the ACM SIG ontology and (3) a Generic Rule Reasoner for handling

the ACM SIG rules. Finally all models and the Reasoner are binded to an Inference Model for processing (Appendix 6).

Jena's inference machinery defines some specialized services that are not exposed through the addition of extra triples to the model. These are exposed by the InfModel interface;

InfModel infmodel = ModelFactory.createInfModel(reasoner, ontology, data);

Inference models will add many additional statements to a given model, including the axioms appropriate to the ontology language. At the end, SPARQL queries are used for querying the added triples to the model as true value (valid) and the other triples will be as invalid. Finally, the output of SPARQL queries is converted into a report which contains all information about valid/invalid ACM SIG formats.

```
String queryString = ""
+ "prefix sig: <http://www.semanticdoc.org/ontology/2015/v1.6.owl#> "
+ "SELECT ?sub ?pre "
+ "WHERE"
+ "{"
+ "     ?sub ?pre <http://www.semanticdoc.org/ontology/2015/v1.6.owl#true> ."
+ "}"
+ "ORDER BY ?pre";
```

Figure 15: A SPARQL query for retrieving the newly added triples by the rule engine

5.7 Jena Query Result

After querying the Jena for retrieving the new triples that have been produced, it needs to be rearranged, for better viewing and customization. Moreover the N3 triples have not been designed to be presented to human such as shown in Figure 16. So, the report must be shown to the user in a user friendly format.

```
http://www.semanticdoc.org/ontology/2015/v1.6.owl#titleBold
http://www.semanticdoc.org/ontology/2015/v1.6.owl#validTitleBold
http://www.semanticdoc.org/ontology/2015/v1.6.owl#titleFont
http://www.semanticdoc.org/ontology/2015/v1.6.owl#validTitleFont
http://www.semanticdoc.org/ontology/2015/v1.6.owl#titleJustify
http://www.semanticdoc.org/ontology/2015/v1.6.owl#validTitleJustify
http://www.semanticdoc.org/ontology/2015/v1.6.owl#titleSize
http://www.semanticdoc.org/ontology/2015/v1.6.owl#validTitleSize
http://www.semanticdoc.org/ontology/2015/v1.6.owl#titleText
http://www.semanticdoc.org/ontology/2015/v1.6.owl#validTitleText
```

Figure 16: A sample of SPARQL query result before generating report

The output of SPARQL query against Jena inference model is shown in Figure 16. The odd lines related to ?sub variable and even lines related to ?pre variable in SPARQL query in Figure 15 . In fact the Jena in most case will produce a validity report to check whether the newly added facts are correct and satisfy the RDF discipline or not. Then the SPARQL queries are used to select the desired new data that has been added.

By first looking at Figure 16, it will be very difficult to understand which new fact has been produced, so the ADFCS will convert the result of SPARQL query to a report to let the user to understand which part of document is formatted according to ACM SIG standard and which part has incorrect format. The result of SPARQL query in Figure 15 declares that the data of the document is formatted correctly. However, if the font of title in Figure 4 was *Tahoma*, thus in data extraction process of ADFCS, produce titleFont as *Tahoma* instead of *Helvetica*, and after binding the N3 file to Jena, the new fact validTitleFont will not be produced because it is not consistent with titleFont in ACM SIG ontology, and the titleFont in ACM SIG ontology is defined as individual, String as data type property and *Helvetica* as value.

For converting the output of SPARQL query to have a better view for user, the ADFCS will split the records of query results (e.g. validTitleFont) and then by replacing all capital letters with the same capital letter and one more extra space (valid Title Font), and at the end the report will be generated as (Title Font valid).

CHAPTER 6

SYSTEM IMPLEMENTATION

For implementing the ADFCS, a new website has been built (which is available online at http://semanticdoc.org address) by using PHP; ADFCS software has been installed to this online address. As shown in Figure 18 the input for ADFCS system is a document, where the document is checked for the correctness of its format and structure. For this purpose, a PHP form was used to help the user to upload the document and the PHP code will execute[11] the ADFCS in order to start processing. The function will run the ADFCS in terminal and will read the output of ADFCS and show the result to the user.

6.1 Home Page

The home page of website can be accessed by visiting http://www.semanticdoc.org/ as shown in Figure 17.

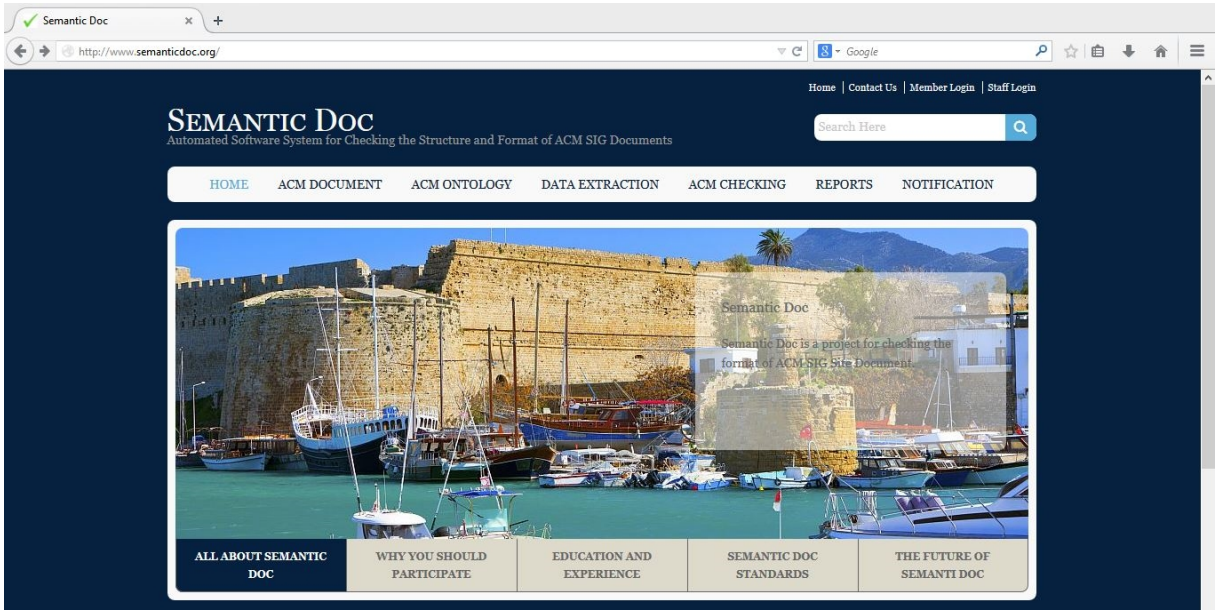


Figure 17: The home page of semanticdoc.org, with the installed ADFCS

6.2 ACM Document Checking Process

For checking the format and structure of ACM SIG documents, the user must follow these steps:

[11] The function is exec("java -Xmx256M java_class_file " , file_name).

- Open http://www.semanticdoc.org in browser
- Click “ACM Checking” in menu bar in Website as shown in Figure 18.
- From the upload form, upload an ACM SIG document. (see Figure 19 and 20)
- Click check.
- If everything goes well and the document is formed well (i.e. in OOXML format), a report is generated (see Figure 21).

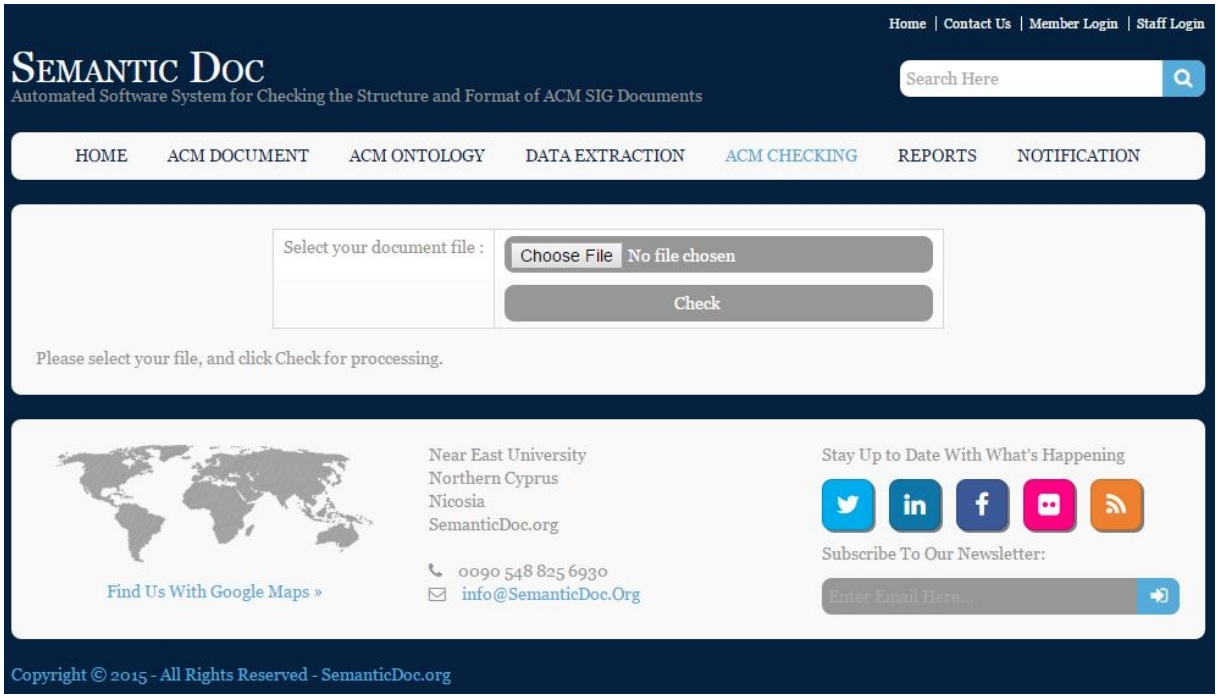


Figure 18: The upload page of semanticdoc.org for uploading the ACM SIG document

The Figure 18 shows the upload form, which allows the user to upload a document for checking. The form has been built by simple HTML form which uploads the document to http://www.semanticdoc.org server and redirects to a report page by a hidden id with a value initialized by 1 for starting the checking process. If the user directly visit the report page this hidden id will not be set and will remain as default value 0. When the id value is 0 the page will inform the user to visit the checking process page for uploading the document (Appendix 4).

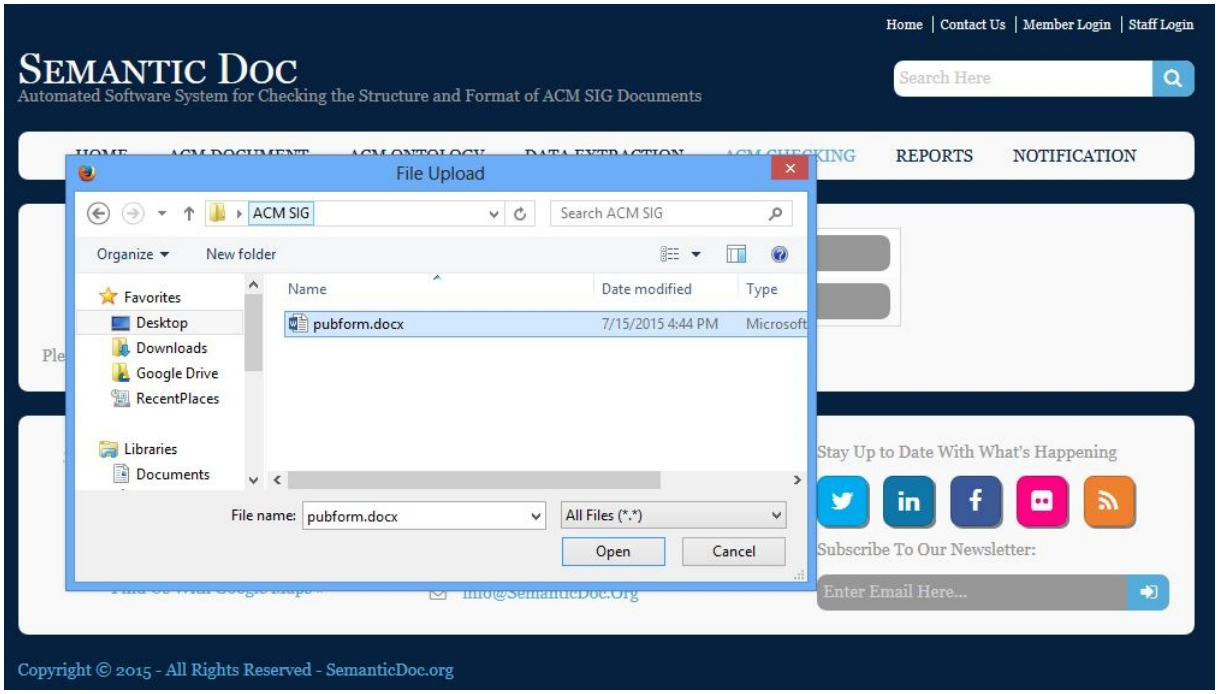


Figure 19: File selection menu for uploading an ACM SIG document

In file selection menu, the user must select the document and then click open. The selected document must be in OOXML format. The selected document must be created by MS Word 2007 and upper version, otherwise the OOXML will not be generated and the ADFCS will fail to extract the metadata from the document.

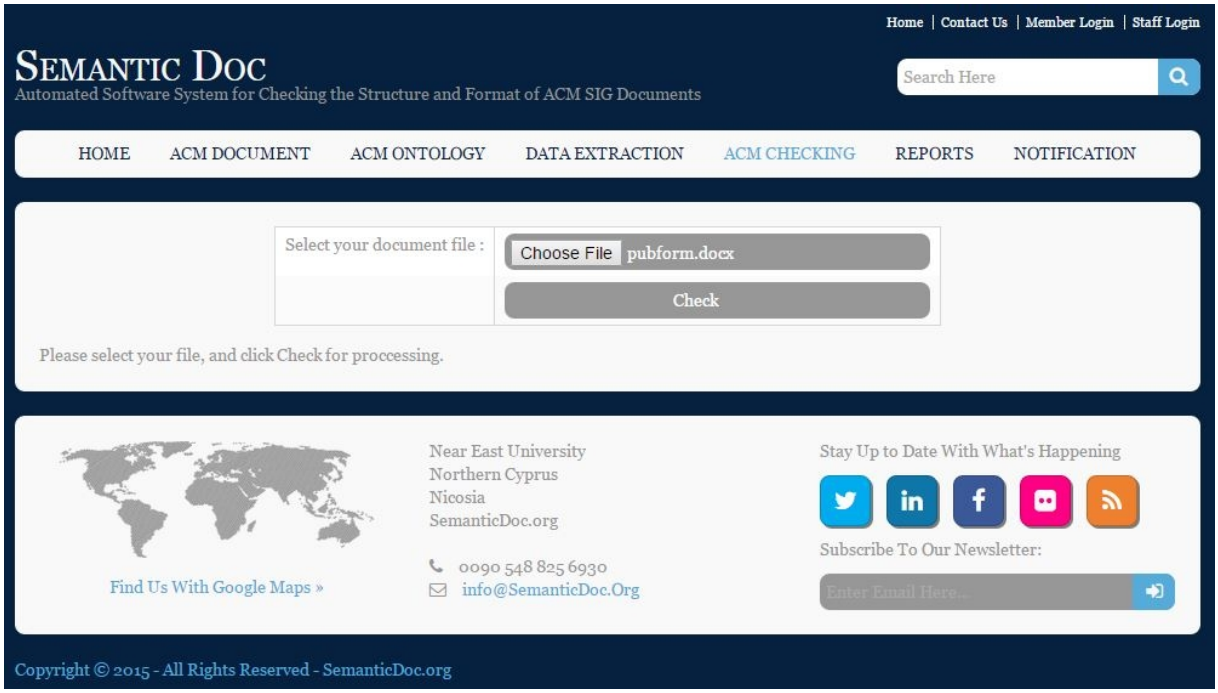


Figure 20: The upload page of semanticdoc.org after document has been chosen

For better understanding the process of checking a document's format, learning the checking process and how the user can download the generated report a demo video was generated and available[12].

6.3 Report View and Download

After checking process, the user has this ability to download the report for correcting the mistyped or incorrect format of the document based on ACM SIG document standards. The user can review the report and correct the wrong format of the document as shown in Figure 21.

[12] https://www.youtube.com/watch?v=rqXInqdx_vw

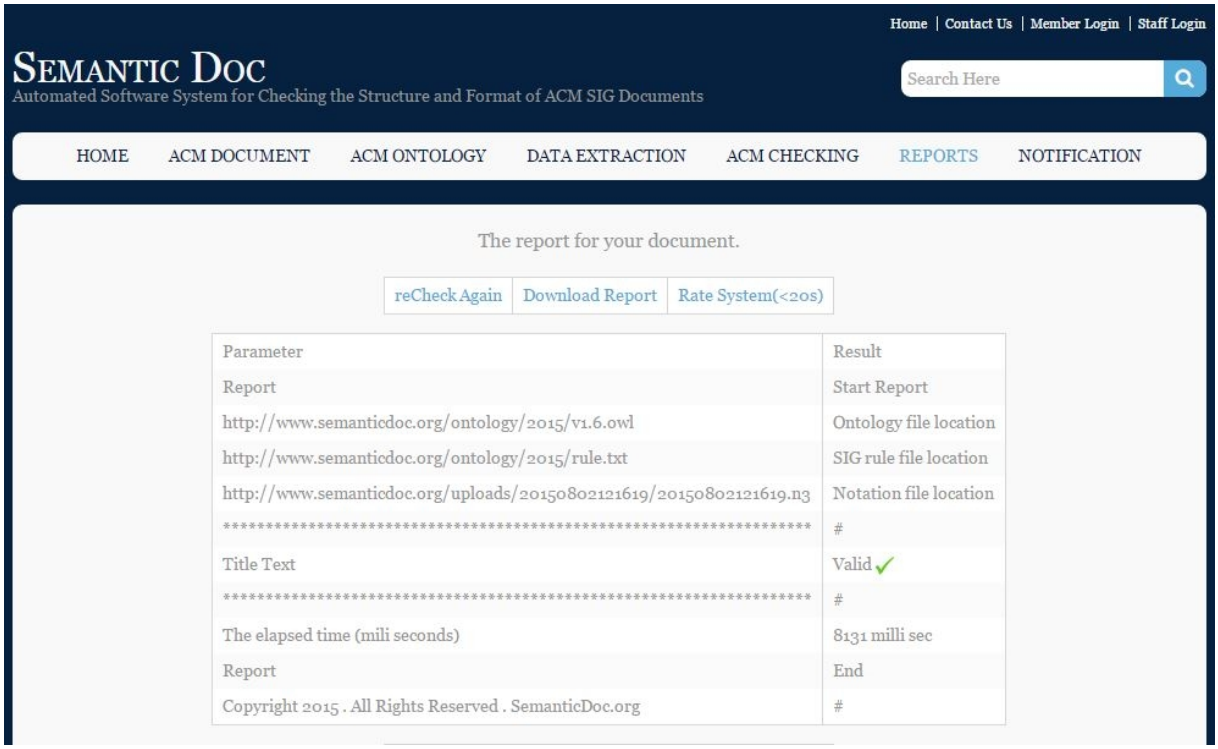


Figure 21: The report page; for viewing and downloading the generated report of the document

As shown in Figure 21, for better viewing, only one parameter is listed, the user needs to scroll down to view all format details. Also the address of the created N3 file, rule file and ACM SIG ontology file are listed in the report.

For better analyzing and referring to the incorrect format in the document, the complete manual with descriptions about the parameter meaning in the generated report is available in (Appendix 5).

6.4 System Rating

For future improvements, there is an online rating page which allows the user to rate the ADFCS System. The rating page is available at http://www.semanticdoc.org/acmnot.php. The rating will help us to improve our system in future updates.

CHAPTER 7

EVALUATIONS, RESULTS AND LESSONS LEARNED

In order to evaluate the proposed framework for checking the format of ACM SIG documents, three different types of experiments has been performed. The proposed experiments aim to evaluate in different situation; (i) the required time for checking the ACM SIG document with different number of pages and system performance with respect to various system properties, (ii) the ability to support user in completing the checking process and the usability of the system in user perspective, and finally (iii) a checking process with two different types as manual and automatic checking. In manual checking the user will act as a proofreader to find incorrect formats from the generated 15 wrong formats of an ACM SIG document and compare this process with the automatic format checking with the ADFCS system.

7.1 Time Evaluations

The performance of ADFCS system is highly depending on the size of document, number of rules and ACM SIG ontology. The number of rules which has been defined in Figure 13 plays a key factor for system performance. More rules need more time for checking the document format, and less rules will take less time for processing. The test has been completed with 112 defined rules for each document with different number of pages as illustrated in Figure 22.

The test was run on a personal computer with an operating system of Windows 8.1 on HP Hewllet-Packet with Intel(R) Core(TM) 4 CPUs 2.1GHz processor, 4096 MB RAM installed memory and 348 MB dedicated memory for java virtual machine.

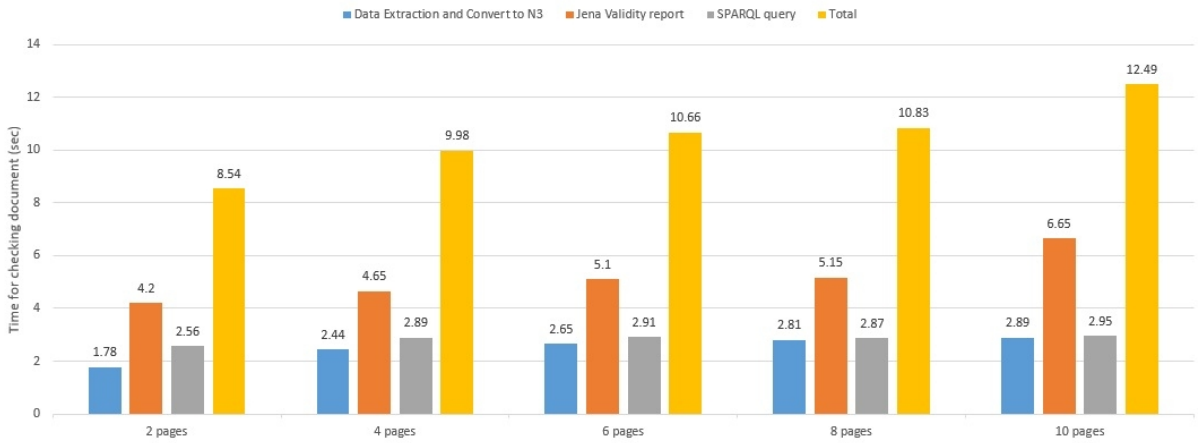


Figure 22: Average elapsed time of checking ACM SIG document with different page sizes

Figure 22 shows that in all cases, the total time for checking the document format belongs to the Jena Reasoner to produce a validity report; data extraction and querying takes comparably less time than Jena reasoning. Figure 22 illustrates the elapse time for (i) document upload and data extraction and conversion by ADDFCS system, (ii) the elapse time for Jena to produce the validity report, (iii) running the SPARQL query by ADFCS against Jena validity report and generating report, (iv) And the total elapse time of all operation from the beginning to generated report.

7.2 User Evaluations

In order to evaluate the proposed automated document format checking system for ACM SIG documents, a user study was performed. In particular, the proposed approach was evaluated in terms of (i) the ability to support users in completing format checking tasks (ii) comparing the user ability for finding and correcting incorrect formats in both Manual and Automatic checking and (ii) the usability from the users' perspective (i.e. user satisfaction). For this purpose, the ADFCS has been compared with manual format checking in order to assess how ADFCS help users to complete document format checking.

7.2.1 Experimental setup

Before starting any experiment with a user, first we explained the ACM SIG standards, gave a list of ACM SIG format standards and illustrated a small demo about how to find incorrect formats using MS Office Word software and ADFCS software. This took an average of 10-

15 minutes before the user studies. In addition, to remove the learning effect, users were swapped between different systems. For instance, group A users were shown with the manual checking system first, and then they did the automatic checking evaluation after completing the manual checking. Whereas, group B users, were shown with the automatic checking system first, and then they did the evaluations on the manual checking system later. Before the evaluations, we also gave users a pre-questionnaire in order to learn about their background and previous experiences with document format checking.

Then in the user study, we gave the following tasks to the users as follows:

Task 1: In this task the user will be asked for finding incorrect formats of an ACM SIG document. The document[13] contains 15 incorrect formats and the user did not know the number of incorrect formats in the document. It is desirable that a system requires users to invest the least amount of effort in order to find incorrect formats as quickly as possible.

The document which a user was used to find incorrect formats is in soft copy and the reason that the soft copy has been chosen is that, in hard copy it is not possible for user to know that the text inside document has which type of style? And how it is formatted? (Like, text font and text size). We asked the user to use MS Word Office software to manually find incorrect formats by investigating the document.

After the task has been completed by the user, a post questionnaire was given to the user to understand user's judgments about the manual format checking process (Appendix 1, Post-Questionnaire Manual Checking). In this task, the correct number of incorrect formats found by the user and the elapsed time during the manual checking was recorded.

Task 2: The System Usability Scale (SUS) (Brooke, 2013) is one of the most efficient ways of gathering statistically valid data about system usability; giving a clear and reasonably precise score. The System Usability Scale is a ten-item questionnaire administered to users for measuring the perceived ease of use of manual checking. User satisfaction can be viewed as the perceived usability of various functionalities provided by the manual document format checking system. Users were asked to fill the SUS usability for manual checking after

[13] The document that was used in this task is available at http://www.semanticdoc.org/withError1.docx

completing the post-questionnaire (Appendix 1, Post-Questionnaire SUS usability for manual checking).

By filling the SUS usability form for manual checking, the data can be collected and measured based on user satisfaction. Despite major changes in technology, SUS can provide a reliable, valid and quick measure of ease of use any software. SUS is a widely used questionnaire for testing and scoring usability of any software independent of its properties. SUS score was calculated for manual checking, based on SUS Score Calculation (Brooke, 2013).

To generate a SUS score, first all responses from users will be converted to range from 0 to 4, with four being the most positive response. This can be done by subtracting one from user responses to the odd-numbered items and subtracting the user responses to the even-numbered items from 5. Next, adding up the converted values and multiply the total by 2.5. As a result, SUS scores will range from 0 to 100.

Task 3: In task 3, users were asked to find incorrect formats of an ACM SIG document by using ADFCS system (automatic checking). Another document[14] which contains 15 incorrect formats (different from manual checking) was designed, and the user did not know the number of incorrect format in the document. The user opens the upload page, for selecting the document file and click the check button for finding wrong formats. After the report has been produced by ADFCS, the user tries to find an incorrect format in the document based on the indication in the generated report. Then, the user uploads the same document again for finding new format errors. This process continues until all wrong formats have been corrected.

After task completion by the user, the user was asked to fill the post-questionnaire for automatic checking (Appendix 1, Post-Questionnaire Automatic Checking). In this task, the correct number of incorrect formats found by the user and the elapsed time during the automatic checking was recorded. The same post-questionnaire was given to the manual checking process as well so that both manual and automatic processing can be compared.

[14] The document that was used in this task is available at http://www.semanticdoc.org/withError2.docx

Task 4: The SUS usability form for automatic checking will be the final form for measuring the perceptions of usability of ADFCS system (Appendix 1, Post-Questionnaire SUS usability for automatic checking).

Experiment: For implementing all tasks, two documents with ACM SIG format have been created with 15 incorrect formats for each document. One of document was used for manual checking and the other one was used for automatic checking. The reason for choosing 2 different documents with different incorrect formats is that, avoiding the user to learn the incorrect format from the previous checking system. To balance effect of bias, document order has been swapped among users, the first user will use the first document for manual checking and second document for automatic checking, and the document for next user will be swapped so that the second user will use the second document for manual checking and first document for automatic checking.

Participant: In this study, 15 users were participated and we anonymously record their academic background, experience on ACM SIG document format and any type of software that they use for checking their documents (Appendix 1, Pre-Questionnaire Document Experience). Participants were divided into two groups such as group A and group B. At the beginning, we started with explanations and demos; explaining the ACM SIG document styles and showing them a correct format. This took an average of 10-15 minutes. The main aim was introducing, what the ACM SIG document is? And, how the document can be formatted according to ACM SIG document style writing.

7.2.2 User study results

The pre-questionnaire revealed some background information about user's document usage experiences. 93.75% of the participants said they have used MS Office for managing their documents (Figure 23). 93.33% said they are not using any software for checking the format of their documents. 78.57% of the participants said they use MS Office word either "several times in a day" or "several times in a week" to manage their document.[15]

[15] The user responses, SUS score and SUS score calculation, produced numbers and figures of this chapter is available at http://www.semanticdoc.org/sus_semanticdoc.xlsx

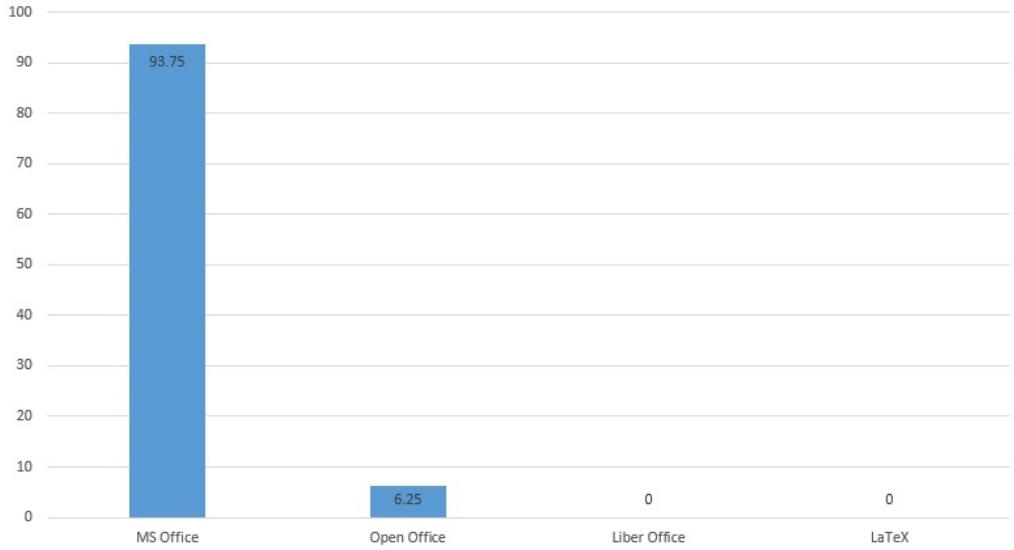


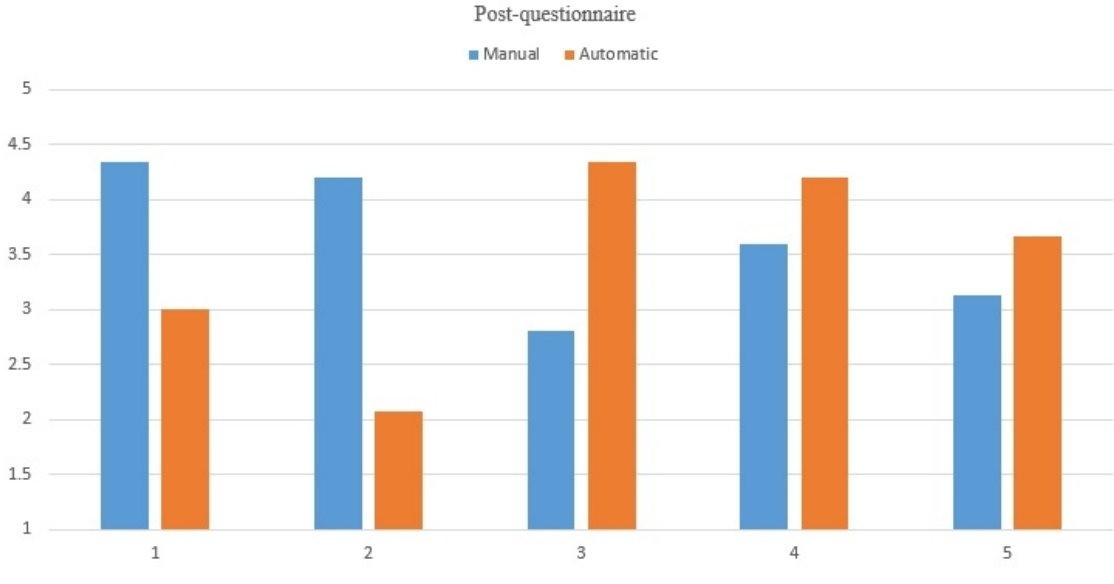
Figure 23: Word processing markup language user experienced

60 % of all participants indicate that they use Web search engines for gathering information about the correctness of their document format. The Figure 23 shows that generally most of the participant use MS Office for their documents.

7.2.2.1 Results of tasks

As stated previously, the goal of the automated document format checking system is to better assist users to find the incorrect format of document than traditional manual checking system. In manual checking system, the user must check every statement in document for its correctness, beside of the requirement to have a good skill on document formatting and standards. As shown in Figure 24 which shows the comparison of task 2 and task 4, most of the users declared that they need to search a lot more with manual checking with an average of (4.33) compared to automatic checking with an average of (3) for finding incorrect format in document. In addition, they found that the task for completion in manual checking was more complex (an average of 4.2) than automatic checking (an average of 2.06). These figures shows that users were struggled more with manual checking to find an incorrect format compared to automatic checking. In automatic checking, users received notifications as a generated report and this helped them to focus on specified statement in a document for its correctness. While users were working with automatic checking, it was easier for them to find an incorrect format compared to manual checking with MS Office Word. However, it is clear that in both cases, correcting the wrong format with the right format depends on the

user's editing skills in a document. Furthermore, users perceptions of automatic checking was more positive than the manual checking; they thought that they did well on tasks (average of 4.33 for automatic versus average of 2.8 for manual), the guidance was helpful (average of 4.2 for automatic versus average of 3.6 for manual) and felt guided to invalid formats (average of 3.66 for automatic versus average of 3.13 for manual).



Q1. I had to search a lot before I found an incorrect format.
Q2. The task was complex.
Q3. I did well on tasks.
Q4. The guidance manual was helpful to solve the tasks.
Q5. I felt guided to invalid results thus I can correct them.

Figure 24: Post-questionnaire for manual and automatic checking

A key challenge between manual and automatic checking is the number of incorrect formats that have been found by users in both systems. In this task the average number of incorrect formats found by using the manual checking is less than automatic checking (4.13 versus 9.86). This shows that users were able to find and correct more incorrect formats using the ADFCS. However, the average elapsed time for automatic is higher than manual (10.9 minutes versus 8.54 minutes) as shown in Figure 25. The reason for this is, the elapsed time in automatic checking is not just the time that the user spend for finding the incorrect format. This includes time spent for opening pages, uploading the document number of times and generating reports. And also as shown in Figure 22, 40% to 60% of the total elapsed time

for generating the report is related to the Jena Reasoner. Nevertheless, with a similar time with manual, users were able to find ~100% (4.13*2=8.26) more incorrect formats with automatic checking with ADFCS.

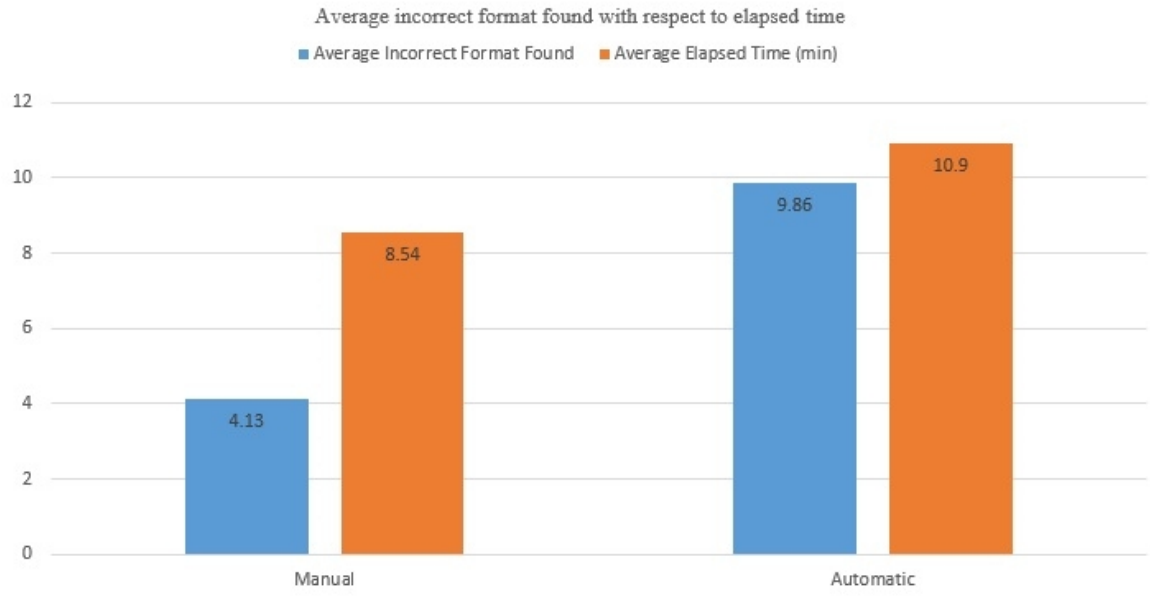
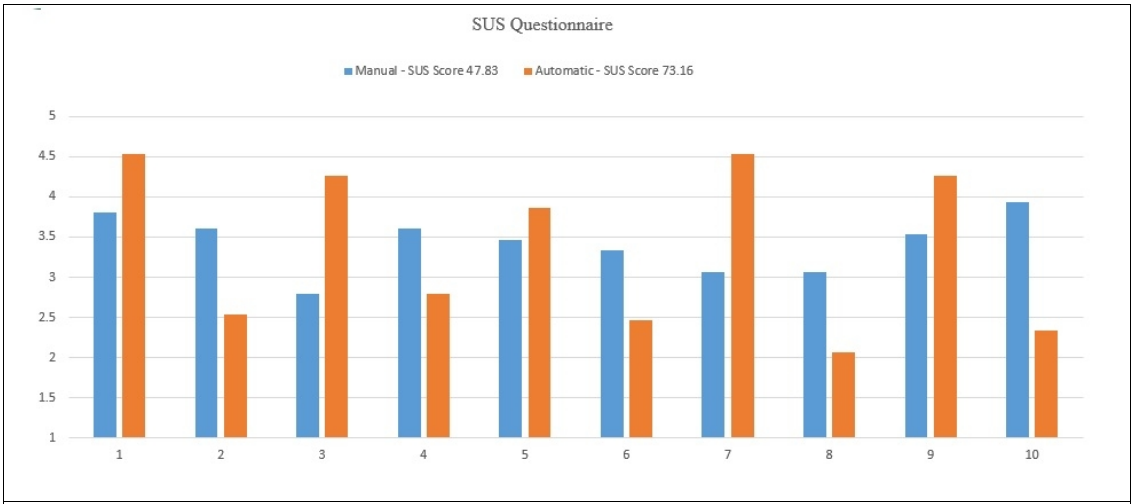


Figure 25: Average incorrect format found and elapsed time for manual and automatic checking

7.2.2.2 Results of user satisfaction

User satisfaction is intending to identify users satisfaction concerning to usability and the different functionalities of the proposed document checking system. SUS is an independent usability questionnaire, which can be used to compare different systems independent of their application design and functionality.



Q1. I think that I would like to use the system frequently.
Q2. I found the system unnecessarily complex.
Q3. I thought the system was easy to use.
Q4. I think that I would need assistance to be able to use the system.
Q5. I found the various functions in the system were well integrated.
Q6. I thought there was too much inconsistency in the system.
Q7. I would imagine that most people would learn to use the system very quickly.
Q8. I found the system very cumbersome/awkward to use.
Q9. I felt very confident using the system.
Q10. I needed to learn a lot of things before I could get going with the system.

Figure 26: Standard usability scale (SUS) Questionnaire for manual and automatic

By using SUS the overall usability of both systems can be determined. The automatic checking system, ADFCS achieved an average SUS score of 73.16, whereas the manual checking system scored an average of 47.83 (Figure 26). The average value for manual checking is 2.8 and for automatic checking is 4.26 where most of users said that the automating checking is easier to use than manual checking (figure 26, Q3), and in Q7 most of users imagine that most people will learn to use the automatic checking very quickly.

By dividing the SUS question into two groups (positive question and negative question), odd questions as positive and even question as negative, it has been shown (Figure 26). In all cases, ADFCS system consistently obtains better values than the manual checking.

During the experiments, we also had an observation about the generated report which affects the usability of ADFCS. The generated report of ADFCS system is completely depending on the validity report of Jena. And the Jena validity report is completely randomized and the triples position cannot be ordered, so that the report which is obtained by SPARQL queries (Figure 14) against the validity report of Jena was ordered by Predicates. This problem with

ADFCS system which was produced by Jena makes the system to be more complex and users spent more time for finding the incorrect format within the document (Figure 26).

CHAPTER 8

CONCLUSIONS AND FUTURE WORK

8.1 Conclusion

In our work, we showed how to take the advantage of the OOXML metadata, in order to automatically check the format and structure of MS word documents. Our new ontology for ACM SIG documents represents the structure and format of these documents. In addition, we illustrated that metadata extraction is fully automated and we can test if the documents are formatted according to ACM SIG standards using extensive rules. Our software system can ease the job of authors, reviewers and organizers in terms of time and effort, since documents can be tested approximately in an average of 9.6 seconds using the online system. The ADFCS can be applied for checking other type of document formats such as *IEEE* and *Elsevier*. In future work, we will publish the open source of our software for reuse and improvements.

8.2 Future Work

In future, multiple type of document will be added to ADFCS system for checking their format and also by adding bulk upload to the system will help users to check multiple document at the same time.

In this version of ADFCS system there is not any modification on document by automatic checking except than capturing the original metadata of document. In future work and upcoming next version of ADFCS system, the generated report can be removed and ADFCS let the user to download his/her document with highlighted wrong format inside the document.

The problem with ADFCS system which is the order of invalid formats of document is randomized by Jena validity report. In our future work we investigate to find a solution for these types of problem.

REFERENCES

26300, I. (2006). *Information technology - Open Document Format for Office Applications (OpenDocument) v1.0.* Geneva: ISO. Retrieved from http://www.iso.org/iso/catalogue_detail?csnumber=43485

29500, I. (2012). *Information technology - Document description and processing languages - Office Open XML File Formats.* Switzerland: ISO. Retrieved from http://standards.iso.org/ittf/PubliclyAvailableStandards/c061750_ISO_IEC_29500-1_2012.zip

Al-Ghanim, M., Noah, S. A., & Sembok, T. M. (2001). Automating XML Schema Matching: A Composite Approach. *Electrical Engineering and Informatics (ICEEI)* (pp. 1-5). Bandung: IEEE. doi:10.1109/ICEEI.2011.6021797

Bakkas, J., Jakjoud, W., & Bahaj, M. (2014). Semantic mapping at the schema level of XML documents to ontologies. *Next Generation Networks and Services (NGNS), 2014 Fifth International Conference* (pp. 165-169). Casablanca: IEEE. doi:10.1109/NGNS.2014.6990247

Bosch, T., & Mathiak, B. (2011). XSLT Transformation Generating OWL Ontologies Automatically Based on XML Schemas. *Internet Technology and Secured Transactions (ICITST), 2011 International Conference* (pp. 660-667). Abu Dhabi: IEEE.

Bott, E., & Leonhard, W. (2006). *Special Edition Using Microsoft Office 2007.* Indiana 46240: Que.

Brooke, J. (2013). SUS: A Retrospective. *Journal of Usability Studies, 8*(2), 29-44. Retrieved from http://uxpajournal.org/wp-content/uploads/pdf/JUS_Brooke_February_2013.pdf

Deursen, D. V., Poppe, C., Martens, G., Mannens, E., & Walle, R. V. (2008). XML to RDF Conversion: a Generic Approach. *Automated solutions for Cross Media Content and Multi-channel Distribution, 2008. AXMEDIS '08. International Conference* (pp. 138-144). Florence: IEEE. doi:10.1109/AXMEDIS.2008.17

Guarino, N. (1998). Formal Ontology and Information Systems. *Proceedings of FOIS'98* (pp. 3-15). Trento,: IOS Press. Retrieved from http://www.mif.vu.lt/~donatas/Vadovavimas/Temos/OntologiskaiTeisingasKoncepcinisModeliavimas/papildoma/Guarino98-Formal%20Ontology%20and%20Information%20Systems.pdf

Guo, R., Wu, F., Li, Y. Z., Zhu, R., & Sheng, K. (2015). The information hiding mechanism based on compressed document format. *International Journal of Computing Science and Mathematics, 6*(1), 97-106. doi:10.1504/IJCSM.2015.067547

Harth, A., Janik, M., & Staab, S. (2011). *Handbook of Semantic Web Technologies.* (J. Domingue, D. Fensel, & J. A. Hendler, Eds.) Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-540-92913-0_2

He, W., Lv, T., Meis, M., & Yan, P. (2013). Visual Evaluation of XPath Queries. *Computational and Information Sciences (ICCIS), 2013 Fifth International Conference* (pp. 434-437). Shiyang: IEEE. doi:10.1109/ICCIS.2013.121

Hebeler, J., Fisher, M., Blace, R., & Perez-Lopez, A. (2009). *Semantic Web Programming.* IN 46256: Wiley Publishing, Inc. Retrieved from http://ia1213.googlecode.com/files/semantic-web-programming.9780470418017.47881.pdf

Hou, X., Li, N., Yang, H.-b., & Liang, Q. (2010). Comparison of Wordprocessing Document Format in OOXML and ODF. *Semantics Knowledge and Grid (SKG), 2010 Sixth International Conference* (pp. 297-300). Beijing: IEEE. doi:10.1109/SKG.2010.44

Hu, X., Lian, X., Mo, Y., Zhang, H., & Yuan, X. (2012). Query XML Data in RDBMS. *Web Information Systems and Applications Conference (WISA), 2012 Ninth* (pp. 15-20). Haikou: IEEE. doi:10.1109/WISA.2012.12

International, E. (2012, December). ECMA-376, 4th Edition Office Open XML File Formats — Fundamentals and Markup Language Reference. *2012*. Retrieved from http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-376,%20Fourth%20Edition,%20Part%201%20-%20Fundamentals%20And%20Markup%20Language%20Reference.zip

Jieping, T., & Zhaohua, H. (2010). Discovering OWL Ontologies from XML. *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference. 6*, pp. 517-519. Chengdu: IEEE. doi:10.1109/ICACTE.2010.5579194

Khalid, A. S., Syed, A. H., & Qadir, M. A. (2009). OntRel: An Optimized Relational Structure for Storage of Dynamic OWL-DL Ontologies. *Multitopic Conference, 2009. INMIC 2009. IEEE 13th International* (pp. 1-6). Islamabad: IEEE. doi:10.1109/INMIC.2009.5383093

Kwok, T., & Nguyen, T. (2006). An Automatic Method to Extract Data from an Electronic Contract Composed of a Number of Documents in PDF Format. *Proceedings of the 8th IEEE International Conference on E-Commerce Technology and the 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC/EEE'06)*. Retrieved from http://ieeexplore.ieee.org/ielx5/10920/34369/01640288.pdf?tp=&arnumber=1640288&isnumber=34369

Lu, X., Wang, J. Z., Mitra, P., & Giles, C. L. (2007). Automatic Extraction of Data from 2-D Plots in Documents. *Ninth International Conference on Document Analysis and*

Recognition (ICDAR 2007). 1, pp. 188-192. Pennsylvania: IEEE. doi:10.1109/ICDAR.2007.4378701

Milicka, M., & Burget, R. (2013). Web Document Description Based on Ontologies. *Informatics and Applications (ICIA),2013 Second International Conference* (pp. 288-293). Lodz: IEEE. doi:10.1109/ICoIA.2013.6650271

Mishra, G., & Yagyasen, D. (2013). Semantic descriptions of resources with proactive behavior of autonomous condition monitoring applications. *International Journal of Scientific & Engineering Research, 4*(7), 943-947. Retrieved from http://www.ijser.org/researchpaper/Semantic-descriptions-of-resources-with-proactive-behavior-of-autonomous-condition-monitoring-applications.pdf

Pellet, J. P., & Chevalier, M. (2014). Automatic Extraction of Formal Features from Word, Excel, and PowerPoint Productions in a Diagnostic Assessment Perspective. *Education Technologies and Computers (ICETC), 2014 The International Conference* (pp. 1-6). Lausanne: IEEE. doi:10.1109/ICETC.2014.6998893

Şah, M., & Wade, V. (2010). Automatic metadata extraction from multilingual enterprise content. *CIKM10 Proceedings of the 19th ACM international conference on Information and knowledge management* (pp. 1665-1668). Toronto: ACM. doi:10.1145/1871437.1871699

Siricharoen, W. V. (2008). Merging Ontologies for Object Oriented Software Engineering. *Networked Computing and Advanced Information Management, 2008. NCM '08. Fourth International Conference. 2*, pp. 525-530. Gyeongju: IEEE. doi:10.1109/NCM.2008.262

TIAN, Y. A., Ning LI, X. H., & LIANG, Q. (2009). Intelligent Processing Based on Ontology for Office Document. *Information Engineering and Computer Science, 2009. ICIECS 2009. International Conference* (pp. 1-4). Wuhan: IEEE. doi:10.1109/ICIECS.2009.5363486

Xu, D., Hongxing, P., & Junjie, L. (2010). Research and application of document format checking technology based on Java. *China academic journal electronic and publishing house*(19), 4309-4315. Retrieved from http://wenku.baidu.com/view/4f5a381e14791711cc791755.html

APPENDICES

APPENDIX 1

QUESTIONNAIRE FORMS

Pre-Questionnaire Document Experience

1. Which one of the word processing mark-up language you have experience with?

Liber Office	Open Office	LaTex	Microsoft Office

2. How often do you use Microsoft Office Word to manage your documents?

Several times in a year	Several times in a month	Several times in a week	Several times in a day

3. How often do you use web search engines to gather information about the format of your document (e.g. title, abstract, text font, alignment)?

Several times in a year	Several times in a month	Several times in a week	Several times in a day

4. Did you use any systems/software in the past for checking the format of your document (yes/no)? (Systems/Software name?)

5. If yes, how often have you used them?

Very little	Little	often	Very often

6. Your academic background.

B.Sc.	M.Sc.	PhD	Assist. Prof	Associate. Prof	Prof

7. Your department.

Post-Questionnaire for Manual Checking

	Strongly Disagree	Disagree	Fair	Agree	Strongly Agree
1. I had to search a lot before I found an incorrect format.	1	2	3	4	5
2. The task was complex.	1	2	3	4	5
3. I did well on tasks.	1	2	3	4	5
4. The guidance manual was helpful to solve the tasks.	1	2	3	4	5
5. I felt guided to invalid results thus I can correct them.	1	2	3	4	5
6. The number of incorrect format that you have found. (number)					
7. The elapsed time for manual checking in seconds. (leave it blank)					

SUS Usability Questionnaire for Manual Checking

Instructions: For each of the following statements, mark <u>one</u> box that best describes your reactions to the system *today*.

	Strongly Disagree	Disagree	Fair	Agree	Strongly Agree
1. I think that I would like to use the system frequently.	1	2	3	4	5
2. I found the system unnecessarily complex.	1	2	3	4	5
3. I thought the system was easy to use.	1	2	3	4	5
4. I think that I would need assistance to be able to use the system.	1	2	3	4	5
5. I found the various functions in the system were well integrated.	1	2	3	4	5
6. I thought there was too much inconsistency in the system.	1	2	3	4	5
7. I would imagine that most people would learn to use the system very quickly.	1	2	3	4	5
8. I found the system very cumbersome/awkward to use.	1	2	3	4	5
9. I felt very confident using the system.	1	2	3	4	5
10. I needed to learn a lot of things before I could get going with the system.	1	2	3	4	5

What features/characteristics did you like most about the system?

What features/characteristics did you least like about the system?

Comments?

Post-Questionnaire for www.SemanticDoc.Org – Automatic Checking

	Strongly Disagree	Disagree	Fair	Agree	Strongly Agree
1. I had to search a lot before I found an incorrect format.	1	2	3	4	5
2. The task is complex.	1	2	3	4	5
3. I did well on tasks.	1	2	3	4	5
4. The guidance manual was helpful to solve the tasks.	1	2	3	4	5
5. I felt guided to invalid results thus I can correct them.	1	2	3	4	5
6. The number of incorrect format that you have found. (number)					
7. The elapsed time for manual checking in seconds. (leave it blank)					

SUS Usability Questionnaire for www.SemanticDoc.Org – Automatic Checking

Instructions: For each of the following statements, mark <u>one</u> box that best describes your reactions to the system *today*.

	Strongly Disagree		Disagree		Fair		Agree		Strongly Agree
1. I think that I would like to use the system frequently.		1		2		3		4		5
2. I found the system unnecessarily complex.		1		2		3		4		5
3. I thought the system was easy to use.		1		2		3		4		5
4. I think that I would need assistance to be able to use the system.		1		2		3		4		5
5. I found the various functions in the system were well integrated.		1		2		3		4		5
6. I thought there was too much inconsistency in the system.		1		2		3		4		5
7. I would imagine that most people would learn to use the system very quickly.		1		2		3		4		5
8. I found the system very cumbersome/awkward to use.		1		2		3		4		5
9. I felt very confident using the system.		1		2		3		4		5
10. I needed to learn a lot of things before I could get going with the system.		1		2		3		4		5

What features/characteristics did you like most about the system?

What features/characteristics did you least like about the system?

Comments?

APPENDIX 2

COMPLETE JENA RULES FOR ACM DOCUMENT

@prefix sig:<http://www.semanticdoc.org/ontology/2015/v1.6.owl#>.

#Rules

#Title

[rule001:(?f sig:titleText?d)->(?f sig:validTitleText sig:true)]

[rule002:(?f sig:titleFont"Helvetica")->(?f sig:validTitleFont sig:true)]

[rule003:(?f sig:titleSize"36")->(?f sig:validTitleSize sig:true)]

[rule004:(?f sig:titleBold"bold")->(?f sig:validTitleBold sig:true)]

[rule005:(?f sig:titleJustify"center")->(?f sig:validTitleJustify sig:true)]

#Author

[rule006:(?f sig:authorName?d)->(?f sig:validAuthorName sig:true)]

[rule007:(?f sig:authorJustify?d),notEqual(?d,"center")->(?f sig:invalidAuthorJustify sig:true)]

[rule008:(?f sig:authorJustify"center")->(?f sig:validAuthorJustify sig:true)]

[rule009:(?f sig:authorFont?d),notEqual(?d,"Helvetica")->(?f sig:invalidAuthorFont sig:true)]

[rule010:(?f sig:authorFont"Helvetica")->(?f sig:validAuthorFont sig:true)]

[rule011:(?f sig:authorSize?d),notEqual(?d,"24")->(?f sig:invalidAuthorSize sig:true)]

[rule012:(?f sig:authorSize"24")->(?f sig:validAuthorSize sig:true)]

#AffilateAuthor

[rule013:(?f sig:affilateAuthor?d)->(?f sig:validAffilateAuthor sig:true)]

[rule014:(?f sig:affilateJustify?d),notEqual(?d,"center")->(?f sig:invalidAffilateJustify sig:true)]

[rule015:(?f sig:affilateJustify"center")->(?f sig:validAffilateJustify sig:true)]

[rule016:(?f sig:affilateFont?d),notEqual(?d,"Helvetica")->(?f sig:invalidAffilateFont sig:true)]

[rule017:(?f sig:affilateFont"Helvetica")->(?f sig:validAffilateFont sig:true)]

[rule018:(?f sig:affilateSize?d),notEqual(?d,"20")->(?f sig:invalidAffilateSize sig:true)]

[rule019:(?f sig:affilateSize"20")->(?f sig:validAffilateSize sig:true)]

#Address1Author

[rule020:(?f sig:address1Author?d)->(?f sig:validAddress1Author sig:true)]

[rule021:(?f sig:address1Justify?d),notEqual(?d,"center")->(?f sig:invalidAddress1Justify sig:true)]

[rule022:(?f sig:address1Justify"center")->(?f sig:validAddress1Justify sig:true)]

[rule023:(?f sig:address1Font?d),notEqual(?d,"Helvetica")->(?f sig:invalidAddress1Font sig:true)]

[rule024:(?f sig:address1Font"Helvetica")->(?f sig:validAddress1Font sig:true)]

[rule025:(?f sig:address1Size?d),notEqual(?d,"20")->(?f sig:invalidAddress1Size sig:true)]

[rule026:(?f sig:address1Size"20")->(?f sig:validAddress1Size sig:true)]

#Address2Author

[rule027:(?f sig:address2Author?d)->(?f sig:validAddress2Author sig:true)]

[rule028:(?f sig:address2Justify?d),notEqual(?d,"center")->(?f sig:invalidAddress2Justify sig:true)]

[rule029:(?f sig:address2Justify"center")->(?f sig:validAddress2Justify sig:true)]

[rule030:(?f sig:address2Font?d),notEqual(?d,"Helvetica")->(?f sig:invalidAddress2Font sig:true)]

[rule031:(?f sig:address2Font"Helvetica")->(?f sig:validAddress2Font sig:true)]

[rule032:(?f sig:address2Size?d),notEqual(?d,"20")->(?f sig:invalidAddress1Size sig:true)]

[rule033:(?f sig:address2Size"20")->(?f sig:validAddress2Size sig:true)]

#PhoneAuthor

[rule034:(?f sig:phoneAuthor?d)->(?f sig:validPhoneAuthor sig:true)]

[rule035:(?f sig:phoneJustify?d),notEqual(?d,"center")->(?f sig:invalidPhoneJustify sig:true)]

[rule036:(?f sig:phoneJustify"center")->(?f sig:validPhoneJustify sig:true)]

[rule037:(?f sig:phoneFont?d),notEqual(?d,"Helvetica")->(?f sig:invalidPhoneFont sig:true)]

[rule038:(?f sig:phoneFont"Helvetica")->(?f sig:validPhoneFont sig:true)]

[rule039:(?f sig:phoneSize?d),notEqual(?d,"20")->(?f sig:invaidPhoneSize sig:true)]

[rule040:(?f sig:phoneSize"20")->(?f sig:validPhoneSize sig:true)]

#EmailAuthor

[rule041:(?f sig:emailAuthor?d)->(?f sig:validEmailAuthor sig:true)]

[rule042:(?f sig:emailJustify?d),notEqual(?d,"center")->(?f sig:invalidEmailJustify sig:true)]

[rule043:(?f sig:emailJustify"center")->(?f sig:validEmailJustify sig:true)]

[rule044:(?f sig:emailFont?d),notEqual(?d,"Helvetica")->(?f sig:invalidEmailFont sig:true)]

[rule045:(?f sig:emailFont"Helvetica")->(?f sig:validEmailFont sig:true)]

[rule046:(?f sig:emailSize?d),notEqual(?d,"24")->(?f sig:invalidEmailSize sig:true)]

[rule047:(?f sig:emailSize"24")->(?f sig:validEmailSize sig:true)]

#Abstract

[rule048:(?f sig:hasAbstract?d)->(?f sig:validAbstractText sig:true)]

[rule049:(?f sig:abstractJustify?d),notEqual(?d,"both")->(?f sig:invalidAbstractJustify sig:true)]

[rule050:(?f sig:abstractJustify"both")->(?f sig:validAbstractJustify sig:true)]

[rule051:(?f sig:abstractFont?d),notEqual(?d,"TimesNewRoman")->(?f sig:invalidAbstractFont sig:true)]

[rule052:(?f sig:abstractFont"TimesNewRoman")->(?f sig:validAbstractFont sig:true)]

[rule053:(?f sig:abstractSize?d),notEqual(?d,"18")->(?f sig:invalidAbstractSize sig:true)]

[rule054:(?f sig:abstractSize"18")->(?f sig:validAbstractSize sig:true)]

#CategoryandSubjectDescriptor

[rule055:(?f sig:hasCategory?d)->(?f sig:validCategoryText sig:true)]

[rule056:(?f sig:categoryJustify?d),notEqual(?d,"both")->(?f sig:invalidCategoryJustify sig:true)]

[rule057:(?f sig:categoryJustify"both")->(?f sig:validCategoryJustify sig:true)]

[rule058:(?f sig:categoryFont?d),notEqual(?d,"TimesNewRoman")->(?f sig:invalidCategoryFont sig:true)]

[rule059:(?f sig:categoryFont"TimesNewRoman")->(?f sig:validCategoryFont sig:true)]

[rule060:(?f sig:categorySize?d),notEqual(?d,"18")->(?f sig:invalidCategorySize sig:true)]

[rule061:(?f sig:categorySize"18")->(?f sig:validCategorySize sig:true)]

#GeneralTerms

[rule062:(?f sig:hasGeneral?d)->(?f sig:validGeneralText sig:true)]

[rule063:(?f sig:generalJustify?d),notEqual(?d,"both")->(?f sig:invalidGeneralJustify sig:true)]

[rule064:(?f sig:generalJustify"both")->(?f sig:validGeneralJustify sig:true)]

[rule065:(?f sig:generalFont?d),notEqual(?d,"TimesNewRoman")->(?f sig:invalidGeneralFont sig:true)]

[rule066:(?f sig:generalFont"TimesNewRoman")->(?f sig:validGeneralFont sig:true)]

[rule067:(?f sig:generalSize?d),notEqual(?d,"18")->(?f sig:invalidGeneralSize sig:true)]

[rule068:(?f sig:generalSize"18")->(?f sig:validGeneralSize sig:true)]

[rule069:(?f sig:generalTerm?d)->(?f sig:validGeneralTerm sig:true)]

[rule069:(?f sig:generalTermWrong?d)->(?f sig:invalidGeneralTermDesignated sig:true)]

#[rule069:(?f sig:generalTerm"Algorithms")->(?f sig:validGeneralTermAlgorithms sig:true)]

#[rule070:(?f sig:generalTerm"Management")->(?f sig:validGeneralTermManagement sig:true)]

#[rule071:(?f sig:generalTerm"Measurement")->(?f sig:validGeneralTermMeasurement sig:true)]

#[rule072:(?f sig:generalTerm"Documentation")->(?f sig:validGeneralTermDocumentation sig:true)]

#[rule073:(?f sig:generalTerm"Performance")->(?f sig:validGeneralTermPerformance sig:true)]

#[rule074:(?f sig:generalTerm"Design")->(?f sig:validGeneralTermDesign sig:true)]

#[rule075:(?f sig:generalTerm"Economics")->(?f sig:validGeneralTermEconomics sig:true)]

#[rule076:(?f sig:generalTerm"Reliability")->(?f sig:validGeneralTermReliability sig:true)]

#[rule077:(?f sig:generalTerm"Experimentation")->(?f sig:validGeneralTermExperimentation sig:true)]

#[rule078:(?f sig:generalTerm"Security")->(?f sig:validGeneralTermSecurity sig:true)]

#[rule079:(?f sig:generalTerm"HumanFactors")->(?f sig:validGeneralTermHumanFactors sig:true)]

#[rule080:(?f sig:generalTerm"Standardization")->(?f sig:validGeneralTermStandardization sig:true)]

#[rule081:(?f sig:generalTerm"Languages")->(?f sig:validGeneralTermLanguages sig:true)]

#[rule082:(?f sig:generalTerm"Theory")->(?f sig:validGeneralTermTheory sig:true)]

#[rule083:(?f sig:generalTerm"LegalAspects")->(?f sig:validGeneralTermLegalAspects sig:true)]

#[rule084:(?f sig:generalTerm"Verification")->(?f sig:validGeneralTermVerification sig:true)]

#Keywords

[rule085:(?f sig:hasKeyword?d)->(?f sig:validKeywordText sig:true)]

[rule086:(?f sig:keywordJustify?d),notEqual(?d,"both")->(?f sig:invalidKeywordJustify sig:true)]

[rule087:(?f sig:keywordJustify"both")->(?f sig:validKeywordJustify sig:true)]

[rule088:(?f sig:keywordFont?d),notEqual(?d,"TimesNewRoman")->(?f sig:invalidKeywordFont sig:true)]

[rule089:(?f sig:keywordFont"TimesNewRoman")->(?f sig:validKeywordFont sig:true)]

[rule090:(?f sig:keywordSize?d),notEqual(?d,"18")->(?f sig:invalidKeywordSize sig:true)]

[rule091:(?f sig:keywordSize"18")->(?f sig:validKeywordSize sig:true)]

#Section

[rule092:(?f sig:hasSection?d)->(?f sig:validSectionText sig:true)]

[rule093:(?f sig:sectionJustify?d),notEqual(?d,"both")->(?f sig:invalidSectionJustify sig:true)]

[rule094:(?f sig:sectionJustify"both")->(?f sig:validSectionJustify sig:true)]

[rule095:(?f sig:sectionFont?d),notEqual(?d,"TimesNewRoman")->(?f sig:invalidSectionFont sig:true)]

[rule096:(?f sig:sectionFont"TimesNewRoman")->(?f sig:validSectionFont sig:true)]

[rule097:(?f sig:sectionSize?d),notEqual(?d,"18")->(?f sig:invalidSectionSize sig:true)]

[rule098:(?f sig:sectionSize"18")->(?f sig:validSectionSize sig:true)]

#SubSection

[rule099:(?f sig:hasSubSection?d)->(?f sig:validSubSectionText sig:true)]

[rule100:(?f sig:subSectionJustify?d),notEqual(?d,"both")->(?f sig:invalidSubSectionJustify sig:true)]

[rule101:(?f sig:subSectionJustify"both")->(?f sig:validSubSectionJustify sig:true)]

[rule102:(?f sig:subSectionFont?d),notEqual(?d,"TimesNewRoman")->(?f sig:invalidSubSectionFont sig:true)]

[rule103:(?f sig:subSectionFont"TimesNewRoman")->(?f sig:validSubSectionFont sig:true)]

[rule104:(?f sig:subSectionSize?d),notEqual(?d,"18")->(?f sig:invalidSubSectionSize sig:true)]

[rule105:(?f sig:subSectionSize"18")->(?f sig:validSubSectionSize sig:true)]

#SubSubSection

[rule106:(?f sig:hasSubSubSec?d)->(?f sig:validSubSubSectionText sig:true)]

[rule107:(?f sig:subSubSecJustify?d),notEqual(?d,"both")->(?f sig:invalidSubSubSectionJustify sig:true)]

[rule108:(?f sig:subSubSecJustify"both")->(?f sig:validSubSubSectionJustify sig:true)]

[rule109:(?f sig:subSubSecFont?d),notEqual(?d,"TimesNewRoman")->(?f sig:invalidSubSubSectionFont sig:true)]

[rule110:(?f sig:subSubSecFont"TimesNewRoman")->(?f sig:validSubSubSectionFont sig:true)]

[rule111:(?f sig:subSubSecSize?d),notEqual(?d,"18")->(?f sig:invalidSubSubSectionSize sig:true)]

[rule112:(?f sig:subSubSecSize"18")->(?f sig:validSubSubSectionSize sig:true)]

#REFERENCES

[rule113:(?f sig:hasReferences?d)->(?f sig:validReferencesText sig:true)]

[rule114:(?f sig:referencesJustify?d),notEqual(?d,"left")->(?f sig:invalidReferencesJustify sig:true)]

[rule115:(?f sig:referencesJustify"left")->(?f sig:validReferencesJustify sig:true)]

[rule116:(?f sig:referencesFont?d),notEqual(?d,"TimesNewRoman")->(?f sig:invalidReferencesFont sig:true)]

[rule117:(?f sig:referencesFont"TimesNewRoman")->(?f sig:validReferencesFont sig:true)]

[rule118:(?f sig:referencesSize?d),notEqual(?d,"18")->(?f sig:invalidReferencesSize sig:true)]

[rule119:(?f sig:referencesSize"18")->(?f sig:validReferencesSize sig:true)]

#TABLE

[rule120:(?f sig:hasTable?d)->(?f sig:validTableText sig:true)]

[rule121:(?f sig:tableJustify?d),notEqual(?d,"center")->(?f sig:invalidTableJustify sig:true)]

[rule122:(?f sig:tableJustify"center")->(?f sig:validTableJustify sig:true)]

[rule123:(?f sig:tableFont?d),notEqual(?d,"TimesNewRoman")->(?f sig:invalidTableFont sig:true)]

[rule124:(?f sig:tableFont"TimesNewRoman")->(?f sig:validTableFont sig:true)]

[rule125:(?f sig:tableSize?d),notEqual(?d,"18")->(?f sig:invalidTableSize sig:true)]

[rule126:(?f sig:tableSize"18")->(?f sig:validTableSize sig:true)]

[rule127:(?f sig:tableBold?d),notEqual(?d,"bold")->(?f sig:invalidTableBold sig:true)]

[rule128:(?f sig:tableBold"bold")->(?f sig:validTableBold sig:true)]

#PageSize

#[rule129:(?f sig:pageWidth"12240")->(?f sig:validPageWidth sig:true)]

#[rule130:(?f sig:pageHight"15840")->(?f sig:validPageHight sig:true)]

#PageMargin

#[rule131:(?f sig:marginTop"1080")->(?f sig:validMarginTop sig:true)]

#[rule132:(?f sig:marginRight"1080")->(?f sig:validMarginRight sig:true)]

#[rule133:(?f sig:marginButtom"1440")->(?f sig:validMarginButtom sig:true)]

#[rule134:(?f sig:marginLeft"1080")->(?f sig:validMarginLeft sig:true)]

#[rule135:(?f sig:marginHeader"720")->(?f sig:validMarginHeader sig:true)]

#[rule136:(?f sig:marginFooter"720")->(?f sig:validMarginFooter sig:true)]

#[rule137:(?f sig:marginGutter"0")->(?f sig:validMarginGutter sig:true)]

APPENDIX 3

ACM SIG ONTOLOGY

<rdf:RDF xmlns="http://www.semanticdoc.org/ontology/2015/v1.6.owl#"

xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"

xmlns:v13="http://www.semanticdoc.org/ontology/2015/v1.6.owl#"

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:v12="http://lwww.semanticdoc.org/ontology/2015/v1.6.owl#"

xml:base="http://www.semanticdoc.org/ontology/2015/v1.6.owl">

<owl:Ontology rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl">

<owl:versionIRI rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl"/>

</owl:Ontology>

<owl:ObjectProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#hasAbstract"/>

<owl:ObjectProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#hasAuthor"/>

<owl:ObjectProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#hasCatSubDes"/>

<owl:ObjectProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#hasGeneralTerms"/>

<owl:ObjectProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#hasKeyword"/>

<owl:ObjectProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#hasSection"/>

<owl:ObjectProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#hasTitle">

<rdfs:domain rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Title"/>

</owl:ObjectProperty>

<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#abstractFont"/>

<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#abstractJustify">

<rdfs:comment>

the text alignment in Abstract part of the document it must be Justified.

The value (both) is refer to the OOXML in metadata extraction.

</rdfs:comment>

<rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topDataProperty"/>

</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#abstractSize"/>

<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#address1Author"/>

<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#address1Font"/>

<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#address1Justify"/>
<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#address1Size">
<rdfs:comment>the font of abstract must be Times New Roman font.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#address2Author"/>
<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#address2Font"/>
<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#address2Justify"/>
<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#address2Size"/>
<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#affilateAuthor"/>
<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#affilateFont"/>
<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#affilateJustify"/>
<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#affilateSize"/>
<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#authorFont"/>
<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#authorJustify"/>
<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#authorName"/>
<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#authorSize"/>
<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#categoryFont"/>
<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#categoryJustify"/>
<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#categorySize"/>
<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#emailAuthor"/>
<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#emailFont"/>
<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#emailJustify"/>
<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#emailSize"/>
<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#generalFont"/>
<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#generalJustify"/>
<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#generalSize"/>
<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#generalTerm"/>
<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#hasAbstract"/>
<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#hasCategory"/>
<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#hasGeneral"/>
<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#hasKeyword"/>
<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#hasReferences"/>
<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#hasSection"/>

<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#hasSubSection"/>

<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#hasSubSubSec"/>

<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#hasTable"/>

<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#keywordFont"/>

<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#keywordJustify"/>

<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#keywordSize"/>

<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#phoneAuthor"/>

<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#phoneFont"/>

<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#phoneJustify"/>

<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#phoneSize"/>

<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#referencesFont"/>

<owl:DatatypeProperty
rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#referencesJustify"/>

<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#referencesSize"/>

<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#sectionFont"/>

<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#sectionJustify"/>

<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#sectionSize"/>

<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#subSectionFont"/>

<owl:DatatypeProperty
rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#subSectionJustify"/>

<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#subSectionSize"/>

<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#subSubSecFont"/>

<owl:DatatypeProperty
rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#subSubSecJustify"/>

<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#subSubSecSize"/>

<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#tableBold"/>

<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#tableFont"/>

<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#tableJustify"/>

<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#tableSize"/>

<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#titleBold"/>

<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#titleFont"/>

<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#titleJustify"/>

<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#titleSize"/>

<owl:DatatypeProperty rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#titleText">

<rdfs:domain rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Title"/>

<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>

</owl:DatatypeProperty>

<owl:Class rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Abstract">

<owl:equivalentClass>

<owl:Restriction>

<owl:onProperty rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#hasAbstract"/>

<owl:onClass rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Abstract"/>

<owl:qualifiedCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:qualifiedCardinality>

</owl:Restriction>

</owl:equivalentClass>

<rdfs:comment>

the abstract part of ducument. Abstract (Times New Roman, 12 point, Bold, left)

Abstract Text (Times New Roman, 9 point, Justify)

</rdfs:comment>

</owl:Class>

<owl:Class rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Author">

<owl:equivalentClass>

<owl:Restriction>

<owl:onProperty rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#hasAuthor"/>

<owl:onClass rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Author"/>

<owl:minQualifiedCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:minQualifiedCardinality
>

</owl:Restriction>

</owl:equivalentClass>

<owl:equivalentClass>

<owl:Restriction>

<owl:onProperty rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#hasAuthor"/>

<owl:onClass rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Author"/>

<owl:maxQualifiedCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">3</owl:maxQualifiedCardinality
>

</owl:Restriction>

</owl:equivalentClass>

<rdfs:comment>

The author(s) section. min 1 max 3 author.

line 1 Name (Helvetica , 12 point, center)

line 2 Affiliation (Helvetica, 10 point, center)

line 3 Second affiliation (Helvetica, 10 point, center)

line 4 Address (Helvetica, 10 point, center)

line 5 Second address (Helvetica, 10 point, center)

line 6 Email (Helvetica, 12 point, center)

</rdfs:comment>

</owl:Class>

<owl:Class rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#CatSubDes">

<owl:equivalentClass>

<owl:Restriction>

<owl:onProperty rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#hasCatSubDes"/>

<owl:onClass rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#CatSubDes"/>

<owl:minQualifiedCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">3</owl:minQualifiedCardinality
>

</owl:Restriction>

</owl:equivalentClass>

<owl:equivalentClass>

<owl:Restriction>

<owl:onProperty rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#hasCatSubDes"/>

<owl:onClass rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#CatSubDes"/>

<owl:maxQualifiedCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">10</owl:maxQualifiedCardinalit
y>

</owl:Restriction>

</owl:equivalentClass>

<rdfs:comment>Category and Subject Dexcriptor.</rdfs:comment>

</owl:Class>

<owl:Class rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#GeneralTerms">

<owl:equivalentClass>

<owl:Restriction>

<owl:onProperty rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#hasGeneralTerms"/>

<owl:onClass rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#GeneralTerms"/>

<owl:qualifiedCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:qualifiedCardinality>

</owl:Restriction>

</owl:equivalentClass>

<rdfs:comment>

General terms. Algorithms, Design, Documentation, Economics, Experimentation, Human Factors, Languages,

Legal Aspects, Management, Measurement, Performance, Reliability, Security, Standardization, Theory, Verification.

</rdfs:comment>

</owl:Class>

<owl:Class rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Keywords">

<owl:equivalentClass>

<owl:Restriction>

<owl:onProperty rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#hasKeyword"/>

<owl:onClass rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Keywords"/>

<owl:qualifiedCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:qualifiedCardinality>

</owl:Restriction>

</owl:equivalentClass>

</owl:Class>

<!--

http://www.semanticdoc.org/ontology/2015/v1.6.owl#Section

-->

<owl:Class rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Section">

<owl:equivalentClass>

<owl:Restriction>

<owl:onProperty rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#hasSection"/>

<owl:onClass rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Section"/>

<owl:minQualifiedCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:minQualifiedCardinality
>

</owl:Restriction>

</owl:equivalentClass>

<rdfs:comment>

Section of the document. like Introduction and references Section (Times New Roman, 12 point, Bold, Left, Numbered List,

All Capital Letter) Section text (Times New Roman, 9 point, Justify, First letter Capital)

</rdfs:comment>

</owl:Class>

<owl:Class rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#SubSection">

<rdfs:subClassOf rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Section"/>

</owl:Class>

<owl:Class rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#SubSubSection">

<rdfs:subClassOf rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#SubSection"/>

</owl:Class>

<owl:Class rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Title">

<owl:equivalentClass>

<owl:Restriction>

<owl:onProperty rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#hasTitle"/>

<owl:onClass rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Title"/>

<owl:minQualifiedCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:minQualifiedCardinality
>

</owl:Restriction>

</owl:equivalentClass>

<rdfs:comment>

Title of Document. Helvetica, 18 point, Bold, Center

</rdfs:comment>

</owl:Class>

<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#abstractFont">

<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Abstract"/>

<abstractFont>Times New Roman</abstractFont>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#abstractJustify">

<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Abstract"/>

<abstractJustify>both</abstractJustify>

<rdfs:comment>

the text alignment in Abstract part of the document it must be Justified. The value (both) is refer to the OOXML in metadata extraction.

</rdfs:comment>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#abstractSize">

<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Abstract"/>

<abstractSize>18</abstractSize>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#address1Author">

<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Author"/>

<address1Author/>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#address1Font">

<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Author"/>

<address1Font>Helvetica</address1Font>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#address1Justify">

<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Author"/>

<address1Justify>center</address1Justify>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#address1Size">

<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Author"/>

<rdfs:comment>the font of abstract must be Times New Roman font.</rdfs:comment>

<address1Size>20</address1Size>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#address2Author">

<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Author"/>

<address2Author/>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#address2Font">

<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Author"/>

<address2Font>Helvetica</address2Font>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#address2Justify">
<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Author"/>
<address2Justify>center</address2Justify>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#address2Size">
<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Author"/>
<address2Size>20</address2Size>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#affilateAuthor">
<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Author"/>
<affilateAuthor/>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#affilateFont">
<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Author"/>
<affilateFont>Helvetica</affilateFont>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#affilateJustify">
<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Author"/>
<affilateJustify>center</affilateJustify>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#affilateSize">
<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Author"/>
<affilateSize>20</affilateSize>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#authorFont">
<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Author"/>
<authorFont>Helvetica</authorFont>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#authorJustify">
<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Author"/>
<authorJustify>center</authorJustify>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#authorName">

<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Author"/>

<authorName/>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#authorSize">

<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Author"/>

<authorSize>24</authorSize>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#categoryFont">

<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#CatSubDes"/>

<categoryFont>Times New Roman</categoryFont>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#categoryJustify">

<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#CatSubDes"/>

<categoryJustify>both</categoryJustify>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#categorySize">

<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#CatSubDes"/>

<categorySize>18</categorySize>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#emailAuthor">

<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Author"/>

<emailAuthor/>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#emailFont">

<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Author"/>

<emailFont>Helvetica</emailFont>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#emailJustify">

<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Author"/>

<emailJustify>center</emailJustify>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#emailSize">

<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Author"/>

<emailSize>24</emailSize>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#generalFont">
<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#GeneralTerms"/>
<generalFont>Times New Roman</generalFont>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#generalJustify">
<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#GeneralTerms"/>
<generalJustify>both</generalJustify>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#generalSize">
<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#GeneralTerms"/>
<generalSize>18</generalSize>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#generalTerm">
<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#GeneralTerms"/>
<generalTerm/>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#hasAbstract">
<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Abstract"/>
<hasAbstract/>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#hasCategory">
<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#CatSubDes"/>
<hasCategory/>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#hasGeneral">
<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#GeneralTerms"/>
<hasGeneral/>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#hasKeyword">
<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Keywords"/>
<hasKeyword/>

</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#hasReferences">
<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Section"/>
<hasReferences/>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#hasSection">
<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Section"/>
<hasSection/>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#hasSubSection">
<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Section"/>
<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#SubSection"/>
<hasSubSection/>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#hasSubSubSec">
<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#SubSubSection"/>
<hasSubSubSec/>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#hasTable">
<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Section"/>
<hasTable/>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#keywordFont">
<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Keywords"/>
<keywordFont>Times New Roman</keywordFont>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#keywordJustify">
<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Keywords"/>
<keywordJustify>both</keywordJustify>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#keywordSize">
<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Keywords"/>
<keywordSize>18</keywordSize>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#phoneAuthor">

<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Author"/>

<phoneAuthor/>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#phoneFont">

<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Author"/>

<phoneFont>Helvetica</phoneFont>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#phoneJustify">

<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Author"/>

<phoneJustify>center</phoneJustify>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#phoneSize">

<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Author"/>

<phoneSize>20</phoneSize>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#referencesFont">

<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Section"/>

<referencesFont>Times New Roman</referencesFont>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#referencesJustify">

<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Section"/>

<referencesJustify>left</referencesJustify>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#referencesSize">

<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Section"/>

<referencesSize>18</referencesSize>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#sectionFont">

<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Section"/>

<sectionFont>Times New Roman</sectionFont>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#sectionJustify">

<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Section"/>

<sectionJustify>both</sectionJustify>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#sectionSize">

<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Section"/>

<sectionSize>18</sectionSize>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#subSectionFont">

<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#SubSection"/>

<subSectionFont>Times New Roman</subSectionFont>

</owl:NamedIndividual>

<owl:NamedIndividual
rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#subSectionJustify">

<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#SubSection"/>

<subSectionJustify>both</subSectionJustify>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#subSectionSize">

<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#SubSection"/>

<subSectionSize>18</subSectionSize>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#subSubSecFont">

<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#SubSubSection"/>

<subSubSecFont>Times New Roman</subSubSecFont>

</owl:NamedIndividual>

<owl:NamedIndividual
rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#subSubSecJustify">

<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#SubSubSection"/>

<subSubSecJustify>both</subSubSecJustify>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#subSubSecSize">

<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#SubSubSection"/>

<subSubSecSize>18</subSubSecSize>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#tableBold">
<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Section"/>
<tableBold>bold</tableBold>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#tableFont">
<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Section"/>
<tableFont>Times New Roman</tableFont>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#tableJustify">
<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Section"/>
<tableJustify>center</tableJustify>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#tableSize">
<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Section"/>
<tableSize>18</tableSize>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#titleBold">
<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Title"/>
<titleBold>bold</titleBold>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#titleFont">
<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Title"/>
<titleFont rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Helvetica</titleFont>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#titleJustify">
<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Title"/>
<titleJustify>center</titleJustify>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#titleSize">
<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Title"/>
<titleSize>36</titleSize>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://www.semanticdoc.org/ontology/2015/v1.6.owl#titleText">

<rdf:type rdf:resource="http://www.semanticdoc.org/ontology/2015/v1.6.owl#Title"/>

<titleText/>

</owl:NamedIndividual>

</rdf:RDF>

<!-- Generated by the OWL API (version 3.4.2) http://owlapi.sourceforge.net -->

APPENDIX 4

PHP CODE FOR UPLOADING ACM DOCUMENT AND RUN ADFCS

```
<?php
if($_POST["val"]==1)
{
if($_FILES["fileToUpload"]["size"] <= 0)
{
        echo "Sorry, No file has been chosen. or your file size is Zero.<br>";
        echo "<a href=acmchk.php>Back to file submission</a><br>";
}
else
{
        // Check file size
        if ($_FILES["fileToUpload"]["size"] > 50000000)
        {
                echo "Sorry, your file is too large.<br>";
                echo "<a href=acmchk.php>Back to file submission</a><br>";
        }
        else
        {
                $date_now = date("Ymd");
                $time_now = date("His");
                $id = $date_now."".$time_now;
                $target_dir = "uploads/".$id."";
                $target_out = $target_dir."/output.txt";
                $target_file = $target_dir ."/"."main.zip";
                $target_not3 = $target_dir ."/".$date_now."".$time_now.".n3";
                mkdir($target_dir);
                fopen($target_out, 'w');
```

```
        fopen($target_not3, 'w');
        //echo "dir ".$target_dir."<br>";
        //echo "out ".$target_out."<br>";
        //echo "fil ".$target_file."<br>";
        //echo "Not ".$target_not3."<br>";
if (move_uploaded_file($_FILES["fileToUpload"]["tmp_name"], $target_file))
{
        $zip = new ZipArchive;
        if ($zip->open($target_file) === TRUE)
        {
                $zip->extractTo($target_dir);
                $zip->close();
                echo "<h2>The report for your document.</h2>";
                //echo "The file ". basename( $_FILES["fileToUpload"]["name"]).
                //echo" has been uploaded.<br>";
                //exec("javac acm_doc.java");
                exec("java -Xmx256M acm_doc > ".$target_out );
                //exec("java -Xmx256M acm_doc");
                $handle = fopen($target_out , "r");
                if ($handle)
                {
                        echo "<table><tr>
                        <td><a href=acmchk.php>reCheck Again</a></td>
                <td><a href=dowpdf.php?id=".$id.">Download Report</a></td>
                <td><a href=acmnot.php>Rate System(<20s)</a></td>
                </tr></table>";
                echo "<table><tr><td>Parameter</td><td>Result</td></tr>";
                while (($line = fgets($handle)) !== false)
                {
                        //echo "<tr><td>".$line."</td><td></td></tr>";
```

```
                        $lin = split("-",$line);
                        //echo "<tr><td>".$lin[0]."</td><td>".$lin[1]."</td></tr>";
                        //echo "<br> str len . ".strlen($lin[1]);
                        if(strlen($lin[1])==9)
                        {
echo "<tr><td>".$lin[0]."</td><td>
<font color=#FF0000>".$lin[1]."</font> <img src='wrong.png'></td></tr>";
                        }
                        else if(strlen($lin[1])==7)
                        {
echo "<tr><td>".$lin[0]."</td><td>".$lin[1]." <img src='right.jpg'></td></tr>";
                        }
                        else
                        {
                        echo "<tr><td>".$lin[0]."</td><td>".$lin[1]."</td></tr>";
                        }
                }
        echo "</table>";
        fclose($handle);
echo "<table><tr><td><a href=acmchk.php>reCheck Again</a></td>
<td><a href=dowpdf.php?id=".$id.">Download Report</a></td>
<td><a href=acmnot.php>Rate System(<20s)</a></td></tr></table>";
        }
        else
        {
                echo "error opening the file.";
        }
}
else
{
```

```
                echo 'Unable to extraxt metadata of document.';
        }
        //echo "<br>END of java Execution<br>";
        //echo '<META HTTP-EQUIV="Refresh" Content="0; URL=acmrep.php">';
}
else
{
        echo "Sorry, there was an error uploading your file.";
}}}}
else
{
echo "<h2>No document has been chosen, please for checking your document go <a
href=acmchk.php>ACM Checking</a>";
echo "<h2>The report of your document will be shown here.</h2>";
}?>
```

APPENDIX 5

ADFCS USER MANUAL FOR GENERATED REPORT

Below is the guideline table for generated report by ADFCS, each parameter in report refer to a specific part, text, section, subsection, etc. inside document. If user upload a document and obtain any invalid result, this manual can help user to correct the format.

Report Parameter	Parameter Meaning / Text position in ACM document
Title Text	The title is the first paragraph in ACM document. This parameter is related to the Text that a document has. Title cannot be empty, If the title of document is wrong, you will get this parameter as invalid, all first letter in title must be capitalized, except than connectors like for, as, the. But if it come at the first, first letter must be capital. After title text there must be a section break for ADFCS to distinct title from other part of ACM SIG document.
Title Font	The font style of title must be Helvetica.
Title Size	The font size for title is 18 pt.
Title Bold	Title of document must be Bold
Title Justify	Title of document must be justified to center.
Author Name	Cannot be empty. If you get Author name as invalid, it maybe author name empty, is not in right order, or it cannot be detected by ADFCS.
Author Justify	The justification of author name must be centered.
Author Font	The author name font style must be Helvetica.
Author Size	The author name font size must be 12 pt.
Affilate Author	Cannot be empty, and it must be in new line.
Affilate Justify	The Affiliation Author must be justified to center.
Affilate Font	The Affiliation Author font style must be Helvetica.
Affilate Size	The Affiliation Author font size must be 10 pt.
Address1 Author	Cannot be empty. It must be in a new line.
Address1 Justify	The author first address must be justified to center.
Address1 Font	The author first address font style is Helvetica.
Address1 Size	The author first address font size is 10 pt.
Address2 Author	Cannot be empty, and it must be in new line.
Address2 Justify	The author second address must be justified to center.
Address2 Font	The author second address font style is Helvetica.
Address2 Size	The author second address font size is 10 pt.
Phone Author	Cannot be empty, and in new line.
Phone Justify	The author phone, justification must be centered.
Phone Font	The author phone, font style is Helvetica.
Phone Size	The author phone, font size it must be 10 pt.
Email Author	Cannot be empty, and in new line.
Email Justify	The author email must be justified to center.

Email Font	The author email font style must be Helvetica.
Email Size	The author email size is 12 pt.
Abstract Text	Cannot be empty, and it must be in the main body of text. First page of ACM SIG document, first column. The text is fix it must be written as ABSTRACT, otherwise you will get invalid result. If this parameter is invalid, all other parameter cannot be detected by ADFCS. The Abstract is the first paragraph after author section. At the end of author section, there must be a section break to indicate that the author section has been finished and the main body of text start. The Abstract text must be justified to Left and the Font Style is Times new Romans with Font size 12 pt. and it must be Bold.
Abstract Justify	Each paragraph in abstract must be justified (both side of paragraph are straight).
Abstract Font	All paragraph in abstract must be in Times new Romans Font style.
Abstract Size	The abstract paragraphs size are 9 pt.
Category Text	Cannot be empty. It has the same style as Abstract. Font style is Times New Romans, font size 12 pt., justified to left and Bold. The correct text of category must be written as Category and Subject Descriptors. If this parameter is wrong, all other parameter related to this part will be as invalid. It comes after ABSTRACT.
Category Justify	The paragraph in category section is justified.
Category Font	The font style of each paragraph is Times New Roman.
Category Size	The font size of each paragraph must be 9 pt.
General Text	Cannot be empty. It must be written as General Terms. Bold, justified, font size 12 pt., font style Times new Roman. Start after Category and Subject Descriptors.
General Justify	The General Terms must be justified.
General Font	The font style for paragraph is Times New Roman.
General Size	The font size must be 9pt.
General Term Designated	It is allowed to define the designated terms for General Terms, it must be chosen between 16 defined terms which is defined by ACM.
Keyword Text	Cannot be empty. The right text it must be written as Abstracts, font style Times New Roman, font size 12 pt., Bold, justify to left. If this parameter is invalid all other parameter related to abstract will be as invalid. Start after General Terms.
Keyword Justify	Keyword paragraph is justified.
Keyword Font	Keyword paragraph font style is Times New Roman.
Keyword Size	Keyword paragraph font size is 9 pt.
Section Text	ALL sections in ACM SIG document (e.g. INTRODUCTION, REFERENCES, etc.) have the format and Style. Any section in ACM SIG document must be all letter

	capitalized and justified to the left with font size 12 pt., font style Times New Roma, Bold. If the format change the ADFCS will not detect the text as Section.
Section Justify	Sections paragraph must be justified.
Section Font	Sections paragraph font style is in Times New Roman.
Section Size	Sections paragraph font size must be 9 pt.
SubSection Text	Subsections in ACM SIG document comes at level 2 and have the same style as sections but instead of all capitalized just the first letter of words are capitalized. Font size 12 pt., font style Times New Roman., Justified to left, and bold.
SubSection Justify	Subsections paragraph must be justified.
SubSection Font	Subsections paragraph font style must be in Times New Roman.
SubSection Size	Subsections paragraph font size must be 9 pt.
SubSubSection Text	Subsubsections must be Italic, justified to left, font size 11pt., font style Times New Roman.
SubSubSection Justify	Subsubsections paragraph must be justified.
SubSubSection Font	Subsubsections paragraph font style must be Times New Roman.
SubSubSection Size	Subsubsections paragraph font size must be 9 pt.
References Text	Cannot be empty. It must be written as REFERENCES abd it is the last section in ACM SIG documents, it must be Justified to left, font style Times New Roman, font size 12 pt., and Bold. If this parameter is invalid all other parameter related to references will be counted as invalid.
References Justify	All paragraphs in references section must be justified to left.
References Font	References paragraphs font style must be Times New Roman.
References Size	References paragraphs font size must be 9 pt.
Table Text	Cannot be empty.
Table Justify	The title of tables must be justified to center.
Table Font	The title of table's font style must be Times New Roman.
Table Size	The title of table's font size must be 9 pt.
Table Bold	The bold property is enabled for title of tables.
Page Width	The width of the page must be 18 centimeter.
Page Hight	The hight of page must be 23.5 centimeter.
Margin Top	The top margin must be 1.9 centimeter.
Margin Right	The right margin must be 1.9 centimeter.
Margin Buttom	The buttom margin must be 2.54 centimeter.
Margin Left	The left margin must be 1.9 centimeter.
Margin Header	The margin header size will be arranged by ACM.
Margin Footer	The margin footer size will be arranged by ACM.
Margin Gutter	Margin header must be 0 centimeter.

APPENDIX 6

JAVA CODE FOR JENA REASONNING

```
System.out.println("Report-Start Report");

tStart = System.currentTimeMillis();

String owlv_file = "ontology/2015/v1.6.owl";

String rule_file = "ontology/2015/rule.txt";

String not3_file = n3;

System.out.println("http://www.semanticdoc.org/" + owlv_file + "-Ontology file
location");

System.out.println("http://www.semanticdoc.org/" + rule_file + "-SIG rule file location");

System.out.println("http://www.semanticdoc.org/" + not3_file + "-Notation file location");

tEnd = System.currentTimeMillis();

jStart = System.currentTimeMillis();

Model data = FileManager.get().loadModel(not3_file);

List<Rule> rules1 = Rule.rulesFromURL(rule_file);

Reasoner reasoner = new GenericRuleReasoner(rules1);

OntModel model = ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM);

model.read(owlv_file);

InfModel infmodel = ModelFactory.createInfModel(reasoner, model, data);

jEnd = System.currentTimeMillis();

sStart = System.currentTimeMillis();

validity_check_report(infmodel);

sEnd = System.currentTimeMillis();

System.out.println("The elapsed time for Data Extraction (mili seconds)-" + (tEnd - tStart)
+ " milli sec");

System.out.println("The elapsed time for Jena Reasoner (mili seconds)-" + (jEnd - jStart) +
" milli sec");

System.out.println("The elapsed time for SPARQL query (mili seconds)-" + (sEnd - sStart)
+ " milli sec");

System.out.println("Report-End" );

System.out.println("Copyright 2015 . All Rights Reserved . SemanticDoc.org-# ");
```