I/S: A JOURNAL OF LAW AND POLICY FOR THE INFORMATION SOCIETY

Lost in Translation: Interoperability Issues for Open Standards

RAJIV SHAH* & JAY P. KESAN**

Abstract: Open standards are widely considered to have significant economic and technological benefits. These perceived advantages have led many governments to consider mandating open standards for document formats. Document formats are how a computer stores documents such as memos, spreadsheets, or slides. Governments are moving away from Microsoft's proprietary DOC format to open standard document formats, such as the OpenDocument Format (ODF) and Office Open XML (OOXML). The belief is that by shifting to open standards, governments will benefit from choice, competition, and the ability to seamlessly substitute different vendor products and implementations.

This paper examines whether open standards by themselves can deliver these promised benefits. The study examines interoperability for three document formats: ODF, OOXML, and DOC. The research assesses interoperability among different software implementations of each document format. For example, the implementations for ODF included KOffice, Wordperfect, TextEdit, Microsoft Office, and Google Docs. A set of test documents is used to evaluate the performance of other alternative implementations.

Our results show very significant issues with interoperability. The best implementations may result in

* Adjunct Assistant Professor, Department of Communication at the University of Illinois at Chicago; rajiv.shah@alumni.illinois.edu.

** Professor and H. Ross & Helen Workman Research Scholar, University of Illinois at Urbana Champaign; kesan@illinois.edu. The authors would like to thank Brett Hudspeth for his research assistance.

> *formatting problems, while the worst implementations actually lose information contained in pictures, footnotes, comments, tracking changes, and tables. Our findings also include specific scores for each implementation. There was considerable variation in how well each implementation for any particular document format performed. The raw scores for ODF, for example, ranged from 48 to 151.*
>
> *The results raise questions about the assumption that open standards guarantee interoperability and thereby promote competition and vendor choice. The interoperability issues are troubling and suggest the need for improved interoperability testing for document formats. The results also highlight the importance of ensuring interoperability for open standards. Without interoperability, governments will be locked into the dominant implementations for any standard. These results have significant policy implications for governments setting open standard policies.*

This material is based upon work supported by the National Science Foundation under Grant No. IIS-0429217. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

I. INTRODUCTION

Open standards have grown in prominence within the last few years. A simple definition of an open standard is a specification that is publicly available and freely implementable. Examples of open standards include the webpage language HTML, the image format JPEG, and transmission protocols such as FTP. Many observers have asserted that the growth, widespread use, and popularity of the Internet are largely the result of its reliance on open standards.[1]

Open standards are widely touted to have positive economic and technological benefits. When standards are open and freely available, it becomes possible for anyone to develop an interoperable implementation. Proponents of open standards focus on these benefits without qualifications or caveats. For example, Louis Suarez-Potts, the Community Manager of OpenOffice.org, states the following:

---

[1] Brian Kahin, *The Internet and the National Information Infrastructure, in* PUBLIC ACCESS TO THE INTERNET 3 (Brian Kahin & James Keller eds., MIT Press 1995).

> ODF, coupled with OpenOffice.org, shakes the foundations of monopoly, the status quo. With an easily usable open standard and Foss technology, one is not limited to a single vendor; there is, as the phrase puts it, no vendor lock-in. . . . It need not be OpenOffice.org. It can be any other application; it just has to be implementing the ODF or otherwise supporting it. Think of it as real consumer choice, or consumer freedom. We call this freedom "no vendor lock-in." And it's a freedom that goes beyond simple consumer choice . . . .[2]

Many advocates of open standards assume an open standard will lead to a vibrant, competitive market that removes vendor lock-in. This perception of economic, socio-political, and technical benefits flowing from open standards has garnered widespread support within industry, academic, and policymaker circles. Numerous reports have called for government policies that strongly encourage or mandate open standards. Some prominent examples include reports by the Berkman Center at Harvard, and the Committee for Economic Development, as well as the Office of Government Commerce in the United Kingdom.[3] Several state governments in the United States, such as Massachusetts, Minnesota, and New York, as well as other national governments, such as Belgium, Finland, France, Japan, and South Africa, are moving toward or considering open standards policies. These policies are led by a desire to save money and to give users greater flexibility with respect to IT.

This study examines whether open standards reduce vendor lock-in and promote competition by fostering multiple, interoperable implementations. We test the belief held by some, as represented by Suarez-Potts, that open standards will reduce vendor lock-in and provide more choices for users. This is accomplished by assessing the

---

[2] Louis Suarez-Potts, *OOXML and ODF and the Politics of Technology as Foss Comes of Age on the Desktop, in* O'REILLY OPEN SOURCE CONVENTION 2–3 (2008), *available at* http://www.oscon.com/oscon2008/public/schedule/detail/4772.

[3] BERKMAN CTR. FOR INTERNET & SOCIETY AT HARVARD LAW SCH., ROADMAP FOR OPEN ICT ECOSYSTEMS (2005), *available at* http://cyber.law.harvard.edu/publications/2005/The_Roadmap_for_Open_ICT_Ecosystems; COMM. FOR ECON. DEV., OPEN STANDARDS, OPEN SOURCE, AND OPEN INNOVATION: HARNESSING THE BENEFITS OF OPENNESS (2006); OFFICE OF GOV'T COMMERCE, OPEN SOURCE SOFTWARE: USE WITHIN UK GOVERNMENT (2004), *available at* http://www.epractice.eu/files/media/media_239.pdf.

effects of open standard document formats on interoperability. The first section of the article provides more background on open standards, vendor lock-in, interoperability, and why we chose to focus on document formats. The second section contains the methodology for our study. This is followed by our results and their implications.

II. BACKGROUND ON OPEN STANDARDS AND DOCUMENT FORMATS

Standards within information technology support interoperability, which allows complex systems to interact and share information. Standards provide "the digital equivalent of a common gauge for railroad tracks."[4] In the past, standards were created through two processes: de facto means and de jure means. De facto standards are those that achieve their dominance through public acceptance or market forces, e.g., QWERTY standard for typewriters. Within information technology, de facto standards are typically controlled by one company and are often termed proprietary standards to emphasize the ownership issue. A simple example of a proprietary standard is the Microsoft Word DOC document format. In contrast, other standards are developed by cooperation. Some of these may go through formal standard-setting organizations, such as the International Organization for Standardization (ISO) and become de jure standards.

Recently, there has been a rise in open standards within information technology. Open standards can emerge through either a de facto process or a de jure process. However, the most popular route is the use of consortia, such as the World Wide Web Consortium (W3C) or the Internet Engineering Task Force (IETF). As an example, Massachusetts defines open standards as "specifications for systems that are publicly available and are developed by an open community and affirmed by a standards body."[5] There are a number of other more detailed definitions of open standards.[6]

---

[4] Steven Lohr, *Plan by 13 Nations Urges Open Technology Standards*, N.Y. TIMES, Sept. 9, 2005, *available at* http://www.nytimes.com/2005/09/09/technology/09open.html.

[5] COMMONWEALTH OF MASS., ENTERPRISE TECHNICAL REFERENCE MODEL - SERVICE ORIENTED ARCHITECTURE (ETRM V. 5.1) (2011), *available at* http://www.mass.gov/anf/research-and-tech/it-pols-stnds-and-guidance/ent-architecture/etrm-v-5-1.

[6] Bruce Perens, *Open Standards: Principles and Practice,* PERENS, http://perens.com/OpenStandards/Definition.html (last updated Oct. 13, 2010, 16:58:05

A. OPEN STANDARDS

There is very limited research on the benefits of open standards. The existing research focuses on the development of open standards. For instance, Egyedi and Koppenhol argue that developing dual open standard document formats, ODF and OOXML, may impede innovation.[7] Simcoe examined the development process for open standards within the Internet Engineering Task Force (IETF) using a quantitative approach.[8] Nickerson and zur Muehlen studied the development of open standards for Web services.[9] Purao *et al.* conducted an empirical investigation of development processes for web service standards.[10] Fomin *et al.* proposed a meta-framework for standardization processes.[11] While the development of open source standards has been studied, that work is not directly related to the benefits of adopting open standards.[12]

The benefits of open standards can best be understood by using concepts from economics and strategic behavior. Open standards are publicly available and therefore they reduce barriers to entry for new competitors. As a result, because open standards foster the use of multiple vendors' products, they can limit vendor lock-in and thereby

---

PDT); Ken Krechmer, *The Meaning of Open Standards*, 4 INT'L J. OF IT STANDARDS & STANDARDIZATION RES. 43 (2006).

[7] Tineke Egyedi & Aad Koppenhol, *The Standards War Between ODF and OOXML: Does Competition Between Overlapping ISO Standards Lead to Innovation?*, 8 INT'L J. OF IT STANDARDS & STANDARDIZATION RES. 49 (2010).

[8] TIM SIMCOE, COMMITTEES AND THE CREATION OF TECHNICAL STANDARDS (2003); Tim Simcoe, *Delays and de Jure Standards: What Caused the Slowdown in Internet Standards Development*, *in* STANDARDS AND PUBLIC POLICY 260 (S. Greenstein & V. Stango, eds., 2007).

[9] Jeffrey V. Nickerson & Michael zur Muehlen, *The Ecology of Standards Processes: Insights from Internet Standard Making*, 30 MIS Q. (SPECIAL ISSUE) 467 (2006).

[10] Karthikeyan Umapathy & Sandeep Purao, *Representing and Accessing Design Knowledge for Service Integration*, IEEE INTERNATIONAL CONFERENCE ON SERVICES COMPUTING 67 (2008).

[11] Vladislav V. Fomin, *The Role of Standards in the Information Infrastructure Development, Revisited*, *in* PROCEEDINGS OF THE WORKSHOP ON STANDARD MAKING: A CRITICAL RESEARCH FRONTIER FOR INFORMATION SYSTEMS 302 (John L. King & Kalle Lyytinen eds., 2003).

[12] Patrick Y.K. Chau & Kar Yan Tam, *Factors Affecting the Adoption of Open Systems: An Exploratory Study*, 21 MIS Q. 1 (1997).

decrease switching costs. Vendor lock-in occurs when customers' buying choices are tied to an original purchase for a related product.[13] The economic argument is that reducing vendor lock-in reduces barriers to entry and increases substitutes, improving the buyer's position.[14] Vendors may try to fight this process; for example, West finds that when open standards reduce lock-in, vendors will try to use other strategies to lock in customers such as marketing, customer service, and product design.[15]

Governments are increasingly focusing on open standards. By our count, there are over fifty countries with electronic government policies that address open source or open standards.[16] A substantial portion of these policies include preferences or mandates for open standards. For example, some countries use Government Interoperability Frameworks that incorporate open standard policies.[17] Scholars have analyzed a few of these policies, including Sweden, Denmark, and Massachusetts.[18] However, there has been little empirical evaluation of the effects of open standard policies.

B. DOCUMENT FORMATS

---

[13] CARL SHAPIRO & HAL R. VARIAN, INFORMATION RULES: A STRATEGIC GUIDE TO THE NETWORK ECONOMY 103 (1998); Paul A. David, *Clio and the Economics of QWERTY,* 75 AM. ECON. REV. 332 (1985).

[14] Stanley M. Besen & Joseph Farrell, *Choosing How to Compete: Strategies and Tactics in Standardization*, 8 J. OF ECON. PERSP. 117 (1994).

[15] Joel West, *How Open is Open Enough? Melding Proprietary and Open Source Platform Strategies*, 32 RES. POL'Y 1259, 1279 (2003).

[16] JAMES A. LEWIS, CTR. FOR STRATEGIC & INT'L STUDIES, GOVERNMENT OPEN SOURCE POLICIES (2010), *available at* http://csis.org/files/publication/100416_Open_Source_Policies.pdf.

[17] Laura DeNardis, *E-Governance Policies for Interoperability and Open Standards*, 2 POL'Y & INTERNET 13 (2010), *available at* http://ssrn.com/abstract=1629833.

[18] Kim Anderson, Daniel Veit, Rony Megaglia & Helle Henriksen, *One Inch Wide and One Inch Deep: The Role of Policies in Shaping the Adoption of Open Standards and Software in Government*, *in* ELECTRONIC GOVERNMENT AND THE INFORMATION SYSTEMS PERSPECTIVE 100 (K. Andersen, E. Francesconi, Å. Grönlund & T. van Engers eds., 2010); Bjorn Lundell, *E-Governance in Public Sector ICT Procurement: What Is Shaping Practice in Sweden?*, EUROPEAN J. OF EPRACTICE 12 (2011); Rajiv C. Shah, Jay P. Kesan & Andrew Kennis, *Lessons for Government Adoption of Open Standards: A Case Study of the Massachusetts Policy*, 5 J. OF INFO. TECH. & POL. 387 (2008).

Virtually all open standard policies set by governments have emphasized the role of document formats. Document format standards specify how word-processing documents, spreadsheets, and presentations should be saved. A common document format for word processing is Microsoft's DOC format. Policies for document formats seem to offer a way for governments, such as Massachusetts, a way to avoid vendor lock-in.[19] The general goal of these policies is to move away from proprietary document formats, e.g., Microsoft's DOC format, to new open standard formats, such as the OpenDocument Format (ODF)[20] and Office Open XML (OOXML).[21]

To analyze vendor lock-in, we do not use an economic approach but instead follow a socio-technical methodology. Specifically, we examine whether users are locked-in to a specific implementation or whether they have choices between different implementations of a standard. This socio-technical approach requires attention and sensitivity to the social and technical aspects of lock-in. It should be noted that a purely economic analysis of vendor lock-in is untenable. There is little financial or market share data about document formats.

In analyzing document formats from a socio-technical perspective, it is clear that there are five critical sets of actors who may affect whether open standards ultimately reduce vendor lock-in. The first set is the actors involved in the standards development process; these actors can influence how a standard is specified. The second set is the actors involved in implementing the standards; these are typically software programmers. The third (and most often overlooked) set is the users. We deliberately conflate both users of software and purchasers of software. The final two sets are technological actors: the standard itself and the various implementations thereof.

The measure of vendor lock-in is whether users are able to seamlessly move documents between various implementations. Within software engineering, this problem is known as interoperability. Interoperability is defined as "the ability of two or more systems or components to exchange information and to use the

---

[19] Shah, Kesan & Kennis, *supra* note 18.

[20] ORG. FOR THE ADVANCEMENT OF STRUCTURED INFO. STANDARDS, OPEN BY DESIGN: THE ADVANTAGES OF THE OPENDOCUMENT FORMAT (ODF) (2006), *available at* http://www.oasis-open.org/committees/download.php/21450/oasis_odf_advantages_10dec2006.pdf.

[21] ECMA INT'L, OFFICE OPEN XML OVERVIEW (2007), *available at* http://www.ecma-international.org/news/TC45_current_work/OpenXML_White Paper.pdf.

information that has been exchanged."[22] Interoperability allows technologies to interface with each other easily, in addition to allowing data to move transparently. Once implementations are interoperable, users have the ability to seamlessly swap between them. In the area of document formats, this would allow users to move between Microsoft Office, Wordperfect, OpenOffice.org and other software in a seamless manner.

In sum, our efforts are focused on studying the benefits of open standards in terms of reducing vendor lock-in and promoting competition. To study this issue, we have chosen to focus on document formats, which are the leading exemplar of open standards. To assess the impact of vendor lock-in, we have chosen a socio-technical approach that considers the interoperability between implementations for a particular document format. The next section details how we measure interoperability.

III. RESEARCH METHODOLOGY

This study investigates if users can seamlessly move between various implementations of open standard document formats. As a starting point, the dominant implementation for ODF is OpenOffice.org, while the dominant implementation for OOXML is Microsoft Office.[23] By dominant we refer to their market share and regarded as the "best" implementations of the ODF and OOXML standards. Besides these, there are many other implementations that are capable of operating on the three major operating systems: Windows, Mac, and Linux. However, there is no research or data on the interoperability of these implementations.

Research on document formats has largely been focused on interoperability between document formats. For example, a German government study investigated interoperability between OOXML and Microsoft's DOC using several converters.[24] They found many problems in converting documents between ODF and DOC, such as missing information and incorrect formatting. The Danish

---

[22] IEEE, IEEE STANDARD COMPUTER DICTIONARY: A COMPILATION OF IEEE STANDARD COMPUTER GLOSSARIES 114 (1990).

[23]Alex Hudson, *OpenOffice.org to Apache*, WOAH! WORLD OF ALEX HUDSON (June 9, 2001), www.alexhudson.com/2011/06/09/openoffice-org-to-apache.

[24] WERNER LANGER, EXPERIENCES IN FORMAT CONVERSIONS (German Ministry of Foreign Affairs 2008).

government arrived at a similar conclusion in their study of interoperability between ODF, OOXML, and DOC.[25] While it is widely acknowledged that there are problems with interoperability *across* different formats, e.g., going from ODF to OOXML, there is an assumption here that all implementations of a particular document format produce, for example, the same ODF or OOXML document. In fact, it appears that governments believe (albeit entirely incorrectly) that various implementations of a document format will be interoperable, as evidenced by not considering the role of implementations in their policies.

This research investigates how interoperability functions for ODF and OOXML. Simply put, do the various implementations of a particular document format act alike? Or, are there incompatibilities that may cause loss of data or formatting issues? This study assesses how well electronic documents in three formats (ODF, OOXML, and DOC) can be transferred seamlessly across a variety of word processing programs (e.g., Microsoft Office, OpenOffice, and Wordperfect). The results are useful not only for evaluating individual implementations, but also for determining whether to adopt either ODF or OOXML as an open standard. After all, the benefits of open standards only accrue when vendor lock-in is reduced and users have a choice in selecting an implementation.

A. BACKGROUND ON INTEROPERABILITY AND CONFORMANCE TESTING

There are two general approaches for evaluating interoperability—conformance testing and interoperability testing. Conformance testing examines whether an implementation faithfully meets the requirement of a standard.[26] To perform conformance testing, a standard needs to have a conformance clause or statement that puts forth the criteria that must be met. After a set of criteria is spelled out, a test suite is developed. To test conformance, implementations then

---

[25] JENS JAKOB ANDERSEN, ODF AND OOXML IN DENMARK (National IT and Telecom Agency 2008), *available at* http://www.bigwobber.nl/wp-content/uploads/2009/06/20080228-ODFworkshop-JensJakobAndersen.pdf.

[26] James D. Kindrick, John A. Sauter & Robert S. Matthews, *Improving Conformance and Interoperability Testing*, 4 STANDARDVIEW, Mar. 1996, at 61, 62–63; Scott Moseley, Steve Randall & Anthony Wiles, *Experience Within ETSI of the Combined Roles of Conformance Testing and Interoperability Testing*, *in* PROCEEDINGS OF THE 3RD IEEE CONFERENCE ON STANDARDIZATION AND INNOVATION IN INFORMATION TECHNOLOGY 177, 179 (2003), *available at* http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1251206&isnumber=28013.

run the test suite. This provides an objective method for evaluating implementations and promotes portability and interoperability.

Conformance testing relies on a method of falsification testing. An implementation must execute various legal and illegal inputs. The output is then compared to "expected results." With this approach, a large number of tests and input combinations must be used. However, falsification can only prove that an implementation is not conformant; it cannot prove that an implementation is conformant.

Developing a set of conformance criteria and the related test suite can be difficult. To develop the conformance criteria, a standard must clearly set forth the relevant requirements. A basic test is then created for every requirement to see if the functionality is implemented. This test is then followed by other tests that examine the boundaries of that functionality, e.g., its minimum and maximum values. The combination of these tests is referred to as a test suite. Once the test suite is completed, every implementation can run the test suite to ensure its compliance.

The National Institute of Standards and Technology has been involved in developing tests for XML in cooperation with the World Wide Web Consortia (W3C). They have worked together to develop test suites containing thousands of individual tests for several XML technologies.[27] However, these groups have not developed tests for either ODF or OOXML.

A test suite for ODF has been started by researchers at the University of Central Florida. It covers the text document format, the presentation format, the drawing format, and the chart format, but it does not cover the spreadsheet format. Developing it has already taken over 300 hours. However, the test suite does not fully cover the specification, even in areas such as the text document format. Rob Weir, who works on developing the ODF standard for IBM, noted:

> A test suite is a daunting task. Some work was started at the University of Central Florida and picked up by the OpenDocument Fellowship but it has only a few hundred test cases. ODF, a 700 page specification probably has on the order of 5 testable statements per page, each one of which could require 4 test cases to test the main and edge conditions, positive and

---

[27] Carmelo Montanez, *XML Technologies Conformance Testing*, NAT'L INST. OF STANDARDS & TECH. (Mar. 30, 2004), http://www.itl.nist.gov/div897/docs/xml_tech_conformance.html.

> negative tests. So we're talking 14,000 test cases. Even if I'm off by a factor of 2 or 4, this is clearly a large undertaking. Project this out to OOXML's 6,000 pages and you would need 120,000 test cases.[28]

The difficulty of conformance testing for ODF and OOXML led us to focus on an interoperability testing approach. Within interoperability testing, one approach is to rely on a reference implementation, which is a fully functional implementation of a standard to which other implementations could be compared and evaluated. Ideally, a reference implementation would implement 100% of the standard, including optional parts. It would have a mode for strict compliance with the standard (i.e., it would not extend the standard with proprietary features). Other implementations could then be tested against the reference implementation. One advantage of interoperability testing for implementations is that it is not constrained by the requirements of a standard, and we can look at other factors.[29]

The reference implementation approach does save one from the time-consuming task of creating a test suite. However, it does not guarantee true interoperability. Interoperability is not commutative: if A=B and B=C, this does not assure that A=C. The only way to fully ensure interoperability is to use a full matrix system where every implementation is tested against every other implementation. This approach quickly becomes cumbersome as the number of implementations rises. A related method is to use "bake-offs" or "plug-fests," which are meetings of all the developers with their implementations for the purpose of testing interoperability.[30] By meeting together with various implementations, vendors can address interoperability issues. However, bake-offs or plug-fests require the cooperation of all vendors.

B. TESTING PROCESS

---

[28] Rob Weir, *The Anatomy of Interoperability*, ROB WEIR: AN ANTIC DISPOSITION (last updated Mar. 29, 2007), http://www.robweir.com/blog/2007/02/anatomy-of-interoperability.html.

[29] Kindrick, Sauter & Matthews, *supra* note 26.

[30] Peter Zehler, *Interoperability Testing for Internet Printing Protocol*, 6 STANDARDVIEW 180, 180 (1998).

This research tested the interoperability of ODF, OOXML, and DOC document formats using a reference implementation approach. For ODF, the test documents were developed in OpenOffice.org, which is currently the dominant implementation for ODF. For OOXML, the test documents were developed in Microsoft Office 2007 for Windows. For DOC, the test documents were developed in Microsoft Office 2007 for Windows. These are not reference implementations for ODF and OOXML in a true sense because they do not perfectly implement the standard. However, they act as de facto reference implementations because they are the dominant implementations that all developers seek compatibility with.

The next step was developing several test documents within each reference implementation. The test documents were then opened or imported into other implementations to assess how well other documents can read the standard. The testing then quantified any changes to the actual content (as this would be a major problem for the user) as well as changes to the layout of the document. The results show the compatibility of these implementations. We use the term compatibility when the testing only assesses reading the documents. The term interoperability is used when documents are saved in other implementations and then opened again in the reference implementation. This process is also known as round-tripping a document.

In developing the test documents, the goal is to test the majority of elements routinely used by the average user. We do not attempt an exhaustive study of every possible element. The goal here is to see whether these implementations would be "good enough" for most users. The test documents are based on features that are commonly used. Specific features were identified by examining various instructional materials for using office productivity software.

The current test involves five test documents for word processing. The first test document focused on commonly used formatting features; the specific elements are listed in Table 1. The second test document concerned the use of images; the specific criteria are listed in Table 2. The third test document focused on the use of tables; the specific criteria are listed in Table 3. Headers and footers were tested in the fourth test document, as shown in Table 4. The final test document contained a table of contents, footnotes, endnotes, comments, and tracking changes, as shown in Table 5. Tables 1-5 are listed in the Appendix. Each element in each test document was assigned a single point. The scores were then summed up across all five test documents.

Implementations were graded based on their ability to meet the criteria based on an accurate translation. For example, an ODF

implementation could get a perfect score of 148 if every feature was successfully met on all five test documents. If features were not met, then the implementation would not receive a point. In order to compare the differences between implementations, we have shown percentages instead of raw scores. The total percentage is shown in Tables 6, 7, and 8. Therefore, the higher the percentage, the more features that were met, and the better the compatibility.

The implementations are first scored based on how well they can read the test documents. The raw scores' percentages for each implementation are presented in Tables 6, 7, and 8 as the "read only" scores. It is important to recognize that the scores conflate compatibility with a standard and the lack of features/incomplete support in other applications. For example, TextEdit is not designed to handle images. KOffice has limitations in its handling for tables, images, and footers. So their lower scores may be due to the capabilities of the word processors, their implementation of either ODF or OOXML, or a combination of both.

The implementations are also scored based on how well they can read and then write documents. This is known as the "round trip" test. For this part of the test, the test document is opened in each implementation and then saved. The saved document is then opened in the reference implementation. The resulting document is then scored. This test provides insights into how well implementations can write ODF or OOXML. These results are found in Table 6 and 7 as the "read/write" scores.

The final part of the scoring focused on several metadata elements. These included attributes for styles, page numbers, tables of contents/headers, document information (e.g., time or number of words in documents), and tracking changes. Preserving these metadata elements was important for document usability. For example, after a document has been round tripped a page number should still appear at the bottom of the page. A key issue for users was whether the page number is still metadata or just a number. If a person inserts a page into the middle of the document, it would be expected that the page numbers throughout the document would increment appropriately. If the page number was metadata, then the update will happen correctly. If the page number was just a number and static, it will not update throughout this document. This part of the testing involved manually manipulating each document file to identify whether the document metadata was preserved. The results were then graded for each metadata element on a three-point scale. We chose this scoring format because, without a range of grades, it would obscure implementations that partly preserved metadata. A "3" was perfect, a "2" represented minor errors, a "1" represented major

errors, and a "0" was given for no metadata support. The results were then tabulated and are presented in Tables 6 and 7 as metadata score percentage.

For ODF, the test documents were created in OpenOffice.org 2.3. The criterion for implementations was to select a variety of implementations across several operating systems. The tested implementations included StarOffice, OpenXML/ODF Translator v3.0 Plug-in for Word 3, Sun Plug-in for Word 3.0, Wordperfect X4 (14), KOffice 1.6, Google Docs (May 2008), TextEdit 1.5, and AbiWord 2.4. For OOXML, the test documents were created in Office 2007 and tested in TextEdit 1.5, Pages 3.0.2, Office 2008 for Mac, ThinkFree (online application), Wordpad, and Novell's OpenOffice.org 3.0 with Open XML translator plug-in. For DOC, the test documents were created in Office 2007 and tested in TextEdit 1.5, Pages 3.0.2, Office 2008 for Mac, OpenOffice.org 3.0, Google Docs, KOffice 1.6, Wordperfect X4 (14), and AbiWord 2.4.

Table 9
Tested Implementations

	ODF	OOXML	DOC
Reference Implementations	OpenOffice.org 2.3	MS Word 2007	MS Word 2007
Other Implementations	AbiWord 2.4	Novell's OpenOffice.org 3.0	AbiWord 2.4
	OpenXML/ODF Translator v3.0 Plug-in for Word 3	Office 2008 (Mac)	Google Docs
	Google Docs	Pages 3.0.2**	KOffice 1.6**
	KOffice 1.6	TextEdit 1.5	Office 2008 (Mac)
	MS Word 2007 SP 2	ThinkFree Office	OpenOffice 3.0
	Novell's OpenOffice.org 3.0	Wordpad 7	Pages 3.0.2
	StarOffice	Wordperfect X4 (14)**	TextEdit 1.5
	Sun Plug-in for Word 3.0		Wordperfect X4 (14)
	TextEdit 1.5		
	Wordpad 7		
	Wordperfect X4 (14)**		

** Only supports reading document format and not writing to the document format

IV. RESULTS

Table 6
The Results of the ODF test; Scores for ODF Implementations

Implementation	Read Only	Round Trip	Metadata
OpenOffice.org	100%	100%	100%
OpenOffice.org Novell	99%	100%	100%
StarOffice	99%	100%	100%
MS Word 2007	95%	95%	93%
Sun Plug-in for Word	95%	95%	93%
ODF Translator Plug-in	93%	95%	100%
Wordperfect	80%	N/A	N/A
KOffice	80%	82%	67%
Google Docs	78%	86%	20%
WordPad	42%	37%	0%
TextEdit	37%	35%	0%
AbiWord	32%	38%	0%

There are no independent implementations that offer 100% compatibility with OpenOffice.org. It was surprising to see a difference between OpenOffice.org and StarOffice in the read only test. StarOffice, a commercial product, uses the same source code as the freely available OpenOffice.org, but offers some additional third party licensed components. The lost points are attributed to StarOffice not having the correct number of pages. However, as the read/write (round trip) results show in Table 6, this issue disappeared when the document was reopened in OpenOffice. In sum, even though both implementations share the same codebase, there were slight differences in their implementations of ODF when tested.

The best compatibility was found with the two plug-ins for Microsoft Word. While these plug-ins were developed independently, they offer similar results. Both offer good compatibility (>90%) with an assortment of minor formatting issues. Wordperfect and KOffice offer fair compatibility (>80%) with numerous issues. Wordperfect is not interoperable because it is not capable of writing ODF documents. Google Docs, TextEdit, and AbiWord have significant problems correctly reading the test documents. Specifically, KOffice has lots of minor problems with images, tables, and headers and footers. Wordperfect also has minor formatting problems, especially with

tables and headers and footers. Google Docs has significant problems reading test documents, but performed much better in the round trip test. This result is further discussed in the next section. Abiword and TextEdit all contain numerous problems, which are so extensive that information present in tables, headers, footers, comments, and incorporated images is completely lost.

The metadata test showed that only the Microsoft Word plug-ins offer good results as other implementations begin to lose important amounts of metadata. For example, KOffice does not support tracking changes, and hence, that information is lost.

The results of the OOXML test can be found in Table 7.

Table 7
Scores for OOXML Implementations

Implementation	Read Only	Round Trip	Metadata
Office 2007	100%	100%	100%
Office 2008 (Mac)	99%	100%	100%
OpenOffice.org	97%	97%	80%
Pages	96%	N/A	N/A
Wordperfect	77%	N/A	N/A
ThinkFree Office	68%	76%	53%
Wordpad 7	64%	50%	
TextEdit	35%	38%	0%

OOXML had similar results with no 100% compatibility with implementations other than Microsoft Office for Windows (2003 or 2007). Microsoft Office 2008 for Mac had a slight issue with the number of pages for a test document. This was an unanticipated result, because it was expected that Microsoft would be able to ensure 100% compatibility between its two implementations of OOXML. However, as with the differences between StarOffice and OpenOffice.org, the slight issues disappeared when the document was reopened in Microsoft Office 2007. This was still unexpected because, although both OOXML implementations do not share a common codebase, they are developed within the same organization, which should allow them to minimize interoperability issues. Novell's version of OpenOffice.org with its plug-in translator for OOXML provided good compatibility (>90%). Apple's Pages word processor also provided good compatibility, but the application is not interoperable. Pages and Wordperfect can only read OOXML

documents; they cannot write OOXML documents. ThinkFree office performed poorly, but did a bit better on the round trip test. TextEdit performed poorly among OOXML implementations.

The metadata test also showed that all other implementations began to lose information. OpenOffice.org lost a little metadata, while ThinkFree Office had more substantial information loss, and TextEdit did not maintain any metadata.

The results of the DOC test can be found in Table 8.

Table 8
Scores for DOC Implementations

Implementation	Read Only	Round Trip	Metadata
Office 2007	100%	100%	100%
Office 2008 (Mac)	99%	100%	100%
OpenOffice.org	98%	99%	93%
Pages	97%	95%	80%
Wordperfect	95%	95%	47%
KOffice	81%	N/A	N/A
AbiWord	55%	52%	40%
Google Docs	73%	78%	0%
TextEdit	74%	58%	0%

The DOC format interoperability testing revealed several good implementations for DOC. The Mac version of Office, OpenOffice.org, Pages, and Wordperfect all offered good compatibility with the DOC format. Interestingly, none of the implementations offered fair compatibility in the round trip test, but a number offered poor compatibility, including AbiWord, Google Docs, and Textedit.

V. DISCUSSION AND IMPLICATIONS

The implications of the empirical results are in two areas. The first is interoperability for document formats. The second is broader and concerns governmental policy for open standards.

A. INTEROPERABILITY FOR DOCUMENT FORMATS

There are several significant implications for document formats that flow from these tests. They include the lack of 100%

interoperability between implementations, the lack of independent implementations, and the overall performance of OOXML implementations.

The first issue concerns 100% compatibility or 100% interoperability for document formats. A 100% score for compatibility (read only) was not found. A 100% score for interoperability occurred between related implementations (e.g., either StarOffice and OpenOffice or Microsoft Office 2008 (Mac) and Microsoft Office 2007), but never between independent implementations. This result highlights the complexity of attaining complete (100%) substitutability of implementations. The only way to ensure full 100% fidelity is to use the leading implementations exclusively. Mixing implementations ensures that users will not realize full fidelity when transferring documents between various implementations.

The cause of the incompatibility of OpenOffice.org and StarOffice may not be related to document formats. All word processors have the issues of pixel level compatibility. Slight changes in spacing can happen because of variations in the font-rendering ability of word processors. These slight changes are not due to the document format. Nevertheless, users may blame document formats for these problems. These differences disappeared in the round trip test when documents were reopened in their original implementation. Nevertheless, developers will need to work together to minimize this problem, so users can have multiple interoperable implementations.

The question is often raised whether 100% interoperability should be the goal. After all, interoperability with HTML is not 100%; different web browsers may render websites differently. Should governments accept 99% or 90% document interoperability? Many people utilize alternatives to Microsoft Office for the DOC format, such as Google Docs, that are not 100% interoperable. Some urge that interoperability is one of many factors for governments to consider in choosing an implementation. If governments are seeking 100% fidelity, then they should use a format designed for preserving information and formatting, e.g., PDF. The problem with a reliance on PDF is that documents cannot be manipulated and edited; PDF is the last stop on the train. PDF is only appropriate for a small fraction of documents that are deemed "finished." A substantial set of working documents will not be saved in PDF, but in another editable format, whether that is DOC, ODF, or OOXML. Governments need to be able to flawlessly access these documents.

We believe that it is important for interoperability scores to be close to 100%. We recognize that 100% interoperability for every possible feature may be impossible; however, we still believe it is important for governments to push for this. There are two main

reasons. First, without 100% interoperability the value of these document formats as archival information is significantly reduced. Governments need to know that their documents will be readable and usable in timelines of 10, 50, 100 years, and longer. With an open standard, anyone could implement the document format. This means that someone fifty years from now will be able to implement the standard and read the documents. However, if no one else can develop an independent implementation that is 100% interoperable, this suggests users will be locked-in to the original implementation.

A second reason for interoperability close to 100% stems from network effects. Document formats gain strength when more people use them. They are an example of a network technology.[31] Consider two word processors, one with an 80% market share and the other with 20%. Network effects suggest that everyone will want to be interoperable with the 80%. This makes the more widely used word processor more valuable to each user and the less widely used one less valuable, thus leading to people to move towards the more widely used word processor. Ultimately, network effects will push the 20% to become interoperable or lead users to abandon the word processor with less market share. If interoperability problems exist, users will shift towards the most widely used one (who wants to switch between different word processors?). It is this dynamic that has made Microsoft Office dominant on desktops with 95% of the revenue from Office suites.[32]

Nevertheless, there are valid reasons for implementations to not offer 100% interoperability. For example, an implementation may not want to support all the features in the standard. However, there need to be some independent implementations with close to 100% interoperability. Most people need 100% interoperability for document formats for many reasons, ranging from actual problems to avoiding potential problems. It is for this reason that governments have an interest in improved interoperability.

To achieve close to 100% interoperability, an emphasis needs to be placed on conformance and interoperability testing. There are two steps governments can take to assist this process. First, governments can promote testing by emphasizing that interoperability is an

---

[31] Michael L. Katz & Carl Shapiro, *Systems Competition and Network Effects*, 8 J. OF ECON. PERSP. 93 (1994).

[32] *Rivals Set Their Sights on Microsoft Office: Can They Topple the Giant?*, KNOWLEDGE@WHARTON (Aug. 22, 2007), http://knowledge.wharton.upenn.edu/article.cfm?articleid=1795.

important facet of their procurement decisions. Hopefully this would push developers to improve their testing. Second, governments can directly support testing by either funding testing or developing conformance tests themselves. For example, the National Institute of Standards and Testing has a history of developing conformance standards for XML.

A second implication for document format interoperability is the lack of independent implementations for ODF and OOXML. Users are limited to choosing between Microsoft Office and OpenOffice.org. In the case of ODF, the only implementations that performed well were the plug-ins for Microsoft Office. In the case of OOXML, the only implementation that performed well was a special version of OpenOffice.org developed by Novell that runs only on Windows and on Novell's version of Linux named SUSE Linux. The other independent implementations either lack interoperability (Pages, Wordperfect) or provide poor performance (TextEdit, AbiWord).

The lack of independent implementations that can offer good performance is troubling. Users that require features such as footnotes, page numbers, and tracking changes must choose between Microsoft Office and OpenOffice.org. Users also have to deal with minor formatting glitches in the exchange of ODF and OOXML documents. The only way to eliminate these minor formatting problems is to "standardize" an organization on one implementation of ODF and/or OOXML.

The poor performance by open source implementations is significant. The results here indicate that if people want open source implementations, they need to provide more resources to these projects. One such example is the Netherland non-profit NLnet's funding of AbiWord aimed at improving ODF filters.[33]

The final implication for document format interoperability stems from the surprisingly good results for OOXML implementations. Critics of OOXML have argued that it is too complex and difficult to implement. While OOXML is a long and complex standard, it offers good compatibility nonetheless. Our results suggest that implementations of OOXML work as well as implementations of ODF in the read only test. At the level of basic word-processing that we examined, neither standard had a dominant advantage over the other in terms of compatibility scores. While ODF has had a head start that has led to more implementations, there appears to be no reason why

---

[33] Ryan Paul, *AbiWord Project Gets Funding for ODF Improvements*, ARS TECHNICA (Jan. 7, 2009, 1:43 PM), http://arstechnica.com/open-source/news/2009/01/abiword-project-gets-funding-for-odf-improvements.ars.

OOXML cannot catch up. After all, several developers have provided independent implementations that are capable of reading OOXML. The next step would be to push developers to save or write into OOXML, so complete interoperability can be provided.

An interesting comparison and historical perspective is offered by the results for the DOC format in Table 8. The DOC format has been widely used for a number of years, although the exact specification has been kept by Microsoft (until recently). It is interesting to note that the implementations offer either good compatibility or poor compatibility. Our belief is that this bifurcation probably occurs because implementations have a lack of willingness to incorporate all the features of Microsoft Office and a lack of resources to adequately understand and implement the DOC format. As a result, there are two classes of word processors, those that can compete with Microsoft Word and those that offer a second tier experience. We believe a similar process will occur for implementations of ODF and OOXML. As a result, it is likely that there will only be a handful of good implementations for either document format.

B. IMPLICATIONS FOR OPEN STANDARD POLICIES

There are many governments with open standards policies and many more considering these policies, as noted earlier. Governments see open standards as a way to eliminate vendor lock-in and lower costs. For example, DeNardis argues that open standards can and should be promoted by government.[34] She believes procurement policies of government can be used to push open standards, as they are the least interventionist method for governments to use in standards setting. She has put forth recommendations for how governments can adopt open standard policies.[35] However, the results for document formats show that these government policies are misguided in their current form. The results show open standards will not automatically result in lower costs, more choices, and flexibility. Simply put, open standards are not a silver bullet. The ODF open standard lacks multiple independent interoperable implications. Open standards do not ensure interoperability; therefore, there is no guarantee of vendor choice and resultant price competition, which is the policy objective that caused governments to embrace open standards in the first place. In sum, any policy option that blindly

---

[34] DeNardis, *supra* note 17, at 22.

[35] *Id.* at 22–30.

pursues open standards as an end in itself will not gain any substantial benefits.

While we agree open standards can be beneficial, governments should ultimately be focused on the outcomes of open standards, i.e., competition in the marketplace. Competition with multiple interoperable implementations is the best measure of success. The results of the DOC format testing show that the market can overcome proprietary closed standards. Despite Microsoft's wishes, a number of implementations have provided good compatibility with the DOC format. In fact, at this point in time, there are more good implementations of the DOC format than of either OOXML or ODF format. This shows how vendors can overcome proprietary formats and how multiple independent implementations can arise without open standards. It is this form of competition that governments need to encourage and sustain.

It is more desirable to have multiple vendors and a closed standard than one vendor with an open standard. To this end, we suggest government policies should focus on running code, which is shorthand for multiple independent interoperable implementations. Running code assures governments of multiple vendors and competition. Without running code, governments are going to be in a relationship with one vendor and thus suffer the costs of vendor lock-in. Consequently, open standard policies should be written with an emphasis on running code and having multiple implementations of a standard. The focus should be on this competition and not on a standard that has been anointed "open" but is without interoperable implementations. We have expounded on this argument elsewhere.[36]

Governments can add a running code requirement to their policies and interoperability frameworks. It will allow them to meet the goals of reducing vendor lock-in and reducing cost. A running code requirement puts an emphasis on how the standard is actually being used. We believe if adopters of open standards insist on running code, software developers and vendors will further support open standards and their interoperability. The result will be an array of economic and technological benefits.

---

[36] Rajiv Shah & Jay Kesan, *Running Code as Part of an Open Standards Policy*, 14 FIRST MONDAY 1 (2009), *available at* http://ssrn.com/abstract=1416115.

VI. LIMITATIONS

There are a number of limitations to this study that need to be considered. First, there is an assumption that the chosen reference implementations, e.g., OpenOffice.org and Microsoft Office 2007, accurately implement the standards. However, there is no evidence that either of these standards is 100% compliant with the published ISO/IEC standards. Moreover, Microsoft has readily admitted that they will not support the ISO/IEC version of OOXML until Microsoft Office 2012.[37] As a result, other implementations could be compliant with the actual standards, but lose points because the chosen reference implementation for our study does not conform to the standard.

Second, our study conflates several aspects of document fidelity together in scoring implementations. These aspects include compatibility with a document format, full support of tested features, and the issue of pixel level compatibility. As a result, lost points may not be related to document format compatibility, but to other issues. Nevertheless, we believe all three of these issues must be addressed to ensure interoperability.

Third, this testing was limited to word processing. Both ODF and OOXML have a much wider scope and cover other document types, such as spreadsheets and presentations. Consequently, these results are only applicable to the word processing aspects of these standards. We would expect worse results for these other aspects simply because there has been more emphasis by the developer community to ensure interoperability for word processing.

Fourth, this study used versions of Microsoft Office and OpenOfffice.org that have been since updated. There is a chance that interoperability issues could have been improved in new versions. However, while implementations have been updated, the open standards themselves have also been updated. As noted above, it will not be until Office 2012 that Office will use the ISO/IEC version of OOXML. Similarly, the ODF was recently revised to version 1.2 in March 2011.[38] If a standard is changing, implementations must change as well. As a result, interoperability issues will persist for ODF and OOXML.

---

[37] Ryan Paul, *Microsoft: Office 15—Not 2010—to be Fully OOXML Compliant*, ARS TECHNICA (Apr. 7, 2010, 9:10 AM), http://arstechnica.com/microsoft/news/2010/04/microsoft-office-15not-2010to-be-fully-ooxml-compliant.ars.

[38] Posting of Robin Cover, robin@oasis-open.org, to office@lists.oasis-open.org (Mar. 24, 2011), http://lists.oasis-open.org/archives/office/201103/msg00089.html.

Finally, this testing focuses on a homogenous environment with only one standard. In a real world setting, implementations deal with many standards, such as DOC, ODF, and OOXML. This requires implementations to continually convert between these document formats, which could introduce other errors or formatting problems.

VII. CONCLUSIONS

This study highlights the importance of interoperability for open standards. This was illustrated by studying various implementations of ODF, OOXML, and DOC. To capture the perceived economic and technological benefits of open standards, there is a need for multiple independent, interoperable implementations. This study shows that optimism toward open standards and the rush to mandate open standards need to be tempered because interoperability issues can dramatically reduce the advantages of open standards.

This study focused on testing a subset of document formats with basic word processing features. The results here are discouraging for those seeking the promised benefits of open standards, i.e., avoiding vendor lock-in and fostering competition. The only implementations of ODF that provide good compatibility with OpenOffice.org were the Microsoft Office plug-ins. Similarly, the only implementation of OOXML that provides good compatibility with Microsoft Office 2007 was OpenOffice.org with the Novell plug-in. Our results show that while the best implementations may result in formatting problems, the worst implementations actually lose information found in pictures, footnotes, comments, tracking changes, and tables.

It is surprising and ironic that the best implementations of ODF are when using Microsoft Office. After all, Microsoft was slow to adopt the competing format. Similarly, the best implementation of OOXML is OpenOffice.org. The domination of Microsoft Office and OpenOffice.org is especially troubling because it leaves users with little choice. The much vaunted "choice" promised to users of open standards has left them with a duopoly for both ODF and OOXML. This suggests that governments adopting either of these standards will be forced to choose between the two implementations as well.

While the results here may be discouraging, we believe improvements can occur. Supporters of both ODF and OOXML have suggested improved conformance and interoperability testing. On the ODF front, there are organizations, such as OASIS, that are focusing on the interoperability problems. Nevertheless, governments and other interested organizations need to encourage this testing. Without more pressure and funding for testing, the promise of ODF and

OOXML will be lost. Instead, users of these standards will be locked into the dominant implementations of OpenOffice.org for ODF and Microsoft Office for OOXML.

Vendors may argue that interoperability is too costly. For example, they may urge that document formats that are mostly compatible are good enough. Achieving interoperability may involve costs in development, e.g., updating the technical specification and testing of implementations through plug-fests and the like. Nevertheless, when products are not interoperable, users bear the burden of increased switching costs and vendor lock-in.

There is still much research and testing to be done. Each of these implementations is continually being improved and therefore needs to be continually reassessed. Future research needs to expand the tests to spreadsheets and presentations. This work serves as a first step in providing empirical data on interoperability for ODF and OOXML. It is hoped that this will serve as a wake-up call to governments and developers to improve the current state of interoperability for document formats. After all, it is only with interoperability that the promise of open standards will be achieved.

Finally, this study should also force governments to pause before blindly embracing open standards. What governments need to understand is that having open standards policies does not ensure interoperability and, therefore, there is no guarantee of vendor choice and resultant price competition. Instead, governments need to encourage interoperability among implementations, not just simply embrace open standards. To this end, we suggest a requirement ensuring multiple independent interoperable implementations, known as running code, in open standard policies. It is through running code that governments can receive the benefits of choice and competition.

Appendix

Table 1
Test document #1, Basic Formatting

Correct Number of Pages
Margins
Left Justification
Center Justification
Right Justification
Tabs
Correct font
Font size
Single Spacing
1.5 Spacing
Double Spacing
Bold
Underline
Italics
Bold-Underline
Bold-Italic
Italic-Underline
Bold-Italic-Underline
Single Strike Through
Double Strike Through
Small Caps
Superscript
Subscript
Color background
Color font
Hyperlink
Block Text (Full Justification)

Table 2
Test document #2, Images

Correct Number of Pages
JPEG Image Present
JPEG Image Positioned Correctly
JPEG Image Wrapped Correctly
BMP Image Present
BMP Image Positioned Correctly
BMP Image Wrapped Correctly
GIF Image Present
GIF Image Positioned Correctly
GIF Image Wrapped Correctly
Absolute Positioning Test

Table 3
Test document #3, Tables

Correct Number of Pages
Table Present
Table Positioned Correctly
Table Correct Rows/Columns
Table Borders Correct
Text And Characters in Cells
Bold Text in Cell
Italic Text in Cell
Underline Text in Cell
Combination Text in Cell
Red Background
Green Text
Yellow Background Violet Txt
Superscript
Subscript
Hyperlink
Single Strike
Double Strike
Small Caps
Vertical Splits
Horizontal Splits
Text Rotation**
Center Justification
Right Justification
Full Justification
Cell Center Alignment
Cell Bottom Alignment
Cell Top Alignment
Red Cell Fill
Picture in Cell
Cells The Correct Sizes

Table 4
Test document #4, Headers and Footers

Correct Number of Pages
Headers Exist
Two Different Headers
Header Bold
Header Italic
Header Underlined
Header Combos
Header Superscript
Header Subscript
Header Hyperlink
Header Background Colors
Header Font Colors
Header Different Fonts
Header Different Sizes
Header Strike Out
Header Double Strike
Header Crossed Out
Header Center
Header Left
Header Right
Header Full Justification
Header Date Fixed
Header Date Updating
Header Time Fixed
Header Time Updating
Header Author
Header Page Number
Header Page Count
Header Title
Header File Name
Header Word Count
Header Paragraph Count
Footers Exist
Footer Date Fixed
Footer Date Updating
Footer Time Fixed
Footer Time Updating
Footer Author
Footer Page Number
Footer Page Count
Footer Title
Footer File Name
Footer Word Count
Footer Paragraph Count

Table 5
Test document #5, Footnotes, Endnotes, Tracking Changes, Table of Content

Correct Number Of Pages
Footnotes Present
Footnotes Located Correctly
Endnotes Present
Endnotes Located Correctly
Table Of Contents Correct/Present
Tracking Changes Recorded
Tracking Additions
Tracking Deletions
Bold in Footnotes
Italics in Footnotes
Underlines in Footnotes
Bold-Italics in Footnotes
Bold-Underline in Footnotes
Italic-Underline in Footnotes
Bold-Italic-Underline in Footnotes
Superscript in Footnotes
Subscript in Footnotes
Colors in Footnotes
Small Caps in Footnotes
Hyperlinks in Footnotes
Bold in Endnotes
Italics in Endnotes
Underlines in Endnotes
Bold-Italics in Endnotes
Bold-Underline in Endnotes
Italic-Underline in Endnotes
Bold-Italic-Underline in Endnotes
Superscript in Endnotes
Subscript in Endnotes
Colors in Endnotes
Small Caps in Endnotes
Hyperlinks in Endnotes
Bulleted List Present
Numbered List Present
Bulleted List Correctly Leveled
Numbered List Correctly Leveled
Bold in Lists
Italics in Lists
Underlines in Lists
Bold-Italics in Lists
Bold-Underline in Lists
Italic-Underline in Lists
Bold-Italic-Underline in Lists
Superscript in Lists
Subscript in Lists
Colors in Lists
Fonts in Lists
Hyperlinks in Lists
Comments Present