University of Illinois College of Law

Law and Economics Working Papers

Lost in Translation: Interoperability Issues for Open Standards – ODF and OOXML as Examples

Rajiv Shah[*]   Jay P. Kesan[†]

[*]University of Chicago, rajiv.shah@alumni.illinois.edu

[†]University of Illinois College of Law, kesan@law.uiuc.edu

Lost in Translation: Interoperability Issues for Open Standards – ODF and OOXML as Examples

Rajiv Shah and Jay P. Kesan

Abstract

Open standards are widely considered to have significant economic and technological benefits. This has led many governments to consider mandating open standards for document formats. Document formats are how a computer stores memos or spreadsheets. 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 implementations.

This paper suggests that governments seeking the benefits of open standards need to consider the role of interoperability. Without multiple interoperable implementations, i.e., "running code", users will not gain the advantages of competition and substitutability. To highlight the issues around interoperability, we examined the interoperability issues around ODF and OOXML.

This research assesses interoperability among different software implementations of each document formats. For example, the implementations for ODF included KOffice, Wordperfect, TextEdit, Microsoft Office, and Google Docs. A set of test documents was used to evaluate the performance of other alternative implementations.

Our analyses show that there are significant issues with interoperability among various implementations. Users face numerous issues when transferring files between different implementations. While the best implementations may result in

formatting problems, the worst implementations actually lose information, e.g., information found in pictures, footnotes, comments, tracking changes, and tables. Our findings include specific scores for each implementation. There was considerable variation among how well each implementation performed. For ODF, the compatibility scores ranged from a raw score of 151 (100%–weighted percent) to 48 (55%–weighted percent).

We consider several implications of these results including the lack of perfect compatibility between implementations, the lack of good implementations outside of Windows, and the surprisingly good overall performance of OOXML implementations. The interoperability issues are troubling and suggest the need for improved interoperability testing for document formats. The results also highlight the importance of interoperability for open standards in general. Without interoperability, governments will be locked-in to the dominant implementations for either standard and in the process lose many of the benefits that might accrue from adopting an open standard in the first instance.

Lost in Translation: Interoperability Issues for Open Standards – ODF and OOXML as Examples

Rajiv Shah

University of Illinois at Chicago

rajiv.shah@alumni.illinois.edu

Jay P. Kesan

Professor & Mildred Van Voorhis Jones Faculty Scholar

Director, Program in Intellectual Property & Technology Law

University of Illinois at Urbana Champaign

kesan@illinois.edu

1

ABSTRACT

Open standards are widely considered to have significant economic and technological benefits. This has led many governments to consider mandating open standards for document formats. Document formats are how a computer stores memos or spreadsheets. 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 implementations.

This paper suggests that governments seeking the benefits of open standards need to consider the role of interoperability. Without multiple interoperable implementations, i.e., "running code", users will not gain the advantages of competition and substitutability. To highlight the issues around interoperability, we examined the interoperability issues around ODF and OOXML.

This research assesses interoperability among different software implementations of each document formats. For example, the implementations for ODF included KOffice, Wordperfect, TextEdit, Microsoft Office, and Google Docs. A set of test documents was used to evaluate the performance of other alternative implementations.

Our analyses show that there are significant issues with interoperability among various implementations. Users face numerous issues when transferring files between different implementations. While the best implementations may result in formatting problems, the worst implementations actually lose information, e.g., information found in pictures, footnotes, comments, tracking changes, and tables. Our findings include specific scores for each

implementation. There was considerable variation among how well each implementation performed. For ODF, the compatibility scores ranged from a raw score of 151 (100%--weighted percent) to 48 (55%--weighted percent).

We consider several implications of these results including the lack of perfect compatibility between implementations, the lack of good implementations outside of Windows, and the surprisingly good overall performance of OOXML implementations. The interoperability issues are troubling and suggest the need for improved interoperability testing for document formats. The results also highlight the importance of interoperability for open standards in general. Without interoperability, governments will be locked-in to the dominant implementations for either standard and in the process lose many of the benefits that might accrue from adopting an open standard in the first instance.

Lost in Translation: Interoperability Issues for Open Standards–ODF and OOXML as Examples

1. INTRODUCTION

Document formats based on open standards have emerged as a central issue for governments, software vendors, standards bodies, and policymakers. A number of governments around the world have enacted policies that strongly encourage or mandate open standards for document formats. These policies are allowing governments to move away from proprietary document formats, e.g., Microsoft's DOC format, to new open standard formats, such as the OpenDocument Format (ODF) (Organization for the Advancement of Structured Information Standards 2007) and Office Open XML (OOXML) (Ecma International 2007).

Open standard document formats, such as ODF and OOXML, are believed to provide a wealth of economic and technological benefits (Ghosh and Schmidt 2006; Berkman Center for Internet & Society at Harvard Law School 2005; Ditch 2007). Governments are pushing for the use of ODF and OOXML, because the open nature of these document formats fosters more competition for Office productivity applications, which is currently dominated by Microsoft. However, there is a lack of research studying the alternative implementations for ODF and OOXML. This research assesses the current state of interoperability for the software implementations of ODF and OOXML.

The benefits of open standards arise with competition. Competition requires the existence of independent, interoperable implementations of an open standard or "running code". The power of running code is that it offers substitutability. For example, organizations can switch their SMTP mail servers without it affecting how mail is sent or received. This provides

organizations with the power to choose the solution that best fits their needs. The key to this seamless swapping is interoperability. Without interoperability, competition is muted.[1]

ODF and OOXML are both relatively new open standards. ODF was approved as an ISO/IEC standard in 2006 (ISO/IEC 26300), while OOXML was approved in April 2008 (ISO/IEC 29500). The dominant implementation for ODF is OpenOffice.org, while for OOXML the dominant implementation is Microsoft Office.[2] Both standards have been incorporated in implementations that run on the three most significant operating systems, Windows, Mac, and Linux. However, this is no research or data on the interoperability of these standards. As a result, there is no data on whether there is 100% interoperability between the dominant implementations and other implementations.

Governments and organizations need information regarding interoperability for document formats. Within the United States, several states including Massachusetts, Minnesota, New York, and Texas have studied implementing ODF and OOXML. On an international basis, there are currently around 20 countries actively evaluating and implementing ODF and/or OOXML. These governments have not yet examined the interoperability issue, largely because they assume various implementations of a standard will be interoperable. However, our research shows that ODF as written by OpenOffice.org will not be read 100% correctly in other implementations, such as Microsoft Office or Wordperfect. This research quantifies the current state of interoperability for implementations of ODF and OOXML.

---

[1] As a helpful reviewer pointed out, in some situations, competition and multiple implementations may not arise because there are other significant barriers, e.g., technical barriers to entry in telecommunications equipment.

[2] OOXML in this study focuses on the Ecma standard that all implementations of OOXML use. The ISO is in the process of developing the official version of OOXML, which has a number of changes from the Ecma standard.

The existing research on document formats was conducted by governments considering the adoption of ODF or OOXML. A study by the Swedish Agency for Public Management found that Sun's StarOffice and Microsoft's Word 2003 supported the OOXML formats differently (Vestin 2003). Most other studies have not considered interoperability for one format, but instead focused on converting documents between ODF and other formats. A German government study investigated interoperability between OOXML and Microsoft's DOC using several converters (Langer 2008). They found many problems in converting documents between ODF and DOC. The Danish government arrived at a similar conclusion in their study of interoperability between ODF, OOXML, and DOC (Andersen 2008). 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 produce the same ODF or OOXML.

This research investigates how interoperability functions for ODF and OOXML. Simply put, do the various implementations act alike? Or are there incompatibilities that may cause the loss of data or formatting issues? The goal of this project is to assess how well electronic documents in a particular format, either ODF or OOXML, transfer across a variety of word processing programs using the same format (Microsoft Word, OpenOffice, StarOffice, Pages. . .). 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, adoption of open standards should hinge on the existence of multiple independent and interoperable implementations.

2. BACKGROUND ON INTEROPERABILITY AND CONFORMANCE TESTING

The best method to enhance interoperability is conformance testing (Kindrick, Sauter, and Matthews 1996; Moseley, Randall, and Wiles 2003). Conformance testing examines whether an implementation faithfully meets the requirement of a standard. 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 then developed. To test conformance, implementations then 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 and the output is then compared to "expected results". With this approach, a large number of tests and input combinations must be tested. However, falsification can only prove an implementation is not conformant; the test can't prove an implementation is conformant.

Developing a set of conformance criteria and the related test suite can be difficult work. To develop conformance criteria, a standard must be written in such a way that it is clear what requirements are being set forth. Then a basic test is created for every requirement to see if functionality is implemented. This test is then followed up with other tests that examined the boundaries of that function, e.g., minimum and maximum values. The combination of these tests is referred to as a test suite. Once the test suite is completed, then 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 developed test suites with thousands of individual tests for several XML technologies (National Institute of

Standards and Technology 2008). However, they 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 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. (Weir 2007)

The difficulty of conformance testing for ODF and OOXML led us to focus on an interoperability testing approach. One method 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 does not extend the standard with proprietary features). Other implementations could then be tested against the reference implementation. One advantage of testing implementations is that it is not constrained by the requirements of a standards, and we can look at other factors (Kindrick, Sauter, and Matthews 1996).

The reference implementation approach does save one from the time-consuming task of creating a test suite. However, it doesn't guarantee true interoperability. Interoperability is not commutative: If A=B and B=C this does not assure that A=C. The only way to ensure fully interoperability is a full matrix system where every implementation is tested against every other implementation. This approach quickly becomes cumbersome as the number of implementation rises. A related method is the use of "bake-offs", which is a meeting of all the developers with their implementations for the purpose of testing interoperability (Zehler 1998). By meeting together with various implementations, vendors can address interoperability issues. However, bake-offs require the cooperation of all vendors.

3. METHODOLOGY

This research tested the interoperability for ODF and OOXML document formats based on a reference implementation approach. For ODF, the test documents are developed in OpenOffice.org, which is currently the dominant implementation for ODF. For OOXML, the test documents are developed in Microsoft Office 2007 for Windows. These are not reference implementations 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 (this would be a major problem) as well as changes to the layout of the document. The results would provide data on the compatibility of these implementations.

We use the term compatibility, because our testing does not truly assess interoperability, which would require reading and writing.

In developing the test documents, our goal is to test what 90% of users use. We are not trying to test extremely complex elements, but elements that are routinely used. 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 and 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 test document #4 for ODF was different than the test document for OOXML. ODF tested 63 features, while OOXML tested only 29 features. The OpenOffice.org implementations provided more flexibility in allowing multiple headers and footers, which resulted in many more features being tested. 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.

The implementations are then scored based on how well they can read the test documents. The raw scores for each implementation are presented in Table 6 and Table 7. It's 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 can be due to the capabilities of the word processors and/or their implementation of either ODF or OOXML.

The raw score does not adequately measure the overall performance of an implementation, because some features are more important than others. For example, having the correct number of rows in a table is more important than having a subscript in a table. This led us to characterize each criterion as either major or minor. In tables 1 through 5, the major criteria are in bold. Also an emphasis on raw scores may favor features with many overall elements, e.g., headers. To emphasize the need to support basic formatting, i.e., test document #1, we then weighted each test document for a final weighted score. Test document #1 was worth 30% and the other four test documents were each worth 17.5%. The resulting score is a better indicator of the performance of an implementation than the raw score. And is a more accurate score for comparing the performance of ODF versus OOXML.

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, Clever Age Plug-in for Word 1.1, Sun Plug-in for Word 1.1, Wordperfect X4 (14), Koffice 1.6, Google Docs (May 2008), TextEdit 1.5, 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), Novell's OpenOffice.org 2.4 with Open XML translator plug-in, and Office 2003. Several versions of Microsoft Office were tested, because of the claims that OOXML was a developing standard and was hard to implement. This test sought to examine if Microsoft was capable of implementing OOXML on different platforms over time.

4. RESULTS

The results of the ODF test can be found in Table 6.

Implementation	Raw Score	Raw Score Percentage	Weighted Percent
OpenOffice.org	151	100%	100%
StarOffice	149	99%	97%
Sun Plug-in for Word	142	94%	96
Clever Age/MS Plug-in for Word	139	92%	94%
Wordperfect	122	81%	86%
Koffice	121	80%	79%
Google Docs	117	77%	76%
TextEdit	55	36%	47%
Abiword	48	32%	55%

Table 6. Scores for ODF Implementations

The results of our test are currently limited to testing the word processing for the ODF standard. There are no implementations that offer 100% compatibility with OpenOffice.org. It was surprising to see a difference between OpenOffice.org and StarOffice. StarOffice, a commercial product, uses the same source code as the freely available OpenOffice.org. StarOffice offers some additional third party licensed components. The lost points are attributed to StarOffice not having the correct number of pages. In sum, even though both implementations share the same codebase, when tested, there were slight differences in their implementations of ODF.

The best compatibility was found with the two plug-ins for Microsoft Word. Both of these plug-ins were developed independently, but they offer similar results. They both offer good compatibility with an assortment of minor formatting issues. Wordperfect and Koffice offer fair compatibility with numerous issues, while 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, especially with tables and headers and footers. Google Docs, Abiword, and TextEdit all contain numerous problems. The problems are so extensive that information is lost that is present in tables, headers and footers, comments and incorporated images.

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

Implementation	Raw Score	Raw Score Percentage	Weighted Percent
Office 2007	148	100%	100%
Office 2003	148	100%	100%
Office 2008 (Mac)	147	99%	99%
OpenOffice.org	141	95%	96%
Pages	142	96%	95%
Wordperfect	114	77%	84%
ThinkFree Office	101	68%	83%
TextEdit	52	35%	43%

Table 7. Scores for OOXML Implementations

OOXML had similar results with no 100% compatibility with implementations other than Microsoft Office for Windows (2003 or 2008). 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 would be expected that Microsoft would be able to ensure 100% compatibility between its two implementations of OOXML. While both implementations do not share a common codebase, they both 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. Apple's Pages word processor also provided good compatibility, but the

application is not interoperable. Pages can only read OOXML documents, it cannot write OOXML documents. Wordperfect and the online ThinkFree office provided fair compatibility with numerous problems.

5. IMPLICATIONS

There are several significant implications that flow from these tests. They include the lack of 100% compatibility between implementations, the lack of good implementations outside of Windows, and the overall performance of OOXML implementations.

A 100% compatibility score between implementations only occurred for Microsoft Office 2003 and 2007 for Windows. Every other implementation had minor differences. For example, even though OpenOffice.org and StarOffice share the same source code, there are minor differences in their compatibility for ODF. Similarly, the Mac version of Microsoft Office had a minor difference from the Windows version of Microsoft Office. This result highlights the complexity of attaining complete (100%) interoperability for document formats. This suggests that the only way to prevent interoperability issues 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 problem between OpenOffice.org and StarOffice may not be related to ODF. All word processors have to face 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 the difference is lost on users. Developers will need to work together to minimize this problem, so users have multiple interoperable implementations.

The only way 100% compatibility can be approached is with an emphasis on conformability and/or interoperability testing. First, governments should emphasize that compatibility is an important facet of their decisions. It is hoped that this would push developers to improve their testing. Second, governments should 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 from these tests is the lack of good performance for implementations other than Microsoft Office. The plug-ins for Microsoft Office are clearly better than any other independent implementations for the ODF format. This places Microsoft Office and the Windows operating system at a significant advantage compared to other operating systems. Users of other operating systems face lock-in to OpenOffice.org unless they wish to deal with minor formatting glitches in the exchange of ODF documents.

Even if the bar for alternative implementations is lowered to implementations that offer fair interoperability, there is still a significant advantage for Windows as compared to other operating systems. Windows has the choice of two plug-ins and Wordperfect, whereas the Linux platform is limited to Koffice. Users of the Mac OS, on the other hand, have no other useful alternatives for ODF. The number of independent, i.e., not related to OpenOffice.org, implementations by operating system with fair performance for ODF is shown in Figure 1.
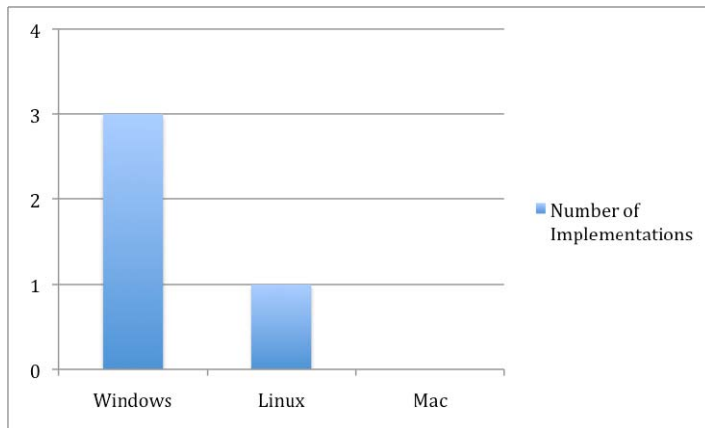
Figure 1. Alternative Implementations for ODF by Operating System

In the case of OOXML, the best results were for Microsoft Office and OpenOffice.org. The OpenOffice.org is a special version developed by Novell that only runs on Windows and Novell's version of Linux named SUSE Linux. At this point, for OOXML, Windows users have several implementations with good compatibility, while Mac users are limited because Pages cannot write OOXML. Linux users are also limited in that they must run SUSE Linux to use OOXML.

By lowering the bar to fair interoperability, users now have the choice of Wordperfect for Windows and the ThinkFree Office which runs on Windows, Macs, and Linux. However, just as with ODF, users not running Windows have limited choices for using OOXML.

The lack of good performance by open source implementations is significant. Many governments and organizations are considering or mandating the use of open source products. The results here indicate that if users want open source implementations, they need to provide more resources to these projects.

The final implication stems from the surprisingly good results for OOXML implementations. Critics of OOXML have argued that it was too complex and difficult to

implement. While OOXML is a long and complex standard, it is possible to offer good compatibility. In fact, our results suggest that implementations of OOXML work as well as implementations of ODF. 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 lead to more implementations, there appears no reason why OOXML cannot catch up. After all, several developers have provided independent implementations of OOXML.

6. 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, in the case of OOXML, Microsoft has readily admitted that they will not support the ISO/IEC version of OOXML until their next major revision of Microsoft Office. 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 together in scoring implementations. Specifically, 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 the result of document format compatibility, but 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 type such as spreadsheets and presentations. As a result,

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 testing was limited to correctly reading documents and not writing documents. In a real world situation, adopters need to be confident that their implementations both read and write in conformance with the standard. Our test has not yet examined the issues for writing, which are important for some features such as styles and tracking changes.

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

7. CONCLUSIONS

This study sought to investigate interoperability for various implementations of ODF and OOXML. After all, to receive the perceived economic and technological benefits, there is a need for multiple independent, interoperable implementations. The results clearly indicate that both ODF and OOXML implementations need to improve interoperability.

This study only tested a small subset, basic word processing features, of what is needed for multiple interoperable implementations. Additionally, this test did not consider the writing performance of implementations, only the read or import function was tested. Nevertheless, the only implementations of ODF that provided good compatibility with OpenOffice.org were the Microsoft Office plug-ins. Similarly, the only implementation of OOXML that can provide good compatibility with Microsoft Office 2007 was OpenOffice.org with the Novell plug-in. A

number of other implementations of ODF and OOXML such as Wordperfect, Google Docs, and KOffice lacked good compatibility.

It is surprising and ironic that the best implementations of ODF are when using Microsoft Office. Similarly, the best implementation of OOXML is OpenOffice.org. (Pages provided similar results but lacks the ability to write OOXML, a needed feature for an interoperable implementation.) The domination of Microsoft Office and OpenOffice.org is especially troubling for users of other operating systems, such as Mac OS and Linux. These users do not have a choice when using ODF or OOXML.

The results here show that developers need to work together to improve this situation. 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.

Supporters of both ODF and OOXML have suggested improved conformance and interoperability testing, there has been little progress on this front. 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.

There is still much research and testing to be done. Each of these implementations is continually being improved and needs to be continually reassessed. Future research needs to expand the tests to spreadsheets and presentations. Research also needs to test both reading and writing documents to determine if features such as styles and tracking comments are working properly. 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.

8. ACKNOWLEDGMENTS

9. REFERENCES

Andersen, Jens Jakob. 2008. ODF and OOXML in Denmark: National IT and Telecom Agency.

Berkman Center for Internet & Society at Harvard Law School. 2005. Roadmap for Open ICT Ecosystems.

Ditch, Walter. 2007. XML-based Office Document Standards. *JISC Technology & Standards Watch*.

Ecma International. 2007. *Office Open XML Overview* 2007 [cited November 1 2007]. Available from http://www.ecma-international.org/news/TC45_current_work/OpenXML%20White%20Paper.pdf.

Ghosh, Rishab A., and Philipp Schmidt. 2006. Open Source and Open Standards: A New Frontier for Economic Development. *United Nations University* 1.

Kindrick, James D., John A. Sauter, and Robert S. Matthews. 1996. Improving Conformance and Interoperability Testing. *StandardView* 4 (1):51-58.

Langer, Werner. 2008. Experiences in Format Conversions: German Ministry of Foreign Affairs.

Moseley, Scott, Steve Randall, and Anthony Wiles. 2003. Experience within ETSI of the combined roles of conformance testing and interoperability testing. Paper read at Standardization and Innovation in Information Technology.

National Institute of Standards and Technology. 2008. *XML Technologies Conformance Testing*, Feb. 19 2008 [cited May 22 2008]. Available from http://www.itl.nist.gov/div897/docs/xml_tech_conformance.html.

Organization for the Advancement of Structured Information Standards. 2007. *Open by Design: The Advantages of the OpenDocument Format (ODF)*. OASIS, December 10, 2006 2007

[cited September 12 2007]. Available from http://www.oasis-open.org/committees/download.php/21450/oasis_odf_advantages_10dec2006.pdf.

Vestin, Johanna. 2003. Interoperability Test and XML Evaluation of StarOffice Writer 6.0 and Office Word 2003 Beta 2 Swedish Agency for Public Management,.

Weir, Rob. 2008. *Anatomy of Interoperability* 2007 [cited May 22 2008]. Available from http://www.robweir.com/blog/2007/02/anatomy-of-interoperability.html.

Zehler, Peter. 1998. Interoperability Testing for Internet Printing Protocol. *StandardView* 6 (4):180-184.

[cited September 12 2007]. Available from http://www.oasis-open.org/committees/download.php/21450/oasis_odf_advantages_10dec2006.pdf.

Vestin, Johanna. 2003. Interoperability Test and XML Evaluation of StarOffice Writer 6.0 and Office Word 2003 Beta 2 Swedish Agency for Public Management,.

Weir, Rob. 2008. *Anatomy of Interoperability* 2007 [cited May 22 2008]. Available from http://www.robweir.com/blog/2007/02/anatomy-of-interoperability.html.

Zehler, Peter. 1998. Interoperability Testing for Internet Printing Protocol. *StandardView* 6 (4):180-184.

10. APPENDIX

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 1. Test document #1, Basic Formatting

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 2. Test document #2, Images

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 3. Test document #3, Tables

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 4. Test document #4, Headers and Footers

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

Table 5. Test document #5, Footnotes, Endnotes, Tracking Changes, Table of Contents