Format Wars

File formats: the past, the present and a possible future

Marco Fioretti

Real programmers love their applications' source code: the faster and more elegant it is, the better. Users are after very different things: they seem to want simplicity, flashy colors, nice icons and tons of options. In spite of these reasons, or perhaps because of them, programmers and users often forget what lies in the middle of it all: information.

Who owns the information?

Almost all software applications are used to manage *information* so these applications are worthless without information to process, store and display. For example, you could use a word processor to write letters or video editing suites to edit footage of your girlfriend at the beach.

Almost all software applications are used to manage *information* so these applications are worthless without information to process, store and display

If information exists before (and independent of) the applications, the file format used to store the information should be defined before hand. In this ideal situation, you could potentially write several programs (released under free or non-free licenses) to handle your information.

Please keep in mind that here "information" means *any* kind of creative work: blog entries, private movies, essays, government reports, court rulings, road projects… In an ideal world, the format used to store this information doesn't matter: it should simply belong only to its author, or whoever paid for its production.

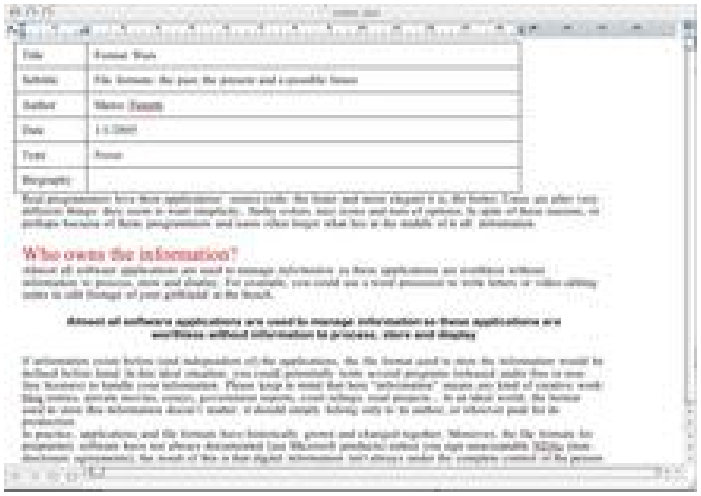
Fig. 1: An OpenOffice RTF file opened with Word X for Macintosh

In practice, applications and file formats have historically grown and changed together. Moreover, the file formats for proprietary software have not always been documented (see Microsoft products) unless you sign unacceptable NDAs (Non-Disclosure Agreements); the result of this is that digital information isn't always under the complete control of the person who created it.

In my opinion this problem has been underestimated for a long time, probably because in the beginning people didn't think it was such a big deal.

First of all, far fewer people had computers. When they did have them, they weren't often networked and were physically incompatible (think of Mac and PCs, which even had problems sharing a floppy disk!). Resources were very limited: monitors, processors and hard drives weren't even remotely comparable to what we have today, and therefore visually "fancy" information wasn't as important as it is today (think WYSIWYG). Even complex spreadsheets were

Fig. 2: The same OpenOffice RTF file opened with Word 2004 for Macintosh



stored as CSV format (plain text separated by commas) or as binary files. Back then, there was a situation similar to today's: if the information was stored as text files, you could use powerful text processing tools like sed, awk and then Perl. If it was stored in binary format, reverse engineering and black magic fixed most of the problems. Exchanging information at that point wasn't often a problem; even when binary-only format became more common thanks to WordStar and AutoCAD, the end product was nearly always a stack of paper that was to be shipped or archived somewhere.

This paper could then be read even centuries after it was written, without a concern for what "brand" of paper, or which printer or pen had been used to write on it.

In a way, paper was the *lingua franca*.

> In a way, paper was the *lingua franca*

Today, with the internet, CDs and search engines, any file can be used and distributed in several different ways without ever turning into durable, non proprietary (and non-searchable, I must add), printed paper. Talk about progress...

Today's scenario

Today's scenario is somehow very similar to what it was a few years ago - just a bit more complicated. Proprietary file formats are now more complex than before and therefore harder to reverse-engineer. Text-based file formats are still based on text (obviously!), but they have gained a level of complexity as well: rather than representing the information directly (like plain text documents or CSV spreadsheets do), they are usually based on XML.

For example, the content of a cell in OpenOffice.org could be represented with this:

```
<style:properties style:column-width="1.785cm"/>
\ldots{}
<table:table-cell><text:p>600000</text:p>
  </table:table-cell>
```

These two lines above simply state that the width of the column containing this cell must be 1.785 cm and that the cell stores the number 600000.

A paragraph in a letter could be:

This is the first paragraph

This is the second one

The advantages of XML files are clear: anybody can write an application which manipulates them, as long as they know what every XML tag means in that specific context.

A word on encoding

Even "plain text" can mean different things, depending on how it's *encoded*. The encoding defines which sequence of bits represents a particular character (such as a letter, a white space, symbols like "©" and "#", and so on) used in a written language.

In ASCII (*American Standard Code for Information Interchange*), for example, the sequence "01000001" corresponds with the capital letter "A".

> Even "plain text" can mean different things, depending on how it's *encoded*

The ASCII encoding (or format) is really ubiquitous these days, but has simply outlived its meaning in a wired world where most people *don't* speak English. Over the last few years many more types of encoding have been created in order to deal with almost any other language on the planet including non-alphabetic ones (Chinese, Hindu, Korean, Japanese...). The resulting confusion has been made worse by the fact that "plain text files" don't contain, by definition, any headers to declare their internal encoding. Consequently, the programs processing them have to guess, or be told, which encoding they should use to display them; otherwise, blank or strange characters are displayed instead of the correct ones.

Today, the Unicode family of standards provides a definitive solution; unfortunately, it will take a lot of effort and time to have it accepted - not to mention used - everywhere. When, in 2002, Red Hat Linux switched to Unicode, many people complained on mailing lists because “everything had become slower, just to make the French happy”.

The problem with closed file formats

Why should end users care at all about these issues? Because in the last two decades at least, file formats have been used to avoid free market competition, making it harder for customers to switch to newer and better products, or to place restrictions on how people use programs or the information produced with them.

In the last two decades at least, file formats have been used to avoid free market competition, making it harder for customers to switch to newer and better products

This is evident in fields as varied as office automation, industrial design and video streaming. In the first case, almost every user knows that the only guaranteed way to open “.doc” or “.xls” files *reliably* is by using the same version of Microsoft Word or Excel which created them in the first place. Remember that this applies to *all* of your files, starting from your personal diary...

When it comes to engineering, many projects for buildings, mechanical parts, furniture and bridges are stored in the DWG file format of AutoCAD, produced by AutoDesk. In 1998, competitors launched cheaper products based on an equivalent format. AutoDesk’s reaction was not limited to improving features, service and discounts. Their advertising campaign focused on reminding people that only AutoDesk’s products were 100% capable of keeping *existing* projects completely accessible. The full story can be read online in the FAQ and history pages on the website of the Open Design Alliance (`http://www.opendesign.com`) founded just to create an alternative file format.

What about multimedia? MPEG-4 is an advanced format for compressed video: DivX and many other decoders are based on it. Now, according to the MPEG-4 License page (`www.mpegla.com/m4v/m4v-faq.cfm`):

Fig. 3: The home page of the Open Design Alliance

> Video providers who receive remuneration for offering MPEG-4 video either directly (e.g. subscription or title-by-title fees) or indirectly (e.g. advertising or underwriting fees) pay a royalty for the right to use the decoders and encoders to receive and transmit the remunerated video.

The fear of having to pay MPEG-4 fees even when you place banners and video clips of your holidays on your home page has been enough to start projects like Theora (`http://www.theora.org`).

XML: the savior?

The cases above are just a few examples of how file formats have effectively been used to enforce a much greater control on end users than was possible before.

The family of technologies known as XML (*eXtensible Markup Language*) can play an essential role in solving these problems (at least in some areas).

XML was designed to make it easy to *exchange* information rather than locking it.

XML was designed to make it easy to *exchange* information not easy to lock information

XML files are in a plain (Unicode!) text format similar to HTML. This alone makes reverse-engineering of XML files much easier, compared with binary formats. Of course text can never be as compact and fast to parse as pure binary data, but it has a huge advantage: it can be processed with

Fig. 4: The OASIS consortium's home page

any of the existing text-processing tools, known to and improved on by Unix users since the '70s.

For example to extract the XML code shown above I only had to unzip the original spreadsheet and open the *content.xml* file with a text editor.

However, XML is no more or less proprietary or open than binary formats. Its full benefits are only available when it's completely and openly documented, guaranteed to stay that way and, above all, legally usable without asking permission or paying fees to anybody.

Format wars: the next episode

The OASIS consortium (http://www.oasis-open.org) produces open XML standards in all fields of business and computing activity. Perhaps its most important achievement is the OpenDocument format for word processing, spreadsheets and presentations, directly derived from the one used in OpenOffice.org and submitted to the International Standard Organization (ISO).

OpenDocument is more powerful than XHTML and, unlike other formats, there are already some cross platform applications which use it. OpenOffice.org 2.0, due for release around March 2005, will use it by default, and other products, from Koffice (http://koffice.kde.org/) to IBM's Workplace and servers like Plone (http://www.plone.org/), can already read and write files in this format. Just add a Firefox plug-in, and OpenDocument will be immediately accessible from your browser! For these reasons, some people think that it could eventually replace HTML as the default format for the internet.

However, there is no need to look that far. There is something much more important already happening today. Namely, the European Union (EU) wants to make it possible for all EU public administrations to (re)take ownership of the documents they manage on behalf of their citizens. In order for this to happen, these administrations, or anybody willing to do business with them, will eventually have to produce, exchange and store files in the right format.

In 2003 an EU study called the Valoris Report (http://europa.eu.int/ida/en/document/3439) concluded that an XML file format, highly portable and very open, is required to reach this goal. The report mentions the efforts in this field by Sun (OpenOffice.org/OASIS) and Microsoft (MSXML), pointing out several limitations of the latter. The main limit is the fact that Microsoft prefers not to completely separate the file format from the applications. They would much rather assist *selected* partners in enabling their applications to read and interoperate with MSXML. It doesn't sound like much of a concession, does it?

This episode of the Format Wars is still being quietly fought while we're writing (mid December 2004): stay tuned for further news. Hopefully, if it all ends as it should, the utopia described at the beginning of the article, can come into being: that file formats are defined *before and independently* of any implementation, in any field of computing.

Conclusions

In my opinion, one of the best signs that software is still in its infancy is the way this issue of formats has been ignored so far, by professionals and casual users alike. Luckily the tide has started to turn. Things like *OpenDocument* are certainly steps in the right direction. Nobody can predict what combinations of proprietary and free software will be used twenty years from now. The most probable guess is that there will be a lot of them, and each user will be free to choose the best combination for his or her real needs. In any case, I hope that in twenty years the era where information is locked up by proprietary and application specific formats will be just a laughable memory.

Copyright information

About the author

Marco Fioretti is a freelance writer based in Italy