

On Data Formats for the »epoline« System

PA Axel H. Horns (PAK, epi)

1. Introduction.

1.1 The Findings of the OCC.

Recently D. K. SPEISER in his capacity as Chair of the *epi Online Communications Committee* (»OCC«) has delivered a report¹ on certain considerations and conclusions concerning technical details of the on–line filing procedure (»OFP«) as implemented by the epoline system of the EPO. According to this report, the OCC conveyed the message to the EPO that representatives using the present on–line filing (»OLF«) system have a severe liability problem stemming from the danger of possible omissions caused by the software. The report further states that the problem of incomplete disclosure originates from the desire of the EPO to receive the texts in a particular common format called *Portable Document Format* (»PDF«). The arguing as put forward by the OCC starts with an assertion saying that the PDF format is not the native storage format for most word processing and drawing programs but requires a separate conversion step. To convert the output of those products into the needed PDF format a *PDF converter software* is required and up to now there is no PDF converter known that works error–free.

The analysis as given hereinafter aims to foster a thorough discussion of the related subject–matter of concern. The question posed by the OCC should be debated on the basis of a full understanding of the various technical concepts underlying the structure of PDF and word processor files. To this end, a brief introduction into this field of Information Technology (»IT«) is presented together with an assessment of certain proposed conclusions for the epoline practise.

1.2 Bits in a Digital File and the Role of the »Encoding Scheme«.

When drafting a patent application on a computer by utilising some kind of a word processing software and/or a suitable graphics program, the result usually manifests itself in the form of one or more *files*. A file is an ordered collection of bits on a storage medium e.g. on a hard disk drive. This collection must be interpreted in accordance with a proper *encoding scheme* as supported by the respective authoring software which is used on the applicant's side. Without having reached amongst all of the involved parties an agreement on the applicable encoding scheme, these ordered bits as such forming the file would be completely meaningless for anyone except the creator; without such agreement, the string of bits in a file can be interpreted arbitrarily. For text files, the encoding scheme effectively is a method of converting a sequence of ordered bits into a sequence of characters.

A very widespread rule for the interpretation of bits in a computer file for text representation purposes is the ASCII² code. The ASCII code requires that the string of bits forming a certain computer file is divided into groups of eight consecutive bits each, called *byte*. Then, the eight bits of every byte are seen as representation of a number identifying a printable or non–printable character as indicated in the ASCII table.³ However, for more complex applications, the ASCII code is not seen as being sufficiently rich because of the maximum of 256 available different ASCII characters is far too small e.g. when it comes to representing characters from different character sets like the Latin, Greek or Cyrillic alphabets. Also special mathematical operators like »√« or »ℵ« are not covered by the early ASCII code. In order to overcome this problem, the concept of

¹ D. K. Speiser: Report of the Online Communications Committee for the time period May–June 2002. epi Information 02/2002, pages 71–72.

² American Standard Code for Information Interchange.

³ See e.g. <http://www.asciitable.com/> [Visited on 2002–10–17].

codepages^{4,5} has been invented. A codepage is a language specific ordered set of characters in which a numeric index is associated with each character.⁶ Although each codepage is confined to a maximum of 256 characters, by switching between different codepages a computer program can distinguish more than 256 characters taken from more than a single alphabet. However, proper handling of different codepages is complicated and may cause that the structure of software becomes somewhat clumsy. A much better solution is to abandon the byte-oriented character encoding of all ASCII-style codetables and instead to introduce a different approach known as *Unicode*^{7,8} wherein each character is encoded in a sixteen bit quantity (two bytes) offering a wealth of 65,536 different character codes. Unicode encoding taken alone does not solve the problem of representing complex mathematical and/or chemical formulas in patent application documents. For such purposes, not only a large character set is needed; furthermore tools must be available to arrange characters in a specified pattern. To this end, a traditional word processing program is able not only to process sequences of characters taken from one or more alphabets; instead it is capable to encode text attributes like line alignment, font type, font size etc. etc. This makes the encoding scheme of a modern text processing program still more complex; the file created by such software usually comprises is a very detailed data structure representing the various aspects of document content as displayable on a screen or on sheets of paper.

When electronically filing a patent application, the applicant or a respective representative prepares a file which should be interpreted in the targeted Patent Office along exactly the same encoding scheme which has been applied during creation. If the Patent Office should ever apply any means embodying a different encoding scheme, the disclosure as perceived by the Patent Office might well significantly differ from that on the applicant's side. Before committing the electronic submission, usually the person acting on behalf of the applicant will use a *displaying component* which may be the word processing software or some other program to view the contents of the patent application in order to scrutinise the contents thereof, the displaying component necessarily being trusted to resemble exactly the encoding scheme as enforced by the Patent Office. But how is it possible to make sure that the encoding schemes applied by the applicant's displaying component are exactly the same as those later to be applied by the Patent Office?⁹

1.3 Problems of Overspecified Patent Application Documents.

When preparing and filing a patent application – regardless whether on paper or by digital electronic means – not all aspects of the document as filed are equally significant. Of course, the mere sequence of characters of the application text forms the basic substrate of the disclosure of the description and of the claims. In most cases, details of text formatting are irrelevant in view of the disclosure. For example, the aesthetic particulars of the font used for printing the application text in general do not affect the disclosure, e.g. the difference between a sans-serif font like »Arial« and a font with serifs like »Times« does not matter when typing a patent application.

When writing a patent application, a standard word processing software assigns a full set of style data to each portion of the edited document regardless of whether such information is meaningful in terms of disclosure or

4 With regard to Microsoft Windows Codepages, see on-line via the URL

<http://www.microsoft.com/globaldev/reference/WinCP.asp> [Visited on 2002-11-01].

5 There also related standards on codetables. ISO 8859 is a standardised series of 8bit character sets for writing in Western alphabetic languages. It was designed by the European Computer Manufacturer's Association (ECMA). For a rough list of the languages accommodated in the ISO 8859 series, see on-line under the URL

<http://www.terena.nl/library/multiling/ml-docs/iso-8859.html> [Visited on 2002-11-01].

6 See <http://www.microsoft.com/globaldev/reference/Glossary.asp> [Visited on 2002-11-01].

7 See <http://www.unicode.org/unicode/standard/WhatIsUnicode.html> [Visited 2002-10-18].

"[...] No single encoding could contain enough encoding systems for assigning these numbers. No single encoding could contain enough characters: for example, the European Union alone requires several different encodings to cover all its languages. Even for a single language like English no single encoding was adequate for all the letters, punctuation, and technical symbols in common use. These encoding systems also conflict with one another. That is, two encodings can use the same number for two different characters, or use different numbers for the same character. Any given computer (especially servers) needs to support many different encodings; yet whenever data is passed between different encodings or platforms, that data always runs the risk of corruption. Unicode is changing all that! Unicode provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language.[...]"

8 Unicode Standard Annex #28 Unicode 3.2: see on-line under the URL

<http://www.unicode.org/unicode/reports/tr28/> [Visited on 2002-10-18].

9 Evidently, no software ever written for practical purposes is 100% error-free. Up to now, there is even no utilisable theoretical model how to validate program code against a desired program behaviour. Hence, it would be a rather unsound assumption to start with some kind of axiom saying that any PDF generator program is considered to be completely error-free.

not. Therefore, authoring a patent application by utilising one of the usual off-the-shelf word processor software products means to inevitably clutter the document file with a lot of additional formatting information which do not contribute to the disclosure of the contents thereof. However, when it comes to the inclusion of mathematical or chemical formula expressions, also the fact e.g. that a certain character is set as a lower or upper index might well be relevant. Furthermore, in particular in fractions or matrices, the position of characters relative to each other becomes highly relevant. Not to forget the indication of the alphabet from which a certain character has been taken is relevant in a formula – usually there is a difference in the disclosure if a Latin »a« is replaced by a Greek »α«. In such cases, the provision of a detailed and feature-rich formatting information is mandatory. Therefore, the file of a patent application document created by a standard word processing software is not only cluttered with unnecessary formatting information in normal text paragraphs beyond the requirements of proper disclosure but there is also no meta-information available within the file utilisable on the EPO's side for some kind of automated discrimination of non-essential formatting data (which should be ignored) against vital formatting attributes e.g. in formula expressions (which is critical in terms of disclosure).

It looks as if it would be advantageous to restrict the information passed to the Patent Office to such data elements only which are relevant in view of the technical disclosure thereof. The more feature-rich an encoding scheme, the more complex the corresponding word processing and/or displaying software will be. And, in turn, the more complex such software is the more likelihood will be there for encoding conflicts, i.e. situations where the contents of a file is interpreted at the Patent Office in a different way than by the IT system on the applicant's side. Moreover, if the target system at the EPO is flooded with irrelevant formatting data, more errors might emerge out of some accidental mis-interpretation of such superfluous data.

Obviously there seems to exist an *optimum of richness* of the encoding scheme somewhere between an under-featured pure ASCII-style encoding scheme and an over-featured word processing format conveying a lot of text attributes which are irrelevant with regard to the disclosure. For the purposes of OLF patent business, preferably the chosen encoding scheme should resemble this optimum as closely as possible.

The other quite self-explanatory aspect should clearly be that the encoding schemes are to be published in form of an *open standard* governed by some independent trustworthy body. If the encoding schemes required by the OLF process would be kept secret by some vendor of a proprietary software, the only way to view the contents of the files submitted would be to buy and use the software as provided by the vendor. In addition, there would be a significant likelihood that the vendor changes the encoding scheme at any time from version to version or in response to the operating system platform on which the software is running because of some unilateral business-driven considerations without consultation or even advance notification of their various customers.

2. The »Portable Document Format«.

2.1 Overview of PDF.

PDF is a proprietary quasi-standard of ADOBE SYSTEMS INCORPORATED,¹⁰ SAN JOSE, California, USA. The technical details of the PDF are published by Adobe in the PDF Reference¹¹ manual. The origins of the Portable Document Format and the ADOBE ACROBAT product family date to early 1990. At that time, another quasi-standard, the POSTSCRIPT page description language,¹² was rapidly becoming the world-wide standard for the production of the printed page. PDF builds on the PostScript page description language by layering a document structure and interactive navigation features on PostScript's under-lying imaging model and aims to provide a file format for representing documents in a manner independent of the application software, hard-

¹⁰ See also on-line under the URL <http://www.adobe.com/aboutadobe/contact.html> [Visited on 2002-10-16].

¹¹ ADOBE SYSTEMS INCORPORATED: PDF reference: Adobe portable document format version 1.4. Boston: Addison-Wesley, Third Edition, 2001. Also available for free on-line under the URL <http://partners.adobe.com/asn/developer/acrosdk/docs/filefmtspecs/PDFReference.pdf> [Visited on 2002-10-14].

¹² ADOBE SYSTEMS INCORPORATED: PostScript Language Reference. Boston: Addison-Wesley, Third Edition, 1999. Also available for free on-line under the URL <http://partners.adobe.com/asn/developer/pdfs/tn/PLRM.pdf> [Visited 2002-10-18].

ware, and operating system used to create them and of the output device on which they are to be displayed or printed.

A *PDF document* consists of a collection of *objects* that together describe the appearance of one or more pages, possibly accompanied by additional interactive elements and higher-level application data. Correspondingly, a PDF file contains a number of such objects making up a PDF document along with associated structural information, all represented as a single self-contained sequence of bytes. The pages of a PDF document may contain any combination of text, graphics, and images. A page's appearance is described by a *PDF content stream*, which contains a sequence of graphics objects to be painted on the page. This appearance is fully specified; all layout and formatting decisions have already been made by the application that generated the content stream.

2.2 Details of the Text Representation in the PDF.

The text of a PDF encoded patent application is stored in one or more *PDF text objects* located within a *content stream* of the body. A PDF text object consists of operators that can show text strings, move the text position, and set text state and certain other parameters. The text operators specify the *glyphs* to be painted, the glyphs being represented by *string objects*, whose values are interpreted as sequences of character codes. A character is an abstract symbol, whereas a glyph is a specific graphical rendering of a character. For example, the glyphs A, **A**, and *A* are renderings of the abstract »A« character. A glyph is a graphical shape and is, within PDF, subject to all graphical manipulations, such as co-ordinate transformation. Because of the importance of text in most page descriptions, PDF provides higher-level facilities that permit an application to describe, select, and render glyphs conveniently and efficiently.

Glyphs are organised into *fonts*. A font defines glyphs for a particular character set; for example, the »Helvetica« and »Times Roman« fonts define glyphs for a set of standard Latin characters. A font for use with a PDF viewer application is prepared in the form of a program. Such a font program contains glyph descriptions that generate glyphs. It is written in a special-purpose language which is understood by a specialised *font interpreter* software modules included in every PDF viewer, respectively.¹³ A list of relevant font types available in PDF documents is given below:

- The *TrueType font* format¹⁴ was developed by Apple Computer and has been adopted as a standard font format for MICROSOFT WINDOWS.¹⁵
- A *Type 1 font* program is a stylised PostScript program that describes glyph shapes.¹⁶ It is the font format for single-byte Roman fonts for use with Adobe Type Manager software and with PostScript printers. In particular, Type 1 fonts use a specialised subset of the PostScript language which is optimised for better performance and a more compact representation.
- *Type 3 fonts* differ from the other fonts supported by PDF. Type 3 fonts can use the full PostScript computer language to express a font. Because Type 3 fonts can use the full PostScript language, they can do some things that Type 1 fonts cannot do, such as specify shading, colour, and fill patterns. Type 3 fonts are mainly useful only for special purpose or very complex fonts such as complex logos or converted *Encapsulated PostScript* (»EPS«) art files. The format also provides a way to represent bitmap characters. In Type 3 fonts, glyphs are defined by streams of PDF graphics operators. These streams are associated with character names.

¹³ See also <http://partners.adobe.com/asn/developer/type/ftypes.html> [Visited on 2002-10-31].

¹⁴ For specification documents, see on-line under the URL <http://www.microsoft.com/typography/specs/default.htm> [Visited 2002-10-31].

¹⁵ Meanwhile, *TrueType* is undergoing a process to be replaced by *OpenType*. OpenType is a new standard for digital type fonts, developed jointly by Adobe and Microsoft. OpenType supersedes Microsoft's TrueType Open extensions to the TrueType format. OpenType fonts can contain either PostScript or TrueType outlines in a common wrapper. An OpenType font is a single file, which can be used on both Macintosh and Windows platforms without conversion. For details, see on-line via the URL <http://www.adobe.com/type/topics/info9.html#opentype> [Visited on 2002-10-31].

¹⁶ Adobe Type 1 Font Format, see on-line via the URL http://partners.adobe.com/asn/developer/pdfs/tn/T1_SPEC.PDF [Visited 2002-11-01].

In PDF, the term »font« refers in particular to a *font dictionary*, a PDF object that identifies the font program and contains additional information about it. The several different types of fonts are identified by the subtype entry of the font dictionary.

The content streams paint glyphs on the page by specifying a font dictionary and a string object that is interpreted as a sequence of one or more character codes identifying glyphs in the font. This operation is called *showing* the text string. To paint glyphs, a content stream must first identify the font to be used. A so-called *Tf operator* embedded in the content stream specifies the name of a font resource, i.e. an entry in a font sub-dictionary of another entity called *resource dictionary*. The value of that entry is a font dictionary. The font dictionary in turn specifies the type of font, its PostScript name, its encoding, and information that can be used to provide a substitute when the font program is not available.

Glyphs in the font are selected by single-byte character codes obtained from a string that is shown by the text-showing operators. Logically, these codes index into a table of 256 glyphs; the mapping from codes to glyphs is given by the font's encoding. Each font program has a built-in encoding. Under some circumstances, the encoding can be altered by certain means. A font's encoding is the association between character codes (obtained from text strings that are shown) and glyph descriptions. Except for Type 3 fonts, every font program has a built-in encoding. Under certain circumstances, a PDF font dictionary can change a font's built-in encoding to match the requirements of the application generating the text being shown. This flexibility in character encoding is deemed to be valuable for the general PDF user but it clearly introduces further complexity potentially deteriorating the robustness of the corresponding software. In particular, two important features are provided:

- It permits showing text that is encoded according to any of the various existing conventions. For example, the MICROSOFT WINDOWS and APPLE MAC OS operating systems use different standard encodings for Latin text, and many applications use their own special-purpose encodings.
- It allows applications to specify how characters selected from a large character set are to be encoded. Some character sets consist of more than 256 characters, including ligatures, accented characters, and other symbols required for high-quality typography or non-Latin writing systems. Different encodings can select different subsets of the same character set.

The possible encoding modifications depend on the font type, as discussed below:

- The *TrueType* font format provides within the *TrueType Font File* an entity called *Character To Glyph Index Mapping Table* (»cmap«).¹⁷ This table defines the mapping of character codes to the glyph index values used in the font. It may contain more than one subtable in order to support more than one character encoding scheme. Character codes that do not correspond to any glyph in the font are generally mapped to glyph index 0. The glyph at this location must be a special glyph representing a missing character.
- A *Type 1* font program uses a compact encoding for the glyph descriptions. A specification of the font's character encoding may be given if different from its built-in encoding. The value of *Encoding* may be either the name of a predefined encoding (»MacRomanEncoding«, »MacExpertEncoding«, or »WinAnsiEncoding«) or an encoding dictionary that specifies differences from the font's built-in encoding or from a specified predefined encoding.
- A *Type 3* font dictionary defines the font itself, while the other font dictionaries simply contain information about the font and refer to a separate font program for the actual glyph descriptions. A separate encoding entry maps character codes to the appropriate character names for the glyphs. An encoding dictionary whose *Differences* array specifies the complete character encoding for this font.

Optionally, the font program itself can be embedded as a stream object in the PDF file. In fact, the most predictable and dependable results are produced when all font programs used to show text are embedded in the PDF file. On the other hand, if a PDF file refers to font programs that are not embedded, the results depend

¹⁷ See *The TrueType Font File*, on-line available via the URL http://www.microsoft.com/typography/tt/ttf_spec/ttch02.doc [Visited 2002-10-31].

on the availability of fonts in the viewer application's environment. If the specified font is not available in the environment of the viewer, a different font will be chosen as a substitute without any assertion that the disclosure of the content is preserved properly. Embedding a font into a PDF file is, however, significant in terms of copyright law; i.e., a copyright-protected font program would be copied and distributed when being embedded into a PDF document authored and distributed by a third party. This is in general not allowable without the consent of the right holder. In particular, the TrueType font format includes a dedicated data element set by the creator of a font to specify the copyright rules governing potential embedding of the font into other documents.¹⁸ The TrueType format allows the creator of a font to specify one of four different levels of embedding for the font:

- no embedding at all,
- embedding for viewing and printing, but not editing,
- embedding for viewing, printing and editing, or
- fully installable embedding.

Some PDF generator programs as for example newer versions of the ACROBAT DISTILLER software will check this condition before including any font file data into a PDF document. Sadly in this particular case there is even no error message if the requested embedding step was skipped.¹⁹ Therefore, a user preparing an electronic patent application on a MICROSOFT WINDOWS platform utilising TrueType fonts who is unaware that the particular selected font is restricted by the terms of the copyright license might later find that despite explicit program settings to this effect actually no font data were embedded and passed to the EPO.

The brief overview of the PDF file format as presented hereinbefore shows that the primary design goal was to provide a tool for describing a page layout completely and with the highest degree of flexibility that was possible. It was clearly not the goal to set up a simple encoding scheme designed for the unambiguous representation of a textual content.

2.3 The Practise of the Creation of PDF Files for »epoline« Submission.

Usually, documents containing text and/or graphics are created by certain pieces of software like word processor software (e.g. MICROSOFT WORD, SUN STAR OFFICE, or COREL WORDPERFECT, etc.), desktop publishing software (e.g. QUARKXPRESS of QUARK, INC., etc.), or by pure graphics software (e.g. Gimp etc.). ADOBE's PDF business concept is not necessarily to attempt to replace all of these software products by certain ADOBE counterparts but to offer a widespread and (quasi-)standardised software interface which users of other third-party software may utilise in order to transform the result of their work into PDF files.

The benefit promised to the general PDF user in return for his additional efforts to perform a conversion step to PDF is the assertion that by utilising PDF the *visual appearance* of every kind of document can be *completely* defined independent on the particular IT environment on which the produced PDF files are further processed.²⁰ On a PC running MICROSOFT WINDOWS, the exact layout of a printed page is best known to the printing subsystem of the operating system. This is the very reason which the creation of a PDF file in a Window environment usually starts with a printing step but not by invoking an export filter from the *File Menu* (»Save as ...«) or the like. A MICROSOFT WINDOWS printer driver is an executable file that, among other tasks, converts calls from the *graphics device interface* (»GDI«) into device commands which generate output on a page of printer paper. In general, the printer device driver is proprietary for each and every printer device model. However, a certain class of printers capable of directly processing PostScript print jobs can be addressed by a common software interface. In order to establish this general interface, and in view of the

18 See the report "Monotype: (Font) Embedding and the Digital Millennium Copyright Act Does statement on installable embedding of fonts infer Adobe Acrobat violates the DMCA? – 5 September 2002" on-line via the URL <http://www.planetpdf.com/mainpage.asp?webpageid=2245> [Visited 2002-10-19].

19 Reported on 2002-03-21 in a contribution to the efs@patents.com mailing list by JOHN BAMBRIDGE, Director epoline Project of the EPO.

20 In practise, PDF software in general is well known for its robustness in this respect; for example, a layout designer creating a brochure with a dedicated layout of text plus graphics and delivering the result of his efforts to a printing company in form of a PDF file can trust on that the large-scale printing systems installed there will exactly reproduce the said layout in the same way as it was seen on the proof sheets created from the same PDF file by means of a colour laser printer.

similarities between PostScript and PDF, Adobe created a piece of proprietary software called »Acrobat Distiller« to transform a page layout description given by a PostScript file into a PDF file. Other vendors meanwhile have offered similar competing products, one of them is the »Amyuni« software featured by the EPO for use in the context of the »epoline« OLF system.

In general, this goal is accomplished only indirectly as follows:²¹

- In a first step, the user creates a patent application document e.g. by utilising MICROSOFT WORD or the like. The user types in certain characters by hitting the respective keys on the computer keyboard. A complex layout information is automatically assigned to the components of the document (e.g. font attributes on the character level, adjusting information on the paragraph level, or header plus footer information on the page level) during the authoring process. If necessary, by certain mouse and/or function key actions the user may confer special attributes to a character(s) or other components, e.g. by assigning a certain font to a character in order to let it appear as a Greek symbol in the context of a formula.
- In a second step, a printout is initiated to a PostScript compatible printing device. However, this printout must not be directed to a real printer; instead it has to be redirected by selecting an option offered by the printer menu to redirect the output stream into a file collecting the PostScript output. In a typical MICROSOFT WINDOWS environment, the word processing software does not know anything about how to drive a specific PostScript printer. This work is done instead by a *printer device driver* installed within the Windows operating system. Usually, the printer device driver is provided by the manufacturer of the respective printer device. However, most patent professionals do not own or use a PostScript compatible printing device,²² i.e. a printer device which can directly be fed with a PostScript data stream directly interpreted and rendered by a powerful embedded processor installed therein. But the user will in general see that he can obtain a printer driver software module for some PostScript printer device for free from the Internet or from the distribution kit CDs of the MICROSOFT WINDOWS operating system without having to buy any related hardware.
- Then, in a third and final step, the PostScript output file so created is presented as input data to the PDF converter program which creates a PDF file for electronic submission with the EPO. Usually this step is implemented in a parameterised way. For example, the Acrobat Distiller software allows a considerable number of options to be set by the user. One of the more important parameters specifies which of the font files present on the user's computer system are to be embedded into the resulting PDF file and which are not; cf. the discussion above in section 2.2.

Although the creation of PDF files is generally possible on any operating system platform, various data structures in the resulting output files will retain a specific legacy of their platform of origin. For example, if the source files have been authored and processed on a MICROSOFT WINDOWS platform, it is most likely that TrueType fonts will be used by the majority of users because of such computers are per default equipped with this sort of font files. To the contrary, if a LINUX or APPLE MAC OS platform has been utilised on the applicant's side, there is a much higher probability that Type 1 fonts are available and also used. Moreover, the character encodings may be different when processing text files under APPLE MAC OS compared to MICROSOFT WINDOWS or LINUX; cf. the discussion of the PDF font dictionary data structure as given above. Hence, assuming that PDF software never is 100% error free if the IT system on the EPO's side were e.g. be entirely based on a MICROSOFT WINDOWS platform, it would not be very surprising if the probability of encoding mismatches would turn out to be significantly higher for PDF files created on LINUX or APPLE MAC OS platforms compared to those PDF files created under MICROSOFT WINDOWS, and vice versa.

²¹ Because of a certain word processing software, namely Microsoft Word, is extremely widespread and creates an attractive market for add-in products, Adobe offers a plug-in software module which enables the creation of a PDF file out of a Microsoft Word document on the click of a button. However, this is only a measure to enhance usability and the user's comfort; the underlying basic technical principles are essentially the same.

²² Most popular printer models use different schemes; e.g. the Printer Control Language (PCL) utilised by HP's LaserJet printer models.

2.4 Extraction of Text from PDF Files in the »epoline« Facilities.

The ultimate goal of the »epoline« OLF system of the EPO is to obtain the patent applications submitted electronically in a character-encoded format for various internal purposes, in particular for preparation of the publication of patent documents, without having to perform any expensive *Optical Character Recognition* (»OCR«) process. Therefore, a simple conversion of the submitted PDF files into a bitmap image representing the paper layout would be completely insufficient for gathering the data needed for the various EPO business processes. Hence, at the Patent Office, the sequences of characters encoded within the submitted PDF files have to be extracted. To this end, the software installed on the IT system of the EPO has to parse the incoming PDF files in order to get access to the content streams embedded therein. The starting point for the extraction process are the string objects providing a chain of character codes. However, such retrieved character codes taken alone are meaningless because of their encoding must be determined in order to learn the true designation of the respective character for re-encoding the same in accordance with the conventions of the EPO IT system. The necessary data for this step resides in the font dictionary which must be accessed in accordance with the applicable *Tf* operator.

The extraction process can be very difficult if the disclosure of the content expressed in the submitted PDF file is based on font variations. For example, if in the context of a formula the Latin letter »a« is used in italics as well as in regular shape with different meanings within a formula, or, in another case, the letter »R« is used in a modern-style font like Arial with one meaning and in an old-fashioned German type font like »℞« in another meaning, a very scrutinising examination of the font dictionary data structure has to be performed at the EPO's end in order to sense the difference. A complicated condition might perhaps occur if someone would have the idea to use some more advanced desktop publishing software for the creation of formulas in a PDF file where a variable is designated by a letter, say, of the Greek alphabet rotated by 180° utilising the glyph co-ordinate transformation capabilities of the PDF. One might argue that such undertaking on the applicant's side is not very wisely considered; however, unless there are clear and unambiguous regulations prohibiting such constellations the EPO would have also to cope with extreme situations.

The situation might become still more complex if for some reason mentioned above not all of the non-standard fonts referenced to by the content streams are actually embedded within the PDF file. In such cases, when at the EPO there is no such font available, the font substitution mechanisms of PDF might cause that on the EPO's side a different substitution font is used for content determination in a manner which is completely intransparent to the applicant. Furthermore, there might exist a number of fonts of the same name which are, however, technically different. So, if on the applicant's side a certain font originating from a first vendor is used whereas the EPO has installed a similar font with the same name but with minor differences, more problems might show up.

2.5 Conclusion.

The technical discussion as presented above shows that the representation of the content and, hence, the disclosure, of a document in PDF files is strongly glyph-oriented. That what was intended by its creators when specifying PDF is to have a most versatile tool for defining the shape and position of glyphs together with other objects on a sheet of paper as precisely as possible. PDF is clearly a document interchange tool across various platform boundaries if the term »document« is understood in the sense of a visual layout appearance of a two-dimensional object. To this end, PDF shows a fair level of robustness²³ over platform boundaries; nevertheless, many Internet savvy PDF users have already suffered problems when attempting to display or print PDF documents authored by various institutions round the world, e.g. when trying to read a downloaded PDF file originated by, say, Japanese institutions, on a European-style PC equipped with Latin TrueType fonts. Despite the fact that such document e.g. merely contains an English text made up purely of Latin characters, cases are known where the European version of a PDF viewer was unable to display the contents.

²³ The concept of robustness should to be seen in the context of the intended target industry, i.e. in the context of the publishing process where omissions of other faults can more easily be rectified than in the patent business.

PDF never was specified to serve as a document content or document structure exchange tool. It is true that Adobe as well as other vendors offer various extraction tools for extracting text and/or graphics objects from a PDF file. However, these tools are designed merely for occasional object extraction procedures e.g. if a PDF user should wish to transfer a portion of text or an image from a PDF document via the clipboard into another document. In particular the text extraction strongly depends on the font-related data structures which in turn are typically different in terms of font format as well as of font encoding across different operating systems. Proper adoption of such data access to cross-platform boundary scenarios is always fragile. Although text and graphics extraction from PDF documents seems to work for small quantities of simple text, any attempt to set up an electronic filing system in a highly sensitive legal environment appears to be something like a creative abuse of a tool designed for completely different purposes.

On a more abstract level, the transfer of disclosure-relevant contents by means of a PDF file submitted on behalf of the applicant involves at least three separate steps:

- Creation of a source file on the applicant's side by means of an authoring tool and conversion of the same into a PostScript program file and later on into a PDF file,
- On the EPO's end, extraction of text as a chain of characters including determination of character font, character encoding and various layout style parameters, and
- Also on the EPO's end, discrimination of a first set of text attributes retrieved during the first step which are irrelevant in terms of disclosure against a second set of text attributes which are relevant in terms of disclosure.

Already the first step opens a Pandora's box of potential side effects and/or faults which can hardly be effectively mastered by the average user. The properties of the intermediate PostScript program file are set by some third party's printer driver software in a way which is entirely intransparent to the user. Evidently the particulars of the PostScript intermediate file strongly influence the data structures of the resulting PDF file. Important parameters like font management data cannot be controlled by the user at this stage. The second one of both steps is somewhat error-prone due to the enormous complexity of the data structures involved and because of this process is very sensitive against flaws in font data e.g. when font embedding problems occur. The third step cannot be performed on an automated basis because of PDF lacks metadata elements indicating the *structure* of a patent application document e.g. for distinguishing normal text from a formula etc.

Assume that there is an applicant drafting a patent application using a MICROSOFT WINDOWS platform with TrueType fonts installed thereon. After having scrutinised the resulting PDF file by inspecting it with the Acrobat Reader software on the same computer, the file is submitted to the EPO via the »epoline« OLF facility. Let us further assume that due to a font embedding problem the particular TrueType font used by the applicant in fact is due to copyright restrictions neither embedded in the PDF file nor otherwise available on the IT system of the EPO. The PDF font substitution mechanism replaces this non-available font chosen by the applicant by another similar font available on the IT system of the EPO. However, assuming it turns out that an important formula in the description looks different when viewed on the applicant's system compared to the appearance when displayed in the EPO using the substitute font: What about the *original disclosure*? What would be made available to the general public 18 months after the filing date?

- If the EPO would allow access to the original PDF file as submitted by the applicant, different groups of the public would perceive a different original disclosure, depending on the particular configuration of their respective IT systems. Some members of the public having a system with the original font installed would view the formula as intended by the applicant, others would obtain a different disclosure. Such ambiguity of the original disclosure seems to be not acceptable.
- If the EPO would, in an initial processing step, convert the PDF file as submitted on behalf of the applicant into a bitmap graphics file as single basis for public file inspection in order to freeze the layout as perceived by the Office, the question of the legal basis for such restricted interpretation of the PDF semantics would arise.

- A similar problem would arise if the result of the text extraction process done by the EPO would be deemed to be the original disclosure solely made accessible for public file inspection.

More problems show up when discussing the fact that PDF is a proprietary quasi-standard being the property of a certain company. Albeit Adobe laudably has chosen to publicly disclose the PDF specification in full detail, it should not be forgotten that it is at Adobe's discretion to alter the specification at any time without prior notice. Then, if an applicant submits a PDF file of a higher version level than anticipated by the EPO in good faith, some kind of encoding mismatch might happen if the facilities on the EPO's side are not yet prepared to process PDF files under this new version.

Hence, it seems to be clear that the current technical concept of the »epoline« document exchange is much too complex in its very technical details in order to be mastered by the applicants or their representatives. Moreover, it might well be that the risks of a technical fault will have to be borne in an asymmetric fashion by the applicant or the representative; what can be said is that at least no effective legal remedies are visible which might shift some of this burden away from the applicant's side. As a consequence, a truly workable solution should at least be much simpler in terms of the semantics of the data structures involved.

3. The Microsoft ».DOC« Format.

In order to escape from the malaise caused by the PDF-based epoline OLF facilities, the OCC has put forward the proposal to open the epoline system also for co-submission of genuine file formats of various word processing programs. Although the appeal issued by the OCC literally seems to cover all kinds of word processing software, the main focus certainly was centred on DOC-files created by the MICROSOFT WORD program.^{24,25}

- The specification of the syntax and the semantics of the binary format of DOC files never was openly published by MICROSOFT. Although every Internet user being familiar with modern powerful search engines no doubt will be successful in retrieving from somewhere something like a DOC-file marked with »Microsoft Confidential« and titled e.g. »Microsoft Word 6.0 Binary File Format«, it seems hardly imaginable that a public authority like a Patent Office should be in a position to interpret an electronic submission on the basis of the *trade secrets* of any third party.
- Moreover, any attempt to do so would be simply unfeasible in practical terms. In the past, MICROSOFT constantly has changed the DOC binary file format specification from time to time when releasing new versions of the MICROSOFT WORD software. Does that mean that the EPO should create and maintain over decades a pool of PC computer systems with the various versions of this software running thereon in order to be able to obtain the correct visual appearance of a word processor file co-submitted with the PDF file? Of course, it is to be expected that newer versions of MICROSOFT WORD will be able to import files of the generic format of elder versions by certain import filters. But who assures that these input filters are always working correctly?
- The MICROSOFT WORD software stores characters as integer numbers attributed by some information defining the font and its encoding in a way being different in nearly every detail but based on similar principles compared to that described above with regard to PDF files. If the software would have been well designed it should be expected that the character typed in will appear exactly in the same way when retrieved the generic file format used by this particular software. However, it has been demonstrated that even a transfer of a DOC file created using MICROSOFT WORD for Apple running on an Apple Macintosh computer to a MICROSOFT WINDOWS environment fails in certain cases.²⁶

Hence, any co-submission of original word processor files together with a PDF file would only serve to increase the confusion in case of a legal debate on the question of what in fact should be treated as original

24 See "Decision of the President of the European Patent Office dated 29 October 2002 on the electronic filing of patent applications and other documents" via http://www.epo.co.at/epo/president/e/2002_10_31_e.htm [Visited on 2002-11-10].

25 See also "Notice dated 29 October 2002 concerning the electronic filing of patent applications and other documents" via http://www.epo.co.at/news/info/2002_10_31_e.htm [Visited on 2002-11-10].

26 Dirk Fox: "Zu einem prinzipiellen Problem digitaler Signaturen", DuD 22 (1998), Nr. 7, S. 386 bis 388 [In German].

disclosure. The word processor file as co-submitted by the applicant would only be suitable as a means for proving the original disclosure if rendered on exactly the same computer in exactly the same state it occupied at the time of creation of the word processor file. Months or years after the filing date such attempt would be futile. Moreover, allowing such retrospective rendering processes would open the door for any kind of fraudulent arguing before the EPO.

4. The »Extensible Mark-up Language« (»XML«).

4.1 XML Overview.

The *Extensible Mark-up Language*²⁷ (»XML«) is a simple, very flexible text format derived from the earlier *Structured Generalised Mark-up Language*²⁸ (»SGML«) which has been standardised as ISO 8879:1986. Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Internet and elsewhere. XML describes a class of data objects called XML documents and also partially describes the behaviour of computer programs which process them. XML is an application profile or restricted form of SGML. By construction, XML documents are conforming SGML documents. XML documents are made up of storage units called *entities*, which contain either *parsed* or *unparsed* data. Parsed data is made up of characters, some of which form *character data* and some of which form *mark-up*. A parsed entity contains text, a sequence of characters, which may represent mark-up or character data. A character is an atomic unit of text as specified by ISO/IEC 10646. Legal characters are tab, carriage return, line feed, and the legal characters of Unicode and ISO/IEC 10646. *Mark-up* encodes a description of the document's storage layout and logical structure.

Contrary to PostScript and PDF, in XML there is no direct linking provided between characters on one hand and a particular visual presentation by certain chains of glyphs on the other hand. No layout is associated with an XML document as such, and the characters are not mapped to glyphs from a particular font. This overall structure of XML means that the digital representation of content with relevance for the disclosure can be separated from any digital representation of layout. A visual presentation of a given XML document can be defined by assigning a *stylesheet document* to it. A stylesheet document can also be expressed by means of a special flavour of XML called *Extensible Stylesheet Language* (»XSL«).²⁹ XSL is a language for expressing stylesheets. It consists of three parts:

- *XSL Transformations*³⁰ (»XSLT«) which is a language for transforming XML documents,
- *XML Path Language*³¹ (»XPath«), an expression language used by XSLT to access or refer to parts of an XML document, and
- *XSL Formatting Objects* (»XSL-FO«),³² an XML vocabulary for specifying formatting semantics. An XSL stylesheet specifies the presentation of a class of XML documents by describing how an instance of the class is transformed into an XML document that uses the formatting vocabulary.

There is another important feature of XML-based solutions for OLF systems: Mathematical expressions can be expressed utilising dedicated mark-up languages. In particular, MathML 2.0³³ was released in 2001 as a W3C Recommendation. MathML is a low-level specification for describing mathematics as a basis for machine to machine communication. It provides a much needed foundation for the inclusion of mathematical expressions in Web pages. Even if MathML is found not to be the optimum solution for representing formula

27 Extensible Mark-up Language (XML) 1.0 (Second Edition) W3C Recommendation. 6 October 2000. See on-line under the URL <http://www.w3.org/TR/2000/REC-xml-20001006.pdf> [Visited on 2002-10-16].

28 See e.g. MARTIN BRYAN: An Introduction to the Standard Generalised Mark-up Language (SGML) <http://www.sgml.u-net.com/sgml.htm> [Visited on 2002-10-18].

29 See on-line under the URL <http://www.w3.org/Style/XSL/> [Visited 2002-10-18].

30 See on-line under the URL <http://www.w3.org/TR/xslt> [Visited 2002-10-18].

31 See on-line under the URL <http://www.w3.org/TR/xpath> [Visited on 2002-10-18].

32 See on-line under the URL <http://www.renderx.com/Tests/doc/fo/tutorial.pdf> [Visited on 2002-10-18].

33 See on-line under the URL <http://www.w3.org/Math/> [Visited on 2002-10-31].

expressions within patent application documents, similar concepts tailored to be more problem specific could be created.

One of the most advantageous features of XML is the option to validate the parsed data of a document against a formal grammar. The *XML document type declaration* (»DTD«) contains or points to markup declarations that provide a grammar for a class of documents. XML processors fall into two classes: validating and non-validating. Validating and non-validating processors alike must report violations of this specification's well-formedness constraints in the content of the document entity and any other parsed entities that they read. Validating processors must, at user option, report violations of the constraints expressed by the declarations in the DTD, and failures to fulfil the validity constraints given in this specification. To accomplish this, validating XML processors must read and process the entire DTD and all external parsed entities referenced in the document.³⁴ By utilising this validating technology it becomes e.g. possible to verify that essential data elements are included in the file representing an electronic patent application.

4.2 XML OLF scenarios for patent professionals.

XML technology is already in use in various places throughout the IP business. For example, the USPTO operates an *Electronic Filing System* (»EFS«) which is an electronic system for submitting patent applications, computer readable format (»CRF«) biosequence listings, and pre-grant publication submissions to the USPTO via the Internet.^{35,36} It includes, inter alia, authoring tools to help the applicant prepare a patent specification in XML format.

Also the OHIM is currently working on the fundamental structure for XML use, namely core business data for Community Trade Marks and Designs.^{37,38} This XML standard will be defined in iterative phases, which correspond to the Office's needs. Initially, the first phase will only include related Community Trade Mark data – which in turn will only include a number of data sets for the definition of persons, including agents, employees and owners, for definition of publication information, for the definition of mark information, for the definition of priority information, and for the definition of seniority information.

The utilisation of XML-based file formats in the context of electronic filing solutions for the Intellectual Property business have widely been investigated recently by the MIPEX Project.^{39,40} The project has delivered, inter alia, a generic software called »PaTrAS« to enable electronic filing of patent and trade mark applications. The software is now in use at the UK Patent Office for electronic filing of trade mark applications. Plans are in hand to introduce the software at the Danish Patent and Trademark Office for electronic filing of both patent and trade mark applications, at the German Patent and Trademark Office for patent applications⁴¹ and at the Swedish Patent and Registration Office for trade mark electronic filing. A change management board has been set up to oversee future development of the PaTrAS software, supported by a technical board to manage future software changes.

5. epoline – quo vadis?

In former times, patent applications have been typed on sheets of paper by means of mechanical typewriters. Later on, the mechanical typewriters were replaced by PCs with word processing software. Writing patent applications by utilising word processing software is clearly an acceptable procedure as long as the final result comes as a pile of printed paper sheets – in this case the final quality check is made as done since early

34 See on-line via the URLs <http://www.w3.org/TR/2000/REC-xml-20001006#dt-doctype> and <http://www.w3.org/TR/2000/REC-xml-20001006#proc-types> [Visited on 2002-11-01].

35 See on-line via the URL <http://www.uspto.gov/efc/efs/index.html> [Visited on 2002-10-31].

36 Further materials are available via the URL <http://www.patents.com/efs/> [Visited on 2002-10-31].

37 See on-line under the URL <http://oami.eu.int/EN/examplesCTM/default.cfm> [Visited on 2002-10-31].

38 See also on-line under <http://oami.eu.int/EN/examplesCTM/XML/Overview.ppt> [Visited on 2002-10-31].

39 See on-line under the URL <http://www.patent.gov.uk/about/projects/mipex/> [Visited on 2002-10-31].

40 The MIPEX II project completed its work in September 2001 and the final report of the project can be viewed under the URL <http://www.patent.gov.uk/about/projects/mipex/final.pdf> [Visited on 2002-10-31].

41 See a statement issued by the German Patent and Trade Mark Office [In German]: http://www.dpma.de/infos/messe/cebit_d.html [Visited on 2002-10-31].

times by carefully proof-reading the printed output. But as soon as the final work result is no longer printed paper but a digital computer file which cannot be proof-read without employing further technical means, different considerations seem to be necessary.

The present PDF based »epoline« OLF solution might be seen as an attempt to preserve the concepts of the old world of paper-based file wrappers in the offices of the patent professionals even in the age of the electronic Information Society of global computer networks. The drawbacks of such undertaking – lack of structural economic benefits, increased legal risks – are substantially caused by the enormous *complexity* of this approach. *Complexity* is meant here in the sense of *semantic complexity* of the underlying data structures, not of the workflow in the offices on the applicant's side. On a management level, this approach might be clearly seen as a less complex solution because of no change of business procedures and no further training of office staff seems to be required. But it looks as if this way would lead to a deadlock situation because of it does not promise economic benefits as well as acceptable liability risks for the patent professionals.

In view of the technical discussion as set out hereinbefore in greater detail, the question to be discussed here seems not to be whether or not XML would be suitable for the »epoline« OLF procedure. The answer can hardly be other than affirmative. There is, however, no reason to assume that any XML based OLF procedure would be expected to be 100% error-free. What can be said is that in terms of semantical complexity a suitable tailored XML solution can provide a feature-rich data structure which is very detailed in critical parts (formulas etc.) and more shallow e.g. for running text portions of the description. In this sense, XML can serve to approach a solution resembling the *optimum of richness* of the encoding scheme as postulated in section 1.3. It is to be expected that *displaying components* can be implemented which are sufficiently reliable for practical purposes. At the EPO's end, no cumbersome and error-prone text extraction step from a complex glyph-oriented data structure is necessary. Furthermore, the XML validation options can be used to provide excellent tools to eliminate certain more formal faults like omission of essential structure elements of a patent application document. Hence, switching to a XML based solution would significantly reduce the complexity of the relevant information to be managed because of the structure of the patent application document is encoded in the file to be submitted with the Patent Office.

Evidently, there would be no gain for the patent professionals if the potentially error-prone conversion step from a word processor file format to PDF would be replaced by another error-prone conversion step from the word processor file format to XML. The simple truth might well be that on the long term patent applications should no longer be authored using word processing software like Microsoft Word, SUN StarOffice, Corel WordPerfect or the like. All these tools are designed to create files representing a layout on a sheet of paper. *However, a patent application well-suited for electronic submission is not a layout but a data structure.* And, no word processor software does know anything significant about the logical structure of the content behind a layout. A transition from a word processing software towards genuine XML editor software would be advantageous although such step surely will cause some headaches with regard to questions like the qualification of secretary staff usually doing the typing work in conjunction with the preparation of patent applications.

Although there are technical XML based solutions devisable where the XML files are nicely printed for proof-reading purposes on sheets of paper utilising XSL transformations, the risks of software faults in the course of this data transformation could be further reduced by considering that perhaps the patent professional in future should accept to do proof-reading directly on paper printouts showing the full XML mark-up structure.

* * * * *