kpt. mgr inż. Karol KREŃSKI
Szkoła Główna Służby Pożarniczej
prof. dr hab. inż. Tadeusz MACIAK
Politechnika Białostocka, Szkoła Główna Służby Pożarniczej
kpt. mgr inż. Adam KRASUSKI, Szkoła Główna Służby Pożarniczej

AN OVERVIEW OF MARKUP LANGUAGES AND APPROPRIATENESS OF XML FOR DESCRIPTION OF FIRE AND RESCUE ANALYSES

This article reviews markup languages focusing primarily on XML language. This paper concludes that a new language could be created, that is based on XML, which could be suitable for describing fire & rescue analyses in the State Fire Service.

W artykule przedstawiono przegląd języków znakowania z położeniem szczególnego nacisku na charakterystykę języka XML. Na podstawie przeglądu wysunięto tezę, że bazując na XML można opracować nowy język nadający się do opisywania akcji ratowniczo-gaśniczych w Państwowej Straży Pożarnej.

1. Introduction

There is various documentation created for the needs of the *National Fire Service of Poland (PSP)*. Part of this documentation, particularly that related to operational matters could be collected and would form Knowledge Banks that could be utilised during fire and rescue actions, trainings, analyses, etc.

Unfortunately, there is a problem resulting from the way this data is stored − the collected documentation is stored in PSP in a way that cannot be processed by computer systems. This article outlines disadvantages in current documentation storage policy as well as proposals for their improvement possibly triggered by switching to *XML* language.

The first part of this paper outlines the genesis and development of markup languages. The currently most popular and developed markup language XML is then described along with two examples of its use: *SVG* and *MathML*. Lastly, weaknesses in the current way to document fire and rescue operations analyses are pointed out. The paper concludes that a special language to document the analyses, designed preferably as XML language could resolve the issues outlined in this paper.

2. Genesis and development of markup languages

The term *computer language* is often misidentified as equal to programming language [1]. However, it must be noted that computer language refers to a group of languages among which there can be distinguished:

- programming languages, e.g. C;
- markup languages, e.g. HTML (Hypertext Markup Language);
- scripting languages, e.g. Perl;
- query languages, e.g. SQL (Structured Query Language);
- transformation languages, e.g. XSLT (Extensible Stylesheet Language Transformations).

Markup languages are a combination of text and some additional information about text itself. The additional information is used to organise text and usually forms a hierarchical structure that encodes the information. Historically, the term *markup* refers to the process of making notes on manuscripts margins [2]. The notes contained typesetting guidelines specifying font type and size, typeface, indentation, spacing and relevant information [3]. In 1967 the idea of a computer markup language was introduced and soon an early implementation named *GenCode* was developed [4].

One of the well recognised markup languages is TEX, created in the seventies by Knuth [5]. TEX is a professional computer typesetting system including a dedicated language and its compiler responsible for transformation of the data into formats suitable for graphical equipment such as printers. There is also a popular TEX macros set named LATEX which is considerably easier to use than TEX [6].

A significant milestone in markup languages history was *Scribe* introduced by Reid in 1980, since it was the first language to explicitly separate a document's structure from its formatting [7]. The innovation was the use of styles and their relocation to another document.

Scribe became an inspiration to create one more language − *SGML (Standard Generalized Markup Language)* [8]. SGML defines the document's syntax and specifies where and which tags are allowed within the document. The tags are special objects appearing in the text, which shape its structure; this term will be described in more detail in a further section of this paper. SGML is in fact a *metalanguage* [9], i.e. it is a collection of rules which allow for defining actual markup languages; HTML is an example of a markup language conformed to SGML specification. Unfortunately, SGML includes an awkward problem. The authors of the SGML specification created the tool which is too universal and too complex and in effect it restricted the practical applications of the language [10].

In the eighties, in *CERN (Conseil Européen pour la Recherche Nucléaire)* research institute the *WWW (World Wide Web)* project was started. Berners-Lee defined

a small group of hypertext tags and called the new language *HTML* [11, 12]. Despite the fact that HTML is the most common way to present Web pages today, it focuses exclusively on the presentation which restricts the language and makes it subject to frequent critique. The Semantic Web [13], the future Web built upon computer friendly content must be developed on top of an appropriate computer language. Web studies proved that *XML* meets the criteria for intelligent Semantic Web.

3. XML Metalanguage

Metalanguage (language for describing other languages) *XML* was developed by *W3C (World Wide Web Consortium)* [14]. The complete XML specification is available from the W3C Web site [15]. XML originates from SGML and represents a major simplification of the new language as compared to its ancestor. The explicitly visible difference seems to be the introduction of strict syntax rules that simplify processing of the files. The SGML designers assumed that the files would be generated manually by humans so as to ease their laborious work they allowed for relatively free syntax. For example, certain tags could be omitted in certain constructions. This resulted in more complex software needed to process SGML files as well as a requirement for knowledgeable programmers able to handle a more complex language. XML designers addressed the issue and the resulting language is more friendly for both humans and machines. XML, as a metalanguage is just a set of rules which other languages for specific applications can be built upon.

There are many popular XML-based languages, for example:

- SVG (Scalable Vector Graphics) − language for describing vector graphics [16, 17];
- MathML (Mathematical Markup Language) − language for describing mathematical notation [18];
- MusicXML - language for describing music notation [19];
- ODF (Open Document Format) − language for describing office documents [20];
- XHTML (ang. Extensible HyperText Markup Language) − language for describing Web pages [21].

Each XML document must be *well-formed* and in the event its document type is defined the document must also be *Valid* against its document type.

3.1. Well-formed document

XML syntax is considerably strict and intuitive. The structure of the document is defined by special *tags* which form the definition of the objects being described. The tags are arbitrary words enclosed by angle brackets:

- opening tag, e.g. <age>

- closing tag, e.g. </age> - the closing tag uses the same name but is preceded by a slash (/).

There are usually values, e.g. <age>64</age>, or other tags, e.g. <person><name>...</name></person> placed between tags. Nesting tags inside other tags is what shapes the object's structure. To describe the syntax better, an example document chess.xml is presented and described below:

```
1.<?xml version="1.0" encoding="UTF-8"?>
2.<Chess>
3.  <heavy>
4.     <piece>Queen</piece>
5.     <piece>Rook</piece>
6.  </heavy>
7.  <light>
8.     <piece>Bishop</piece>
9.     <piece>Knight</piece>
10. </light>
11.    <piece>King</piece>
12.    <piece>Peon</piece>
13.</Chess>
```

The numbering was introduced only for the purpose of simplification of comments and is not part of the XML file. The first line declares that the file is in XML format and that it uses *UTF-8 encoding* [22]. The <Chess> element is the so-called *root element*. The XML specification requires that a document must contain one and only one root element - all the remaining elements must be descendant *(child)* relatively to the root element. The data is structured by nesting elements inside their ascendant elements *(parents)*. The structure contains the information that Bishop (8) is a piece belonging to the group light (7). All the tags are aligned in pairs: each opening tag <some_tag> requires its relevant closing tag </some_tag>. XML also settles a strict closing rule: the next tag to be closed is the last opened one, i.e. <a> is the valid code while <a> is not. The objects version and encoding appearing in the first line are *attributes*. Each attribute has a quoted or double quoted value assigned that additionally describes the given element. This means that element Bishop could be fully described using attributes exclusively:

```
<Bishop group="light"></Bishop>
```

Presenting Bishop in this way results in information loss with regards to its place in Chess structure. The attributes make more sense when they are used to describe the features which are irrelevant from the point of view of a document's structure,

e.g.:

```
<piece colour="white" value="3">Bishop</piece>
```

In practice, it is often not easy to decide whether to describe the object via attributes or by nesting the consecutive elements - the judgement depends on the designer's preferences.

XML documents are processed by parsers - programs that analyse input data in order to obtain their grammatical structure. The XML standard requires that any syntax error encountered must be fatal and cause parser exit. Other languages, e.g. HTML are designed to ignore many syntax errors. On one hand, it is comfortable for the programmer since the parser "forgives" minor mistakes. On the other hand, the description of a more flexible syntax is always internally more complex and can't be easily reconciled with documents which don't depend on a DTD/schema.

By default, XML doesn't specify which elements are allowed in the document or what structure the document must have. There is only a requirement that each XML document must be well-formed, so that any parser could at least read it. However, it is possible to control the structure and properties of appearing elements, e.g. the following rules may be defined:

- the document can only contain the elements: <Chess>, <heavy>, <light>, <piece>;
- element <Chess> can only contain the elements: <heavy>, <light>, <piece>;
- element <heavy> must appear before the element <light>;
- element <piece> can appear multiple times;
- element <piece> must contain a value;
- element <piece> must be of the type text;
- … .

All the rules must be then gathered together in the so called document type. Over the years, there were a few document types created [23]. The most important ones are listed below:

- DTD;
- XML Schema;
- RELAX NG.

If the document has the correct syntax and is compliant to its document type it is considered Valid. There are validators available, i.e. special programs that allow for checking for their syntax and structure conformance.

4. Examples of XML-based languages

As mentioned above there are many XML-based languages destined for actual use. This chapter presents characteristics of two such languages, SVG and MathML for describing vector graphics and mathematical expressions respectively.

4.1. SVG language

There are multiple two-dimensional vector graphics file formats in use. The conversion among those formats is usually cumbersome and it often happens that the converted image isn't perfectly accurate. In W3C there emerged an initiative to create an open vector Web standard that could possibly have a chance to become a universal vector format resolving compatibility issues. Major graphics and computer industry vendors that would guarantee the broad acceptance for new standard were successfully involved in the project. The usage of XML language which was fast gaining popularity in late nineties was an important factor leading to increased significance of the new standard. The new format was given the name *SVG* and it was a compromise between two propositions [24]:

- VML (Vector Markup Language), proposed by Microsoft and Macromedia;
- PGML (Precision Graphics Markup Language), proposed by Adobe Systems and Sun Microsystems.

The support for SVG on the Web is getting improved in subsequent releases of the major Web browsers, however it is not yet fully supported. There is an example of a graphic image along with it's SVG code presented below:

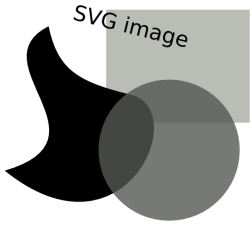

Fig. 1. *An example SVG image. Source: Author's work*

```
1.<? xml version="1.0" encoding="UTF-8" ?>
2.<svg height="113.23981" %width="127.15089">
3.  <g id="layer1">
4.     <rect
5.        style="opacity:1;fill:#ef2929"
6.        id="rect2161"
7.        width="67.14286"
8.        height="52.857143"
9.        x="55"
10.        y="6.8112359">
11.    </rect>
```

```
12.    <text
13.       style="font-size:
13.       37824154px;font-style:normal;font-variant:normal
14.       id="text2168"
15.       x="19.971537"
16.       y="14.66838">
17.       SVG image
18.    </text>
19. </g>
20. <g id="layer2">
21.    <path
22.       style="opacity:1;fill:#ad7fa8"
23.       id="path2183"
24.       d="M 71.76,238.87
25.       C 76.02,259.12   85.20,264.83   15.18,87.11
26.       C 139.04,277.92  107.42,351.67  51.24,306.47
27.       C 102.71,291.39  38.12,255.98   71.76,238.87 z">
28.    </path>
29.    <path
30.       style="opacity:0.8;fill:#729fcf"
31.       id="path2165"
32.       type="arc">
33.       ry="36.42857"
34.       rx="36.42857"
35.       cy="55.382664"
36.       cx="49.285713"
37.    </path>
38. </g>
39.</svg>
```

The above code was intentionally simplified (it is not a valid SVG file) and it is limited to contain only significant information. The code describes the graphic presented on the Fig. 1. The image was subdivided into two layers: layer1 (3) and layer2 (20). The first layer contains objects <rect> (4) (the rectangle) and <text> (12) (inscription “SVG image”). The style attribute controls the appearance of the objects. Line (5) defines the colour of the rectangle in RGB model [25]: fill: #ef2929 and the opacity of the object: opacity: 1 (the object is fully opaque, opposite to the semitransparent disk: opacity = 0.8 (30)). The width, height, x, y (lines 7−10) attributes define object geometry. In case of the objects with more complex geometry, such as the irregular object on the left of the image the *Bezier Curves* can be used [26]. The lines (24−27) describe the coordinates of the

consecutive curve control points *P*. This allows for defining the curvature of the consecutive curve segments (the segments are separated in an SVG file by the letter *C*). The equation 1 determines the coordinates of the points *B* which form the curve:

$$B(t) = P_0(1-t)^3 + 3P_1t(1-t)^2 + 3P_2t^2(1-t) + P_3t^3 \quad (1)$$

4.2. MathML language

Just like SVG, *MathML* standard was developed by W3C. The main goal for the project was to solve the issues in regards to putting mathematical formulas on the Web sites. Since HTML language doesn't provide appropriate mechanisms, mathematical expressions were usually converted to graphic files and presented on the Web.

There are a few Mathematics' description standards in existence, particularly TEX. However, TEX is somewhat limited due to being designed as a presentation language only. In MathML, on the other hand, the mathematical expression's elements are exposed as separate objects rather, which allows for revealing the sense behind the expression. Thanks to this approach, the presentation, as well as the transfer of mathematical expressions to mathematical programs for the purpose of making calculations is becoming simpler.

In the most popular mathematical programs, such as Mathematica, Scientific Workplace, Maple and Mathcad, at least a partial usage of MathML has been implemented [27]. MathML specification has been designed with additional aspects related to mathematical descriptions on internet Web sites in mind [28, 29]:

- the possibility of conducting word or phrase searches - MathML will enable search engines (such as *Google*) to search out mathematical expressions;
- the availability for those with vision impairments - the appropriate software will enable Web browsers to read aloud given mathematical expressions;
- in the area of e-learning systems, there exists a great need for the generation and interactive work with mathematical expressions.

A division was introduced in version 2.0 of MathML between:

- Presentation MathML, which serves to project expressions;
- Content MathML, which serves to describe the meaning of expressions and allow for carrying out of calculations

Both of the above forms can be applied simultaneously in one document. The semantic description (Content MathML) only allows to delineate expressions from basic mathematical domains, in contrast to the somewhat complete description allowed by presentation tags. This disparity is caused by the expressions' nature - it is much easier to describe the expression's appearance than the underlying logic within it, more so in further advanced mathematical

domains. Below are both forms of MathML presented describing the same expression $(a + b)^2$:

a) Presentation MathML

```
1.<msup>
2.  <mfenced>
3.     <mi>a</mi>
4.     <mo>+</mo>
5.     <mi>b</mi>
6.  </mfenced>
7.  <mn>2</mn>
8.</msup>
```

b) Content MathML

```
1.<apply>
2.  <power/>
3.  <apply>
4.     <plus/>
5.        <ci>a</ci>
6.        <ci>b</ci>
7.  </apply>
8.  <cn>2</cn>
9.</apply>
```

In the presentation part a) msup (1) defines an expression with a superscript, which expects two operands - the base and the exponent. The element mfenced (2) stands for a fragment captured in brackets. Elements mi, mo, mn in lines (3-5,4,7) signify consecutively: variables, operators and numbers.

In the content part b) element apply (1) initiates description of an exponential operation power (2). This operation requires two operands, from which the first one is apply (3) initiating an add operation plus (4) on two variables *a* and *b*; The second operand of the exponential operation is the number 2.

The expression considered could be described more easily in other languages, for example as (a+b)^2. However, this notation could be inappropriate for further computer processing, such as making calculations or reading the expressions aloud by the computer (the method must be universal and apply to arbitrary complex expressions).

The majority of mathematical languages are designed to allow for comfortable, manual entering of expressions by the user. However, MathML, because of a significant redundancy in the description code is rather meant for use with special MathML editors.

5. The problem of documenting actions analyses

A regulation exists in PSP, which orders that analyses be conducted from incidents as defined in the regulation [30]. All significant or otherwise interesting incidents should undergo analysis. These analyses are conducted in conformity with the template found in the attachment to the regulation and concerns the following issues [30]:

- Elementary data.
- Description of engaged fire and rescue operations.
- Operational protection of the factory, plant, incident area, etc.
- Preventional protection of the factory, plant, incident area, etc.
- General information.
- Conclusions.
- Directory of manpower and resources.
- An outline of the situation.

Analyses are created in a text editor, such as MSWord and, in this form, are sent and registered in the system. Registration is focused on the data input process rather, (data entry, registrar, the object of the analysis, the actual location of the file/documentation) than on the analysis content. This results in a limited ability to search out analyses, even on the computer on which it is installed. The effect is that potentially interested PSP units can't access the analyses.

The problem of analyses accessibility concerns the organization of databases and is the subject of independent research undertakings [31]. The research mentioned above deals with access to data and does not cover the method of organizing data in the file itself. The scope of that research is limited to working with metadata on the content of documents stored in the system (ie. keywords) and has limited abilities to process binary office documents (MSWord). Data can be organized in the database in the following way:

Keywords: ..... fire, warehouse, paints, varnish, warsaw

The date of the action: 15.07.2005

Analysis commencement: 21.03.2006

File location: ..... /2005/fire/warehouse_warsaw_2005.doc

In the file pointed in the field "File location" MSWord binary data is stored:

```
73 32 2f 61 63 63 65 6c 65 72 61 74 6f 72 2f        s2/accelerator/c
75 72 72 65 6e 74 2e 78 6d 6c 03 00 50 4b 07        urrent......PK..
00 00 00 00 02 00 00 00 00 00 00 00 50 4b 03        ............PK..
14 00 00 00 00 00 03 52 51 37 00 00 00 00 00        .......RQ7......
```

```
00 00 00 00 00 00 18 00 00 00 43 6f 6e 66 69        ..........Config
75 72 61 74 69 6f 6e 73 32 2f 66 6c 6f 61 74        urations2/floate
```

From the point of view of the computer, binary data does not contain any concrete structure. Even though all the information about the incident is available inside the file, one cannot ask the question: "What equipment was used in the action?" Defining a data structure acceptable and apparent for the computer would introduce new possibilities for the processing of analyses:

- the possibility of extracting more information about a given action by asking a question;
- the possibility of carrying out analyses on the given set of analyses;
- the possibility of action visualisation;
- other.

As was shown above, XML has the potential to be used as description language of an arbitrary domain. The development of a suitable language, based on XML, for the description of actions analyses, would solve the problems mentioned above.

6. Conclusions

As outlined in the article, the way that fire and rescue analyses are documented, does not allow for computer systems to query the analyses content. The introduction of a special language for the description of analyses, would give an array of new processing possibilities. According to the examples given, XML-based language gives the possibility to create, on its basis, a new language which would describe the analyses of fire and rescue operations. The authors plan to continue this research and to propose specification for such a language.

S T R E S Z C Z E N I E

Karol KREŃSKI,
Tadeusz MACIAK,
Adam KRASUSKI

PRZEGLĄD JĘZYKÓW ZNAKOWANIA I MOŻLIWOŚCI ZASTOSOWANIA XML DO OPISU AKCJI RATOWNICZO-GAŚNICZYCH

Przedstawiono podział języków komputerowych oraz genezę i rozwój języków znakowania, wspominając o najważniejszych przedstawicielach tej grupy: TEX/LATEX, SGML, HTML. Opisano metajęzyk XML oraz wymieniono języki tworzone na jego bazie: SVG, MathML, ODF, XHTML. Podano także ogólną, techniczną specyfikację języka XML, w tym informacje o typach dokumentów/

schematach. Pokazano, jak w praktyce modelowane są wybrane dziedziny za pomocą XML. Przedstawiono sposoby opisu grafiki za pomocą SVG oraz opisu wyrażeń matematycznych za pomocą MathML. Przedstawiono słabość w sposobie opisu analiz akcji ratowniczo-gaśniczych w PSP oraz zaproponowano język XML, który nadaje się do modelowania dowolnej dziedziny, w tym analiz akcji.

BIBLIOGRAPHY

1. Naomi S. Baron: The future of computer languages: implications for education. In *SIGCSE '86*: Proceedings of the seventeenth SIGCSE technical symposium on Computer science education. ACM Press USA , New York 1986, p. 44−49.
2. R. Toshniwal, Dharma P. Agrawal: Tracing the roots of markup languages. *Communications of the ACM* 2004, vol. 47, № 5, p. 95−98.
3. Dustin Swallows, David C. Yen, J. Michael Tarn: XML and WML integration: An analysis and strategies for implementation to meet mobile commerce challenges. *Comput. Stand. Interfaces* 2007, vol. 29, № 1, p. 97–108.
4. Patti Anklam: Technical communications as knowledge management: evolution of a profession. In *SIGDOC '99: Proceedings of the 17th annual international conference on Computer documentation*, ACM Press, New York, USA 1999, p. 36–44.
5. D. E. Knuth: TEX and METAFONT: New directions in typesetting. American Mathematical Society, Boston, MA, USA 1979.
6. L. Lamport: LATEX: a document preparation system: user's guide and reference manual. Addison-Wesley Longman Publishing Co., Boston, MA, USA 1994.
7. B.K. Reid: Scribe: A Document Specification Language and its Compiler. PhD thesis. Department of Computer Science, Carnegie-Mellon University, 1980.
8. C.F. Goldfarb: The SGML Handbook. Oxford University Press, 1990.
9. E. D. Pendergraft, N. Dale: Automatic linguistic classification. In Proceedings of the 1965 conference on Computational linguistics. Association for Computational Linguistics, Morristown, New York, USA 1965, p. 1–37.
10. R. Price: Beyond SGML. Proceedings of the third ACM conference on Digital libraries, 1998, s. 172–181.
11. T. Berners-Lee: WWW: Past, present and future. *IEEE Computer* 1996, vol. 29, № 10, p. 69–77.
12. R. Cailliau, H. Ashman: Hypertext in the Web − a History. *ACM Computing Surveys* 1999, vol. 31, nr 4es.
13. A. Ankolekar, M. Krotzsch, T. Tran, D. Vrandecic: The two cultures: mashing up web 2.0 and the semantic web. Proceedings of the 16th international conference on World Wide Web 2007, p. 825–834.
14. T. Berners-Lee: About the World Wide Web Consortium. Online in Internet: http://www.w3.org/Consortium, 9, 1997.

15. T. Bray, J. Paoli, C. M. Sperberg-McQuee: Extensible Markup Language (XML) 1.0. Technical report, W3C, 1998. 10.10.2007.
16. S.V. Graphics. XML Graphics for the Web: World Wide Web Consortium (W3C), http://www.w3.org/Graphics/SVG, 2003.
17. A. Quint, F. Design: Scalable vector graphics. *IEEE Multimedia* 2003, vol. 10, № 3, p. 99–102.
18. D. Carlisle, P. Ion, R. Miner, N. Poppelier: Mathematical Markup Language (MathML) version 2.0. W3C recommendation. World Wide Web Consortium, 2001.
19. M. Good: MusicXML: An Internet-Friendly Format for Sheet Music. *XML Conference and Expo* 2001, p. 3–4.
20. J. Tramullas, P. Garrido: OpenDocument standard for digital documents. *UPGRADE, The European Journal for the Informatics Professional* 2006, vol. 7, nr 6, p. 3–4.
21. C. Musciano, B. Kennedy: HTML & XHTML: The Definitive Guide. O'Reilly, 2006.
22. K. Whistler, M. Davis: Character Encoding Model. *UNICODE Consortium*, 2003.
23. J. Kosek, P. Nálevka: Relaxed: on the way towards true validation of compound documents. Proceedings of the 15th International Conference on World Wide Web 2006, p. 427–436.
24. S. Probets, J. Mong, D. Evans, D. Brailsford: Vector graphics: from PostScript and Flash to SVG. Proceedings of the 2001 ACM Symposium on Document engineering 2001, p. 135–143.
25. J.D. Foley: Computer Graphics: Principles and Practice. Addison-Wesley Professional, 1995.
26. G. Farin: Curves and Surfaces for Computer Aided Geometric Design. New York, USA 1988.
27. M. Froumentin: Mathematics on the Web with MathML. Proceedings of IAMC 2002 Workshop, 2002.
28. C. Bernareggi and D. Archambault: Mathematics on the web: emerging opportunities for visually impaired people. Proceedings of the 2007 international cross-disciplinary workshop on Web accessibility (W4A) 2007, p. 108–111.
29. R. Ross. Mathematics on the Web: *ACM SIGACT News* 1998, vol. 29, № 2, p. 33–41.
30. Rozporządzenie ministra spraw wewnętrznych i administracji z dnia 29 grudnia 1999 r. w sprawie szczegółowych zasad organizacji krajowego systemu ratowniczo-gaśniczego. Dz. U. 1999, nr 111.
31. A. Krasuski, T. Maciak: Rozproszone bazy danych, możliwości ich wykorzystania w Państwowej Straży Pożarnej. *Zeszyty Naukowe SGSP* 2006, nr 34.