SGML: About and My Approach *

Boris Tabotras
boris@xtalk.msk.su

Produced by TEItools on Dec 19, 2001

1 Introduction

Today you can encounter SGML abbreviation almost everywhere. There are various documentation sets distributed in SGML format. Ubiquitous web markup language HTML is an SGML application. SGML applications are emerging: editors, formatters, document management systems ... Adobe, Corel, SoftQuad, Microsoft sells SGML-related products. It is SGML what IBM, Sun, O'Reilly and many others rely for their documentation needs. IT infrastructure of heavy industry leaders heavily relies on SGML.

Borned in early 1980s SGML technology is now in its best form [1]. What is it for? What it gives to its users? How to work with it? These and some other questions I'm trying to address in this short article.

It was first written in 1997, and industry have faced many changes since then. I'm trying to note where it has been fallen behind status quo. Please also remember this is my own history and my own experience and my own opinions. It works beautifully for me, it might work less beautifully for you.

2 How Did I First Find SGML Technology

All of this begun from Intranet. This fashionable word meant for me that documents I've routinely written had to be available online. Before that I just happily typed them in TEX and printed on paper.

I could just make two separate versions of each document: one for paper and one for Web. But this will immediately rise a problem of synchronization.

Soon I've faced another trouble. Some papers had to be passed outside my company "in electronic form", meaning "in Microsoft Word format". I had my reasons not to work with Word myself, and even if I had not, this only added third independent version of the same paper, making maintenance a real nightmare.

Hence I needed some tool to automatically convert my documents to several formats. At least, I needed: TEX — for printing on paper, HTML — to place at Intranet server and MS-Word — for Windows interchange. Moreover, I wanted this mean to meet several other requirements:

- be modular, to add new output format or to change existing converters;
- be open, free, not overly tied to particular tools;
- be portable to at least different UNIX platforms;
- be easily localizable[2].

*SGML is one of the ISO standards, but ISO standards are proprietary. XML, an uglier implementation of S-Expressions, is a subset of SGML for the web, it is a set of open stardards from W3C *http://www.w3c.org*. We at FSM suggest our readers avoid to use SGML for the future free software projects. —**FSM**

[1] At least, if you count XML universe to belong to SGML.

[2] Please note that I'm Russian and vast majority of my pa-

I then went to Altavista, and the first reference I found was the one to the linuxdoc-sgml package. It did exactly what I wanted to: generate HTML, TEX and RTF from single input file. It was widely used within Linux Documentation Project. Its input format was something new for me. It was SGML. However, it wasn't anything complex. It resembled familiar HTML and looked understandable. For example, fragment of SGML document can look this way:

```
<DIV1>

<HEAD>The Structure of a TEI Text</HEAD>

<P>All TEI-conformant texts contain
<LIST TYPE="bullets">

<ITEM>a <TERM>TEI header</TERM> (marked
up as a <GI>teiHeader</GI> element) and
</ITEM>

<ITEM>the transcription of the text proper
(marked up as a <GI>text</GI> element).
</ITEM>

</LIST></P>

</DIV1>
```

In order to write using it I only had to remember several basic markup elements, for example that `<P>` denotes paragraph, `<LIST>` denotes list etc.

Text marked up in SGML is further passed to one of several filters. One filter converts it to HTML, another one to TEX, yet another one to RTF and so on.

pers are in Russian or bilingual, Russian/English.

3 What is SGML and what good is in it

3.1 History and Terminology

GML (Generalized Markup Language) was developed within ubiquitous IBM corp. His successor SGML (Standard Generalized Markup Language) was standardized in 1986 as an International Standard[3]. His intention were methods for electronic text preparation independent of input/output devices or operating environments. More precisely, SGML is metalanguage, that is a mean to formally describe a languages, in our case a markup languages.

Historically markup denoted annotations or any other marks within text, made for human compiler or typesetter to note how exactly typeset specific place in document. Examples include wave underlining, denoting italic, special marks to omit parts of text or specific font typeset, etc. When formatting and typesetting became automated, this word got used to encompass all markup codes, inserting into electronic texts to control formatting, printing or any other processing.

To generalize, markup or encoding is any mean to explicify text interpretation. On trivial level, any text is encoded in this sense: punctuation marks, spaces, capital letters, page layout and even inter-word spaces are markup. They help reading human to distinguish where one word ends and another starts, or how to identify structure of text, or where are headlines and simple syntactic elements like subordinate clauses. So text encoding for computer processing is like deciphering manuscript — a process of making explicit how exactly text content should be interpreted.

Markup language is a set of agreements about text encoding. Markup language should specify which markup is allowed, which is mandatory, how to distinguish text from markup and what markup means.

[3] ISO 8879:1986

SGML provides answers to first three questions, to answer the last separate documentation is usually required.

3.2 SGML Features

Three characteristics of SGML distinguish it from other markup languages.

3.2.1 Descriptive Markup

Descriptive markup uses markup codes that simply provide names to categorize parts of document. Codes like `<para>` or `\end{list}` simply identify portion of document and claim that "it is paragraph" or "it is end of recently opened list". To the contrary, procedural markup defines what processing should take place in this specific point of document: "call PARA procedure with arguments 1, b and x" or "move left margin 2mm to the left, right margin 2mm to the right, skip one line and go to new left margin", etc. In SGML, instructions required to process document (for example, to typeset it) are separated from descriptive markup encountered within document. They are usually reside outside of document in separate procedures or programs.

With descriptive, rather then procedural, markup the very same document can be processed with various programs for various purposes. Each program can apply different processing to those parts of document that are important to it. For example, content analysis program can ignore footnotes, but typesetting program can collect them and print at the end of each part. Different processing can apply to the same part of the text. For example, one program can extract people names and geographic names to create index or database, while another one processing same text, can just print names in differentiating font.

3.2.2 Document Types

SGML introduces document type and, correspondingly, document type definition (DTD). Documents are now having types exactly like other computer processed objects. Document type is defined by its constituent parts and their structure. Definition of say report can be that it consists of title, optional author, then annotation and sequence of one or more paragraphs. Any document lacking title, given this formal definition, will not be taken as a report, as well as a sequence of paragraphs followed by annotation, despite those might be very close to report from the human reader point of view.

Since documents are now of known types, one can use special program called parser to process document claiming some type and check if all required elements are in place and in correct order for that type. What is more important, however, is that different documents of the same type can be processed in unified way. One can write more intelligent programs using knowledge about information structure of documents.

3.2.3 Data Independence

The main goal of SGML was to guarantee the fact that document encoded in it will be portable from one hardware and software environment to others without any loss of information. Its two characteristics described above address this requirement on abstract level, third one does it on byte sequence level. SGML provides generic mechanism of string substitution, that is simple hardware-independent way to instruct processor to substitute one string of symbols with another one. One obvious application of this mechanism — to provide common terminology across the document. Another less obvious one is to fight infamous inability of some systems to read character sets of others. One, for example, can pass one any needed graphical symbols to any system by descriptive encoding of unreadable symbols. Strings so defined are

called entities.

3.3 Advantages

What does SGML give in real life? For me, it gave an ability to produce documents in any required form. It happened to be very convenient for me to produce "MS Word papers" right in XEmacs environment I used to. All used programs are free and are distributed in source form, so I was able to install them on all systems I use. I've also note that it was very simple to begin work with SGML. Anybody who ever edited their own Web page won't be scared by SGML constructs.

Only later, during step by step learning of SGML universe, I've come to understanding that this technology gives many advantages over typical "desktop typography":

3.3.1 Productivity:

Sharply separated information entering and its formatting allow author to concentrate on explaining his thoughts, not deviating to moving text around the page and choosing styles.

3.3.2 Common stylistics:

It's easy to keep different documents in single style using common terminology. If style of terms have to be changed, it's simple to do in all documents at once, not touching their contents.

3.3.3 Reuse:

This term is familiar to almost any programmer. It is an ability to use parts of older documents in new one, with minimum changes. Part of document enclosed in single SGML element can be easily moved to other documents, referenced from them, repeated in different places in text.

3.3.4 Information longevity:

SGML is a simple and standard data storage format. One needs not to reformat or re-convert his data because of hardware or software platform change. Information is simply available forever. It carries along everything needed to create document.

3.3.5 Better data management:

With SGML one can define information elements and operations over them with any desired depth of detailedness. Marked elements can carry attributes defining their characteristics and features. For example, ID attribute can uniquely identify single paragraph or entire division, footnote, figure, task — any element, like this:

```
<para id=431>Information</para>
```

Since identifiers are machine-readable they can interlink bits of information or control it in different way. For example: Security control allows only chosen people read or change data. Automated data movement, for example, renovation of one piece of data can initiate change of related information in other applications.

3.3.6 Information share:

Structuring of document allows to compile it from separate parts, assigning parts to different people throughout an organization. Users can share information without copying it.

3.3.7 Portability:

In information networks connecting different computers, OS and applications, portability becames a key to common access to information. Since SGML doesn't rely on specific hardware or applications, one can easily pass a documents between different systems.

3.3.8 Flexibility in applications:

SGML allows to use information far beyond typical "desktop typography". Possible applications include:

- Web pages;
- Information databases;
- Diagnostic/expert systems;
- Electronic mail;
- Hypertext documentation;
- CD-ROM publishing;
- Interactive electronic manuals;
- eBooks;
- ....

3.4 Now Why SGML?

Let's now review my reasons of using SGML. Why SGML?

Why not MS-Word? Because it uses proprietary document formats. Because it is format of single application. Because it is unavailable in Unixes where I work. Because there aren't any good tools to automatically generate these documents. Because there are no good tools to version control and collaborative work. Because it is paper typesetting format, and there are no means to put any sensible structure into the documents. Because it gives horrible HTML when exports. Because typesetting quality is below any critics.

Why not TEX? Because it's too low-level language. Because it has too steep learning curve. Because it is paper typesetting language and not a structure markup[4]. Because there aren't many TEX-oriented editors out there. Because there aren't any good means to export to MS Word.

Why not HTML? Because the very term of HTML is too vague given current Web race for commerce and browsers wars. Because HTML still is oriented to visual appearance rather than pages' structure. Because it is very poor in expressive means, and there aren't any robust ways to extend it. Because there is no answer to the challenge of correct localization. Because there are no adequate tools to export to MS Word.

3.5 SGML versus WYSIWYG

For those new to SGML but used to "word processors" such as MS Word or WordPerfect, it might be interesting to compare these to SGML editors. These two classes of application, while being somewhat similar in task and visual presentation are vastly different in their architecture and functionality.

Typical word processor is a typewriter on steroids. It is meant to work with on screen (or on paper) text presentation with sets of styles: styles of symbols (font, size), paragraphs (indent, flush left/right, position within page), pages (headings/footers, footnotes), documents (paper size, table of contents) etc. Word processor does not know anything about internal structure of the document it edits.

On the contrary, SGML editor is oriented to document content and its structure. For it, division is an element containing subdivisions and paragraphs. For word processor it is just a space between one para-

[4] LATEX makes a step in right direction, but that's only one step.

graph typeset with ``Heading 1'' style and next one.

SGML editor helps author to navigate his document, to work with its content and not to bother with its presentation.

4 Software

To begin real work with SGML documents user needs at least two basic tools: editor and tools to export (format) his documents.

4.1 SGML Editor

SGML editor differs from both text editors and word processors. It supports structured texts unlike the formers and lacks support for visual formatting unlike the laters. Editor parses DTD of document being edited and uses it to guide the user. For example, if DTD allows for `<section>` element, which can contain only `<subsection>` and `<paragraph>`, user while edition `<section>` will be presented with choice of only those two nested element. SGML editor usually contains a means to navigate document hierarchy.

Popular SGML editors include[5] ArborText ADEPT*Editor, SoftQuad Author/Editor, psgml, Adobe FrameMaker+SGML, Corel WordPerfect and many others.

4.2 Formatting Tools

There are many tools to process SGML documents. Most of them are different formatters — tools to export SGML into other formats to print, view etc. Output formats might be anything, depending only on existing software and users needs. For example, I need to output my documents to HTML, RTF, LaTeX, PS, PDF and plain text.

SGML processors can have different architectures. There are already several generations of these tools (remember that SGML is two dozens year old). Usually they contain:

- analyzer, parsing SGML document, checking for its correctness and building internal representation of document's element hierarchy;
- core, offering basic functions of SGML processing (possibly forming single program with the analyzer);
- set of specifications, being a concrete program for specific document processing.

Syntax analysis of SGML is rather complex, and there aren't too many full scale SGML analyzers [6]. SP package by James Clark is reference implementation of it, and is freely available, known now as an OpenSP.

Specifications or stylesheets are written in core-dependent programming language. There are SGML processors programmable in Perl, Tcl, Lisp, Python, Java — virtually any known programming language.

To standardize document hierarchy representations and different functionality of SGML processors there are DSSSL standard [7] (Document Style Semantics and Specification Language). It specifies language nearly indistinguishable from Scheme.

4.3 Software I Use

As an SGML editor I prefer XEmacs, which includes SGML module psgml. As a kernel of SGML processor

[5] Please note that those were popular in 1996 when I've prepared this text.

[6] This was one of XML development driving force. Note that for this article you can substitute SGML with XML freely: everything told about SGML still holds true for XML.

[7] ISO 10179

I've choose CoST. To format my texts into HTML, RTF and LaTeX I've wrote set of specifications for CoST, known as TEItools. Parser used is nsgmls, parser from OpenSP suite. Version control in use is CVS.

4.4 Choice of DTD

When going into SGML technology, one have to choose one or several DTDs. Usually this process require some trial and errors. One can try industry-standard DTDs, for example, TEI Lite or DocBook. One can create his own DTDs, when working with typical document structure encountered in daily work. Since you can write an SGML processor transferring one DTD into another, you can more or less freely experiment with different DTDs.

5 SGML and Web

HTML, the language of markup for WWW pages, was once introduced as an SGML application. With time and explosive expansion of WWW, HTML started to quickly enhance to give authors more freedom and more control over page presentation. New elements and attributes such as <FONT> and <BGCOLOR> was aimed and visual presentation rather than structure markup. Emerged and gained widespread use non-markup means: imagemaps, Java applets, plugins etc. Many HTML elements became supported only by some browsers, or work differently in different browsers. Given all that, HTML is not really SGML anymore. Very few Web pages are created in accordance with HTML specifications and appropriate DTDs.

This problem is alleviated to some extent by cascaded stylesheets, standardized by WWW consortium. CSS1 separates stylesheet to visually represent elements from element markup itself.

Recently XML metalanguage was developed as a modern ancestor to SGML and HTML. It is a variant of SGML, oriented to Web application. It doesn't require DTD and the language itself is simplified. Rarely used complex constructs are removed. Because of that, XML analyzers are tiny, fast and there are already a lot of them.

XML removes one of the most boring HTML restrictions: fixed set of elements and fixed semantics of them. For example, if one wants to publish frequently asked questions in XML, he may use natural structure of appropriate elements, just like SGML allows:

```
<FAQ>
  <Q>What is XML?</Q>
  <A>XML is eXtensible Markup Language.</A>
</FAQ>
```

Specified in stylesheets appropriate visualization styles for <FAQ>, <Q> and <A> elements allow browser to display this document just like author meant.

6 Publishing from SGML

How can one publish SGML documents right now, without waiting for widespread XML acceptance? One can imaging several approaches.

If SGML is used to distribute large amounts of information, for example on CD-ROMs, SGML browser can be put on the same CD. There was several of them[8], but it seems they are not developed anymore.

If documents are distributed via Web, the easiest way is to translate them into HTML. This can be done when preparing publication or "on the fly", using CGI or any other server-side interface.

[8] Panorama Viewer or DynaText Browser

7 Food for Brain

I have only scratched here big and interesting theme of SGML and its applications. Those with Internet access can explore following resources.

7.1 Documentation

The XML Cover Pages contains huge amount of SGML/XML-related information.

Good introduction to SGML can be found at *http://www-tei.uic.edu/orgs/tei/sgml/teip3sg*.

There is fine book by Martin Brian SGML and HTML explained, available from *http://www.sgml.u-net.com/book/home.htm*.

7.2 Software

Home page of GNU Emacs is at *http://www.gnu.org*

Home page of XEmacs is at *http://www.xemacs.org*.

Free SGML toolkit SP is available from *http://www.jclark.com/sp*, as well as an DSSS-L processor Jade.

SGML processor CoST I've used is still available from Joe English at *http://www.flightlab.com/cost*.

My specifications for CoST to process TEI Lite documents are placed under GPL and available from *http://xtalk.msk.su/SGML/TEItools*.

About the Author Boris Tobotras is a Head of sotware laboratory of Jet Infosystems company, Moscow. This facilty develops and promotes Internet security related products. Prior to that, Boris worked as a service engineer and an expert (UNIX, Sun servers, clusters). Working on different projects, Boris have developed his own document preparation system based on free software (Emacs, CVS, bunch of SGML-related scripts known as TEItools). Before joining Jet Infosystems Boris has a developer and system architecht in small startup company. He can be reached by email `boris@xtalk.msk.su`.