CLASSIC COMMUNICATION

SUMMARY
- Describes the origins of Standard Generalized Markup Language from the perspective of one of its inventors
- Provides an anecdotal history of markup language development from the late 1960s to the 1980s

The Roots of SGML: A Personal Recollection

CHARLES F. GOLDFARB

The wonderful SGML '96 conference and its theme have awakened all sorts of interesting memories and ideas, going back over the nearly thirty years that I've been involved with generalized markup. I'd like to share some of them with you. But as this memoir will be short, idiosyncratic, focused on events in which most important figures in the SGML community did not participate, and taken from memory rather than research, I feel a disclaimer is in order.

PARALEGAL DISCLAIMER AND APOLOGY

All persons, organizations, and activities (hereinafter "Entities") mentioned in this brief reminiscence are described only as I've been able to recall them and such descriptions are not necessarily how those Entities would describe themselves. I may have missed a detail here or there in an Entity description, but the overall result is pretty accurate. The inclusion or omission of an Entity should not be interpreted as an indication of the importance of said Entity to SGML, the author, or the world at large, although the omission of really important Entities, as always, could cause a parsing error.

BEFORE GENERALIZED MARKUP LANGUAGE (GML)

In 1966 I was an attorney practicing in Boston, MA, two years out of Harvard Law school. I knew nothing about computers, but I knew there had to be a better way to produce documents than dictating them, reviewing a draft, marking up the draft with corrections, reviewing the retyped draft, and then, in frustration, seeing that the typist had introduced more errors while making the corrections.

My hobby was being a "rallymaster," a person who created the route instructions for sports car rallies. These normally read something like:

26. Left at light onto Jones Rd.
27. Right onto Smith St.

Mine looked like:

26. Left at light onto Jones Rd.
27. (Repeat instructions 20–26, substituting "left" for "right.")
28. Second right.

I also did things like hand out road maps of Yugoslavia and expect contestants to turn right in response to the instruction "go toward Sarajevo."

Eventually a friend told me that my rally instructions looked like computer programs. I said "Really? What's a computer program?" Shortly thereafter, in November 1967, I joined IBM to find out what kind of business would pay people for writing rally instructions. (The idea had also crossed my mind that the experience might equip me to attract legal clients from Boston's burgeoning high tech scene.)

My job was to design and install accounting systems for small businesses using not-so-state of the art punched card tabulating machines and occasionally, for the wealthier customers, a small computer. One assignment was different, though, and it eventually changed my career: installing a typesetting system for a local newspaper.

The system was an IBM 1130 computer, a machine the size of a desk with 8 KB of main memory, a 512 KB disk drive, a Teletype CX paper tape reader and BRPE paper tape punch, and a Photon 713 photomechanical typesetter. The assignment was my first experience with managing a machine-readable document database: I learned to roll the punched paper tape carefully so that it could be stored neatly in cylindrical waste paper baskets.

In the meantime, though I didn't know about it, the roots of generalized markup were being planted. Historically, electronic manuscripts contained control codes or macros that caused the document to be formatted in a particular way ("specific coding"). In contrast, generic coding, which began in the late 1960s, uses descriptive tags (for example, "heading," rather than "format-17").

Many credit the start of the generic coding movement to a presentation made by William Tunnicliffe, chairman of the Graphic Communications Association (GCA) Composition Committee, during a meeting at the Canadian Government Printing Office in September 1967: his topic—the separation of information content of documents from their format.

Bill went on teaching the world about "generic coding" under the auspices of Norm Scharpf and the GCA, then as now (and for all the years in between) unflagging believers, contributors, and promoters of the cause. At the same time, a New York book designer named Stanley Rice was publishing articles about "Standardized Editorial Structures," parameterized style macros based on the structural elements of publications.

INTEGRATED TEXT PROCESSING AND GML

In early 1969 I had had my fill of wiring tab machines and I was ready to resume my accustomed place before the bar. Instead, IBM convinced me to join its Cambridge Scientific Center and figure out how to apply computers to legal practice. That project required integrating a text editing application with an information retrieval system and a page composition program. The documents had to be kept in a repository from which they could be selected by queries. The selected documents could be revised with the text editor and returned to the database, or rendered for presentation by the composition program.

Standard stuff for SGML systems today, perhaps, but far from the way most people thought about document processing in 1969. So far, in fact, that the applications we needed to integrate were not only not designed to work together; they couldn't even run on the same operating system. Fortunately, we had access to CP-67, an early hypervisor that allowed multiple concurrent operating systems to run on the same computer and share files. The problem was that, even when Ed Mosher, Ted Peterson, and I finally got the programs to talk to one another, we found they each required different procedural markup in the document files.

I remember discussing this first attempt at integration with a senior IBM Industry Marketing manager named Steve Furth, whom IBM people thought of as the father of document information retrieval. (He'd written a book on the subject in the days when a database was as likely to use cardboard media as magnetic.) I mentioned that I thought it best to remove the procedural markup. He said something about that being wrong because the markup could have other uses. I said something like "you mean figuring out that some text is a caption because it is centered." He said "something like that" and referred me to Stan Rice's work. The rest, as they say, is history (or pre-history).

Later in 1969, together with Ed Mosher and Ray Lorie, I invented Generalized Markup Language (GML) to solve the data representation problem. GML was not merely an alternative to procedural markup, but the logical representation that motivated all processing. Ed recalls:

> *We called it Text Description Language at first, because I think that's what we thought it was. We certainly very early intended to use it as a common and general markup to be "translated" into Script [formatting] controls, ATMS & TERMTEXT & MTSC [formatting] codes, STAIRS [information retrieval descriptor] paragraph codes, as well as using an un-filled-in outline of tags as a prompter from which to create a new document.*

IBM decided that our work had value beyond the law office application, and the focus of our project shifted to text processing in general. The project was given a name, "Integrated Text Processing," and the first prototype was dubbed "Integrated Textual Information Management Experiment" (InTIME). Our manager, Andy Symonds, gave us permission to report on the work in the *Proceedings* of the 1970 Annual Meeting of the American Society for Information Science. But we were not allowed to write about TDL/GML because IBM had decided that it had serious product potential. We could only hint at the need for codes "to identify the structure and purpose of the parts of text. . . . The composition program would identify the codes as calls to stored formats; the retrieval program would use them for classification."

(There's more about InTIME in the 25th anniversary edition of the *Journal of the American Society for Information Science*, in the form of an annotated version of the original 1970 paper.)

Ed Mosher's technical notebook indicates that by 1971 we had succeeded with tag interpretation and multiple use (which Ed had implemented using Script set-symbols) and moved along into thinking about models and finite state machines. Ed that year developed the first production quality DTD, designed for the manuals for IBM's *Telecommunications Access Method* (TCAM). He was aided by TCAM publications manager, Joe Groppuso, whom I remember being particularly impressed that all the headings of a given head-level were formatted identically. That was a level of consistency they had not been able to achieve by their normal methods.

In 1971 the GCA Annual Meeting was held in Boston, and Norm Scharpf, a former IBM Marketing Manager, had inquired as to whether our lab had anything interesting to show off on a site tour. I agreed to demonstrate the InTIME prototype (without going into technical details), and to give a paper on "context editing." (That was heady stuff in 1971: you could actually navigate a file by searching for text strings instead of specifying line numbers!)

Norm invited me to a meeting of the "System X" committee, where I met Bill Tunnicliffe for the first time. There were 8 or 10 of us crowded into a hotel room in Boston, with steak dinners perched on our knees, discussing markup codes. I'm not sure about the technical results of the meeting, but I can say one thing for certain, having benefited from Norm's generosity in nurturing SGML and HyTime standards activities over the decades since: He's never fed another committee quite as well.

The GCA continued to work independently of our efforts in Cambridge. System X evolved into the "GenCode® concept," which recognized that different generic codes were needed for different kinds of documents, and that smaller documents could be incorporated as elements of larger ones. GCA and I eventually joined forces in 1978, when development of the SGML standard began.

(Bill Tunnicliffe became the first chairman of WG8, the ISO committee that developed and maintains the SGML family of standards. I mention it, although it is outside the period of this memoir, because Bill passed away on 12 September 1996, at the age of 74. We had a chance to honor him for his contributions in person at SGML '92. We won't have that chance again, so I want to thank him here.)

Later in 1971, when product development was imminent, I gave GML its present name so that our initials would always prove where it had originated. One of the ugly truths of technology transfer is that developers tend to be grateful for research work when first received, and virtually oblivious to it by the end of a lengthy development cycle, which in those days could take years and years. (Actually, it still takes that long today; they just bring the software to market much earlier in the development cycle.)

As a by-product of this exercise in branding, I coined the term *markup language*. I did so because it seemed to describe what we were doing more precisely than *programming language*, *encoding*, or even *metalanguage*, which would have satisified the acronymical requirements. I never imagined that it would come to be the name of a field of endeavor for tens of thousands of people, the subject of a journal (*Markup languages: Theory and practice*), or—thanks to Tim Berners-Lee and the World Wide Web—a household word.

GML finally saw the light of day under its own name in 1973, shortly before the release of its first (relatively primitive) implementation in the "Advanced Text Management System" (ATMS). Here is that first public appearance, from my paper, "Design considerations for integrated text processing systems," IBM Cambridge Scientific Center Technical Report G320-2094, May 1973 (but written in 1971):

> *This analysis of the markup process suggests that it should be possible to design a generalized markup language so that markup would be useful for more than one application or computer system. Such a language would restrict markup within the document to identification of the document's structure and other attributes. This could be done, for example, with mnemonic "tags." The designation of a component as being of a particular type would mean only that it will be processed identically to other components of that type. The actual processing commands, however, would not be included in the text, since these could vary from one application to another, and from one processing system to another.*

After the completion of GML, I continued my research on document structures, creating additional concepts, such as short references, link processes, and concurrent document types, that were not part of GML. By far the most important of these was the concept of a validating parser that could read a document type definition and check the accuracy of markup, without going to the expense of actually processing a document. At that point SGML was born—although it still had a lot of growing up to do.

IBM'S DOCUMENT COMPOSITION FACILITY: INDUSTRIAL-STRENGTH GML

In 1975 I took a position as a market planner for IBM's printer products in San Jose, CA. The move accomplished two long-held goals: Linda got to give our sons' snowsuits to charity, and I got a chance to prove there was a business case for a GML-based document composition product. The product was officially called the *Document Composition Facility* (DCF), but everyone called it "Script." It was derived from the language, designed by Stewart Madnick in the late 1960s, that was used in the Integrated Text Processing project.

I developed a cost-justification model, based on market surveys and case studies, that showed the enormous value of generalized markup over the procedural markup that was common at the time. On the strength of this, GML support was added to Script. Geoff Bartlett developed a macro language with built-in SGML functions, including controls for delimiter assignment and association of element types with processing procedures.

Peter Huckle, DCF's Chief Programmer, designed and implemented a notable "starter set" application, the precursor of the "General Document" in ISO 8879. The implementation was done entirely in the macro language, which was also available to the product's users. The application design was driven by the needs of IBM publishing, as chiefly articulated by Truly Donovan, the first professional document type designer. Truly was also the leader of what was surely the first multi-site, multinational, generic markup project.

Here's a markup example.

:h1.Chapter 1: Introduction
:p.GML supported hierarchical containers, such as
:ol
:li.Ordered lists (like this one),
:li.Unordered lists, and
:li.Definition lists
:eol.
as well as simple structures.
:p.Markup minimization (later generalized and formalized in SGML) allowed the end-tags to be omitted for the "h1" and "p" elements.

The *DCF GML user's guide* (IBM SH20-9160), which I wrote in 1978, includes the first published formal document type "descriptions" (DTDs), for this "General Document" and also for a "GML Markup Guide" document type. The General Document example, except for the delimiter strings, should look very familiar. It was not only the source for the homonymous DTD in ISO 8879, but also, thanks to Anders Berglund's championing of DCF at CERN, it was the source for the World Wide Web's HTML document type as well. The *User's guide* itself became the first working paper of the ANSI SGML committee (X3J6/78/33-01).

Before DCF, sophisticated GML applications existed only in a research environment. DCF was a commercial product, subject to all the constraints of what was then the largest and highest-quality software development organization in the world. And it was designed to support the requirements of the world's second-largest publisher. Although not technical in nature, these considerations proved vital for SGML. The World Wide Web, for example, succeeded commercially while many nobler, more technically interesting hypermedia systems proved only of academic interest, because of the Web's artful compromise in connecting technology to the needs of a real user community. DCF and GML succeeded for the same reason. Chuck Cooper was the product planner who made that vital connection for DCF.

DCF/GML, which is still widely used today, has probably produced more pages of output than any other single generalized markup product. It established beyond doubt the viability of generalized markup, and initiated the major change (still going on) in the way that large enterprises view their document assets. The SGML community owes a real debt to IBM and to the many talented and dedicated (present and former) IBM people who made it possible.

CONCLUSION: 30 YEARS OF GENERALIZED MARKUP

This memoir has focused on the roots of SGML: the people and activities that directly influenced the invention of the language and, ultimately, the development of the standard (two very different things). Those roots were solidly planted in the industrial sector, but it is worth noting that there were other descriptive markup activities going on in the academic world.

Brian Reid's Scribe system, for example, begun at Carnegie-Mellon in 1976, had independently arrived at several of the key concepts of SGML, though many years later. Brian, however, personally influenced SGML by encouraging me to write "A generalized approach to document markup" for *SIGPLAN notices* in June 1981. That paper eventually became—after a global change from "GML" to "SGML"—Annex A of ISO 8879.

I like to think of the history of SGML as—what else—a tree structure. One root—from Rice to GML to my basic SGML invention—joined at the base of the trunk by the other—Tunnicliffe to Scharpf and GenCode. The trunk, of course, is the extraordinary 8-year effort to develop ISO 8879, involving hundreds of people from all over the world. The products and tools that came after are the branches, the many applications the leaves, and they are all still growing.

And for all these 30 years, while the technologies of both computers and publishing have undergone overwhelming and unpredictable changes, the tree continues to bear the fruit that I described in 1971:

> *The principle of separating document description from application function makes it possible to describe the attributes common to all documents of the same type. . . . [The] availability of such "type descriptions" could add new function to the text processing system. Programs could supply markup for an incomplete document, or interactively prompt a user in the entry of a document by displaying the markup. A generalized markup language, then, would permit full information about a document to be preserved, regardless of the way the document is used or represented.*