Succession in standardization is usually a problem. The advantages of improvements are weighed against those of compatibility. If compatibility considerations dominate, a grafting process takes place. This process need not lead to compatibility. According to our taxonomy of successor standards, there are three types of succession (outcomes). Type I, where grafting is achieved, entails compatibility between successors, technical paradigm-compliance, and continuity in the standards trajectory.

In this paper, we examine issues of succession and focus on the Extensible Markup Language (XML). It was to be grafted on the Standard Generalized Markup Language (SGML), a stable standard since 1988. However, XML was a profile, a subset and an extension of SGML (1988). Adaptation of SGML was needed (SGML1999) to forge full (downward) compatibility with XML (1998). We describe the grafting efforts and analyze their outcomes.

We conclude that XML largely fits the SGML paradigm. SGML was a technical exemplar for XML developers. In contrast, widespread use of HTML exemplified the desirability of simplicity in XML standardization. The latter issue and HTML’s user market largely explain discontinuity in SGML-XML succession.

Priorities in standardization change. Rules for developing standards are revised. Standards are updated or become obsolete. This is part of the dynamics of standardization, irrespective of the area of interest. For example, a number of pressing problems in the field of Information and Communication Technology (ICT), our focus in this paper, show an interesting similarity with those in Esperanto, the neutrala lingva fundamento created by Ludwik Lejzer Zamenhof. Zamenhof addressed these problems in his foreword to the Fundamento Krestomatio, an anthology of Esperanto texts, and specifically in the foreword to the first (1903) and the fifth (1907) edition (Esperantista Centra Librejo, Paris, 1931). The aim of the anthology was to supply models and encourage the development of a common style in the use of Esperanto. Zamenhof feared that without such models different dialects would develop (1903). After rereading the anthology in 1907, Zamenhof noted that Esperanto authors spelt words differently (e.g. Jesuo and Jeso). Moreover, he ached to make some improvements. His taste for Esperanto had evolved. However, Zamenhof did not impose consistency. He restrained himself because he felt that continuity should take precedence over perfection. The future should decide which use of language was to survive. [Foreword, 1]

In the software sector, this dilemma is also strongly felt. It is worded in terms of ‘continuity in standards development’ versus ‘incompatible standards revisions’. For example, in 1988 the X.400 Series (message handling protocols) was updated and expanded in order to meet new needs. Standardizers discussed whether to choose downward compatibility with the 1984 version, or making a clean start with a new standard. But they could not reach full agreement [2]. Compatibility preserves earlier investments in R&D, training and equipment. But of what value is a standard if it is compatible with earlier versions but unworkable or not up-to-date?

In this paper, we further explore the dilemma between stable standards and incompatible revisions. The guiding question is how standardizers deal with heritage relations between standards. We study one such case: the

---

1 We very much thank Pim van der Eijk, Charles Goldfarb, Diederik Gerth van Wijk, and Willem Wakker for their insightful comments on this paper, and the European Commission DG Enterprise for contributing funding towards the underlying research. They are in no way responsible for and need not agree with its content.
Extensible Markup Language (XML) standard, which many see as the successor to the Standard Generalized Markup Language (SGML) standard.

**Conceptual Framework**

Many standards have *functional equivalents*. That is, they address the same problem and offer similar functionalities. Sometimes competition between them leads to standards wars (e.g. Wideband Code Division Multiple Access (W-CDMA) versus cdma2000 in mobile telecommunications [3]; Open Systems Interconnection (OSI) versus Internet standards series [4]). Such wars are interesting objects of analysis for technology innovation studies. Less attention has been paid to sequential relations between standards, that is, to the way standards (predecessors) are revised and succeeded by new standards (successors). Succession in standardization - whether with regard to company, *de facto*, consortium or formal standards - implies change and renewal. Renewal comes in various shapes: new editions, revisions (new versions, technical corrigenda, amendments, annexes etc.) and new standards. The successor addresses the same area, and is an improvement on its predecessor. It is designed to succeed and thus take over the predecessor’s role. New entrants in the market (standards users) naturally prefer and implement the successor.

Those who standardize the successor may or may not seek compatibility with the predecessor. They usually do, and will have good reasons not to seek compatibility (e.g. technically impossible or a change in the product). There are many kinds of compatible successors. The most common one is the downward compatible successor. It has more possibilities than its predecessor, and usually has a higher version number. For example, WordPerfect 5.1 software, a *de facto* standard at the time, could handle WordPerfect 4.2 documents. The reverse also occurs. That is, in practice sometimes companies only need a subset of standard’s options. They can then specify a standards profile which replaces the more elaborate original standard.

If the successor standard is compatible, compliant technologies should be able to work together with products that interoprated with its predecessor. Such is typically the aim when the successor is a new edition or a minor revision of a standard. Concerned are *incremental* innovations: the improvements made are part of normal problem solving [5]. The problem addressed by the old standard has not changed and - in essence - neither has the means to solve it. Both standards are part of the same *technological paradigm* [6]. Because the new standard supports earlier developments, the latter’s development is said to proceed along a *standards trajectory* (i.e. analogous to the meaning of ‘technological trajectory’ [7]). The new standard exploits its predecessor’s installed user base. It imbues continuity in technology development.

Evolutionary innovation theory distinguishes between incremental and radical innovations, but not between kinds of incremental innovation [8]. We develop an approach that identifies types of successor standards and heritage relationships. To start with, we draw an analogy with the process of grafting in horticulture. With grafting, a scion (added, improved functionalities of a new standard) is inserted upon a stock (prior standards functionalities). Scion and stock need to be closely related if the desired plant part is to survive [9]. With regard to standardization, we use the term *grafting* to refer to

the process of developing a standard (successor) based on another standard (predecessor) with the intention to improve the latter’s functionality and/or usefulness in other respects (e.g. ease or cost of implementation) while preserving *compatibility* [10] (and/or interoperability) with its predecessor’s context of use.

---

2 "Profiles (...) define conforming subsets or combinations of base standards (...) to provide specific functions. [They] identify the use of particular options available in the base standards (...)” (ISO/IEC JTC 1 N 5154, 1998.01.05)

3 “The term ‘technological paradigm’ refers to the set of technology-related cognitions which structure and focus the behavior of practitioners in a field of technology. Shared methods, heuristics and rules develop. Exemplars, that is exemplary achievements in the practitioner field further focus activities (...).” (Adaptation of [6, p.152] to applied sciences in [2, p.356].)

4 An exception is Christensen, who distinguishes between sustaining and disruptive technologies to explain why many exemplary, successful firms fail [8, p.XV]. However, his focus colors his terminology, which diverges unnecessarily from Kuhn and Dosi (e.g. [8, footnote 7, p.27]).

5 The ISO defines compatibility as the "suitability of products, processes or services for use together under specific conditions to fulfill relevant requirements without causing unacceptable interactions." [10] The term has a general use and is therefore preferred here. It includes upward and downward compatibility, and interoperability - a term sometimes preferred by information technology practitioners.
In other words, grafting refers here to a specific type of succession: compatible succession. It may indicate a simple update, but it may also involve novelty (e.g. the concept of message store, which was included in the downward compatible X.400 (1988) standard of the Comité Consultatif International Télégraphique et Téléphonique (CCITT), was innovative).

<table>
<thead>
<tr>
<th>Compatibility Relation →</th>
<th>Compatible Standards</th>
<th>Incompatible Standards</th>
</tr>
</thead>
<tbody>
<tr>
<td>Similar Standards ↓</td>
<td>Stability, Continuity</td>
<td>Discontinuous, Disruptive, Rivalry, Reengineering</td>
</tr>
<tr>
<td>Successor Standards (Improved Functionality)</td>
<td>Grafting (e.g. Update)</td>
<td>e.g. Embrace-and-extend</td>
</tr>
<tr>
<td>Unrelated Standards (Equivalent Functionality)</td>
<td>Cooperation</td>
<td>Competition</td>
</tr>
<tr>
<td></td>
<td>e.g. Bridges: Multi-protocol stacks, Routers</td>
<td>e.g. Fragmented Market</td>
</tr>
</tbody>
</table>

Table 1: Characterization of relations between standards with a similar functionality categorized in respect to whether they are successors or unrelated, and whether they are compatible or not. Examples of relational phenomena are included as bullets.

In the following, we develop a taxonomy aimed to identify and analyze the outcomes of standardization efforts rather than the intention and the process. The relevant dimensions are listed in Table 1 and 2. There are two main categories of incompatible successor standards. The first category concerns those that are paradigm-compliant but incompatible with their predecessor (discontinuous standards development). For example, the Internet Protocol version 6 (IPv6) is not compatible with IPv4. Although IPv6 standardizers would have preferred it, among other measures, a separate standard on “Transition Mechanisms for IPv6 Hosts and Routers” [11] has been developed. As it is, IPv6 is a Type II successor, incompatible and discontinuous, but paradigm compliant (see Table 2).

Some Type II successors are developed with the intent to be incompatible. In the past, situations have occurred where market players added extra functionalities to a standard so that implementations of the original and the extended standards did not interoperable. (E.g. Microsoft’s idiosyncratic Java Application Programming Interfaces threatened to fragment Sun Microsystems’s Java™ platform.) Where such behavior is used to frustrate the development of standards-conform technology, the company is said to apply the embrace-and-extend strategy. The outcome of this strategy is a Type II successor.

The second category of incompatible successors concerns improvements to standards that are radical and signal a paradigm shift (disruptive standards development). More radical changes can take place if standardizers do not seek compatibility. The decision to start afresh occurs when the circumstances of technology use or technology production have changed. New technologies may allow a simpler and more effective standards solution than earlier technologies did (technological anachronism). A radical revision will also be preferred to downward compatibility if the context of use changes and the standards requirements diverge too much (migration of use). If an incompatible revision is preferred, the standards trajectory will be disrupted. The successor and the old standard become rivals. Telefax standardization (CCITT, 1989-1992) illustrates how, in such a situation, rivaling islands of technology threaten to develop that fragment the market [12]. The problems started when a Group 3 digital telefax standard was proposed. At the time, there already were two standards: the Group 3 for analogue networks and the Group 4 for digital networks. The main opposition against the new proposal came from the Group 4 supporters. To prevent the Group 3 digital telefax standard from being accepted, they came up with a compromise. In the ensuing stalemate, the CCITT decided to accept the compromise as well as the proposal for a Group 3 digital telefax.

The succession of Group 3 for analogue networks by the Group 4 for digital networks is an example of a Type III succession (see Table 2). The rivalry that ensues is no different from that between unrelated equivalent standards. Bridges and interfaces will be needed, should at a later stage compatibility be desired (e.g. multi-protocol stacks between OSI and Internet; see Table 1).
### Successor categories

<table>
<thead>
<tr>
<th>Features of standardization ↓</th>
<th>Type I</th>
<th>Type II</th>
<th>Type III</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trajectory</td>
<td>continuous</td>
<td>discontinuous</td>
<td>disruptive</td>
</tr>
<tr>
<td>(In)compatibility</td>
<td>compatible</td>
<td>incompatible</td>
<td>incompatible</td>
</tr>
<tr>
<td>Paradigmatic change</td>
<td>incremental</td>
<td>incremental</td>
<td>radical (shift)</td>
</tr>
</tbody>
</table>

Table 2: Taxonomy of successor standards according to the characteristics of standards trajectory, (in)compatibility, and paradigmatic change. (Notably, while compatible standards characterize incremental change, incompatibility between successive standards need not indicate a radical change - or paradigm shift.)

According to the XML standard of World Wide Web Consortium (W3C), XML is a case of grafting [19]. It is positioned as a Type I successor, that is, as one that is compatible with its predecessor, the SGML standard of 1988. As the next sections demonstrate, in many respects it is, but not in all.

### Markup Languages

The XML standard has been well received by the IT community. The IT press mostly hails it as a functionally rich sequel to the HyperText Markup Language (HTML). Sometimes it describes XML as a welcome leaner version of the Standard Generalized Markup Language (SGML). In the following sections, we examine in the standards history, the technology, and the main applications of SGML and XML, respectively.

### SGML

Work on SGML started in 1969 with the development of a language called the Generalized Markup Language (GML) at IBM [13]. It was used to manage the large amount of complex industrial documents at IBM. GML was designed to record document structures independent of how these structures would subsequently be processed. For example, GML documents recorded headings, paragraphs, lists and figures –that is, information that is useful for editorial applications– but no formatting instructions. In this manner, GML separated the document description from the formatting languages (IBM used several such languages for printing). Also, because GML identified document structures, fragments of documents could be addressed and reused in different contexts.

In 1978, the ANSI took an interest in IBM’s work on GML. By the efforts of Charles Goldfarb, one of the three inventors of the language, work started on a more generic version: SGML. A major addition to the original design was made. In order to determine the validity of the document structure, and to support a wide variety of lexically different languages (e.g. different signs for start-tag), a formal description, or grammar, would accompany each document. Firstly, this grammar identified the type of components (elements) and their interrelations (content model). It was defined separately in what was called a Document Type Definition (DTD). Secondly, the DTD included a descriptive lexical and syntactical model that defined how the data was to be recorded, archived and distributed.

Working drafts were published between 1980 and 1983. In 1983, the Graphic Communications Association (GCA) produced the first SGML recommendation. It was adopted by the US International Revenue Services and the US Department of Defense (DoD). The International Organization for Standardization (ISO), too, became interested. It started a working group on SGML (ISO/IEC JTC1/SC18/WG8, now equivalent to ISO/IEC JTC1/SC34). This led to an international standard in 1986 (ISO 8879: 1986). An amendment was issued in 1988 (ISO 8879: 1988).

The 1988 version remained stable for eight years. In that period, ISO also published a number of SGML-related, supplementary standards. We mention two important ones (see Table 3 for other examples). The first is the Hypermedia/Time-based structuring language (HyTime) [14], a standard that addresses hypermedia relations. It offers a rich model for addressing and linking SGML documents as well as other type of information objects.
Another important standard, called the Document Style Semantics and Specification Language (DSSSL) [15], addresses styling. It specifies rules for transforming and formatting SGML documents. Furthermore, various tools and applications were created. Because the SGML concept was based on process-independent document structures, the same data in SGML documents could be understood by, for example, database and text processing tools. The range of SGML supporting tools included word processors, parsers, transformers, publishing engines, browsers, document management systems, and even dedicated programming languages and libraries. Areas of application included publishing (e.g. so used by the American Association of Publishers, IBM, and the US Department of Defense in the CALS initiative), text research (Text Encoding Initiative), and the exchange of product information (Society of Automotive Engineering J2008).

<table>
<thead>
<tr>
<th>SGML Objectives</th>
<th>Design goals for XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Source: ISO 8879:1986, Clause 0.2)</td>
<td>(Source: XML 1.0, 2nd ed., 2000)</td>
</tr>
<tr>
<td>a) Documents “marked up” with the language must be</td>
<td>1. XML shall be straightforwardly usable over the Internet.</td>
</tr>
<tr>
<td>processable by a wide range of text processing and</td>
<td>2. XML shall support a wide variety of applications.</td>
</tr>
<tr>
<td>word processing systems.</td>
<td>3. XML shall be compatible with SGML.</td>
</tr>
<tr>
<td>b) The millions of existing text entry devices must be</td>
<td>4. It shall be easy to write programs which process XML</td>
</tr>
<tr>
<td>supported.</td>
<td>documents.</td>
</tr>
<tr>
<td>c) There must be no character set dependency, as</td>
<td>5. The number of optional features in XML is to be</td>
</tr>
<tr>
<td>documents might be keyed on a variety of devices.</td>
<td>kept to the absolute minimum, ideally zero.</td>
</tr>
<tr>
<td>d) There must be no processing, system, or device</td>
<td>6. XML documents should be human-legible and</td>
</tr>
<tr>
<td>dependencies.</td>
<td>reasonably clear.</td>
</tr>
<tr>
<td>e) There must be no national language bias.</td>
<td>7. The XML design should be prepared quickly.</td>
</tr>
<tr>
<td>f) The language must accommodate familiar typewriter and</td>
<td>8. The design of XML shall be formal and concise.</td>
</tr>
<tr>
<td>word processor conventions.</td>
<td>9. XML documents shall be easy to create.</td>
</tr>
<tr>
<td>g) The language must not depend on a particular data</td>
<td>10. Terseness in XML markup is of minimal importance.</td>
</tr>
<tr>
<td>stream or physical file organization.</td>
<td></td>
</tr>
<tr>
<td>h) “Marked up” text must coexist with other data.</td>
<td></td>
</tr>
<tr>
<td>i) The markup must be usable by both humans and programs.</td>
<td></td>
</tr>
</tbody>
</table>

**Box 1: Aims of the SGML and XML standardizers.**

One of the important uses made of SGML was the HyperText Markup Language (HTML). It was developed by Tim Berners-Lee (CERN) for the World Wide Web, and first standardized by the IETF in 1995 [16]. HTML did not start out as a fully SGML-compliant application. It complied from the second version onwards. Many of the rules imposed on SGML documents were not—and are still not—enforced by browsers for HTML documents. Most browsers even accept and process invalid HTML documents.

**XML**

The W3C installed the SGML Editorial Review Board (ERB) in 1996 to develop XML [17]. Its members all had SGML expertise. Many also participated in SGML(-allied) ISO working groups. Apart from bringing the power of SGML to the web (XML), the ERB aimed to develop specifications for 'XML hypertext link types' and for DSSSL use in an Internet context. [18]

The review board became a regular working group (XML WG) the year after. Microsoft, one of the three active members of the XML WG, was an early adopter of XML for Internet Explorer. Netscape, likewise an active member, supported XML at a later stage. Together, these two companies covered a large share of the HTML market, which is of interest because at the time the web-browser was the main platform for XML document exchange. The W3C recommendation for XML 1.0 was published in February 1998 [19].
A wide range of XML applications, tools and standards has emerged since. Presently, the number of public applications exceeds 250. They address very different areas: publishing, electronic data interchange (XML/EDI), data modeling (UML/XMI), workflow management (WfMC), software engineering (SOAP), and so on. The functionality offered by XML-based software tools is equivalent to those for SGML. But, firstly, the advent of web content delivery, and the emergence of XML servers and middleware has led to additional XML functionality. Secondly, many libraries and XML extensions to existing programming environments have become available. Thirdly, the number of W3C XML-based specifications and standards by far exceeds those for SGML. W3C has produced additional recommendations on naming (namespaces), normalization (XML information set), transformation (XSLT), publication (XSL, Associating style sheets), implementation (DOM), addressing (Xpath) and linking (Xlink) of XML documents.

<table>
<thead>
<tr>
<th>Aspects ↓</th>
<th>SGML</th>
<th>XML</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ISO 8879:1986/Cor 1:1996</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ISO 8879:1986/Cor 2:1999</td>
<td></td>
</tr>
<tr>
<td>Standards committee</td>
<td>ISO/IEC JTC1/SC18/WG8 (later: ISO/IEC JTC1/SC34)</td>
<td>W3C XML WG</td>
</tr>
<tr>
<td>Initiators</td>
<td>IBM, GCA, AAP, DoD etc.</td>
<td>Microsoft, Netscape, Sun etc.</td>
</tr>
<tr>
<td>Primary goal</td>
<td>Publishing and document management</td>
<td>Web-based information interchange</td>
</tr>
<tr>
<td>Supplementary standards e.g.</td>
<td>Linking: HyTime</td>
<td>Xlink / Xpointer / Xpath</td>
</tr>
<tr>
<td></td>
<td>DSSSL</td>
<td>XSL and XSLT</td>
</tr>
<tr>
<td></td>
<td>Architectural forms</td>
<td>Namespaces</td>
</tr>
<tr>
<td></td>
<td>Property sets</td>
<td>DOM</td>
</tr>
</tbody>
</table>

Table 3: Overview of some main aspects of SGML and XML.

Grafting Efforts

The participants in XML development were SGML experts. They partly were or had been active in SGML or SGML-allied standards developments (e.g. DSSSL-O), and often knew each other from, for example, GCA conferences. The constituency of W3C's working group and JTC1's WG8 overlapped. Because of overlapping membership there was reciprocal influence. However, there was also a degree of group identification (we-them)\(^6\) and standards politics (e.g. personal differences and the Not-Invented-Here syndrome)\(^7\).

Compatibility Intent

When the W3C's working group started, it was clear that ")(... the ultimate goal of this effort is the creation of a form of SGML that can be used to transmit documents (or document fragments) to a future generation of Web browsers and similar Internet client applications." [20] But whether this XML would be an SGML subset, a

---

\(^6\) Tim Bray quoted in [30].

\(^7\) Private communication.
derivative, a conformance level, or an application profile was not yet decided and, as the chair of the working group writes, "our uncertainty has two levels: we're not sure where the optimum balance is between SGML compatibility and ease of implementation as a general goal, and we're not sure which specific features of SGML should be retained in XML. (...)"

The starting point was, that XML would be compatible with SGML. That is, existing SGML tools should be able to read and write XML data, and XML instances were to be SGML documents without changes to the instance.

Overlap between the constituents of the W3C and the JTC1 working groups kept the compatibility intent alive. In September 1996, soon after the electronic discussion list of the W3C working group started, Eve Maler posted a contribution which illustrates some of the compatibility concerns and dilemmas that were at stake [21]. For example,

"Who is the customer/audience for XML -- existing robust-SGML users, existing Web/HTML users who are not SGML-aware, or both? (...) I'd rather think of XML as an effort to define a cohesive SGML 'application profile' that benefits both tool creators and document creators, rather than a set of unrelated cool hacks that make it easier to write parsers. (...) What should happen when existing SGML documents (including valid HTML) are processed by XML tools? Should a 'round trip' between the two forms be possible, or is only XML->SGML or SGML->XML okay?”

Partly these were resolved. Some were impossible to resolve satisfactorily [22]. The outcome was a largely but not fully aligned XML specification in respect to SGML (1988).

**Technical relation between SGML and XML.** XML had the following properties: it was a profile, a subset and an adaptation of that SGML version [23]. Firstly, XML was a *profile* of SGML and, thus, complied with a predefined set of SGML options. These options addressed, for example, character set (XML applies ISO/IEC 10646 [24]), name case sensitivity (in XML all names are case sensitive), lexical form of the markup (in XML form of the start- and endcodes, processing instructions etc. are fixed), and minimization 9 (not allowed in XML; other so-called *SGML features* are also not supported). So, if XML had been purely a profile of SGML, an XML application would automatically have been an SGML application. XML documents would automatically have been processable by SGML tools.

However, XML was also a *subset* of SGML. That is, some SGML constructs were not accepted in XML anymore (e.g. so-called 'exceptions' and the *and*-connector in content models were dropped, and elements with mixed content required a fixed content model structure.) In order to enforce additional constraints, extra validation was required. SGML compliant software did not impose these constraints.

Had XML only been a profile and subset of SGML, all XML documents would have been processable by SGML-based systems. However, XML was also an extension of SGML. It allowed additional constructs. These were a true breach with the SGML tradition: XML documents that used these constructs were not processable by SGML (1988) systems. An important example is the *well-formed* document. In XML, the DTD is optional, which allows for a very flexible use of XML. But it requires that the document has a predictable lexical form, and explicitly encodes every element, attribute and data character. Conversely, DTD use was mandatory in SGML.

---

8 Two other aims are “For any XML document, a DTD can be generated such that SGML will produce "the same parse" as would an XML processor.”, and "XML should have essentially the same expressive power as SGML.” [31] hyperlinked in [20].
9 Minimization is the technique of omitting or reducing markup. For example, the omission of end-tags.
10 There are many reasons to part with DTDs. E.g. in 1996 many IT developers believed that the PC would become extinct, and that users would be connected by Java-based Web-centre thin clients with little processing power. In this context of use, DTD validation would make XML software too complex. Experience with SGML had shown, that, apart from editing, DTDs and validation were mostly superfluous. A second reason for DTD-less XML was that an influential person (Erik Naggum) had convincingly demonstrated that DTDs in the SGML declaration made the difference between data and markup processor-dependent, which was what SGML tried to avoid. (Private communication)
SGML included no such concept as well formedness. The optionality of DTDs in XML documents is but one - important - example of an extension.

To make XML compatible with SGML, design compromises were made. There are features in XML 1.0 that almost nobody uses (NOTATION, ENTITIES, attributes with ENTITY values), and features that a minority thinks superfluous (see the Simple Markup Language initiative) meant merely to confirm the relation with SGML. XML was grafted on the SGML trunk. However, the process did not lead to full compatibility with the SGML (1988) standard. As Charles Goldfarb remarked, in answer to a claim to SGML compatibility 'in spirit and in fact',

"If XML is a subset of SGML 'in fact', why should existing 'compliant' SGML tools require any adjustments? One can argue (...) whether a DTD-less markup language is a subset of SGML in spirit, but to be one in fact means that conforming XML documents must conform to 8879. At present, they don't." [25]

To become compatible, the SGML (1988) standard needed to be adapted.

Re-forging Compatibility

Two initiatives took place to address the incompatibility between XML and SGML. They focused on how XML documents could nevertheless be processed by SGML (1988) tools. Firstly, WG8 developed a Technical Corrigendum 2 (Cor 2: 1999) to re-establish compatibility with XML. It contained two annexes, the normative Annex K on Web SGML Adaptations and the informative Annex L for Added Requirements for XML. Annex K was an optional extension of SGML [N1929]. Together, the two annexes corrected a range of incompatibilities. The draft second corrigendum was finalized December 1997, a year after the XML draft. It was formally published in 1999, that is a year after the XML 1.0 recommendation. Full implementation of both technical corrigenda, which we refer to as SGML (1999) in Table 4, would make an SGML system XML compatible. In practice, most software providers focused on XML rather than on elaborating SGML along the lines laid down in the technical corrigenda.

Secondly, as part of the standards process in W3C, the XML working group included non-binding recommendations in the standard. Their implementation should allow XML documents to be processed by SGML (1988) software. The standard, however, gives no guarantee (i.e. it “increases chances”). Many XML systems ignore these guidelines.

Table 4 summarizes the compatibility status for different combinations of SGML and XML standard versions. In the previous, our focus was on the succession of SGML (1988) by XML, and the compatibility efforts that were undertaken. The outcome is that normative XML documents are not processable by SGML (1988) compliant software. The table further shows that SGML (1999) compliant software does handle XML documents.

Interestingly, SGML (1999) thus partly grafted on its successor to be, and largely on SGML (1988).16

---

11 Note that the SGML DTD is not only a formal description of the logical structure of documents. It also provides the rules for the syntactical form of the document. This includes rules for tag omission. In XML only logical rules are defined. This means that the document syntax itself must be completely transparent to a processor, i.e. it must be well formed.

12 Other examples are that XML allows several attribute definition lists to be declared for the same element, and the same name token may occur more than once in the enumerated attribute types of a single element type. This is also not accepted in SGML.

13 XML should be "(...) a lightweight, lean, mean, easy-to-learn on-ramp for SGML (...)" [28]

14 Annex L, an informative annex, is explicitly XML-oriented. The final text of Annex K [N1955] is foremost presented as the outcome of the ongoing review - largely independent of XML. "This annex remedies defects revealed by the multiple adaptations of SGML for the World Wide Web (WWW), intranets, and extranets. The annex corrects errors, resolves ambiguities for which there is a clear resolution that does not cause existing conforming documents to become non-conforming, and provides a choice of alternative resolutions for other ambiguities. Although motivated by the World Wide Web, applicability of this annex extends to all uses of SGML. [32]

15 Although SGML (1999) does not enforce many of the constraints of XML (e.g. "&").

16 In JTC1, an ongoing review of SGML took place. Within this framework, for example, the principle was formulated that any conforming SGML (1988) document would continue to conform under the provisions of future SGML versions (i.e. guaranteed upward compatibility"(...) [This] should not be construed to mean that no changes can be made to ISO 8879. To meet evolving user requirements, for example, some changes of the following types are possible without violating the above principles: (a) Relaxing restrictions, (b) Adding new constructs, (c) Partitioning existing optional features, (d) Introducing options to allow the suppression of troublesome existing constructs, when experience indicates that the constructs tend to induce user errors with serious consequences." [N1855, Att.1, Att.2] However, DTD-less SGML was never at stake.
Processable by →

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SGML document (1988)</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>XML document (1998), normative part only</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>XML document (1998), also recommended parts (&quot;Interoperability&quot;)</td>
<td>&quot;increases chances&quot;</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 4: Overview of the state of compatibility between the most relevant combinations of documents and software that conform to SGML with (1999) and without (1988) technical corrigenda, and to XML 1.0 with and without the recommended part.

Analysis of grafting outcome

According to the taxonomy developed in section 2, the factors that distinguish a Type I successor (grafting) from Type II and III successors are the ‘standards trajectory’ and ‘paradigmatic change’. To analyze why this discrepancy arose, we explore whether the transition from SGML (1988) to XML represents a paradigm shift, and what sort of causes led to discontinuity in standardization.

Paradigm Features

Briefly recapitulating, a paradigm is a set of shared views, heuristics, exemplars, etc. that guide and structure the way a practitioners community normally solves its problems. In other words, the question is which paradigmatic features structure the way SGML practitioners (i.e. standards developers and implementers) work.

That which characterizes the SGML problem and started things off was the need to reuse information fragments and share documents across publication systems in a future-proof way. IBM addressed the problem by separating the syntactical and the logical document structure. It determined – as it were – the sort of answers with which to solve the puzzle and laid the fundaments for the SGML approach.

XML developers were raised with the principles of SGML. SGML was a technical exemplar and an exemplar in respect to the functionality it could offer: the identification, exchange and reuse of information fragments in different contexts. XML, too, was initially document-oriented. Furthermore, in discussions XML was called a "lean-and-mean dialect of SGML."[26]. It was to become a simpler version of SGML. It could become so, for example, because different from SGML it could refer for character sets to Unicode and the ISO 10646 standard. Simplifications like these emphasize continuity rather than deviation from SGML features. Except for the DTD-less document, which we would typify as a shift within the SGML paradigm, the general SGML mindset and strategies also apply to XML.

Causes for Discontinuity

Context of Use. What explains the discontinuity between the standards successors? Firstly, the web-based context of use had little in common with the context of SGML in the 1980s (technological anachronism). The technology of the 1990s offered new opportunities and posed different constraints. The domains of use shifted together with the practitioners involved (migration of use). See Figure 1. Although the information modeling approaches of SGML and XML were in principle identical, the SGML problem was foremost how to manage the company-internal, complex flow of documents. XML, on the other hand, developed as a solution to the limitations of HTML in respect to company-external, web-oriented document exchange. See Table 5.

In the JTC1 SGML process, DTD-less documents are first mentioned in the review report that followed the SGML ’96 Conference, where the first XML draft was presented. "(...) There shall be a defined simple-as-possible-but-useful conformance level of SGML, tentatively called 'simple SGML'. It will have a simplified declaration syntax and it will be possible to parse the element structure without reference to the DTD.(...)" [N1893, N1925]
Context of Standardization. Some thought the XML market would only be of interest to SGML users [27]. Others hoped to target the huge, well-funded, energetic Web population. [28] There would be an important marketing advantage in being able to say "XML processors can read HTML". [29] Therefore compromises to XML compatibility with SGML were considered that left the installed base of HTML untouched. The deliberations are well illustrated by the following quotation, and explains part of the discontinuity in the SGML trajectory. "(...) For the 99% of the world that doesn't care a bit about SGML (...). They know HTML, so we must make things look like HTML. But when it comes to adding the important things that HTML doesn't have, we should make them as attractive as possible. (...) The SGML folks need a standard, as well as capability so they will continue to need SGML. But for the rest of the world, clean extendible markup is the biggest need, not SGML compatibility." [22]

<table>
<thead>
<tr>
<th>Causes for discontinuity ↓</th>
<th>SGML</th>
<th>XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information problem</td>
<td>Company-internal</td>
<td>Company-external</td>
</tr>
<tr>
<td>• Orientation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Context of use</td>
<td>1980s (mainframes etc.)</td>
<td>1990s (Internet, chips etc.)</td>
</tr>
<tr>
<td>• Technology</td>
<td>Publishing</td>
<td>B2B, application integration</td>
</tr>
<tr>
<td>• Domains</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standardization</td>
<td>GML</td>
<td>SGML, HTML</td>
</tr>
<tr>
<td>• Frame of reference</td>
<td>ISO: stability, accountability</td>
<td>W3C: pragmatism, speed</td>
</tr>
<tr>
<td>• Standards body: culture</td>
<td>Ubiquitous applicability</td>
<td>Simplicity, implementability</td>
</tr>
<tr>
<td>• Problem, emphasis on</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5: Causes for discontinuity: differences between the problems, context of use, and context of standardization of SGML and XML.

However, more influential was the successful spread of HTML use. It was an exemplary achievement for XML developers. The standardization implication was: aim for simplicity. If we compare the SGML aims with the design goals of XML, the latter's emphasis on ease of implementation and usability is salient. (See Box 1.) Simplicity was difficult to align with compliance to SGML. (Moreover in section 3 we mentioned that, besides bringing SGML to the Web (XML), the XML initiators also aimed to include HTML issues like Styling and Linking as part of the XML working group charter. This underscores the relevance of HTML as a relevant, second frame of reference for XML development.)

Lastly, XML grafting did not take place in the ISO. It took place in the W3C environment. This non-ISO environment made it easier for XML developers to deviate from standard SGML solutions. As other cases testify, a change of standards body facilitates incompatible succession (i.e. the Not-Invented-Here syndrome).
Conclusion

Our case focused on XML as a successor of SGML (1988). We started by introducing a taxonomy for succession along three dimensions: continuity in the standards trajectory, (in)compatibility, and paradigmatic change. The case showed a discrepancy between stated grafting of XML on SGML (Type I succession: compatibility) and the actual standardization outcome (Type II succession: incompatible). XML deviated from the decade-old stable SGML standards trajectory. This did not occur because of any paradigmatic change in the way SGML principles were used in XML - although the abandonment of DTDs was a revolutionary step. There were other reasons to deviate.

XML’s context of use differed from SGML’s. It was company-external and web-oriented, where SGML foremost had a company-internal focus. Furthermore, the HTML context of standardization played a role. The spread of HTML use was exemplary for XML developers. The message was: aim for simplicity in standardization, which at times conflicted with aim of compatibility with SGML.

In other words, two exemplars guided XML development and colored the its heritage relation with SGML: SGML, which was a technical exemplar and an exemplar in respect to the functionality it could offer, and HTML, which was an exemplar in terms of standardization and standards diffusion.

Dilemmas regarding compatible succession are, as the previous sections illustrate, often of a mixed socio-technical nature (technical, implementation, esthetic, etc.). Characteristic for dilemmas is that the conflicting arguments are both persuasive. Unsurprisingly, XML and SGML developers as well as Zamenhof can be seen to switch between them or combine conflicting ones.

References

ISO/IEC JTC1/SC18 Documents:

[18] Jon Bosak: "Re: Welcome to w3c-sgml-wg@w3.org!", one of the first submissions to the discussion list of the W3C SGML Working Group, [33], Aug 28 1996.
[26] "Re: Capitalizing on HTML (was Re: equivalent power in SGML and XML)" , C. F. Goldfarb, [33], Sept. 19 1996.
[27] Charles F. Goldfarb, 'Re: Make DTDs optional?', [33], 30 Sep 1996
[28] Tim Bray, 'XML, HTML, SGML, life, the universe, and everything', [33], 08 Nov 1996.
[29] Tim Bray, 'Recent ERB votes', 06 Nov 1996. In: Reports From the W3C SGML ERB to the SGML WG And from the W3C XML ERB to the XML SIG, Compiled for the use of the WG and SIG by C. M. Sperberg-McQueen, 4 December 1997.
[33] A contribution to w3c-sgml-wg@w3.org discussion list.