Mat-2.108 Independent Research Projects in Applied Mathematics

The OtaStat Publication Platform

Jussi K. Virtanen

54233J

July 19, 2005

Contents

List of Abbreviations

ASCII American Standard Code for Information Interchange

BMP Basic Multilingual Plane

CSS Cascading Style Sheets

DCMI Dublin Core Metadata Initiative

DTD Document Type Definition

GIF Graphics Interchange Format

HTML Hypertext Markup Language

ISO International Organization for Standardization

MathML Mathematical Markup Language

SP Supplementary Plane

URI Uniform Resource Identifier

URL Uniform Resource Locator

UTF Unicode Tranformation Format

W3C World Wide Web Consortium

WWW World Wide Web

XHTML Extensible Hypertext Markup Language

XML Extensible Markup Language

XSL Extensible Stylesheet Language

XSLT XSL Transforms

1 Introduction

Although Tim Berners–Lee invented the World Wide Web (WWW) at the European Organization for Nuclear Research (CERN) in 1989, the Hypertext Markup Language (HTML), the *lingua franca* of the Web, has always contained very limited capabilities for mathematically oriented publishing [34]. In 1998 the World Wide Web Consortium (W3C), the governing body of Web standardization, released the new Mathematical Markup Language (MathML), a purpose-built language for the task of embedding mathematics into documents [23]. Unfortunately support for MathML in Web browsers has so far been weak and, thus, the standard method for inclusion of mathematics into plain HTML pages is still the use of generated images depicting mathematical constructs.

Scientific papers and other mathematical documents in the academia are often prepared using Leslie Lamport's LaTeX, which is based on Donald E. Knuth's TeX typesetting system [27, 25]. Although LaTeX is also used for Web publishing with dedicated tools that generate HTML documents with embedded images for mathematical symbols and equations, the traditions of LaTeX-based tools greatly differ from the established ways of working on the World Wide Web.

The OtaStat Web publication system, developed in the OtaStat project at the Systems Analysis Laboratory at Helsinki University of Technology, Finland, during the years 2003–2004, attempts to combine native World Wide Web methods and techniques, such as the Extensible Markup Language (XML), with the benefits of LaTeX-based tools. In this paper, the applied technologies are described both in general and in the context of the implemented system. First, a short description of the OtaStat project and a rationale for the system development is given in Section 2, then, start-

ing from page 5, Section 3 provides a concise presentation of the underlying standards and technologies used. Section 4, starting from page 17, explores the architecture and implementation of the system and, finally, Section 5 concludes by evaluating the implemented system and identifying the needs for further research.

2 Background

The OtaStat project, led by Ilkka Mellin at the Systems Analysis Laboratory at Helsinki University of Technology, produces World Wide Web resources for students of statistics. The study materials are organized so that they can aid self-access students as well as act as a support for lectures. One of the intents of the project is to establish a publicly available online encyclopædia of statistics.

The encyclopædia is in the centre of the entire OtaStat project. The aim is to combine traditional reference articles, interactive components that illustrate the concepts of the articles, and direct links to the statistics courses offered at the university. So far the project has produced material that fits the aforementioned content types, but the presentation of the material is clearly bound to the courses offered. Unifying structure and links between the different content types are missing.

The majority of the current material is produced using Microsoft Word, Microsoft Powerpoint, and Design Science MathType equation editor, and is offered on the project Web site in Adobe Systems' open Portable Document Format (PDF). Computer exercises require either NCSS, Mathworks Matlab, or Microsoft Excel, depending on the topic. Interactive applets that, for example, illustrate the parametrization of distributions are implemented on Sun Microsystems' Java platform.

The initial planning of the content management and publication system began in June 2003 by investigating the currently used techniques and their fitness for the encyclopædia. One quickly identified problem was the reliance on PDF for basically all text documents. While the PDF document format is excellent for printed material, usability studies have shown online PDF documents to be problematic for average site visitors [1, 30, 31]. Clearly

the advanced features required by an encyclopædia, such as extensive inter-document references, would only make matters worse.

Although Microsoft Word and Design Science MathType have the option to convert documents into HTML, on a closer inspection the conversion appears to be of very low technical quality—the resulting documents are, in practice, "broken" and do not honor the standards of the World Wide Web Consortium.

Because not a lot of material destined exclusively for the encyclopædia existed yet, the decision to start seeking for other suitable tools was not hard. The TEX-based systems LATEX2HTML and TEX4ht, arguably the most popular platforms for mathematical publishing on the World Wide Web, were the most promising candidates. The latter was also already used in another similar project at the university. Initial tests revealed that neither of the systems produces output of as high aesthetical quality as the TEX engine is capable of. In addition, the systems were quite hungry for resources.

Finally the development work on a brand new system started. The idea was to base the work on established standards and technologies and to come up with a lightweight, flexible, and easily extensible solution to the problem.

3 Technical Foundation

This section presents the fundamental standards and technologies that are used in the OtaStat system implementation. Both benefits and drawbacks of the chosen solutions are discussed and, where applicable, competing solutions and other related technologies are mentioned. The discussion begins with low-level concepts and gradually advances towards high-level abstractions.

3.1 Unicode

Although a computer is, in essence, just a powerful calculator and therefore able to process only purely numeric data, computers are nowadays used for a variety of tasks beside bare calculation. This is possible via the utilization various encoding methods: non-numeric information is in some systematic way transformed into numbers and back. Most types of information require such a transformation—pictures, sounds, even plain text—, and the approach and techniques used in the encoding process depend heavily on the nature of the information.

While text encoding is in principle relatively straightforward, the different writing systems and scripts used around the world complicate matters considerably. The early encoding schemes, such as the American Standard Code for Information Interchange (ASCII), bypassed these issues simply by being national standards. Later international standards, however, have had varying success while tackling the challenge.

On most modern computers, the basic unit of information is octet, 8 bits. In text encoding, the basic unit of information is *character*, a logical entity, whose exact role is highly dependent on the used writing system [37]. In many encoding schemes a character is encoded as an octet and, consequently, the

maximum size for the *character set* is 256 characters.

The still widely used ISO 8859 standard series is an enlightening example of the problems related to 8-bit text encoding. The series contains sixteen standards that specify multiple variants of the Latin character set as well as Arabic, Cyrillic, Greek, Hebrew, and Thai character sets, which augment the 7-bit ASCII character set with at most 128 additional *code points* each. The character sets are mutually exclusive: in different character sets, a specific code point is assigned to different characters. Thus, it is impossible to combine, say, Greek passages into a typical Finnish text, because the character set ISO-8859-7, needed for the Greek alphabet, does not contain, for example, the character 'Ä' LATIN CAPITAL LETTER A WITH UMLAUT [22].

The development of a truly sustainable solution to the various problems related to text representation on computers was initiated by Apple Computer in 1988. Three years later the Unicode Consortium was formally found and other key players in the computer industry, including IBM, Hewlett–Packard, Microsoft, and Sun Microsystems, had joined the effort. Today the Unicode Standard is implemented on almost all major operating systems for desktop computers and embedded systems.

In the core of the Unicode Standard is the Unicode character set, which consists of 1,114,112 code points. In the latest version of the standard 96,382 of these code points have assigned characters representing the scripts of virtually all modern and many classical languages [37]. The entire code point space is divided into 17 distinct elements, *planes*, of which the first is called the Basic Multilingual Plane (BMP) and the subsequent Supplementary Planes (SP). While each of these planes consists of 65,536 code points, mainly the three first ones, BMP and SPs known as Supplementary Multilingual Plane and Supplementary Ideographic Plane, are in active use. The character set

Glyph	Α
Code Point	U+0391 GREEK CAPITAL LETTER ALPHA
Octet Stream	C6 87

Figure 1: The Unicode character encoding scheme.

is covered not only by the Unicode Standard but also by the international standard ISO 10646 [20].

3.1.1 UTF-8

In eight-bit character encoding schemes, such as the ISO 8859 series, the logical and the physical representation of a character, that is, the code point and the corresponding binary string, are the same. In Unicode, however, these two concepts are decoupled, and the physical representation of a code point is dictated by the chosen *encoding form* [37]. Three different encoding forms are specified in the standard: UTF-8, UTF-16, and UTF-32. In these a code point is represented as a sequence of one or more *code units* of the length of one, two, or four octets, respectively [37].

In UTF-32 a code point is stored always as a single code unit, a 32-bit word. This is possible, because the entire Unicode character set can be represented in just $\log_2 1{,}114{,}111 = 21$ bits. However, this encoding form is not particularly effective as the code unit has excess capacity of $32 - 21 = 11$ bits. The variable-length encoding forms, UTF-8 and UTF-16, are therefore often preferred to UTF-32. They both have unique traits that make them preferable for specific tasks: while UTF-8 proves more efficient for storage purposes of most Western languages and is completely backwards-compatible with ASCII, processing of UTF-16 encoded characters is simpler and thus faster, and UTF-16 is also more efficient storage format for languages that

utilize the Chinese Han script, for example [37].

Figure 1 on page 7 describes the three layer architecture of the Unicode approach to text encoding. The topmost layer consists of *glyphs*, symbols of characters or character combinations. In the Figure, the glyph represents U+0931 GREEK CAPITAL LETTER ALPHA. As it happens, the glyph of that letter is the same as of U+0041 LATIN CAPITAL LETTER A. The middle layer, which is based on code points, could be thought as the logical layer, a precise description of the text. Finally, on the bottom lies the physical layer—an octet stream conforming to an encoding form, in this case UTF-8.

3.2 TEX

The invention of a computer has affected significantly typography, the craft of printing. One of the remarkable pioneers of digital typography was—and still is—TEX, a sophisticated typesetting system developed by Donald E. Knuth, a professor emeritus at Stanford University [25]. TEX was designed during late 1970s to early 1980s, and since 1990 the system has been stable and feature-complete: only fatal errors have been corrected [26]. Its source code has been freely available since the very beginning, and nowadays ports exist for most computer systems in use.

TEX is a batch processing system that takes input in the form of a eight-bit encoded plain text document and produces output in the form of a device independent (DVI) file. Handling of the input data inside the TEX processor occurs in four distinct units of which the two first are concerned with interpreting the input data and the remaining two with input serialization, and output composition and pagination, respectively [14]. The operation of the program is reminiscent of a compiler: an input file, the source code, is compiled to, in TEX's case, a typeset document ready for printing to paper

command[*optional-arguments*]{*mandatory-arguments*}

Figure 2: The syntax of a typical LaTeX command.

or previewing on a display device.

The quality of TeX's output is generally considered excellent, thanks to the elegant techniques used in the micro-typography engine, such as automatic ligatures and fine control of kerning, and in the macro-typography engine, the sophisticated pattern-based hyphenation and the novel total-fit algorithm for optimal line-breaking inside a paragraph, for example [36]. Despite this, TeX is probably best-known for its mathematical capabilities. In 1984, Knuth described the system as follows [25]: "a new typesetting system for the creation of beautiful books—and especially for books that contain a lot of mathematics." Due to the open nature of the system, TeX has been further enhanced by a variety similarly open projects.

3.2.1 LaTeX

The input for a TeX processor consists of the content itself and fairly low-level instructions that define font changes, special characters, and other control information embedded to the text flow. As writing documents directly in the format TeX expects would be awkward, a multitude of macro packages are available. Plain TeX, by Knuth himself, and LaTeX, originally by Leslie Lamport, are two of the most important ones [25, 27]. LaTeX's approach is to hide the formatting details and concentrate on the semantic structure of the document. Its combination of simplicity and flexibility has made it the most popular TeX macro package.

The LaTeX language consists of various commands that are used to convey the structure of the document. Majority of these commands are based on the

```
x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}
```

Figure 3: Solution to the second-degree polynomial in LaTeX.

syntax shown in Figure 2 on the previous page; for example, a section called "Example" is started with the command `\section{Example}`. Note that in this case, the command has no optional arguments. The exact appearance of the typeset section title depends at least on the document class and possibly also on the optional packages in use. The syntax for mathematics is different from the normal syntax: as can be seen in Figure 3, the command for binary subtraction, for example, is just '−' U+002D HYPHEN-MINUS. Thanks to the short and logical command names, the LaTeX mathematics language is both fast to write and reasonably easy to read.

3.3 XML

The World Wide Web Consortium released the first version of the Extensible Markup Language (XML) recommendation in 1998. XML has quickly become one of the most used structural information storage formats in information and communications technology. XML, like its predecessor Standard Generalized Markup Languge (SGML), is a meta language that specifies the grammar of a generic structural language [9, 19]. With XML, one can specify new languages, known as *document types*, easily. The most important benefits of XML are the increasing interoperability of applications—many programs support importing and exporting of data using some document type—and the prolific supply of applications for document processing in different development environments.

Technically XML is a subset, or *application profile*, of the more complex SGML, which is based on the earlier Generalizer Markup Language (GML),

developed by IBM in 1960s [15]. The versantile grammar of SGML is a mixed blessing: document creation is flexible but, as a consequence, document interpretation is hard and *parsers*, language interpreters, are complex programs. In contrast, the more strict grammatical rules of XML ease interpreting documents without hindering the power of expression of the language.

The World Wide Web Consortium has specified two presentation forms for an XML document. In the primary specification, the language is defined as a context-free grammar using the Backus–Naur Form (BNF) and Unicode code points as terminals [9, 37]. This presentation form is natural if the XML document is an octet sequence in computer's mass storage. The other presentation form, Document Object Model (DOM), provides a tree abstraction of the document [28]. It can be used to manipulate the document in the random access memory (RAM).

Figure 4 on the next page contains a simple XML document composed of four elements: `addressbook`, the root element of the document, two `person` elements and a `mobile` element. The `person` elements have both two attributes: `firstname` and `surname`. The order of the attributes does not matter. Two attributes with the same name are not allowed on a single element. The latter `person` element is the only empty element in the document.

3.3.1 Namespaces

A namespace is a set of names in which each name is unique [18]. Namespaces are frequently encountered in different contexts. The international telephone numbering plan, Uniform Resource Identifier (URI) addressing in the World Wide Web, and the qualified naming scheme in the Java programming language are some examples of namespaces [5, 16]. In each of these examples uniqueness of names is of importance. Yet many other systems in which the

```
<?xml version="1.0" encoding="UTF-8"?>
<addressbook>
  <person surname="Duck" firstname="Donald">
    <mobile>+121 232 313</mobile>
  </person>
  <person firstname="Scrooge" surname="McDuck" />
</addressbook>
```

Figure 4: A simple XML document.

uniqueness property would be useful or otherwise desirable are not namespaces: names of humans and geographical locations among others. In these naming schemes one name may represent simultaneously many things and, consequently, result in so-called name clashes.

In XML, an explicit namespace mechanism for element and attribute names is specified. With it, a potentially ambiguous element name, such as "title", can be associated with an URI, the namespace name [8]. As the semantic context of the element is fixed to the namespace, two elements with the same name but different namespace are handled as separate entities: "title" element belonging to the namespace A may refer to a book title in a store catalog while "title" in namespace B refers to a window title in a data file of a desktop application.

3.3.2 XHTML

HTML is an SGML document type, a language defined in terms of the SGML meta language [34]. As it was designed before XML, it does not enforce the strict syntactic restrictions of XML but allows, for example, boolean attribute minimization and other short-cuts provided by SGML [34, 19]. After W3C

issued the XML specification in 1998, the next step was to produce a version of HTML as an XML application, the Extensible Hypertext Markup Language (XHTML) [32]. The initial version is essentially the same as HTML: semantics of elements or attributes have not changed at all.

The second version, XHTML 1.1, however, has some significant changes [3]. Many presentational elements and attributes, such as the `i` element for italic font, are deprecated in favor of elements that emphasize semantic value, for example the `cite` element for a citation [3]. Yet, a more profound development is the *modular document type* and, as an example of a such concept, the Modularization of XHTML [2]. The language is divided into multiple small and largely independent *modules* that each fulfill a particular function. Some examples of languages that are based on these modules are XHTML Basic, a simplified language destined for mobile terminals, and XHTML-Print for technically constrained printing environments [4, 6]. While both of these languages are merely subsets of the core language, the modular approach allows one also to augment the XHTML language with new modules or merge XHTML modules into other languages.

3.3.3 XSLT

Extensible Stylesheet Language Transformations (XSLT) is a language for transforming an XML document to either an XML document, a HTML document, or plain text [11]. Like XML, also XSLT is specified by the World Wide Web Consortium in a series of technical documents. Third parties provide both free and commercial XSLT processors, products that implement the language according to the specifications, for a variety of development environments.

The XSLT language revolves around the following notions: *source tree*,

stylesheet, and *result tree* [11]. The source tree refers to a tree representation of the source document. The structure of the tree is largely similar to the aforementioned DOM of an XML document [12]. The stylesheet itself is also an XML document consisting of elements in the XSLT namespace, primitives of the language, and, usually, elements in one or more target document namespaces. As the stylesheet is applied to a source document, the source tree is traversed starting from the root element and proceeding according to the template rules of the stylesheet.

A template rule is defined using the `template` element [11]. The element may have multiple attributes of which the most important is `match`: it specifies a pattern that is used as a trigger condition for the template rule. The pattern is an XPath expression; XPath is a simple language for the sole purpose of addressing inside an XML document [12]. While the language contains quite an extensive set of primitives, template rules are arguably the most important. Inside a rule the stylesheet can contain instructions that create new elements, iterate over portions of the source tree, and so on while constructing the result tree.

3.4 Dublin Core

As well as it is important to define how data is stored and processed it is equally important to define how metadata, information describing the content, is handled. The uses for metadata in general are vast. For example, a reference to another paper in an academic paper relies on metadata properties of the object: its author, title, publisher and so forth.

Dublin Core is a relatively new metadata scheme that is designed for the Internet and the World Wide Web in particular [17]. It is developed by an open organization, the Dublin Core Metadata Initiative (DCMI), and

standardized by ISO [21]. While Dublin Core is not tied to any specific technologies, it is often used in conjuction with HTML or XML. DCMI has produced a series of practical guidelines that promote interoperability by specifying how metadata properties should be encoded into documents of these formats [33, 24].

The most fundamental concepts in the metadata model of Dublin Core are elements and qualifiers. The 15 elements, enumerated in Table 1 on page 16, are simple metadata properties, bindings between established identifiers, element names, and values based on the described resource [17]. As some of the elements, such as "date", are specified in rather vague fashion, additional qualifiers may be used to further pinpoint the scope of an element. Two kinds of qualifiers exist: element refinements and encoding schemes. An element refinement, such as "created", shares the overall meaning of its parent element, in this case "date", and narrows it to a more specific context: the date of creation of the resource [13]. An encoding scheme, on the other hand, describes how the value of an element shall be interpreted [17]. The metadata description of a resource may contain zero or more instances of a specific element and an element may or may not have an explicit encoding scheme. The diagram in Figure 5 on page 16 summarizes the mutual relationships between the discussed concepts.

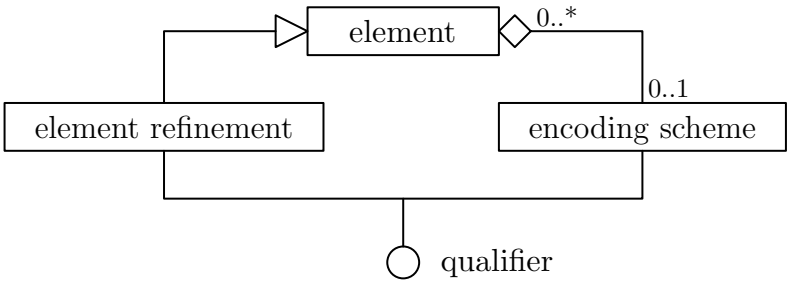

Figure 5: The core concepts of the Dublin Core metadata model.

Table 1: The elements of the Dublin Core metadata model [13].

Name	Description
contributor	An entity that is contributing to the content of the resource
coverage	The extent or scope of the resource
creator	The primary responsible for the content of the resource
date	A date in the life-cycle of the resource
description	A human-readable description of the content of the resource
format	A description of the physical or digital form of the resource
identifier	An unambiguous reference to the resource
language	The language used for the content of the resource
publisher	The entity that is making the resource available
relation	A reference to a related resource
rights	A description of the various rights held over the resource
source	A reference to a resource from which the resource is derived
subject	The topic of the resource, e.g. a classification code
title	The formal name of the resource
type	The nature or genre of the content of the resource

4 Implementation Overview

This section provides an overview to the software architecture and implementation details of the OtaStat publication system. Roles of the technologies presented in the previous section—Unicode, TEX XML, Dublin Core—are described in relation to the entire system.

The system can be divided roughly into two distinct subsystems: the OtaStat Markup Language (OSML) document type, and the tools for producing HTML documents based on OSML documents. The document type is based on the Modularization of XHTML and is defined in a DTD [2]. The processing tools are written in Bourne Shell, Perl, and XSLT languages. In addition, several external applications are utilized, namely Michael Kay's Saxon XSLT processor, Jan-Åke Larsson's Dvipng program, and ElCel Technology's XML Validator.

4.1 Document Type

The OSML document type is heavily based on XHTML [3]. Majority of the elements and attributes of XHTML are present and new functionality is provided by the Dublin Core module and the three OtaStat modules. Table 2 on page 20 contains a listing of all the modules. Of the conformance levels specified for the Modularization of XHTML, the OSML document type fulfills the criteria of XHTML Integration Set, and the Dublin Core module follows the guidelines set forth by the DCMI [2, 24]. While the OtaStat modules do not conform to any particular standards, they try to follow the existing conventions. As required, the new modules do not use the XHTML namespace: the Dublin Core module operates in the official namespace, and the OtaStat modules utilize their own namespace, shown in Figure 6 [13].

```
http://www.otastat.hut.fi/osml
```

Figure 6: The OtaStat namespace.

The OSML Structure Module defines three elements: `osml`, `head`, and `body`. These elements are used similarly than their counterparts in XHTML [3]. The contents of `head` element are defined by the Dublin Core Module. Table 3 on page 20 contains a list of supported Dublin Core elements and their qualifiers. Only a "title" element is mandatory. As a convention, `dc` is used as the prefix for the Dublin Core namespace and `o` for the OtaStat namespace, and the XHTML namespace is used as the default namespace of an OSML document. An author can, however, redefine the usage of prefixes for the purposes of a single document [2].

The OSML Math Module provides two additional elements, `dm` and `im`, that are used to embed equations and other mathematical entities to documents. The `dm` element is used for *display mathematics*, primarily equations and other statements are emphasized. In contrast, the `im` element, for *inline mathematics*, is meant for mathematical symbols and other entities enclosed in the running text. The content of both elements shall follow the LaTeX mathematics syntax.

The remaining new module, the OSML Hypertext Module, adds one more element to the vocabulary: `link`. It allows one to create hyperlinks between documents in the same publication. The title of the target document is used as a reference. Functionality of the `link` element as well as the `requires` and `references` elements in the Dublin Core Metadata Module are based on the concept of a *document tree*. The document tree is a simple XML document with two roles: it defines the hierarchic order of the documents in the publication and acts as an associative table that matches a document

title to the document's physical location, URL. The document tree is used also for automatic generation of navigational elements on the published Web pages. The rationale behind the associative table is an effort to minimize *link rot*, increasing number of broken hyperlinks in the course of time [29].

4.2 Page Processing

A Web page suitable for publishing is produced by automated processing of an OSML document. The process can be divided into three subsequent phases: validation of the source document, an XSLT transform from the OSML document to an XHTML document, and generation of images based on the mathematics elements.

4.2.1 XSLT Transformation

The XSLT transformation produces a valid XHTML document based on an OSML document. As the OSML document type is based on the XHTML language, many elements and attributes from the source document will fall through the transformation unchanged. The primary responsibilities of the transformation process are as follows:

1. To construct the overall structure of the Web page and mark up element identifiers and Cascading Style Sheets (CSS) class names for styling [7]

2. To generate content, such as figure numbers and navigational elements, in the correct locale

3. To create HTML hyperlinks from references to other documents

The transformation is defined as a modular XSLT stylesheet comprising of multiple small stylesheets. For all the three OtaStat modules and the Dublin

Table 2: The modules of the OSML document type.

Name	Description
OSML Structure Module	Document structure
Dublin Core Metadata Module	Document metadata
XHTML Text Module	Text structure
XHTML Presentation Module	Text presentation
XHTML Hypertext Module	Generic hypertext
OSML Hypertext Module	OSML hypertext
OSML Math Module	Mathematics
XHTML List Module	Lists
XHTML Table Module	Tables
XHTML Image Module	Images
XHTML Object Module	Embedded objects

Table 3: The Dublin Core elements used in OSML.

Name	Refines	Encoding	Notes
contributor			
created	date	ISO 8601	
creator			
description			
modified	date	ISO 8601	Updated automatically
publisher			
requires	relation		Relation to an OSML document
references	relation		
subject			A list of keywords
title			Mandatory

Core module there exist individual stylesheets containing the transformation logic. All the XHTML modules are handled by a single stylesheet and almost by one simple template that is applied to most elements in the XHTML namespace.

The numbering of figures and other such elements is achieved with the `number` instruction of the XSLT language [11]. Generation of static content, such as headers and footers, is based on *result tree fragments*, portions of XHTML pages stored in individual files that are accessed from the stylesheets with the `document()` XPath function and copied to the result tree with the `copy-of` XSLT instruction [11, 12]. With the exception of numbering, content generation is based on the locale of the source document. As a result, the ISO 8601 date 2005-07-17 shows up as "July 17, 2005" in an English document and "17. heinäkuuta 2005" in a Finnish document, for example.

Hyperlink construction based on references, such as the "requires" and "references" Dublin Core elements and the `link` element of the OSML Hypertext Module, utilizes the `document()` XPath function to gain access to the document tree from the stylesheets. The document tree is used also for automatic generation of hyperlinks to the neighboring documents in the hierarchy.

4.2.2 Image Generation

In the image generation phase the still remaining `dm` and `im` elements are removed from the produced XHTML document and replaced with `img` elements of the XHTML Image Module [3]. The replacement is implemented using a custom Perl script, LaTeX, and the Dvipng program. The Perl script scans the document for instances of the aforementioned mathematics elements, stores their content to a temporary LaTeX document containing one

mathematics entity per page, and runs LaTeX on the temporary document. After that, it uses the Dvipng program to produce a set of Graphics Interchange Format (GIF) image files corresponding to the mathematics entities of the document. As an optimization, each publication directory has a hash table containing the fingerprints of all generated images. The script does a lookup on each new image's fingerprint and if the lookup fails, the image is stored and the table updated. Otherwise the existing identical image is used instead. Finally the `dm` and `im` elements are replaced with references to the generated images.

To improve the accessibility of the produced Web pages, the image generation process inserts alternate text descriptions—`alt` attributes—to all generated images [10]. These descriptions are based the LaTeX code with some simple text transformations to enhance readability.

4.3 Customization

The appearance of the generated XHTML pages can be easily customized. The most obvious way to customize the pages is to set the options in the site wide configuration file, these options are shown in Table 4 on the next page. More profound customization is possible by modifying the CSS stylesheets [7]. The XSLT stylesheets are designed so that practically all layout elements have identifiers, and referencing these identifiers in CSS stylesheets the entire layout can be redesigned without inflicting changes to the XSLT stylesheets or the generated XHTML documents. Constant layout elements, such as headers and footers, can be customized simply by modifying or replacing the result tree fragments. Naturally, as a last resort, the XSLT stylesheets can be modified as well.

Table 4: Options in the site wide configuration file.

Option	Description
document-language	The default language for documents
document-tree	URL of the document tree file
localization-path	URL of the directory for the localization files
fragment-path	URL of the directory for the result tree fragments
master-stylesheet	URL of the CSS stylesheet
site-top	URL of the front page of the entire site
site-title	The title of the entire site
site-publisher	The default publisher for documents

5 Conclusion

This paper has introduced the OtaStat publication system, a lightweight content management system for mathematical documents in the World Wide Web. The underlying technologies have been presented as well as the overall architecture of the implementation.

During the design process of the system, the focus was on strong foundation, flexibility, and extensibility. The decision to base the architecture on existing standards, such as Unicode and XML, as well as established best industry practices, such as TEX, promotes the first goal. Both flexibility and extensibility were attained by the approach of a modular and open design based on small and interchangeable building blocks. As a result, the system, while originally planned for the purposes of the OtaStat project, is generic at its core and could benefit also other applications with similar requirements.

5.1 Further Research

An interesting further research topic would be to investigate the developing state of MathML implementations in Web browsers and whether a move from TEX-renditions of mathematical entities to native MathML markup on the HTML pages would be beneficial. The LATEX syntax could be preserved in the OSML documents with the help of an automatic conversion tool, such as Jacques Distler's ITEX2MML, that would produce MathML fragments from LATEX input. While, for example, Mozilla Firefox browser does already support MathML quite well, one remaining problem is the lack of suitable free math fonts. This limitation will, however, soon disappear as the STIX Fonts project by Scientific and Technical Information Exchange, a group of scientific publishers including the American Mathematical Society (AMS),

Institute of Electrical and Electronic Engineers (IEEE), and Elsevier Science, finishes [35]. The goal of the project is to produce free, comprehensive set of high quality glyphs for the needs of mathematical and scientific publishing.

References

[1] Adobe Systems. *PDF Reference 1.6*, 5th edition, 2004.

[2] Murray Altheim et al. *Modularization of XHTML*. World Wide Web Consortium, 2001.

[3] Murray Altheim and Shane McCarron. *XHTML 1.1 – Module-based XHTML*. World Wide Web Consortium, 2001.

[4] Mark Baker et al. *XHTML Basic*. World Wide Web Consortium, 2000.

[5] Tim Berners-Lee, Roy Fielding, and Larry Masinter. *Request for Comments 3986 – Uniform Resource Identifier (URI): Generic Syntax*. Internet Engineering Task Force, 2005.

[6] Jim Bigelow. *XHTML-Print*. World Wide Web Consortium, 2004.

[7] Bert Bos, Tantek Çelik, Ian Hickson, and Håkon Wium Lie. *CSS 2.1 Specification*. World Wide Web Consortium, 2004.

[8] Tim Bray, Dave Hollander, and Andrew Layman. *Namespaces in XML*. World Wide Web Consortium, 1999.

[9] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, and François Yergeau. *Extensible Markup Language (XML) 1.0*. World Wide Web Consortium, 3rd edition, 2004.

[10] Wendy Chisholm, Gregg Vanderheiden, and Ian Jacobs. *Web Content Accessibility Guidelines 1.0*. World Wide Web Consortium, 1999.

[11] James Clark. *XSL Transformations (XSLT) Version 1.0*. World Wide Web Consortium, 1999.

[12] James Clark and Steve DeRose. *XML Path Language (XPath) Version 1.0*. World Wide Web Consortium, 1999.

[13] DCMI Usage Board. *Guidelines for Implementing Dublin Core in XML*, `http://www.dublincore.org/documents/dc-xml-guidelines/` (Cited July 5, 2005).

[14] Victor Eijkhout. *TEX by Topic*. Addison–Wesley, 1993.

[15] Charles F. Goldfarb. *The Roots of SGML—A Personal Recollection*, `http://www.sgmlsource.com/history/roots.htm` (Cited July 17, 2005).

[16] James Gosling et al. *The Java Language Specification*. Addison–Wesley, 2nd edition, 2000.

[17] Diane Hillmann. *Using Dublin Core*, `http://www.dublincore.org/documents/usageguide/` (Cited July 5, 2005).

[18] Dennis Howe et al. *The Free Online Dictionary of Computing*, `http://www.foldoc.org/` (Cited July 5, 2005).

[19] International Organization for Standardization. *ISO 8879:1986/Cor 1:1996 – Information technology – Document description and processing languages – Standard generalized markup language*, 1987.

[20] International Organization for Standardization. *ISO/IEC 10646:2003 – Information Technology – Universal Multiple-Octet Coded Character Set (UCS)*, 2003.

[21] International Organization for Standardization. *ISO/IEC 15836:2003 – Information and documentation – The Dublin Core metadata element set*, 2003.

[22] International Organization for Standardization. *ISO/IEC 8859-1:2003 – Information Technology – 8-bit single-byte coded graphic character sets – Part 7: Latin/Greek alphabet*, 2003.

[23] Patrick Ion and Robert Miner. *Mathematical Markup Language (MathML) 1.0 Specification*. World Wide Web Consortium, 1998.

[24] Pete Johnston and Andy Powell. *Guidelines for implementing DC in XML*. Dublin Core Metadata Initiative, 2003.

[25] Donald E. Knuth. *The TEXbook*. Addison–Wesley, 1984.

[26] Donald E. Knuth. The Future of TEX and METAFONT. *TUGboat*, 11(4), 1990.

[27] Leslie Lamport. *LATEX: A Document Preparation System*. Addison–Wesley, 2nd edition, 1994.

[28] Arnaud Le Hors et al. *Document Object Model (DOM) Level 3 Core Specification*. World Wide Web Consortium, 2004.

[29] Jakob Nielsen. Fighting Linkrot. *Jakob Nielsen's Alertbox*, 1998.

[30] Jakob Nielsen. Avoid PDF for On-Screen Reading. *Jakob Nielsen's Alertbox*, 2001.

[31] Jakob Nielsen. PDF: Unfit for Human Consumption. *Jakob Nielsen's Alertbox*, 2003.

[32] Steven Pemberton et al. *XHTML 1.0 – The Extensible Hypertext Markup Language*. World Wide Web Consortium, 2nd edition, 2002.

[33] Andy Powell. *Expressing Dublin Core in HTML/XHTML meta and link elements*. Dublin Core Metadata Initiative, 2003.

[34] Dave Raggett, Arnaud le Horns, and Ian Jacobs. *HTML 4.01 Specification*. World Wide Web Consortium, 1999.

[35] *STIX Fonts*. Scientific and Technical Information Exchange. `http://www.stixfonts.org/` (Cited July 16, 2005).

[36] Hàn Thé Thành. *Micro-typographic extensions to the TEX typesetting system*. PhD thesis, Masaryk University in Brno, the Czech Republic, 2000.

[37] The Unicode Consortium. *The Unicode Standard, Version 4.0*. Addison–Wesley, 2003.

The journal Studies in Applied Mathematics is published by Wileyâ€“Blackwell on behalf of the Massachusetts Institute of Technology. It features scholarly articles on mathematical applications in allied fields, notably computer science, mechanics, astrophysics, geophysics, and high-energy physics. Its pedigree came from the Journal of Mathematics and Physics which was founded by the MIT Mathematics Department in 1920. The Journal changed to its present name in 1969. Mathematical Biology & Bioinformatics * Concerns the fields of Combinatorics, Probablity and Statistics * Examples of research interests: detection of functional signals in biological sequences & protein networks. Algorithm ans models for inferen...Â With no prior experience, Kyle Dennis decided to invest in stocks. He owes his success to 1 strategy. Additional: This course Mat-2.108 Independent Research Projects in Applied Mathematics. Language: Finnish replaces the course Mat-2.4113 Queuing Theory (3-6 cr) P Not lectured this academic year. Lecturer: N.N. Contents: Queuing problems as stochastic processes, the problems including finite or infinite set of users and servers, queuing disciplines, queues as Markov processes.Â Prerequisites: Mat-2.2105 or Mat-2.3140 Additional: This course replaces the course Mat-2.174 Programming of Mathematical Algorithms. Language: Finnish Mat-2.4177 Seminar on Case Studies in Operation Research (5 cr) P V Spring Lecturer: Prof Ahti Salo Contents: Teamwork will be practised in the context of techno-economic projects.