A simple management tool for medium-sized Web sites

Igor Fischer[1] and Andreas Zell[1]

Wilhelm-Schickard-Institut für Informatik, Universität Tübingen,
Köstlinstr. 6, 72074 Tübingen, Germany
{fischer, zell}@informatik.uni-tuebingen.de
http://www-ra.informatik.uni-tuebingen.de/

Abstract. Site management is one of the key issues in Web publishing. Whereas smaller sites can be still managed manually, large Web sites generally require a systematic approach, usually involving sophisticated tools operated by skilled personnel. Between these two extremes are medium-sized sites. Although site management tools exist for both extremes, the market coverage of the mid-size segment is somewhat sparse. In this paper we describe our own tool, aimed at small and medium sized Web sites, of some 1000 pages. It is simple to use and needs no special skills or knowledge to be operated. Yet it is powerful enough to manage a site of a small organization and to enforce a common corporate identity. We use it for management of our department site, as well as for managing contents of various on-line education projects.

1 Introduction

The continuous and exponential growth of the WWW has been marked not only by an increasing number of Web sites, but also by increasing size and complexity of the sites themselves. Today, it is not unusual for a Web site to contain tens, or even hundreds of thousands of documents linked with each other. For example, the AltaVista search engine lists over 20000 documents in the domain java.sun.com alone, and there are probably many more[1]. Moreover, large sites can even span many servers distributed around the world. Since sites are non-static entities, with pages continuously being added, removed, changed or linked in a different way, their management has increasingly become an issue.

The term "Web site management" is used to describe a wide variety of activities related to publishing contents on the WWW, ranging from authoring, over maintaining consistent structure, to traffic analysis and site optimization. For the purpose of this paper, by managing a Web site we shall mean a subset of those activities, namely keeping site structure under control, as well as ensuring common appearance of corresponding pages, regardless of their contents.

[1] Experience shows us that, as a rule, search engines do not index all pages on the WWW. For example, our department site has some 200 pages, but only 16 appear in the AltaVista listing.

Under structure we understand the way pages are linked to each other and how they interact with server-side modules, like servlets or CGI scripts. More precisely, the site structure describes which pages comprise the site and how they relate to each other. Although the structure of a web site can generally be any kind of a directed graph, for navigational simplicity tree-like structures prevail. Consequently, most relationships are of hierarchical (like super-/subordinated, or parent/child) or peer level (predecessor or successor) nature, with few exceptions, most notably the home page, which is in many sites linked to from all pages.

By common appearance we mean coherent use of fonts, colors, graphics, logotypes, navigational elements and layout in all pages.

Although smaller sites of, say, few dozen Web pages still can be managed manually, this approach relies heavily on the human factor and is therefore error-prone. For large sites, with many thousands of pages, management has to be machine-supported in some way. Different tools have been developed to cope with the challenge, ranging from simple visual tools, like Microsoft FrontPage, for managing smaller sites, to big database systems with differentiated authoring, access and version control, capable of dynamically generating personalized contents on the high end.

In the former approach, a web site manager uses a visual site editor for determining the site structure. This approach is intuitive and simple as long as the display remains comprehensible, but for large sites, the screen easily becomes confusing and jammed with pages and links.

The latter approach does not rely on a visual representation of site, although it may still include it as a convenience. In that approach, content and site management are specialized tasks, being generally performed by different people.

The rest of this document is structured as follows: We first take a look at some existing site management tools, then we state the scope and target audience of our tool, analyze the tool structure and show subsequently some example sites managed by the tool. In conclusion we summarize the results and discuss possibilities for further development.

2 Related tools

Rapid growth of the WWW has been followed by development of various Web-related tools and applications, from rather specialized ones, like link checkers, to comprehensive site management and analysis applications. Analyzing all of them would be beyond the scope of this article, but we shall take a brief look at some of their representants.

One very popular site management tool for smaller sites is Microsoft FrontPage [1], available for Windows platforms. Developed as a complete solution for Web publishing, it concentrates mainly on page design, but also includes visual site management. Former versions relied on an integrated Web server and generated proprietary code, but in the "2000" version those issues are improved.

For the design of sites with common look-and-feel, FrontPage provides modifiable page templates. The tool is itself simple to use, especially for Microsoft Office-experienced users, but, due to predefined templates, limits the designer's freedom. A mightier tool, aimed at professional programmers, is Visual InterDev [2].

Another popular tool is NetObjects Fusion [3], designed for small business. Like FrontPage, Fusion is a visual tool with focus on desktop-publishing-like page design, but also allows drag-and-drop design of sites. It encapsulates the site in a proprietary database format, thus disallowing any manual modifications by the author. It is reportedly very good for designing pages with pixel-level precision, but has problems with importing existing sites [4]. Fusion is available for Windows and Macintosh platforms.

Symantec Visual Page [5] for Windows and Macintosh is primary a Web authoring tool, but also offers some basic site management functions. A site is stored in a project file, but it does not describe any linear or hierarchical structure. Therefore, no automatic generation of navigation elements is provided. Instead, all links have to be entered manually.

A tool more oriented on content management than on page design is the UserLand Frontier [6], specifically aimed at news-oriented sites. Common appearance of pages is ensured through use of templates, and linear navigation is simple to implement, but not site maps, table of contents etc. All relevant data (contents, templates and site structure) is stored in an object-oriented database in a proprietary format. Contrary to previously mentioned tools, a visual representation of site structure is not provided. Frontier is available for Windows and Macintosh platforms.

The SGI Site Manager [7] is capable of representing the complete site structure in three-dimensional hyperbolic space. This tool is more sophisticated, consisting of a client and a server part, and should be configured by a Webmaster or a user who has experience with server configuration. Besides for content management, this tool provides for site traffic analysis. Common appearance of pages is provided through templates, which can be created not only for HTML, but for any file format. It does not generate navigation elements automatically, and is available for IRIX 6.2 and higher platforms.

On the high end are site management tools like Interwoven TeamSite [8]. Based on a hybrid architecture (file system and database), this tool is suitable both for existing and legacy sites, as well as for well structured state-of-the-art sites, with templates, XML, dynamic contents generation and many more. It is available for Windows, Mac and Solaris platforms, but its price, starting at USD 70000, places is out of reach for smaller organizations.

3 Scope

Site management tools developed up to date mainly concentrate on large Web sites, usually involved in e-business. The coverage of small and medium sites, consisting of roughly 1000 pages and needing coherent and appealing corporate

identity is notably sparser. This is especially true for non-Windows platforms. Medium-sized sites are already too large to be efficiently managed by simple visual tools. Additionally, many of those tools, most of them with emphasis on Web design, although offering a number of predefined templates, limit the authors freedom in designing his own common appearance of pages. On the other hand, a high-end professional solution is likely to be too expensive to acquire and to support for an organization with a relatively small site.

A typical example of such organization would be a small or mid-size organization or company with rented Web space. The server administration would be generally outsourced to the Internet Services Provider, but the organization is likely to remain in charge for the contents. The following model describes the constellation:

- the organization designs a visual appearance for presentation on the Web, corresponding to its corporate identity, or hires a professional designer for the job,
- it creates the contents to be published on the Web,
- it converts the contents into HTML and shapes it to suit the designed appearance, and
- passes it to the Web-space provider, who publishes it.

The whole publishing process should be preferably performed modularly, in independent steps. The Web designer needs neither contents nor Web access for designing the appearance. He or she designs only templates, which are to be filled with contents. The contents creator, on the other hand, doesn't rely on the design when creating the contents. The two components meet only when melting them together into final HTML files which are to be published. But even then, the resulting site can be viewed off-line, as long as no interactive components, like database queries, are involved. The need for a Web server comes into appearance only in the final step, when the content is published on the Web.

To meet the needs of that group, we developed a Web site management tool that is still relatively easy to use for Web authors, but powerful enough to manage such medium sized sites. In the development of our tool, we were guided by the idea of simplicity. More specifically, we wanted the tool to fulfill the following requirements:

- easy to install,
- simple to use,
- easy to maintain,
- able to maintain legacy HTML pages without further modifications,
- not to use proprietary data formats,
- independent of specialized editors or viewers,
- able to work off-line, independent of Web server

The target group we had in mind were content authors. The only assumption we made about them was possession of some HTML knowledge. Therefore, the tool has to offer maximum possible freedom in developing design and contents and spare the authors of technicalities connected with server administration.

As could be expected, developing such a tool could not be done without trade-offs. Above requirements could not be met without imposing some constraints on the site structure. They were the following:

- the site structure can be considered static,
- page contents are independent of their position in the site structure.

The first constraint simply states that the number of pages remains the same and that links between them don't change. At the first glance it might seem as a very hard constraint, since it excludes dynamically generated pages, like from a database or search engine. Fortunately, this is not the necessarily the case: if only a limited number and kind of pages is generated dynamically, one can reserve proxy pages in the site structure, which acquire their content dynamically. For example, one can reserve a "search" page in the site structure, whose content is then generated according to the search results.

What the constraint really makes impossible is management of dynamic sites, like personalized sites, where users practically create their personal view of the site, or where sites automatically adapt to users. Those features are, however, not very common yet and for small and medium sites of limited interest.

The other constraint is even easier to meet. It aims at uncoupling the development of contents from development of site structure. In practice it means that contents authors should not reference other pages relatively to the current one (like "see previous page"), but only by their file name (like "see page *introduction.html*"). For hyperlinked medium like WWW it is anyway the preferred way, because the reference can be implemented as a hyperlink to the referenced page. Those hyperlinks can still be relative inside the directory tree containing the pages. The directory structure does not have to correspond to the site structure, but site maintenance might be somewhat simplified if it does.

4 Tool structure

The core of our tool is a Perl script which takes site description, page templates and contents pages as input and produces complete linked site as output (figure 1). Instead of using a database or some proprietary format for storing the site structure and its contents, we basically rely on HTML. This makes the tool extremely easy to use for a HTML author, because HTML can easily be edited, even with pure text editors. So, all inputs to the processing tool are HTML-based, with some extensions that control the processing and the output is strictly HTML — at least to the degree the author of contents and templates adhered to the standard.

The notorious HTML property of mixing contents and structure comes helpful in our case. We use HTML in description file to describe the site structure. In templates, we mix the structure with contents to achieve a page appearance that is dependent on the page position in the site structure. The tool imposes no restriction on the way authors write contents pages, except that it expects them to be in HTML.

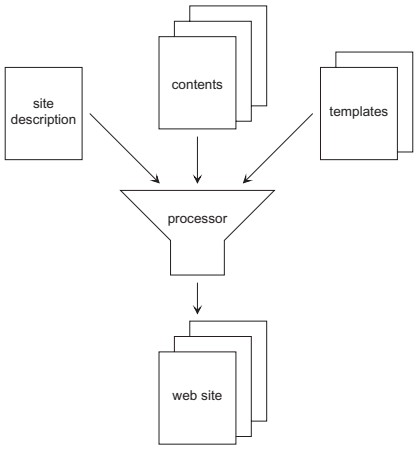

Fig. 1. Schematic tool structure.

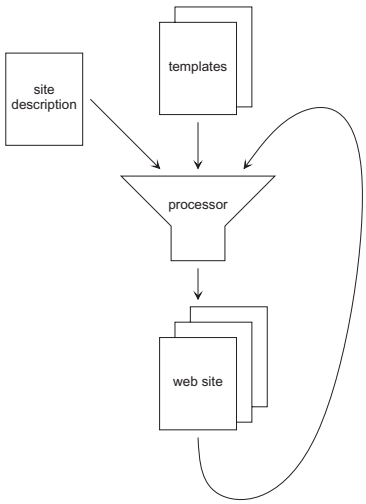

Fig. 2. Schematic tool structure for "site recycling".

The description file describes site structure in a similar way the Netscape bookmark file organizes user's bookmarks. The logical organization of a site is hierarchical, in a tree structure. The tree consists of nodes and branches, branches representing hierarchical relationships between nodes. Nodes are generally named, and each one *can* have a page assigned to it. This is a difference to most other site management tools, which *require* a page to be assigned to each node. By allowing nodes without assigned pages, we provide a way for jumping directly to lower-level pages, without having to stop at intermediary pages of no interest, but still maintaining a consistent site structure. This concept resembles the idea of abstract classes in object-oriented programming, which have no functionality but to serve as basis for child classes.

The tree structure in the description file is represented by nested HTML *definition lists* (`<DL>`), with each node being a *definition term* (`<DT>`). If a node has a page assigned to it, the page is represented by a HTML link to it (`<A HREF="...">`).

To facilitate the processing, we have included some proprietary extensions to standard HTML. All extensions are optional and influence in no way standard HTML parsing of the file. At the beginning of the file, a block with system and user variables can be included. This block is a HTML comment, starting with `<!--VARIABLES` and ending, like all HTML comments, with `-->`. The system variables control the processing: they specify where to find contents files, where to put the generated files, which templates to use and so on. An example is given in table 1 in the next section.

Additionally to them, a user can define his own variables it the `VARIABLES` block, like copyright note, author name and so on. These variables, if given in template files, get substituted for their values during processing.

After the `VARIABLES` block, the site structure, as nested definition lists, follows. The proprietary extensions to the `<DT>` tag, analogous to the Netscape extensions used in bookmarks file, appear after `DT` and before closing angle bracket. They are used to force use of another template, to implicitly specify a sequence of source files to be inserted at that point etc. Table 2 describes some example extensions.

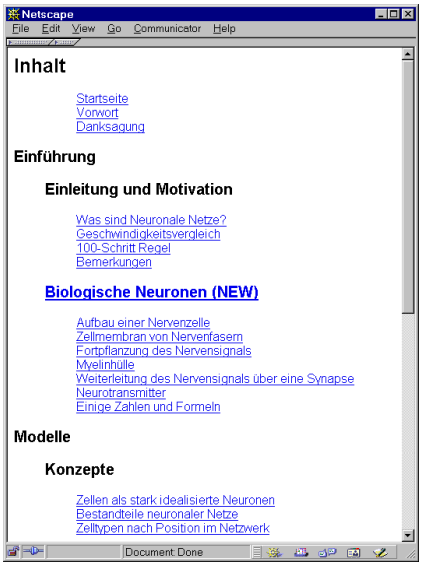

Fig. 3. Site structure, viewed with a HTML browser.

Site structure can be viewed with an ordinary HTML browser, as shown in figure 3. Both hierarchical site structure, as well as linear order of pages, is easy to understand.

The template files can be seen as wrappers for contents files. They define the appearance of web pages, including navigation bars, buttons and banners. They are basically HTML pages with processing directives, specifiers and variables that during processing get substituted for currently applicable values. All these HTML extensions appear as HTML comments, i.e. surrounded by `<!--` and `-->` strings. They represent page title, hypertext references to this and other pages, and so on. For some of these elements, the current value depends only on the page being processed and for others on the site structure and the page position in it. Together, they are used for automatic generation of navigation elements, indices and site maps. Some examples are presented and described in section 5.

It is possible to use a same directory for both contents (input) pages and the resulting web site. In that case, the processing tool does not take whole HTML pages as a source, but extracts the original contents from them for processing. This feature makes it possible to update a complete site – for example, change its design or update links – without having a separate copy of it for publishing. It is very useful for maintaining existing sites, since one can make changes directly on its pages, without the need for a separate publishing step. The tool structure that corresponds to this concept is shown in figure 2.

Design of template files should preferably be done by a HTML-experienced designer. The basic layout can still be designed with a visual tool, and then manually extended and composed together to build a suitable template file. As far as contents authors are concerned, this tool imposes no obstacles, but allows them to concentrate on contents, without worrying about appearance.

5 Example sites

The site management tool described here has been originally developed in the *VirtuGrade* project, for the management of educationally oriented Web sites. Although it is still its main application area, it can also be used for general site management. For example, we use it for the management of our department web site.

```
<!--VARIABLES
  $FILTER *.html
  $COPY_FILES logo/*.jpg logo/*.gif
  $TEMPLATE TemplateStart.html
  $SOURCE source
  $DEST dest
  $PAGE a
-->

<H2>Inhalt</H2>
<DL>
  <DL>
    <DL>
      <DT><A HREF="index.html">Startseite</A>
      <DT TEMPLATE="Template.html"><A HREF="book.00.html">Vorwort</A>
      <DT><A HREF="book.01.html">Danksagung</A>
    </DL>
  </DL>
  <DT PAGE=1><H3>Einf&uuml;hrung</H3>
  <DL>
    <DT><H3>Einleitung und Motivation</H3>
    <DL>
      <DT TEMPLATE="TemplateChapter1.html"><A HREF="chapter1/book.1.html">Was sind Neuronale Netze?</A>
      <DT><A HREF="chapter1/book.2.html">Geschwindigkeitsvergleich</A>
      <DT><A HREF="chapter1/book.3.html">100-Schritt Regel</A>
      <DT END><A HREF="chapter1/book.7.html">Bemerkungen</A>
    </DL>
```

Fig. 4. An excerpt from a site description file.

Figure 4 shows the same site description file as figure 3, but this time in source code. The `VARIABLES` block defines which files are to be processed, which simply copied, with which page number and template to start and which directories to use. The meaning of all these variables is described in table 1.

Variable	Meaning
`$SOURCE`	The directory containing source pages. The default value is `source`.
`$DEST`	The directory for processed pages. The default is `dest`. If source and destination are the same, the processing tool extracts original contents from the pages and reprocesses them.
`$FILTER`	A file filter (wildcards for listing files), defining which files other than those explicitly listed are to be processed. By default no other files are processed.
`$COPY_FILES`	Files to be automatically copied from source into destination, like images, Java classes etc. May include wildcards. Default: none.
`$TEMPLATE`	Template file to use for the pages. By default `Template.html`.
`$PAGE`	Starting value of the page counter. Default value is 1, but it can be set to any number, or to a letter, for alphabetical enumeration.

Table 1. Control keywords for `VARIBALES` block in site description file

Further, the site structure is stated: the `<DT>` tags define nodes in the hierarchy given by nested `<DL>` tags. Some of these tags are used with extensions, whose meanings are described in table 2.

The files *TemplateStart.html* and *Template.html* are used as templates during processing: it starts with *TemplateStart.html* but switches later to *Template.html*. Figure 5 shows a small part of a template file which governs the appearance of the navigation bar. In this example, the whole navigation bar is packed in a table. The `<!--INDEX-->` tag in the second line starts the navigation bar description. The `<!--L1 TYPE="current"-->` defines first-level entries for index entries belonging to the same branch as current page: the name of the branch is displayed (`<!--TITLE-->` tag) and a new table started. Inside the table, the appearance of second-level entries is defined. The current entry appears with an icon left to it (`<IMG ...>` tag) and its name is shown in a different color. Other entries (`<!--L2 TYPE="other"-->`) are shown without an icon and in default color.

`<DT>` Extension	Meaning
`END`	Used in conjunction with `$FILTER` from the `VARIABLES` block. All files selected by the filter and which, alphabetically sorted, appear between currently processed file and the file given in the `END` variable, get processed and subsequently inserted at the current position in the site structure.
`TEMPLATE="..."`	Switches to another template file for processing current and subsequent source files.
`PAGE=...`	Sets the page counter to the given value.

Table 2. Extensions to the `<DT>` tag

```
<TD VALIGN="TOP" WIDTH=128 BGCOLOR="#99ccff">
  <FONT SIZE=-1><B><!--INDEX-->
  <!--L1 TYPE="current"--><P><!--TITLE--><BR>
  <TABLE CELLSPACING=2 CELLPADDING=0 BORDER=0>
    <!--L2 TYPE="current"-->
    <TR><TD VALIGN="TOP"><IMG src="logo/tag2.gif" WIDTH=8 HEIGHT=12></TD><TD><FONT COLOR="#003366" SIZE=-1
    ><!--TITLE--></TD></TR></TR>
    <!--/L2-->
    <!--L2 TYPE="other"-->
    <TR><TD></TD><TD><SMALL><A HREF="<!--HREF-->"><!--TITLE--></A></TD></TR>
    <!--/L2-->
  </TABLE>
  <BR>
  <!--/L1-->
  <!--L1 TYPE="other"--><A HREF="<!--HREF-->"><!--TITLE--></A><BR>
  <!--/L1-->
  <!--/INDEX-->
  <FONT SIZE=1><BR> </FONT>
  </B></FONT>
</TD>

<TD WIDTH=8> </TD>
<TD COLSPAN=3 VALIGN=TOP>
<!--BODY CONTENT-->
  <P> </P>
```

Fig. 5. An excerpt from a template file.

After the closing tag for current branch first-level entries (`<!--/L1-->`), the appearance of other first-level entries is defined. They simply get their name shown, but their subordinated branches are not shown. The `<!--/INDEX-->` tag closes the definition of the navigation bar. Finally, the body of the source file is inserted in place of the `<!--BODY CONTENT-->` tag. Table 3 summarizes the HTML extensions used in template files.

In place of `HEAD CONTENT` and `BODY CONTENT` in a template file, the HTML code from `<HEAD>` and `<BODY>` parts of source files are inserted. Further, extensions like `PREV` and `UP` are available for generating linear navigation in the site.

The `<INDEX>` block is used for generating flexible and powerful navigation bars. For example, if current page or branch should be highlighted in the navigation bar, then the HTML code inside the Ln block marked as "CURRENT" should differ from the code in the block marked as "OTHER". Also, if different levels are to appear in different colors in the navigation bar, the code in different Ln tags has to control it.

A page from a toy example site, generated using the above described template and site description file, is shown in figure 6. This page has a standard banner at the top, with buttons for linear navigation on its right edge. The left side of the page contains a navigation bar, with the current page being marked with a small quadratic icon. At the bottom, the page contains a logo, some textual information about the page and the author and again buttons for linear navigation. All these elements are defined in the template file. The information content of the page, as defined in a source file, appears in the large right part of the screen.

As a real world example of sites managed by the tool, a page from our department's Web site [9] is shown in figure 7. Like the toy example above, it also contains the top banner with some information and navigation elements, and a navigation bar on the left, with highlighted current page. This is today a very frequent appearance on the WWW and can almost be considered a standard.

Extension	Meaning
HEAD CONTENT	The contents of current source file's <HEAD> block.
BODY CONTENT	The contents of current source file's <BODY> block.
DATE	Current date, useful as the last modification date.
PREV	File name of a previous page in the site structure. If no LEVEL is specified, it refers to the previous page regardless of hierarchy level. Else, it referes the previous page at the given hierarchical level.
NEXT	File name of the next page in the site structure. Considerations similar to PREV hold.
LEVEL	Level specifier for PREV and NEXT.
UP	File name of the parent page in the site structure, i.e. one hierarchy level higher.
DOWN	File name of the next page in the hierarchically higher level, i.e. the next branch in the hierarchy.
TITLE	This variable can have different meanings, depending on where it appears: – inside the <!--INDEX--> ... <!--/INDEX--> block, it is the name of the index point – inside the <TITLE> ... </TITLE>, it is the title of the currently processed page
HREF	This variable also has different meanings, depending on where it appears: – inside the <!--INDEX--> ... <!--/INDEX--> block, it is the relative URL of the page assigned to the index point – outside the <!--INDEX--> ... <!--/INDEX--> block, it is the URL of the current page
PAGE	Current value of page counter (page number).
INDEX	Start of an index block. Index block contains complete or partial site index and is useful in navigation bars.
/INDEX	End of an index block.
Ln	Start of n-th level in an index, n being a number between 1 and 6. Additionally, a type should be specified: – TYPE="CURRENT": marks block to be used for processing index points referring to the current page or pages on other hierarchy levels belonging to the same branch. – TYPE="OTHER": opposite of current: marks block to be used for processing index points referring to the pages that are neither current, nor belonging to the same branch on another level. – TYPE="ALL": means CURRENT or OTHER: marks block to be used for processing all index points, regardless if they belong to the same branch or not.
/Ln	End of n-th level in an index.

Table 3. HTML extensions for template files

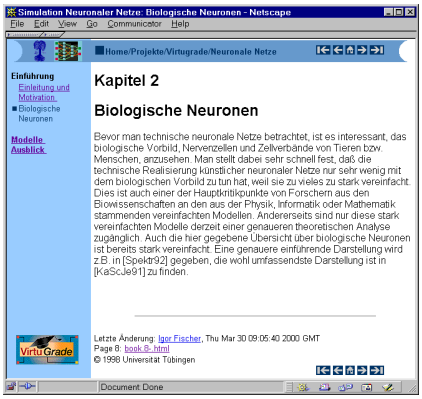

Fig. 6. Example page generated by the site management tool.

The index appears twice in this page: in a single line in the top banner, with only current branch for each level displayed, and in the left navigation bar, with current level expanded and shown with all branches. Contrary to the toy example before, which represents a small Web course book with a preferred order (a learning trajectory) of traversing it, the department site is a more free structure and does not provide for a linear (forward/backward) navigation, but only for a hierarchical one. It also includes a search function, which is accessed on a separate search page through the link "*Suchen*" in the lower part of the navigation bar. The contents of the department site changes almost daily, with different authors being responsible for different pages. Therefore, to facilitate the maintenance, only the published version of the site exists and all changes are made directly on it. Only for the case when pages are added or removed, the webmaster updates the site description file and "recycles" the site, as illustrated on figure 2 above.

6 Discussion

Effective Web site management can be done with relatively simple tools and very limited resources. In this paper, we have described a site management tool implemented in Perl, suitable for managing small and medium-sized sites.

Most commercially available site management tools rely on a database or some proprietary format for storing a site before publishing. This approach is systematical and allows management of very big sites, including dynamic generation of pages or even whole sites. The main drawback lies in its complexity: skilled personnel and maybe expensive hardware and software is needed, which can be behind the reach of smaller organizations.

The other extreme, manual site management, is impractical and can be performed only for very small sites. It is labor-intensive, error-prone and easily be-

Fig. 7. A page from the department for computer architecture's web site.

comes very expensive as site grows. Additionally, it is hard to enforce a common appearance ("corporate identity") on pages.

The tool described in this article aims to be simple enough to be used by anybody familiar with HTML, but still capable of managing sites containing hundreds of pages. It uses HTML for all its inputs (contents, templates and site description) with minor extensions, which are hidden in HTML comments and serve as control elements for page processing. As of software, it needs only a Perl interpreter and works independently of a Web server or any other back-end process, like database or search engine. That makes it a kind of "plug-and-play" program, which is not the case for more complex and sophisticated solutions.

The tool's capabilities have been shown in managing different Web sites, from different department and faculty sites at our university, to various sites in *VirtuGrade* and *Bioinform@tik* projects.

A drawback of the tool is that it can generate only static sites, i.e. sites where structure remains the same for all visitors, regardless when they request a page. Although it can be an important issue in some cases, for the users in our scope (small and mid-sized organizations), it is seldom the case.

It can also be argued that HTML is not the ideal language for describing a site structure, or even contents. Whereas there certainly is something to this argument – otherwise we would not experience continuous improvement of existing standards and establishment of new ones, like XML [10] for general data description, or WebDAV [11] for content management – HTML is for the time being still the most common format on the Internet, simple yet powerful. In authors' opinion, this balance of simplicity and capability has been one of the

crucial reasons for the success of the WWW. Although the tool would perhaps benefit from a better format for site and contents description, it is likely that it would discourage many of its users. To that concern adds the fact that new standards, like XML and XLST, are still under development and sparsely supported by simple and powerful tools. And, even if the time should render the described tool obsolete, we believe that the underlying ideas and principles regarding site management will still remain valid.

7 Acknowledgments

The authors wish to thank Uwe Oestermeier, the originator of the concept for site management, who kindly provided his own tool as a sample. Many thanks go to the authors' colleagues Simon Wiest and Clemens Jürgens, who further extended the described tool and supported the author with new ideas. The *VirtuGrade* project is funded by the state of Baden-Württemberg and the *Bioinform@tik* project by the state of Baden-Württemberg and the Deutsche Telekom AG.

References

1. Microsoft Corporation, *Erste Schritte mit Microsoft FrontPage 98*, 1998.
2. Microsoft Corporation, "Visual interdev web solutions kit," http://msdn.microsoft.com/vinterdev/wsk/default.asp, 2000.
3. NetObjects, *NetObjects Fusion 5.0*, ftp://ftp.netobjects.com/pub/products/documentation/nf5_docs/NOF5manual.pdf, 1999.
4. B. Steppan, "World-Wide-WYSIWYG," *iX*, vol. 6, pp. 40–49, 1999.
5. Symantec Corporation., *Symantec Visual Page User's Guide*, 1998.
6. UserLand Software, "Frontier news," http://frontier.userland.com/news/, 2000.
7. Douglas B. O'Morain, *Site Manager User's Guide*, Silicon Graphics, Inc., http://techpubs.sgi.com/library/manuals/3000/007-3320-002/pdf/007-3320-002.pdf, 1998.
8. Interwoven, Inc., "Interwoven home page," http://www.interwoven.com, 2000.
9. Simon Wiest and Michael Plagge, "Praktikum: Mobile Roboter," http://www-ra.informatik.uni-tuebingen.de/lehre/praktikum.html, 1999.
10. T. Bray, J. Paoli, and C. M. Sperberg-McQuee, *Extensible Markup Language (XML) 1.0*, W^{3}C Consortium, http://www.w3.org/TR/1998/REC-xml-19980210.html, Feb. 1998.
11. David Sussman, "WebDAV: A panacea for collaborative authoring?," *IEEE MultiMedia*, vol. 6, no. 2, pp. 76–79, 1999.