Computer Science

Publishing Content on the Web

Content Management fitting any structure

Isil Ozgener and Thomas Dillig

We propose a method for dynamic content management of large and structurally heterogeneous websites. The content management system (CMS) discussed in this paper allows even non-specialists to easily update webpages and add new ones without knowing anything about the details of the HTML language. The database driven implementation of this CMS adopts a recursive structure that constructs a complex web page using structural units like a layout table to arrange the elementary atomic units of information. This CMS completely separates content from structure and always guarantees well-formed websites, including a well-formed navigation structure. This system is currently used by the Carnegie Institution's Department of Plant Biology as a tool for expanding their project website and for incorporating new experimental data with highly heterogeneous structure.

Many organizations need to provide accurate and up-to-date information on their web pages. While this task is relatively straightforward for micro sites with fewer than 10 pages, large and frequently revised websites benefit greatly from the use of a content management system (CMS) [1]. A CMS allows people to easily enter or update information without knowing anything about the details of HTML or webpage implementation.

Most of the well-known tools ([2], [3] and [4]) for web content management target very large and structurally uniform sites such as online stores. In such websites, every product has certain pieces of information attached to it, such as a price tag, a picture, and a description, making the structure uniform and allowing a single pre-prepared template to fit each product [5]. Many commercial tools are available for updating such structurally uniform websites, but there are no convenient means of managing content when the nature of the information is variegated and when future revisions are unforeseeable.

Problems associated with a lack of CMS for structurally heterogeneous websites arise especially for scientists who need to share newly-acquired experimental data through their websites. For instance, biologists often need to conveniently post downloadable DNA sequences and new microarray data on their website to allow new experimental data to be shared among collaborators and fellow specialists. They also need to incorporate tools for comparative genomics into their websites. In cases like this where the information posted on the web has no homogeneous structure, expert webpage implementers must be consulted before any new information can be shared via the Internet. However, a CMS to handle structurally-heterogeneous content eliminates this need and allows research groups and business organizations to enter new information conveniently and without significant delay.

In this paper, we discuss the implementation of a CMS that works also for structurally heterogeneous websites. This CMS is based on atomic units of information (AUI) nested hierarchically. A webpage is constructed by nesting together many different kinds of AUI's, where each kind corresponds to a unique structural scheme. This technique of using hierarchically-arranged AUI's helps to differentiate between distinct kinds of structural information and allows content to be entered within a strictly hierarchical relationship that makes it almost impossible even for a non-expert to delete crucial information from the website.

Content on One Page

In our implementation of this CMS, we differentiate between five distinct types of AUI that compose a typical webpage: Text, Image, Hyperlink, Downloadable File, and Scripting Module. Text represents plain HTML text with a specified format; Image allows the user to insert image files with an optional caption. Downloadable files are any binary or text data download such as a PDF file. Finally, Scripting Modules are any script (i.e. code) written to perform a special function such as displaying a news ticker.

Any information on one given page can be represented as a collection of these AUI's. However, in order to structure these elements we need to define relationships between them. The relationship between these AUI's is defined via the atomic unit of structure (AUS) within one page. An AUS, which we more informally name a "layout table", groups zero or more AUI's or AUS's together and defines the displayed layout of the information. This grouping, however, is only uni-level, resulting in at most one other object per cell. Graphically, each AUS corresponds to a graph node with one parent and multiple (possibly zero) children, where all children are mapped to fixed screen layout. The exact nature of this mapping needs to be defined for each AUS. For instance, layout tables map each child to its row/column position on the screen (see Figure 1).

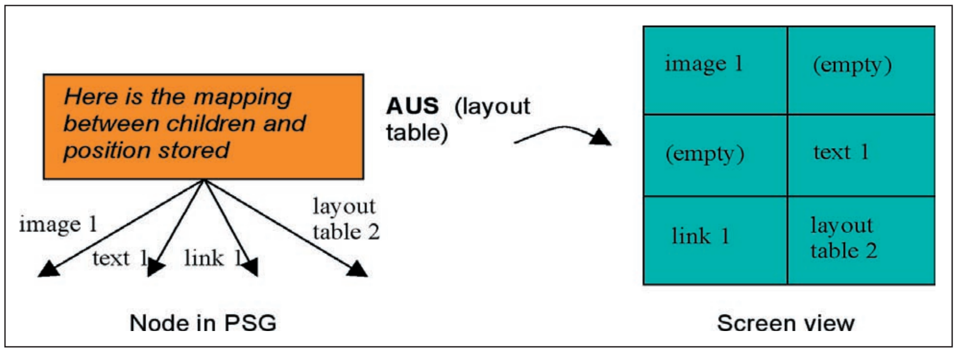


Figure 1: An AUS such as a layout table defines how information is displayed on the webpage, where each piece of information can be an AUI or another AUS.

Therefore, AUS's and AUI's recursively define a Page Structure Graph (PSG), which represents the hierarchical content structure displayed on a webpage. The PSG for any well-formed structure is always guaranteed to be a tree. The root of this tree is a special node containing a list of objects – i.e AUI's or AUS's- , mapped to a simple consecutive screen layout (see Figure 2). Effectively, the root of the PSG is a default AUS that allows only one column and has as many rows as the number of its children. The leaves of a PSG are always AUI's. To add a new object to the PSG, it's sufficient to state its parent to uniquely determine its place in the PSG. The default parent of any newly created object is the root node of the PSG. The implementation of the PSG as a tree structure makes it especially easy to move content within a page from the user's perspective. The user only needs to specify a new parent for the object to reorganize the PSG and hence the displayed layout of the webpage. As discussed later, this approach generalizes across page boundaries. Figure 2 gives an example of a typical PSG. Here, the root contains a list consisting of one image and one layout table. This layout table contains a link and an image as children. The transition between Figure 2a and Figure 2b shows how one can reorganize information within a PSG. For instance, if we want "link 1" to be another child of the root, we simply specify it as a child of the root, which also automatically deletes it from among the children of layout table 1.

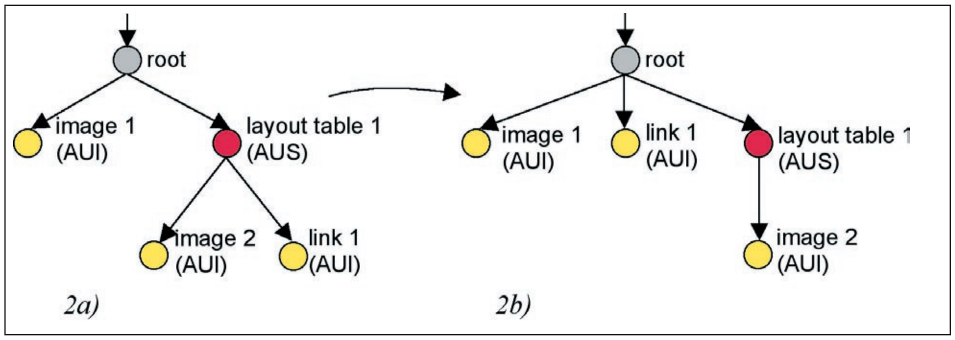


Figure 2: A typical PSG. The transition between Figure 2a and Figure 2b illustrates how the structural layout of information on the webpage can be changed by an adjustment in the PSG.

Structuring Multiple Pages

Just as the PSG describes an ordering of objects resulting in a specific screen layout, the SSG describes an ordering of pages resulting in a specific navigational structure. While the kind of navigational links displayed may vary for each different site, the complete navigation structure is determined by the SSG. Also, the navigation scheme is completely unaffected by any structural changes once that navigation scheme is specified for the website. Because our structure is guaranteed to always preserve meaning, it is impossible to "break links" or distort the structure in any way while updating or adding content.

The SSG is therefore constructed out of page nodes, each page being a PSG graph. Figure 3 illustrates an example of a small SSG. The root node contains some basic information about the page and some pictures and has two child pages, each with its own content. Also, "Page 4" of Figure 3 is not connected to the root and therefore not visible to website visitors. Nonetheless, "Page 4" can be made reachable from the navigational structure by specifying its parent page. Furthermore, one can also add part of the content of "Page 4" to any other PSG to make it "visible" within the navigational structure. Implementing an SSG as an acyclical graph rather than as a tree has some advantages. Since an acyclical graph allows disconnected parts, we can easily temporarily deny access to certain pages by not specifying its parent without having to completely remove the page.

Another advantage of the SSG structure is that we can easily move objects across different web page boundaries rather than just within a single page. Just as a PSG remains valid if we change the ordering in the graph while obeying the afore-mentioned restrictions, the site structure also remains valid when moving objects between pages. Here, for example, we can move "image 3" into the root level of "Page 2" by incorporating it into the PSG that defines "Page 2." The same holds true for moving any piece of content, resulting in a site-wide separation of content, structure, navigation and design that is applicable to any web page.

Implementation

We chose to implement the ideas discussed above using a standard SQL database and readily available free software components. The code base was written in PHP (Version 5), and we chose Apache and MySQL for hosting the data ([6], [7]). Our system is composed of two parts, the display engine and the content modification framework. The display engine receives the page requests sent by website visitors and generates the actual web page. In other words, the display engine queries the SQL database and builds an HTML representation from this retrieved information. The content modification framework allows the users of this CMS to change and update the web content.

The Display Engine

The display engine performs two distinct tasks by retrieving information from the SQL database. First, it queries the site structure graph to generate a navigation bar that is displayed according to a pre-defined template. Our implementation displays a navigation bar on the left side of a web page and displays the parent, children, and siblings of the current page in a hierarchical fashion. In addition, we utilize the same navigational structure information retrieved from the SSG to display a "depth indicator" at the top. The depth indicator shows all the ancestors of the current page, thereby allowing the user to go back to any parent page within the website (see Figure 4).

The second task performed by the display engine is to retrieve the content information of the current page from the PSG. As discussed before, this information retrieved from the PSG defines content as well as structure. We translate the retrieved information into HTML whenever a visitor requests a web page. Default header files and pre-defined style sheets which define default color, spacing, and alignment dictate how information is to be presented on the web page.

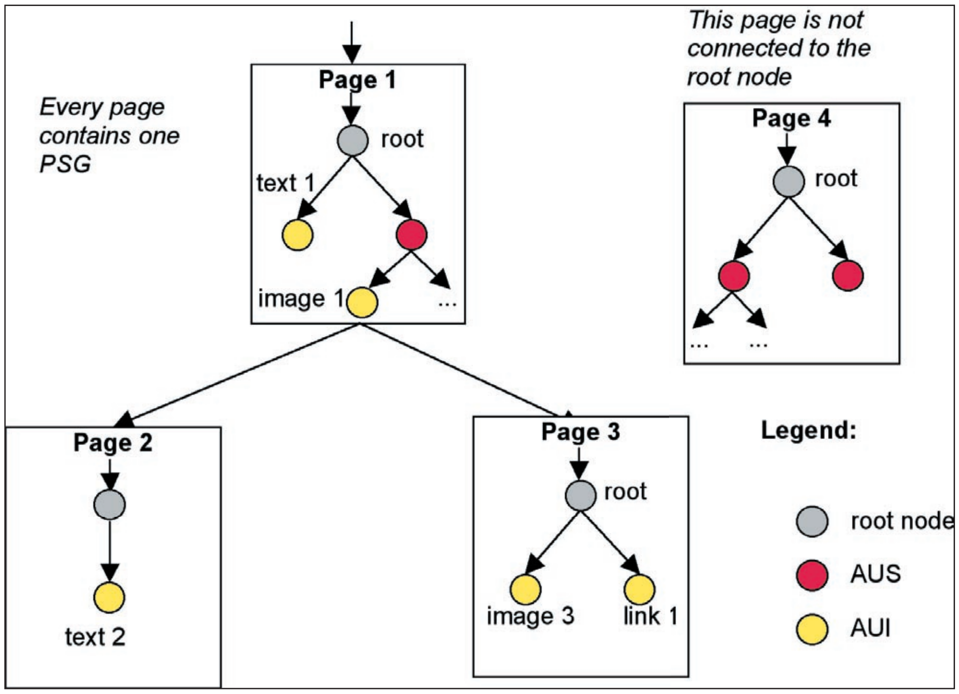


Figure 3: The relationship between an SSG and PSG's. Each PSG is a node of the SSG. The SSG defines the navigational structure of the website while the PSG represents the mapping between information and layout on any given page.

The Content Modification Framework

The content modification framework, here referred to as CMF, is a user-friendly environment that allows even non-technical users to easily update and add content. Because this framework is meant to be used by anyone operating the website to modify information, the ease of use as well as elimination of redundancy were the paramount design goals. The complete interface is accessed through web pages in order to ensure independence from the client's operating system. Access to this interface is restricted by username/password combinations.

After accessing the CMF, the user is presented with a list of pages currently available from within the site (see Figure 5). If the user wishes to create a new page, the parent of this page needs to be specified. A page is allowed to be deleted only if it has empty content and has no children in the SSG. With these stipulations, the SSG is always guaranteed to be well-formed without restricting flexibility and prevents accidental deletion of crucial pieces of information.

If an existing page is selected, all the content on the web page is displayed according to the structure indicated in the PSG so that the user can get a feel for what the real web page looks like. The user can easily create, modify, and delete AUI's and AUS's. It is also convenient to change the location of an AUI within the current page and to move it to a different page as discussed before (see Figure 6). Any allowed operation is guaranteed to result in a valid PSG. Graphically, moving content can be viewed as specifying a different parent for a given subtree of the PSG, where the parent can be an AUS within the current page or in a different page. Here, extensive checks are necessary to ensure the legality of a move operation. For example, it is illegal to move a node into some part of its own sub-

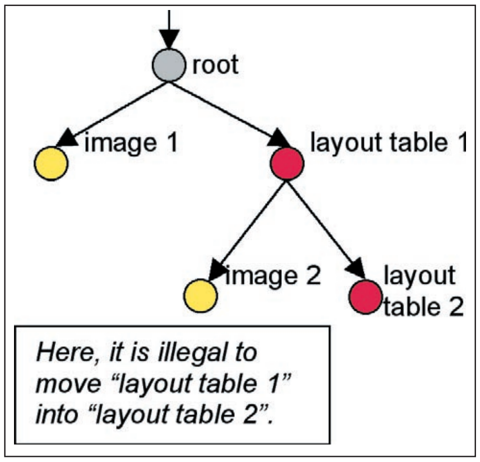
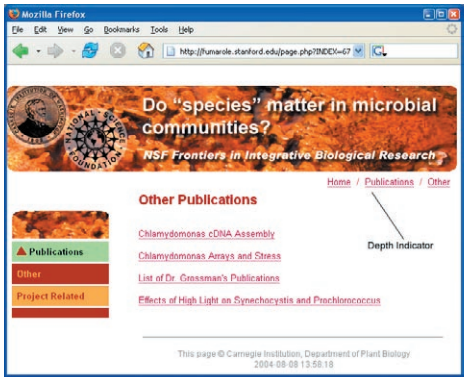

root
image 1
layout table 1
image 2
layout table 2
Here, it is illegal to move "layout table 1" into "layout table 2".

Figure 7


Mozilla Firefox

http://fumarole.stanford.edu/page.php?INDEX=67

Do "species" matter in microbial communities?

NSF Frontiers in Integrative Biological Research

Home / Publications / Other

Depth Indicator

Publications

Other

Project Related

Other Publications

Chlamydomonas cDNA Assembly

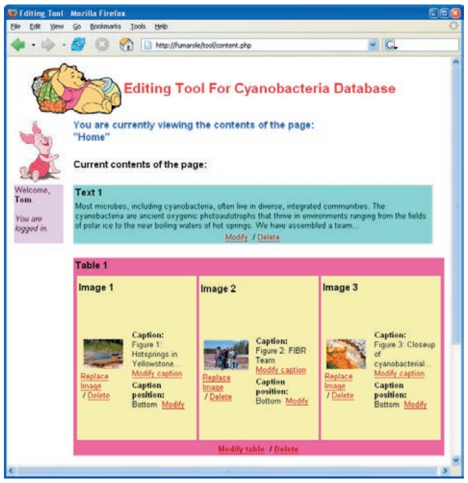
Chlamydomonas Arrays and Stress

List of Dr. Grossman's Publications

Effects of High Light on Synechocystis and Prochlorococcus

This page © Carnegie Institution, Department of Plant Biology
2004-08-08 13:58:18

Figure 4: A page taken from the Carnegie Plant Biology website developed using the CMS discussed in this paper.


Editing Tool - Mozilla Firefox

http://fumarole/tool/content.php

Editing Tool For Cyanobacteria Database

Welcome, Tom. You are logged in.

You are currently viewing the contents of the page: "Home"

Current contents of the page:

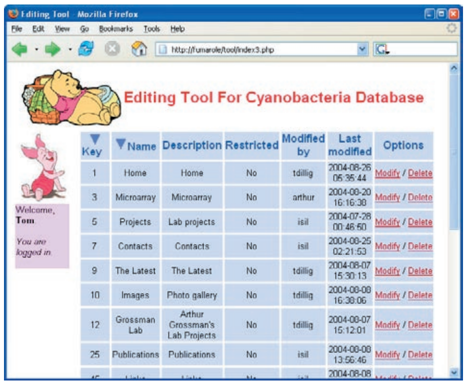
Text 1
Most microbes, including cyanobacteria, often live in diverse, integrated communities. The cyanobacteria are ancient oxygenic photoautotrophs that thrive in environments ranging from the fields of polar ice to the near boiling waters of hot springs. We have assembled a team...
Modify / Delete

Table 1

Image 1	Image 2	Image 3
Replace Image / Delete. Caption: Figure 1: Hotsprings in Yellowstone... Modify caption. Caption position: Bottom Modify	Replace Image / Delete. Caption: Figure 2: FIBR Team. Modify caption. Caption position: Bottom Modify	Replace Image / Delete. Caption: Figure 3: Closeup of cyanobacterial... Modify caption. Caption position: Bottom Modify

Modify table / Delete

Figure 5: A user-friendly CMF shows all pages that are reachable in this website


Editing Tool - Mozilla Firefox

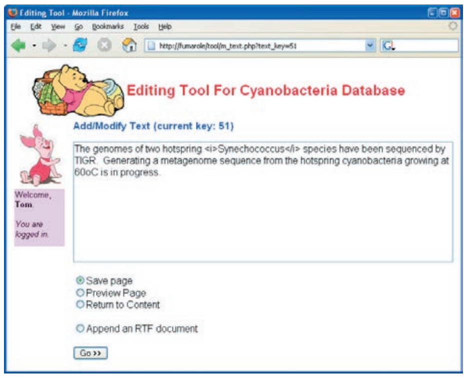
http://fumarole/tool/index3.php

Editing Tool For Cyanobacteria Database

Welcome, Tom. You are logged in.

Key	Name	Description	Restricted	Modified by	Last modified	Options
1	Home	Home	No	tdillig	2004-08-26 06:35:44	Modify / Delete
3	Microarray	Microarray	No	arthur	2004-08-20 16:16:30	Modify / Delete
5	Projects	Lab projects	No	isil	2004-07-28 00:46:50	Modify / Delete
7	Contacts	Contacts	No	isil	2004-08-25 02:21:53	Modify / Delete
9	The Latest	The Latest	No	tdillig	2004-08-07 15:30:13	Modify / Delete
10	Images	Photo gallery	No	tdillig	2004-08-08 16:38:06	Modify / Delete
12	Grossman Lab	Arthur Grossman's Lab Projects	No	tdillig	2004-08-07 15:12:01	Modify / Delete
25	Publications	Publications	No	isil	2004-08-08 13:56:46	Modify / Delete

Figure 6: The content of one page, composed of one AUS (here Table1) and many AUI's. The structure of content represents the structure of the PSG.


Editing Tool - Mozilla Firefox

http://fumarole/tool/m_text.php?text_key=51

Editing Tool For Cyanobacteria Database

Welcome, Tom. You are logged in.

Add/Modify Text (current key: 51)

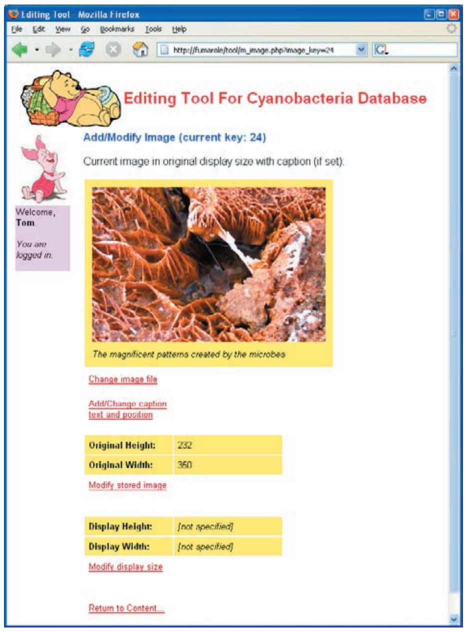
The genomes of two hotspring <i>Synechococcus</i> species have been sequenced by TIGR. Generating a metagenome sequence from the hotspring cyanobacteria growing at 60oC is in progress.

[✓ Save page]
Preview Page
Return to Content

Append an RTF document

Go >>

Figure 8: Text can either be manually entered or imported as an RTF document.


Editing Tool - Mozilla Firefox

http://fumarole/tool/m_image.php?image_key=24

Editing Tool For Cyanobacteria Database

Welcome, Tom. You are logged in.

Add/Modify Image (current key: 24)

Current image in original display size with caption (if set):

The magnificent patterns created by the microbes

Change image file

Add/Change caption text and position

Original Height:	232
Original Width:	350

Modify stored image

Display Height:	[not specified]
Display Width:	[not specified]

Modify display size

Return to Content...

Figure 9: Images can be easily uploaded and resized without using external tools like Photoshop.

tree (see Figure 7). All the algorithms employed for this purpose are based on standard graph-traversal techniques and are a major source of complexity in implementing this stage. For an overview of the major algorithms modified for our purposes, see [8].

Every AUI and AUS can be added to the page currently under modification. Because AUI's always form leaf nodes in the PSG, only their content and parent needs to be specified when creating them; no layout information is necessary to fully define an AUS. For the AUI type "Text", the user can enter text with its correct HTML tags, such as those that define bold, italic, etc. (see Figure 8). In addition to specifying text using HTML tags, our system offers an RTF (rich text format, [9]) to text converter which automatically converts an RTF document to its HTML representation. The RTF format was chosen as a least common denominator, since all current text processing tools allow the user to save in RTF. The tool used in our implementation is a modified version of RTF to HTML, which discards any formatting applied to the RTF document and only extracts headings, paragraphs and bulleting/numbering [10]. Even though this process is based on heuristics, it does a reasonably good job when adding content to our reference page. Out of 17 documents added to our test site, all of which were at least a page long, only six mistakes had to be corrected. While this fact was not established in a statistically significant way, the tool greatly reduces the effort and time required to publish existing Word/Text Processing documents as part of a web page—an extremely frequent need in many organizations.

Images are added in a similar fashion (see Figure 9). In addition to simply uploading the file, the user can also perform basic editing options such as resizing or turning. This feature, combined with a tool provided by our interface to resize every image to "reasonable" dimensions, makes the task of posting images easy and reliable for even the most inexperienced users. If a certain website features many pages with identical images, it will be more desirable to create an "image alias" pointing to one image to allow for easy duplication. However, the complexities introduced by this solution are real and its benefit is doubtful for sites benefiting most from this interface. Downloads and links are analogues to the AUI's already described; there are no special features associated with them.

When adding an AUS, the user is first asked to choose the dimensions of the layout table he/she wishes to use for content. At this stage, all the auxiliary attributes such as spacing and alignment are also specified. Then, already existing content on the current page can be added to any field in the table if desired. The table size as well as all of its attributes and children can also be changed.

Conclusion

We described a content management system that can deal with any arbitrary structure and content while accomplishing the prime goals of any CMS: ease of content modification, separation of content and structure, and guaranteed well-formed pages. There are no restrictions on the kind of content that can be displayed. Even if it cannot be built form one of the described AUI's, the modularity of our solution allows new AUI's to be introduced very easily.

The most significant hurdle is performance. Every page construction by the display engine consists of multiple database queries and code to be executed. Even though we found this solution to scale extremely well to medium load, it cannot handle extremely high volume sites. This however can be easily compensated by caching. Many commercial and free tools exist for this purpose and a special version that regenerates pages only when the content has changed is in planning for our system [11].

Our system is currently deployed at the Carnegie Institute, Department of Plant Biology (http://fumarole.stanford.edu).

Acknowledgements

The system described in this paper was developed in the summer of 2004 at the Carnegie Institute of Washington, Department of Plant Biology, as a result of Dr. Arthur Grossman's and Dr. Devaki Bhaya's wish for a totally flexible web-page design instrumental to their work. We would like to thank Dr. Bhaya and Dr. Grossman for their excellent mentorship and steadfast support for our project.

References

1. Dudek, David and Wieczorek, Heidi. "A simple web content management tool as the solution to a web site redesign." Proceedings of the 31st annual ACM SIGUCCS conference on User services, September 2003.
2. Sitecore .NET CMS. http://www.sitecore.net/.
3. The OpenCMS project. http://www.opencms.org/.
4. Vignette. http://www.vignette.com/.
5. Challenger, Jim, Iyengar, Arun and Witting, Karen. "A Publishing System for Efficiently Creating Dynamic Web Content." Proceedings of IEEE INFOCOM, 2000.
6. PHP – Hypertext Preprocessor. http://www.php.net/
7. MySQL – The World's Most Popular Open Source Database. http://www.mysql.com/.
8. Sedgewick, Robert. Algorithms in C++. Addison-Wesley Publishing, 1992.
9. RTFtoHTML. http://www.pa.msu.edu/reference/rtftohtml-docs/rtftohtml_overview.html/.
10. RTF Standard Version 1.6. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnrtfspec/html/rtfspec.asp/
11. K. Selcul Candan, Wen-Syan Li, Qiong Lup, Wang-Pin Hsiung Divyakant Agrawal. Enabling Dynamic Content Caching for Database-Driven Web Sites. ACM SIGMOD, May 2001.

Thomas Dillig and Isil Ozgener

Thomas Dillig is a junior, who is majoring in computer science and minoring in Management Science and Engineering. Tom was born in Munich, Germany.

Isil Ozgener is also a junior, majoring in computer science and minoring in mathematics. She is originally from Istanbul, Turkey.

Tom and Isil are currently both interested in areas of computer science related to program analysis and verification. They are both planning on pursuing doctoral degrees in Computer Science after getting their bachelor's degrees next year.