

University of Illinois, Urbana-Champaign

The Brutally Honest Web Publishing Guide

A review of the book
Philip and Alex's Guide to Web Publishing
written by Philip Greenspun

Submitted to:

Dr. Ralph Johnson
CS428 – Software Engineering II

April 27, 2006

By

Scott R. Armstrong
409 West Dixon Street
Polo, Illinois 61064
(815) 946-9103
Scott.Armstrong@acm.org

Table of Contents

1.	Introduction	1
2.	Review Points	1
2.1.	Suck.com and Grubby Internet Entrepreneurs	2
2.2.	Scalable systems versus static Web pages	3
2.3.	HTML in 21 minutes, and add pictures while you're at it	4
2.4.	Publicize, but don't offend	5
2.5.	Running your own Web server and why it's hard	5
2.6.	Tracking the user and analyzing the information	6
2.7.	Sites that prosper are dynamic	6
2.8.	Database management systems and interfacing with the Web	7
2.9.	The grubby business of ecommerce	8
2.10.	Case studies	8
2.11.	Even better than just ecommerce, influencing change	9
2.12.	The future is coming, the future is coming!	9
3.	Other comments	10
4.	Summary	11

The Brutally Honest Web Publishing Guide

1. Introduction

Philip Greenspun provides the readers with an entertaining, extremely informative and point blank delivery of what a Web server should do and how to make it do so. Though the summary sections at the end of each chapter outline the main points of the chapter, simply extracting the summaries would leave the reader with a very empty helping of the *Web Publishing* content. Reading the entire book, though, gives the reader quite a bit more detail on Tcl scripts and Philip's personal opinions that is necessary. Strewn throughout the book are photographs that serve as pleasant diversions, and more importantly, as practical implementations of some of his points. Dr. Greenspun is able to present countless genuine examples of what could go wrong (or what has gone wrong), how to solve a Web problem correctly, and then how to do it better. He also interspersed his opinions on everything from education and social obligation to the politics related to various career choices. His delivery of the information is well organized, easy to understand, interesting (much more so than the traditional "how to" books), and most importantly, very, very direct. I didn't notice an effort to be sensitive to anyone except Alex (his dog), but that may be part of the formula that makes this book so enjoyable to read.

Because most of my references to this book are from the online version, I won't include page numbers to indicate the location of the reference. Generally speaking, my comments and opinions are identified as such; otherwise, the content in this paper is based on Dr. Greenspun's book.

2. Review Points

Dr. Greenspun's presentation style is extremely upfront, direct and to the point, and at times, almost "in-your-face" direct. I started reading his book using both the online version and the "dead trees" version, referring between the two for any significant differences. Though I thought I would be more comfortable holding the processed tree carcass while lounging in my chair, I eventually read mostly from the Internet version (the paper version was certainly easier to read while traveling). The photographs in the book were mostly the same in both versions, but while reading the Internet version of *Web Publishing*, I realized that the book had become very interactive and alive. The links to other pages, information and examples added the support or clarification when needed without distracting the reader—it was always easy to get right back to where I was reading when I linked away.

If there had been no technical content at all, this would still be a wonderful book, serving as a practical implementation of a different solution for publishing. The technical details he provided, though, made this a very complete "how to" book that is enjoyable to read and extremely informative. Using this book alone, a reader should be able to set up an interactive Web community on the Internet and maintain it to a reasonable degree of efficiency.

The book is organized as a series of subject-specific chapters, and though I won't step through each chapter individually, I'll discuss them in groups or pairs to tie together some subtle

themes. Some of the resounding themes throughout the book, though, include the need for organized data to support and automate a Web site, as well as the need to create an interactive, collaborative environment that will serve, and be improved by, a virtual community of users.

2.1. Suck.com and Grubby Internet Entrepreneurs

Dr. Greenspun urges the readers (soon to be Web publishers) to avoid creating boring Web sites. Well, “urges” probably isn’t a strong enough word. How to do so is to ask the marketing questions that are standard in any domain: 1) what will attract interest (visitors, buyers, etc.); 2) what will keep them at the site or make them return; and 3) once the users are looking at your site, what will they do? His almost simplistic advice that resounds throughout the early parts of the book is to post some “magnet content” that will attract users, develop a technical means of online collaboration, and be prepared to interact with and respond to the users. Anything else is a static Web site and will deserve honors at suck.com (A shining example of a static Web site, indeed—in the spring of 2006, suck.com had this message posted at the top of the main page: “for 27 December 2000. Updated every WEEKDAY.” [SUCK]).

Additionally, the user should make the Web site the primary source. His example of the MIT Media Lab celebration using a Web site (www.1010.org) with the adjunct role sans “magnet content” on a very expensive system was very clear and to the point—don’t let the Web content play the supporting role unless it’s okay to ignore it completely and throw away the money spent on the system.

He also urges us to step away from the traditional vehicles of communication and look for innovative means of collaboration. He used WimpyPoint, a Web-hosted presentation system, as an example of a centrally stored presentation server that can be accessed across a network or even the Internet. This presentation aid doesn’t require specific versions of client applications, templates or settings on the presenter’s computer, and the presentation content can be modified collaboratively for just-in-time presentations.

He also provides four methods for using the tools—tools that were originally intended to share information freely—to make money on the Web, if we must. How much we make depends on what we do. The traditional method for making money on the Internet is best suited for a more static site and includes providing traditional content to attract users, and getting “kickbacks” from advertisements posted on the site. This requires the least amount of imagination and the greatest amount of effort. It also allows for a greater degree of error and lost profits. The Web site with collaboratively created information is much more interactive and will have more current content, but the revenues will still come from advertisement “kickbacks.” If you’re running a collaborative site (eBay and Consumer Reports are examples of Web sites with collaboratively created content), you should have already automated the logging and billing features for the click-through advertisements. Sites that provide a service are the third type, such as a virtual storefront. The fourth category of grubby entrepreneurialism on the Internet is a site that provides a standard to seamlessly query multiple databases or systems. He provides some examples, but my favorite “category 4” site is www.pricewatch.com - a great place for a computer addict to get his fix.

These types of grubby, moneymaking sites aren't mutually exclusive, either. He recommends using a category 1 site to attract business to the other three categories, adding to the importance of magnet content. Interestingly, the companies with virtual storefronts haven't been the big winners in the online business world. He claimed that the "unexpected achievers" have been those companies with online communities that aren't used in support of any other static form of business presence.

2.2. Scalable systems versus static Web pages

I've regularly claimed that the human's ability to collaborate with others at a very high intellectual level is one of the most significant of the abilities that set us apart from the rest of the animals. New forms of collaboration change the social order and expand the limits of the traditional group far beyond the physical boundaries of a building, a town, or a region. Online collaboration can create real communities, far different from the dysfunctional Junior High School community that Philip used as an example. Online groups will gather because they have common interests—not because they happen to live within the same municipal boundaries. Our Software Engineering courses are a form of an online community, allowing people interested in learning more about everything related to Software Engineering, regardless of where they live (so true in my case). Our project group, for example, included people from Korea, Texas, Massachusetts, New York and various parts of Illinois. Such a community wouldn't have been possible 20 years ago, but we're now so used to them that we expect to be able to find them everywhere.

Dr. Greenspun presents a short history of the evolution of communities as 1) ancient—face to face communication; 2) modern—mass media subjected to government control or influence; 3) early Internet—E-mail, mailing lists and shared documents; and 4) Internet circa 2003 (or 1999, depending on the version of the book)—corporate involvement in or creation of communities that will obsolete the modern communities.

He identifies the challenge facing online communities as the technical ability for the community to grow to support growing interest in the community. In order to serve the communities, the online community system needs to scale, and as it scales, the system will need more maintenance and administrative attention. Adding to this challenge, Dr. Greenspun states that the revenues from online communities rarely pay for the appropriate system support staff because the cost per user is too high. In order to be relatively successful, and to reduce the cost per user as much as possible, you should consider using Philip's recipe for success: 1) use magnet content that has been authored by experts, 2) provide a means for online collaboration, 3) provide powerful utilities to browse both the magnet content and the contributed content; 4) delegate the duties of moderating the community; 5) identify the members who cause undue work for the community and moderators and find a way to change their behavior or exclude them from the community without them knowing it (he gave an example toward the end of the book of an automated script that ran in response to queries from unwanted users—the script gave those users the impression of an unresponsive server and they left the community of their own accord); 6) provide a means for the community to extend the software or the system themselves.

This will help the community grow, but you still need to be able to keep metrics on everything from users to expected revenues. For this, he recommends using a relational database and a 7-module software system to do the following: 1) track users, their contact information and collaboration preferences; 2) track site content, who submitted it, and how it relates to the other content; 3) track who looked at what content; 4) identify cost in terms of effort for each user; 5) identify how users access the site and whether they are clicking through other sites to get there; 6) track which advertising banners have been used and whether they were effective; and 7) help the site maintainers keep in contact with the different types of users. This is good marketing advice for any company or organization, but he provides advice on the technical solutions to make this work for online communities. Such solutions require that you identify and know the server operating system, select a relational database, choose a Web-database integration tool, write SQL data models to reflect your data structure, design a user interface for the transactions that you want to allow, and write dynamic pages to insert and display the database contents. A means to automate the tasks resonates throughout the book. It should be painfully clear to anyone who maintains a Web site that updates and maintenance are the most tedious chores. Any automation is good. Bruce J. MacLennan's principles of programming languages apply to this task as well, especially "Automate mechanical, tedious or error-prone activities" [MACL, 5].

Not everyone can master all aspects of a solution needed to run an online community, so his recommendation is to find an existing package that solves most of the problems that your online community will generate.

Some good advice that is very similar to a software engineering process is the development plan to create a static web site. Draw a site map (architecture), assemble and structure the content (design), make a text-only site (implementation), hire a graphic designer to make it look nice (iteration), and set up a maintenance plan (maintenance). Though this advice is directed toward the Web site as a complete system, the software engineering approach to solving a problem using a procedural solution is useful in many domains. I have often borrowed themes and solutions from computer science to help overcome challenges and find solutions to problems completely unrelated to computers.

Challenges that will persist in any online community will include how to manage and control versions of the system and over-optimism on the part of the community manager. His solution for every problem, it seems, is a relational database.

2.3. HTML in 21 minutes, and add pictures while you're at it

It is quite interesting that Dr. Greenspun provides such a nice, entry-level tutorial on how to code a Web page using HTML and then recommends that we ignore HTML in favor of something completely different. I've seen several static Web pages that Philip would describe as fancy but worthless. I agree with his assessment that flash can't make up for a lack of content, and I try to avoid flashy features in pages that I create. His advice about making all of the content interactive and meaningful is in line with his approach to depart from the traditional publishing paradigm. The seemingly random photos in his digital book link to pages on his photo.net Web site, include a short description, and even a link to a very high-resolution image of the photo. His references link to the appropriate source and his examples link to extended

descriptions. Almost all of the content managed by his servers appears to be arranged in a database, called with code written in what appears to be his favorite language, Tcl. HTML can't automate a Web site, but the HTML can be automated to create a dynamic, database-driven, and easily maintained Web site.

Though it is possible to learn HTML in 21 minutes (probably less), Philip doesn't imply that you can create a virtual online community in mere minutes. That takes time, work and most importantly, trolling for community members. Without community members, you don't have a community.

2.4. Publicize, but don't offend

Dr. Greenspun explains the components of a search engine (Web page crawler, full-text indexer, and query processor) and describes how to take advantage of those search engines to enhance the magnet content on a Web site. The key feature is that the text is what gets indexed, so the more text on the page, the better. Adding descriptions to "meta" HTML tags can increase the indexed references to a site, as can huge amounts of content (see the *Web Publishing* book, for example). There are methods to exclude the crawlers from Web sites (specific code in a robots.txt file), but the search engines are also being designed to identify unfair methods of getting pages indexed.

I've seen an example of this feature already with Google. Our group's new, transition Web site was linked from the Software Engineering page, and we submitted to Google for indexing as well. It did show up as the number two link (behind our project's link on the Software Engineering Wiki) when searching for our project name on Google for a couple of weeks. The indexing systems look for duplicates or mirror sites and exclude the non-original sites, and it seems that the Google indexing system considered our project's new location to be a mirror site and has since excluded it from the responses to queries on our project name. We'll have to change the first page of our new Web site significantly in order to get it listed again. Our project's new location does show up in a Yahoo search, but it is only ranked fifth in the listing.

2.5. Running your own Web server and why it's hard

Though Dr. Greenspun doesn't say not to run your own Web server, he does explain why it's hard work. Like most things, it costs money and time. He describes the layers involved in running a Web server in terms of non-technical and technical layers, each with its own challenges. The idea, money, and content with all of the related marketing issues are non-technical, but without having solutions for each of these layers, there's no need to worry about the technical layers (server software, database and its toolkit, server operating system, network and power).

This book isn't just about software, and this section clearly describes the entire set of requirements that need to be satisfied if you want to host an online community. The problem, though, can be more easily solved by using traditional problem solving methods, and most certainly by using the common problem solving methods used for software engineering. He breaks the problem into modules (layers) that can be dealt with separately. He then describes the

system architecture (more than just software, but still a system), and details the design of the various systems throughout the book. For those who don't have the time or experience to tackle a system, he does provide information on hosting options that include solutions for many of the technical layers. I have had some experience with Web sites hosted within an organization and Web sites hosted by hosting companies. As with any solution, each has its own set of trade-offs.

2.6. Tracking the user and analyzing the information

The book describes the implementation of Netscape's magic cookie technology and provides some sample implementations as practical exercises. The cookies provide a means of overcoming the lack of state in Web-based interaction with a server. A cookie allows the server to recognize the user and to let the user continue their interaction as if it had never stopped (shopping, signing up for classes, etc.). By using a database to store the user's identity (tracked using cookies), the server can be configured to tailor the services for the specific user. Cookie information can be treated like mailing lists and misused, but the use of cookies will provide the users a much nicer and simpler Web experience.

Collecting user activity information through the use of cookies makes it possible to collect a ton of information, but you may have to be very creative with the data if you hope to glean more from the data than is typically possible with open-source analysis tools. Regardless of how you choose to collect information, you need to have a good data model in mind in order to make the data truly worthwhile.

2.7. Sites that prosper are dynamic

Dr. Greenspun stresses that you can use almost any programming language for the server-side programming, but he doesn't hide his love of Tcl and AOLServer. Regardless of which programming language you use, anything is better than a static HTML page. He provides good guidance on how to select the appropriate language for our particular needs, and again gives us a list to follow: 1) decide between a dynamic document or a program with a Web interface; 2) select a computer language; 3) select a program invocation mechanism; and 4) select a Web server program that will support the other three choices. He also highlights the problems that we'll need to keep track of, including process forks and database connections. The fewer forks, the better. He provides some entertaining examples of Web pages as programs, discusses the evils of each implementation, and after taking the opportunity to climb onto his various soapboxes, offers us the source code.

Despite the focus on the Web server program, he repeats throughout the book (and emphasizes here) that the content is the most expensive asset, and also the most critical.

He continues to tout the benefits of databases supporting Web sites and states "the most profitable sites on the Web are really databases." [GREE, 323] Referring to the four categories mentioned earlier, database-backed systems and systems that query multiple databases have the potential to generate more revenue than other types of systems. The simple plan to profit using a database-backed site includes: 1) develop a good data model; 2) identify the legal transactions

that you want to allow to run against the data; and 3) map the transactions onto a Web-based form to allow access to the legal transactions.

Though there are many middleware solutions on the market to solve these problems for you, he warns against using junkware and middleware, and though he has some kind words to say about CORBA, he warns against all middleware solutions in favor of server-side programs and databases. He covers security regularly, but his architecture of choice is a Unix system, an Oracle database, and AOLServer interpreting Tcl scripts to generate and present the Web content.

The content on Web sites isn't all static HTML pages. Some Web sites are programs that dynamically generate HTML code to present the content. Some Web sites use server-side programs to access database information, and others allow client-side code (Java, for example) to directly access a database (or through middleware). He warns of the security risks with any solution that allows a client-side application to directly access a database.

As a final plug for programs instead of static Web pages, Greenspun notes that Web site programs are easier to maintain than static content on a Web site. They're easier to maintain because you don't need to add information daily if you can program the system to accept, organize and present content submitted by users. They're easier to maintain because they're automated.

2.8. Database management systems and interfacing with the Web

Database systems manage data transactions quite well. Dr. Greenspun details the challenges of database transaction engines and makes it clear that there is no need to recreate that wheel. As a declarative language, databases are easier to program and troubleshoot and the logic is much simpler than a procedural language. He provides a nice SQL primer along with his applause for Oracle's ability to handle concurrent reads and writes, a quality absent in Microsoft's SQL Server (Since the most recent updates to this section, Microsoft released SQL Server 2005 that is supposed to handle write locks by row, allowing concurrent reads and writes.). Other than finding the correct way to log into the database system, a reader should be able follow Dr. Greenspun's examples to create tables, populate the tables, and generate SQL query statements that return the desired information.

Of course, Web users shouldn't access the database directly using SQL statements. They need a means to access only what they should have access to, and that means of access should protect the integrity of the database and the system. CGI Scripts, Java applets, and Tcl scripts can solve that problem, but again, Philip's solution of choice is the AOLServer-Tcl script combination. He says that CORBA doesn't improve the security offerings over classical HTTP choices. Coupled with the added complexity of CORBA and the limited implementation of ORBs as middleware Web solutions, Dr. Greenspun doesn't make a sales pitch in favor of CORBA or any middleware system.

To the reader, he advises horsepower over finesse when supporting a database system. Buy big servers with many large disks, and arrange for support from genuine database administrators. Oh, and backups shouldn't be an afterthought!

2.9. The grubby business of ecommerce

Though Dr. Greenspun's summary of ecommerce is quite short, the advice he provides is very in depth. Not only does he provide practical examples for the hardware and software needed to do business on the Internet, but he also provides a nice eBusiness-101 lesson for the reader. I like that he identifies business rules as not only difficult, but important. The software and hardware solutions on the market today are really supposed to simplify the chores of doing business, but the most important part of any business is knowing the business process.

Business processes and rules can't always be packaged and shrink-wrapped. He uses MIT's bookstore as an example of appreciated functionality that would be absent from a generic software package.

The payment transaction functionality, though, is best done through an existing service. He provides examples of different payment transaction services that are separated into two different architectures. One of the architectures stores the customers' credit card information on the merchant's servers, and the other stores the credit card information on the transaction service's servers. The merchant needs to decide who will manage the credit card information, understanding that the user tracking information (in terms of past purchases) will likely accompany the payment transaction history. There are tradeoffs for either solution.

Using a service that stores the users' transaction history adds another layer of database interaction challenges. Network connection problems can result in failed credit card authorizations, payments not posted, or even undelivered messages reflecting a completed transaction. Confirmation of changes committed in the payment transaction service's system may not successfully be returned to the merchant's database. These kinds of problems with an added layer of complexity can leave orders unprocessed but paid for, processed but not paid for, or worse yet, neither processed nor paid for.

Dr. Greenspun shared his experiences with CyberCash as an example of what to avoid. The frustrations of dealing not only with a bank, but also a card processor and a gateway between the two when trying to isolate and resolve payment problems are only compounded by the costs associated with setting up these services. He also provides some interesting perspectives on calculating sales tax (and what it costs to automate that feature), the complexity of system failure when using so many services, and first-hand experience with a live system test that failed miserably.

2.10. Case Studies

The case studies toward the end of the book demonstrate the structure of his solutions, including user interface design, expected or intended user interaction with the system, and data modeling suggestions. The structured examples are intended to help the Web publisher plan the

system. Any plan is better than no plan, and his examples of how to plan are easily better than average. What he refers to as the user interaction design could probably be presented as system architecture and design based on user requirements, but the focus of his efforts are truly on the user's experience instead of the system design or structure. The system processes (database-driven, of course) involve the already described steps of defining the data model, defining the legal transactions, mapping the transactions to a Web form, and writing the supporting code. The moral in each of the case studies is that a good data model, a good database (Oracle is Philips' preference), and a plan will result in success.

2.11. Even better than just ecommerce, influencing change

Dr. Greenspun highlights some of the great benefits of collecting, organizing and offering information on the Web, including the use of Web-based databases to expand the awareness of issues affecting our environment. People, armed with facts that have meaning, can group together and effect change. He uses www.scorecard.org as a shining example of a system that can organize information about toxins that is interpreted to have meaning, is locally relevant for the reader, indicates how the reader's area ranks relative to other areas, gives details about the specific toxins in that area and what makes them harmful, and provides a mechanism to influence change. The benefit of such a system is that it feeds the belief that information should be freely available, and it makes it possible for people to become active in making their world a better place to live. This type of a system is especially useful for non-profit organizations hoping to make a difference.

As with most of the other examples in the book, Dr. Greenspun provides the code to support the features that are unique to this solution. He used the www.actionnetwork.com site as an example of some challenges faced when trying to interface with a database system using a batch upload instead of the single-record entry using Web forms that he described throughout most of the book. He finds the opportunity to talk briefly about login credentials for users hoping to upload information to the Web site and how he designed the solution, but this appeared to be almost an afterthought.

2.12 The Future is coming, the future is coming!

Philip Greenspun summarizes his book with a reminder that many of the novel ideas advertised today are really just repackaged ideas from the past. The difference is the packaging, and if purchased or accessed through the Internet, the delivery. It is very clear that Dr. Greenspun sees the future solutions for sharing information as Web-based activities. It makes sense that an environment so nicely suited for collaboration beyond the physical walls of a building would help usher in a new paradigm for sharing information. Isolated information will be of little use as people begin to appreciate, expect and demand extended, no-physical-bounds collaboration. Anything that plugs into the wall will probably plug into both power and a network. Paradigms will continue to change as information organization and sharing become commonplace.

The paradigms encapsulating the software we use will probably change as well, or at least, Greenspun says, they should. His software licensing "new world order" will require

changes in thought and a huge dose of trust as applications are accessed or delivered via the Web, licensing fees are reduced to rental use fees, and software companies collaborate and truly take make use of “code reuse” in order to enhance and simplify the user’s computing experience. Appliances would share information (through databases, no doubt) across the home network and the future, ubiquitous Internet. Of course, we’ll have to find a way to make the Web servers in the appliances secure.

Greenspun’s final bash is against junkware, in which he didn’t specifically include middleware (this time). He is an advocate for identifying the requirements, creating an architecture to satisfy the requirements, designing and implementing a solution (in Tcl with AOLServer accessing an Oracle database), testing the system with live users, and ultimately, finding the easiest means to maintain the system. He doesn’t believe that non-programmers should select shrink-wrapped junkware solutions for each module of the system; rather, he believes that non-programmers, wishing to use an existing package, should use a known and tested standalone package that has an active user community.

3. Other Comments

The “dead trees” version of his book from 1999 (reprinted from the 1997 version) differed from the online version, primarily because the digital version has been updated since 1999. The digital version includes updates from his other writings that are posted on the photo.net Web site, and not having gone through the detailed review reserved for “dead trees” books, this version includes some silly writing errors (“...an girlfriend...” and the original, linked article had “...and old girlfriend...” – he removed a modifier but didn’t make sure that the sentence grammar was correct). This highlights the issue of the value and quality of digital versus “dead trees” publishing. Printed books typically undergo a fairly detailed review process prior to putting the ink on the paper. Web-based publishing doesn’t always require the same degree of review, and in many cases, the content provider (the person who runs the site) posts information without any external review for the accuracy of grammar or content. Even though print sources can contain errors, I would expect that there would be far fewer errors and much more accurate information on a processed tree carcass than in an HTML presentation of a book.

Finding solutions for the online community presents challenges that must consider collaboration and communication. The ability to change the content is an attractive feature of these communities, and that feature creates challenges in using Web-based content as a static reference for such things as research papers. I suspect that the contents of the digital version of *Web Publishing* haven’t changed dramatically in the last 3 or 4 years, but they have changed. Either our method of citing references will need to change (and be widely accepted), or the easy-to-change Web-based sources will not provide the value and support needed to further a line of intellectual research or discussion. I applaud Dr. Greenspun’s crusade to spare the trees, and I try to encourage paperless collaboration at work whenever possible (I’ve made some progress!), but I will continue to seek at least some static documentation for references when references are required.

The “coffee table” style of this book (it includes several interesting pictures, inserted throughout the text in a seemingly random manner) is a rather interesting way to keep the reader

focused on the content without boring the reader to tears. The linked pictures in the digital version allow for a quick diversion and an easy return to the text.

Dr. Greenspun loves relational databases. He loves relational databases so much that I suspect he would be able to propose a database solution to almost any problem. He also opposes the use of middleware—apparently at any cost—in favor of server-side applications triggered by a Web pages generated by with Tcl script. In keeping with the theme of change, it would be worthwhile for him to consider other solutions.

He also climbs onto several soapboxes during the presentation of the book, including copyright issues, his personal views on the cost of tuition at MIT, his contempt for Bill Gates, the general ignorance and stubbornness of managers (and users), and other issues. He is as enthusiastic about Web publishing as he is about his soapbox topics.

And Alex photographs much better than Philip.

4. Summary

From a computer science perspective, Dr. Greenspun provides implementation guidance and formulae for publishing Web sites. He provides detailed software architecture options and an overview of hardware and network architecture requirements to support Web sites. It is possible to interpret his lists, guidance, instructions and comments in the form of a classic software engineering process—requirements development, system architecture development, system and module design, implementation, testing and maintenance. Though he doesn't describe his process as simply as this, the classic parts of the process are all there.

From a personal perspective, I found the digital version much more enjoyable, and the linked references added the extra content that made the book very informative. I also enjoyed the links to other stories and examples that weren't printed in the "dead trees" version of the book.

From the perspective of a reader hoping to publish a Web site, this book provides the full range of tools, advice, instruction, examples and related information needed to create a worthwhile Web community. This is a very good book and should be the book of choice ahead of anything Web-related with a title that includes "for Dummies." This is a book for smart people, and it'll make even the not-so-smart reader smarter. Of course, users without existing knowledge of other programming languages and scripting tools will end up creating an AOLServer-based system using Tcl scripts to access an Oracle database. I can't complain about his love of Oracle, but I'm not sure that I'm convinced of the superiority of Tcl and AOLServer over other tools.

I started to tire of the self-congratulatory remarks on the slickness of his solutions toward the end of the book (especially in the ecommerce chapter), and I lost interest in looking through the Tcl code very early on. I know the examples are there if I ever want to implement a Tcl-AOLServer system, but I'll probably only refer to the code for design information.

I thoroughly enjoyed this book, partly because Dr. Greenspun didn't make any effort to hide his true feelings, and partly because it gave me insight into some Web site problems that I hadn't yet taken the time to resolve. I plan to revisit the Web sites that I have been taking care of (or charged with the maintenance of—charity work for local organizations) to implement some automated maintenance features.

I've known for some time that trying to make money by maintaining Web sites is hard work with little revenue, so I'm not hoping to find a way to strike it rich in the Web communities. I do, however, hope to apply the software engineering principles from class, along with the guidance and suggestions from the book, and make my own Web pages a bit more interesting and interactive. After reading *Web Publishing*, it is clear that I have to continually try to change my existing paradigm to keep up with the younger generations. My son just turned 14 years old in mid-April but realized last year that his Web site (www.thesolarflare.net) needed to be interactive. Using the tools provided by his hosting company, he created a Web page to reflect his interests, posted a link to his site in forums and discussion groups with similar interests, and then set up both a chat room system and an online forum on his Web site (it's not the most popular Web site on the Internet, but as a parent, I won't complain). I should have had an interactive Web site years ago, and Dr. Greenspun's book has convinced me to wait no longer—well, perhaps I'll wait until the end of the semester.

I've already recommended this book to our Webmaster at work, and to a couple of the programmers in our department. I've recommended it to friends and family as well. I'll recommend this book to anyone interested in having a digital presence on the Web and I'll recommend it to anyone who needs to develop a different paradigm of organizing and presenting information. There is no need to use the power of the computing systems to simply print paper faster—we need to use the power of computing systems to change the way we think and collaborate. I shall go forth and digitally collaborate!

Works Cited

[GREE] Greenspun, Philip, *Philip and Alex's Guide to Web Publishing*. San Francisco: Morgan Kaufmann Publishers, 1999.

[GREE2] Greenspun Philip, *Philip and Alex's Guide to Web Publishing*, 2003,
<http://philip.greenspun.com/panda>.

[MACL] MacLennan, Bruce J., *Principles of Programming Languages*, 2nd edition. Orlando, Florida: Harcourt, Brace, Jovanovich, College Publishers, 1987.

[SUCK] Web page, April 2006, <http://www.suck.com>.