Technology Electronic Reviews (TER) Vol. 9, No. 4, Sept. 2002

ter
Technology Electronic Reviews
Formerly *Telecommunications Electronic Reviews*

Volume 9, Issue 4, September 2002

REVIEW OF: John M. Cohn, Ann L. Kelsey and Keith Michael Fiels. (2001). Planning for Integrated Systems and Technologies: A How-To-Do-It Manual for Librarians. New York: Neal-Schuman Publishers.

by Corey Seeman

An entry in the series *How-To-Do-It Manuals for Librarians* by Neal-Schuman, this book is an excellent resource for smaller libraries looking to automate their library services. The book follows two editions of the authors' "Planning for Automation" in the same series (published in 1992 and 1997). The earlier books no doubt guided many libraries through the tough move from a paper system to an automated one. I suspect that this latest work will be an equally needed guide for the libraries making that move in the future.

Though there are sections dealing with system migrations, this book is clearly geared to meet the needs of libraries that do not currently have an Integrated Library System (ILS) -- a surprisingly large number do not at this dawn of the new century. The sections defining and outlining the basics of Machine-Readable

Cataloging (MARC) and the underlining style that shows nothing is taken for granted, make it easy to determine that the focus of the book is on libraries moving into automation for the first time. These libraries primarily fall into the following categories:

- Among public libraries, the ones that are not automated are typically the smaller ones and those in rural areas. Many are undertaking the process currently and are joining regional consortiums rather than bearing the fiscal and administrative burden of maintaining systems by themselves.
- Historical Society libraries represent wonderful resources and collections that do an admirable job of documenting local history. However, the majority of historical societies are small (county or city level) and do not have a professional library staff or resources to maintain a library system.
- Business libraries take many shapes throughout the various medium- and large-size corporations in the country. Many businesses have libraries that utilize "off-the-shelf" software such as Microsoft Access or other relational databases to manage their collections.
- Very few academic libraries are not automated at this point in the United States, but there are certainly some that have paper-based systems.

Currently, the majority of library system purchases end up being migrations from one system to another. For libraries in this situation, the book might still be useful because it provides a great deal of forms, tables, and reminders that even the most veteran library system professional might not remember when going through the process. Yet even with this said, many of the sections might be a bit elementary and of little benefit to technically sophisticated librarians and libraries.

The book has many nice features that make it worthwhile for any library venturing into automation for the first time. At the end of every chapter, the authors have included annotated bibliographies of additional readings that are very useful to readers wishing to learn more about some of the concepts touched on in the chapters. Nowhere is this more valuable than in the section on Requests for Proposals (RFPs), where one of the citations leads to sample RFPs available on the Internet. Additionally, the authors show good humor in adding a document entitled "50 phrases that kill creativity" (p.43). Part I, which deals primarily with establishing a technology plan, is useful in encouraging libraries to identify the components and functions that they want to automate. Part II is also quite strong, spelling out the steps that are needed by libraries to fully select and implement their systems. The section on training does a good job addressing the realistic level of training the library is going to receive from the vendor and what the library is going to have to do for itself. On the whole, the book is very well written and provides numerous examples of forms, tables, and checklists that are crucial for people -- especially those who have not gone through the process before -- to use as they prepare to purchase an Integrated Library System.

While I generally find the book extremely strong and well suited to this audience, I did encounter one section that I could not quite get past. In talking about who will run the system, the authors offer the ideal candidate as a "systems librarian" and provide the skills that he or she would need. I worked with an ILS vendor for two years and ran into numerous types of people who were leaders of automation projects. Many had the role of the head of technical services, and others were subject bibliographers, head reference librarians, and people dedicated to systems. Additionally on the same topic, the description for this "systems" person included some job qualifications that did not necessarily apply to each system (knowledge of UNIX, Structured Query Language (SQL), proxies, etc.). For example, a turnkey solution may not require UNIX expertise because the interface may be menu-driven. Still, this is minor and does not diminish my enthusiasm for the work one bit.

In summary, I find that the book is invaluable for small libraries moving to automation systems for the first time, and of possible use for more experienced libraries migrating to new systems.

Corey Seeman (corey.seeman@utoledo.edu (mailto:corey.seeman@utoledo.edu)) is Assistant Dean for Library Systems at the University Libraries of the University of Toledo.

REVIEW OF: Matthias Felleisen, Robert Bruce Findler, Matthew Flatt, Shriram Krishnamurthi. (2001). How to Design Programs: An Introduction to Programming and Computing. Cambridge, MA: The MIT Press.

by Stef Morrill

In the introduction to *How to Design Programs: An Introduction to Programming and Computing*, the authors explain their reason for writing their book in one, bold-faced sentence:

...everyone should learn to design programs.

As their contribution to this goal, the authors* have written a book that strives to separate programming languages from programming technique, focusing on the design process of programs. They introduce various "design recipes," guides for designing programs, throughout the book. Each step of the design recipe produces a product, allowing students of programming to have their progress checked and to have a tangible product at each phase of the design process. The six basic steps of the design recipe are:

1. the description of the class of problem data;
2. the informal specification of a program's behavior;
3. the illustration of the behavior with examples;
4. the development of a program template or layout;
5. the transformation of the template into a complete definition;
6. the discovery of errors through testing.

The design recipes are expanded and modified throughout the book as the student encounters more complex problems and programming concepts.

In hopes of introducing the concepts of program design without overwhelming the learner with a complex programming language and environment, the authors use a programming language and environment called DrScheme, which they developed themselves after observing beginning students. (Information about DrScheme is available on the book's companion Web site, http://www.htdp.org/ (http://www.htdp.org/).) Based on the Scheme language, DrScheme was designed for the beginning programmer. It is easy to install, with low system requirements. It also has an "Interactions Window" to encourage students to try out new concepts immediately after learning them. The DrScheme language has various "levels" which can be added as the learner progresses. This allows the learner to avoid some mistakes and confusing error messages they would encounter in other environments.

The authors also wanted to avoid the rote memorization of "traditional programming" languages by using a language where students would only have to learn a few things in order to get started. Having worked with a number of traditional programming languages, I found the lack of information and clarification of the DrScheme language a distraction from the design skills, which should have been the focus of the book. Because I did not know DrScheme or any Scheme variation when I began this book, I found myself worrying more about understanding the syntax and flow of the examples than the design recipes and concepts.

This book seems to be written primarily for undergraduate computer science students or high school students, in a classroom environment. Although many books designed for classes can also be used effectively by individual readers, there are difficulties with using this book outside of a classroom:

- Each chapter features exercises to reinforce concepts introduced in the section. However, there are no answers to the exercises. On the book's companion Web site, answers are available, but only to registered teachers using the text in their classroom. Independent learners cannot easily check their understanding of concepts as they progress through the book.
- Many of the examples and exercises focus on geometric and algebraic problems, including calculating areas of geometric shapes (without providing formulas) and problems related to quadratic equations. The assumption seems to be that readers are very familiar with these concepts, and that is most likely true for high school and undergraduate students. However, for those of us who have a few years between us and the time we last solved quadratic equations, this assumption adds another layer of distraction from the stated focus of the book.
- Despite the authors' desire to separate programming language from program design, it is impossible to completely remove the need to learn basic elements of DrScheme to understand and work with this book. The book attempts to mix the DrScheme and design lessons; receiving instruction in the necessary elements of DrScheme during a classroom lecture would clarify the language and allow the student to focus more completely on the design aspects. There is a small DrScheme companion available on the book's Web site, which is not mentioned in the book and which I did not discover immediately. While the companion information is helpful, particularly with the installation and setup of the DrScheme environment, additional classroom instruction would still be desirable.
- There are many cases in the book where examples are unclear, incomplete, or incorrect, and where concepts used in examples and exercises are not introduced until later in the text. Using the book in a classroom setting would allow for additional clarification and explanation, which would help focus the reader on program design.

The purpose and ideals of this book, as explained in the introduction, are pragmatic. The "design recipes" concept would be helpful to many programmers, teaching them to plan and focus their designs and not just begin writing code. However, because this book is tied so closely to the DrScheme language, translation of the concepts to other programming languages is difficult. A book that could outline the "design recipes" concept, explain how these recipes would be modified for more complex programs, and separate this design theory from any specific programming language, would be valuable for everyone who programs.

For example, the book outlines a modification of the design recipe for programs that use compound data. In DrScheme, compound data include structures and lists, but all programming languages work with some type of compound data (lists, arrays, hashes, etc.). A book where the impact of the use of compound data on the design process could be explained, without tying it specifically to DrScheme's lists and structures, could improve all programmers' design process. This "outline" approach could still benefit beginning programmers, allowing them to learn whatever programming language they would choose, but also allowing them to develop good design skills and techniques at the same time. Providing this type of true separation between programming language and program design could help everyone learn to design programs, even those choosing to learn a language other than DrScheme.

**At the time this book was written, Matthew Felleisen, the primary author, was a Professor of Computer Science at Rice University. The secondary authors (Findler, Flatt, and Krishnamurthi) were all graduate students under the advisement of Felleisen.*

Stefanie Morrill (smorrill@scls.lib.wi.us (mailto:smorrill@scls.lib.wi.us)) is the Library Development Coordinator for the South Central Library System in Madison, WI.

REVIEW OF: Brett McLaughlin. (2001). Java & XML (2nd ed.). Sebastopol, CA: O'Reilly & Associates.

by Craig S. Booher

Internet technologies continue to develop at a breakneck pace and Java and Extensible Markup Language (XML) are no exceptions. Within a year after releasing the first edition of this work, McLaughlin is back with a second, more in-depth edition. He's condensed most of the basic material in the first three chapters of the previous edition into a single chapter, reworked the examples, and added many new examples and chapters.

The text can be divided into two parts. The first nine chapters cover XML and the core Java Application Program Interfaces (APIs) for handling XML. Following an introduction, chapter 2 lays down a short course on XML -- markup syntax, Document Type Definitions (DTDs), XML Schema, namespaces, and XML transformations. Don't expect this chapter to make you an XML expert. It does, however, give you enough information to understand the subsequent discussions on manipulating XML files.

The next six chapters discuss three XML manipulation APIs - Simple API for XML (SAX), Document Object Model (DOM), and Java Document Object Model (JDOM). For each, McLaughlin provides an introductory chapter followed by a chapter that deals with more advanced features of the API. The introductory chapters guide the reader through the basics of each API including, SAX content and error handlers, the Document Object Model and Serialization, and JDOM classes and I/O. Here, the reader will also find instructions on where to obtain and how to install such critical Open Source tools as an XML parser, a SAX reader, and a DOM serializer.

The corresponding advanced chapters extend the discussions for each of the APIs. Chapter 4 covers SAX properties and features, five more handlers (EntityResolver, DTDHandler, DefaultHandler, LexicalHandler, and DeclHandler) and filters and writers. The advanced DOM chapter discusses XML modifications, namespaces, DOM Level 2 modules beyond core and, briefly, DOM Level 3 (which was just being developed as this edition was being completed). Finally, chapter 8 reviews the use of factories, wrappers, and decorators in JDOM.

Early in the introductory chapter on JDOM, McLaughlin notes, in the interest of full disclosure, that he is one of the co-creators of JDOM. Despite his acknowledged partiality to JDOM, the author is exemplary in his balanced coverage of all the APIs. There is no hidden agenda in this publication.

A brief chapter on JAXP (Sun's Java API for XML Processing) concludes the discussion of Java APIs for XML. Covering both versions 1.0 and 1.1 of JAXP, this chapter quickly covers the use of JAXP with SAX and DOM.

In the second half of the book, McLaughlin explores some of the more significant applications of XML implemented using Java. Individual chapters are dedicated to Web publishing frameworks, XML-RPC (remote procedure calls), Simple Object Access Protocol (SOAP), Web services, content syndication, and data binding. Each chapter provides a solid introduction to the application, complete with pages of code. It is in these chapters that the reader begins to understand the potential and versatility of XML, especially when it is combined with a Java environment.

The author uses a very informal communication style. He writes as if he is carrying on a conversation with the reader. He reminds the reader if something was covered in a previous chapter and notifies the reader when he is postponing a certain discussion until later in the book. In addition to the text, McLaughlin includes information in sidebars, warnings, and notes. Each of the seven API chapters includes a "Gotcha!" section where the author describes some pitfalls commonly encountered when using the API. To enhance continuity, every chapter concludes with a brief look at what is coming next.

Caveat emptor: This book is not for the Java novice. Early in the book, McLaughlin advises the reader that he assumes "you know the Java language and understand some server-side programming concepts." He moves "rapidly through the basics so the bulk of the book can deal with advanced concepts." Therefore, acquire at least an introductory knowledge of Java before tackling this book.

Fortunately, McLaughlin is not content with simply describing the various technologies in the book. Each chapter provides pages of code to demonstrate the tool or application. Typically, McLaughlin walks the reader through the code, discussing what he wants to accomplish, and then presents a skeleton of the code. Following some deeper explanation of a specific feature he is trying to illustrate, McLaughlin then fleshes out the skeleton, using boldface to highlight the really interesting aspects of the code. Often, he will compare the code for, say, a SAX implementation, to code for a similar implementation using one of the other APIs, e.g., DOM. In this way, the reader gains a better understanding of the programming style and utility for the various tools.

Many of the examples build on earlier chapters. In some cases, this leads to code several pages long. Thankfully, the reader is not required to re-key the code. All examples can be obtained from the O'Reilly Web site associated with the book.

Rapid developments in Java and XML pose a significant challenge to the currency of the book's contents. Standards are continually evolving and the version numbers of parsers and APIs continue to increase. McLaughlin recognizes this and tries to not only discuss current (circa 2001) technologies, but trends and cutting-edge developments. He uses code written to the most stable version available at the time of writing. Nevertheless, as readers move further away from the publication date, they will undoubtedly begin to experience some additional challenges in reproducing the application environments described in the book.

The solution? Why, a third edition, of course. Given the intense interest in these two areas and their ongoing development, don't be surprised to see a third edition of the text by 2003. If it's authored by McLaughlin, be sure to order it.

Craig S. Booher (cjbooher@athenet.net (mailto:cjbooher@athenet.net)) has over 20 years experience as an information professional designing, developing, and implementing information systems in academic libraries and corporate information centers.

REVIEW OF: Michael P. Sauers. (2001). Using Microsoft Outlook: A How-To-Do-It Manual and CD-ROM Tutorial. New York: Neal-Schuman Publishers, Inc.

by Stacey Greenwell

Librarians often prefer Neal-Schuman publications for their clear organization, copious illustrations, and thorough coverage of the subject. *Using Microsoft Outlook: A How-To-Do-It Manual and CD-ROM Tutorial* is no different, as this book covers Outlook from a beginner's perspective (see the chapter 1 subheading, "What is Email?") to more specific topics (see "Customizing My Outlook Today"). More advanced users might choose to skip over the early chapters, but they do provide a good introduction and explanation of electronic mail systems -- the terminology and the basics of how e-mail works. It's entirely possible that the reader, at any skill level, might learn something new from these chapters.

A good way to start any training session is to discuss what is initially displayed on the screen: the toolbars, the menus, and changing the way the information on the screen is displayed (views). While the second chapter is an "Outlook Overview," the different view options aren't discussed much. For example, "Folder List" is not mentioned at all, which would have been particularly helpful for users operating mail programs like Eudora or Outlook Express. However, the Outlook Bar and the different Calendar views are covered nicely in this chapter.

Each component of Outlook 2000 is covered in a lengthy chapter (Email, Calendar, Contacts, Tasks, and Notes). The e-mail section covers the topic fairly thoroughly, from the basics of sending an e-mail to blind carbon copying, creating signatures, and flagging mail for follow-up. The author discusses the important topic of e-mail organization as well, urging users to move beyond the inbox and create folders and organize messages into a hierarchy. The e-mail section also covers creating rules, both basic and complex.

The Calendar section covers most aspects of using the calendar: changing calendar views, creating appointments, adding recurrence to an appointment, setting reminders, and linking appointments with contacts. Changing sorts in contacts can be very useful, and the author spends some time with organization in the Contacts section. While they may be lesser-used components of Outlook (at least for some of us who still adhere to sticky notes), the author covers both the Tasks and Notes components of Outlook in separate chapters.

The book concludes with customizing and searching in Outlook. In addition to the simple search, the author also shows the advanced find, a tool that can be very useful in searching for a specific word or phrase in an e-mail. The chapter also covers customizing the display of Outlook Today.

In addition to the discussion of the basics of each component of Outlook, the author also includes some neat features that might be new to many users. For example, in Contacts you can display the map of an address (by clicking on the Display Map icon in Contacts, your web browser will launch and display the location of the contact using MSN Expedia Maps). In Tasks, you can create a timeline view of task due dates. The appendices also include some helpful tidbits: Outlook shortcut keys (which are also included on the CD-ROM); what Vcards are and how to create one if you wish; and changing Outlook to use Word as its e-mail editor.

Each chapter has several "Hands-On Practice Sessions." The practice sessions assume that you have a PC with Outlook installed (as well as a Web browser for a few of the exercises). The practice sessions are helpful as they take the reader step-by-step through tasks such as sending an e-mail message to multiple recipients or scheduling a monthly recurring appointment. The practice sessions mostly involve creating appointments, contacts, e-mail, etc. A few of the practice sessions do involve settings changes (which makes me nervous when I recommend a book to our library users). However, the settings changes in the practice sessions are minimal changes, such as adjusting how often Outlook checks for new messages. As the reader is prompted to click through different tabs, the author kindly notes to leave most settings alone, e.g. "DO NOT change any of the information in any of the fields on the Servers tab."

At the end of each chapter, the book includes a bulleted summary of major points in the chapter, as well as a quiz over important terms and concepts. Taking the quizzes can help the reader to retain more of the tips from the book. For example, true or false: "You can assign your own WAV sound files to a Calendar reminder." (This is true.)

The book also includes a CD-ROM tutorial. The CD-ROM in my review copy of the book was damaged in the mail, so I requested a new one since I expected the tutorial to be an integral part of the book. I was excited at the thought of a "CD-ROM tutorial;" I expected a step-by-step lesson plan, exercises, instruction through sound files -- something lengthy and full of information, something that a new user could work through step-by-step. I was disappointed to find that the tutorial consists of only three files and a "readme" file. While it was helpful to include these files for the book exercises (a set of driving directions in a text file, a GIF file, a helpful handout of Outlook keyboard shortcuts), this certainly falls short of "a CD-ROM tutorial."

Aside from the disappointing CD-ROM, this book is highly recommended for beginning users of Outlook 2000. The easy-to-read format, generous use of illustrations, and clear writing style make the book ideal for a beginner to learn the ins and outs of Outlook. The book could also be helpful in teaching a class on Outlook, as it could naturally be used to organize a class session. The book is full of useful tips -- while a more advanced user might find most of the information to be a review, it is likely that most anyone could pick up a tip or two from this book.

Stacey Greenwell (staceyg@email.uky.edu (mailto:staceyg@email.uky.edu)) is Desktop Support Librarian for the University of Kentucky Libraries in Lexington, Kentucky.

About TER