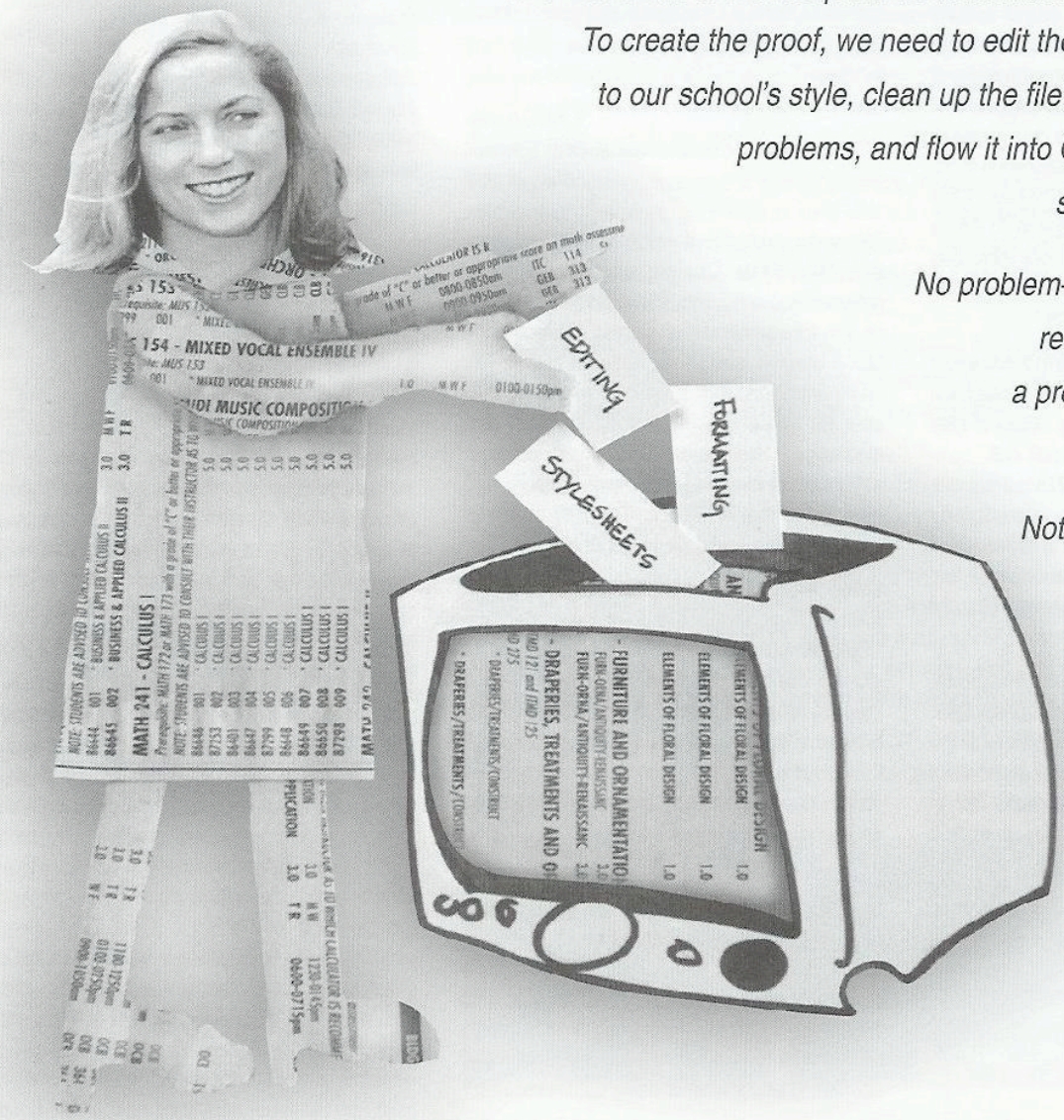
The Economics of Macros

On Friday afternoon, we receive a call from the Office of the Vice President of Instruction. They have a text file of the school's credit class schedule, downloaded from Banner—the school's administrative database—and would like a finished proof for review before the end of the day. To create the proof, we need to edit the document to conform to our school's style, clean up the file's numerous formatting problems, and flow it into QuarkXPress, applying stylesheets throughout.

No problem—just minutes after we receive the call, we have a proof ready for the client.

Sound implausible? Not if you employ macros.



By Dana Martin

Birth of a Macro

By definition, publications departments are constrained by the need to meet deadlines. With short turnaround times, we often focus simply on getting jobs out the door. During a job's production, we don't have time to experiment with new processes, even if such experimentation could result in a better workflow. And once the job is completed, we're on the next project with little time to reflect on the one we've just finished. In my case, I had written a few simple macros before I worked at Johnson County Community College (JCCC), and I knew how helpful they could be. Yet I let three years pass before writing a macro for the credit schedule.

One reason for this delay stemmed from the fact that I was always in a mad rush during production, and once the job was sent to the printer, I didn't want to think about it again. In addition, the task of writing macros for the schedule simply seemed too daunting. First, the download itself was huge, more than 100 pages in length. I knew that to effectively create macros for a job this size, I would have to do a great deal of planning, organizing, and troubleshooting. Also, I knew I would have to review the process of creating macros before I could even get started. Every semester, as my wrist ached from making so many manual changes, I told myself it was easier than writing macros or that I'd get around to the macros "sometime."

That time came last spring, after we received our download as usual, and the designer and I spent about a week working on the first proof. The proof went to the client and came back with more than 1,000 changes marked on it. Somewhere along the line, our process had fallen apart. Information from several academic areas had been entered into Banner incorrectly or not at all, resulting in countless handwritten updates.

We decided to scrap the proof and all our work on it. We asked the academic areas to key their changes directly into Banner, after which we would start over with a new download. By the time we got that download, our print deadline was only a few days away, and the designer and I had to start from scratch on the piece. How easy it would have been if we'd only had macros. We wouldn't have wasted so much time editing and formatting a download that ultimately was discarded, and we wouldn't have had to redo all our work with only days left in our production schedule.

That's when I knew what I had to do. Now, in addition to having fully functional macros for the credit schedule, I have also written macros for another catalog and am in the process of creating them for the continuing education schedule.

Creating a Macro

A macro, short for "macroinstruction," is simply a series of commands grouped under one name. The macros addressed in this article are written in Microsoft Word using Visual Basic for Applications, a component of the Word application. Macros improve the functionality of Word, allowing you to tailor the application to your needs. For instance, you can use macros to combine all your global search and replace commands into one action. As with any find and replace command, you can use macros to search for characters, format symbols, or a combination thereof. You can also use wild cards. Macros can be written to execute other instructions as well, such as saving a file after changes are made. And, rather than writing one monstrous macro for large projects, you can apply a series of smaller macros, which will ease future macro maintenance and keep your macros from becoming unwieldy.

Macros are created in one of two ways: either by recording keystrokes or by working directly in Visual Basic. If you don't want to delve into VBA programming, simply record your macros by choosing **Macro | Record New Macro** from the **Tools** menu. A dialogue box will prompt you to name the macro and specify where you want to save it (on Macs, the global template is the default setting; on PCs, the default is normal.dot). A recording toolbar will appear on your screen. As you execute the commands you want included in your macro, your keystrokes are translated into VBA code. To stop recording, click stop on the recording toolbar. The code is now saved under the assigned macro name. You can retrieve and run the macro whenever you need it by choosing **Tools | Macro | Macros**, selecting the macro you want to use from the **Macro Name** scroll box and clicking **Run**. You can also create toolbars for your macros or assign keyboard commands to execute them.

If you prefer, you can edit the macro in Visual Basic Editor, which allows you to review and modify the code translation of your recording. To edit your macro, select **Macro | Macros** from the **Tools** menu, then highlight the name of the macro you

"By definition, publications departments are constrained by the need to meet deadlines."

"...using macros on the credit class schedule has allowed our office to shave our production time on the credit schedule from more than two months to less than 10 days."

wish to edit and click the **Edit** button in the dialogue box. In Visual Basic, you can simplify and clean up code, as well as remove mistakes. The VBA code is easy to read once you are familiar with it. You should quickly relate the keystrokes you recorded with the resulting VBA translation.

The first macro I run on a document addresses format issues (e.g., two tabs where I only want one). The second addresses style (e.g., 5 p.m. instead of 5:00 p.m.). The third applies tags (e.g., @head:). This macro is the easiest to create and is the most beneficial from a design standpoint. Once I've applied all the macros, I perform the following steps:

1. Save the document as a .txt file.
2. Open QuarkXPress.
3. Select "Get Text."
4. Check "Import Stylesheets" in the dialog box.
5. Select my .txt. file.

In moments, the entire document flows into Quark. It's has been substantially edited, all formatting idiosyncrasies have been eradicated and QuarkXPress has translated the tags I applied into the correct stylesheets.

Macros: The Good, the Bad and the Ugly

Beneath their pretty exteriors, many macros are riddled with ugly little secrets. The macro recorder follows your every keystroke, right or wrong, and interprets your actions to a tee. This specificity can result in the insertion of extraneous or incorrect lines of code. But the real inner beauty of macros is that the code doesn't have to be flawless to be functional. For example, if you include instructions for finding "&" and replacing it with "and," but there are no instances of "&" in your document, the macro's performance will not be compromised. Likewise, the extraneous lines of code the recorder inserts won't harm the macro—but they will slow it down.

Another benefit of macros is their flexibility. You can customize them to your project and work style, use them minimally or extensively, and update them when needed. If you write more than one macro for a document, you can even write a macro that runs all your macros in the desired order. But the most important benefit is that, with repetitive tasks out of the way, macros allow you address more sophisticated editorial and design issues on your documents.

In spite of their near-miraculous attributes, macros do have a downside: they can have a steep learning curve, especially if you aren't familiar with Visual Basic or another programming language. To get started, take a look at books on Word macros and VBA. Next, record a simple macro and build from there. Don't try to write macros for a document when you are short on time. I began working on the macros for the credit class schedule three months before I received the download for the upcoming semester. That relaxed time frame allowed me to experiment with my macros—and make a few mistakes—without the pressure of imminent deadlines. Next, and certainly before you tackle complicated or lengthy documents that require more elaborate macros, create a game plan. Careful planning and logical arrangement of your code will ensure a beneficial long-term relationship between you and your macros. Finally, be ready for experimentation and great deal of trial and error.

Tips for Trouble-free Macros

Always back up your macros. When you write a macro, it will reside in your global template, in a specific template or in a specific document. Back up these template files whenever you modify your macros. If a template becomes corrupt, you will lose the macros stored inside that template. To back up a macro, simply copy the template in which you have created the macro, then save the copy to your hard drive or a network. Naming the template by date will allow you to easily identify the most current backup and revert to an earlier version if necessary. Because macros have become an integral part of my work process and I never want to be left in a lurch if one fails, I save my dated backup files on my hard drive as well as the network, and periodically I burn them to a CD.

Test, test, test. Ever do a search and replace for a term—"and" to "&," for example, only to realize you've replaced every instance of "and" in your document? You now have "l&scape" design classes, a seemingly r&om change you hadn't intended. If you have such results with one search and replace command, imagine the problems that can arise when stringing dozens of commands together. You need to test drive your macro to find and correct these unanticipated results.

To ease the process of testing a new macro, especially on a long document, I turn to Word's **Track Changes** feature. Before testing the results of a macro, I choose **Tools | Track Changes**.

I then select **Track Changes While Editing** from the dialogue box, click on the preferences button and specify that changes be marked in red and replaced copy be hidden. These preferences allow the macro to perform properly without altering its functions. Next, I run the macro I want to test. The red marks that appear in the document allow me to quickly and clearly see successful changes as well as unintended ones. Running a macro while tracking changes also allows you to see what wasn't changed but should have been. You can then return to the macro and review the problem areas in your code.

In the End, What's a Macro's Net Worth?

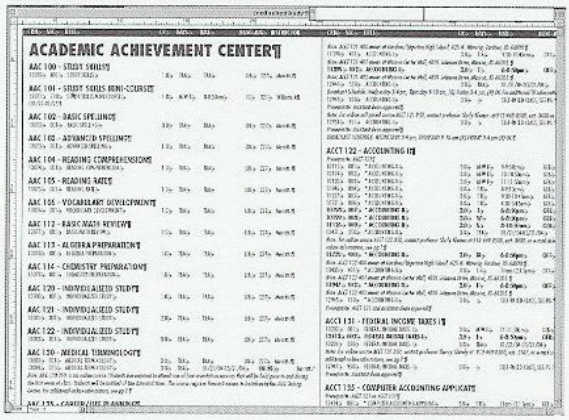
To put all of this in tangible terms, using macros on the credit class schedule has allowed our office to shave our production time on the credit schedule from more than two months to less than 10 days. Within minutes, we can create "throw-away" proofs for our academic departments to review. By "throw-away," I simply mean we don't need or want the proof back. The academic areas can use the proofs to spot problems and correct them directly in Banner. Once these changes are made in Banner, the Information Services department creates a new download, and we can again apply the series of macros to do the lion's share of the work on the document. We no longer spend valuable time with rote editorial and layout tasks. Now, we minimize our time spent on the document until a few days before the file is due at the printer. At that point, I read through the document for clarity, awkward syntax and other problems a macro can't address, and the designer finesses the layout. And best of all, my wrist doesn't hurt anymore.

Suggested Reading

Hart-Davis, Guy. *Word 97 Macro & VBA Handbook.* San Francisco: Sybex, 1997.

Leonhard, Woody, Lee Hudspeth, and T.J. Lee. *Word 97 Macro Annoyances.* Sebastopol, CA: O'Reilly & Associates, 1997.

Roman, Steven. *Writing Word Macros: An Introduction to Programming Word Using VBA.* Sebastopol, CA. O'Reilly & Associates, 1999.



On the left: the downloaded text file for the school's credit schedule.
Above: The file as it appears about four minutes later, after applying our credit schedule macros to the text file.

About the Author

Dana Martin is College Information writer/editor at Johnson County Community College, Overland Park, KS. She has a bachelor's degree in English with a dual major in language and literature, as well as creative writing/ journalism from the University of Missouri-Kansas City.