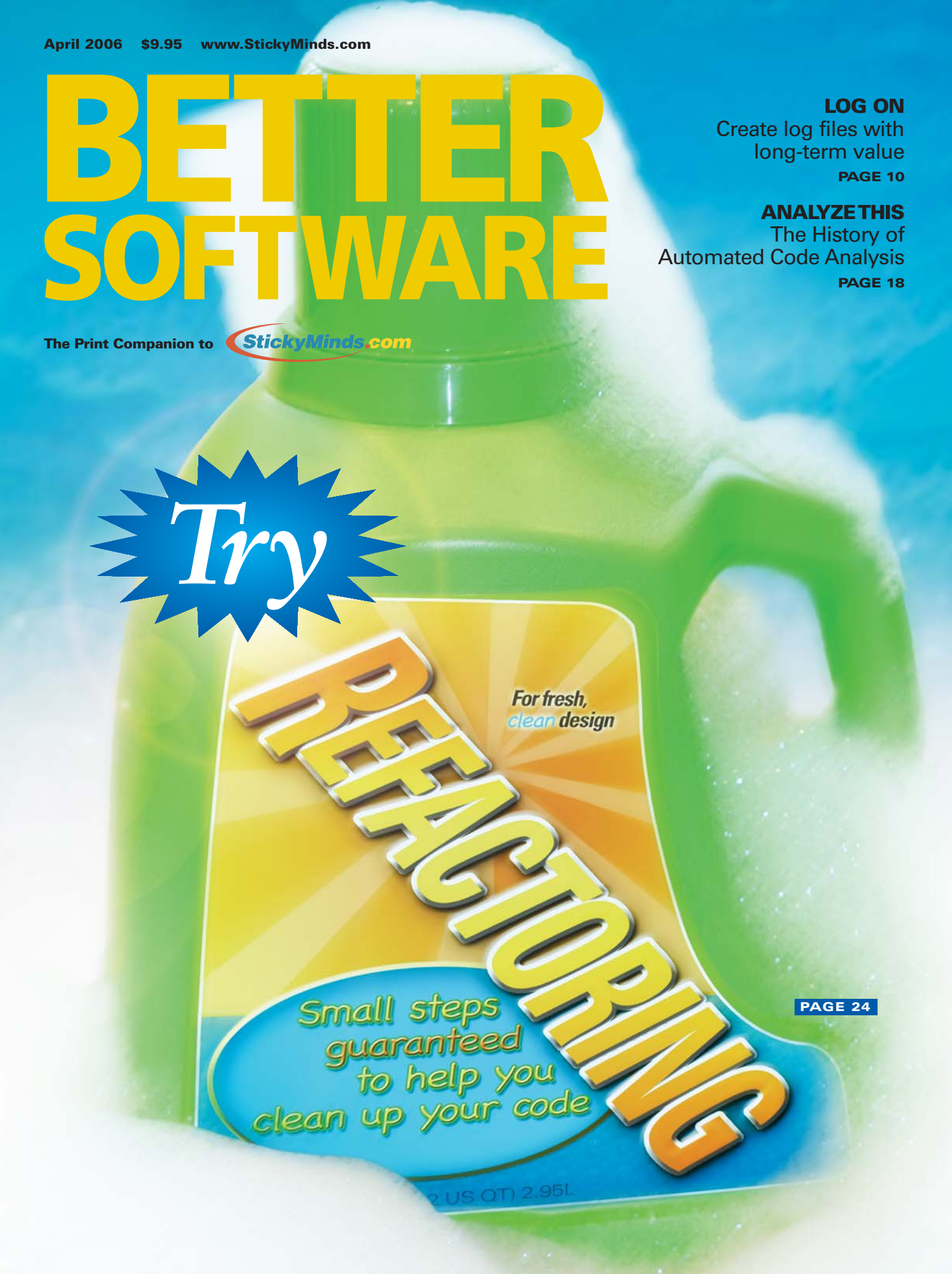
April 2006 $9.95 www.StickyMinds.com

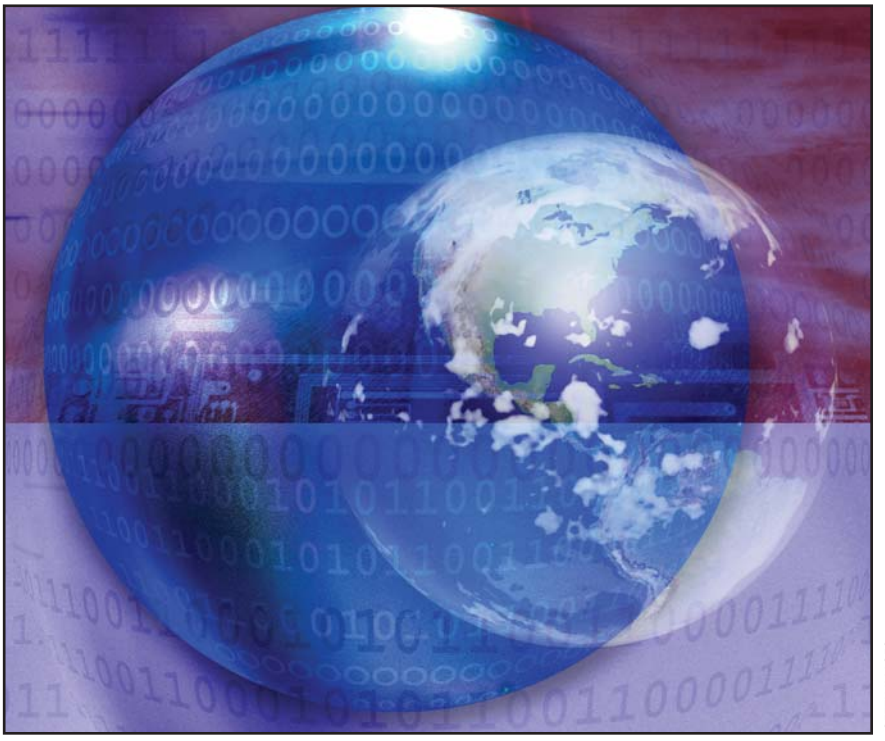
BETTER SOFTWARE

The Print Companion to StickyMinds.com



Where in the World?

by Michael Bolton



GETTY IMAGES

Wherever money and products go, software usually follows closely behind. Your organization's software is more valuable the more places it can go. "Localization" is the generally accepted term for the process of adapting to a new local market; "internationalization" refers to the process of building the product to be adaptable. These tasks have traditionally been difficult, although technologies for internationalization have gradually improved. For example, the Unicode standard for representing international characters and glyphs has been adopted and is increasingly better supported by operating environments and programming tools. Pragmatic development teams now realize that if the product is intended to serve international markets, then text, dialogs, graphics, and the like should be provided as separate resource files—not hard coded within the application—so that they can be translated easily.

To Rapid Testers, these innovations are welcome, but Rapid Testing tries to take a view beyond purely functional considerations by using the word "localizability" as a guideword heuristic. (A heuristic is a fallible method for solving a problem; it serves as a useful stand-in for more thorough and rigorous analysis when time is at a premium.) Our goal is to discover risks associated with the product's operating differently with respect to some distinct locale. We prefer "localizability" to "internationalization," because the new target market may not be in a different country at all. For example, reporting regulations and standards might differ across state lines. Different parts of a country might be in different time zones, and some locations might support daylight-saving time while others do not. Even though it operates within a single country, a Web site, Internet kiosk, or bank machine might provide support for multiple languages. Some local differences are logistical; you can count on the big hardware store to stock fewer snow shovels in Miami than in Minneapolis and to carry a wider variety of seasonal goods in Fairbanks than in Los Angeles.

Does your test strategy account for these kinds of variations? Can the product be reconfigured quickly, easily, and automatically to work in another location? Do your system and end-to-end test scenarios account for different locations and time zones for purchase, billing, and delivery? What if someone buys a product in Reno, Nevada (where there is no sales tax), and returns it just across the state border in California (where there is)?

You might not have considered the possibility that your software or service might be used in another country. In Canada, where I live, there are both federal and provincial retail sales taxes. Tax rates change from province to province, and one province has no retail sales tax. The federal sales tax is sometimes levied separately and sometimes blended with the provincial sales tax in a single line item. That adds complexity, but your product managers may consider the extra work worthwhile depending on the size of the market.

For Web-based applications, look for strings that set locale differences, like `US_en` for "US English" in the URL for GET messages. What happens if you try to change this to another supported language (such as "`CA_fr`" for Canadian French) or to an unsupported locale and language (such as "`XX_zz`")? Does the product recover gracefully if you simply delete elements from the string (such as "`US_e`")?

Not too long ago, I used this approach to find bizarre behavior in an application that had limited local language support. Interestingly, the problem manifested itself most obviously by omitting certain graphics files and corrupting the intended page layout. In that case the consequences were minor, but it prompted me to look for other mishandling of the GET message that could have resulted in security vulnerabilities.

Currency and foreign exchange support

make for some interesting testing challenges related not only to internationalization but also to maintainability and test design. Are currencies formatted properly? If your product deals with multiple currencies, can the product handle updated exchange rates quickly and smoothly? Can your test tools and data files be adapted to deal with constant and instantaneous change?

There are cultural considerations to localization. Holidays and weekends may result in differences in the way business periods and reports are handled. Graphics and icons appropriate for one culture may be meaningless to another. The traditional North American rural mailbox, commonly seen in electronic mail programs, had little relevance for European countries, where door-to-door mail delivery is the norm. Even certain colors and numbers might have cultural significance. The Chinese word for the number four sounds like the word for death and is consequently very inauspicious. If you regard this as trivial, consider whether your customers would happily buy a computer called "G-Death."

There are several ways to display text on GUI systems. The recommended approach is to use operating system calls to display text and to keep that text separate from the graphical elements of the product. Text embedded within a graphic usually makes localization more difficult and more expensive—yet everything on the screen looks graphical to some degree, so how can we recognize this problem? One way is to use a string-dumping program to search for the text within the program's files; if you don't find that text in the string dump, there's an indication that the text might be embedded in a graphic. On Windows systems, we can perform an instant analysis of a screen element by trying to grab text from it (using products like TechSmith's SnagIt, Boilsoft's Resource Hunter, or Microsoft Visual Studio's built-in resource editor). For Rapid Testers, the fact that you *can't* do something with a given tool is often revealing.

If you have access to the source for the program's resources, try replacing all of the product's text strings and captions with the letter Z—you can use a bit of script in a language like Perl, Ruby, or Python to do this quickly—and then rebuild the product. If you find any text other than Z on the screen, you have evidence of hard coding within the application, which makes the product much harder and riskier to translate.

Another approach is to replace strings with Egg Language, in which "egg" is placed before every vowel (theggis eggis eggan eggexeggamplegge); then try using the program, looking for things that appear normal. Egg Language has several virtues. First, it can be encoded easily with a script (and writing such a script is a fine exercise for a tester who is learning automation). Second, it can be decoded (weggith seggome smeggall eggeffeggort) by a human reader, which might be important for navigation if you're not completely familiar with the program. And finally, it lengthens significantly the strings in the program. That's important if the product is to be translated into European languages such as German, where the translated strings will be longer than the English versions and may require more screen space. Non-European languages present more difficult challenges, such as text that displays right-to-left or ideogrammatic script.

Testers are fallible. We may need to consult with experts, since there are too many dimensions to internationalization for non-specialists to recognize every cultural difference. However, we can be aware of the significance of those differences and aware of the potential for trouble if our organization is oblivious to them. Test managers can help by fostering cultural diversity as one of the criteria for a strong test team. Individual testers can help by at least recognizing that things can be different when software goes elsewhere. **{end}**

Michael Bolton lives in Toronto and teaches heuristics and exploratory testing in Canada, the United States, and other countries as part of James Bach's Rapid Software Testing course. He is program chair for the Toronto Association of Software Quality and is a regular columnist for Better Software *magazine. You can contact Michael at mb@developsense.com*

STAR EAST SPEAKER

Don't Stop Now!

Log on to **StickyMinds.com** and join Michael Bolton and your peers in a conversation about this topic. At the end of the digital column, add your views or just read what others have to say.