# It's one thing to say "somebody should do this", but doing it is another matter

devblogs.microsoft.com/oldnewthing/20080124-00

January 24, 2008

Raymond Chen

A common response when I describe a programming error is, ""Programs should have to pass a test that includes testing for this case." The case could be a program mishandling a message, a program responding incorrectly to `IUnknown::QueryInterface`, whatever. But these suggestions fall into the same trap that I see when grading student essays: They're good with the *what* but very weak on the *how*.

Saying "Somebody should do X" is easy, but without some sort of suggestion as to how that could be accomplished, the suggestion rarely gets off the drawing board. It's like saying "Somebody should solve world hunger."

Let's look at that first example again. The topic at hand was window procedures which fail to pass unhandled messages to the `DefWindowProc` function. How would one "test for this"? Would the test walk through every code path in the program that creates a window, and then send each of those windows a fake `WM_QUERYENDSESSION` or a fake `WM_APPCOMMAND` message to see what happens? First of all, it's unclear how a test could exercise all the window-creation code paths of a program without insider knowledge of that program. Therefore, this test would have to be written by the authors of the program.

Next, even if you sent the message and saw that the message was passed to the `DefWindowProc` function, that wouldn't actually prove that the message was handled properly. Maybe the window procedure for a window goes something like this:

```
...
case WM_QUERYENDSESSION:
 if (GetTickCount() / 1000 % 2) return 0;
 return DefWindowProc(hwnd, uMsg, wParam, lParam);
```

Even if you managed to get this window created, if you send it a fake `WM_QUERYENDSESSION`, you'll catch the issue only half the time. It's not enough just to exercise every window procedure; you also have to exercise every code path.

But wait, there's more. What if the program really wanted to prevent the user from logging off?

```
...
case WM_QUERYENDSESSION:
 if (FileHasBeenEditedSinceLastSave()) {
  switch (MessageBox(hwnd,
             TEXT("Save changes before exiting?"),
             TEXT("Title"), MB_YESNOCANCEL)) {
  case IDYES: Save(); break;
  case IDCANCEL: return 0; // user cancelled logoff
  }
 }
 return DefWindowProc(hwnd, uMsg, wParam, lParam);
```

In this case, there is a code path that cancels the logoff, and it is legitimate, since it was done as the result of a user decision. Your test would somehow have to know this and consider that case to be a pass and not a failure. This sort of reasoning is hardly something that a generic test suite can do; it has to be tailored for each program.

It's one thing to say that something should be tested, but without an idea as to *how* it should be tested, the suggestion is much less valuable. I may as well say, "Programs should have to pass a test that verifies that there are no bugs."

Raymond Chen

**Follow**