Fortuity Research Report 3

Programmer's Pride

The current economic downturn has been good news for Fortuity Research's Counselling Section, which has had an unprecedented number of enrollments in their SE2 Course – Self-Esteem for Software Engineers, known to participants and facilitators as *Programmer's Pride*. I spoke to the Department Head, Dr. Amanda Grayle, about the course.

J: So programmers are doing it tough?

AG: Programmers have been doing it tough for a long time. The real heyday of programming was in the late 1980s and early 90s, when a computer program was a long screed of unintelligible instructions, linked together in arcane ways. Writing one was like creating a long romance novel while simultaneously translating it into Urdu. Or, in the case of our programmers who speak Urdu, into Taiwanese.

J: What went wrong?

AG: A whole range of things. First off, there was the move to graphic interfaces spurred by the Macintosh and later Windows. Programmers are trained to care about function, not form – and suddenly they had to spend their time laying out dialog boxes and designing menus. It was very demoralising. I've seen grown men weeping over their pizza, just because they couldn't get their text fields to line up properly.

Then came the rise of modular programming, which meant that a single application was constructed out of tiny independent parts. Programmers had to learn to share their work with each other – that was a real shock after all those years building up a reputation for being surly and uncooperative. Worse still, it grew harder and harder to convince clients that you need to hire a highly-paid programmer just to write a three-line routine that makes the computer go 'boop' when the user presses a button.

J: But you fought back?

AG: Oh, yes. The last twenty years have seen enormous strides towards making programming much more complicated and difficult. There was a threat in the early 90s from Pascal programs, which were so obvious and readable that even a client could understand them. But we responded with C, and soon the danger was past. That taught us an important lesson: whenever a language is in danger of becoming clear and well understood, create a new variant. Thus C has become C++, and Visual Basic, which was once dangerously simple, has spun off into so many different application-based dialects that nobody understands them all any more. The rise of Web programming has created endless possibilities for confusion and bafflement: a single website can easily use up to eight different programming languages -- JavaScript, PHP, Flash, CGI -- you name it.

J: And more recently?

AG: I'm proud to say that it was one of our researchers who found the way to turn one of the strengths of modular programming into a weakness. Older systems used to include everything they needed to create working programs, but new ones come with dozens of separate bits and pieces of code. The programmer now has to link half-a-dozen files together before she can get

the computer to do something as obvious as adding two and two – one file with routines for adding numbers, one file with routines for screen display, one file with routines for keyboard input ... Keep 'upgrading' all the bits every week or two, and the possibilities for obfuscation are enormous.

J: How does the public feel about all this?

AG: Fortunately we don't have much to do with the general public any more, except for providing the programs that add half a million dollars to a gas bill, or cause their traffic lights to turn red for ten minutes when there's nobody coming the other way. If they want a particular kind of program they can usually find one on the Web, and if they need something done regularly they can usually write or record a macro to do it. Our favourite clients are desperate clueless executives with a looming deadline.

J: But I gather they have been troublesome lately too.

AG: Yes, there's a real problem on the horizon. You see, all our attempts to complicate and obfuscate programming are based on the claim that complicated programming makes computers run more quickly. This was an easy line to take when people were used to waiting overnight for their data to be processed, but the hardware now is much faster. It's hard to explain why it's necessary to put in six hours of programming in order to gain a quarter of a second, when the whole program only runs for a minute and a half. We need to find some other excuse – er, I mean rationale. 'Security' is a promising one. You can get away with anything if you call it a security feature.

J. Tell me about the course.

AG: We teach software engineers to stand up for themselves. We draw our facilitators from a range of oppressed minorities – environmentalists, authors, feminists, right-wing think tanks and so on – and run small-group training sessions to teach programmers how to be angry, offended and self-righteous. They're generally calm and sensible people, so it takes a while for them to pick up the language and gestures, but we've had some good results.

Then we move on to role-playing. One of our facilitators plays a chief executive with a bizarre list of contradictory specifications. We train the programmers not to try and talk them out of it, but to accept the whole thing and set a ridiculous completion date by which time the CEO will have been sacked anyway. Points are awarded for how long the programmer can go on without giggling.

We do have a third session planned. We hope to retrain our programmers so that they can move into another highly-paid profession where success depends on accurate reasoning and calm logical thought. The problem is that we can't find one. I don't suppose you know ...?

J: No, sorry. Now, what if our readers wanted to learn programming?

AG: They have a wide range of options. All the standard MS Office programs now have a macro language which supports standard programming features -- conditionals, loops, variables and so on -- with a fairly user-friendly environment in which to create and debug the code. For GUI programming that involves form design, Microsoft Access, with a built-in version of Visual Basic, is a good place to start. Users wanting to try a traditional programming approach can download and install a free version of the veteran QBasic called QB64, which is available for both Linux and Windows: see http://www.qb64.net for details.

For stand-alone object-oriented programming, Visual Basic itself is available for Windows in a free 'Express' edition. You can download it from Microsoft via http://www.microsoft.com/ex-Press/download. A similar program for Linux users is called Gambas -- which is Spanish for 'prawn'. Gambas is available from several sources, depending on their version of Linux.

J: And what other projects do you have in hand?

AG: One of our researchers has obtained a government grant to investigate why people want eBooks to be more like books on paper. For instance, the Internet Archive has a new system called 'Flip-book' which stitches the two sides of a scanned double-page spread together and displays them on the screen. (See http://www.archive.org/stream/library01dobsgoog for an example – J.) He's trying to find out why they think it helps to depict the right-hand page curling up and swooping over, like a real page. Who exactly do they think they're fooling?

If eBook developers can be believed, people are hopelessly nostalgic for the good old days. It's the same kind of thinking that results in the Smell of Books sprays (http://smellofbooks.com – *though I was a little wary of that after checking the date – J*) and the offer by CafeScribe in 2007 to send out scratch-and-sniff old book smell stickers to their electronic readers (http://tinyurl.-com/coencc).

Research by Fortuity is still in the early stages, but our Dr. Whiffle has already patented several inventions based on the same principle. For instance, we're offering users a set of curtains, mounted on the front of a television set, which will open automatically when a program begins and close when it finishes. An underlay attached to the screen filters out all the colour, so viewers can enjoy the delights of black-and-white. Our digital radios now supply bursts of static and an occasional high-pitched whine for your listening pleasure, and we're working on software that will allow mobile phone users to communicate with each other in real time by pressing down a single key in one of two different ways – we call them a 'dash' or a 'dot'. In the race to the past, Fortuity Labs is a market leader!

J: Thank you, Amanda Grayle.

AG: You're welcome.

April 15, 2009