The Embedded Muse 119

Editor: Jack Ganssle (jack@ganssle.com) Nov. 4, 2005

EDITOR: Jack Ganssle, jack@ganssle.com

CONTENTS:

Editor's Notes

Do you want to increase your team's productivity? Reduce bugs? Meet deadlines? Take my one day Better Firmware Faster seminar. You'll learn to thwart schedule-killing bugs, manage reuse, build predictable real-time code, better ways to deal with uniquely embedded problems like reentrancy, heaps, stacks, and much, much more.

I'm presenting this on two dates:
- Chicago, IL on December 5
- Irvine, CA on December 7

Want to be your company's embedded guru? Join us! There's more info at http://www.ganssle.com/classes.htm , including cheap flights to these cities from around the USA.

If your outfit has a dozen or more engineers who can benefit from this training I can present the seminar on-site. See http://www.ganssle.com/classes.htm .

After getting many requests I've decided to start a blog. It'll contain my thoughts and observations about all sorts of things, and won't exclusively be about embedded systems. See http://jackganssle.blogspot.com .

How Much Software can be Developed in a Year?

QSM, a company that maintains a vast database about software projects, looked into how much code we can develop in a year. See http://www.qsm.com/Develop_12%20months.pdf for the full report.

Only about 10% of projects they studied delivered more than 75,000 SLOC in 12 months. But those efforts spent a pile of money cranking so much code so fast. These big projects burned 117% to 419% more cash than a similar 139,000 SLOC project delivered over a longer period. Fast is expensive.

About 60% of the projects shipped under 10,000 SLOC in a year.

The report concludes that the biggest program practically possible in 12 months runs about 180,000 SLOC, and will use a team of 70 to 100 people. It will cost two to four times more than the same project with a more relaxed schedule.

That works out to 150 to 214 SLOC per programmer per month.

Call for Articles for Computer

Rick Schrenker is a frequent correspondent who is working on an special issue of Computer magazine about the safe use of embedded systems in hospital settings, and is looking for contributors. Here's the solicitation:

IEEE Computer seeks articles for a special issue to appear in April 2006 on the software engineering and application of software-based medical devices and device systems. The guest editor is Rick Schrenker, Department of Biomedical Engineering, Massachusetts General Hospital; http://biomed.partners.org. The Institute of Medicine's 1999 report, To Err Is Human, estimated that as many as 98,000 Americans were dying annually as a result of medical errors. The section titled "Why Do Accidents Happen?" states "People ... become accustomed to design defects and learn to work around them, so often they are not recognized" and "Accidents are more likely to happen in certain types of systems. When they do occur, they represent failures in the way systems are designed. The primary objective of systems design ought to be to make it difficult for accidents and errors to occur and minimize damage if they do occur." Given the hundreds to thousands of technically and clinically heterogeneous medical equipment models (and their attendant software) used in large medical facilities, the increasing complexity facing medical device users and the clinical engineering community is almost overwhelming.

This special issue looks at the challenges facing the medical industry as a whole, and the clinical engineering community in particular, in realizing the promise of modern medical technology. Topics of interest include

* The application of software engineering in the development of systems that span the spectrum of care-for example, home, community, community hospital, academic medical center, and regional health care systems;

* The application of software engineering in the development of systems in high-intensity care-delivery settings-for example, intensive care units or operating rooms, where various clinically and technically heterogeneous devices and systems can be applied to provide patient care, often in concert; and

* Lifecycle maintenance, management, and safety assurance of software-based medical devices and systems, duly considering both the care environment and the technological sophistication of users and potential interactions with coresident devices.

The submission deadline is 1 December 2005. Author guidelines and submission instructions are available at www.computer.org/portal/pages/computer/mc/author.html. Send inquiries to the guest editor at raschrenker@partners.org.

Bill's Lesson

Sharon Foster has kindly contributed another piece:

I was working at a company named T-bar, which made data communications equipment for switching mainframes and peripherals. They had been building electro-mechanical systems for many years, and had recently developed an automated system using the Intel blue-box. The initial software development for the automated switch had been done by an outside consultant, but now the time had come--as it does in the life of any truly useful product--to add some new features, and I had been hired to do that as their first full-time software engineer.

My boss was Bill Landesberg, an electrical engineer who had been involved in some very early developments in the computer field, including mercury delay line computer memory. (You can Google it.) Bill had no software background--which fact he repeatedly reminded me of, sometimes to the point of annoyance. (He also had some very off-color jokes in his vast repertoire, but that's a story for another time.)

My previous experience had been with the PDP-11 family of minicomputers in an R&D environment, so Intel, 8 floppies, PL/M, and commercial products were all new to me. I

was very proud of myself when one day I called my boss into the lab that was also my office, to show him I had implemented the first of several enhancements he had requested. This particular enhancement had to do with giving the operator the ability to monitor or test any of the communications paths, and so Bill dutifully typed first the MON: command and then the TST: command at the command prompt. Everything worked as planned, and I was feeling very confident and ready to humbly accept an AttaGirl!

But that's not what came next. Bill then proceeded to type MON: followed by a string of random characters having nothing to do with the very rigid and well-defined syntax of the command. The system crashed. Bill just shrugged, waved his hands over the machine and said simply, Fix it.

"But but," I sputtered, "You weren't supposed to type that!"

"But I did," he replied, and walked away, shaking his head.

And so it came to pass that an electrical engineer who knew nothing about software taught me the most important lesson I have ever learned, either before or since, about software development. It was Murphy's Law in its most immediate and concrete form. Users will make mistakes, and the software has to protect itself and the hardware--and the user--from those mistakes. It's a lesson that also applies to interfaces between any software components, as well as to interfaces between software systems and human beings. In this case it was not a safety- or mission-critical system, but in the years to come I did have the opportunity to develop and test such systems, and Bill's lesson has been with me every day.

Comment from Jack: 50 years of software engineering has taught us not to expect precision from users. We *know* we have to accept bad inputs, and to handle boundary conditions in functions.

But for some reason too many of us ignore these lessons.

Windows Turns 20

Microsoft Windows is 20 years old. See http://www.pcmag.com/article2/0,1895,1868435,00.asp for a history of the OS.

When Apple introduced the first Mac I went to the local computer store to see this wonder that the world was marveling about. The mouse metaphor was confusing at first – what do you do when the mouse runs off the edge of the desk? But I was captivated and immediately bought one of the first units.

Then Microsoft released Windows 1.0, a bulky, clunky environment that required an ungodly 256KB (that's K, not M) if RAM. Thinking that it might bring a Mac-like environment to the PC I ordered a copy… and found it to be totally unusable. Frankly, the early Mac was almost as bad; to do anything you tediously swapped floppy disks in and out of the drive. But Windows 1.0 was much worse.

Windows 2.0 was, well, nearly as bad. 3.0 Came out and we saw a glimmer of hope. Only with the introduction of Windows 3.1 did Microsoft finally have a more or less useable GUI. And 3.11 brought us networking, in an era when no one was sure if Ethernet, Arcnet, or one of a number of other approaches would dominate.

All these years later any desktop app, let alone an entire OS, that could fit into 256KB of RAM seems quaint. My Windows XP desktop has 400 GB of disk space and a gig of RAM. And it's still not fast enough, too often slowed by monstrous programs that consume every resource available.

I didn't have gray hair before Windows. Maybe those of us middle-aged engineers who spent years of dealing with crashes and control-alt-deletes could pursue a class-action suit against Microsoft for aging us prematurely! But they finally did get Windows more or less right. I use it – and Linux – every day, and benefit from both the capabilities of both.

It'll be interesting to see what *useful* features Vista (once known as Longhorn) brings. And even more intriguing to see how much memory and disk space it requires.

Jobs!

Let me know if you're hiring firmware or embedded designers. I'll continue to run notices for embedded developers as long as the job situation stays in the dumper. No recruiters please.

Microwave Networks, Inc., a leading designer of point-to-point microwave radio products, is currently looking for an embedded software engineer. This engineer will be working on our next-generation microwave radio products. We need someone with 3+ years of embedded programming, with special emphasis on commercial RTOS (we use Nucleus, but any of the more well-known RTOS's is sufficient), TCP/IP programming (telnet, SNMP, TFTP, etc. - data encryption a plus), and basic communications theory. http://www.microwavenetworks.com/employment.htm to apply.

Do you want to work with a global leader? If you do, then bSQUARE wants to talk to you today. bSQUARE is the leading system integrator in the Windows Embedded Market

focused on helping Tier 1 thru Tier 3 device makers bring their products to life! bSQUARE works with Windows Mobile, Windows CE, and Windows XPE. The smart device market is growing rapidly and bSQUARE is growing strong in this increased demand for embedded systems service and solutions.

We are looking for Platform Engineers who will be responsible for the design and development of drivers and system-level OS components for adaptations of Windows Embedded (CE, XP) to various platforms. We are also on the lookout for good Applications Engineers that will design Architecture and develop Win32-based apps for WinCE based platforms. Email your resume for immediate consideration to kimberlyh@bsquare.com. bSQUARE has offices in Bellevue Washington, Akron Ohio, San Diego, and Taiwan. Check us out at www.bsquare.com.

Joke for the Week

Catherine French contributed this one:

1. When computing, whatever happens, behave as though you meant it to happen.

2. When you get to the point where you really understand your computer, it's probably obsolete.

3. The first place to look for information is in the section of the manual where you least expect to find it.

4. When the going gets tough, upgrade.

5. For every action, there is an equal and opposite malfunction.

6. To err is human . . . to blame your computer for your mistakes is not even more human, it is downright natural.

7. He who laughs last probably made a back-up.

8. If at first you do not succeed, blame your computer.

9. A complex system that does not work is invariably found to have evolved from a simpler system that worked perfectly.

10. The number one cause of computer problems is computer solutions.

About The Embedded Muse

The Embedded Muse is an occasional newsletter sent via email by Jack Ganssle. Send complaints, comments, and contributions to him at jack@ganssle.com.

To subscribe, send a message to majordomo@ganssle.com, with the words "subscribe embedded *your-email-address*" in the body. To unsubscribe, change the message to "unsubscribe embedded *your-email-address*".

The Embedded Muse is supported by The Ganssle Group, whose mission is to help embedded folks get better products to market faster. We offer seminars at your site offering hard-hitting ideas - and action - you can take now to ***improve firmware quality and decrease development time***. Contact us at info@ganssle.com for more information.