The Embedded Muse 174

 February 2, 2009

EDITOR: Jack Ganssle (jack@ganssle.com)

CONTENTS:

Editor's Notes

Did you know it IS possible to create accurate schedules? Or that most projects consume 50% of the development time in debug and test, and that it's not hard to slash that number drastically? Or that we know how to manage the quantitative relationship between complexity and bugs? Learn this and far more at my Better Firmware Faster class, presented at YOUR facility. See http://www.ganssle.com/classes.htm

Brazil! The country of samba, feijoada and caipirinha. And, of course, embedded systems. I'll present a two-day version of my Better Firmware Faster seminar in Sao Paulo March 25th and 26th. See http://workshop.embarcados.com.br for more information.

How about a two-fer? I'm presenting a public version of the Better Firmware Faster class April 3 in San Jose, CA. Come to the Embedded Systems Conference that week AND join me for a day to learn my approach to crafting world-class systems in less time. See http://www.ganssle.com/classes.htm for more info.

Mike Barr is looking for Bad Code to illustrate some concepts in an upcoming class. He'll remove company-specific references. Does anyone have examples of really awful code you'd like to share? Send it to me and I'll pass it along to him.

I'll be speaking at the Deep Agile 2009 event in Cambridge, MA April 25-26. For more info see www.agilebazaar.org.

Quotes and Thoughts

Management must define policies and expectations, which should be measurable. Policy is governance and therefore must be enforceable, automatically if possible, and tied to a business objective. Define a standard for acceptable performance and measure it. Process then enforces policy.

Book Review

The oldest known book about engineering is the 2000 year old work "De Architectura" by Marcus Vitruvius Pollio. One historian said of Vitruvius and his book: "He writes in atrocious Latin, but he knows his business". Another commented: "He has all the marks of one unused to composition, to whom writing is a painful task".

Does that sound like the last ten technical books you've read?

Engineers are famous for being very bright, but also for lacking basic writing skills. Yet writing is still our primary means of communication, so we buy heavy tomes created without the benefit of basic grammar and often bereft of a coherent structure. Storyline? Character development? Forget it.

Lisa Simone's If I Only Changed the Software, Why is the Phone on Fire (http://www.amazon.com/Only-Changed-Software-Phone-Fire/dp/0750682183/ref=sr_1_1?ie=UTF8&s=books&qid=1228408672&sr=1-1) isn't the usual dreary work stuffed with arcane wisdom buried beneath paragraph-length sentences seemingly written by someone just starting with English as a second language. This is certainly the first embedded book with characters. The first with action, and with interesting and cool stories.

Bad code that makes a phone burst into flames?

What fun!

This is a James Patterson-style fast-paced book with dialog as close to gripping as one can imagine for a computer book. Its uniquely-embedded focus twists together elements

of hardware and software as we engineers do in our daily design activities. One can't be understood without the other. Code makes the hardware smoke. That's unheard of anywhere but in the embedded industry.

Lisa weaves stories around deep technical issues. She's teaching the way humans have learned for 10,000 years. Most of us fought off sleep with varying levels of success in high school history classes. Who really wants to memorize the date of the First Defenestration of Prague or the name of Polk's vice president?

Yet now as adults we eagerly consume historical fiction (like James Michener) and real history assembled as a story (consider David McCullough). Cro-Magnon Grog taught his sons to avoid poisonous berries by telling them of uncles who died; the Old Testament was passed down orally as a collection of stories rather than a recitation of facts. Not properly casting an unsigned char sounds pretty dull, but when captured as a story, the interaction of people puzzling out a problem in a real-life setting we all identify with, we're engaged and learn the important lessons better.

Lisa shows how people are part of the solution and part of the problem. The concept draws on an oft-neglected axiom of the agile methods: people over process.

Despite the stories and character development, this is a textbook of a sort. There's homework. When Lisa asks you to stop and answer a question, do so! Think. Reflect. Surely Grog asked his sons questions to make them consider the lessons he imparted. We learn best by such interaction. Readers of Watts Humphrey's brilliant yet ineffably dull "A Discipline for Software Engineering" either do the homework and see their skills skyrocket; or read the book, skip the homework, and get no benefit at all.

Buried under the lessons Lisa derives an important zeitgeist, a design pattern if you will, that should guide us in our work. It's one of creating readable work products: use cases, comments, requirement documents, and more. Though we need not emulate her use of story development in writing a report, we should and must abandon our traditional use of tortured English. Write interesting documents. Be lively and engaging. After 2000 years it's time to leave Pollio's legacy behind and realize that if our readers are confused, frustrated or bored by what we produce, we're history.

There's more on her blog at http://www.lisaksimone.com/phoneonfire .

Responses to Computer Science Education

Ralph Hempel wrote: "I'm really interested in your short take on CS Education, and of course I agree with you. I graduated just over 20 years ago and now I'm looking at

Universities with my boys as they are getting ready to select their future paths.

“I have also started to think about CS education some more, and been involved with LEGO MINDSTORMS Robotics. I wrote to you years ago about the LEGO RCX and the Forth that I wrote for it.

“Well, the next generation NXT came out and I was one of four community members that were selected by LEGO to help with some of the later design and test stages. We were featured in Wired Magazine.

“Long story short, I scrapped the LabView based GUI environment and ported the Lua language to the NXT, and the result is a full interpreter/compiler that runs right on the brick: http://www.hempeldesigngroup.com/lego/pblua .

“It was a good exercise in using a lot of software that was already written and tested and combining them with some glue and test code.

“The NXT makes for a really fun way for young programmers to learn about embedded systems, as there are interrupts, motor outputs, sensors, an LCD, buttons, and even shaft encoders for motor position.

“I sure wish a standard kit of parts like that was available 20 years ago :-)

“I'm looking to make ties with companies and/or universities or schools that are interested in developing learning materials based on the NXT kits. Contact me at rhempel@hempeldesigngroup.com .”

Lisa Simone (whose book is reviewed above) also had some comments: “Boy, you hit the nerve again with the computer science education. As an EE who spent many years cranking C embedded code and inheriting embedded messes, I'm right there with ya.

“But it's not just the CS education - it's pretty universal in my experience. And I see it much more clearly having worked both sides of the fence - designing embedded systems in industry, and then teaching engineering design and embedded systems development at a university. The chasm between the two is huge, and very few in the educational system seem to 1) recognize it and 2) do anything about it (or even know what to do about it). Industry is not completely free of blame either.

“I've introduced new engineers into product development teams and found most were eager but unsure how to start solving problems. They just get assigned that first bug and their professional lives begin. It took a while for me to figure out how to train them at the same time we were insanely trying to deliver very late products. They could code, but

often got off track in the big picture. In the face of emergencies, their on-the-job-training was Trial by Fire (as was mine years ago!). Not good.

“Based on my industry experience, I was brought in to completely revamp an engineering design program at a university. I was surprised (and saddened) at the ad hoc nature of the program. It was not based on any standard design process (no customer needs, requirements, specs, formal testing, etc.) and resulted in senior projects that weren't fulfilling for the students and probably were unimpressive to potential employers. The chasm was very clear to both sides, but neither knew how to address it. But amazingly, the administration recognized their lack of experience in this area, and allowed me complete control in the program redesign. (Cool!) I had a blast rolling out my redesign and giving the students a reality check - seeing that deer-in-the-headlights response from the beginning (students late for class on the first days were greeted with a loud and show stopping "You're Late for Work!" And the concept of limited Sick Days. Weekly written and face-to-face status reviews. Accountability to teammates.).And what's incredibly satisfying for me is students contacting me after graduation to relay that yes, all that documentation and "thinking before doing" really helped them in the Real World. Many universities are starting to actively address this with programs that bring education and industry together, but they are still few in number. We need more of this. It isn't' free to accomplish this, but the government provides grants to help.

“As you mentioned, the thought process is more important than the coding itself. For embedded systems, I also see the focus on higher level methods, without exposure to "coding at the metal." I created an embedded systems intro class that did just that - starting at the hardware, turning on LEDs, thermal control. Thinking before coding. Not allowing hundreds of mindless recompiles to "see if that worked.” Students from other engineering disciplines left the higher level embedded classes to take mine because it was all hands-on. Practical - I don't say this to elevate myself, but as an example that even the students THEMSELVES recognized that they needed something different, something more practical in preparation for the Real World. Something they grasp as truly useful.

“Having see the chasm on all sides - industry's frustration, academias' misunderstanding of industry needs, and even the students and young engineers recognition that what they've been taught is just not quite right.

“That, in my mind, is just as important as knowing the engineering design process coming out of school. Getting a better understanding of needs and the huge disconnect between these three players. From this comes a better curriculum that will naturally product better thinkers who don't rush to code and compile. And I suspect that industry would welcome these better prepared new engineers. What an opportunity we should be pursuing!”

“Below was an incredible day for my embedded students - they did a thermal control

system BY THEMSELVES! They were so proud that they wrote the code, had it pass my approval using standard coding templates, and then tested it. How better to learn than to have fun doing it? . The course is described here: http://www.lisaksimone.com/embedded-systems-university-course .”

The article pushed one of Geoff Patch’s buttons:

“> Yet half or more of the firmware developers I meet have EE degrees.
“> It's odd that these experts at transistor theory, electromagnetics,
“> and IC design spend their career cranking C and C++ code.

“It is indeed, and I think it's an astonishing misuse of hard earned expertise...like using a screwdriver to hammer in a bolt! We have avoided this path, so with one exception all of our software developers have degrees in computer science or software engineering. We believe that a software engineer with 6 months hardware training will produce better software than a hardware engineer with 6 months software training. Therefore, we employ software specialists to develop software, and hardware specialists to develop hardware, and we find that our staff are happy doing what they've been trained to do.

“> EEs do learn an awful lot of software in college. But only in the
“> worst possible way.

“Yes!!!!!

“> Few schools stress software engineering; most
“> teach people to write programs. "Here's the fundamentals of the
“> language, now write a program to do this." Process is almost unheard
“> of. Nearly none know about inspections, PSP, CMM, XP, or any other
“> disciplined development strategy.

“I could provide a learned dissertation on this topic. Such people are like someone who knows the legal moves of individual chess pieces, but can't coordinate the moves in order to properly play the game of chess. We have interviewed EEs for software positions, and they generally know the rudiments of C or C++, but have no clue about how to *engineer* a large piece of software through the use of modern, disciplined software development processes. I have a good knowledge of digital hardware, but as I am a software specialist I would defer to an EE regarding the design of a new board. Unfortunately, many EEs appear unwilling to show a similar level of respect for skills in software development. "It's only software...".

“Interestingly, a number of Australian universities are now offering excellent formal degrees in software engineering. Many of my staff have come through these courses and

they are very good at their jobs.

"> My dream is for the new grad of the future, EE, CS or CE, to come out
"> of school expecting businesses to employ some sort of disciplined
"> development process.

"> Let's hope they
"> demand real design, and they avoid plunging into coding too early.

"This is how it is for us right now, and it works. The reason we are able to attract and *retain* high quality staff is because we do The Right Thing. We use our development processes as a major selling point when we are interviewing potential employees, and the candidates look on this favourably. Those organisations that still engage in garage hacking appear to make rapid progress on the surface, but in the long term they lose out through longer development times, higher defect rates, staff turnover etc."

Open Offices

James Grenning had some comments on open offices. He's a well-known agile advocate, and a signer of the Agile Manifesto. He'll be speaking with me at the Deep Agile event mentioned earlier: "The link about open offices does not provide much detail behind the studies. I suspect there are some very bad open office situations and some very good ones. I've seen and helped setup some very good ones.

"The open workspace is one of the practices Agile development toolkit. Teams working in open workspaces find it productive and fun. The best situation I've seen regarding open workspaces is to have a war room for collaborative activities and private smaller workspaces for individual work. The walls in the war room radiate information like design ideas, project goals, metrics, work in progress, and progress against the plan. A big table with high powered development machines is central to the workspace. The space is not mandated but is an opportunity to collaborate. Once people get used to it they gravitate to the open workspace. Most of the product development work is done there where communications are natural and efficient. But individuals still have a place to retreat when solo time is needed.

"We're seeing the overhead to communication drop dramatically. Communication happens through osmosis, rather than through formal documents and reviews. People have studied open offices for teamwork too and found it to be valuable. Alistair Cockburn has done some very interesting work in this area.

"In reflecting back on things that have made for successful projects and products during my career, one thing that comes to mind is effective communications. When projects got

in trouble, we cranked up the communications by meeting daily and working together in the lab, someone's office or a war room. The most effective meetings were ones that just happened when the time was right, no waiting, no scheduling nightmare. The agile approach uses these highly effective communications strategies throughout the project not just when the project is in trouble. They can prevent many communications problems.

“More and more the things we engineers are trying to create require a team, and teamwork. Don't let this vague condemnation make you close your door to collaborating and creating great products.”

Jobs!

Let me know if you’re hiring firmware or embedded designers. No recruiters please, and I reserve the right to edit ads to fit the format and intents of this newsletter. Please keep it to 100 words.

Wow – two weeks in a row with none. How is the recession affecting your company?

Joke for the Week

From Tjark van Dijk:

In the beginning, God created the bit. And the bit was a zero.

On the first day, he toggled the 0 to 1, and the Universe was.

On the second day, God's boss wanted a demo, and tried to read the bit. This being volatile memory, the bit reverted to a 0. And the universe wasn't. God learned the importance of backups and memory refresh, and spent the rest of the day reinstalling the universe.

On the third day, the bit cried "Oh, Lord! If you exist, give me a sign!" And God created rev 2.0 of the bit, even better than the original prototype. Those in Universe Marketing immediately realized that "new and improved" wouldn't do justice to such a grand and glorious creation. And so it was dubbed the Most Significant Bit. Many bits followed, but only one was so honored.

On the fourth day, God created a simple ALU with 'add' and 'logical shift' instructions. And the original bit discovered that – by performing a single shift instruction -- it could become the Most Significant Bit. And God realized the importance of computer security.

On the fifth day, God created the first mid-life kicker, rev 2.0 of the ALU, with wonderful features, and said "Forget that add and shift stuff. Go forth and multiply." And God saw that it was good.

On the sixth day, God invented pipelines, register hazards, optimizing compilers, crosstalk, restartable instructions, microinterrupts, race conditions, and propagation delays. Historians have used this to convincingly argue that the sixth day must have been a Monday.

On the seventh day, an engineering change introduced Windows into the Universe, and it hasn't worked right since.

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.

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.