The Embedded Muse 94

Editor: Jack Ganssle (jack@ganssle.com) March 12, 2004

EDITOR: Jack Ganssle, jack@ganssle.com

CONTENTS:

Editor’s Notes

The Embedded Systems Conference is this month in San Francisco. I’ll be teaching a class about Managing Embedded Projects on Monday March 29th. Tuesday I’ll moderate a Shop Talk discussion called “The Future of Engineering in an Outsourcing World” (from 7:15 AM to 8:15). The conference is always a fun time; do come by and say “hi”.

Thanks to all who voted for a venue for the next public Better Firmware Faster seminar. Chicago won overwhelmingly; it doesn’t hurt that it’s cheap to get to the city from most of the USA. The seminar’s date is May 17. Seating will be limited so sign up early. See http://www.ganssle.com for more details.

Looking for work? Definitely check out http://www.asktheheadhunter.com/newsletter/OE20030617.htm. Nick Corcodilos explodes myths around resume factories like monster.com. It seems only 3.6% of all hires are made through monster.com; all of the other sites fair much worse.

The article is an interesting and revealing read. I’ve always advised people to never abdicate their job hunt to headhunters or automated web sites. Though these might be useful adjuncts to a search, the wise job seeker takes control of the hunt and is always actively involved. It’s just another job to the headhunter… but it’s your career.

Feature Driven Development

The last decade or so has been an exciting time to be in software development. Hardware design has, in my opinion, lost some of the fun now that ICs are so dense and speeds so high. But the software world has been flooded with new ideas and methodologies. Some are brilliant, others goofy, but all are fascinating.

One is Feature-Driven Development (FDD). Do read "A Practical Guide to Feature-Driven Development", by Stephen Palmer and John Felsing (ISBN 0-13-067615-2, Prentice Hall, 2002), which is a readable treatise on this important topic.

Feature-Driven Development (FDD) is a relatively agile development methodology that is, in my opinion, much more suited to most embedded efforts than techniques like eXtreme Programming. XP is an interesting idea with lots of fabulous concepts we should steal, but I'm concerned about how XP shortchanges design. FDD requires considerable initial design, yet preserves much agility in the feature implementation phase.

As an aside, a new article (http://www.stsc.hill.af.mil/crosstalk/2003/12/0312Turner.html, People Factors in Software Management: Lessons From Comparing Agile and Plan-Driven Methods by Richard Turner and Barry Boehm) gives a quite good analysis of where Agile methods fit in the spectrum of projects. The quick summary: Agile methods are best if you have lots of superior people, a project whose reliability isn't critical, quickly-changing requirements, a small team, and a culture that thrives on chaos.

FDD, too, requires above average to superior developers. That seems to be a characteristic of most new methods. Where do all of the average and below-average people go? Obviously, simple math tells us an awful lot of developers won't score in the superprogrammer category.

FDD has a Project Manager who owns the project and acts as a force field to shield the developers from interruptions and administrivia. Day to day direction falls to the Development Manager, a person endowed with both people and technical skills.

A Chief Architect is responsible for the systems overall design.

Chief Programmers own feature sets and lead small teams implementing the code. They rely on Class Owners – the actual workers cranking out software. Unlike XP, where everyone owns all of the code, in FDD the Class Owner is responsible for a particular class.

FDD has 5 processes. The project starts with an overall design, called a "domain object model". From there a features list is constructed. A plan is made, grouping features into sets which are assigned to Chief Programmers.

The fourth and fifth processes comprise the project's engine room. A Chief Programmer selects a set of features that can be implemented in two weeks or so. He or she designs each feature, creating a list of classes which are designed and implemented. This closely resembles Gilb's well-known Evolutionary process, which focuses on "time-boxing" a schedule – that is, figuring out what can be done in a two week timeframe, implementing that, and iterating.

The book includes a 10 page cheat-sheet that details each part of FDD. It's a handy guide for outfits just embarking on a new project using the methodology.

The book has frequent sidebars featuring a dialog between an experienced FDD developer and one just embarking on a project using the technique. I found this distracting and not terribly enlightening. And the authors push TogetherSoft's product just enough to be annoying.

But these are minor complaints. Unlike some programming books that are long on passion while shortchanging substance, this volume gives a clear introduction to FDD, with enough "how-to" to use the method in your own projects.

Highly recommended.

Time Management Tool

Dave Kellogg wrote in about my comments on a time-keeping system:

I have been using "Personal Time Clock" (PTC) daily for about a year for collecting metrics info, and have been quite happy with it. Cost is reasonable ($20). PTC is written by Ken Reek, author of "Pointers on C" (which I highly recommend). See: http://www.kmrconsulting.com

Features that I like about PTC:
1. A two level hierarchy of categories and projects.

2. The log file is in flat ASCII (as all PC data should be).

3. There is an optional programmable interval timer that pops up a reminder box: "Are you still working on project XYZ?" I leave this turned on (at every 15 minutes), since it helps keep me honest and current about being logged into the correct project.

4. Very flexible report generation.

5. Data export to Excel, etc. for additional analysis.

More on Redefining C’s Operators

Many readers had comments on an approach in the last Muse to avoid C’s “==” and “=” confusion. Most had comments about comparing a Boolean like:

```
  #define EQ ==
  if ( BooleanVariable EQ TRUE )
```

Don Brockenfeld wrote:

If you're not going to treat Boolean variables as predicates, then you should avoid comparing the bare variable to TRUE. In C89, all nonzero values are equally true, but only one of them will be equal to TRUE. The danger is that one portion of your code will compare the variable to TRUE and another portion will compare it to FALSE. As a result, your code will follow an inconsistent and "impossible" thread of execution. In embedded systems this can be worse than if the program had uniformly taken the FALSE branches when it should have taken the TRUE branches.

I prefer to use predicate format:

```
   if ( BooleanVariable ) {...}
   if ( ! BooleanVariable ) {...}
```

Alternatives are

```
   if ( BooleanVariable != FALSE) {...}
   if ( BooleanVariable == FALSE) {...}
```

A trickier alternative takes advantage of the fact that the standard guarantees that logical expressions will only evaluate to 1 or 0:

```
   if ( !BooleanVariable == !TRUE) {...}
   if ( !BooleanVariable == !FALSE} {...}
```

(I admire the cleverness of this alternative, but fear the consequences of a forgotten exclamation mark.)

C99 adds a Boolean type. I recommend to anyone using Boolean variables to fake a C99 compiler if you don't have one. Make your own `<stdbool.h>`. This makes your Boolean usage familiar to others, reducing training time, and might even save you some headaches when/if you eventually migrate to C99. The same can be said for making your own `<stdint.h>` header containing the fixed-size integer typedefs.

A simple implementation of `<stdbool.h>` is ...

```
typedef int _Bool;
#define bool _Bool
#define true 1
#define false 0
#define __bool_true_false_are_defined 1
```

The best way to prevent assignment/comparison errors in conditional statements is to use a good static checker. We rely on flexelint (for unix hosts) from Gimpel Software. They also offer pclint for PC hosts.

Thanks Don!

Other readers worried that using EQ, NE and the like as replacement operators extends C and makes it unreadable to untrained people. Several suggested extensions like BITAND for &, LOGAND for &&, etc; some facetiously, others seriously, others to show how bad things can get.

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.

ILX Lightwave has an immediate opening for a firmware engineer at their corporate headquarters in beautiful Bozeman, MT. The ideal candidate will be experienced in digital design/layout, embedded C and C++ programming, FPGA and CPLD design, Verilog, and embedded RTOS. Experience with both Motorola Coldfire and AMD186 is a plus. Requires minimum of a B.S. in EE or CpE, with at least 5 years experience in both digital design and embedded C programming. For confidential consideration, submit your resume with salary requirements to hr@ilxlightwave.com or fax to (406) 556-5895. For information about ILX, visit www.ilxlightwave.com. EEO/AA.

ADC in Minneapolis is looking for several engineers at any given time. There are currently several postings on the ADC website (http:://www.adc.com/careers) that fall into the category of "embedded". Check out the two openings for an Electrical Engineer II and a Sr. Electrical Engineer in our Shakopee, MN location. The posting numbers are #1629 and #1635. These positions are for data

networking engineers familiar with lower level Ethernet standards.
VHDL, Verilog and FPGA experience are highly desirable.

Joke for the Week

Those wild and crazy engineers at ILX Lightwave (who are looking for a firmware engineer – see above) came up with this all-too-appropriate prayer:

Our manager which art in our cube, hallowed be thy schedule.
Thy project end come, Thy schedule be done, minus the testing and fixing.
Give us this day our daily flogging, and forgive us missing features, as we forgive feature creep.
And lead us not into meetings, but deliver us from marketing, for thine is the paycheck, the pain, and pressure for ever and ever.

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.