EMBEDDED IN THIN SLICES

Build an Embedded Systems Consulting Company (Part 4)

Documentation, Commitments, and Insight

COLUMNS

Ready to strike out on your own? This month, Bob presents more tips for building a successful embedded systems consulting company.

By Bob Japenga

Last time, we looked at three axioms or pithy phrases that can easily be memorized and repeated while building a team of embedded systems developers. We have used axioms in our business from the very beginning. When we launched the company in my basement, my business partner's wife would come on occasion to encourage us. When she'd leave, she say, "Carry on bravely." Somehow this has stuck with the company after almost 30 years. When we leave, we often will say to the one left working late: "Carry on bravely."

Another axiom that we heard a long time ago was that "vision leaks." If you are trying to keep people on track about the core things that you are doing, using these pithy phrases can help you stop the vision from leaking. In my previous article, the three we looked at were:

- "If it's not tested, it doesn't work."
- "Always weigh the 'personal'"
- "Ask for help."

This month we will look at a few more.

2 HEADS ARE BETTER THAN 1

Today, I was trying to take out a 50-year-old water pump and pressure tank from my basement. The unit was used when the house was first built to improve the very low water pressure on our street. My son-in-law Jeff was with me. It was a nontrivial task. We debated different strategies of moving it and draining it. It was extremely heavy and bulky. When we finally got it out of the basement, I was so appreciative of having someone else with me to do this. The system had sat idle for 30 years. It was an eyesore and took up space. But I just never removed it because I was afraid of encountering insurmountable difficulties. But having someone else there made all the difference. Now I wonder why I didn't do it years ago.

This is the heart of the first axiom we will look at this month. And this is where the axiom is most effective. Efficiency is certainly important in what we do. So, having two programmers assigned to work together all the time is not the way we apply this axiom. "Extreme Programming" is a software methodology that takes this axiom to the extreme. Two people work together on all aspects of software development. Our experience is that two heads are not better than one—all the time. But very often, we face a challenge that we dread. Or a challenge that we know is technically very daunting.

This axiom is slightly different than the last article's "Ask for Help" axiom. You apply that for very short-term assistance. "Two heads are better than one" is used for a longer-term commitment.

And that's how we apply this axiom at MicroTools. We look out for those situations where having two working together is better than one. It may be something that is technically challenging. Most often, it is like my water pump project. There is something nobody wants to do. Others notice it and offer to join me in the project.

DOCUMENT IT

Many decisions, ideas, and design details

We require that any time our engineers promise something (a deliverable or a schedule milestone), either to each other or to customers, it must be written down."

feel obvious when we first think of them and therefore we do not record or document them. What is obvious to us today may become obscure (or forgotten) next week. Over the years, I have been amazed at how many things were not obvious when I came back to them years later (or even months later).

The challenge is that, in reality, you cannot document everything. I remember working on an avionics system for the F-15 in the 1980s. At the end of the project, we took a picture of the team leaders next to the documentation. The stack of paper was more than 6' high. So, I know that you can over document something.

Here are some guidelines that we give our engineers. Specifications are important. In our business, documenting all requirements is a must. We are not just talking about software requirements specifications (SRS) or test specifications. We are talking about any time we nuance the specification in some way internally or with a customer. For example, the SRS may require you to provide an output within 10 ms ±1 ms upon receipt of a particular input (or combination of inputs). The hardware design or some system constraint might require the software to output to the port on the microcontroller within 5 ms upon receipt of the input in order to meet the 10 ms requirement. Strictly speaking, this, and others like it, should be folded into SRS. But that may become either an untestable requirement (the port output is not brought out of the box) or very difficult to test. You may choose to keep the 10 ms requirement for verification reasons. But you must provide some place where you can document this requirement.

We have often documented many of these via e-mail. It does provide a paper trail. But if the project is large or lengthy, e-mail specification chains can get very unwieldy very quickly. Personally, I like keeping them all in one place where they are easily accessed. I usually put them in our bug tracking system because it is permanent, accessible to any in the project and searchable.

Documenting lessons learned is useful. We try to hold one or more meetings at the end of a project and brainstorm what we learned from this project. For years, we did not write these down. But over time I have realized that a lot of mistakes get repeated because we did not write down our lessons learned. A Wiki is a great place for a company-wide "lessons learned" document. Even if no one ever goes back to it, the process of writing it down increases that chance that you will not make the same mistake again. I also recommend that just reread them before you start something new. It is a great reminder of the mistakes of the past.

A much-overlooked documentation requirement has to do with problems you encountered and how you fixed them. It seems like pulling teeth to get software designers to do this. Here is how it often works. The designer writes some code the way any good programmer would write it using a straightforward and simple algorithm. It fails for some reason. They find that the obvious and simple algorithm doesn't work in this case. They correct it with a much more complicated algorithm. Sometimes after great effort. They move on. The next designer (who might be you) comes along and needs to make a change to the module. They see this very complicated solution and think: "This should be much simpler." They repeat the same mistake.

This happens more often than you might think. We try to encourage our designers to write out the problems and failed algorithms as part of the documentation in the source code. It can make the source code wordier, but it will save you much time in the future.

Promises and commitments are important. We require that any time our engineers promise something (a deliverable or a schedule milestone), either to each other or to customers, it must be written down. We have found that e-mail works reasonably well for this on the size of projects (one to 10 man years) we work on. Of course, if the projects are much larger, other means of documenting these would be required.

Decisions also should be documented. I am amazed how often I find that one of our designers has made (usually with others) a decision that affects others inside or outside the company. This is another one that is difficult to build into the culture. Just as "vision leaks," axioms leak unless relentlessly hammered home with reminders and encouragements.

Another aspect of decision making is: "Why did you make that decision?" For example, it was maddening to get an e-mail saying: "We decided to delay the release of the software until next month." What? Thanks for documenting the decision. But why? Here

ABOUT THE AUTHOR

Bob Japenga has been designing embedded systems since 1973. In 1988, along with his best friend, he started MicroTools, which specializes in creating a variety of real-time embedded systems. MicroTools has a combined embedded systems experience base of more than 200 years. They love to tackle impossible problems together. Bob has been awarded 11 patents in many areas of embedded systems and motion control. You can reach him at rjapenga@microtoolsinc.com.

is something that can happen on a medium-sized project. A demonstration of the software is scheduled for March 1. The team is working hard to meet the deadline. The customer is unavailable on March 1 and so the demonstration is rescheduled for March 8. It isn't enough to say that a decision was made to slip the demonstration schedule by one week. Why has it slipped? Others higher up in the customer's company might assume that we are causing the slip.

In summary, I would not say more documentation is better. We have all seen code comments like:

```
TheCounter++;  // Increment the counter
```

So, we need to be wise in what we document. We should ask ourselves: "If my memory of this project was wiped clean while retaining all of my skills, what do I need to know about this design to modify it, fix it, or re-use it?" It is not easy to make these trade-offs. Another pithy sub-axiom we use is: "When in doubt, write it out."

REVIEW COMMITMENTS

Just recently we delivered a software update to a customer. When the customer got it, he found that we missed an important commitment we'd made early on in the project. This is easy to do when there are a lot of meetings, e-mails, phone calls, and action items.

In the course of doing business or working on a project, we will make commitments, decisions, and agreements. I like to have a single source per project on commitments I make. I use Google Tasks and create a check list of the commitments I have made on a per-task basis. Google Tasks allows me to set a date for the commitment which will be automatically put onto my calendar.

I am amazed at how many fall in the crack until I review these commitments. Google calendar is a help because you can set a notification to the task so many hours or days before it is due.

SHARE THE INSIGHT

We have some great people on our team. I am amazed at how much each one of them knows. They often spend hours or days learning some new skill, new methodology, or really anything new. The problem for us is figuring out how we can share the information. Some things are easy to share. You find a cool tool that you never saw. You send everyone the link.

But sometimes we are learning so much and moving so fast we don't take the time to share a macro or shortcut that we have found helpful. It takes time. We tried creating a Wiki, but we did not have the commitment to really implement it. And I am not sure how to encourage the team to share insights with each other. One idea I've had but haven't implemented is to have a monthly lunch where we share some insight or trick or tool that we have found useful. Perhaps you have a way to encourage your team to share the knowledge and insights they have gained.

KEEP IN TOUCH

I find it particularly maddening when I go looking for help from someone in the office and find that they are not yet in or left early. We are unusual in this day and age in that we don't encourage working at home. We believe in the importance of working together in creating these very complicated devices. That said, we all have a number of responsibilities outside of work which can affect our work schedule. We have opted for the following rule: if you are going to be out of the office, it must be put on Google calendar. You must indicate when you're leaving, where you're going (generally), and when you'll be back. This is essential for continuity and building the right team so you can build the right product.

TRACK WHAT MATTERS

At one level, these axioms appear almost self-evident. Why even say them? We are complex beings creating complex systems in complex environments. Having short and pithy ways that are unique to your organization can help you keep on track with the things that matter in the midst of that complexity. If you have some of your axioms that you would like to share with me, please drop me a line. Thanks.

Next time, we will cover the last of our business axioms. But, of course, only in thin slices.