David Lambertson

Mythical Man-Month

1. Eight Chunks of Wisdom
   1. Enjoying the Craft of Programming is Necessary. – Chapter 1

How does it feel doing something you do not enjoy? Or even despise? You need to enjoy programming and most of its various aspects. You do not need to enjoy all its aspects, but you cannot despise them. Working in an area that you cannot handle or do not understand is the worst experience.

This is illustrated in today’s world as we can see people who do not enjoy what they do for work. They may be successful, but their attitude could be so much better if they actually enjoyed what did, their work, their projects and it reflects in their work.

* 1. Leave the over-the-top Optimism out. – Chapter 2

Optimism is a great trait to have, but it can lead to disastrous results. Have you ever been in a situation when someone said “everything will be fine” when in reality, it would not? It is not a good situation to be in. Not only are you giving a false hope, but in the work place it can lead to a false sense of security that you have time money to keep going at your slow-moving pace.

A situation I had to deal with such over-the-top optimism was on my mission. I served in Australia and the people there have a saying “She’ll be right mate.” It means no matter what happens, all will be good. While this is a good saying, having this attitude all the time can be hindering because it lulls you into a false sense of security.

* 1. Spread the Talent around the Company. – Chapter 3

When you play sports, you try to have the best you can on your team. Doing this within a company is like shooting yourself in the foot. Sure, the one team within the company will crank out the best code the fastest, but at the same time, what of the other teams? They are hindered. By spreading the knowledge and skills around the office, it means that yes, more time will be spent but that also means the code quality of all the other projects that your top people didn’t touch is greatly increased.

A prime example of this in industry is interns. They are learning, and may not be the best. Putting them all together they could still learn, but projects will take a lot longer. Instead, they are spread across different teams so that the projects still come out in a timely manner.

* 1. When doing documentation, do it right. – Chapter 6

To document or not to document, that is the question. Documentation is, you could say, the steel supports for programs and code we write. Without it, it could fall apart at any second. Those who come to work on your code later will not know what is happening. When a user tries to use it, they will struggle. Designing goes hand-in-hand with documentation. Good documentation gives not only the user, but also the programmers something to work with to figure out what is going on.

I saw this on the team I worked on. All our code was housed on GutHub, and it had documentation with it on how to get it on your local computer, but beyond that not much. I had to have my mentor teach me what was going on and how to do specific things within the code. What she didn’t explain, was left up to me to gleam for myself. This made it so I had to work longer on a project when updating old code.

* 1. Communicate like your life depends upon it. – Chapter 7

Communicating brings people together. With those people, it brings projects together also. If teams did not communicate, more projects would fall by the waste side. With communication, teams are able to convey meaning, problems, and success. They can show and explain what is working and what isn’t.

An example of this comes from my internship. Before a page would go live for people to see, our QA had to look at and test it. She would then send back the things she feels needed to be fixed. Sometimes, there was something I just couldn’t fix with what I was working with. I would have to communicate with her and our designer to come up with a plausible solution that everyone liked.

* 1. Size Matters. – Chapter 9

If you had all the space in the world, what would you do? Well the problem is, we do not have all the space in the world. We need to set budgets for size when developing software. We cannot just start coding and hope it will fit and run efficiently. We have to take into account the standard software and hardware users have. This should be done before delivering a product.

At my internship, I had this page I was working on. It would check to see if the user had reserved a name to take to the temple. A problem we ran into is it could run for quite a long time. If a user had over 100 names reserved, it checked them all, causing the software to hold up the browser for a few minutes. We had to redesign how we were doing things to accommodate for this setback.

* 1. Starting with a pilot saves time and money. – Chapter 11

Planning ahead to have a pilot, or a throwaway model, saves time and money. Even though you are making a two, it saves time and money because there is less time maintaining. There will obviously still be maintenance, but the second model you keep will have less bugs, it will be cleaner code and be more efficient as you are able to build it after you know what you are doing.

I saw this at my internship when we were trying to implement google maps on one of our pages by making it a module for later use. The first one we used, but it was not the best. We made a second one which became the standard one for us to use.

* 1. Write code to succeed. – Chapter 13

When writing code, do you plan on it to fail the first time? No. You want it to succeed. When you write code planning on it to succeed you write good code. If you just tell yourself that the first time it’s going to fail and there is nothing to do about it, that’s exactly what will happen.

When I was working, I would always plan for my code to work the first time. It actually made me excited to work on it and kept me on time.

1. The Mythical Man-Month book by Frederick P. Brooks Jr. is a book that applies even today. It talks about specific things which are greatly in need in the software industry. Optimism, working hard, documentation, teamwork. This are just a few of the items the book mentions. Without teamwork, software development is possible, but is it really necessary? Working with others speeds up the process, it brings various knowledge and methods together and you are able to deliver code and software more quickly. Optimism keeps us going. When a project is dominating us and hope seems lost, having that optimism can get us through the tough spot.