Speaker's Notes for "Documentation for Developers"

Delivered to Austin RB at The Capital Factory, 4 January 2016

Biography:
Mike Marotta is a technical writer for projects at huge organizations and tiny companies. He has been granted several literary awards for his non-fiction writing. The first came from a nomination by a Smithsonian curator. That article on the origins of coinage corrected the Encyclopedia Britannica. While learning to fly, he wrote for regional and local aviation periodicals. As a petty officer in the Maritime Regiment of the Texas State Guard, he teaches emergency computer operations.

Abstract:
The title phrase "Documentation for Developers" has several meanings. Specification and presentation are the yin and yang of documentation.
Of course, developers should comment their code. Ethical conduct requires choosing good names for methods, classes, and instances. When we turn to GitHub or Heroku Support, we are looking for help, not a literary experience.
But documentation is more than that because you always have several audiences. Devops, support, production, marketing, and management on your side all have analogs among organizations and individuals who depend on your work. And you never know how they learned English or how they use at work or at home.
Ultimately, in a market society based on division of labor, we all depend on other professionals. Selecting the right technical writer for your project requires knowing the standards for good technical writing.

Slide 2

Eredeti változat:

- Észtül: *Elav kala ujub vee all.*
- Finnül: *Elävä kala ui veden alla.*
- Magyarul: *Eleven hal úszik a víz alatt.*

My username on this email is a play on what is supposed to be the only sentence mutually intelligible across Hungarian, Estonian, and Finnish. In fact, it is not understandable. But it is interesting to me. I pay a lot of attention to words, especially to how they are received subconsciously.

Slide 3

I started as a computer programmer. On a database project for General Motors, no one wanted to write the user manual. I had written two small books and published half a dozen magazine articles, so I wrote it. Over the years, I did more documentation and less programming. In the 1980s and 1990s, I used the TeX/LaTeX typesetting language. TeX was the antecedent to SGML from which came HTML.

Slide 4

The best documentation creates an organic system of knowledge.

Slide5

If you do not love writing documentation, then find someone who does.

The quote is from *The Fountainhead* by Ayn Rand. Peter Keating would like to design a large government housing project, but he never developed the skill. He was always a "people person." He has come to Howard Roark for help – as he did several times since their college days. Roark considers clients to be like steel and rock: part of the technical challenge. Roark understands Keating's desire to help others – though he does not share it as a primary motive. Roark says, "to do things for other people, you need to be the kind of man who can get things done. But to get things done, Peter, you must love the doing."

Slide 6:

Usually attributed to the United States Marine Corps.
We all work hard. The critical difference is in the planning.

Slide 7:

My own rule. Every project begins with some kind of documentation. The best projects begin with a full set of specifications. Those must include the user manuals. The user manuals define the goal. If you do not know where you are going, you never will arrive. Most often, technical specifications alone – perhaps only a statement of work – guide programmers who work intuitively. They are in genius mode – which sounds good. We all like to be geniuses. But it is not a mature model. It is not sustainable. It is not repeatable. At the end of the process, they call in a technical writer to document their work. At that point, it is too late. Documenting "as built" work is like drawing blueprints for a house – or a neighborhood – after carpenters, electricians, masons, and all the others have been working as they pleased to do the best job they know how according to their own best judgment. We can hope that they knew their crafts and got along well.

Slide 8:
The user interface is the primary documentation for most people. That includes devops people. You depend on the I D E, the interactive development environment. We are all users. We program in languages that we did not write in the first place. Someone else created the compiler or interpreter. The UI is supposed to be as intuitively obvious as the dashboard of an automobile. That's why we call now them "dashboards" for Salesforce, SharePoint, Wordpress, Basecamp….

Slide 9
No comment.

Slide 10
These tools work. You do not begin the design of a house with a little house or a replica made of different materials. Such modeling can be important. Roller coasters are tested at one-third scale. But they do not begin that way. I know a local web designer who carries a box of Crayola Crayons. Architects do model in clay. Automobile designers do model in clay and wood. But they do not model a car by building one without any plan. You need a plan.

Slide 11.
Visio is an easy platform, common to the office environment. It provides an array of tools. How you use them is up to you. If you have ever used pliers to get a better grip on a screwdriver, you understand how these things work.

Slide 12.
Visio database charts.

Slide 13.
Visio business processes. You can use them as you need to. You do not have to just be limited by Microsoft's labels. And you can mix and match, again, like the Channel Locks holding the screwdriver.

Slide 14.
See "The Entity-Relationship Model: Toward a Unified View of Data" by Peter Pin-Shan Chen, MIT, *ACM Transactions on Database Systems,* vol 1 no 1. March 1976, pp 9-36.
If you stop and think about this, you realize that there are many ways to conceptualize a program and what it does. There is even a paper that explains the relationship between Object-Oriented Programming and Ayn Rand's theory of Objectivist Epistemology. The important thing is to actually use these kinds of tools because they are the documentation of your design process.

Again, we want to follow the scientific method, and showing your work is required in science. Otherwise, you are an alchemist, which sounds cool. But alchemists claimed to transmute elements, delivering miraculous results by secret methods. The ruse needed to work only long enough for them to escape the king's realm. In our time, scientists transmute elements (and transmute life) by methods that were published first in peer-reviewed journals, and now in textbooks. Too many software developers masquerade as code wizards whose

smoke and mirrors need to work only long enough for them to change jobs.
Documentation - from the design to the helps - allows peer review, testing, validation, and (if need be) falsification of theory.

Slide 15

The best way to learn something is to teach it to others. If you can explain your work, you will understand it better. If you cannot, then perhaps you need to do more thinking, collecting data, talking to users and experts, and otherwise planning your work.

Slide 16

Han Solo did not believe in hokey old religions and ancient weapons. But this method from the days of COBOL – or the Lords of Kobol is you prefer Galactica – actually works. You begin with the output. For each element, you trace it back to the input.
Just put Warnier-Orr in your search engine and read the Wikipedia article.
http://en.wikipedia.org/wiki/Warnier/Orr_diagram: "Warnier/Orr diagram" See also
http://www.mindapp.com/warnier-orr-basics/

Slide 17

Warnier-Orr does allow conditional branching and other controls. However, it is designed for large databases. For **processes**, other tools work better. We will see one, but first …

Slide 18

This is just one out of many. BLiner 6 is a commercial product, but they do offer a free demo download.

Slide 19

We used to get in trouble for calling them "Nazi Spiderman Diagrams."
See "Flowchart Techniques for Structured Programming" by I. Nassi and B. Shneiderman, SUNY Stony Brook, SIGPLAN Notices, 1973 August. See also " **A short history of structured flowcharts (Nassi-Shneiderman Diagrams)"** by Ben Shneiderman (Draft May 27, 2003) at
http://www.cs.umd.edu/hcil/members/bshneiderman/nsd/

You can draw them for yourself and make them into macros. As shown below, firms that sell flowcharting software also sell Nassi-Shneiderman diagrams.

Slide 20

It looks like this.

Slide 21

It works like this.

Slide 22

And you can buy it from several places.'

Slide 23

Why bother?
I overheard a conversation between a programmer working the support desk and a client. The programmer was saying, "I want to show you how it does work. I do not want to talk about how it doesn't work."
If computer science is a science, then it is repeatable, testable, and falsifiable.
Allow me to suggest that if programs really "blew up" you would be more careful.

Slide 24

What if computing were as consequential as flying? We call aircraft "forgiving". What if a bug resulted in an explosion and injury to the programmer?
Pilots spend as much time in flight planning as they do in flight, often more. To love flying, you must love the planning. Sporty's Pilot Shop sells 153 different flight planning aids, from tablets of printed sheets to software to handheld dedicated computers. Jeppesen also serves private pilots but is favored by transport and passenger airline professionals Flight planning starts the documentation. At the end of your flight, professionals and advanced private pilots formally "close the plan" with the controllers. Aviation depends on documentation. What makes software development so special that documentation is at best an afterthought?

Afterthought in Greek is EPIMETHEUS. Forethought is PROMETHEUS. While Prometheus was planning humans, his brother, Epimetheus, gave strength, power, speed, and weapons to all the other animals, leaving us defenseless. So, Prometheus gave us fire. Do you want to be like Epimetheus, all after-thought and make-up and fix-up. Or do you want be like Prometheus, who thought ahead?

Slide 25.

My wife is in computer security and she takes a lot of certification tests. The writer here actually spelled the word right the first time, then misspelled it twice in succession. What if his word processor blew up and killed him for a typo? He would be more careful. Just to say, lots of people claim to create documentation, but, mostly, they just duck shrapnel that hits other people.

Slide 26

In 1976, Joseph Weizenbaum of the MIT Artificial Intelligence Laboratory published this book. Weizenbaum said that like gamblers, programmers have superstitions. Perhaps primary is the superstition that one more fix, one more patch will solve their problem. Weizebaum said that the real problem is that they begin writing programs without any knowledge of the substantive field in which they are working. He meant not the programming language or computer system, but the application, the supposed goal, in other words, the client user's point of view.

If nothing less, your technical writer is the bridge connecting the user community with the devops team.

Slide 27

... bright young men of disheveled appearance, often with sunken glowing eyes, can be seen sitting at computer consoles, their arms tensed and waiting to fire their fingers, already poised to strike, at the buttons and keys on which their attention seems to be riveted as a gambler's on the rolling dice. When not so transfixed, they often sit at tables strewn with computer printouts over which they pore like possessed students of a cabbalistic text. They work until they nearly drop, twenty, thirty hours at a time. Their food, if they arrange it, is brought to them: coffee, Cokes, sandwiches. If possible, they sleep on cots near the printouts. Their rumpled clothes, their unwashed and unshaven faces, and their uncombed hair all testify that they are oblivious to their bodies and to the world in which they move. These are computer bums, compulsive programmers…"

Joseph Weizenbaum, *Computer Power and Human Reason: From Judgment to Calculation* (W. H. Freeman, 1976) quoted in *Hackers: Heroes of the Computer Revolution* by Steven Levy. (Nerraw Manijaime/Doubleday, 1984).

Slide 28

Good technical writing is just good writing. Good technical writing is also easy to translate. I often recommend THE ELEMENTS OF STYLE by Strunk and White. It can be a bit dated, as it was addressed to a college audience in the 1920s, but the fundamental truths still apply (as time goes by).

Active voice: She opens the door. versus **Passive voice: The door is opened by her. Indicative mood: This will be good.** versus **Conditional mood: This would be good. Active voice** shows who is responsible. Passive voice leaves questions**: "The file is updated.**" Who updates the file? How is it updated? By what means? Identify the program that does the work.

Slide 29:

What you are about to see comes largely from Microsoft Word. Project managers and human resources people write specifications for technical writers without any knowledge what we actually do, but let us move forward here anyway.

The important thing to remember is that this is under Tools in Microsoft Word. The tool itself has changed over the years, as you will see. It was a better tool some years ago. But there is no mystery. Before you hire a technical writer, run their sample through the MS Word Tool for Grammar and Spelling.

Slide 30

The Sermon on the Mount is where Jesus preached to the masses. It is understandable to a child. The Microsoft Grammar Checker says "Grade 5 point 3." A ten-year old understands the essential teaching of Jesus.

Slide 31

You need a master's degree, Grade 18, to understand the Declaration of Independence. Is it any wonder that for hundreds of years, people were martyred for Christianity, but no one can explain Democracy? Note that this passage actually has fewer passive voice sentences than the Sermon on the Mount. However, its long sentences- 53 words per sentence - render it difficult to grasp.

Slide 32

From an actual user manual from an actual firm that actually employs people here in Austin. You think it is hard to read now, wait until I blow it up, then it really will be unreadable.

Slide 33

Find the error: "In My Organization Attendance, Time Card Status Added to Employee Absence Details Tab in My Organization Attendance on the Employee Absence Details tab that is displayed when then Absent button is clicked for an employee who has already entered and absence, the Time Card Status column has been added to display the approval status of the absence time card."

Slide 34

Heck, never mind the error, just find the subject.

Slide 35

On a scale of 1 to 100, this rates less than Six-and-a-Half. Also, while the tool does show a 12th grade reading level, I know that this is maxed out. I put in scientific papers and similar material that returned a 12. This is as high as it goes.

Slide 36

I wrote this article for the British Society for the History of Astonomy. It is an academic work that was peer-reviewed. And it is readable at a 10th grade reading level. Easy-peasy. You can explain difficult subject matter in language that anyone can understand.

Slide 37

This was from an actual user manual for a recent project with good metrics, sensible design, and good oversight. I just wanted to use it as a positive example. So, I was surprised to see how poorly it scored. A closer read revealed why. You can see that it has too many words per sentence; the sentences are too long. Also, typical of computer documentation, it is 100% passive voice.

It reads at grade 12 Point 9.

Slide 38

So, this is how you fix it. I marked up the problems; and then fixed them.

Slide 39

I made almost all of it active voice. In doing that, I also shortened the sentences. You get rid of passive voice, you shorten the sentences.

Slide 40

You must invent a system of meaningful names. Cute names are not helpful. Abstract names are useless.

Slide 41

It did not take much time on Github to find an example of bad documentation. The Description should say something different from the Summary.

Slide 42

In other words, it is a well-known fallacy that there is no time to do it right, but plenty of time to do it over. Do it right the first time. What I mean here is that you have one chance to do it right. It is your life, your time. How you invest it is up to you. If you value yourself, your talents, your skills, you will insist on doing it right the first time.

I am talking about your right to the Pursuit of Happiness as expressed in the Declaration of Independence. How would you react to someone who said that you have no right to be happy, that you must sacrifice for the common good? Well, what do you say when someone says that they know it is not right, but we have to ship the updates now? It is not just the company's profits or the user's work – though there are those – it is your life (not just mine, though there is that) that you give up when you succumb to second-hand standards for documentation.

Slide 43.

My favorite Sally was Sally Mae. She lent me $100,000 to go to college and earn a master's degree.

Slide 44

Too often, when registering for a website, you do not get a message like this until after it rejects your password. Sallie Mae does it right. The worst example I experienced was the Red Cross. My degrees are in criminology. I am a petty officer in the Maritime Regiment of the Texas State Guard. When I went to sign up as a Red Cross volunteer, the Red Cross website was not as nicely implemented as the Sallie Mae website. I got kicked out over my ZIPcode.

Slide 45

Just continuing the good examples. The Sallie Mae website actually tells you what you need to do, **before** you do it and get an error.

Slide 46

Slide 47

Jeff Millington was a software engineer on a project. The software engineers were supposed to write their own documentation and pass it to me for editing. His was horrible. It did not even adhere to the basic rules of grammar. So, I went to him. I said, "Jeff, you program in C. It has a grammar and a syntax and a vocabulary. English is your native language. What is so hard? And he replied that he did not perceive C as a language with a grammar and syntax and vocabulary. To him, he was stringing wires, and connecting pulleys, and attaching gears. Sometime later, he came into my office. He asked about the process of grammatical standards. What happens first, he asked. Do scholars define the language or do they look at how people use the language. I said that languages change, that scholars only record how people actually use the language here and now. He said, "So, if you are following the manual, you are not using the current version."

I get the point, and I get the humor. I am just trying to be helpful.

Slide 48

The End

After the End

How to hire a good technical writer? First of all, anyone who has done it or claims to be doing it is probably all right. That said, it is a lot like programming. How do you know a good developer? I remember one interview I was applying to do some DBMS programming and the managers said, "Don't tell us that you can do anything because that's what the last person said even at the last minute as we escorted him out the door. What's with programmers, anyway?"

As a technical writer, I find success being a professional idiot. I never assume that I know anything. If you can show me, then I can explain it to someone else. TW is a kind of quality testing. Which is why it has to come first because you do not want to be two weeks from ship date when the TW says, "You overlooked this major thing here." Like all the screen prompts.

Sentence length. Word length. Words per sentence. Sentences per paragraph. Active voice. Sermon on the Mount versus the Declaration of Independence. I wrote my own program in Fortran in 1990. You can find out the rules for readability and write your own parser in Ruby or Java or C if you have some aversion to MS-Word. Plus you can find parsers and graders online, for instance, this one…

The harder something is to understand, the lower the reading level should be. I documented the interactive debugger of a multiprocessor industrial controller at a sixth grade reading level. Engineers read well enough, but when a production line is down, no one is looking for a literary experience. I worked with an engineer at a robotics company who was completing a master's. He ran his thesis through the Macintosh grammar checker and re-wrote it past the Grade 50th level. It was just about unreadable. He won high honors, I think.