“What Do I Do? How Do I Do That?”

Mentoring Junior Developers Into Productive Team Members

Dan Mallott

westMONROE

THANKS TO ALL OUR SPONSORS!

BizStream

ATOMIC OBJECT

THE C2 GROUP

KL&A KUNZ, LEIGH AND ASSOCIATES

{M} MICHIGAN SOFTWARE LABS

WEST MICHIGAN TECH TALENT

TEKTON®

KFORCE℠

Leading EDJE

Mutually human

Scout APM

Sweetwater® Music Instruments & Pro Audio

DAN MALLOTT

Twitter: @DanielMallott
GitHub: https://github.com/danielmallott
LinkedIn:
https://www.linkedin.com/in/danielmallott

1. SENIOR PRINCIPAL FOR WEST MONROE
2. DEVELOPING SOFTWARE SINCE 2011
3. STARTED AS A DBA – NOW A DEVELOPER
4. PRIMARY EXPERIENCE WITH MICROSOFT TECHNOLOGIES
5. ALSO, A USA HOCKEY REFEREE

WHY ARE WE HERE?

- You are (or expect to be) in a position to manage others
- Your organization hires lots of developers fresh from their degree or training program
- You need your new hires to be productive on Day One (or soon thereafter)
- You have an issue with young developers leaving less than two years after they get hired

westMONROE

Or…
Software is Easy;
People are Difficult

A Couple Quick Notes

- When I say “right” in the context of software development, I mean “following generally accepted best practices and conforming to organizational standards”

- “Need” vs. “Want” is a tricky concept. For our purposes, a “need” is anything that absolutely must be fulfilled for progress to occur – i.e., whatever the “need” is, must happen

- If you have a question, please feel free to put your hand up

Things We Cannot Solve For

We are focused on coaching here; some things are not coachable

My firm calls these the “brilliant basics” – they are behaviors that ***anyone*** can master no matter their skill or experience level:

- Effort (This includes the effort needed to learn the domain)
- Attention to Detail
- Paying Attention
- Showing Up (On time and dressed appropriately)
- Being Willing to Accept Coaching

How Many of You Have Over 10 Years of Experience?

How Many of You Have Over 20 Years of Experience

Do you remember a particular individual who mentored you early on? Maybe the person “responsible for turning you into the developer you are today”?

How many of you remember your first task as a professional developer? How did you feel?

My Story

Or…how I wrote something useful that I would never include on my resume

Mentoring Basics

What to do when you first hire a new Junior Developer

westMONROE

Understand (and Respect) Their Background

Your new junior developer will have had a different educational background than you did

- They might not have a Computer Science or Computer Engineering degree
- They might have graduated from a coding boot camp
- They might not have a degree at all

Your new junior developer may not share your (insert demographic categorization here)

Take the time to understand where they came from (both personally and professionally)

Check any pre-conceived notions at the door

Never assume!

Engage on a Personal Level

1. You don't have to be best friends with the junior developers on your team, but…
2. Get to know them on something of a personal level
3. Let them get to know you on something of a personal level
4. Knowing what's going on in someone's personal life (even in the abstract) can help you to understand performance issues and even anticipate time off needs
5. Understanding what they want out of their personal life can help you understand what they want out of their career (and how to temper expectations appropriately)
6. Include them in team and company activities

Check for Understanding

- Always check to make sure what you have told the junior developer makes sense ***to them***
- Be prepared to explain using several different examples and methods of teaching
  - This includes differences in language
- Understand the three basic learning styles – people are usually better at two of the three
  - Auditory
  - Visual
  - Kinesthetic
  - NOTE: There is research that shows no correlation between learning style and how easily someone learns. In my experience, employing all three leads to the best outcome
- Challenge them to explain back, using their own words and example (preferably from the current domain)
- Learn to recognize when they are not understanding but are unwilling to ask for help or clarity

westMONROE

Learning to Code Professionally is Hard

So is teaching it…

westMONROE

Processes Can Save You

- If you are an Agile team, Grooming and Tasking can help check for understanding, so long as everyone is engaged
- If you are a Waterfall team, spend adequate time up front to ensure understanding
- Ensure you have coding standards, and that they have been communicated!
- Do code reviews, whether as part of a formal check-in process or informally at their desk
  - Have them walk you through what they've done
  - Have them demonstrate working code
  - Reserve judgment on what they've done until the end
  - Let them fix their mistakes!

AVOID COMMON PITFALLS

- Trust, but also verify

westMONROE

Understand Your Time Is No Longer Only Yours

Humans work best uninterrupted	Context switching is more expensive for a human than a computer!
You will be interrupted!	And it will (always seem to) be at an inconvenient moment
Your new normal	As a leader of people, this is your new normal, and one of the hardest thing to learn to handle
If a junior developer is asking a question	Generally, it means they are blocked, or think they are blocked, and need the answer to proceed
NOTE	It is okay to put off a question if you have something that needs to be finished, but never just refuse to answer • Give them a lead or avenue to explore • Challenge them to complete other parts of the task (using a stub method or interface in the meantime)

westMONROE

The Corollary – Knowing When to Say "Trust Me"

- There are going to be things that are not explainable in a reasonable amount of time, whether because of complexity or the time sensitivity of the task
- In these cases, saying "Trust Me" is okay, as long as it is accompanied by a promise to explain at a later (i.e., less stressful) date
- This is really hard, though, because you're going to want to make sure they understand, and they're going to ask (or demand) explanations
- Doing this should be the exception, not the rule – if you always just say "trust me", you're creating a coding monkey that only knows how to implement things one way – yours – for the problem at hand
- Examples could include:
  - Why we use certain patterns
  - Why the ORM is doing what it's doing
  - Framework oddities

What Not To Do

westMONROE

Don’t Assume…

…Knowledge

…Understanding

…You Know Them

…How Capable They Are

Don’t Hold Your Knowledge Over Them

- You have more knowledge…about many (most) software development topics
- There will be things they know better than you do
- No one like to be made to feel like they are inadequate or “less than”
- *Teach, don’t dictate*

Don’t Get Angry (When They’re Present)

- One of the most destructive things you can do is to get angry at an individual
- Your junior developers will screw up (just like you do) – we’re all human
- Most of us are lucky enough to work in fields where the worst thing that happens is our company loses some money – usually no one dies as a result of our messes
- Treat screw ups as learning opportunities. Always give constructive feedback and help them understand what to do different the next time (in fact, they will often be able to tell you that themselves)
- Later, when you’re with your friends and away from the team, you can be as upset as you want to be – particularly if you had to do extra work

westMONROE

IN SUMMARY

- Develop Mentoring Skills
- Get to Know Your People
- Check for Understanding
- Use Process to Help the Teaching and Learning Journey
- Be Okay With Timesharing Yourself
- Know When to Say "Trust Me"
- Follow the "Don't"s

Questions?

Thank You!

@DanielMallott

https://github.com/danielmallott

https://linkedin.com/in/danielmallott

westMONROE