# Transcript: It Podcast – Ep 56 – Multi-Developer Environments with Carolyn Sher-DeCusatis – mark Kinkead - Dave Couch

*The following transcript is a verbatim account of the video or audio file accompanying this transcript.*

Speaker #1 (Narrator):

WGU's IT audio series. Flexible, portable, profound.

Speaker #2 (Carolyn Sher-DeCusatis):

Hello, I'm Carolyn Sher-DeCusatis.

Speaker #3 (Mark Kinkead):

I'm Mark Kinkead.

Speaker #2 (Carolyn Sher-DeCusatis):

We're two course instructors on software team and today we're joined by Dave Couch, who is the evaluation supervisor for programming courses.

Speaker #4 (Dave Couch):

That's the best way to put it, yeah.

Speaker #2 (Carolyn Sher-DeCusatis):

Yeah.

Speaker #4 (Dave Couch):

I'm in charge of all the evaluators that do evaluations for all the programming and software courses.

Speaker #2 (Carolyn Sher-DeCusatis):

My experience in programming comes from a more academic setting where I did a lot of prototyping and building small-scale things that hadn't been built before to accomplish a goal for one or two people. Mark comes from an industrial environment.

Speaker #3 (Mark Kinkead):

Yeah, I've worked a lot in military and in law enforcement. I've done financial commodities trading and various other self-gaming and various other very small projects. Well, I don't know what you'd call small, but yes, I have. But no, I'm not a great academic, I can barely spell my name.

Speaker #2 (Carolyn Sher-DeCusatis):

Dave, you did some ERP programming before you came here?

Speaker #4 (Dave Couch):

Yeah, before I came to WGU, I did large-scale ERPs for manufacturing or sales companies that might have 100 offices across the US and they would need a database to figure out how to query large amounts of data to find just the specific parts you need or how they put together, say a solar panel which has hundreds of little parts, that each all have different numbers. But how do you do that in layers and keep track of every parts of the cost-benefit of factors are all known.

Speaker #3 (Mark Kinkead):

You've done a little bit, that's for sure. Wow.

Speaker #4 (Dave Couch):

Just a little bit. Yeah, there's nothing like spending three weeks on a piece of code and it's still not working and you're like, I'm so tired.

Speaker #3 (Mark Kinkead):

We've all learned different ways of programming and different techniques. I mean, I've started in the old days in the military, it was all waterfall model and have watched it all transform into Agile and beyond and prototyping, which are much better ways to work.

Speaker #4 (Dave Couch):

Absolutely.

Speaker #3 (Mark Kinkead):

What do you know about Pair Programming, Carolyn?

Speaker #2 (Carolyn Sher-DeCusatis):

Well, while studying programming, I wasn't trained as a programmer. I started off with a degree in physics and needing stuff and having friends that needed stuff. So I went back to school when I was in my 40s, when I got my graduate degrees. Everything was about agile, so we did a big emphasis on pair programming, both to teach people like me who were coming [inaudible] even people more like me, but people also from other environments who were getting up to snuff on Agile and XP. But just because it's a really good technique. When you're doing Pair Programming, you've got two people working on the same computer and one of them is driving and one of them is keep it sink, and it really makes you much more efficient. You don't lose as much time from having two people have done the job, you would think, it's something like only 15 percent because you gain so much in the quality of the code by having two sets of eyes looking at it and not having as many delays when you just don't know what to do spinning your wheels.

Speaker #3 (Mark Kinkead):

I was going to say I've worked in larger projects where we've had seven people working on, shall we say, the same three code files that are shared amongst various other activities when we're adding features and those sorts of things. We would use Rational Rose as our version control system. We would all get together on one particular day, probably on Friday or something before no lunch and then we would all get together and merge. We would draw straw sometimes and one-person would merge their code into the codebase first and then the others would, and we would share that day and get together and talk about whatever your buddy has done and when we have conflicts, we would sit there, right there and resolve them and all that other stuff. Those were very large groups. Then with the Agile method that came in, there's more paring down of responsibilities for people, but you still have a lot of that large group sharing that goes on.

Speaker #2 (Carolyn Sher-DeCusatis):

I know that I really focus on the background in Agile whenever I meet with students and we share screens, whether it's debugging, which is a really good technique for because of the two sets of eyes, or whether it's just helping somebody who's stuck because of that sense in Pair Programming that you don't spin your wheels as much.

Speaker #3 (Mark Kinkead):

A lot of this makes me think about open source projects and how big some of those projects are and it just seems to me with how people are spread out. A lot of these techniques are very important for them to follow and keep working through. Definitely, open-source and all that stuff is something that we really like to use, and we're all as programmers and coders, we're very used to looking on the Internet and finding a lot of stuff to use and that sort of thing but that becomes a problem here at WGU when you're actually working on your programming courses, because it's different than the real world, don't you think Dave?

Speaker #4 (Dave Couch):

Yeah, WGU we're a competency-based model, and you need to demonstrate competency to pass each course or assignment or task however it works in which course. The competency of most of the programming courses we have is not whether or not you can look on the Internet and figure out how to modify something like an open-source solution but that you can actually demonstrate. You know how to code something from the ground up without looking at somebody else's solution, which is different. It's almost like this is the first step. The next step would be using open source stuff and since most of our courses are still that first step, using all the tools and stuff out on the Internet would be inappropriate.

Speaker #3 (Mark Kinkead):

Well, so let's say that I want to go to Stack Overflow because I don't understand how to convert a date-time into a zone date-time or something like that and I Google that, would that be considered cheating?

Speaker #4 (Dave Couch):

If you just did one little piece like you trying to figure out how something works, that's probably not considered as much cheating as trying to use pretty much the entire code base or every element that way. Realistically, you should be able to code each of these aspects without having to do too much Google searches because a programmer should be able to know how to do a lot of these things in just general practice. I like to use this example, when I was doing my CS undergrad degree, there was one assignment in C++, where we had to write a program that basically was a checkbook and one of the last functions, because C++ uses functions, one of the last functions was to write the checkout. I'm pretty sure people today probably don't even know what that is. Starting to get myself seem old. But you had to take the numbers and write 1000, 100 and in a sense had to be the fraction where it was 99 over 100. I was actually really struggling on that part. I did all the other stuff. I did all the database stuff where I tract everything but I couldn't figure out, for the life we had to create the loops to get that to print properly. I did actually go to Google and I found one that actually did it just by chance. I documented the heck out of that one little method. I said, this is the only method I did this in, it's not the entire project, it's just this one spot, and because I can't possibly make it any better or do anything different. But in reality, we should really try to shy away from doing that too often or at all.

Speaker #3 (Mark Kinkead):

Like I go to GitHub or something like that and look at other people's projects, if they happen to be available, that would be considered a violation even if I don't take that code. Is that correct?

Speaker #4 (Dave Couch):

I'd say it is a violation because it's hard once you've seen it to not copy it, even subconsciously not copy it. But we see students that get caught with originality errors and the first thing we tell them is, put that aside, don't look at it, don't reference it, and write it from the ground up. You will always write it different. I can guarantee you if I write the same code twice, there will be differences in them. I will never write them exactly the same.

Speaker #2 (Carolyn Sher-DeCusatis):

What are the examples in the learning resource or in the webinars?

Speaker #4 (Dave Couch):

The webinars, I don't think we have any examples that specifically tell you every piece of code, correct?

Speaker #2 (Carolyn Sher-DeCusatis):

We have code snippets, things that are likely equivalent of a Stack Overflow.

Speaker #4 (Dave Couch):

Okay. Yeah, some of the code snippets we know about and we look at them, but we discard them when we're doing an org-check.

Speaker #3 (Mark Kinkead):

Right.

Speaker #4 (Dave Couch):

A good example is you've got the template of the park class in software one. Yeah, that shows up on the tools we use. But we know that that's going to show up, so we don't look at that. We don't have a number we look at in papers. We look at the overall work, then determine, is this really a match to somebody else's stuff, or is it something that just is that's the best way to do it?

Speaker #3 (Mark Kinkead):

Yeah. I guess the bottom line is, even though it's very tempting to look at stuff, you should try your best to learn how to program these things and if you have trouble with them, you should refer to the online resources that we have here at WGU, would that be a good way to put it?

Speaker #4 (Dave Couch):

That would be a good way to put it. A better way to think about it too is if you can't do these simple ones without having to have larger portions of this written for you, you're going to have trouble when you get to the real world and your boss says, hey, can you write this program to figure out the laser wavelength or something?

Speaker #3 (Mark Kinkead):

Yes, that's right.

Speaker #4 (Dave Couch):

Because you didn't learn the fundamentals, won't be good for you. Yeah, you know what I mean?

Speaker #3 (Mark Kinkead):

That's right.

Speaker #4 (Dave Couch):

I don't want to see that. I want to see everybody succeed.

Speaker #3 (Mark Kinkead):

Yeah, that's right. We definitely want to have you confident when you leave the school for sure. Well, I think that's about all we've got for today, and thank you Dave for joining us.

Speaker #4 (Dave Couch):

Thank you.

Speaker #3 (Mark Kinkead):

Carolyn, any last words?

Speaker #2 (Carolyn Sher-DeCusatis):

Code like the wind.

Speaker #3 (Mark Kinkead):

Everyone, take care.

Speaker #1 (Narrator):

Schedule time with your course instructor to explore more deeply. WGU, a new kind of U.