Voiceover Artist 0:00
Are you ready to manage your work and personal world better to live a fulfilling productive life, then you've come to the right place productivity cast, the weekly show about all things productivity. Here, your host Ray Sidney-Smith and Augusto Pinaud with Francis Wade and Art Gelwicks.

Raymond Sidney-Smith 0:17
Welcome back, everybody to ProductivityCast, the weekly show about all things personal productivity. I'm Ray Sidney-Smith.

Augusto Pinaud 0:22
I'm Augusto Pinaud.

Francis Wade 0:23
I'm Francis Wade.

Art Gelwicks 0:24
And I'm Art Gelwicks.

Raymond Sidney-Smith 0:25
Welcome, gentlemen, and welcome to our listeners to this episode of productivity cast. Today, we are going to be talking about software stewardship. And really what that means for the world of productivity, culture and society is remarkably impacted by how software today is developed, there isn't anything that isn't really run by technology, from your water plants, you know, the the treatment plants that are running water and sewage, to your cell phone. And on smartphones in your pocket. Everybody is connected to software in some way, shape, or form, even if we're not using that software directly. And we thought it would be really interesting to look at that through the lens of productivity. And Francis, you brought up this topic to the ProductivityCast. Team. So I wanted to have you kind of preamble us talk to us about what you were thinking as a related to this topic, and then we will get into our discussion.

Francis Wade 1:24
But I was ranting, you know that many productivity folks do when they try to use a piece of software and realize that the developer, or whoever designed the software has totally missed the mark. And what's happened is that someone came up with a very bright idea. And the bright idea seems to be interesting and useful. We either use that using the software, or we try to use it. And we hit upon a block a block, or some kind of stop. And we realize from using the software that the developer had a half of an idea, or a poorly formulated idea before developing the software. And that's why the software doesn't work. So it doesn't fit our needs. It doesn't do what we want it to do, it kind of halfway gets there. But it needs to go away much further. So for example, Google Canada, or outlook Canada, for that matter, those both of those software apps, and all of the Canon most of the counters I've ever seen, a handful of are exceptional, but were designed based on appointment calendars. So they basically copied the appointment calendar that you would see in a doctor's or a dentist office where you put in meetings. And it's meant to track events that take place with other people. And that basic idea. Fast forward to today. And of course, we live in a world in which time blocking is a key element, or timeboxing, or calendar blocking any any time at which you decide to schedule a solo activity into your calendar that does not involve other people, and therefore is quite flexible. So neither calendar, and the two of them are the base calendars of that I think 90% of people use the one of those two, neither of them have Sorry, I should also include Apple calendar. But neither neither of the three, none of the three is geared towards people who do time blocking. And that's a critical activity that we must use in today's world, especially when you're working from home, you have to think very carefully about getting into the flow state. For

example, creating time when you can do focus work away from family, kids, pets, distractions, meetings, calls, all the things that would disrupt you from doing your best work. And there's no provision, no real provision in any of these software programs for individual time blocking. And it's a kind of an inside joke among those who do time blocking because we're trying to do we're trying to force the calendar to do something it really wasn't designed to do. The creators of it, never intended for it to be used in this way. And here we are an even worse, there doesn't appear to be any effort by any of these three companies to do any kind of customization for time blocking. It's as if they've kind of said, Okay, this is enough, we moved on to more interesting things. And that's that has its own crazy logic. But again, back to the main point, designers and developers who think and come up with ideas that they think are pretty bright ideas end up causing problems for lots of many of us just because their theory or the idea they have in mind is just not robust enough.

Raymond Sidney-Smith 4:46
So I'll take your bait, and I'll look at this through the lens of Google and Google Calendar, which is that Google has given those of you who are time blockers The ability to add a calendar event, which is without anyone else, you know, you're not inviting anyone else, you can move it, you know, each calendar event is fungible. And they've even gone the next step of being able to add tasks at specific dates, and times, as well as all day task events in the calendar interface. If you were a developer or a product manager at Google, what would you tell them to do differently?

Francis Wade 5:30
Drag and drop? That's a simple one, simple drag and drop tasks and move them around so that you can manipulate your calendar will without having to go in and edit each individual light. So you can already do that drag and drop. Yeah, I'm gonna try that, right.

Raymond Sidney-Smith 5:47
Absolutely, yeah. Yeah.

Francis Wade 5:49
from day to day, or just

Raymond Sidney-Smith 5:50
yeah, day to day, week to week. Oh,

Francis Wade 5:53
I stand corrected. But still, the interface is just not meant for that purpose. It was invented and created and put into the suites at a time when these were afterthoughts. And they haven't done much develop mental them even since then they've tried. Canada tried to put a Google Calendar tried to put in Google goals. Google Yes, goes right, barring from a scheduling software. But it clearly this wasn't meant for that. And that that particular feature is kind of just dropped off the calendar, I still use it.

Raymond Sidney-Smith 6:30
You know, the funny part though, is that the the Curiosity is in Google goals, when you set a goal that's useful inside of Google Calendar inside of Google Calendar, the mobile application, but when you take it outside of that, say you put your you add your calendar, to Microsoft Outlook, or to woven or to another tool like that, you it then disappears, it's not a part of the feature set. And the reason for that, in my limited understanding of it is that calendars and what is a calendar infrastructure is ancient technology, we're using this old format. And so for greatest interoperability, where we're basically depending upon the legacy structure of how calendars were developed. And so all of this new technology, kind of like SMS, right, we deal with SMS, which is a terrible infrastructure system. But because it works currently, across all of

the various telecommunications carriers, we continue to keep using it. And we know that RCS is the better standard, and we should all get there. But for some reason, we keep using the lowest common denominator. And that's a little different for me than email, which email is the right, interoperable standard, right, everybody uses, it's a part of the internet. And we all know how to make an email function. And what we build on top of that is really where developers start to have more say, hey, which is the new email app, email service from the folks who create Basecamp? You know, they're creating a highly opinionated email software on top of an email service, and it's still email, right? The the infrastructure is solid, the what you put on top of that really makes a difference. And I think here with calendar, just in this specific instance, Francis, I think calendar is I think we just need a better system, like, for example, calendars, that the natural ability to create a feed for your calendar, right, so that I can send you create a calendar and send you a series of events, without having to jump through hoops and make all of these, you know, you just can't do it. And that means that if I were to be a time blocker and want to assign a project to you, I'd have to create multiple events, repeating events, and share those repeating events with you. And all of this kind of nonsense, where if we were across systems, it would it's just, it's still a bit of a pain. So I can see the how the infrastructure part is the part that needs to be fixed here. agree, disagree,

Francis Wade 9:20
right? I have a deadline every two weeks of producing a, an article for the newspapers. Some I'm on contract, but I follow the almost the exact same process each week. Or what I'd love to do is to be able to take the same process the five or six different blocks of time, and move them as our cast them as a project. And then be flexible about what I do with the project. So whenever I wanted to write something, I'd follow the same process. So that's just me and me, but same applies for me and somebody else. That capability doesn't exist.

Art Gelwicks 9:54
Great. So let's talk about the software that we deal with and the issue That we come across akin to what Francis is talking about different types of productivity frameworks methodologies require our software to do these things that sometimes they're not supposed to do, or they should have been designed to do for productivity. But clearly, they missed the mark, we have to divide a line here. There's two types of development. And we have to be clear, there's large corporate development, and there's small developers, small developers are typically playing multiple roles, they're actually writing the code, they're doing the UI, they're coming up with the information architecture, they're actually acting as the product manager itself. So they have to wear all those hats. In a corporate space, a lot of times the developers have no say in what they're building, they have a say, from the standpoint of how the code will actually be written. But the functional requirements are often dictated to them. And in many cases, they get ignored when they try to push back on them. So you have to be clear when you're talking about who when we say developer, let's be really clear as to who needs to accept the responsibility for software stewardship in this. When we're talking about building software to meet requirements. It's whoever's defining the requirements is the responsible party, if those requirements are defined, and then buffaloed, when they're built, that falls on the developer, but getting the requirements right in the first place. That rarely is the developer's responsibility, unless it is that small of a shop where you have a developer playing multiple roles. So with that said, if we use the Google example, the developers build what they're told, the business operatives and the requirements gathers the project managers and the product owners, they're the ones who have to understand the concept of this is the thing that we want to build. Often, though, they will not build it with a solid justification. There's a big dividing line in my mind between building to a strategy and building to an audience, they will often use the classic model of well, the customer's always right, this is what the customer wants. So this is what we will give them. And I'll be honest with you, anybody who's ever coded anything has in their list of stories, things they have been told to code that they knew, as soon as they were writing it, it was not going to work, and it was not going to work well, and it was going to be ineffective,

and it was going to be painful, but they were pushed into doing it. And it's that type of mindset, to not leverage the expertise of the developers who know when this stuff is not going to put together properly, that we get. And I'm going to phrase it this way crappy software, we get software that meets a specific individual use case, or we get software that tries to be everything to everyone and does nothing Well, we've all seen this software out there. And Matter of fact, every application has aspects of that there are compromises that are made in the design and the development. And when we have to go through as a software developer, when a software developer has to go through and take those requirements, which are I'm going to say 60 to 80% of the time complete, maybe are 60 to 80% complete, I mean, there's always huge gaps in the understanding of what the functionality should be. And then it becomes a given take back and forth between the developer trying to put together something that they can actually deliver, and support, which is the other thing we haven't talked about, and meeting these ephemeral requirements for software building. I mean, we're doing it right here. As we talk about this, we're comparing applications to each other, which we have to do, because that's our way of seeing what the functionality could be. But we also have to understand that those applications are completely written differently underneath. They use entirely different platforms. I run into this every day. Because I'm building applications right now, that used to be an infopath, the Microsoft based product, and now have to be in power apps. And over and over again, I will get well it needs to look like it used to. And my answer is very simple. It can't, it cannot the platform will not do that. And we have to as software consumers, we are just as responsible to understand a little bit about what we're going to be running the application on. I mean, if you want a perfect example of this, the people who complain that the iOS app doesn't look like the Android app. No, it doesn't, and it won't and it never will But we're not willing to say, Well, you know, just make it do that, no, you can't do that. And anybody who makes that argument, I will say, has never written a line of code in their life.

Raymond Sidney-Smith 15:10
So you bring up a couple of important points here, which is that software development doesn't happen in a vacuum. And for most software that's developed outside of really the open source software community, for the most part is a business. And so when there's lack of business planning, lack of understanding of the business, and many times lack of a revenue model, problems ensue.

Augusto Pinaud 15:34
Thank you make us I agree with you those those important points. And also, as you said, it is it is a set of compromises, you know, as you need to consider the business, the budget, what do you think that may be a perfect feature for one user, but what is going to be used for the majority of them, and in many cases, some of these things are guesses, you know, on on, on our hard to define software hardware, it doesn't matter what you build, but especially in software plus, then you need to add another element that is how capable technologically speaking is the person you are talking to you are working on is going to be the end user of this because it's not the same capabilities that I can throw to a person who have advanced use of software and technology that I can throw to my parents, you know, we were setting up over the weekend, some lights with Alexa to my parents in remotely, okay. And it's not the first time I do it. But this was was this time was especially challenged, because it was something new for them, that they were talking to this thing that they were doing some switches, and then they need to do two step instead of one. So is the software change to the worst? Now, the software has been pretty much the same. But now the user, if you will have asked my parents, they will have come and told you Well, this is way too complicated, why I cannot turn on the switch. And it was not after it was all done that he agreed that it needed to be done. But for most people, as soon as they hit that barrier, or seems a little complicated, because it's new, they will go and revert uncombed a complaint to why this doesn't work as version 1.0 works. So I think it's difficult. The other part is how much of this guidelines and a staff will limit then they

innovation and and I understand sometimes the version 1.0 that came us innovation, it's terrible. But it is that initial bold step that produced later on? What is the real innovation? Or what is a more polished version of that innovation? You know, if if we remember when Apple decided to remove the floppy disk out of the computers, if you go and look at the articles on the web at that time, people were horrified I was possible this is happening, how we're going to do and now if we bring a floppy disk to the discussion, people will look at us like, Really? You want to use that for what? So I think there is a fine line between what is that guideline? And how much that guidelines will stop that innovation part and that risk take, Hey, where's mistakes? No question about it. But it's part of what will produce that curious to hear from Arthur

Francis Wade 18:50
a little bit on the specific problems that developers have been complaining about? Is it is it the case that they're complaining that the software won't work because of technological reasons? and infrastructure reasons? Or is it because it won't, it won't satisfy the users need? And what's if there's a split between those, what's the percentage like

Art Gelwicks 19:17
now let's see, often what happens is you'll get, you'll get a set of requirements, let's take something that already exists out there, your drag and drop in Google Calendar. That was originally not a feature that was something that was added on after the fact. Now, as a developer, there's two ways you could look at that it would have been a heck of a lot easier to implement that from the very beginning. But you could make the argument that that was an unknown requirement when Google Calendar was initially released. Nobody knew that drag and drop in a calendar would be a useful thing. So had to be added after the fact. Often what will happen is that requiring will be passed down to the developers to say hey, we want drag and drop in the calendar. Now they have to go back and backtrack on decisions that were made to be able to implement the calendar in a consistent, reliable, supportable manner in the way it was originally built, to now include functionality that could likely either significantly change or break the underlying structure of how that calendar works. And to acoustics point, you still have to support the people who use it the old way, unless you're going to say, guess what the old way is gone now. So now this, this request, which may be fit, be targeting, I hate to say it this way, but a very noisy minority, rather than the bulk of the people who are going to be using this functionality. or using the other functionality, you now have a situation where you're trying to meet a requirement that can push the whole thing over if you're not careful. And I can use the example of the Windows operating system. People have complained over the years, you know, Windows is terrible Windows is terrible. And one of the biggest challenges that Windows has is it has so much legacy garbage in it from every version previous that it has to carry forward. And the developers have to not only build this new demand of functionality, but they have to support these old legacy requirements, to be able to meet all of these customer needs. And the developers literally caught at the back end. Let's let's give a different analogy. You go into a restaurant, instead of having a menu and you choose the meal you want. And the chef preps the meal and brings it out to you. How about we do it this way, we give you a list of ingredients, you pick and choose what ingredients you want, you send that list back to the chef. And then you have him send out something and tell him if that's what you want it. Because that's what we so often do to developers, we give them these massive lists of functional requirements. And then we don't give any sort of credence to do they even make sense with each other user experience, user interface design, these are all sciences that are necessary to provide a buffer between the code base and the and the actual user base. Because so often what we've asked for, and we've asked the developers to do, doesn't translate translate into person. We need some sort of a connector bridge between the two. So when you talk about developer frustration and trying to meet developer requirements, if you want to see the experience, just look at the comments on any application in either the App Store or the Play Store. The bulk of the comments will be well, why doesn't it work like this

way? Why doesn't it work like this other program? Why did you change it from what it was? Very rarely will it be? That works the way I expect? Why? Because everybody has different expectations. And this is again, pushed back on the developers. So when we think about software stewardship, and we say, you know, we need standards around this and things like that. Here's a great XKCD thing. We've got 14 standards. Oh my goodness, we've got 14 standards, we need a standard for that. Guess what we now have 15 standards, we cannot as developers, and I'm going to put that mantle on and I and I wear it with great hubris because in no stretch, have I been a developer for any number of years as of late. It's been years since I've done done a serious development work. But I do sympathize with the challenge that they have developers have to have the latitude to push back and say, know, what you want to do won't work won't be effective. And you won't have satisfied people because of these things. Could be a technological reason, like you said, could be that the platform won't outright support. It could be that even if the platform will support it. It's just going to be an awkward and unpleasant experience because it how it has to be implemented. Or it could cause performance issues. Think about your web pages. How if you go to a web page, and it takes more than eight seconds to load? how likely are you to click off that web? how likely are you to sit there as a user and go, Oh, this is slow. This is terrible. You know how hard it is to write a comprehensive, complete website that every page renders in under eight seconds. It's ridiculously difficult because so many of the factors are outside your control as a developer. But we as the productivity community, are happy to fill up the front of the developers pipeline with this requirement and that requirement, this requirement and then turn around to the And say, Well, you know, if you were just more productive, you could handle all this throughput. You could use konban more modeling and agile structure in scrims to get get all your work done. When can we go to testing? Well, when it's done, no, no, no, no, we need to go to testing now. Again, if you're not the chef cooking the meal, you really don't have the right to keep running into the kitchen and saying, Is it done yet? So where's the developers challenge? The challenge is balancing, balancing between what they can do, because in many cases, I'll admit, it's kind of like high school, you're about one page ahead of the teacher, you're figuring this stuff out, as you're trying to implement it in many cases, which doesn't get taken into consideration either. But also, you're trying to take what they've given you and interpreted into something that can be implemented. I'll give you another example. language translation, think of it this way, somebody comes in and starts speaking fluent German to you. You don't speak German. You have to talk to a translator to find out what they're saying. Is there going to be an exact translation through? Probably not, there's probably going to be some variances in the translation. Why? Because of the language differences, because of the conceptual differences between the cultural differences. But we don't take that into consideration. As the receiver of the message, we say, Oh, well, that's not what he said. He says, Who says how we have to be a willing participant in the process. With the developers, this cannot be an adversarial relationship. And when we look at this type of work that we're outlining for them, and we look at this type of a thing, if we want developers to be best served, and to give us the best features that we possibly could have, and the strongest, most reliable software, I'll tell you, we got three jobs, one, think through what we're asking for before we ask for it, to listen to them, when they say how complicated it will be. And three, be willing to work to a point of compromise. If you decide that you want everything to be fuchsia and yellow, in your application, and the developer goes, No, that's that's horrible. work to a point of compromise. Let me change that example. You decide you want everything to be red and green, because it's going to be a holiday application. And the developer turns around to you and says, that's fine. But there's a portion of your audience that's colorblind, they're not going to be able to use it. Do you as a product owner, as a cut as a client as a just general user out there go? No, I want it to match the holiday customers? Are you going to give that developer the opportunity to push back and say, no, it's a really bad idea? Or are you going to force them just to implement something? And then when you have customer feedback that says, Oh, I can't see this? Are you going to go yell at the developer, it's really a no win situation at times that we place developers in. And we have to seriously take this into consideration. We do ourselves a disservice in the

productivity community, when we do not take into consideration the challenge of the work we are asking to be done. And this applies not only to software development, but it applies to anything teamwork, projects, collaboration. It's not a go do it. It should be a, what's it going to take? And how can we make it happen?

Augusto Pinaud 29:00
I love that that analogy was a food. Okay, let me give you the list of ingredients. Let me tell you and bring me exactly what I were thinking when I select that list of ingredients. And I agree that is the problem. Many developers get pushed by management, they get pushed by all these other divisions, and then comes a user of the end of the of the thing and now complains about the fact that who thought about making jello and friction. And on top of that, it's usually a job that is done with not necessarily abundant resources. And I'm talking about time and I'm talking about requirements and I'm talking about urgency on on my experience dealing with developer can projects that require development, you know, none of them said Okay, take your time developer to figure this out. Okay, then the project, prospect instead, come up. We should have released this A week ago. That's that's more of the conversations I don't give you to combat agile, or what do you do on your development shop, most of the conversation is why this is not out already. And that is also an issue that sometimes make developers even great developers cut corners. Hey, it works. reasonably. Okay, let's release. So I agreed with a lot of what you said on an understanding the problems on both sides, the shortcomings of resources, and planning and pressure or an excessive pressures the developer side has, I'm coming from the other side, that is a wishful thinking where I am willing to put myself first on that list of as guilty, okay, give me a piece of software, and I will find something that I want that software to do doesn't matter what it is, okay? that it doesn't do, or it can do better, or it can do differently? Where can we find are there is a middle ground for all this? Or no, really, it is a problem that has no solution?

Art Gelwicks 31:09
Well, and I don't know that I want to say that there's no solution to it. Granted, there is no absolute solution that makes the problem go away. But one of the things that we can do, and we talk about this when we when we're not talking about software development, and that's understanding the project that we're trying to do. Most projects, wind up, if I split it into three phases, design, development, and deploy, which includes testing. The the discovery design part of the project should comprise the majority of the project. I know some, some project managers out there screaming yell at me about that. But I firmly stand by that. I think if you have more than half of your project, focused on the requirements gathering and the testing, and the proof of concept models and the iterative design of getting to something that that customer is going to be happy with receiving, then the second half of the project becomes much easier. Because at that point, you can literally build what you've all agreed to. Does that happen? Rarely. So often, what we will do is we will put half baked requirements out there, say build something to those requirements, we'll look at that. And we'll say, well, that's not right. And change and not realize that not only are we pushing back on what was constructed from ill conceived requirements, we're now changing the requirements completely. And it winds up becoming this back and forth battle between the client, project manager, business analysts, whatever, and the developers who are trying to meet their need sets. We have to spend our time understanding what we want things to do. And specifically why this is one of the things I tell every developer I work with on all of my teams. I say, if somebody asks you for something, and that little bit of hair goes up on the back of your neck, you know, your your developer, spider senses are tingling, you're going this doesn't sound right. ask why they want it. Because if they can't tell you specifically, what's the benefit of that, then they can't justify the effort in putting it in place in the first place. And I think we as I said, we do a disservice to developers, because we can't so often define why we want something, or, and this is probably the most frustrating thing. We'll say we want it because this other software hasn't. That's not a reason we wouldn't accept from productivity clients. The answer of well, why do you want to do that?

Well, it's because we've how that's how we've always done it. We know that's a terrible answer. And we will call people to task on that all day long. But yet, we'll put requirements out for a developer with no substantive support as to why we want those functionality. What is the benefit to the client. And the reason why that is so critical is if you want a developer to be able to truly support you and be motivated to get you what you want. They have to understand what's the benefit, they have to buy into it. You can't just give them the list. A chef who's going to get this ingredients list from you is going to go Okay fine, I'll make you something. But if they know if they understand what you're trying to get to that you want to savory meal that you want something with steak and with vegetables Potatoes not great. They know what you want them. And they can make decisions as part of the development process or part of the cooking process to get you where you want to be based on their experience. And that's, that's probably the last thing that I think is is what gets most stuck in my craw about this is we tend to ignore the experience of our developers in developing good software. We look at developers from the standpoint of, can they develop quickly? Can they write something based to our spec and get it delivered yesterday? Sure, any developer can give you a crap piece of software, and meet your deadlines. But that's their reputation on the line for doing it. And they don't want to do that they want to write the best stuff. truly good developers are closer to artists than they are engineers. They're working in this medium of digital code and creating things that nobody else can create. But you don't go to Picasso and ask him to paint your house. You let him do what he does. You tell him what you like, what inspires you what is the end goal you're trying to get to, and allow his creative instincts to go, one of the best developers I know, she is wonderful at what she does. And so often I see her get constrained from being able to use those skills, because she's just having to meet these black and white straight requirements, with no flexibility to be able to deliver the best thing possible, whether it be by requirement by time constraint by budgetary constraint, whatever,

Augusto Pinaud 36:51
I think you bring a really second important point. That is how we have over time, modify those timelines. And if we look into suffer, we were expecting updates and upgrades and improvement, even newer versions, in a much, much longer cycles than what we expect now. Okay, even you could see on many applications, how the updates are now happen weekly. Okay. And from the other side, from the testing side on all that? Well, no wonder it seems like most applications are more buggy, but it's not that there are more buggy is that the times we are using for testing are also being shorter. And at times, we are expecting on people he suspected that we are going to come with innovative solutions every three weeks. And I don't think that's even possible not for a developer not for any.

Art Gelwicks 37:51
Now, if you look at it from the standpoint of say, like an agile model, where you're dealing with Sprint's two weeks cycle, continuous development, great. If you truly adhere to that model, you may be able to do that. But at some point, you are going to be inheriting technical debt, because of just what you mentioned earlier, at some point, you are going to have to take shortcuts, you're going to have to make technical compromises to meet unyielding business or client side decisions. And let me just be clear, there will always be those there will be times where you have to compromise as a developer, because they have just stuck their feet in the ground said, Look, I'm not moving, this must be in this on this date, fine, I'll do whatever I have to do to make that happen. But when we look at it from the standpoint of this kind of a cyclic deployment of functionality, how often are we taking into consideration this new feature and how it impacts previous and one of our favorite productivity tools out there is has been fighting this battle for two years, notion has been fighting this uphill fight to release an API to their environment. So people can interact with it from the outside. They can write their own interfaces, do whatever with the data that lives within notion. And all they do is they get flack over the fact that they have not released an API yet. Everybody's going well, you know, you promised that you promised it. You promised it. The people who are complaining about this

have no idea how difficult it is to write an API. I'll start with that. But secondarily, I have to go back to my previous question. I just have to say, why do you want it? What are you going to do with it? And is it 5% of your user base that's going to use this thing that's going to suck up 80% of your development resources to get out the door? Well, if it is, from a business standpoint, it doesn't make sense to dedicate resources to it. I could make that argument with money and they're the ones who are going to

Raymond Sidney-Smith 40:00
You know, hold her up arms up in the air when there's a security problem, or there's some kind of, you know, missing element in the API.

Art Gelwicks 40:08
And that's, that's exactly a you think about. And this. Again, this is another aspect, when we are developing software to be run on individual computers machine or individual users computers. That's one thing. you deploy software out there, people do their updates, and then half of the people complain, because it's not compatible with their version of Windows 95, that they're still running. Okay, fine. That's one thing. When you're talking about cloud based software, this is a whole different ballgame. Because if you botch cloud based software, you bought it for your entire client base, everybody using it. And these are not things that you can say, okay, we're just going to turn it off for the weekend. Personally, I consider any productivity tool that you use more than once a week, a mission critical application, because when you need it, and it's not there, there's a huge problem. So when you take applications, like a notion, or you take applications, like an Evernote or a OneNote, that have a mix notion is more pure cloud. But if you take something like an Evernote where there's some local, there's some in the cloud, now break the cloud, because of something slipping through the cracks, some requirement that didn't get completely refined and pressure got pushed through the development group and didn't get successfully tested. And all of a sudden, the world comes screaming down, that you guys totally broke your software. Well, okay, fine. If you want us to make sure that will never happen, then our deployment schedule of of features is going to be on a six month basis. Oh, no, you can't wait that long between new features. Okay, cake and eating it. This is where we're back to, we have to understand that. Just because we can use software, we don't know how to build it doesn't mean we know how to build it. I'll use the analogy of a car. Just because you know how to drive a car doesn't mean you know how to swap out the engine? And I don't suggest you try.

Raymond Sidney-Smith 42:18
I want to switch gears and flip this from the perspective of the person using the software. How do we use this information? To be more productive? How do we by seeing behind the curtains, that developers and businesses that are developing software, free open source software development groups? As we approach being more productive ourselves? What do we do to be more productive? How do we make this work for us? I'm a personal productivity enthusiast. I want to be more productive with my software, software stewards. Those are the developers, they control so much of my world today. What can I do to take best advantage of something that I actually don't have a lot of control over? Is that in the choices of the software I make is that in the way in which I use particular software, what do I do with that information?

Francis Wade 43:23
So I chase down developers of the latest auto schedulers, and tell them Oh, Listen, do you know there's eight others out there, and they go, really, they haven't even, they haven't even done the basic research around the app that they want to develop. So there's just basic information that the web is making the decisions about putting the software into play. This is basic stuff they haven't done, let alone digging into the behavior that they're trying to change. And what what new behaviors are going to resolve from the software, which is an even tougher question, their way back at the beginning, you know, chasing an idea, and thinking, Okay, let

me just keep going and keep going and keep going. And we see what happened. I think the same happens, even for the big companies is that, like Blackberry, for example, I don't think blackberry envision that people would drive and text at the same time that just wasn't in a behavior that they imagined would happen. And it did turn out to be incredibly dangerous. And I think, at the very least, that we can do is give feedback to developers and c are designers and developers and say, hey, look at these new behaviors, or look at what already exists. Or look at the downside of this. Have you thought of that at this? That's really basic, though.

Art Gelwicks 44:40
Well, and I think you're right there. And I don't want to totally pound on. small independent developers, as I said, have to fill multiple roles and it can be a challenge for them to get a total landscape understanding as they're trying to develop their software because they're literally having to be heads down writing code, and making Sure things don't break. So if we want to serve them well, viewing the developers as partners in the software we want to have is critical. We need to reach out to them, we need to engage with them, we need to give them those examples that you mentioned, you know, there's eight others of these out here, and they do it this way. And don't do it from the perspective of getting something out of it. But doing it from the perspective of a partnership to make the best software solution possible for that is what you really is the way to build that bond, to develop to a level of what you really want, and get what you want. So that would be my first thing. Absolutely treating it as a partnership. Second, don't assume you're going to get every feature you asked for. That's one of the biggest things just because you asked for feature B, okay, if they say no, it's not gonna happen. Nothing wrong with asking why. But be willing to accept that because honestly, you're not cooking the meal. So if they're out of asparagus, they're out of asparagus.

Raymond Sidney-Smith 46:14
My one action, and I've said this before on prior episodes, which is that everyone should learn the fundamentals of programming. If you are going to approach software, if you're going to be a good global citizen in a modern age, you must understand the tools you were using. I mean, it, it's akin to like, taking, I don't know, if anybody here, you know, around the table, the virtual table here has flying experience. But it would be like any one of us being put in the cockpit of a 1985 Cessna, and being like, go fly it, like there's bound to be a problem. If you even get it off the ground, you were likely to also crash it. You know, like that's, that's what we're doing here, we're taking the largest portion of our workforce today. And we are telling them to go fly a plane with tools with a dashboard, they don't know how to read that they're not used to, in using for the right reasons. And when they do look under the hood, it's gobbledygook, right? You know, open up the engine of your of your car today, it's all electronic run, you don't really know what parts are what. And we, we need to be able to understand it not, I don't I don't need you to be coding. But I do need you to understand how it works. And that's the part that I really harp on. There are a lot of courses out there that go over the fundamentals of programming just gives you an understanding of how modern code is built, and why it why things are done that way. And once you understand, oh, yeah, that's a fuel injector. Oh, that's the, you know, the exhaust that over here is what's running the air air conditioning, you know, the the climate control in the vehicle, once you start, you start to understand where the pieces are, you're much more likely to say, Okay, this is the most efficient way for me to be able to use this thing, right, you know, the wheels on the on the on the ground. You know, here it goes, how I sit in the vehicle and operate it properly. But underneath the hood there these things happening. Sorry to mix metaphors with playing and car. But you know that my whole point is that the better we understand what's going on under the hood of the technology, the more likely we are to one be understanding when things go wrong. We're also better understanding of how to troubleshoot when things go wrong. And we're probably also better off approaching the software with a level of greater understanding about its features, when we do when we have an understanding of how software is built, but also that specific software that you're using. So one is learn a little bit about the fundamentals of programming that is there is a I've

said this before, but I'll say it again, and I'll put a link to in the show notes. There's a lynda.com course that is called the fundamentals of programming or something like that. And it is a fantastic course and just understanding Okay, this is how stuff is built there. There's like text, and it is run through a compiler. And it outputs instructions. And computers are very literal, even when we talk about machine learning and you know, any other type of artificial intelligence they are. They are literal creatures, and they are reading instructions. And so every time you interact with a computer you are you are giving it instructions and then it's spitting those instructions back out. And sometimes those two things are incompatible and how does a computer deal with those things how to software deal with those things that really is, like powerful to know. And then the other side to that is learn the software you have? Well, I mean, I have a, I have a rule, which is, I should know all the limitations of my software because I've experienced them. And that's when I know I know my software well enough. And I can be most productive. When I have basically seen the edges of, of the software, all of its deficiencies actually gives me control, to be able to not go over the edge where my productivity is going to be impacted. If I know that I'm constantly trying to get it to do something, I can't get it, I don't need it. I know it can't do by No, it can't do what I want it to do, then don't go down that road, do what you need to do to be productive without going in these directions that are are not purposeful or useful.

Francis Wade 50:52
The problem is not it's not it's not a development problem in the sense that the guys who do the coding are at fault. It's if anything, it's a conceptual and organizational problem where whoever, whoever thought of the idea of doing a particular piece of software needs to sort of own this bigger world in which they're influencing user behavior. They are introducing new habits, they're interpreting people's actual needs, and then trying to translate that into software. At the other end, they need to be very tentative about who's going to be implementing this is it implementable? And even as they implemented because this is what I imagine happens, the developer goes back to the designer and says, Do you really want this, this or this, because you can't have all three at the same time. And the designer, designers thinking hasn't gone that far. And they kind of give them a blank look, because they actually don't know. But the interplay that are talked about that needs to happen between the designer and the developer, could clarify a whole lot of problems that and resolve a whole lot of issues that we as users sometimes experience once the product is finished. And we experienced them because they weren't resolved sufficiently during the design process. They were just kind of punted, or they're just kind of left in there, and then we get what comes out at the end. So I think there's a whole there's a whole enhanced way to design and and i would say design, rather than develop.

Art Gelwicks 52:34
Yeah, I'd say the simplest thing to do. involve your developers early. And often. The earlier you can get them involved in the process of designing your software, the more likely you're going to have a successful software built, because they can keep you from designing in mistakes. And they can give you ideas that you may never have considered from a design perspective.

Raymond Sidney-Smith 52:58
That brings us to the end of our discussion today. But it doesn't mean that discussion or the conversation needs to stop there. If you have a question or comment about what we've discussed during this cast, please visit our episode page on productivitycast.net. there on the podcast website at the bottom of the page, you can leave a comment or a question and we will be able to read and respond to those comments or questions. They're also on ProductivityCast dotnet. On each episode page, you'll find our show notes, links to anything we discussed. If something's missing, leave a comment and we'll add it along with our text transcripts to read and download their PDF. So you can go ahead and download they are machine generated. So they're not going to be accurate to like you know, 99.9%, but you can get the idea of what

we're talking about and jump to those sections in the audio. So it's really useful for those references. If you have a topic about personal productivity you'd like us to discuss on a future cast, please visit productivitycast.net forward slash contact, you can leave a voice recorded message it'll record it right there in your browser, either on the desktop or mobile and shoot that over to us. Or you can type us a message you know, with your fingers on the keyboard, virtual or otherwise. And you can then send that along to us. And maybe we'll feature that in a future episode as a topic or in one of our mailbag episodes. I want to express my thanks to Augusto Pinaud, Art Gelwicks and Francis Wade, for joining me this and every week on productivity cast. You can learn more about them and their work by visiting productivitycast.net as well. I'm Ray Sidney-Smith and on behalf of all of us at ProductivityCast here's to your productive life. That's it for this productivity cast,

Voiceover Artist
And that's it for this ProductivityCast, the weekly show about all things productivity, with your hosts, Ray Sidney-Smith and Augusto Pinaud with Francis Wade and Art Gelwicks.