FRED TALKS

12th April 2026

CONTENTS

The Night Watch

JAMES MICKENS · 11 APR 2026 · SOURCE

The Night Watch

J A M E S M I C K E N S

James Mickens is a researcher in the Distributed Systems group at Microsoft's Redmond lab. His current research focuses on web applications,

with an emphasis on the design of JavaScript frameworks that allow developers to diagnose and fix bugs in widely deployed web applications. James also works on fast, scalable storage systems for datacenters.

James received his PhD in computer science from the University of Michigan, and a bachelor's degree in computer science from Georgia Tech. mickens@microsoft.com

s a highly trained academic researcher, I spend a lot of time trying to advance the frontiers of human knowledge. However, as someone who was born in the South, I secretly believe that true progress is

A

a fantasy, and that I need to prepare for the end times, and for the chickens coming home to roost, and fast zombies, and slow zombies, and the polite zombies who say "sir" and "ma'am" but then try to eat your brain to acquire your skills. When the revolution comes, I need to be prepared; thus, in the quiet moments, when I'm not producing incredible scientific breakthroughs, I think about what I'll do when the weather forecast inevitably becomes

RIVERS OF BLOOD ALL DAY EVERY DAY. The main thing that I ponder is who will be in my gang, because the likelihood of post-apocalyptic survival is directly related to the size and quality of your rag-tag group of associates. There are some obvious people who I'll need to recruit: a locksmith (to open doors); a demolitions expert (for when the locksmith has run out of ideas); and a person who can procure, train, and then throw snakes at my enemies

(because, in a world without hope, snake throwing is a reasonable way to resolve disputes). All of these people will play a role in my ultimate success as a dystopian warlord philosopher. However, the most important person in my gang will be a systems programmer. A person who can debug a device driver or a distributed system is a person who can be trusted in a Hobbesian nightmare of breathtaking scope; a systems programmer has seen the terrors of the world and understood the intrinsic horror of existence. The systems programmer has written drivers for buggy devices whose firmware was implemented by a drunken child or a sober goldfish. The systems program- mer has traced a network problem across eight machines, three time zones, and a brief diversion into Amish country, where the problem was transmitted in the front left hoof of a mule named Deliverance. The systems program- mer has read the kernel source, to better understand the deep ways of the universe, and the systems programmer has seen the comment in the sched- uler that says "DOES THIS WORK LOL," and the systems programmer has wept instead of LOLed, and the systems programmer has submitted a kernel patch to restore balance to The Force and fix the priority inversion that was causing MySQL to hang. A systems programmer will know what to do when society breaks down, because the systems programmer already lives in a world without law.

Listen: I'm not saying that other kinds of computer people

are useless. I believe (but cannot prove) that PHP developers have souls. I think it's great that database people keep trying to improve select-from-where, even though the only queries that cannot be expressed using select-from-where are inap- propriate limericks from "The Canterbury Tales." In some way that I don't yet understand, I'm glad that theorists are investigating the equivalence between five-dimensional Tur- ing machines and Edward Scissorhands. In most situations, GUI designers should not be forced to fight each other with tridents and nets as I yell "THERE ARE NO MODAL DIA- LOGS IN SPARTA." I am like the Statue of Liberty: I accept everyone, even the wretched and the huddled and people who enjoy Haskell. But when things get tough, I need mission-crit- ical people; I need a person who can wear night-vision goggles and descend from a helicopter on ropes and do classified things to protect my freedom while country music plays in the background. A systems person can do that. I can realistically give a kernel hacker a nickname like "Diamondback" or "Zeus Hammer." In contrast, no one has ever said, "These semi-

transparent icons are really semi-transparent! IS THIS THE WORK OF ZEUS HAMMER?”

I picked that last example at random. You must believe me when I say that I have the utmost respect for HCI people.

However, when HCI people debug their code, it’s like an

art show or a meeting of the United Nations. There are tea breaks and witticisms exchanged in French; wearing a non- functional scarf is optional, but encouraged. When HCI code doesn’t work, the problem can be resolved using grand theo- ries that relate form and perception to your deeply personal feelings about ovals. There will be rich debates about the socioeconomic implications of Helvetica Light, and at some point, you will have to decide whether serifs are daring state- ments of modernity, or tools of hegemonic oppression that implicitly support feudalism and illiteracy. Is pinching-and- dragging less elegant than circling-and-lightly-caressing?

These urgent mysteries will not solve themselves. And yet, after a long day of debugging HCI code, there is always hope, and there is no true anger; even if you fear that your drop- down list should be a radio button, the drop-down list will suffice until tomorrow, when the sun will rise, glorious and

vibrant, and inspire you to combine scroll bars and left-click- ing in poignant ways that you will commemorate in a sonnet when you return from your local farmer’s market.

This is not the world of the systems hacker. When you debug a distributed system or an OS kernel, you do it Texas-style. You gather some mean, stoic people, people who have seen things die, and you get some primitive tools, like a compass and a rucksack and a stick that’s pointed on one end, and you walk into the wilderness and you look for trouble, possibly while

using chewing tobacco. As a systems hacker, you must be pre- pared to do savage things, unspeakable things, to kill runaway threads with your bare hands, to write directly to network ports using telnet and an old copy of an RFC that you found in the Vatican. When you debug systems code, there are no high- level debates about font choices and the best kind of turquoise, because this is the Old Testament, an angry and monochro- matic world, and it doesn’t matter whether your Arial is Bold or Condensed when people are covered in boils and pestilence and Egyptian pharaoh oppression. HCI people discover bugs by receiving a concerned email from their therapist. Systems people discover bugs by waking up and discovering that their first-born children are missing and “ETIMEDOUT ” has been written in blood on the wall.

What is despair? I have known it—hear my song. Despair is when you're debugging a kernel driver and you look at a mem- ory dump and you see that a pointer has a value of 7. THERE IS NO HARDWARE ARCHITECTURE THAT IS ALIGNED ON

7. Furthermore, 7 IS TOO SMALL AND ONLY EVIL CODE WOULD TRY TO ACCESS SMALL NUMBER MEMORY.

Misaligned, small-number memory accesses have stolen decades from my life. The only things worse than misaligned, small-number memory accesses are accesses with aligned buf- fer pointers, but impossibly large buffer lengths. Nothing ruins a Friday at 5 P.M. faster than taking one last pass through the log file and discovering a word-aligned buffer address, but a buffer length of NUMBER OF ELECTRONS IN THE UNI-

VERSE. This is a sorrow that lingers, because a 2893 byte read is the only thing that both Republicans and Democrats agree is wrong. It's like, maybe Medicare is a good idea, maybe not, but there's no way to justify reading everything that ever existed a jillion times into a mega-jillion sized array. This constant war on happiness is what non-systems people do not understand about the systems world. I mean, when a machine learning

algorithm mistakenly identifies a cat as an elephant, this is

actually hilarious. You can print a picture of a cat wearing an elephant costume and add an ironic caption that will entertain people who have middling intellects, and you can hand out cop- ies of the photo at work and rejoice in the fact that everything is still fundamentally okay. There is nothing funny to print when you have a misaligned memory access, because your machine is dead and there are no printers in the spirit world.

An impossibly large buffer error is even worse, because these errors often linger in the background, quietly overwriting your state with evil; if a misaligned memory access is like a criminal burning down your house in a fail-stop manner, an impossibly large buffer error is like a criminal who breaks into your house, sprinkles sand atop random bedsheets and toothbrushes, and then waits for you to slowly discover that your world has been tainted by madness. Indeed, the common discovery mode for an impossibly large buffer error is that your program seems to

be working fine, and then it tries to display a string that should say "Hello world," but instead it prints "#a[5]:3!" or another syntactically correct Perl script, and you're like WHAT THE HOW THE, and then you realize that your prodigal memory accesses have been stomping around the heap like the Incred- ible Hulk when asked to write an essay entitled "Smashing Considered Harmful."

You might ask, “Why would someone write code in a grotesque language that exposes raw memory addresses? Why not use

a modern language with garbage collection and functional programming and free massages after lunch?” Here’s the answer: Pointers are real. They’re what the hardware under- stands. Somebody has to deal with them. You can’t just place a LISP book on top of an x86 chip and hope that the hardware learns about lambda calculus by osmosis. Denying the exis- tence of pointers is like living in ancient Greece and denying the existence of Krackens and then being confused about why none of your ships ever make it to Morocco, or Ur-Morocco,

or whatever Morocco was called back then. Pointers are like Krackens—real, living things that must be dealt with so that polite society can exist. Make no mistake, I don’t want to write systems software in a language like C++. Similar to the Necro- nomicon, a C++ source code file is a wicked, obscure document that’s filled with cryptic incantations and forbidden knowl- edge. When it’s 3 A.M., and you’ve been debugging for 12 hours, and you encounter a virtual static friend protected volatile

templated function pointer, you want to go into hibernation and awake as a werewolf and then find the people who wrote the C++ standard and bring ruin to the things that they love. The C++ STL, with its dyslexia-inducing syntax blizzard of colons and angle brackets, guarantees that if you try to declare any reasonable data structure, your first seven attempts will result in compiler errors of Wagnerian fierceness:

Syntax error: unmatched thing in thing from std::nonstd::_map<_Cyrillic, _$$$dollars>const basic_string< epic_ mystery,mongoose_traits < char>, _default_alloc_<casual_ Fridays = maybe>>

One time I tried to create a list<map<int>>, and my syntax errors caused the dead to walk among the living. Such things are clearly unfortunate. Thus, I fully support high-level languages in which pointers are hidden and types are strong and the declaration of data structures does not require you to solve a syntactical puzzle generated by a malevolent extraterrestrial species. That being said, if you find yourself drinking a martini and writing programs in garbage-collected, object-oriented Esperanto, be aware that the only reason that the Esperanto runtime works is because there are systems people who have exchanged any hope of losing their virginity for the exciting opportunity to think about hex numbers and their relationships

with the operating system, the hardware, and ancient blood rituals that Bjarne Stroustrup performed at Stonehenge.

Perhaps the worst thing about being a systems person is that other, non-systems people think that they understand the daily tragedies that compose your life. For example, a few weeks ago, I was debugging a new network file system that my research group created. The bug was inside a kernel-mode component, so my machines were crashing in spectacular and vindic-

tive ways. After a few days of manually rebooting servers, I had transformed into a shambling, broken man, kind of like a computer scientist version of Saddam Hussein when he was pulled from his bunker, all scraggly beard and dead eyes and florid, nonsensical ramblings about semi-imagined enemies.

As I paced the hallways, muttering Nixonian rants about my code, one of my colleagues from the HCI group asked me what my problem was. I described the bug, which involved concurrent threads and corrupted state and asynchronous message delivery across multiple machines, and my coworker said,

“Yeah, that sounds bad. Have you checked the log files for errors?” I said, “Indeed, I would do that if I hadn’t broken every component that a logging system needs to log data. I have a network file system, and I have broken the network, and I have broken the file system, and my machines crash when I make eye contact with them. I HAVE NO TOOLS BECAUSE I’VE DESTROYED MY TOOLS WITH MY TOOLS. My only logging

option is to hire monks to transcribe the subjective experience of watching my machines die as I weep tears of blood.” My co- worker, in an earnest attempt to sympathize, recounted one of his personal debugging stories, a story that essentially involved an addition operation that had been mistakenly replaced with

a multiplication operation. I listened to this story, and I said, “Look, I get it. Multiplication is not addition. This has been known for years. However, multiplication and addition are at least related. Multiplication is like addition, but with more addition. Multiplication is a grown-up pterodactyl, and addi- tion is a baby pterodactyl. Thus, in your debugging story, your code is wayward, but it basically has the right idea. In contrast, there is no family-friendly GRE analogy that relates what my code should do, and what it is actually doing. I had the mod-

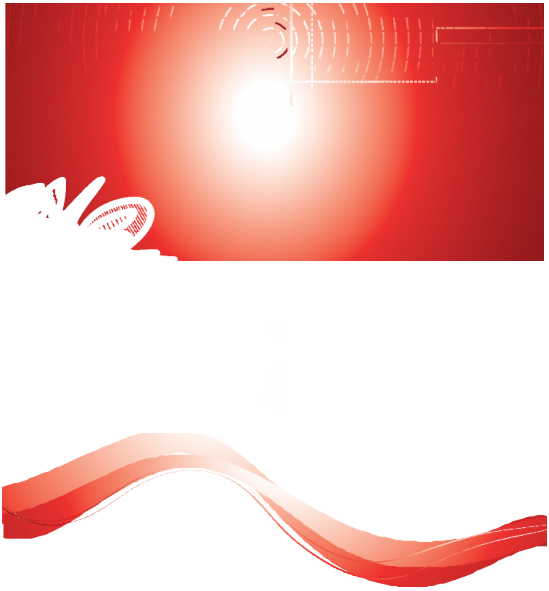
est goal of translating a file read into a network operation, and now my machines have tuberculosis and orifice containment issues. Do you see the difference between our lives? When you asked a girl to the prom, you discovered that her father was a cop. When I asked a girl to the prom, I DISCOVERED THAT HER FATHER WAS STALIN.”

In conclusion, I’m not saying that everyone should be a systems hacker. GUIs are useful. Spell-checkers are useful. I’m glad that people are working on new kinds of bouncing icons because they believe that humanity has solved cancer and homelessness and now lives in a consequence-free world

of immersive sprites. That’s exciting, and I wish that I could join those people in the 27th century. But I live here, and I live now, and in my neighborhood, people are dying in the streets. It’s like, French is a great idea, but nobody is going to invent French if they’re constantly being attacked by bears. Do you see? SYSTEMS HACKERS SOLVE THE BEAR MENACE.

Only through the constant vigilance of my people do you get

the freedom to think about croissants and subtle puns involv- ing the true father of Louis XIV. So, if you see me wandering the halls, trying to explain synchronization bugs to confused monks, rest assured that every day, in every way, it gets a little better. For you, not me. I’ll always be furious at the number 7, but such is the hero’s journey.



usenix
THE ADVANCED
COMPUTING SYSTEMS
ASSOCIATION

Do you know about the USENIX Open Access Policy?

USENIX is the first computing association to offer free and open access to all of our conferences proceedings and videos. We stand by our mission to foster excellence and innovation while supporting research with a practical bias. Your membership fees play a major role in making this endeavor successful.

Please help us support open access.

Renew your USENIX membership

and ask your colleagues to join or renew today!

www.usenix.org/membership

Notes on Hong Kong

ROHIT KRISHNAN · 12 APR 2026 · SOURCE

I recently had the chance to do a quick Hong Kong trip, for a friend's wedding. I've been before though the last time was around 15 years ago. But things have changed, and so have I. The summary opinion is that Hong Kong is amazing, and I found it a curious mix between Singapore and Calcutta. This was confusing, but it wasn't the only part that was. To wit.

1. Hong Kong is absolutely gorgeous. The sight of the skyscrapers against the mountain backdrop is divine. The best of man set against the best of nature.

2. Hong Kong is also set in the past. Many of the skyscrapers are clearly old. And many are decrepit, in an absolute state of disrepair. I was told it was due to a combination of a) people not knowing who owned what since they were built so long ago and changing hands, b) there not being any concept of an HOA or a condo society to take care of maintenance, c) common practice of splitting up an apartment into 3-4 small holes so the outside too explodes with AC units, and d) something to do with organised crime. Maintenance truly is innovation.

3. An uber driver compared it to China, where he lived for many years, saying they don't stand for this over there. They'd just demolish the old ones and rebuild, no questions asked.

4. He also said that Shenzen, a 13 minute train ride from Hong Kong, used to look like it - all mountainous and green - but when China decided to build there they demolished it to be much flatter and built on top of it. He seemed envious.

5. The density makes sense because only a small part of the islands are built up, and the house prices are absolutely insane. I was told even out in the suburbs, which are quite remote, a 2000 sqft house is like 3m USD. In the wonderfully named mid-levels it'd be 1/3rd the size.

6. Hotels are still cheaper than SF or NY or London though. By a lot. And nicer. And with nicer service. Asia is way nicer to travel to.

7. Speaking of suburbs, Hong Kong suburbs do NOT look like Singapore suburbs, which I was expecting. They are villages, especially in the New Territories. Small roads, uneven development, open sewers in some places, even older buildings, fish smell near the water, the works. This also means those places are untouched for the most part and look just like villages do in like Vietnam. And still only 30 mins driving to the city.

8. The sheer scale of building is a bit of a shock to the American sensibility. To know that sure, we can build bridges across canyons, build under water for driving tunnels between islands, roads, trains, multiple 80 storey skyscrapers next to each other on a 40 degree incline that is so steep I found it difficult to walk down. It's a testament to what we can actually do if we have the will. Inspiring.

9. Claude and ChatGPT are blocked in Hong Kong. They now have Gemini as of a couple weeks ago. When I asked my friend what they used, they said "CopIlot". I felt bad.

10. Folks routinely order things they want from China. Flowers and vases for the wedding ($0.5 cents per vase + flowers), electronics, most cars are BYD. The flip side is that ordering is highly error prone (eg some flowers didn't arrive, some came damaged) but since the price is so low it works out. Not unexpected at all but interesting.

11. You see a large number of children every time you're out - in the restaurants, in the parks, in cafes, on the sidewalk, next to the embankments - it's wonderful, counterintuitive considering Hong Kong's TFR, and again makes the US social scene seem so weird. Part of it's that houses are tiny, domestic help is common, and dining out is a default option. More broadly though, the idea that kids should be with you whatever you're doing as opposed to you make special arrangements for the kids to be looked after in your den if you want to go out seem like a clear crux. France as far as I know is the only other place where this attitude is most prevalent in the West, maybe parts of Scandinavia.

12. "There is no industry in Hong Kong, that's why I went to China", said by an Uber driver, who was a mechanical engineer, worked as an R&D supervisor in Shenzen, and moved back to Hong Kong after being forced to retire when he hit 60.

13. There are large numbers of masked people everywhere still. I think the scars of Covid 19 built on the scars of SARS and public attitudes have fundamentally shifted. Many of them wear it outdoors which is quite odd to see, or when driving alone. An interesting laziness vs prudence equilibrium.

14. The funicular tram to the top of one of the peaks is worth it. As is the hike once up top around the mountain. Absolutely wonderful. So is the Lokcha teahouse in Hong Kong Park.

15. There's a street next to the water in Sai Kung which is filled with great coffeeshops, all eclectic and rather wonderful. Named whimsically, like Deer and Arm and Pan da, and Tales and Kachimushi and Winstons. They all have independent decor and cute tchotchkes everywhere and nice waitstaff and really good coffee.
    This is new. Cafes didn't used to be this way even a decade ago. There were a few who believed in the hippy way of life and cared deeply, but now it's an easily consumable customer option. With all the usual options, from matcha to yuzu.

16. The secret is better supply chains as I understand it. China consumption surged. Higher quality green coffee is a large chunk of imports, across the world. Local roasting got popular, exceptional coffee roasting facilities are ubiquitous. And coffee machine production doubled over the decade. Plus the machines are better, so the floor of barista skill to make a good latte is much lower. Another major capitalist win, even at the cost of absorbing the fringe cultures that brought it about and made it an Aesthetic.

17. The cafes have better options too, actual savoury food you can have alongside your coffee. This is one of western civilisations biggest blind spots, that the only things you see in a cafe to buy are croissants and cakes. Asia does this right!

18. The same proliferation due to globalisation also applies to bars. Even more so, actually, since the extra incentive to make things good is often less market forced and more intrinsic.

19. Lan Kwai Fong is fun, but normal. Also hilly, which is probably a plus considering street drinking.

20. Hong Kong does, and did, have a unique culture in movies, music. I grew up with it. But the visible manifestations of it being alive today are sparse. There's an indie scene I'm told, but that's true everywhere. Feels a loss.

21. Because of the combination of old, decrepit and new, the city very much has a blade runner vibe. I could see stories come alive here of a slightly grungy dystopian flavour. This also makes it a better representative of Asia than, say, Singapore. I lived in Singapore for close to a decade and it's intensely comfortable in just the way that Hong Kong isn't. There are no sharp corners there, but Hong Kong does.

22. The upscale parts are standardized just as everywhere in the world. Same restaurants with same decor, same brands selling the same kinds of things (with some Chinese brands that are less visible in the West), same glass and black steel chromed architecture, trendy shops with a bit of wood and vaguely Scandinavian/ Japanese DNA mixed in somewhere, a Starbucks logo discreetly peeking out somewhere so it's visible but not obtrusive. At a glance it's impossible to say where you are, which does have a charm but also engenders a sense of placelessness. This is sort of why Tyler's dining guide exists.

23. The feeling I got was that, seen from China, it used to be the future a couple decades ago, and now it's stood still while it rose up. I had the same feeling about Japan, but here the upkeep is not nearly as good. So it remains an odd mix, with Singapore's ultra modernist charm and greenery and upkeep alongside the unkempt remains of the 1950s and before that drastically require maintenance.

The Center Has a Bias

ARMIN RONACHER · 11 APR 2026 · SOURCE

Whenever a new technology shows up, the conversation quickly splits into camps. There are the people who reject it outright, and there are the people who seem to adopt it with religious enthusiasm. For more than a year now, no topic has been more polarising than AI coding agents.

What I keep noticing is that a lot of the criticism directed at these tools is perfectly legitimate, but it often comes from people without a meaningful amount of direct experience with them. They are not necessarily wrong. In fact, many of them cite studies, polls and all kinds of sources that themselves spent time investigating and surveying. And quite legitimately they identified real issues: the output can be bad, the security implications are scary, the economics are strange and potentially unsustainable, there is an environmental impact, the social consequences are unclear, and the hype is exhausting.

But there is something important missing from that criticism when it comes from a position of non-use: it is too abstract.

There is a difference between saying “this looks flawed in principle” and saying “I used this enough to understand where it breaks, where it helps, and how it changes my work.” The second type of criticism is expensive. It costs time, frustration, and a genuine willingness to engage.

The enthusiast camp consists of true believers. These are the people who have adopted the technology despite its shortcomings, sometimes even because they enjoy wrestling with them. They have already decided that the tool is worth fitting into their lives, so they naturally end up forgiving a lot. They might not even recognize the flaws because for them the benefits or excitement have already won.

But what does the center look like? I consider myself to be part of the center: cautiously excited, but also not without criticism. By my observation though that center is not neutral in the way people imagine it to be. Its bias is not towards endorsement so much as towards engagement, because the middle ground between rejecting a technology outright and embracing it fully is usually occupied by people willing to explore it seriously enough to judge it.

Bias on Both Sides

The compositions of the groups of people in the discussions about new technology are oddly shaped because one side has paid the cost of direct experience and the other has not, or not to the same degree. That alone creates an asymmetry.

Take coding agents as an example. If you do not use them, or at least not for productive work, you can still criticize them on many grounds. You can say they generate sloppy code, that they lower your skills, etc. But if you have not actually spent serious time with them, then your view of their practical reality is going to be inherited from somewhere else. You will know them through screenshots, anecdotes, the most annoying users on Twitter, conference talks, company slogans, and whatever filtered back from the people who *did* use them. That is not nothing, but it is not the same as contact.

The problem is not that such criticism is worthless. The problem is that people often mistake non-use for neutrality. It is not. A serious opinion on a new language, framework, device, or way of working usually has some minimum buy-in. You have to cross a threshold of use before your criticism becomes grounded in the thing itself rather than in its reputation.

That threshold is inconvenient. It asks you to spend time on something that may not pay off, and to risk finding yourself at least partially won over. It is a lot to ask of people. But because that threshold exists, the measured middle is rarely populated by people who are perfectly indifferent to change. It is populated by people who were willing to move toward it enough in order to evaluate it properly.

Simultaneously, it's important to remember that usage does not automatically create wisdom. The enthusiastic adopter might have their own distortions. They may enjoy the novelty, feel a need to justify the time they invested, or overgeneralize from the niche where the technology works wonderfully. They may simply like progress and want to be associated with it.

This is particularly visible with AI. There are clearly people who have decided that the future is here, all objections are temporary, and every workflow must now be rebuilt around agents. What makes AI weirder is that it's such a massive shift in capabilities that has triggered a tremendous injection of money, and a meaningful number of adopters have bet their future on that technology.

So if one pole is uninformed abstraction and the other is overcommitted enthusiasm, then surely the center must sit right in the middle between them?

Engagement Is Not Endorsement

The center, I would argue, naturally needs to lean towards engagement. The reason is simple: a genuinely measured opinion on a new technology requires real engagement with it.

You do not get an informed view by trying something for 15 minutes, getting annoyed once, and returning to your previous tools. You also do not get it by admiring demos, listening to podcasts or discussing on social media. You have to use it enough to get past both the first disappointment and the honeymoon phase. Seemingly with AI tools, true understanding is not a matter of hours but weeks of investment.

That means the people in the center are selected from a particular group: people who were willing to give the thing a fair chance without yet assuming it deserved a permanent place in their lives.

That willingness is already a bias towards curiosity and experimentation which makes the center look more like adopters in behavior, because exploration requires use, but it does not make the center identical to enthusiasts in judgment.

This matters because from the perspective of the outright rejecter, all of these people can look the same. If someone spent serious time with coding agents, found them useful in some areas, harmful in others, and came away with a nuanced view, they may still be thrown into the same bucket as the person who thinks agents can do no wrong.

But those are not the same position at all. It's important to recognize that engagement with those tools does not automatically imply endorsement or at the very least not blanket endorsement.

The Center Looks Suspicious

This is why discussions about new technology, and AI in particular feel so polarized. The actual center is hard to see because it does not appear visually centered. From the outside, serious exploration can look a lot like adoption.

If you map opinions onto a line, you might imagine the middle as the point equally distant from rejection and enthusiasm. But in practice that is not how it works. The middle is shifted toward the side of the people who have actually interacted with the technology enough to say something concrete about it. That does not mean the middle has accepted the adopter's conclusion. It means the middle has adopted some of the adopter's behavior, because investigation requires contact.

That creates a strange effect because the people with the most grounded criticism are often also adopters. I would argue some of the best criticism of coding agents right now comes from people who use them extensively. Take Mario: he created a coding agent, yet is also one of the most vocal voices of criticism in the space. These folks can tell you in detail how they fail and they can tell you where they waste time, where they regress code quality, where they need carefully designed tooling, where they only work well in some ecosystems, and where the whole thing falls apart.

But because those people kept using the tools long enough to learn those lessons, they can appear compromised to outsiders. And worse: if they continue to use them, contribute thoughts and criticism back, they are increasingly thrown in with the same people who are devoid of any criticism.

Failure Is Possible

This line of thinking could be seen as an inherent “pro-innovation bias”. That would be wrong, as plenty of technology deserves resistance. Many people are right to resist, and sometimes the people who never gave a technology a chance saw problems earlier than everyone else. Crypto is a good reminder: plenty of projects looked every bit as exciting as coding agents do now, and still collapsed when the economics no longer worked.

What matters here is a narrower point. The center is not biased towards novelty so much as towards contact with the thing that creates potential change. The middle ground is not between use and non-use, but between refusal and commitment and the people in the center will often look more like adopters than skeptics, not because they have already made up their minds, but because getting an informed view requires exploration.

If you want to criticize a new thing well, you first have to get close enough to dislike it for the right reasons. And for some technologies, you also have to hang around long enough to understand what, exactly, deserves criticism.