Care and Feeding of DARPA Projects

Irwin Reyes

April 7 2026

Hi I’m Irwin

twosix TECHNOLOGIES

Associate Research Director (ARD)

Leading technical efforts

Maintaining relationships with DARPA and other orgs

Setting research direction and securing funding

Career development for staff and recruiting/hiring as needed

2

I work at a medium-sized R&D firm called Two Six Technologies just outside of Washington DC. Government agencies like DARPA pay us to identify and solve high-risk high-reward research problems for government people.

I'm an Associate Research Director at Two Six. I'm directly responsible for a collection of DARPA-funded efforts, each one spanning multiple companies ("performers" in our parlance).

In everyday practice, my responsibilities include....

We'll focus on this first one, but you'll find that these are all intertwined

I didn't start my career doing all this from the beginning. It's taken many years with different experiences along the way.

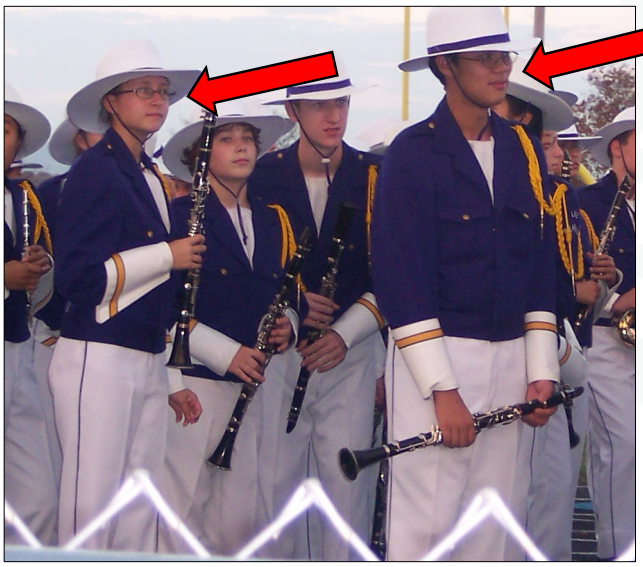
I've been at Two Six for nearly 7 years now. I spent the first few years as an individual contributor to mobile security and privacy projects, then grew into a tech lead and eventual management role.

Before Two Six, I was an academic in California researching privacy violations in mobile apps.

Before that, I was a research engineer at Dell responsible for prototyping mobile authentication methods for two years.

Before that, I was a software engineer at Johns Hopkins Applied Physics Lab working on ballistic missile defense for three years. (definitely a big pivot coming from that!)

And before that, I was a university student studying computer engineering. I diligently studied computers of all shapes and sizes.

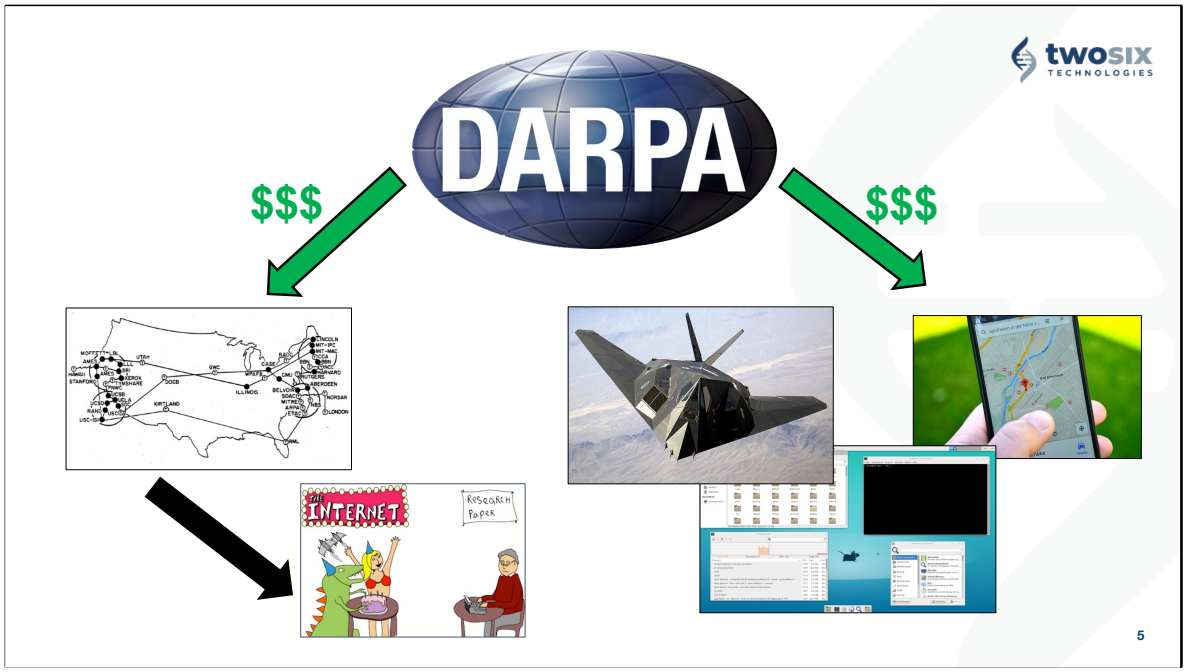


4

And before that, Allie and I were in high school computer science class together.

But I don’t have a picture of that, so here we are in marching band.

This invited talk here has been in the works for a while.

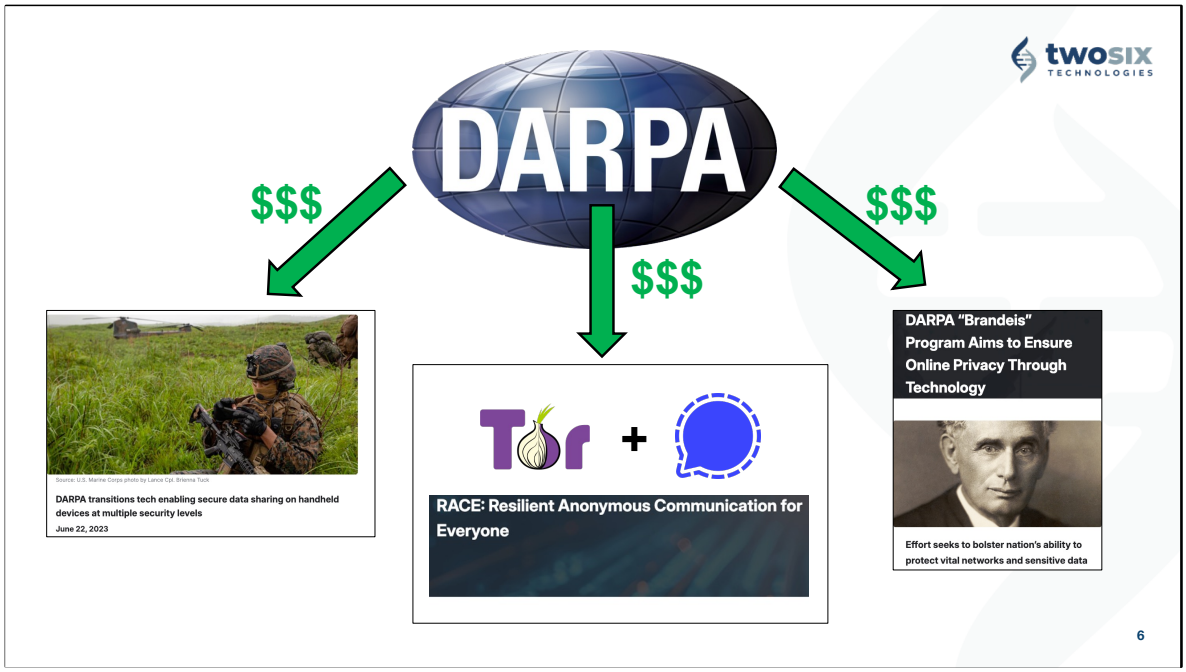


Back to the present day, I mentioned that I'm responsible for a number of DARPA-funded efforts at my company.

For background, DARPA is the Defense Advanced Research Projects Agency. They fund "far out" research for primarily for defense and government purposes, but sometimes spill over into the commercial space.

You've used at least one of their projects. The ARPANET program kicked off in the late 1960s to connect computers at government, industry, and university laboratories. This laid the foundation for the modern Internet.

Some other "greatest hits" include GPS, graphical user interfaces, and stealth aircraft.



I can't claim that my projects are anywhere near as transformative as those greatest hits.

By design, DARPA projects are high-risk high-reward. Most of them run into fundamental limits or languish when transitioning from experimental prototype to fieldable capability. You never hear about the duds.

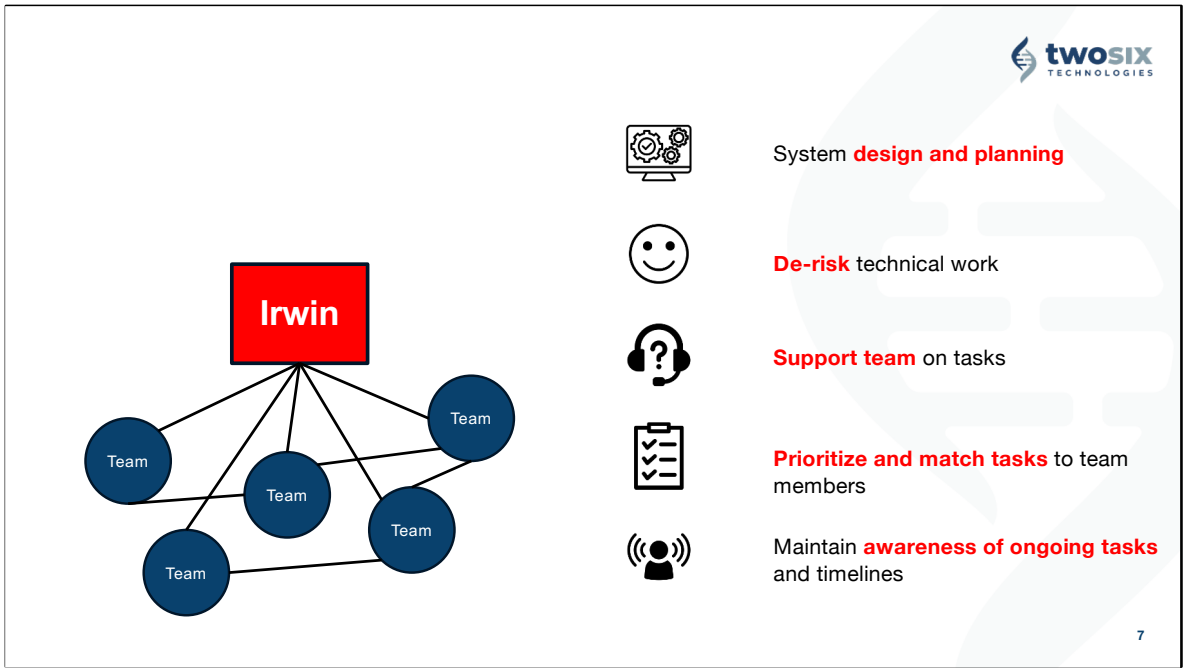
That said, I'm fortunate enough to have participated in efforts that have found useful niches.

There's the SHARE program, about secure and reliable communications in large ad-hoc wireless networks with different kinds of devices popping in and out.

There's RACE, that's basically if you mashed together Tor's network-level anonymity with Signal's end-to-end confidentiality.

And finally, the Brandeis project focused on challenges in online privacy where every company, website, and app wants to know everything about you.

Without getting into specifics, all my current projects are offshoots of Brandeis.

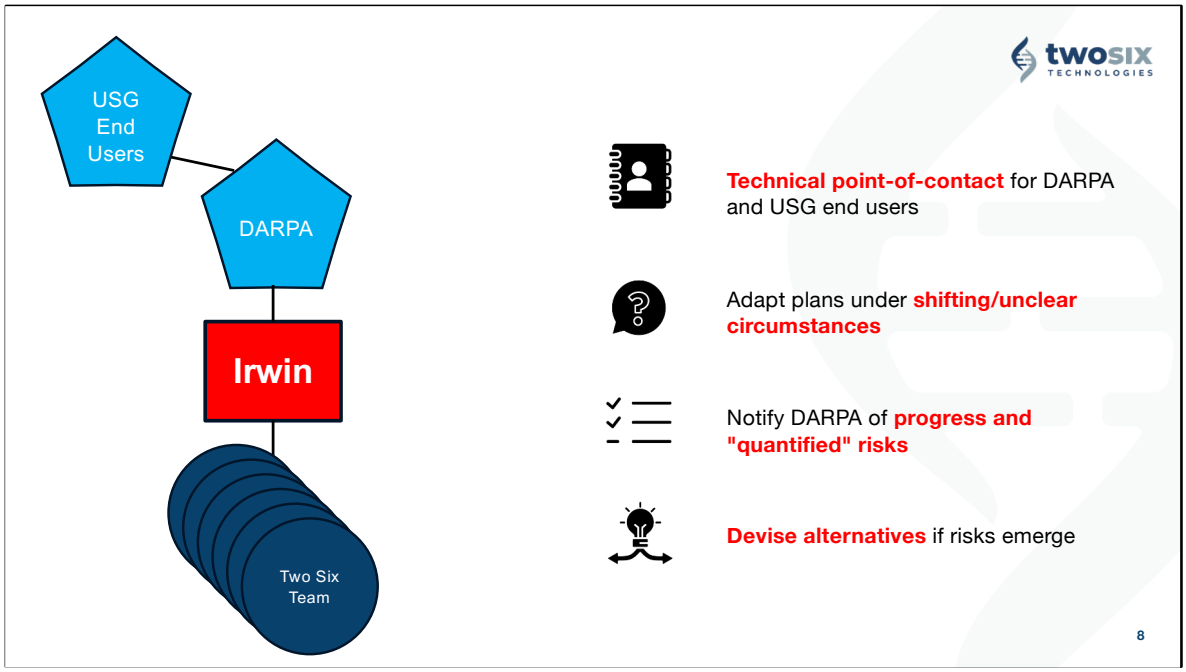


Leading projects requires me to juggle many different roles and relationships.

Day-to-day, managing my team takes up the bulk of my time.

This includes a lot of what you may have encountered in your class projects: systems design (e.g., APIs), initial learning experimentation with novel tech, identifying tasks and assigning them in manageable chunks.

But it also includes a lot of non-technical "soft skills:" keeping people motivated, finding growth opportunities for folks, and making sure people have the support they need---whether from one another, from me, or from someone outside the team.

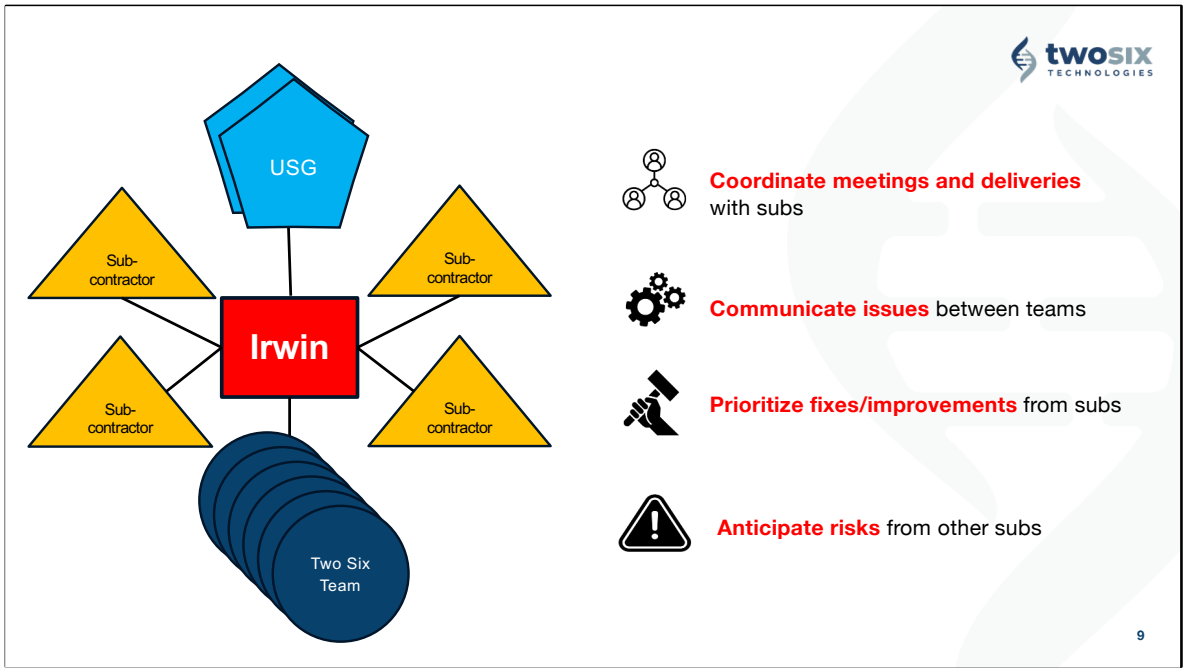


I may be a “team lead” but that doesn’t mean I’m on top of the totem pole. If anything, I’m right in the middle.

I’m the intermediary between my team, and our customer (DARPA and other US government people).

A huge part of my job is to make sure my team has clear priorities that line up with customer needs. Unclear or constantly shifting priorities are very bad for team morale, and it introduces program risk.

Sometimes customers have unclear needs, or requirements with unknown trade-offs, or requests that need more time/effort than anticipated. It’s my responsibility to drill down into those details to avoid or mitigate risk.



I don't just manage "up" and "down" the totem pole. I also need to make sure our subcontractors are doing what the overall program needs.

Our DARPA contract is structured so that the subcontractors are experts on a major isolated component to the program, and my team's job is to make everything play nice together.

For instance, one company is responsible for novel hardware. Two are responsible for data analysis and machine learning models.. Another works on user interface.

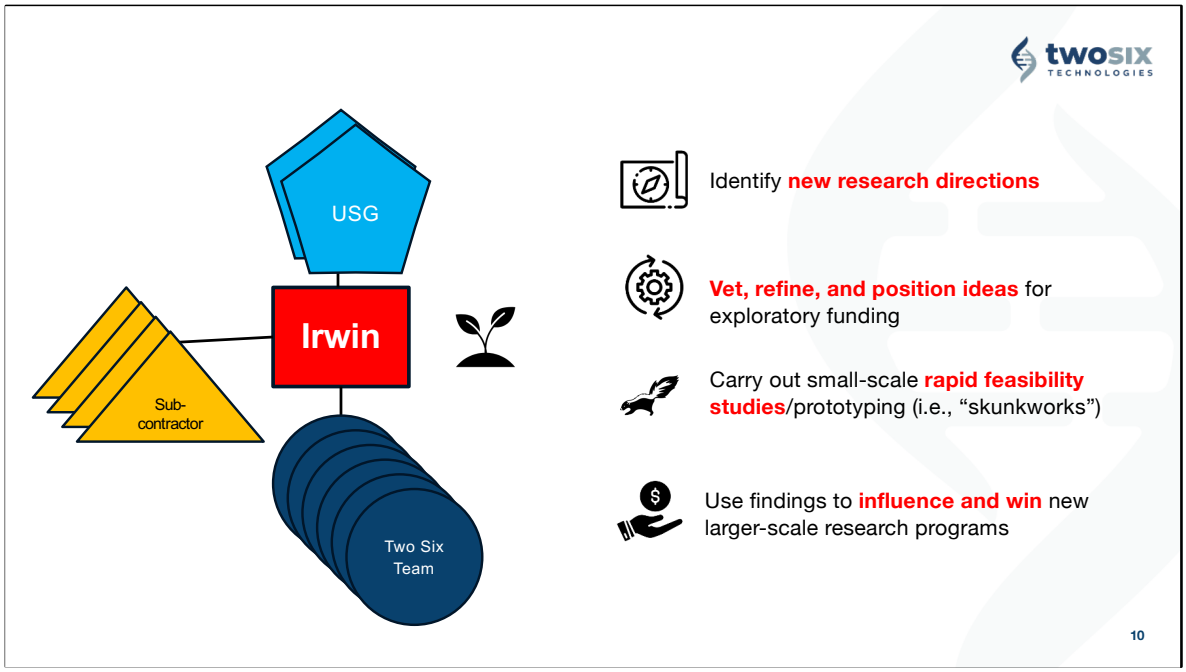
My team is responsible for bringing it all together in an actual platform, in this case an actual phone running actual interoperable software

We also took care of testing everything for usability and validating correct function.

We keep everyone else honest and on the same page. This involves a lot of coordinating communications. If I sense that there might be a

miscommunication or mismatched assumption somewhere, it's my responsibility to bring people together and clear it up.

In many cases, I need to have fine-grained visibility into which individuals on which teams are the actual experts on a given subcomponent.



Over the course of a program, someone on my team (or myself) might have a high-risk high-reward idea that’s just on the edge of our contract’s scope

This could be an overlooked use case for our technology, or an alternative approach that has asymmetric upside, or a different way of looking at the problem that totally turns it on its head

When this happens, my job is to vet the idea, refine it, and if it passes muster, use my credibility to pitch it to DARPA for exploratory funds

(it’s really useful to already have strong relationships and an established reputation from delivering core program capabilities)

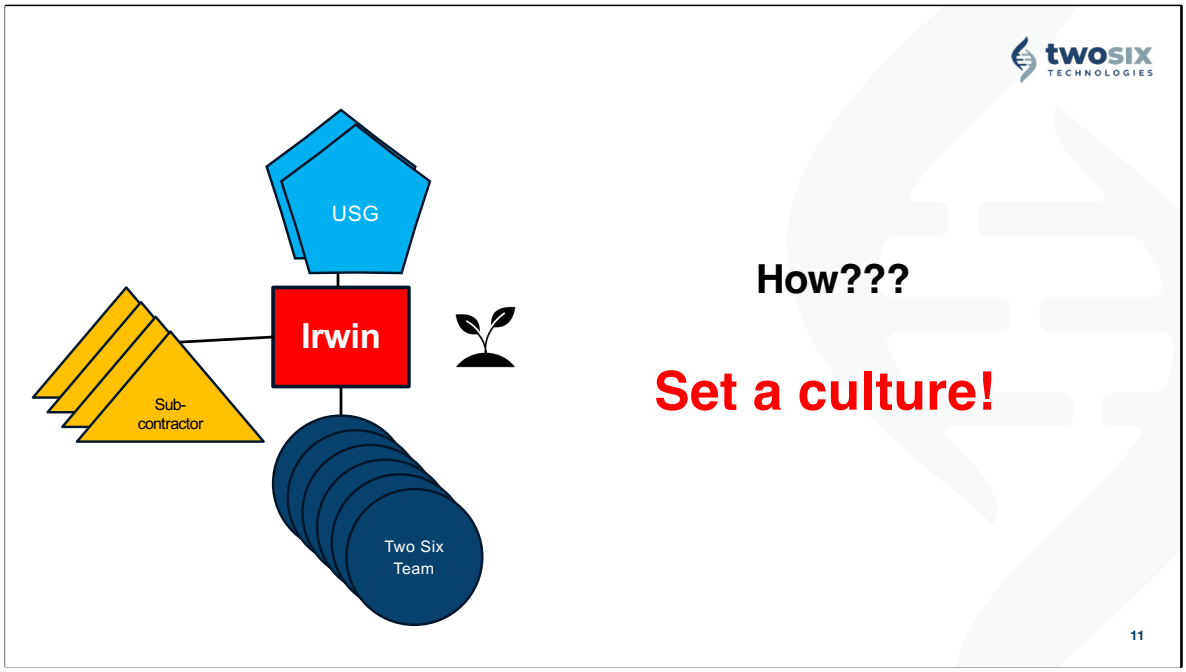
We call these efforts “seedlings,” but you can think of them as small-scale short-term “skunkworks.”

I staff these side efforts with people already on my team. We spend some of our time over the course of ~3 months answering the question: “Is this idea just crazy, or is it crazy enough to work?”

We write up our findings and present them to DARPA. These results end up shaping how DARPA decides to allocate funds in the future.

If our idea doesn't work despite our best efforts, that's actually really valuable information: we detail what fundamental barriers we encountered, and that saves the government time and money from future large-scale efforts that will crash into that wall.

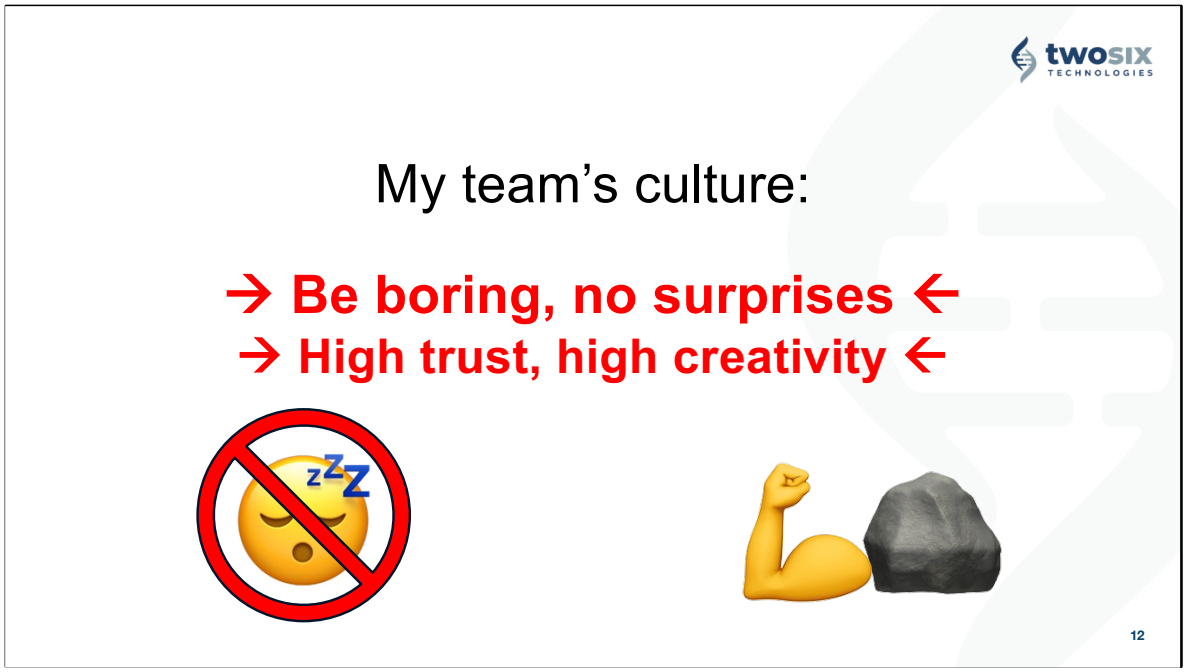
If our idea does work, that gives us the inside track on future work. DARPA knows that Irwin's team at Two Six cracked the door open just a tiny bit towards a wild new capability. Guess who they'll expect to submit a bid for the next round of research programs, which happen to line up quite well with our exploratory work?



If this sounds like a lot of things for me to manage, it really is!

So how can a manager and a team deliver without going crazy?

I handle it by boiling it down to one overarching task that guides me in all that: my job is to set a culture.



The culture I try to set is quite simple.

Be boring, no surprises

Now, ”boring” doesn’t mean lack of interest, or low-effort, or strict adherence to initial ideas.

“Boring” in my team means rock solid reliability, so that we can do our job sustainably long-term with minimal self-inflicted stumbles.

It means that my team and I strive to deliver no surprises. The things we say we’re going to do, we do. The risks we identify are real, but also reasonably well-understood with a variety of practical options for addressing them.

This sets the conditions where high trust and high creativity are likely to emerge

How do I set this culture? By creating an environment where people talk to one another so there are no surprises.

I take a “soft touch” approach for this. Instead of scheduling tons of meetings, I strive for presence and awareness.

Although Two Six doesn’t have an official in-office policy (many folks are hybrid), I’m in the office for four or five days a week.

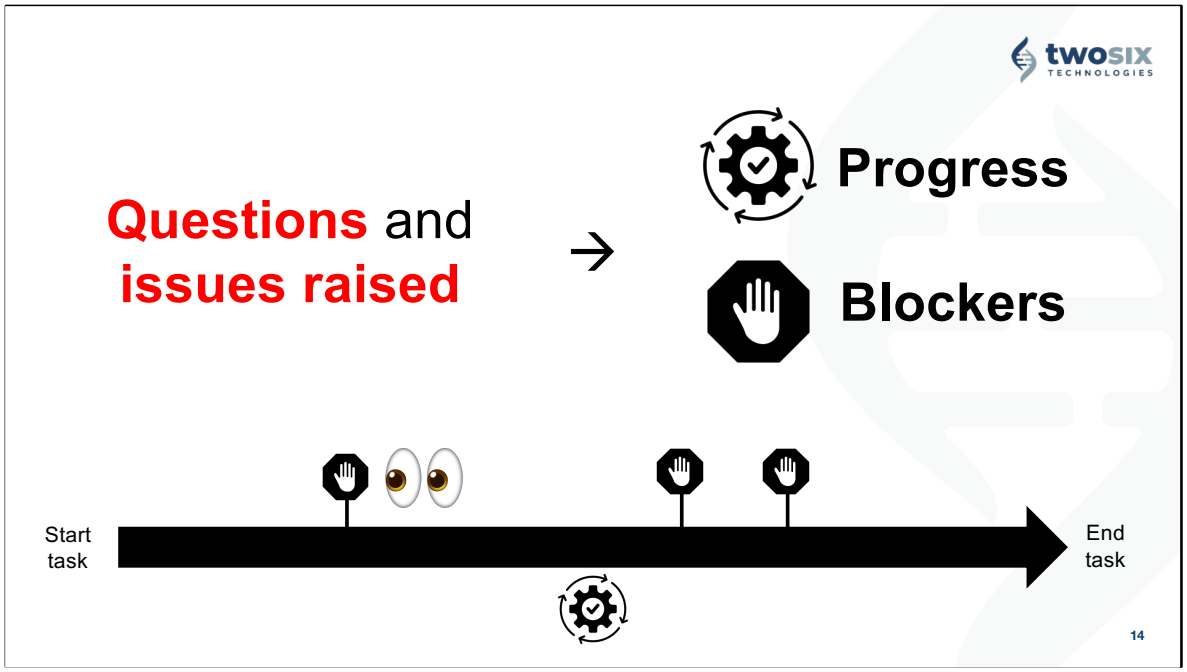
I keep an open door policy and encourage my team to stop by to talk about whatever whenever they’re around.

It doesn’t need to be just about work: if you overheard us, you’ll pick up on gym talk, trashy reality TV, cheesecakes, and puzzle games.

But those conversations are valuable because people will naturally bring up their work with a team that they feel a part of: what’s frustrating them, what they’re proud of, what they want to learn more about.

When they communicate clearly and openly and support one another, I'm able to put them in positions to support other relationships too: as experts for our DARPA/USG customers, or as technical leaders to talk to our partner companies.

I can pitch my wacky ideas to my team, get constructive feedback, and demonstrate that they should feel free to do the same



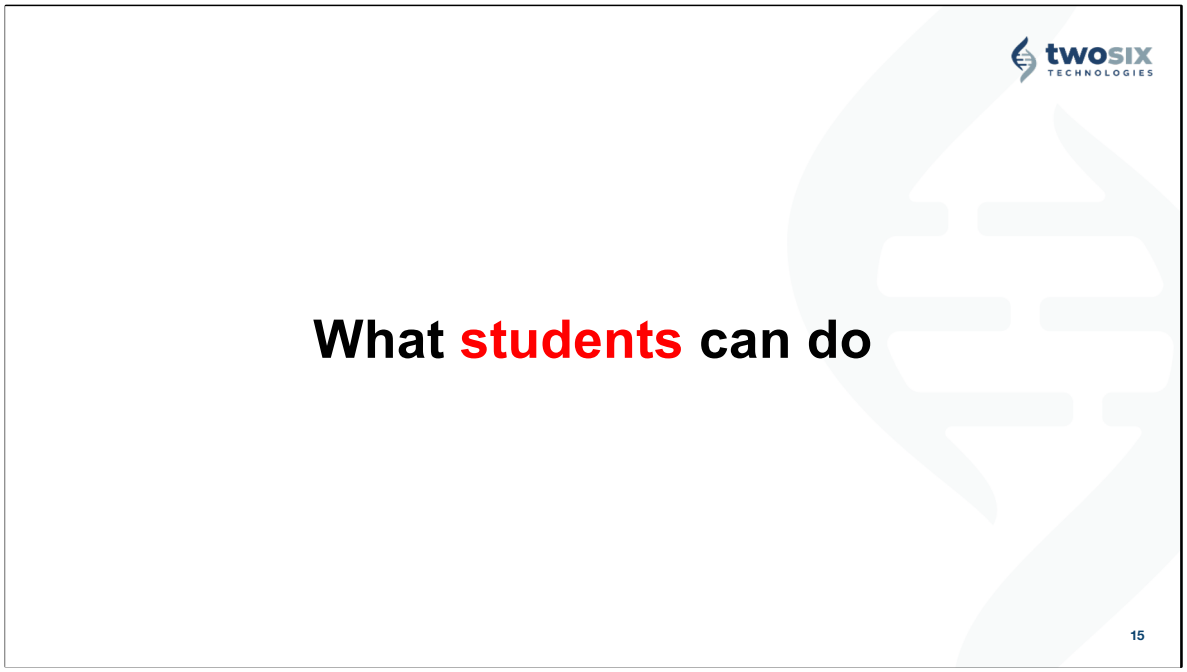
The ideas and questions people raise about their work tell me a lot because those indicate progress and blockers.

When I divvy up tasks for the team, I keep a rough mental timer for how long it should take them to finish it.

That depends on the complexity and open-endedness of the task, and the teammate's strengths and interests.

I anticipate what challenges they might encounter and when. And check in on them if they don't ask questions when I expect them to.

This is how I maintain awareness of where everyone is in their work, all without being a mean guy---that wouldn't be fun for me, and it's not great for morale.

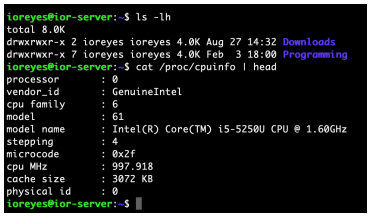


If my job sounds like a big firehose of things to do and keep track of, don't worry: you don't have to manage up, down, and side-to-side, or set a culture for a team for a long time.

I got to where I am gradually, over the course of 15 years of experience.

Whether you want to run a team someday or not, there are a couple things you can do now to set yourself up well to establish yourself in a technical team.

#1 Learn the tools of the trade



Command-line Linux

Basics: File navigation, text editing

Bonus: ssh remote access, git version control

Super bonus: Self-hosting / "homelab"

Git principles

Basics: `clone`, `pull`, `commit`, `push` operations

Bonus: branches, merge requests, code review

16

First, familiarize yourself with the tools of the trade.

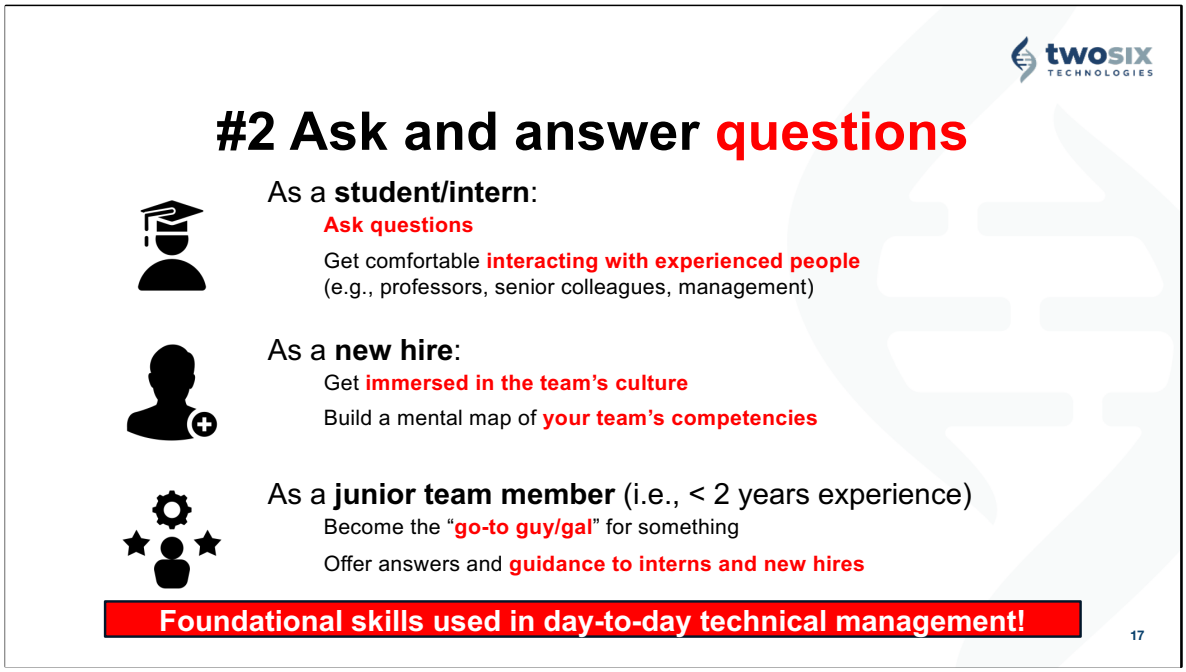
In all the jobs I've occupied in industry and academia, a working knowledge of command-line Linux means you can "talk the talk" of any team you join. That makes learning your team's culture much easier.

If you can navigate files and folders and use a text editor, that puts you ahead of most new graduates coming out of school. Bonus points if you can use git version control and ssh remote access in the command line too.

Speaking of git, knowing how to use it to collaborate with others is huge. Simple code clone, pull, commit, and push operations are all you need day-to-day. Especially impressive if you know how to use branches, submit merge requests, and review code with your team---these help teammates stay aware of what everyone else is doing.

And I'll toss in one bit of super extra credit that I suggest to our interns as a challenge: take a cheapo computer like a Raspberry Pi and have it serve up a static webpage to any client in the world, securely and reliably.

Learning how to self-host is immensely powerful and you demystify how the internet actually works in the process. It's a skill that's practically expected of all senior research and engineering staff in my company.



Next, don't be afraid to ask and answer questions.

If you're a student or intern, your #1 job is to learn. You should be asking all sorts of questions about everything from your team's technologies and processes, to where to get a good cheap lunch near the office.

As a new hire, you should continue learning and connecting with your team. Really pick up on the culture and make it work for you and your growth.

Before you know it, a new batch of interns is starting, and they'll be asking about how merge requests work or where to get a good sandwich. And you'll be in a position to continue and contribute to your team's culture.

twosix TECHNOLOGIES

#3 Hone your systems thinking

"I don't care what anything was designed to do,
I care about what it can do."

Dig into how tech and people operate, and follow the implications

18

Finally, develop your systems thinking skills

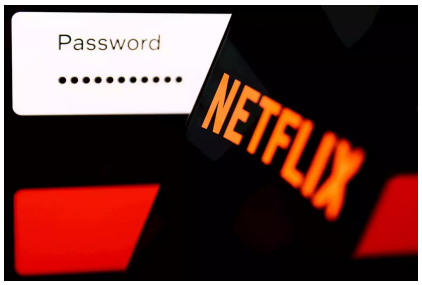
This ties together the last two things I mentioned: think about technologies holistically. Don't just consider isolated pieces of code and metal, but more broadly how technology and people interact with one another.

There's this really great line from the film Apollo 13 that captures what I mean

In other words…

Really understand the system you're dealing with: its scope, its assumptions, its effects on people and people's effects on it, its edge cases and especially undocumented (or emergent) behavior.

#3 Hone your systems thinking



What's the problem at hand? Why does it matter?

What are some reasonable solutions?

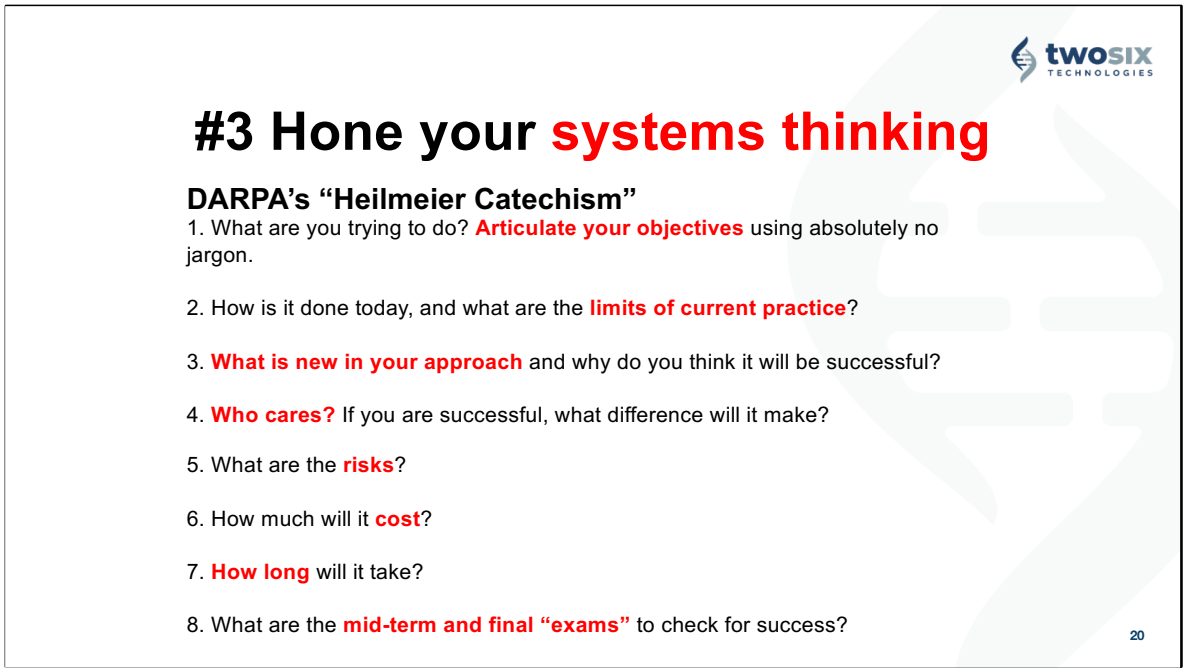
How would you test your idea? What would it take resource-wise and time-wise?

How can the solution fall short? What trade-offs does reality impose?

19

I'll give you a more down-to-earth example that I use as an open-ended interview question. Some time ago, Netflix implemented barriers to discourage people from sharing passwords and accounts.

To gauge systems thinking, I'd ask:
- What underly problem were they trying to solve?
- Why does that problem matter?
- How do you think they did it?
- How would you test your idea?
- What opportunities and limitations would they encounter in deployment?
- How might you undermine your assumed approach?
- How might you hedge against risks?
- What implicit assumptions are you making, and how would you pivot if those assumptions weren't valid?

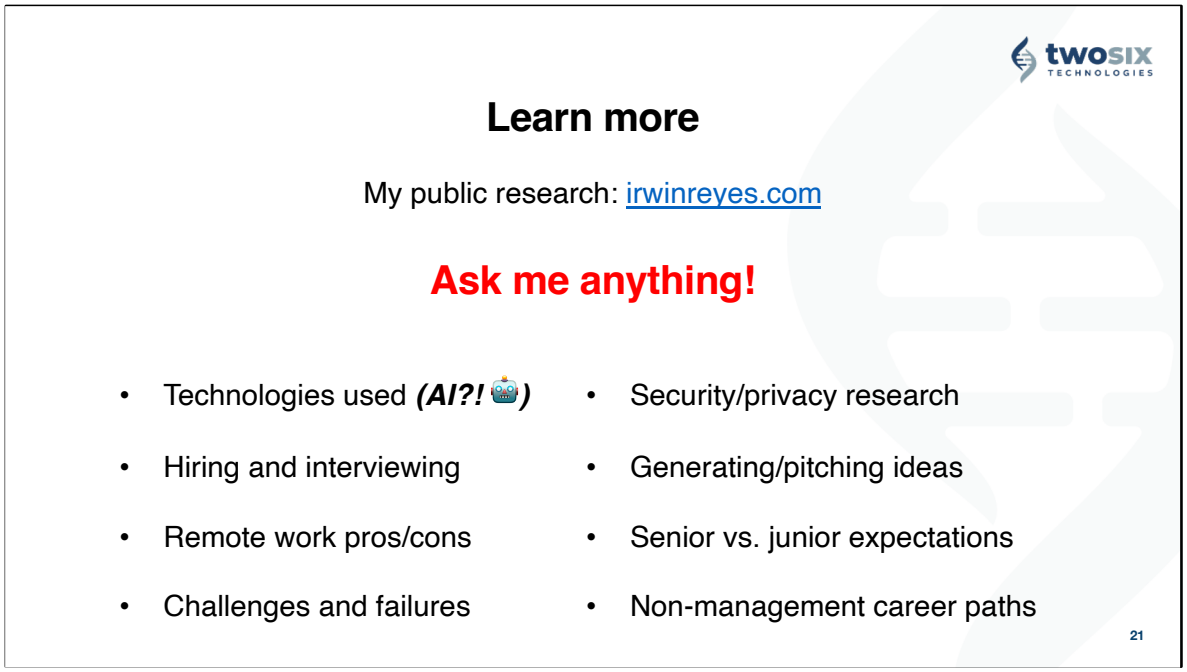


If you practice this early, you actually start thinking in exactly the way DARPA decides what research is crazy enough to work and well thought-out enough to fund.

You can be a junior individual contributor, or a tech lead, or a director overseeing existing projects and new proposals, and you can apply this way of thinking in all sorts of contexts.

You'll find that new ideas and angles can just pop out of nowhere if you turn over some technical artifact or human assumption that everyone just takes as a given.

https://www.darpa.mil/about/heilmeier-catechism



And with that, I'll wrap up this talk and take questions.

I tried to avoid getting mired up in nitpicky specifics, so feel free to ask me for more details about anything I mentioned.

If you'd like to learn more about some of the stuff I've been involved with over the years, I've posted those online.