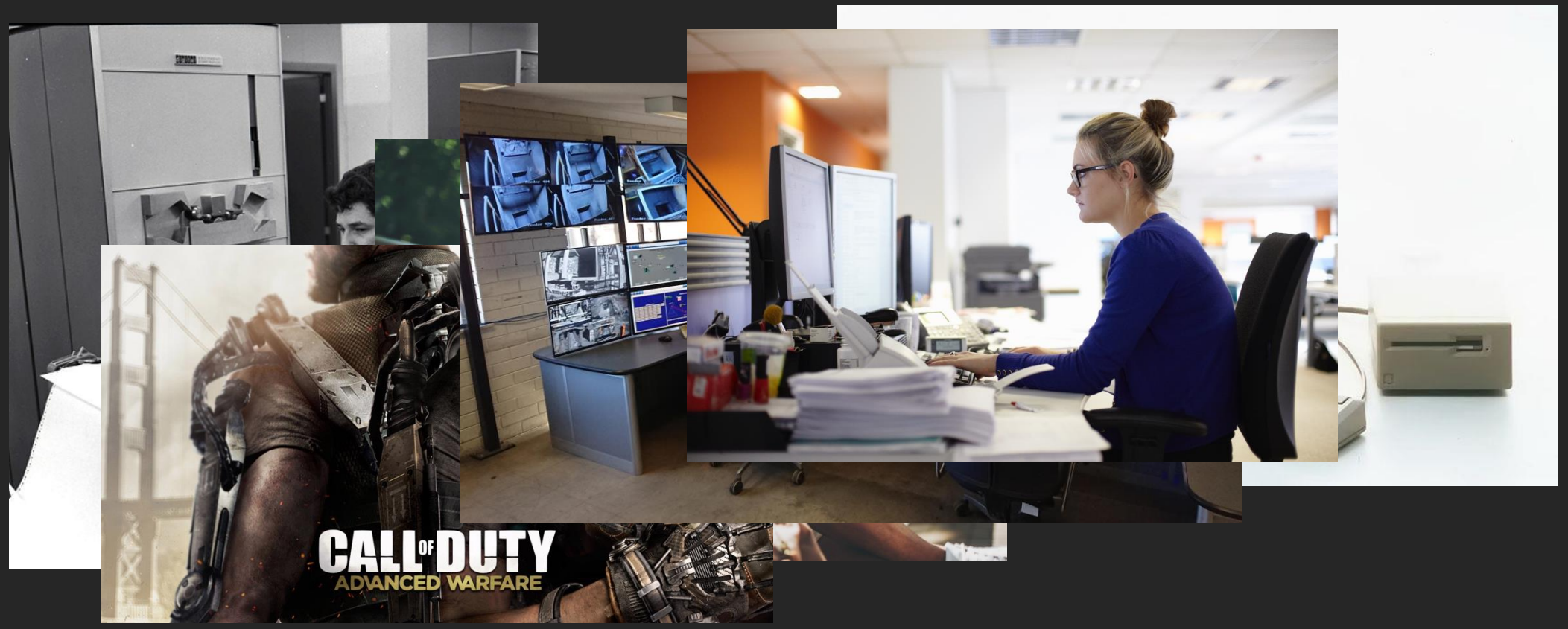
Our Responsibility As Engineers

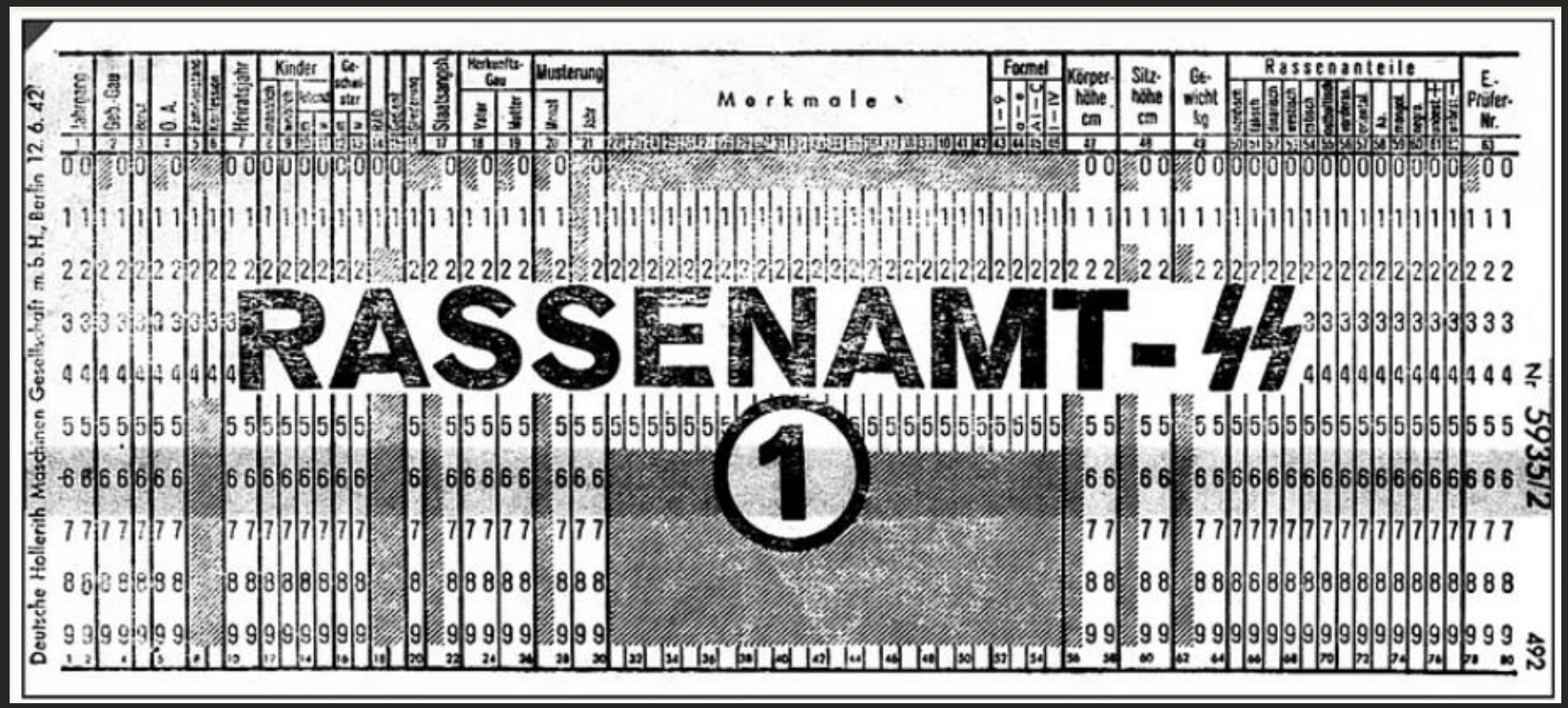
YONATAN ZUNGER

Content Note:

We're about to talk (not graphically) about some serious issues – including injuries, deaths, stalking, and domestic violence.

Our history is one of rapid innovation.

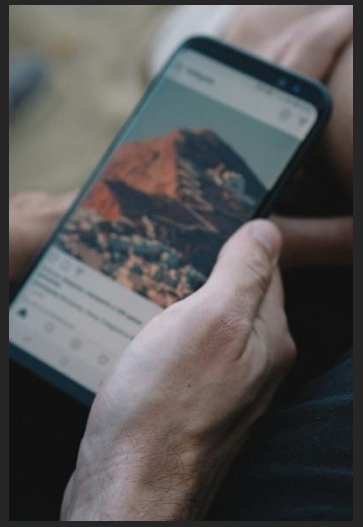



RASSENAMT-SS
1
Deutsche Hollerith Maschinen Gesellschaft m.b.H., Berlin 12. 6. 42
Nr 59352
492
Jahrgang
Geb.-Gau
Kreis
O. A.
Familienname
Vorname
Heiratsjahr
Kinder
Geschwister
Staatsangeh.
Herkunfts-Gau
Vater
Mutter
Musterung
Monat
Jahr
Merkmale
Formel
Körperhöhe cm
Sitzhöhe cm
Gewicht kg
Rassenanteile
E.-Prüfer-Nr.

We tried to split up our profession

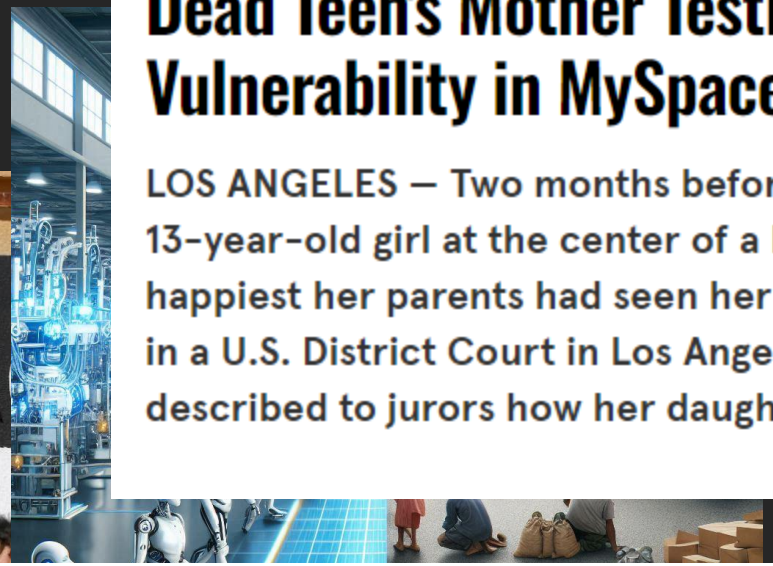
A FEW “SERIOUS” BITS

THE “EXCITING” STUFF



And overall, we succeeded...

Sorta.



KIM ZETTER SECURITY NOV 20, 2008 6:09 AM

Dead Teen's Mother Testifies about Daughter's Vulnerability in MySpace Suicide Case -- Update

LOS ANGELES — Two months before she committed suicide in 2006, a 13-year-old girl at the center of a landmark cyberbullying case was the happiest her parents had seen her in a long time. Tina Meier, testifying in a U.S. District Court in Los Angeles on Wednesday afternoon, described to jurors how her daughter Megan [...]

LOCAL CRIME & PUBLIC SAFETY

D.C. 911 center under fire again after baby dies during computer outage

It is unclear whether delays in delivering advanced medical care contributed to the five-month-old's death.

What we do matters.

And with great power comes great responsibility.

All of engineering has two sides:

HOW THE SYSTEM SHOULD WORK

HOW THE SYSTEM SHOULD FAIL

Safety must become as fundamental to our work as breathing.

The basic principle of safety engineering

- Know the ways your system might fail as intimately as the ways your system should work.

  Brainstorm failure scenarios, and keep that list as fresh as your success scenarios.

  *What you don't know **can** hurt you – so use many eyes and plan for surprises.*

- For each scenario, have a plan.

  Eliminate it

  Reduce its severity or frequency

  Give users a way to solve it themselves

  Have a response plan for when things go wrong

"If something happens to one in a million people once a year, here at Google we call that 'six times a day.'" -- Andreas Schou

How do you do that brainstorming?

- **System-first:** What are the components? What happens if each one fails? What if it gets bad input? *And the components include the users!*
- **Actor-first:** What might someone want to achieve using this software? How might they do it? Under what circumstances are they using it?
- **Target-first:** Who might be affected by someone using this software? What might make them more or less vulnerable? How would they be able to respond?

Diversity matters! You can't do this step if you don't understand the lived experience of people very different from you. Have a broad team.

Example time.

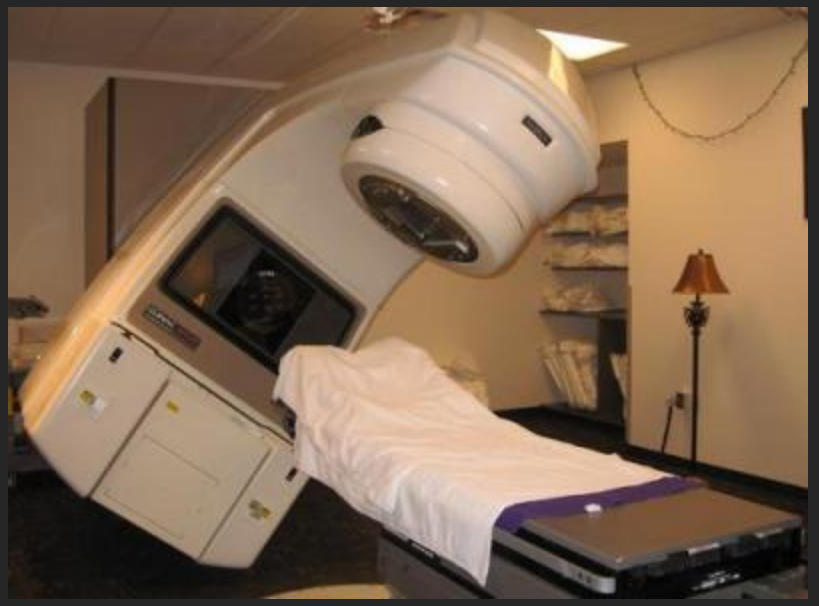
THERAC-25

- Computer-controlled radiation therapy, launched 1982
- Three modes:

  Setup mode: Light pattern to position the patient

  Electron-beam therapy: Low-current electron beam

  X-ray therapy: Photon beam created by firing a *high*-current electron beam at a Tungsten target.
- THERAC-6 and -20 used an electromechanical interlock to select modes; THERAC-25 used software.



What could possibly go wrong?

- Race condition: If you selected X-ray mode then flipped to electron mode within 8 seconds, it left the beam in high-current mode.
- Miscellaneous bug: Sometimes the electron beam fired during setup mode.
- 3 dead, 3 severely injured.

What were the mistakes?

- Investigation: Not a single flaw, but a pattern of poor design.
- No safety check for “never happens” configs to replace the old interlock.
- Little to no feedback in the system to compare actual state to user expectation.
- Errors just said “MALFUNCTION 54” and didn’t distinguish safety-critical issues. Operators just hit “proceed” because that was normally the right thing to do.

Key lessons

- In a complex system, define safety invariants and check for them
- Safety needs to include the operator flow – “operator error” is a euphemism for “UX failure.”
- … and UX isn’t just software, it’s the whole process. We build socio-technical systems.

Google+ and Gender

- Social network, launched in 2011
- Privacy focus: Introduced “circles” of friends.
- At launch, only three fields in the profile had to be public: name, picture, and gender.

  Why? You need gender to form sentences in the UX!

  “X shared a picture with you” requires X’s and your gender in many languages

That couldn’t be a problem, could it?

- Instantly: People were scraping the service to find women, share links to their pics.
- “NBUMM” users – “nice boobs u marry me?” – and many more.
- In social networks, the initial population sets the culture, and the culture shapes the future population. We just seeded our network, with $XXXM investment, with a culture that drove women away.
- This materially affected the success of the product during critical early months when public opinion was forming.

Key lessons

- Without diverse opinions in the room and being listened to, you will miss threat scenarios.

  Yes, everyone in the room who made that decision was a guy.

  One of our PM's raised a red flag a few weeks before launch and we didn't prioritize fixing it.

  We were idiots.

VioGén

- Domestic violence kills thousands of people each year. Police resources are limited.
- The idea: Use data to optimize use of scarce resources.
- The method: 35-question y/n survey administered by police when responding to a DV incident, which feeds into a model that gives a 5-level risk classification used to shape police response.
- Safety mechanism: Police, judges, etc. are encouraged to use their better judgment and override it when needed.

If you’re saying “oh, shit” right now,
then you’ve been paying attention.

What actually happened?

- 95%+ acceptance rate of system predictions.
- Of the 247 dead so far, a deep-dive into 98 revealed that 56% had been rated “negligible” or “low” risk.
- When the system said negligible or low risk, not only was police assistance no longer available, but this was now evidence in court *against* issuing restraining orders &c.
- It might be working overall at reducing deaths! Too early to tell. But if it rates you at the low end of the scale, you’re in trouble.

What did they miss?

- Garbage in, garbage out: The world's best model won't help you if it gets bad inputs.
- Incentives don't align: Police are using this to *reduce* load, but now they have to do extra justification work whenever it says to ignore something.
- No resiliency to bias, insider threat, etc.
- Lots of assumptions: Male abuser of female victim, cop is disinterested but trusted neutral helper, etc. etc.
- Down to the base assumption: Is routing police more efficiently even the problem that needs to be solved?

Key lessons

- Lots of good researchers, not enough operational experience.
- This whole “what could go wrong” modeling should have been the first thing they did.
- The whole system would probably have been designed differently if they had.
- Lots of effort into testing the core component (the model) but most failures happen at the system integration level.
- You can make a system that works great in the common case, but the failures are a killer.

OK, but how do *we* do this?

Staff engineers aren’t usually the final decision-makers.

From the bottom up

People mirror the attitude and behavior of people they respect.

You are the leaders your teams interact with most.

If you create a culture of safety, the team will follow.

Raise safety questions in design reviews, code reviews, launch reviews.

Get people to see this as something they expect to be asked about.

When they start asking it themselves, you know it’s working.

From the top down

Look for senior allies and escalate.

Get them involved by showing both the stakes and the methods.

What if they don’t care?

Sometimes they just aren’t used to it.

Sometimes there’s a particular issue that hits home for *them*.

Sometimes you need to tie it to product success.

Sometimes they need to know their asses are personally on the line.

Mean but effective trick

"We're writing the playbook for [system].

If [event] happens, we'll need leadership to make some tough calls about [legal or life issue].

Are you the decision-maker for this?"

What if they don’t care?

Sometimes they just aren’t used to it.

Sometimes there’s a particular issue that hits home for *them*.

Sometimes you need to tie it to product success.

Sometimes they need to know their asses are personally on the line.

Sometimes they literally don’t care.

So let’s sum up.

What you build matters.

It doesn’t have to be a medical system to have serious impact.
Stalking, bullying, job loss… software can go wrong.

Safety includes everything.

There ain’t no such thing as “out of scope.”

Just because someone used a system in a way we didn’t expect doesn’t mean it’s not our problem.

If it’s your system and someone or something can get hurt, you need a plan.

You need to embody safety.

Model threats from the moment you imagine the system.
Know them as well as you know your features.

That's you, specifically.

As Staff+ engineers, you are the leaders who define your teams' culture.

Do it right, and your teams will follow.
Do it wrong, and your teams will follow.

Now go do good work.