Hi, I’m Andrew Godwin

- Django core developer
- Senior Software Engineer at Eventbrite
- Private + Instrument pilot
Content Warning

Aviation accidents
Road accidents
Discussion of death
Software is difficult.
"Things I won't work with"

By Derek Lowe
"...a more stable form of it, by mixing it with TNT.

Yes, this is an example of something that becomes less explosive as a one-to-one cocrystal with TNT."
"...the operator is confronted with the problem of coping with a metal-fluorine fire.

For dealing with this situation, I have always recommended a good pair of running shoes."

On “Sand Won’t Save You This Time”
Unicode
Locales
Time
Calendars
Geography
Money
Network latency
Hardware unreliability
Deadlocks
Bit flips
Ambiguous specifications
No documentation
Not unique to software

We just *move faster* and hit them at higher speed.
Who's solved this? Aviation.
A Boeing 747 has six million parts
A Boeing 747 has six million parts

...and a 0.000006% accident rate
Deaths per billion hours (Per passenger, UK 1990-2000)

- Train: 30
- Airplane: 30.8
- Car: 130
- Walking: 220
People matter as much as machines
Aviation Accident Causes (2005 Nall report)

- Pilot: 76%
- Mechanical: 16%
- Other: 9%
Let's look at some aviation principles
And how we can apply them to software.
Principle #1

Hard Failure
If something is wrong it turns itself off

Autopilots, engines, air conditioning, and more
This only works if you have **redundancy**

All of these systems have a backup that lets you land.
"We'll ignore errors so the site doesn't crash!"

"Save the invalid data and we'll fix it later"
These are great ways to ensure you never fix something.
No accident or outage has a single cause. Stop your code getting into odd states.
Fail hard if anything unexpected happens

Validate all your data strictly in and out

Deploy changes early and often
Single points of failure **can be good**

Only one place to look when things go wrong!
Principle #2

Good Alerting
Cockpits are incredibly selective about what sets off an audio alarm
Alert fatigue is real. Avoid at all costs.
Never, ever, put all errors in the same place.
Critical
Normal
Background
Critical

Wakes someone up. Actionable.

Normal

Background
Critical  Wakes someone up. Actionable.

Normal  Fixed over the next week.

Background
<table>
<thead>
<tr>
<th>Critical</th>
<th>Wakes someone up. Actionable.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>Fixed over the next week.</td>
</tr>
<tr>
<td>Background</td>
<td>Metrics, not errors.</td>
</tr>
</tbody>
</table>
Have you been ignoring an error for weeks? Then turn off its error reporting.
Principle #3
Find your limits
Everything will fail. You should know when.
What's your Minimum Equipment List?

What can you run the system without?
REQUIRED
Lavatory ashtrays
Air conditioning
Seatbelt signs

OPTIONAL
Passenger video screens
Fuel caps
Weather radar
Did you load test? Did you fuzz test?
You don't have to perfectly scale. But you do have to know where your limits are.
Risk is fine when you're informed!

Unknowns are the most dangerous thing.
Principle #4
Build for failure
No single thing in an aircraft can fail and take it down.
We all want this for our code, but the way to do it is to **build for failure**.
Kill your application randomly

Practice server network failures

Develop on unreliable connections
The majority of pilot training is handling emergencies.
CABIN FIRE IN THE AIR
1. Master switch ........................................... Off if electrical fire
2. Electrical circuits ................................ Off as required
3. Fire extinguisher ................................... Use as necessary then ventilate cabin
4. Forced landing procedure or diversion as applicable

ENGINE FIRE IN THE AIR
1. Throttle .................................................. Closed
2. Mixture .................................................. Idle Cut Off (fully lean)
3. Fuel ...................................................... Off
4. Ignition .................................................. Off
5. Cabin Heaters/Defrost ................................ Off
6. Forced landing (without power) procedure

DO NOT ATTEMPT RESTART
1. Radio ..................................................... Check frequency, volume, on/off switches, squelch,
2. Headset .................................................. operate avionics selector switches
3. Electrics ................................................. Check plugs secure, change headsets, try handheld
4. Transponder ......................................... microphone if fitted
5. Speechless/transmit blind/non-radio procedure as appropriate

RADIO FAILURE
1. Radio ..................................................... Set 7600
2. Headset .................................................. reset once only
3. Electrics .................................................

ELECTRICAL FAILURE
1. Electrical Load ....................................... Reduce (non-essential) electric/radios off
2. Circuit Breakers/Fuses ................................ Check/reset
3. Ammeter/Voltage ..................................... Check
4. Warning Light ......................................... Check

IF OUTPUT RESTORED
Restore essential services singly

IF NO OUTPUT
Reset Master switch
(only for 2 secs then on)

IN THE EVENT OF REPEATED/CONTINUED ELECTRICAL FAILURE
IF OUTPUT RESTORED
Restore essential services singly

IN THE EVENT OF HEAVY ELECTRICAL LOADS
Reduce (non-essential) electric/radios off

IF NO OUTPUT
Reset Master switch
(off for 2 secs then on)

IN THE EVENT OF REPEATED/CONTINUED ELECTRICAL FAILURE
IN THE EVENT OF HEAVY ELECTRICAL LOADS
Divert if applicable, note radio transmission

IN THE EVENT OF REPEATED/CONTINUED ELECTRICAL FAILURE
IF NO OUTPUT
Reset Master switch
(only for 2 secs then on)

IN THE EVENT OF HEAVY ELECTRICAL LOADS
Reduce (non-essential) electric/radios off

IN THE EVENT OF REPEATED/CONTINUED ELECTRICAL FAILURE
IN THE EVENT OF HEAVY ELECTRICAL LOADS
Divert if applicable, note radio transmission
Use checklists. Don't rely on memory.
If you **practice failure**, you'll be ready when the inevitable happens.
Aviation Accident Causes (2005 Nall report)

- Pilot: 76%
- Mechanical: 16%
- Other: 9%
Principle #5

Communicate well
"You have control"
"I have control"
"You have control"
Complex software means separate teams.
As you grow, communication becomes exponentially harder.
Clear communication is vital.
Write *everything* down.

Written specs = less time in meetings.
Have a clear chain of command.
Make decisions.

They don't have to be perfect, just good enough.
Principle #6

No blame culture
How do I know all these aviation stats?
Every incident is reported and investigated.
There is never a single cause of a problem.
Make it very difficult to do again.

Why did your software let this happen? What's the UX of your admin tools like?
Emergency Alert
BALLISTIC MISSILE THREAT INBOUND TO HAWAII. SEEK IMMEDIATE SHELTER. THIS IS NOT A DRILL.
This is the screen that set off the ballistic missile alert on Saturday. The operator clicked the PACOM (CDW) State Only link. The drill link is the one that was supposed to be clicked.

#Hawaii
Encourage reporting.

Don't blame anyone for a mistake. They're unlikely to make it again.
Reward maintenance as well as firefighting

It's easy to look good when you ship broken and are always heroically fixing it.
In aviation, every rule is written in blood.
Software is not yet there. But we are getting closer.
Margaret Hamilton
Her error detection code saved Apollo 11
Patriot Missile
Floating-point bug killed 28
Therac-25
Killed 3, severely injured at least 3 more
Uber Autonomous Vehicle

Saw a pedestrian and chose to hit her.
Hard failure
Good alerting
Find your limits
Build for failure
Communicate well
No blame culture
Thanks.

Andrew Godwin
@andrewgodwin  aeracode.org