

VOLANT

Leading the world to autonomous flight

Saving Time, Reducing Cost, Increasing Quality:
V&V la révolution!

Saving Time, Reducing Cost, Increasing Quality: V&V la révolution!

Anthony Williams
Volant Autonomy

Introduction

- About Volant Autonomy
 - Aerospace startup with focus on compliance and collision avoidance for sUAS (Drones)
 - Key Products
 - Via - Path Planning and Compliance
 - Traject - Detect-and-Avoid (DAA) System
 - Focus of this talk
- About Me
 - Safety Critical Software Engineer at Volant Autonomy
 - 10 Years Experience in Aviation and Rail Industry
 - Focus on systems integration and continuous integration

What's new in the V&V world?

Perhaps not our test methods, but how we apply them:

Increase Quality

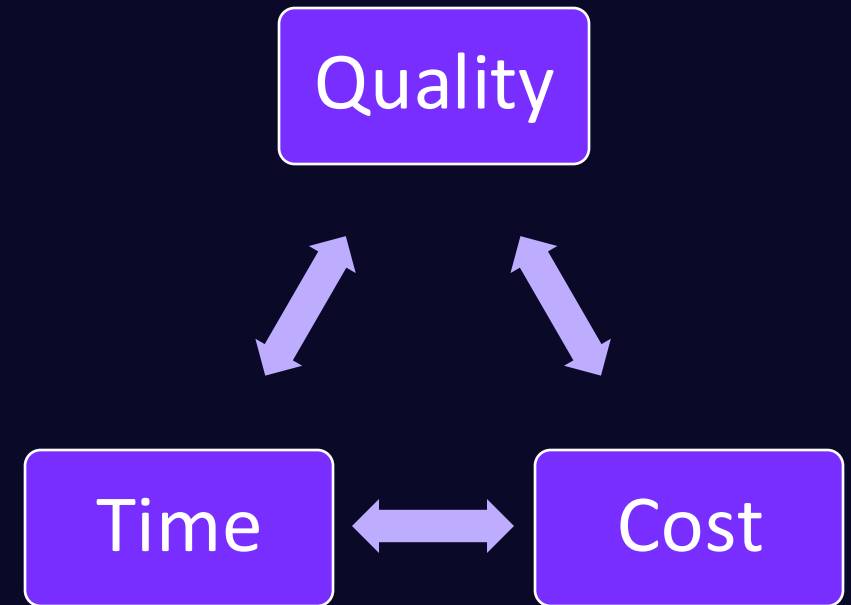
- Testing needs to tell you something
- Who CI's the CI?

Save Time

- Optimise *what* to test, and *when* to test it

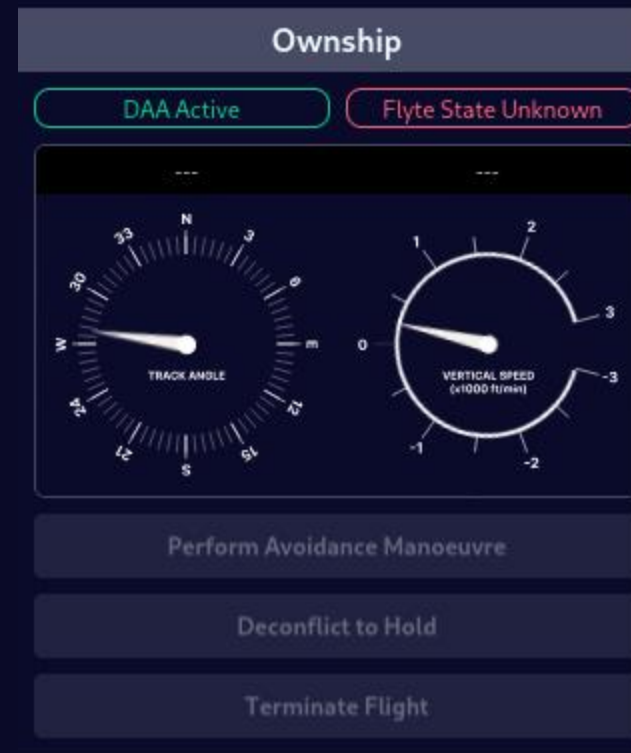
Reduce Cost

- Evaluating *what* to buy, and *where* to place it



Case Study: Volant Traject

- Detect-and-Avoid solution targeting DO-178C DAL-D
- Small project team (~7 FTE) in a smallish startup (~15 FTE)
- Early integration and confidence building essential
- Mixture of languages



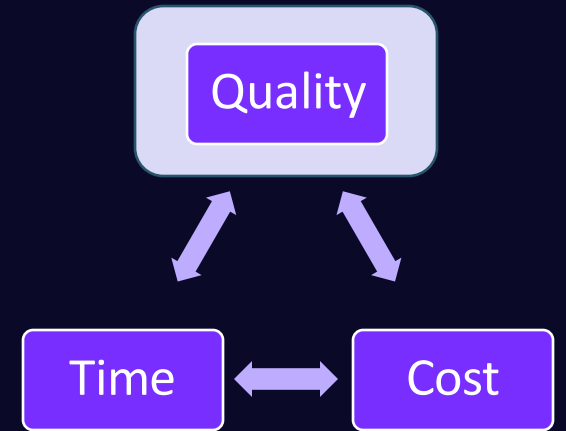
Increasing Quality

Start small and build up. . .

All the following must pass before review approval:

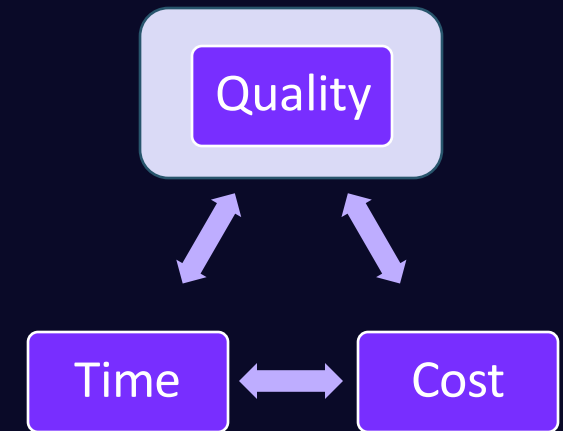
- Compile / Type Checks “does it build”
- Unit Tests (+ Coverage?) “does it work”
- Static Analysis Tools “is it robust”
- SPARK Proof “is it REALLY robust”

Ask: Is this tool doing anything useful?



Caution – Check your CI results

- CI checks usually have Pass/Fail results
- Ensure these actually work!
- Timeouts / less common error conditions can be erroneously marked as passing



```
docker-ada-all / docker-ada-ci
succeeded on Jul 4 in 12m 46s

Search logs

Run Ada Linting 10m 3s
23
24 surveillance-ingest-internalise.adb:61:52: medium: range check might fail
25 61 |           Heading           => Input.Heading,
26    |                               ~~~~^~~~~~
27 reason for check: value must fit in component type
28 possible fix: subprogram at line 50 should mention Input in a precondition
29 50 |   procedure Apply_ICD_Constraints
30    |   ^ here
31
```

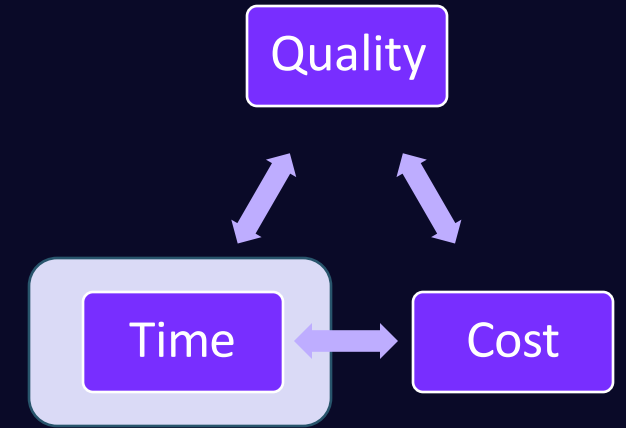
Saving Time – What and When?

Only test when it is needed

- A Python review does not need the Ada CI tests running
- But a Python library change DOES trigger checks for usage

What should I test?

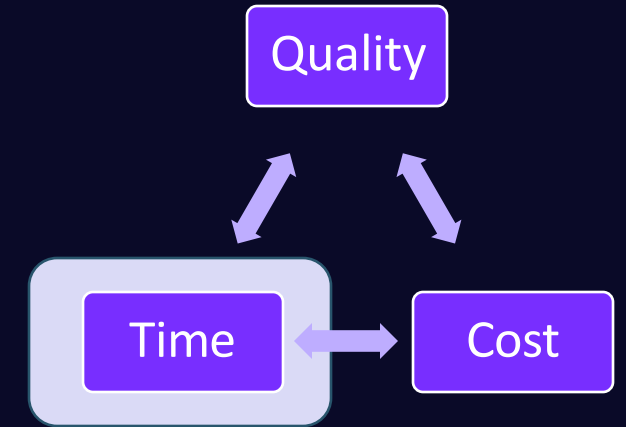
- Integration tests not under CI - complex and bespoke
- Code coverage and cross-builds are run post-merge



Saving Time – Response

Give feedback quickly

- Developers can sense-check before committing
- 5 {seconds, minutes, hours} grouping
- Abort if a test fails – return results promptly



```
docker-ada-all / docker-ada-ci
failed last week in 16m 40s

❌ Run Ada Linting 14m 15s
23
24 gnatcoll-json.ads:191:20: error: function associated to aspect Iterable with controlling result is not allowed in SPARK
25   191 |           Element      => Array_Element);
26       |           ^~~~~~
27   delayed type aspect on "JSON_Array" is required to be in SPARK
28 gnatprove: error during flow analysis and proof
29 Summary logged in /app/obj/development/gnatprove/gnatprove.out
30 Error: Process completed with exit code 1.
```

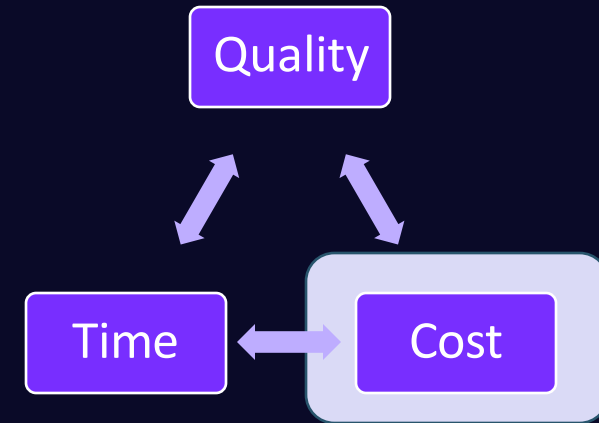
Reducing Cost – Cloud

Understand Cloud Pricing - Yes, it can be daunting!

- Set sensible timeouts and limits / alerts
- Monitor costs closely, especially initially or after change
- If you pay for GitHub (or similar) you may get some 'free' CI credits included

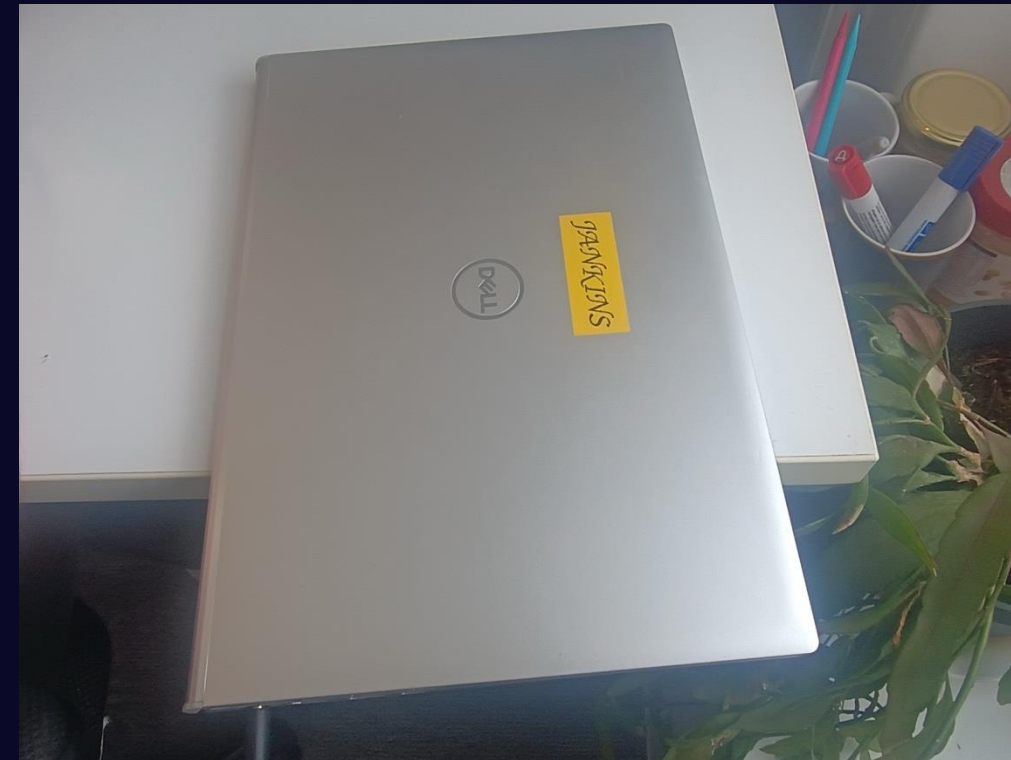
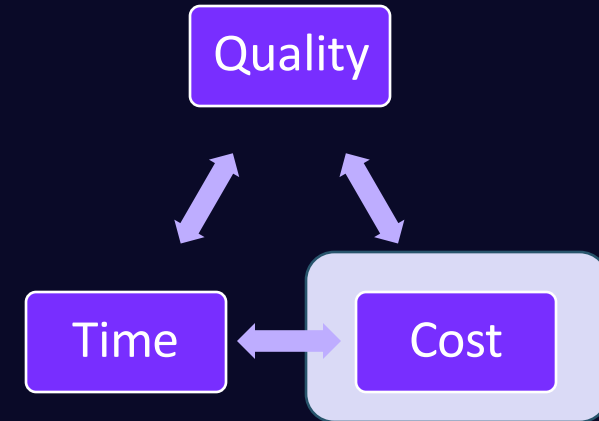
Select Resources carefully

- Free or low-cost tiers suitable for basic CI tasks
- Traject's Python CI "free", Ada CI is ~£25/month



Reducing Cost – On Premises

- An On-Premises CI server needn't be an expensive server!
- Ours is a spare laptop
- How "free" on-prem actually is depends on your company...
- Easy to add SUT, hardware interfaces
 - We have Morello hardware for on-target testing



Case Study : GNATfuzz – A novel tool

- Fuzz tester - produces huge volumes (millions of cases) of randomised input data, logs crashes (exceptions) and the input parameters which caused them
- Configurable with "stopping criteria" to constrain testing
- Fully automated with "fuzz everything" feature – so can be built into a CI task
- See João's talk later this afternoon!

GNATfuzz – Is it useful?

Quality – Several issues found in Traject code, including:

- Soundness issue in SPARK Assume statements
- Unit tests not updated following base type change

```
function Pressure_To_Feet
(Pressure : Common.Types.Pressure_hPa_T)
return Common.Types.Feet_T
is
use Ada.Numerics.Long_Elementary_Functions;

Pres_Division : constant Long_Float := Long_Float (Pressure / 1013.25);

Exponent_Res : constant Long_Float := Pres_Division**0.190284;
pragma Assume (Exponent_Res in 0.0 .. 1.0E52);

Subtraction_Res : constant Long_Float := 1.0 - Exponent_Res;
Result          : constant Long_Float := 145366.45 * Subtraction_Res;
begin
return Common.Types.Feet_T (Result);
end Pressure_To_Feet;
```

```
"Decoded_In_Parameters": [
{
"Parameter_Name": "Pressure",
"Parameter_Type": "Common.Types.Pressure_hPa_T",
"Parameter_Value": " 5.48612406879369E+303"
}
],
"Subprogram_Under_Test": "Common.Utilities.Pressure_To_Feet",
"Testcase_Exception": {
"Exception_Information": "raised ADA.ASSERTIONS.ASSERTION_ERROR",
"Exception_Message": "Assume failed at common-utilities.adb:90",
"Exception_Name": "ADA.ASSERTIONS.ASSERTION_ERROR"
}
}
```

GNATfuzz – How much?

Time – Fixed due to nature of fuzz testing

- Takes 2 hours at the moment – expected to increase as Ada LoC grows
- CPU intensive

Cost

- Cloud run cost for 2 hours, plus many-core CPU - £5/run (£100/month)
- We want to save results to monitor trends and analyse failures

Where to deploy GNATfuzz?

- Time and Quality are linked - if we shorten the fuzz time, we lose the value of fuzz testing
- **Conclusion** – Run using on-premises Jenkins server

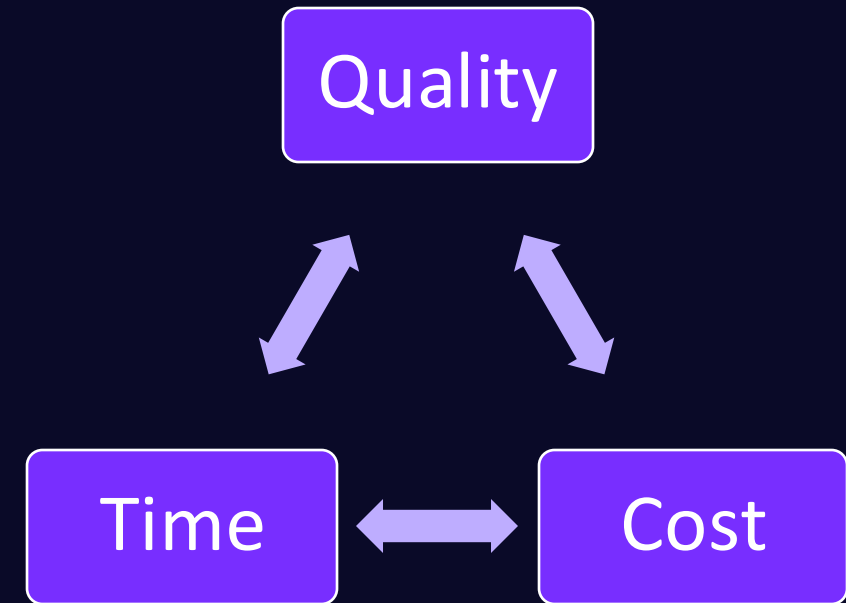
Traject CI - Summary


All artefacts built using Docker Images – can be run "anywhere"

Test Artefact	Location	Trigger
Compile	GitHub Action (Cloud)	Auto (PR)
Unit Test	GitHub Action (Cloud)	Auto (PR)
Unit Test + Coverage + MC/DC	Jenkins (On-Premises)	Periodic + Change
Static Analysis + SPARK Proof	GitHub Action (Cloud)	Auto (PR)
Unit Test + Coverage (Morello)	Jenkins (On-Premises)	Periodic + Change
Fuzz Test	Jenkins (On-Premises)	Periodic + Change
Build and Release	GitHub Action + Pulumi (Cloud)	Manual

Conclusions – Let Them CI-at Cake?

- A constraint problem? $\text{Max}(\text{Quality}) + \text{Min}(\text{Cost}) + \text{Min}(\text{Time})$
- For each tool:
 - Quality – What benefit is this tool bringing?
 - Time – Are there any optimisations?
 - Cost – What are the best resources to select?





VOLANT

Leading the world to autonomous flight

Questions?