Quality Assurance via Successive HAB Flights

CAPITOL TECHNOLOGY UNIVERSITY
1927

Alex Antunes, Ryan Schrenk, Christel Gesterling, Nathan Weideman
Capitol Technology University

The plusses and minuses of how we often substitute formal SE with rapid testing via high altitude balloons

or

Building like an engineer instead of a technician

SE/Rapid Dev Style Conflicts

CAPITOL TECHNOLOGY UNIVERSITY
1927

Process-Based SE

1) Document and capture
2) Regular reviews
3) Use of best practices

The goal of SE is to reduce risk and increase success, not to add a burden that provides no benefit to the project.

How to get initial buy-in from students?

Technician-Led Rapid Dev

1) Build first, document later
2) Review only as issues arise
3) Whatever works plus gut instincts keeps us moving

Hazard of dealing with talented students: If it worked for me before, why change things?

Humans dislike 'bad' or mandated process but like 'good' or personally proven process.

Ignoring SE is learning?

CAPITOL TECHNOLOGY UNIVERSITY
1927

Exposure to SE in courses is abstract and unapplied at first.

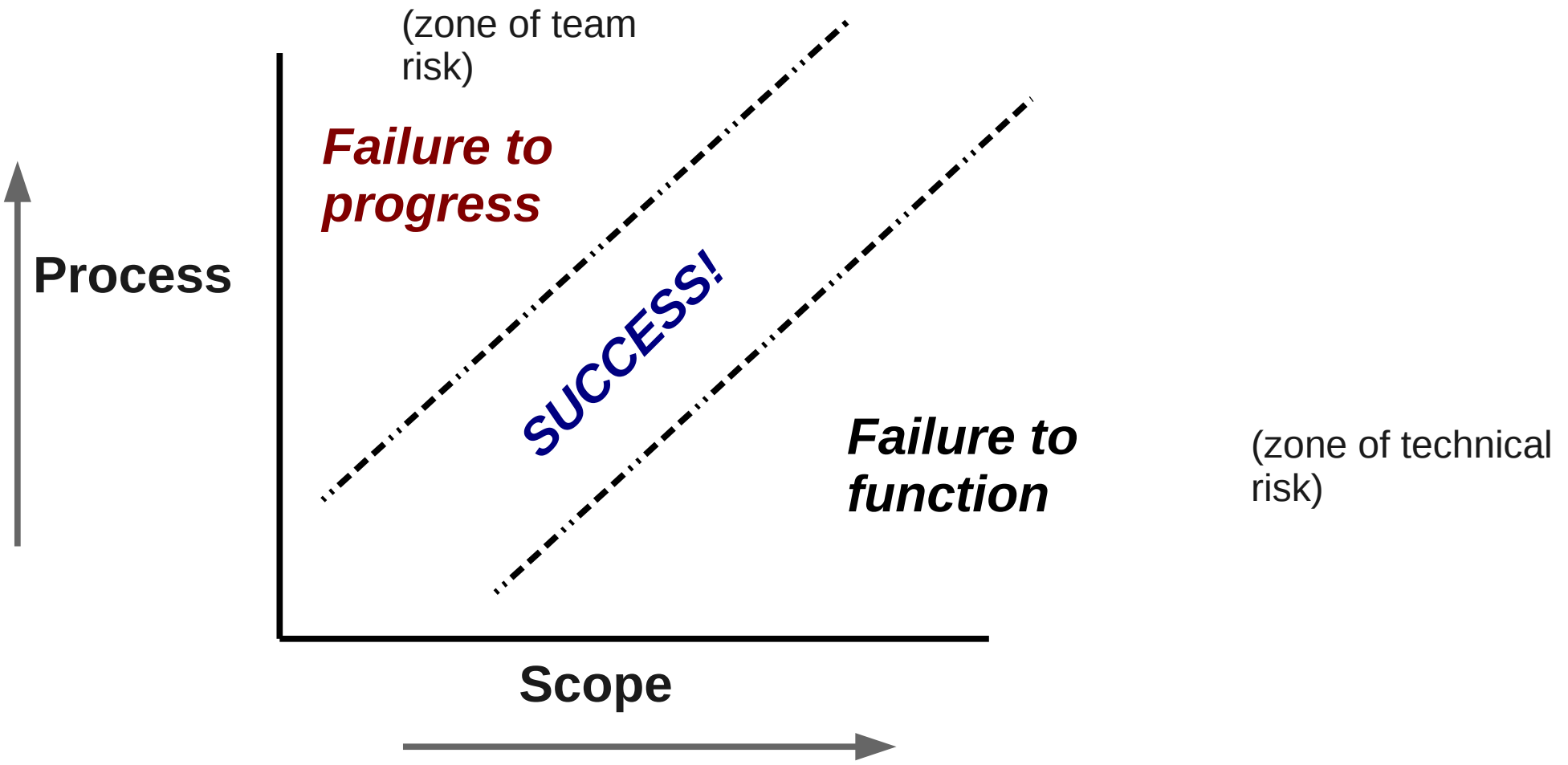
Perception that time spent with 'process' wastes time better spent building

But, when failure occurs, students appreciate and implement that process for future work!

Does this extend-- does 'we failed to follow [X] and now appreciate it' extend to 'as well as process [Y] and [Z]'?

Our Elusive 'Goldilocks' for SE

CAPITOL TECHNOLOGY UNIVERSITY
1927



excessive SE requirements = team risk = overburdening, disengagement, morale issues
lack of adequate SE = technical risk = failure to integrate and/or deploy mission

The Conflict?

CAPITOL TECHNOLOGY UNIVERSITY
1927

Quotes from students:

'Most students tend to go with what they know.'

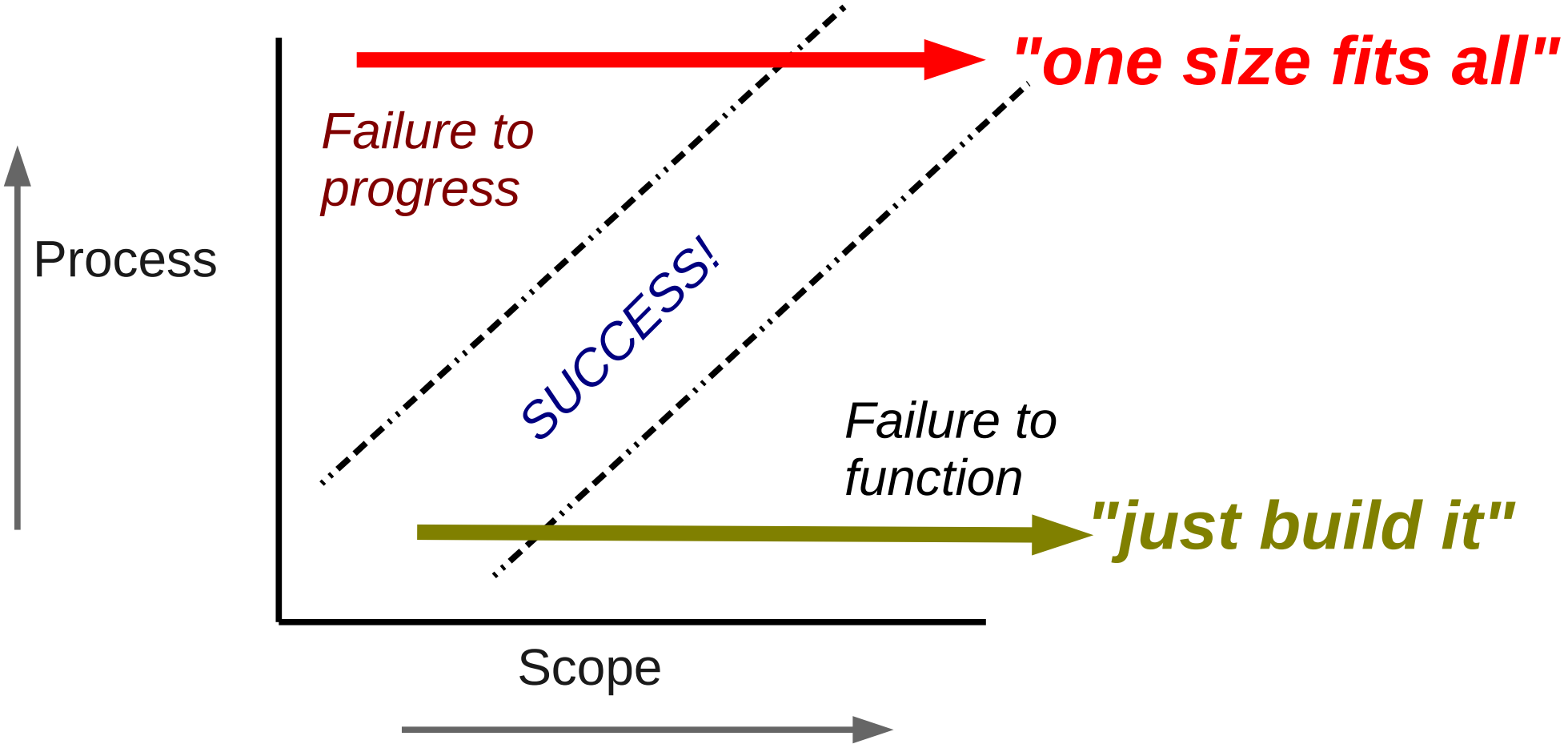
'Students don't even know they aren't following the correct process-- or any process.'

'"Systems engineering" raison d'etre is to generate documentation.'

'Students will typically want to put their own spin on things, without a well-defined scope or constraints, that can lead to trouble.'

e.g. Bureaucratic versus Technician

CAPITOL TECHNOLOGY UNIVERSITY
1927



Why teach SE/QE/QA?

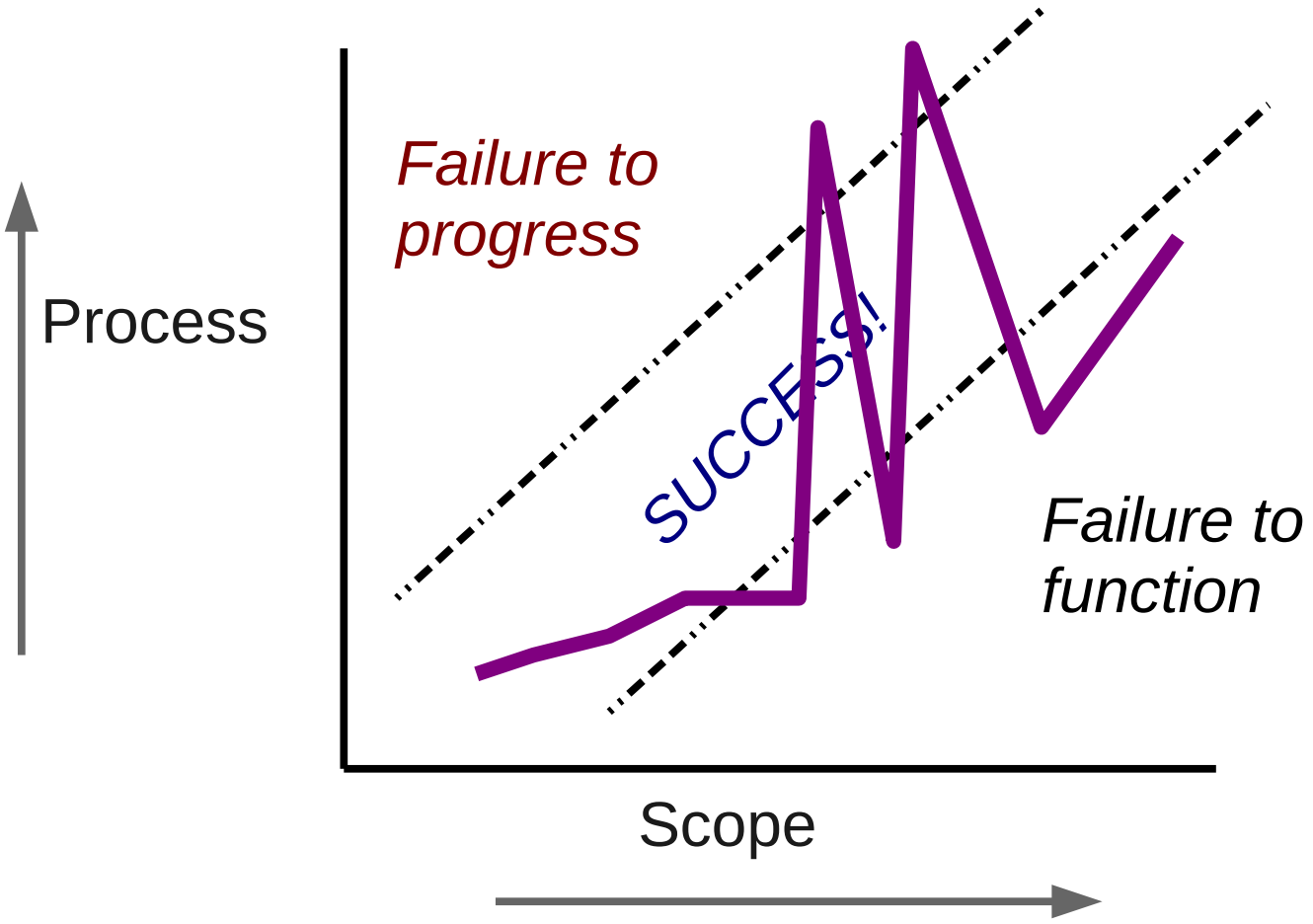
CAPITOL TECHNOLOGY UNIVERSITY
1927

'When following a well-defined systems engineering approach, fewer surprises arise, deadlines are met, and the customer is satisfied.'

'As a young or new engineer, one must take this for granted-- that these processes and constraints are in place for a reason.'

What students often experience

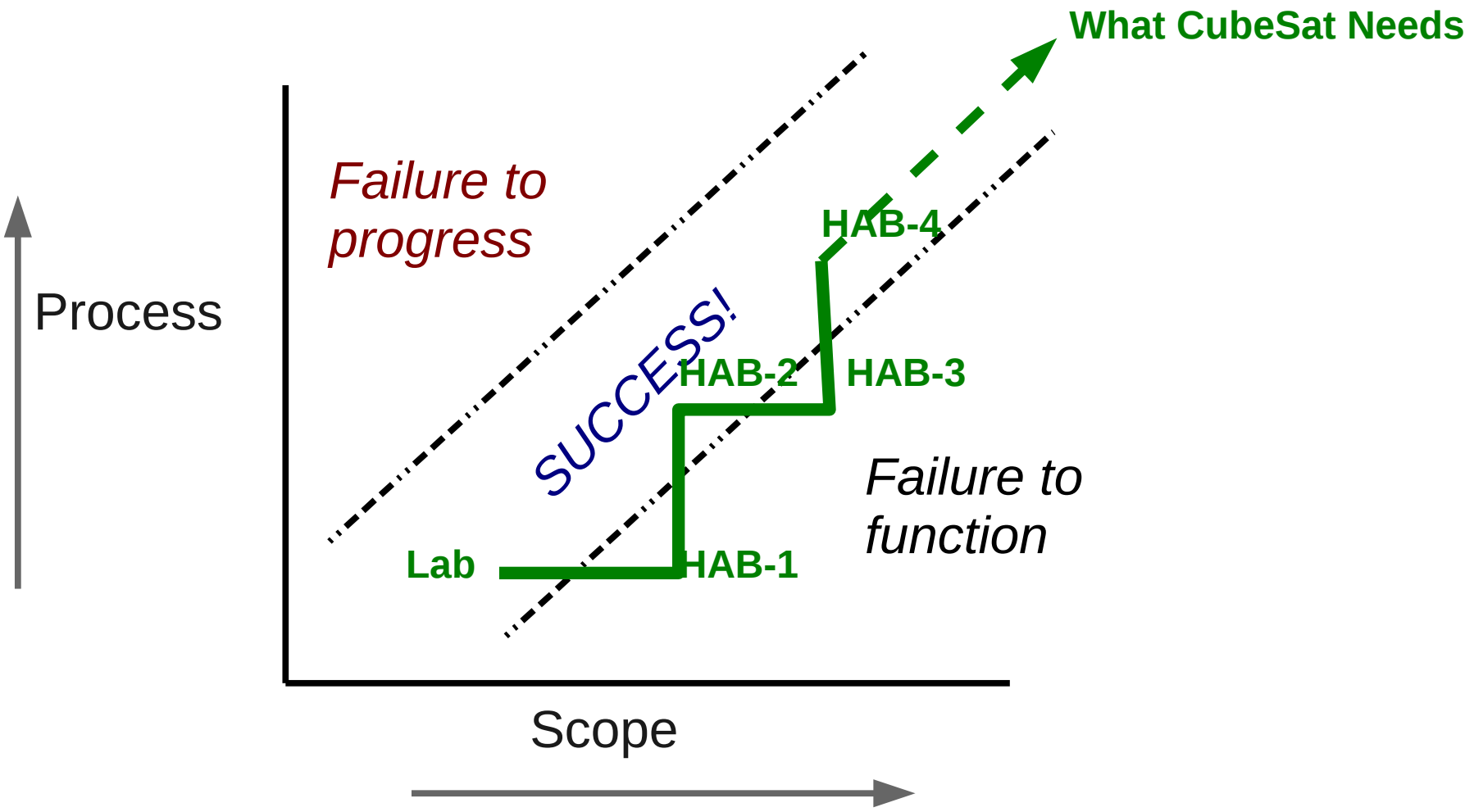
CAPITOL TECHNOLOGY UNIVERSITY
1927



(unscaffolded application of ala carte SE concepts)

Solution: Use failure to teach buy-in

CAPITOL TECHNOLOGY UNIVERSITY
1927



Escalating from Lab builds to HAB up to CubeSat

Backstory: **High Altitude Balloon (HAB)** flights

CAPITOL TECHNOLOGY UNIVERSITY
1927

Student-conceived student-led projects.
Once/semester 1-day flights
Payloads need only survive 2 hours, are recovered

Iterative work: each flight improves or adds to the project.

Rapid dev: teams work to a) fix what broke last flight and b) improve what payload can do for new flight.

Student projects- TRAPSat & Hermes

CAPITOL TECHNOLOGY UNIVERSITY

1927

Case 1 : Interface Control Documents

CAPITOL TECHNOLOGY UNIVERSITY
1927

Case: 2-person hardware/software senior project team ran into schedule conflicts. Fighting ensued; were told to stop teaming and work solo

Solution: Prof. reiterated previous advice to specify interfaces. Then build to the interface, not to the other's design.

As long as they can meet the interface spec, the project can move forward with them working asynchronously.

Outcome: *Team stopped conflict, were able to get project(s) approved.*

Case 2: Review and Testing

CAPITOL TECHNOLOGY UNIVERSITY
1927

TRAPSAT HAB payload-- student-led, student-build aerogel-based particle capture device flying at 80,000 feet.

Case: In Flight 1, power supply attached poorly, caused 200 repower/reboot cycles within 2 hours, resulted in corrupt SD card and all data lost.

Solution: 'This would have been caught in testing, preventing the loss of data all together.'

Outcome: *For future flights, students scheduled testing and improved documentation/capture of past flights.*

Case 3: Test Plans

CAPITOL TECHNOLOGY UNIVERSITY
1927

Case: 4 teams (19 students) soldering up Arduino boards. Dislike writing a test plan-- they built it, they know what to test.

Problem: They must run tests using a different team's test plan. Also get feedback from 'rival' team on their plan.

Outcome: *Afterwards, they appreciate what is in a good test plan.*

Use of Rapid Dev to teach SE

CAPITOL TECHNOLOGY UNIVERSITY
1927

1) Ensure SE & QA taught in curriculum.

2) Per project, mention but do not enforce SE.

3) Wait for failure. Do not penalize it.

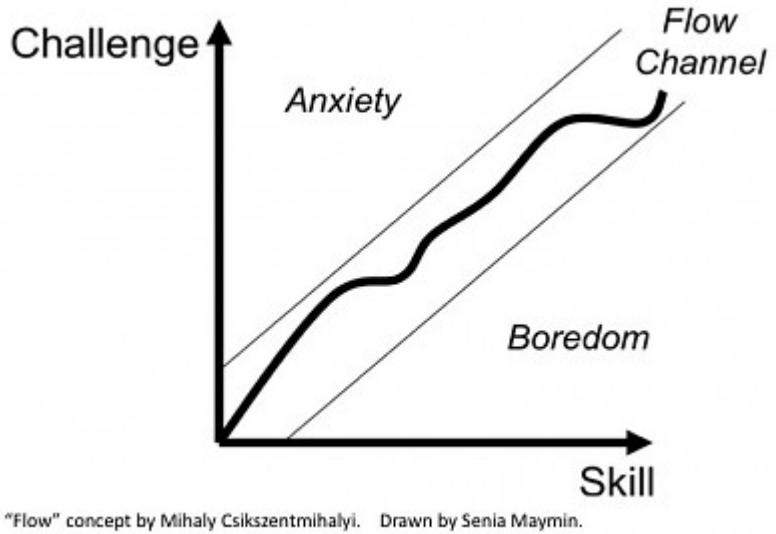
4) Highlight process that will prevent repeat of mishap.

5) Assert an additional SE or QA as well.

6) Let students evolve in their acceptance of what was only theory before.

References

CAPITOL TECHNOLOGY UNIVERSITY
1927



*** Concept of 'Flow'**,
Csíkszentmihályi and Nayamura (2002)

Posters at this conference:

TRAPSAT *Trapping of Space Debris Using Aerogel* by Petrov et al

HERMES *Rocksat-X Flight/QA Testing* by Hansen, Ho, Del Cid, Serano

'Goldilocks' references::

* Goldlilocks for Software Assurance: MDDI (2006): http://www.mddionline.com/article/goldilocks-principle-approaches-software-validation
* Goldilocks for Agile/Scrum: Kent (2014) http://blogs.msdn.com/b/stuart_kent/archive/2014/10/24/kanban-mmfs-and-the-goldilocks-principle.aspx

Quality Assurance via Successive HAB Flights

CAPITOL TECHNOLOGY UNIVERSITY
1927

Questions?

Capitol Technology University
Astronautical Engineering Department

A. Antunes, *Associate Professor*
R. Schrenk, *Research Assistant and student*
C. Gesterling, *Lab Manager and student*
N. Weideman, *Adjunct and masters student*

Open-ended question: is it fair to set up a student project for failure in order to teach a QA or SE principle?

Addendum:

CAPITOL TECHNOLOGY UNIVERSITY
1927

On "Technical vs Engineer"

Technician-minded approach:

- project often not well-defined
- assumptions based on the past drive decisions
- little planning done, challenges tackled as they arise
- formal process kept in the technicians head
- often state 'no time to test!'

Is Scaling a Guide?

Simple builds that are lower tech, lower budget can fail early, fail often. No real impetus to apply QA.

Larger teams and higher cost projects can mandate process & QA, but still need buy-in.

Under rapid dev and 'working to working', you can tackle either project size and allow for frequent failures as 'teachable moments'.