Testing autonomous vehicles

Testing autonomous vehicles

Explains

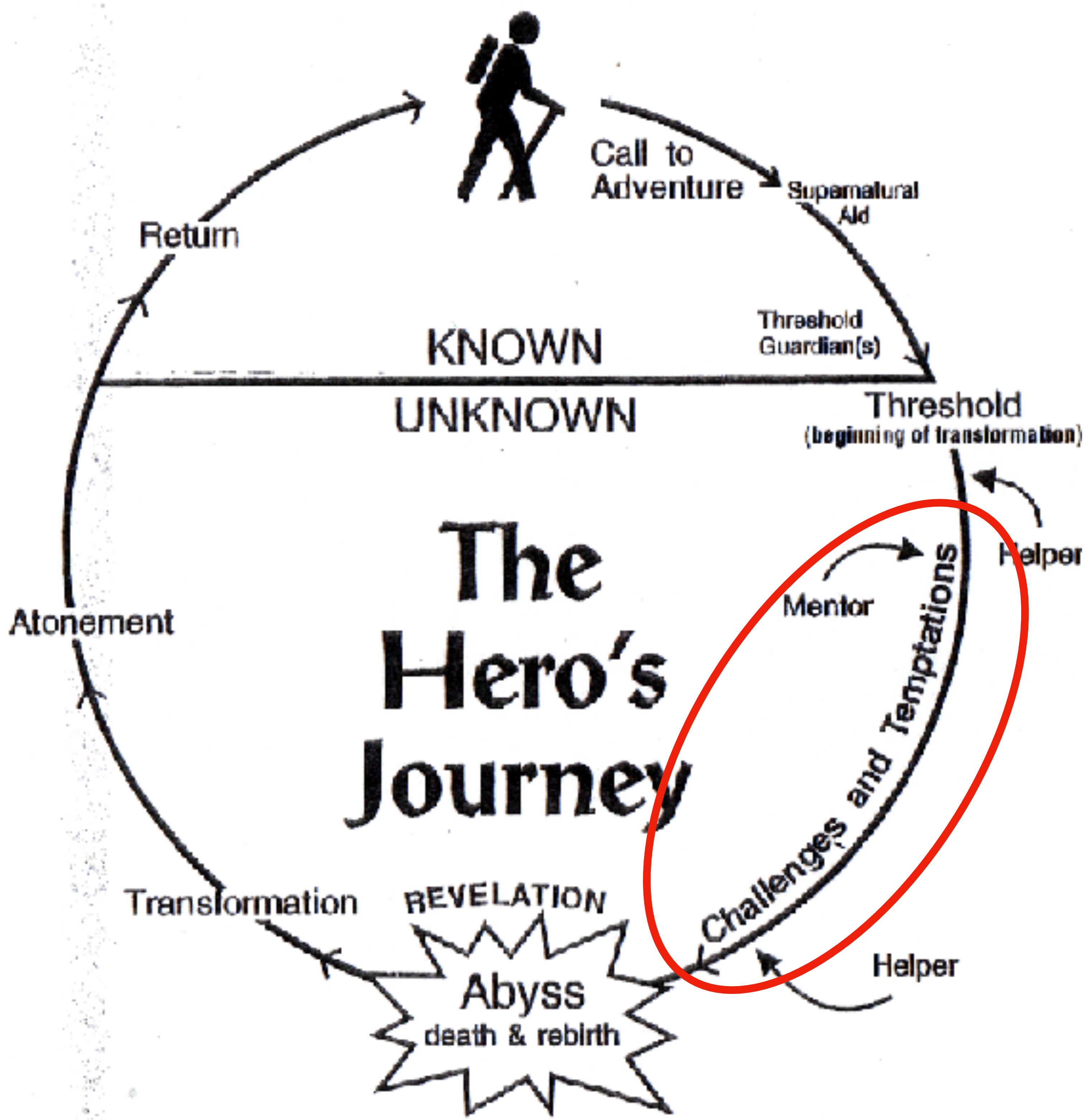
- Verification and validation
- Testing strategies: from unit tests to road tests.

Prerequisites

- Basics of autonomy

Credits

- Ted Steiner (DRAPER) - Spring 2016
- Andrea Censi (ETH Zurich) - Fall 2017


Call to Adventure
Supernatural Aid
Return
Threshold Guardian(s)
KNOWN
UNKNOWN
Threshold
(beginning of transformation)
Helper
Mentor
Challenges and Temptations
The Hero's Journey
Atonement
Transformation
REVELATION
Abyss
death & rebirth
Helper

Verification and Validation (V&V)

- **Mission:** what the system should accomplish;
  - often informally specified; subject to the vagueness of human minds and multitude of stakeholders.
- **Specification**: A (formal) description of how the system must behave.
- **Verification**: Convincing oneself that the system conforms to the specification.

  "Are we building the system right?"

- **Validation**: Convincing oneself that the system, when conforming to the specification, accomplishes the mission.

  "Are we building the right system?"

The difference between Verification and Validation

- **Mission:** *We must save the prince/princess kidnapped by the dragon.*

- **Specification:** *We need a way to kill the dragon.*

- **Proposed solution:** *We are going to launch a satellite equipped with a powerful laser that can kill the dragon from orbit.*

- **Verification:** *An extensive campaign of tests assures us that our proposed satellite design can kill any known species of dragon from orbit.* **Verification passed ✓**

- **Validation:** *Somebody realizes that the princes/princesses tend to die in the resulting heat wave.* **Validation failed ✗**

  Revised specification: We need a way to kill the dragon while the prince/princess remains alive.

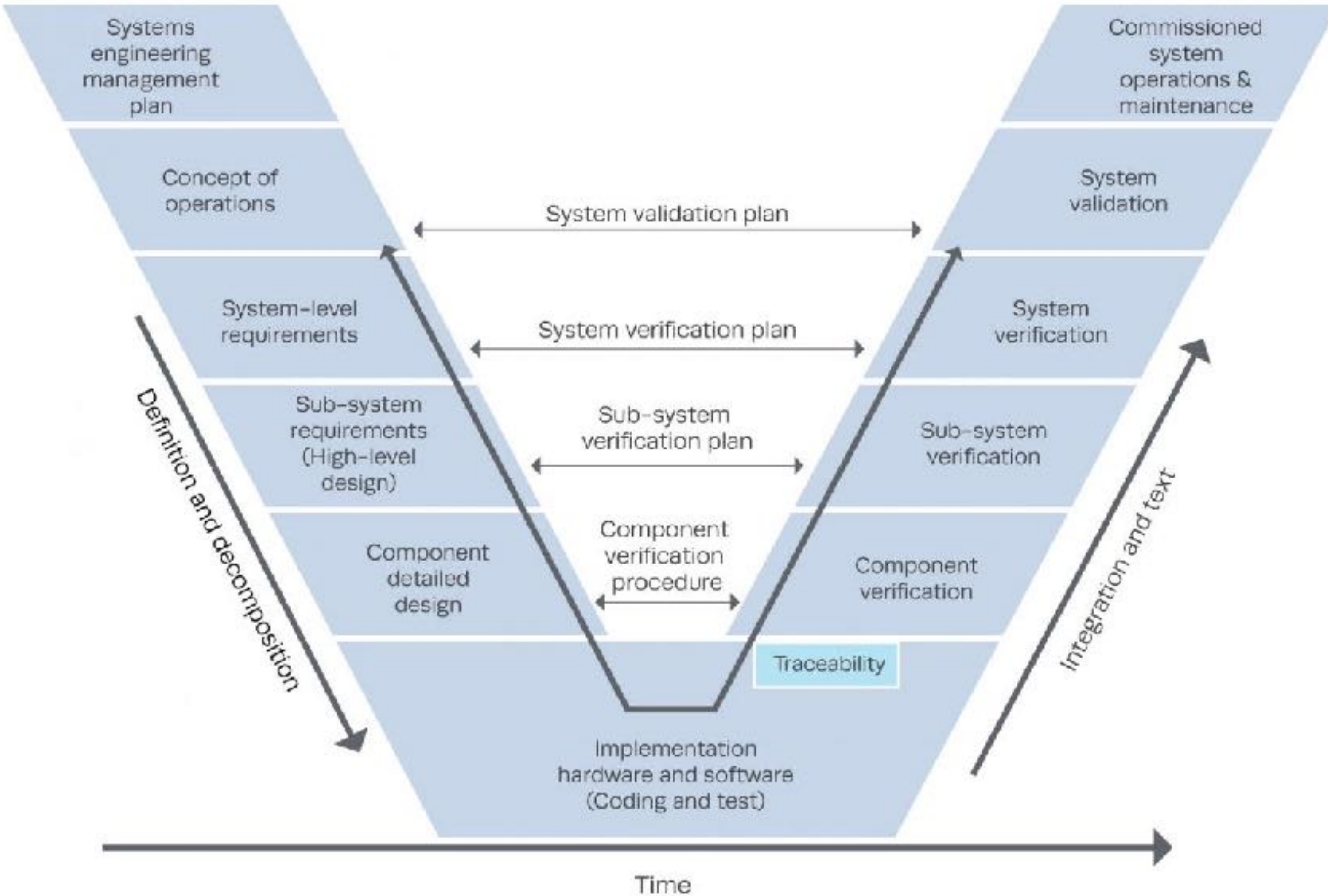
System, subsystem, component V&V

- Validation and Verification (V&V) apply to system-level, subsystem, and component level.

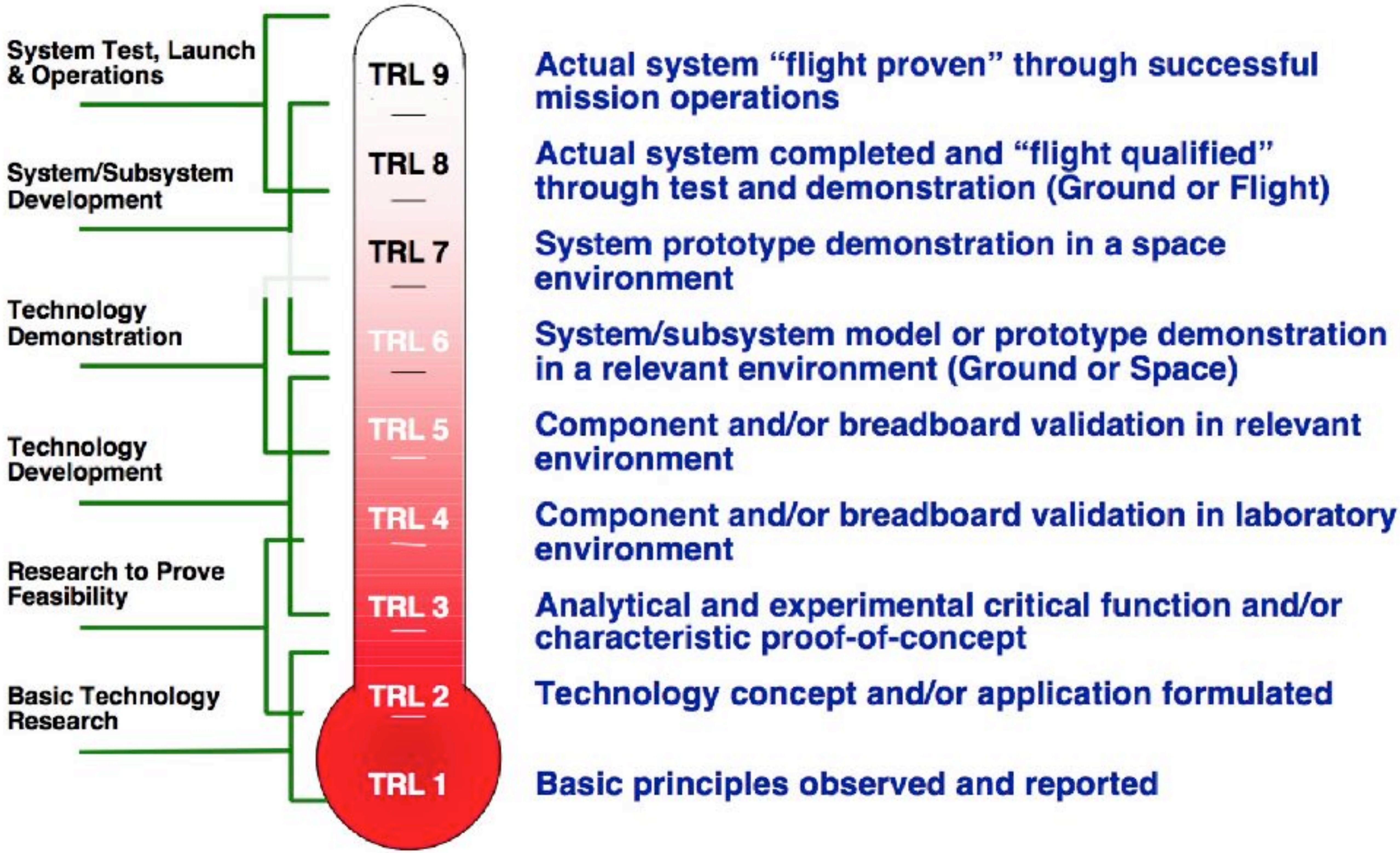
Duckietown examples

- **System-level specification**: Create an autonomous mobility-on-demand system for the city of Duckietown such that the average wait time is less than 5 minutes.
- **Subsystem-level specification**: The subsystem (anti-instagram + line detection + lane filter + controller) must be able to follow a lane and stop at the stop line with accuracy 2cm / 5 deg.
- **Component-level specification**: The lane filter must provide a localization estimate accurate to 0.5 cm / 3 deg with frequency 10 Hz and latency less than 100 ms.

Verification and Validation (V&V)



NASA/DOD Technology Readiness Level



Different types of tests

- **Unit tests**: single function, single module
- **Integration tests**: multiple modules, testing their communication/interaction
- **Functional tests:** multiple modules, testing the end-to-end functionality, often with synthetic data.
- **Regression tests:** multiple modules, testing on real data
- **Simulation tests:** testing in simulation; multiple fidelity levels.
- **Hardware in the loop (HWIL) tests:** tests performance (cpu, network, ...)
- **Flight tests**: closed course (controlled conditions), or actual roads
- **Acceptance testing:** does the customer like the product?

Unit tests

- **Unit tests** verify the functionality of single modules or part of a module (even a single function), in isolation from the rest of the system.
- Speed up debugging - find issues quickly in complex code
- Speed up integration on a new system or environment
- Prevent unexpected changes of functionality.

Duckietown examples

- **Camera geometry tests:** Sample random calibration information. Take an (x,y,0) point, project it to image space, and re-project back to the ground. We should obtain the same coordinates.

Most problems are simple stupid mistakes

The map is not estimated correctly. Maybe I should have used a better preconditioner for that matrix inversion…

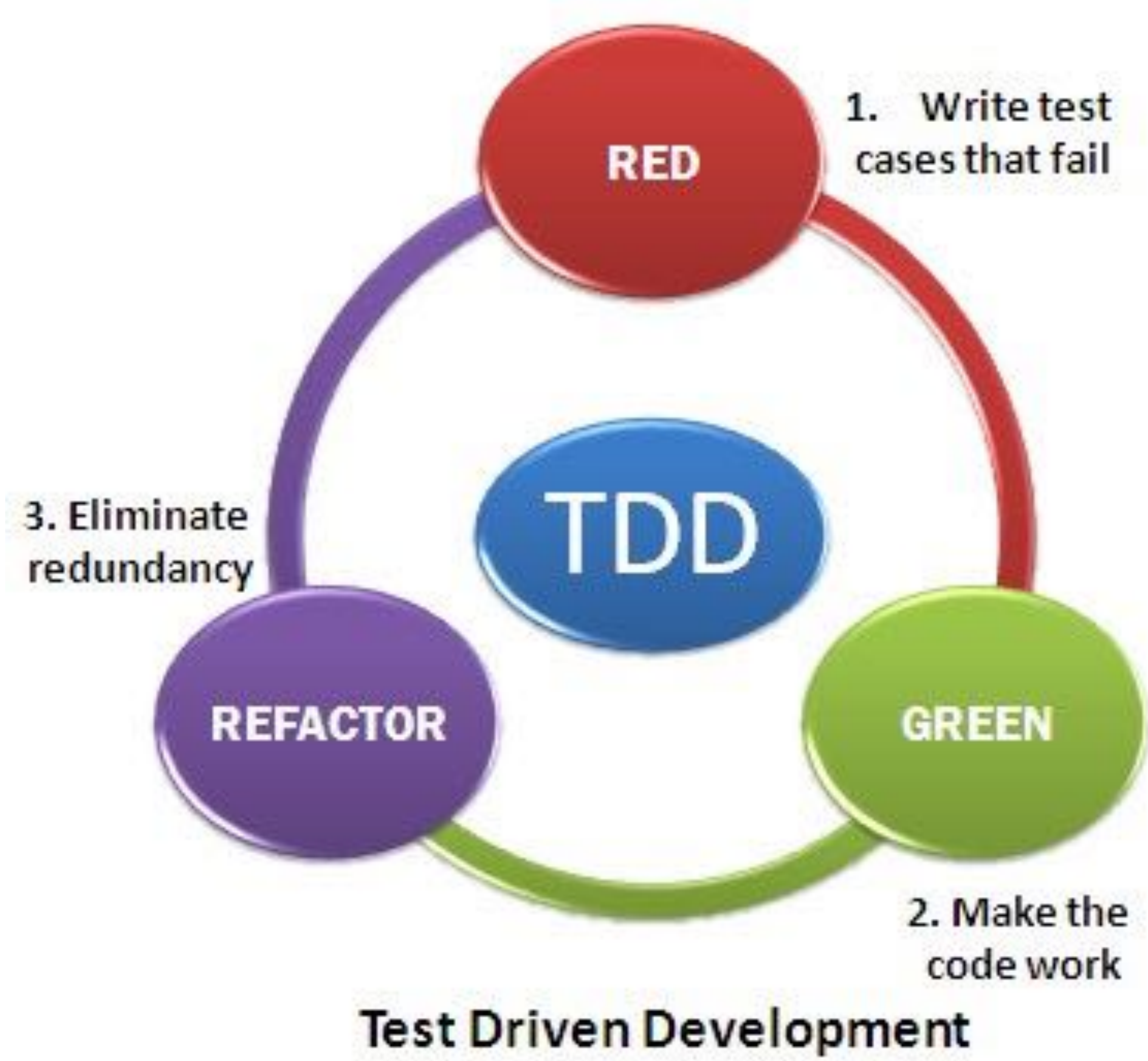
Reality: you forgot a minus sign.

Learning is not converging; maybe I should try implementing that fancy new method everybody is talking about…

Reality: half of the datasets are not loading due to a bug in the dataset-indexing code that you wrote in a hurry last year. The last 12 months of experiments have been useless. You wasted one year of your life.

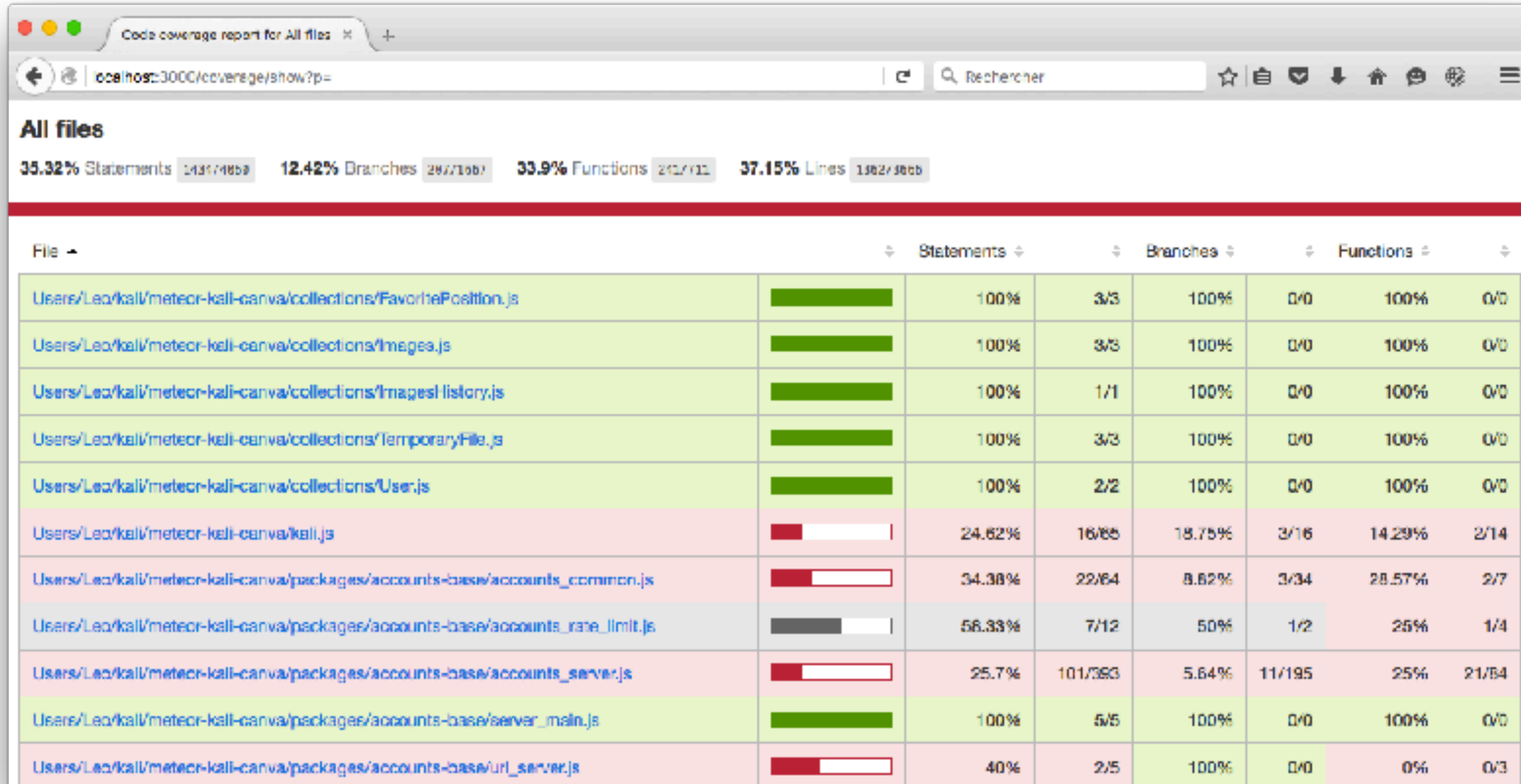
Test-driven development (TDD)

- Write the tests **before** writing the code.



Code coverage

- Code coverage: % of lines of code that are reached by the tests.



File		Statements		Branches		Functions	
Users/Leo/kali/meteor-kali-canva/collections/FavoritePosition.js		100%	3/3	100%	0/0	100%	0/0
Users/Leo/kali/meteor-kali-canva/collections/Images.js		100%	3/3	100%	0/0	100%	0/0
Users/Leo/kali/meteor-kali-canva/collections/ImagesHistory.js		100%	1/1	100%	0/0	100%	0/0
Users/Leo/kali/meteor-kali-canva/collections/TemporaryFile.js		100%	3/3	100%	0/0	100%	0/0
Users/Leo/kali/meteor-kali-canva/collections/User.js		100%	2/2	100%	0/0	100%	0/0
Users/Leo/kali/meteor-kali-canva/kali.js		24.62%	16/65	18.75%	3/16	14.29%	2/14
Users/Leo/kali/meteor-kali-canva/packages/accounts-base/accounts_common.js		34.38%	22/64	8.82%	3/34	28.57%	2/7
Users/Leo/kali/meteor-kali-canva/packages/accounts-base/accounts_rate_limit.js		58.33%	7/12	50%	1/2	25%	1/4
Users/Leo/kali/meteor-kali-canva/packages/accounts-base/accounts_server.js		25.7%	101/393	5.64%	11/195	25%	21/84
Users/Leo/kali/meteor-kali-canva/packages/accounts-base/server_main.js		100%	5/5	100%	0/0	100%	0/0
Users/Leo/kali/meteor-kali-canva/packages/accounts-base/url_server.js		40%	2/5	100%	0/0	0%	0/3

Code coverage

- Code coverage: % of lines of code that are reached by the tests.
- Risk: writing lots of easy tests that give a false sense of security.

Goodhart's law
"When a measure becomes a target,
it ceases to be a good measure."

Integration tests

- **Integration tests** verify that multiple modules work well together; e.g. the data format is compatible.

Two successful unit tests, zero integration tests.

Functional tests

- **Functional tests** verify the end-to-end functionality provided by a subsystem, often with synthetic data.

Duckietown example

Test the pipeline ground projection + lane filter with synthetically generated line detections. Fix a pose **q,** generate line detections at **q,** check estimated pose, which should be equal to **q**.

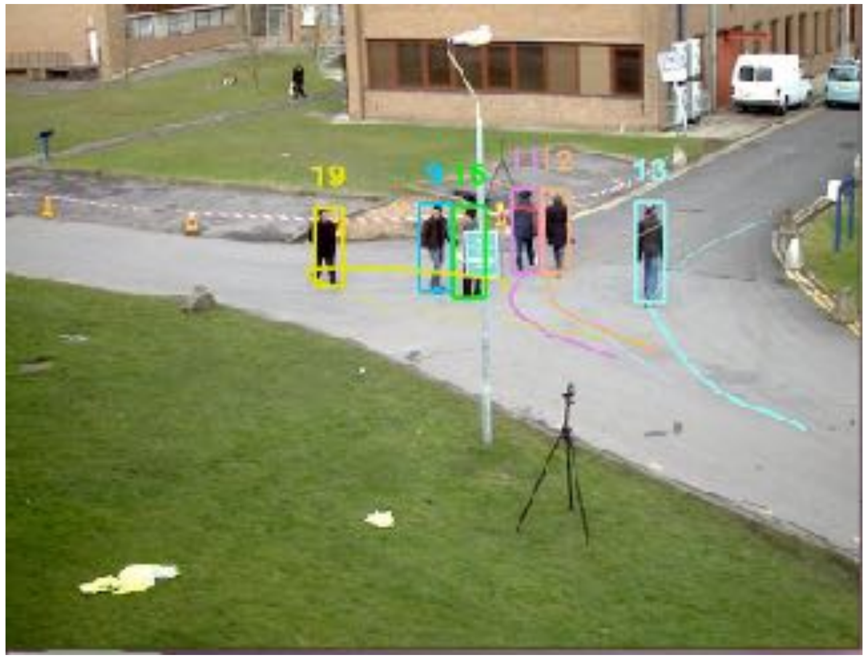
Regression tests

- **Regression tests** verify the end-to-end functionality provided by a subsystem on real data (logs).
- May involve the use of **annotated ground truth on realistic data.**
  - pose information from sensors you might not have in production (motion capture, differential GPS)
  - sensor annotations (images for cameras, point clouds for lidar, tracks for sonar)
- May involve data taken in **controlled conditions.**

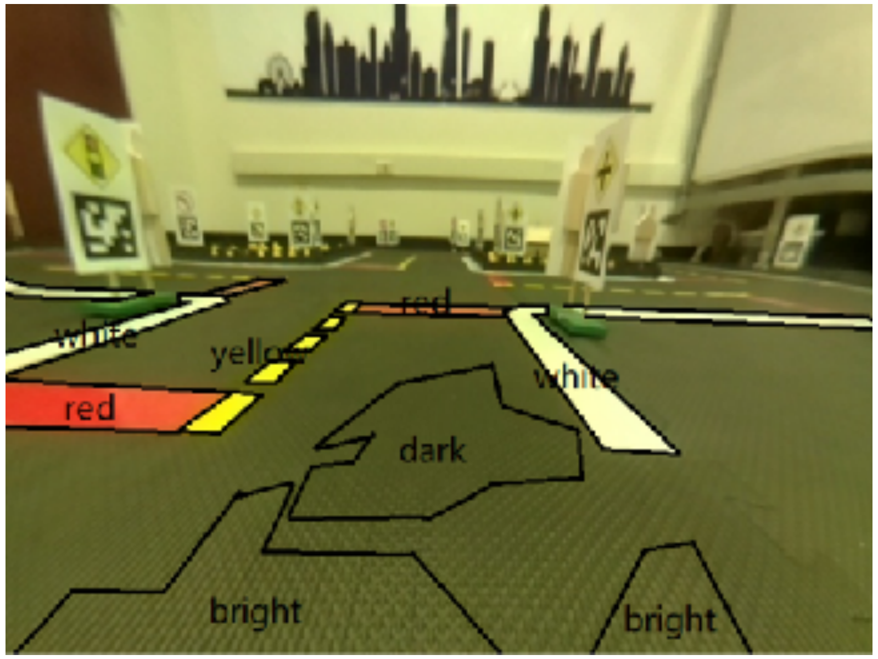
Duckietown examples
Check that we can detect all April tags at a distance of 40 cm.
Check that we have >99% reliability in duckie detection.

Example of image annotations

bounding boxes



polygons



pixel-level segmentation

Data annotations in Duckietown

- We will outsource the data annotations to the company **thehive.ai**
- **First phase:** the teams explore the possibilities for data annotation.
  1. Teams collect the data that they need.
  2. Sign up on **thehive.ai** and setup the annotation efforts.
  3. See results and iterate.
  - Every educational account gets $100 in credits (= 1000 images labeled). Ask us if more is needed.
- **Second phase:** We will consolidate the data annotation efforts.
  - Goal: create a curated quality dataset that can be used for many years.
- **Third phase:** reusable testing framework will be created by the supervised learning team.

Simulation tests

- **Simulations: use responsibly.**
  - **"Simulations are doomed to succeed":** developing first in simulation will lead to failure.
  - Simulations are necessary in robotics to do closed-loop control tests.
  - **Different levels of fidelity** to achieve different goals**:**
    - *Fully photorealistic 3D world with physically based rendering:* use to explore limits of perception.
    - *Not photorealistic*: may still be useful as functional tests for perception.
    - *Ignore sensors and just simulate perception errors:* useful if you have a good statistical model of perception.
    - *Ignore sensors and dynamics*: may still be useful for multi-agent systems

Hardware-in-the-loop (HWIL) tests

- **“Hardware in the loop” (HWIL): bench test with (part of) the system hardware.**
- **HWIL tests are typically used to measure the performance**
    - Example: test total latency with CPUs under full load
    - Example: test reliability of network under stress
    - ...
- At runtime, **health monitoring modules** will check these values.

Duckietown example: Check that the total latency is less than 100 ms when run on a Raspberry PI.

Testing the complete system

- **Tests in controlled conditions:** In the end, the rubber must meet the road…
  - Typically use a **catalogue of maneuvers**
    - intersection with stop signs
    - intersections without stop signs
    - passing on the left, on the right, …
- **On road tests measure system-level performance metrics.**
  - Example: number of takeovers per mile

Uber's test track

University of Michigan's test track

Acceptance tests

- **Acceptance tests are useful for validation**: does the customer accept the product, or should we go "back to the drawing board"?