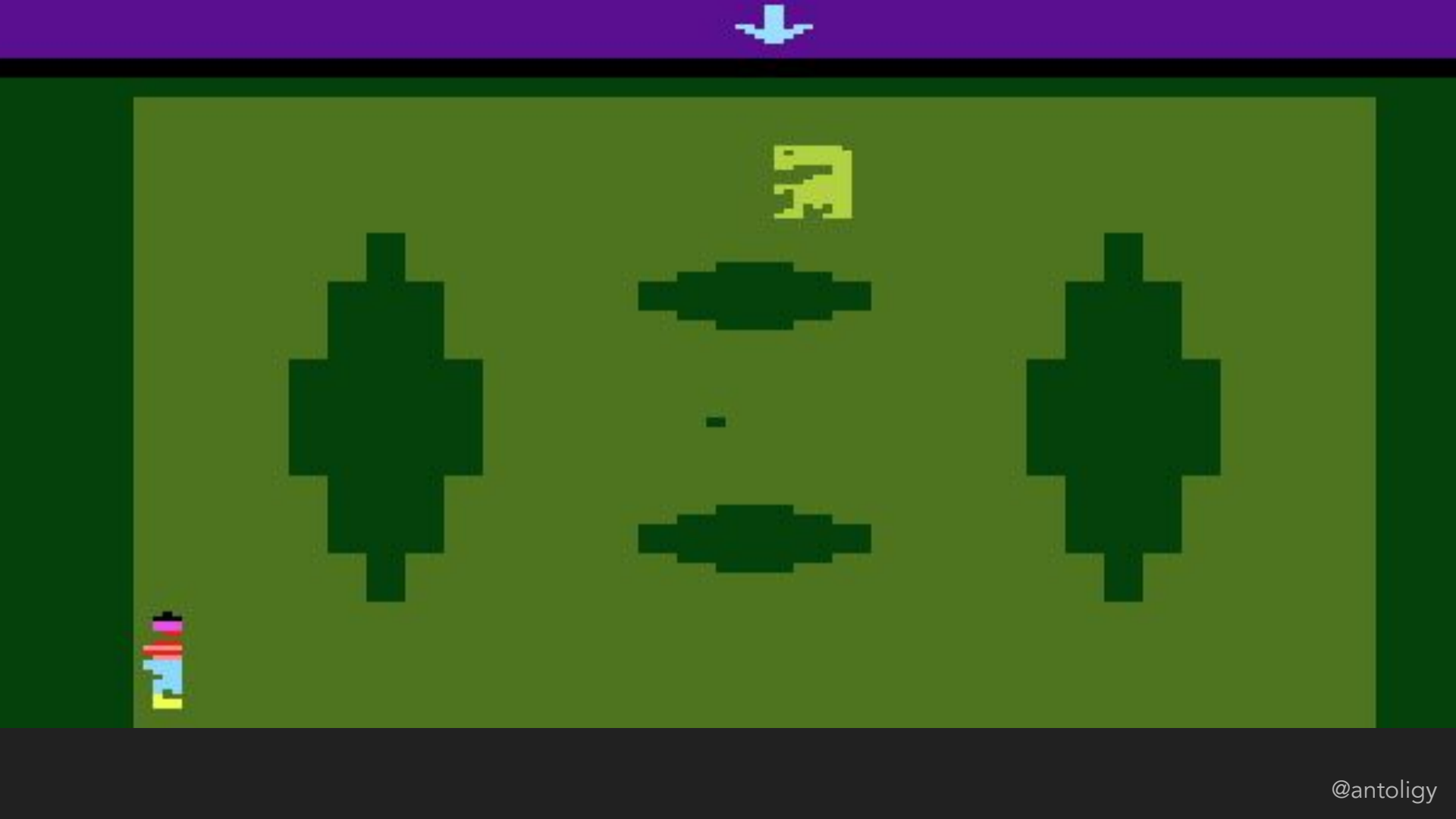
# QA is not Quality

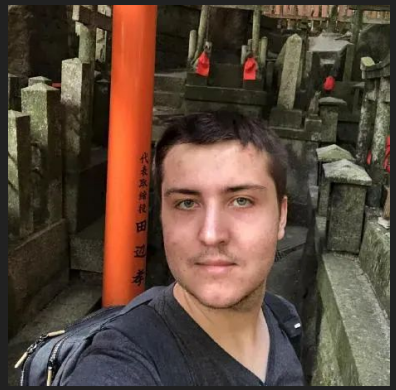

SUPER METROID
Only For
Nintendo
Official
Nintendo
Seal of Quality
SUPER NINTENDO
ENTERTAINMENT SYSTEM



# Who am I?

“Web performance advocate and aspiring polyglot”

Twitter — @antoligy
Web — alexwilson.tech



# Take-away

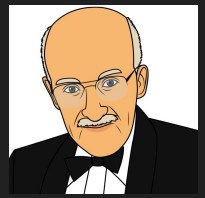
- What Software Quality actually is
- How we can measure it
- Practical ways to improve the quality of what we’re building


So what actually is quality?

# Let’s ask the experts!

The difficulty in defining quality is to translate future needs of the user into measurable characteristics, so that a product can be designed and turned out to give satisfaction at a price that the user will pay.
— William E Deming

Quality is a customer determination, not an engineer's determination, not a marketing determination, nor a general management determination. It is based on the customer's actual experience with the product or service, measured against his or her requirements — stated or unstated, conscious or merely sensed, technically operational or entirely subjective — and always representing a moving target in a competitive market.
— Armand V Feigenbaum



The word quality has multiple meanings. Two of these meanings dominate the use of the word: 1. Quality consists of those product features which meet the need of customers and thereby provide product satisfaction. 2. Quality consists of freedom from deficiencies. Nevertheless, in a handbook such as this it is convenient to standardize on a short definition of the word quality as "fitness for use".
— Joseph M Juran

# In simpler terms...

- Being “fit for purpose”, and serving a user need.
- Free from deficiencies.
- Being able to easily meet the needs of the future.

# Quality Model: ISO-25000 (SQuaRE)

# Functional Suitability

What is it?

A system provides functions that meet stated and implied needs when used under specified conditions

How do you measure it?

- Functional completeness: CSAT, Traceability matrix *(requirements vs test coverage)*

# Functional Suitability

How do you improve it?

- Hire a Business Analyst and QA to define *complete* functional requirements
- Implement user-testing

Example

*The time we reverse engineered a product and its functional requirements while changing its technology stack: It would have been far easier if we had mapped out what we knew.*

# Performance Efficiency

What is it?

The performance relative to the amount of resources used under stated conditions

How do you measure it?

- “Core Metrics” — # Latency, # Request-Rate, # Errors
- User-centric metrics — # Time to interactive, # Time to first meaningful interaction
- Capacity — Load Testing
- Resource Utilisation — Tracing

# Performance Efficiency

How do you improve it?

- Choose the right technologies for the problems you are trying to solve
- Instrument “core metrics” across your stack.
- Load-test under realistic scenarios.
- Link performance improvements to business metrics (e.g. conversion)

Example

*The time a website’s performance problems for years turned out to be an unnoticed, unnecessary synchronous API callout blocking every request.*

# Usability

What is it?

A system can be used by specified users to achieve specific goals with effectiveness, efficiency and satisfaction in a specified context of use

How do you measure it?

- Accessibility — WCAG 2.1 score
- User-friendliness — # User error rates, # L1 support-tickets, CSAT
- Operability — # Production incidents

# Usability

How do you improve it?

- Automate accessibility testing
- Regularly conduct user interviews and perform user research
- Test on real user devices
- Invest in onboarding every type of user a system or product might have

Example

*The time I built multi-language support into an events website to improve the accessibility for its non-English-speaking users, but, didn't train or make it easy for administrators to add new locales.*

# Reliability

What is it?

A component functions consistently under specified conditions for a specified period of time

How do you measure it?

- Stability — Uptime, # of reported outages
- Availability — MTBF (mean-time-between-failures: mean-time-to-failure + mean-time-to-repair)

# Reliability

How do you improve it?

- Chaos-suite: Simulate disasters and failures before they hit you
- Knowledge-sharing exercises in the technology stack
- Apply SLAs and circuit-breakers to dependencies
- Instrument applications and services
- Invest in Observability tooling

Example

*The time Puppet accidentally deleted a production database and server, but because we used Puppet and orchestrated every aspect of AWS, we were able to recover within two hours.*

# Security

What is it?

A system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization

How do you measure it?

- Risk — OWASP Risk Rating (Likelihood * Impact)
- Auditability — # of user actions not recorded/attributed to a user
- Integrity — # of areas where PII may be found
- Compliance with standards such as PCI-DSS

# Security

How do you improve it?

- Scan dependencies for known vulnerabilities (e.g. Snyk)
- Vulnerability scanning of entire system (e.g. automated like Qualys)
- Bug-Bounties
- Role-Based-Access-Control, and Attribute-Based-Access-Control

Example

*The time a Drupal website was hacked and would randomly serve cialis adverts to search-bots only.*

# Maintainability

What is it?

A system can be efficiently and effectively modified to improve it, correct it or adapt it to changes in environment, and in requirements

How do you measure it?

- Testability — % unit code coverage, % scenario coverage, % feature coverage
- Understandability — # of lines of code, average file length, adherence to conventions
- Modifiability — Cyclomatic complexity, Cohesion

# Maintainability

How do you improve it?

- Invest heavily in developer experience
- Lean on unit-testing to improve developer confidence
- Invest in code-review automation (e.g. Codacy)
- Reduce # caching layers and improve their utilisation
- Document your systems inside-out
- Invest in Observability tooling

Example

*The time when revisiting our architectural documentation and increasing the verbosity of our logging helped us diagnose an intermittent problem as being Redis replicating data asynchronously.*

# Portability & Compatibility

*I'll be honest: These are not things I think I have ever successfully solved. They are included for "completeness".*

## Portability

### What is it?

A system, product or component can be transferred from one hardware, software or other operational or usage environment to another.

### How do you improve it?

Containerize

## Compatibility

### What is it?

A component can exchange information with other components, and/or perform its required functions, while sharing the same hardware or software environment.

### How do you improve it?

Namespacing, Containerize

# Practical ways of improving quality

There are also some general principles we can implement.

- Defining “Done”
- Experiment and iterate
- Test in production

# Practical Suggestions: Define “Done”

- Being dogmatic about every principle of quality will get us nowhere fast, so prioritise based on what is important for our users.
- Periodically revisit this definition as a team, asking:
  - Do our stakeholders and users understand how this works?
  - Do we think this has helped improve our quality?
  - Do our users feel the same way?
- Work with “quality champions” early-on to define how new bodies of work will be tested.

https://www.scrum.org/resources/blog/walking-through-definition-done

# Practical Suggestions: Experiment and iterate!

- Get feedback from stakeholders and users early and often.
  - As quality improves, it will be easier to manage further changes upwards.
  - Team-members and developers in other areas of the business are stakeholders too.
- Use a model of quality to measure and experiment with process and tooling, and don’t be afraid to make mistakes.
- Link chosen metrics to tangible, easy-to-understand goals, and evaluate success based on trends.

# Practical Suggestions: Test in production

- Production is the only environment where we can prove things work.
- As well as testing safely out of production, make it safe to test in production.
  - Feature-flagging to decouple roll-out of features, from roll-out of code, so manual tests are possible in prod before showing users a broken experience.
  - Canary releases to test with small percentage of user-base first.
  - Synthetic monitoring of critical features and journeys.
  - Especially on the web: Enhance progressively, and degrade gracefully. (i.e. if JavaScript errors, continue serving critical functionalities)
- We’re already doing this every time we push to production!

https://martinfowler.com/articles/qa-in-production.html

# Thanks for listening!

Any questions?