TEST STRATEGICALLY, NOT ACCIDENTALLY

Erik LeBel <elebel@pyxis-tech.com>

WHAT ARE WE GOING TO COVER?

- Test plans
- Test strategy's
- Non-functional requirements
- SCRUM
- Unit/integration/specification tests
- Developer concerns without code

WHY DO WE TEST?

- To be sure the system does what we think it does
- The sum of the parts is not always what we expect
- We don’t trust the system to do today what it did yesterday

- We deliver software to users

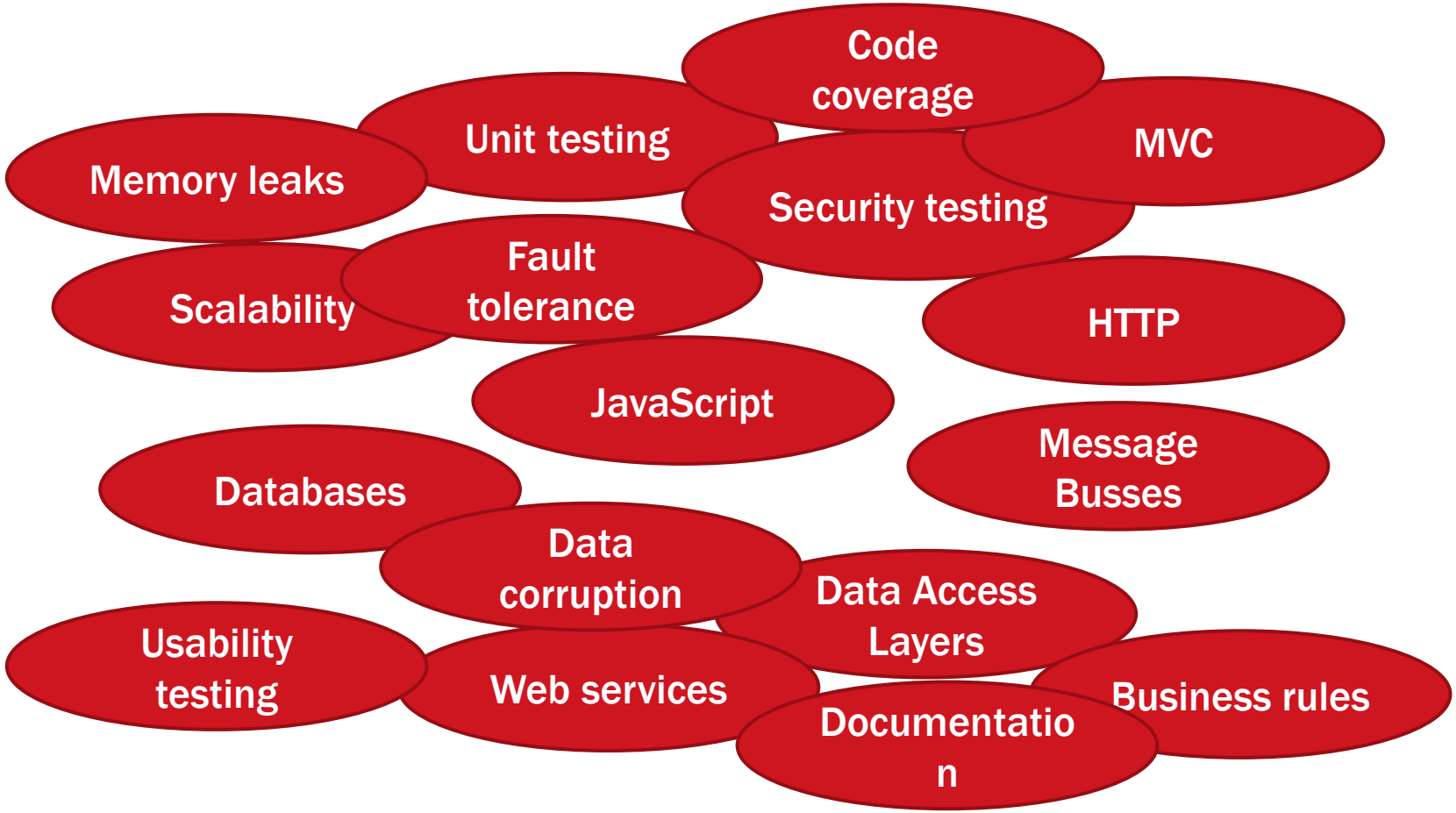
ABOUT THOSE USERS: WHY WE TEST (TAKE 2)

- Because our users have needs and they've expressed things they know they need (a pretty UI, a fancy report)
- Because we're good programmers and we reminded them about things they didn't know they wanted (security, disaster recovery)
- And there are things users want but don't know they want (reliability, and that it does what they think they told us … and not what we think they told us)

SO WHY DO WE TEST?

To help our users attain their goals without getting tripped up by our software or the technology on which it runs

WHY DO WE NEED A STRATEGY?



WE ALSO NEED A STRATEGY BECAUSE

- a commitment to testing isn't very useful if what we're testing has low value
- a commitment to test everything except when its difficult isn't much of a commitment
- a test plan that is excessively expensive can cripple a project
- following a plan even when it is not helping is just plain dumb
- clicking blindly around the software isn't much better

PLANNING FOR SUCCESS

BUILD A TEST STRATEGY THAT

- Identifies the goals of the system and its intended impact on users
- Qualify failure tolerance of the system
- Identifies non-functional requirements that should be considered when defining story success criteria
- Explain the test strategy used to test various parts of the system

IDENTIFY GOALS

- Understand what objectives the software is trying to solve and why
- Understand how the software is intended to impact the users

Note these for inclusion in your test strategy

QUALIFYING FAILURE TOLERANCE

- 9s of uptime
- Longest downtime during peak hours
- Maximum number of issues that can be reported in the period following a release (0 sev1, 2 sev2)

Reported issues	impact
0	System failure
<= 1	Problem without a work-around
<= 4	Problem with work around
<= 10	Minor adjustments

Note these for inclusion in your test strategy

NON-FUNCTIONAL REQUIREMENTS

- correctness
- performance
- security
- recoverability
- scalability
- fault-tolerance
- installability
- usability
- supportability
- learnability
- compliance
- others…

Note these for inclusion in your test strategy

START BRAINSTORMING

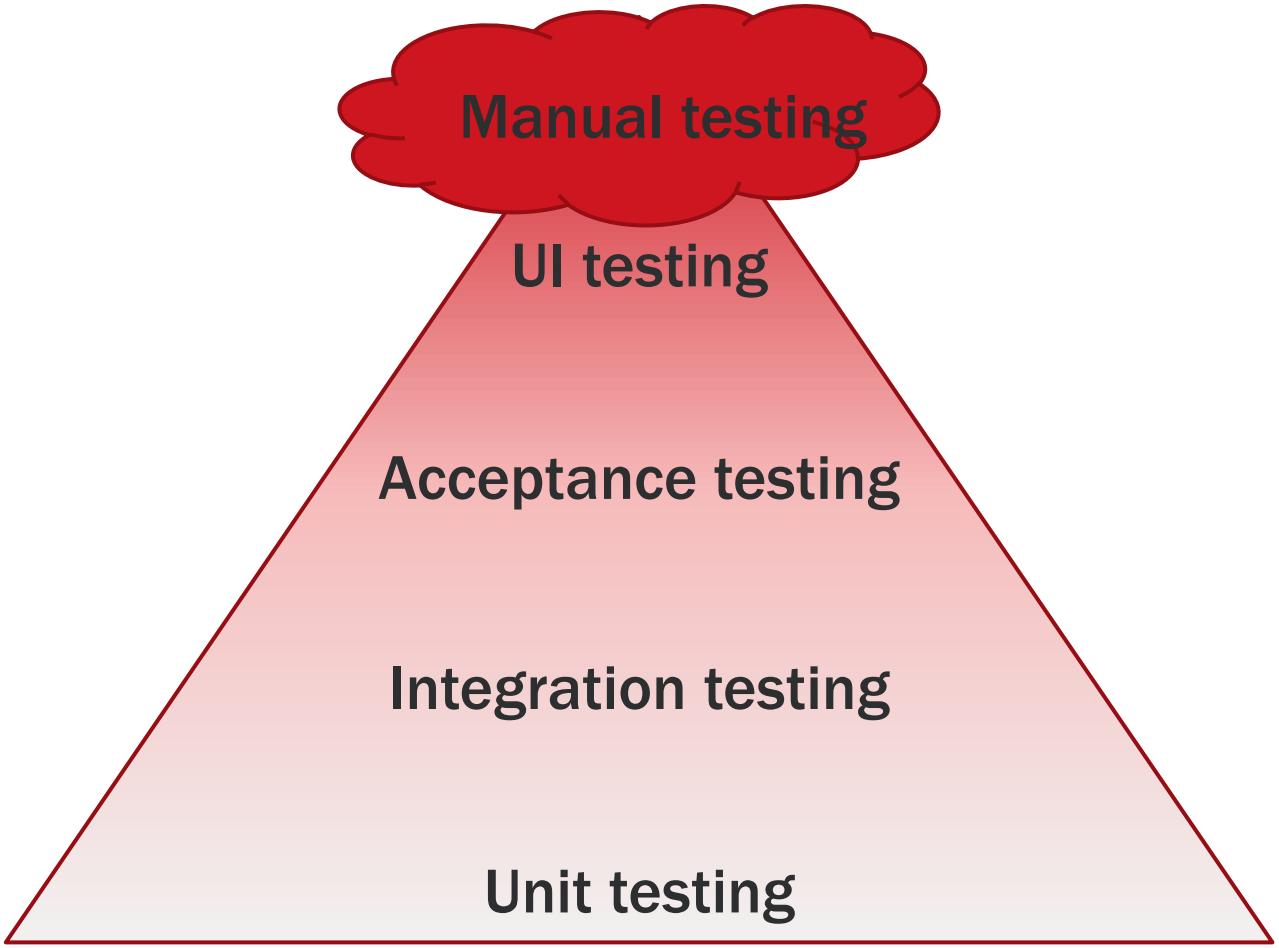
- On how these could be tested and at what cost?
- On what combinations of test tools will help us to ensure these quality criteria are being respected
- How can software architecture facilitate this validation

MANAGE RISK

pick test strategies that will help mitigate risk:

- uncertain requirements or significant complexity – *executable specifications*
- security – *look at existing security catalogs*
- user interfaces – *UI test drivers, layered architecture*
- integration with systems – *contract testing or modular architecture*

REMEMBER THE TEST PYRAMID



PYXIS agile know-how

START BUILDING A TEST STRATEGY…

tests
Unit testing
Integration testing
Cucumber
Selenium
Code coverage 90%

TAKE 2

components	tests
Web component	Selenium
Back-end	Unit/integration testing, cucumber, (cc 100%)

where's the performance testing or security testing?

…MAYBE IT NEEDS MORE DETAIL

component(s)	***tests***
UI	localization, selenium smoke test, unit/integration (70%)
View-Model down	cucumber (use cases - staged)
All public web services	Integration (80%)
rules engine	unit/integration (100%), cucumber, custom perf test harness
authority service	integration testing (100%)
payment processing	unit/integration (100%), manual testing
deployed component	post-deployed smoke test, installed component test

WHAT ABOUT THIS?

component(s)	*how*	*When*
UI	unit/integration (70%)	dev, CI
	selenium smoke test	CI
	localization	sprint, release
View-Model down (ex auth service)	cucumber (use cases - staged)	CI
All public web services (ex auth service, mock DB)	integration	dev
rules engine	unit/integration (100%), cucumber	dev, CI
	custom perf test harness	sprint, release
authority service	integration testing	nightly
payment processing	unit/integration (100%)	dev, CI
	manual testing	sprint, release
full application	post-deployed smoke test, installed component test	CI
	manual testing	sprint, release

YOUR STRATEGY

- System goal: to increase sales of listed products through our web store. Shoppers should have a snappy inventory browsing experience and be able to complete a regular purchase online in less than 3 minutes
- The system should be up 364 of 365 days of the year and during store hours (9-5) never offline for more than 4 minutes. System bugs should never allow for an incorrect transaction to be processed. User cc information must be protected at all cost.
- Stories should consider being validated for:
  - Security, usability, resilience, ...
- The tests should be run

FINALIZE YOUR STRATEGY?

- Present it to your PO
  - And cost of implementing it
- Expect to be challenged on the cost
- Revise it as needed
- And agree to follow it as part of your DOD

TESTING IN SPRINT

BUILD A TEST PLAN

- Identifies use cases
- Add new ones as stories get implemented
- Use a simple tool to capture your script (Mind-map, spreadsheet, wiki…)

WHEN TESTING MANUALLY

- You can use subsets of your plan, but if you are not then check to see if something is missing
- Make a point of noting what you you've tested and capturing it and the result for later reference
- Note what version of the software was under test

AUTOMATE EVERYTHING

- builds that run tests
- security checking
- performance testing
- contract validation
- prod-parallel validation
- even the most bizarre: usability? data recovery?

... and still plan to test manually

TESTING EFFORTS IN THE BACKLOG

- Portion of testing in story to be included in story estimates
- Test setup efforts estimated separately
- Testing sessions to be scheduled in sprint and deducted from team capacity for projection purposes

KEEP IT HEALTHY

- Never leave automated tests in a failed state
- Don’t stop testing things without re-assessing the effectiveness of your strategy

TEST-ENABLING ARCHITECTURE

Design to make it
possible to test the things
you need to test

VALIDATE YOUR TESTS

- With your users
- With other developers

INSPECT AND ADAPT

AT THE SPRINT PLANNING

- challenge stories for unclear or missing:
  - Test cases
  - failure cases
  - NFR
- check your strategy
  - for tests that need to be written
  - for tests that will need to be run

AT THE SPRINT PLANNING

- Bring up any testing spikes that may be needed for upcoming sprints
- The last responsible moment is not when you need to test it but don't know how

REGULARLY REVIEW REVIEW THE ISSUES (BUGS)

To identify:

- problem modules
- weak testing
- fragile integration
- unreliable hardware or systems

and consider how the test strategies or the software architecture can be improved to prevent issues from re-occurring

PYXIS agile know-how

REGULARLY

check your strategy for areas of improvement

-

- have you added deliverables that are not being tested?
- is delivering value?
- is it getting expensive?
- does it have waste?

KEEP IT AGILE

- Aim for the just good enough
- Defer decisions
- Challenge the value
- Adapt your strategy

CONCLUSION

EXISTING ARTIFACTS

There exist standard QA artifacts for a reason. Look to what problem they are trying to solve to see if you can adapt them to help

- test plan
- test strategy
- test cases
- test reports

KNOW YOUR TESTING TOOLS

- Unit testing across technologies your use
  - mocking
- Integration testing
- Executable specifications
- UI piloting
- Standards
- Testing resources

MAKE QUALITY CONCERNS A PART OF ALL CEREMONIES

THIS IS NOTHING NEW

QUALITY IS UP TO YOU

OWN IT!

Erik LeBel <elebel@pyxis-tech.com>

/campus

Learn about **Agility** and **software engineering practices**.

/consulting

What Agility brings to **diagnostics, strategy, project management** and **coaching**.

/studio

Ask for **custom software solutions** to our **expert developers**.



pyxis-tech.com

Questions

Thank you!