AUTOMATING YOUR TEST SUITE

INCREMENTALLY EATING THE ELEPHANT

Sodoto Solutions

See One Do One Teach One

WE'VE BUILT SOME GREAT STUFF!

- We're building more!
- Let's automate the tests!

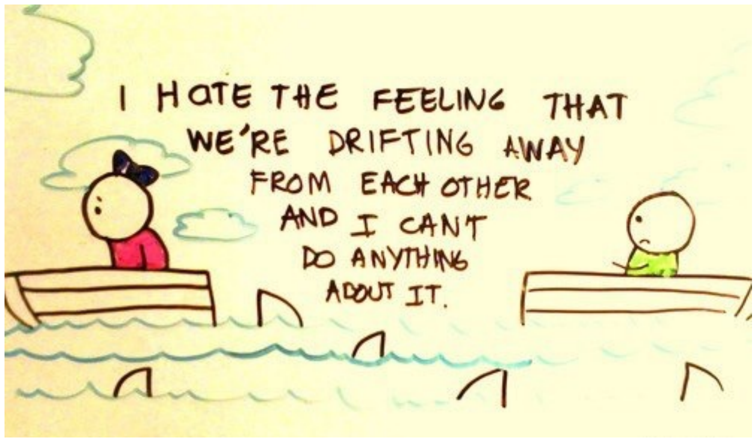
SPECIALIZED TOOLS, SKILLS, DECISIONS, ENVIRONMENTS

SPECIALIZED®

…and quality suffers in the meantime...

ISOLATED, SPECIALIZED TEAM

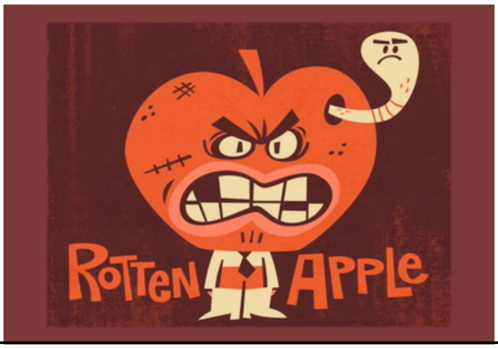
DRIFTING APART…


I HATE THE FEELING THAT WE'RE DRIFTING AWAY FROM EACH OTHER AND I CAN'T DO ANYTHING ABOUT IT.

BIG BANG!

UNKNOWNS, CONFUSION

FAILURE…ROT



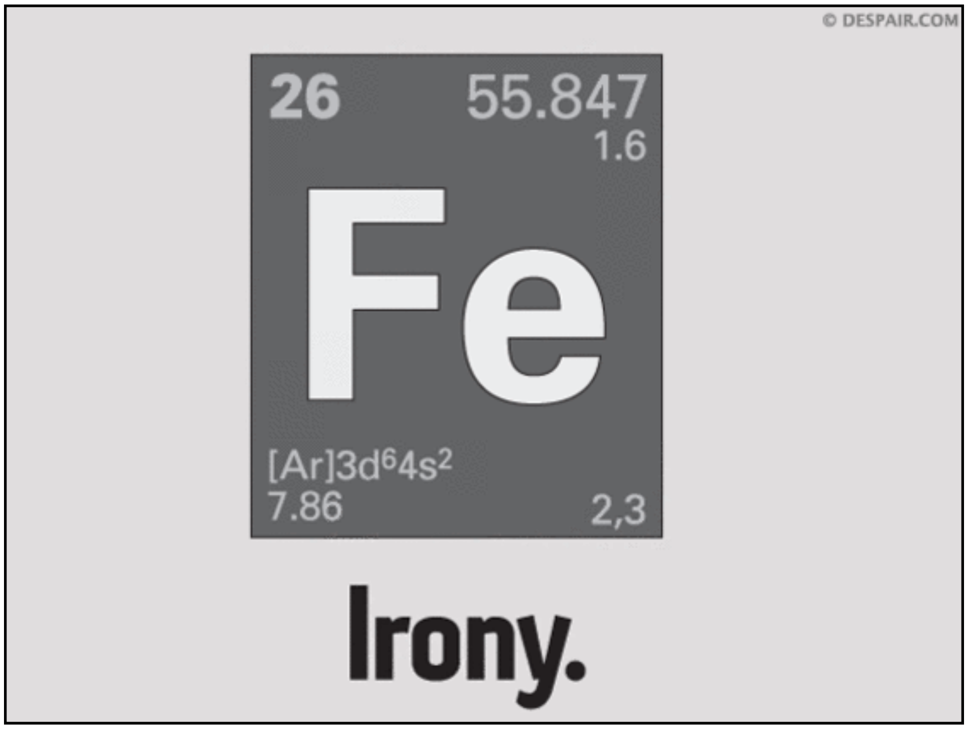
ABANDONMENT…LOST INVESTMENT

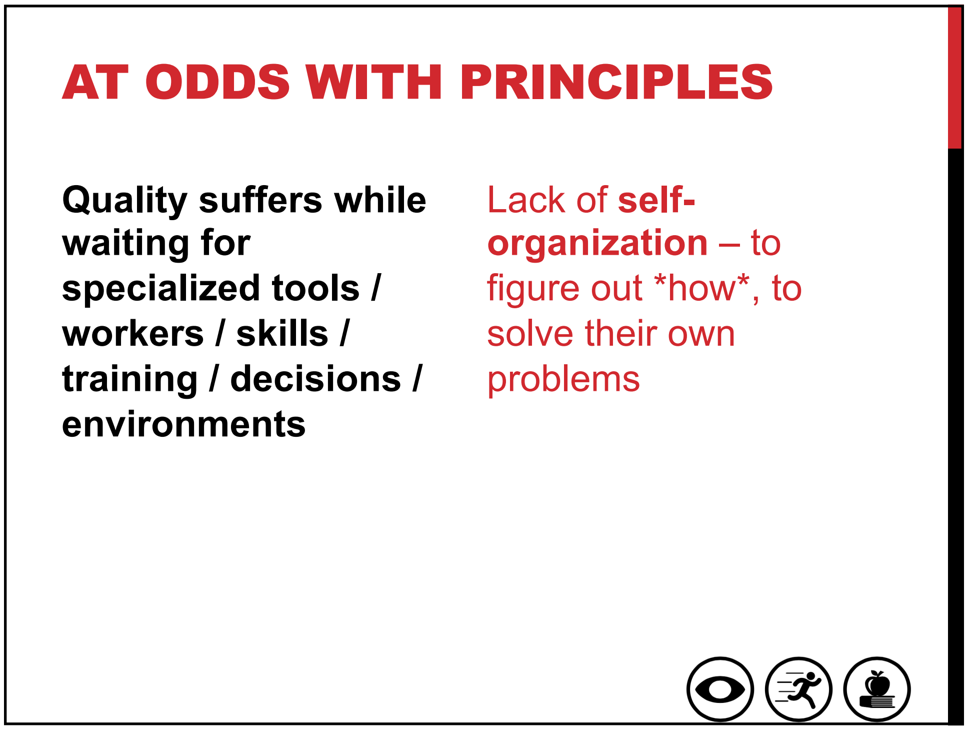
A (TYPICAL?) STORY: SUMMARY

1. Quality suffers while waiting for the specialized tools / workers / skills / training / decisions / environments of test automation
2. Automation built out by a separate, isolated, highly-specialized team
3. New development continues simultaneously, so automation target and current state of product drift apart
4. Automated tests arrive in a “big bang” rollout, many are not valid or passing because of the drift
5. Specialized team/consultants roll off, remaining developers don’t know and struggle to maintain the automated suite
6. More and more tests drift and fail, failures ignored, test suite rots
7. Suite abandoned, investment lost

WHAT’S WRONG HERE?

Where is this story at odds with your understanding of agile principles, professional effectiveness, common sense?

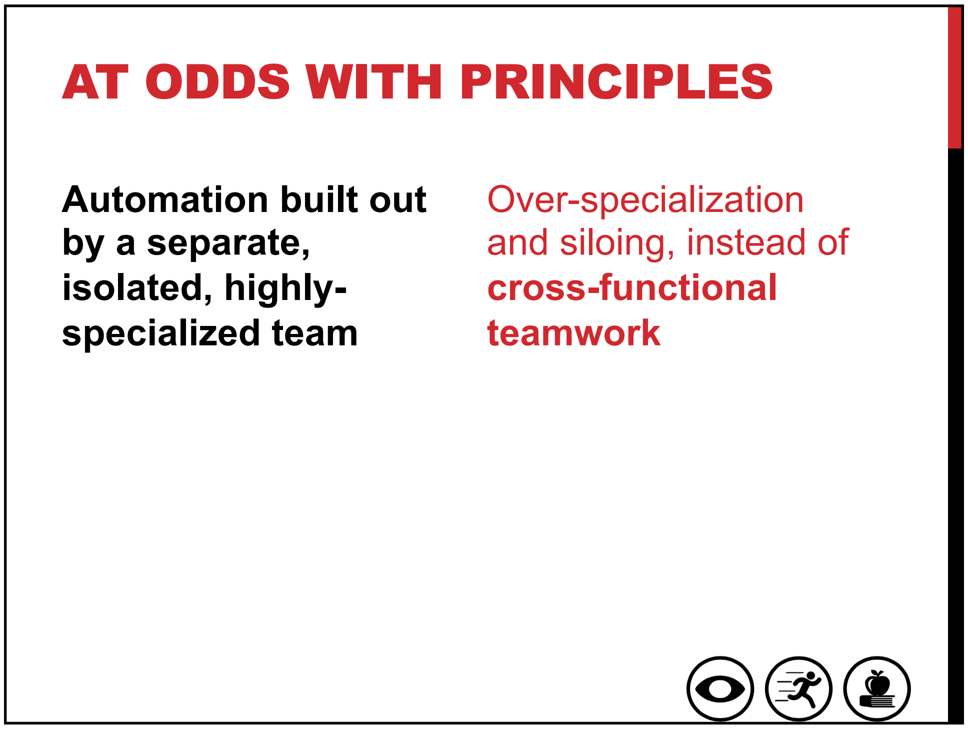




AT ODDS WITH PRINCIPLES

Quality suffers while waiting for specialized tools / workers / skills / training / decisions / environments

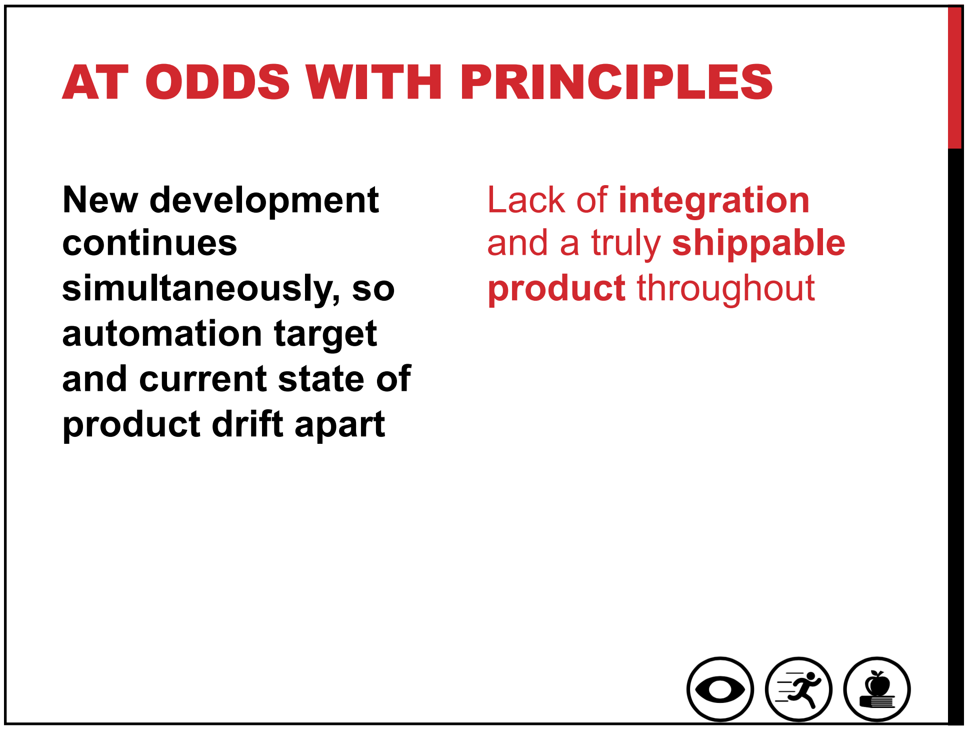
Lack of **self-organization** – to figure out *how*, to solve their own problems



AT ODDS WITH PRINCIPLES

Automation built out by a separate, isolated, highly-specialized team

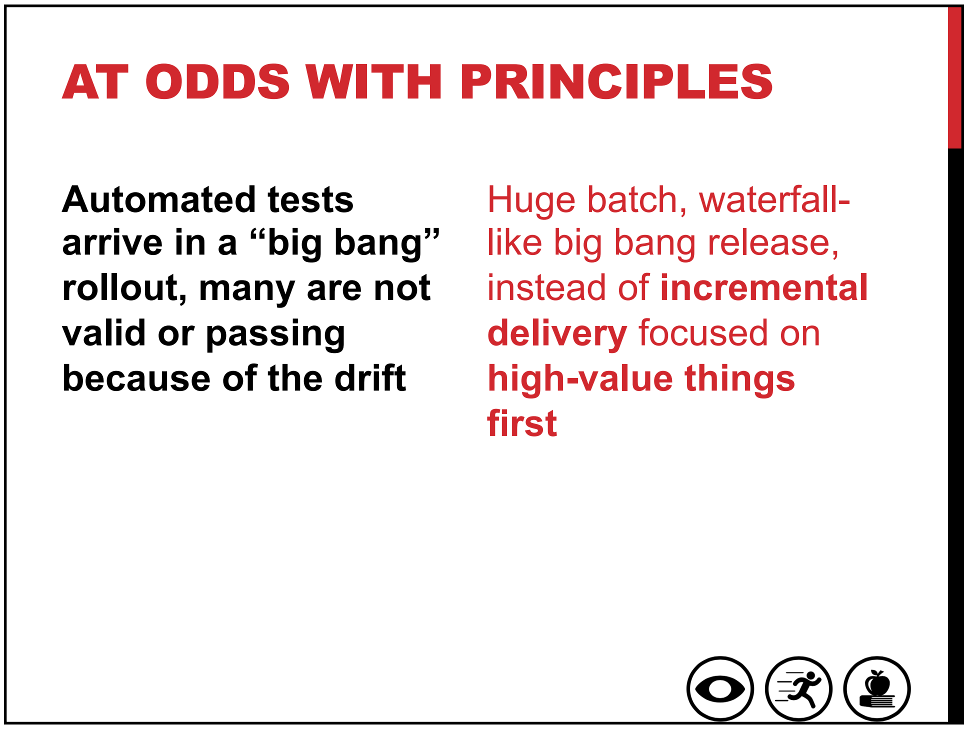
Over-specialization and siloing, instead of **cross-functional teamwork**



AT ODDS WITH PRINCIPLES

New development continues simultaneously, so automation target and current state of product drift apart

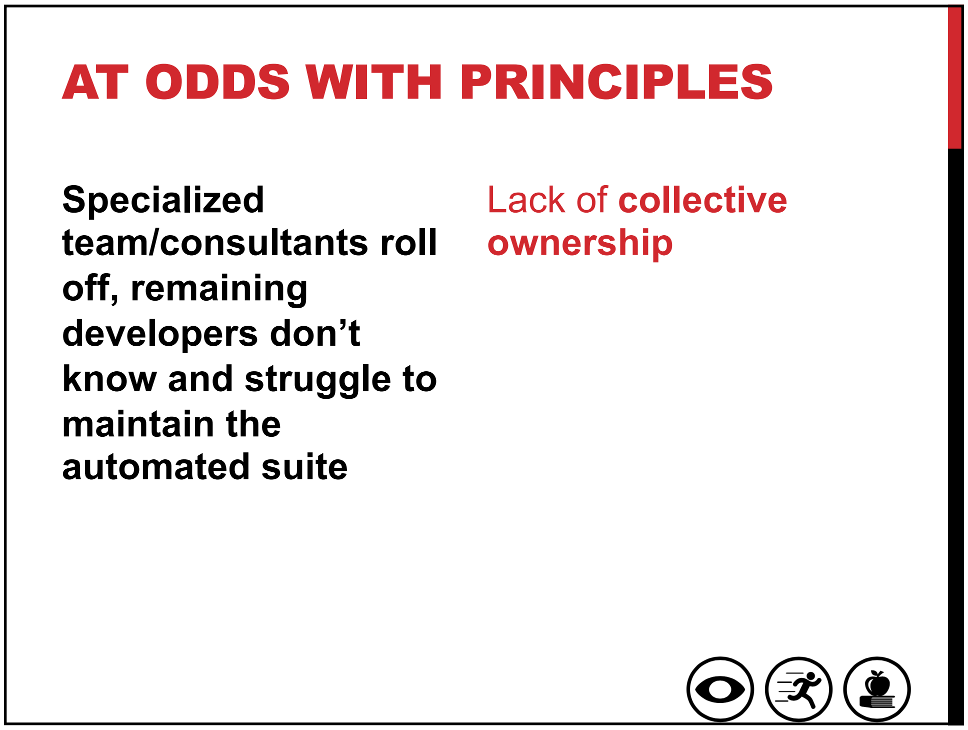
Lack of **integration** and a truly **shippable product** throughout



AT ODDS WITH PRINCIPLES

Automated tests arrive in a "big bang" rollout, many are not valid or passing because of the drift

Huge batch, waterfall-like big bang release, instead of **incremental delivery** focused on **high-value things first**



AT ODDS WITH PRINCIPLES

Specialized team/consultants roll off, remaining developers don't know and struggle to maintain the automated suite

Lack of **collective ownership**

AT ODDS WITH PRINCIPLES

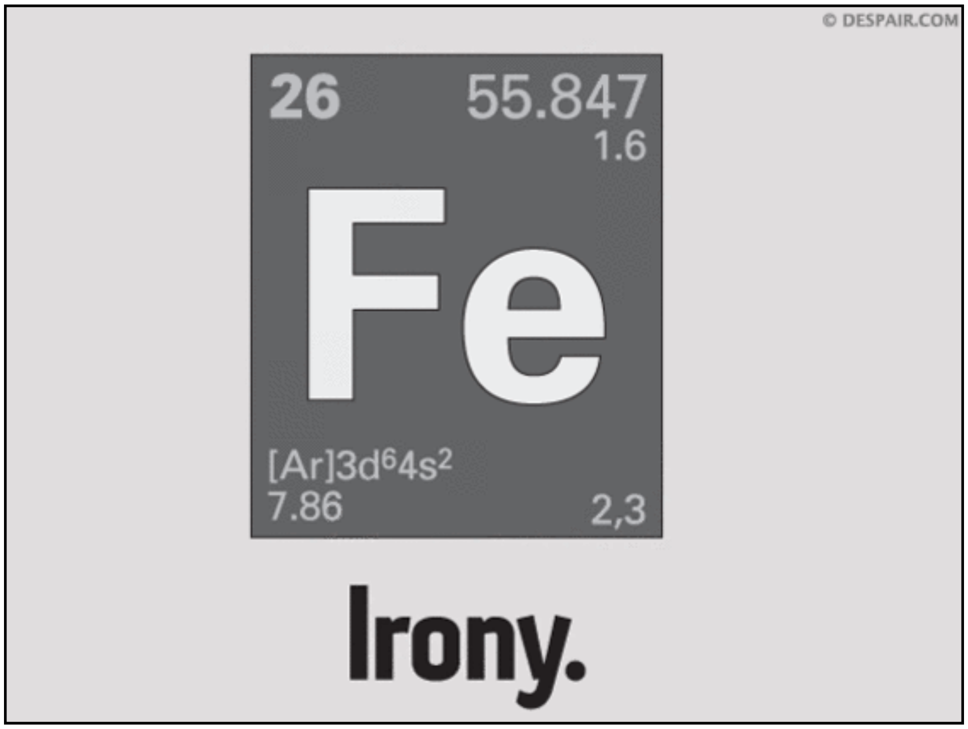
More and more tests drift and fail, failures ignored, test suite rots

Lacking consistent dedication and **ownership of quality**

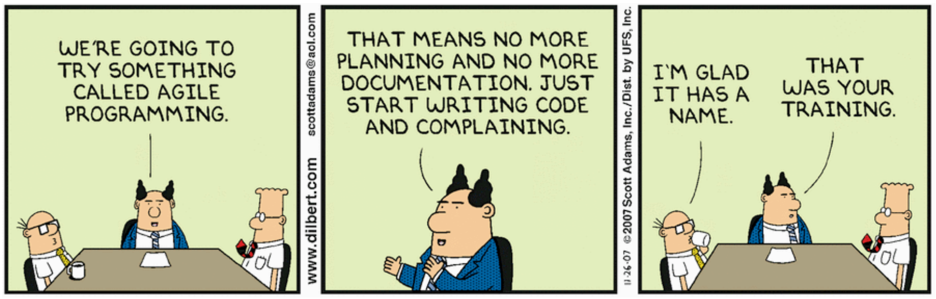
AT ODDS WITH PRINCIPLES

Suite abandoned, investment lost

Disposable commodities, instead of **vision** with focus on **value** (including high ROI, low TCO)

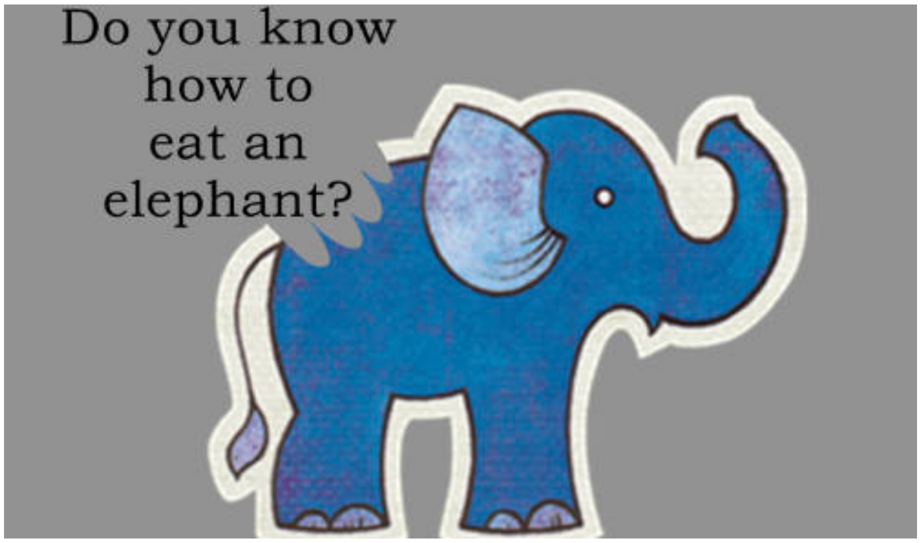


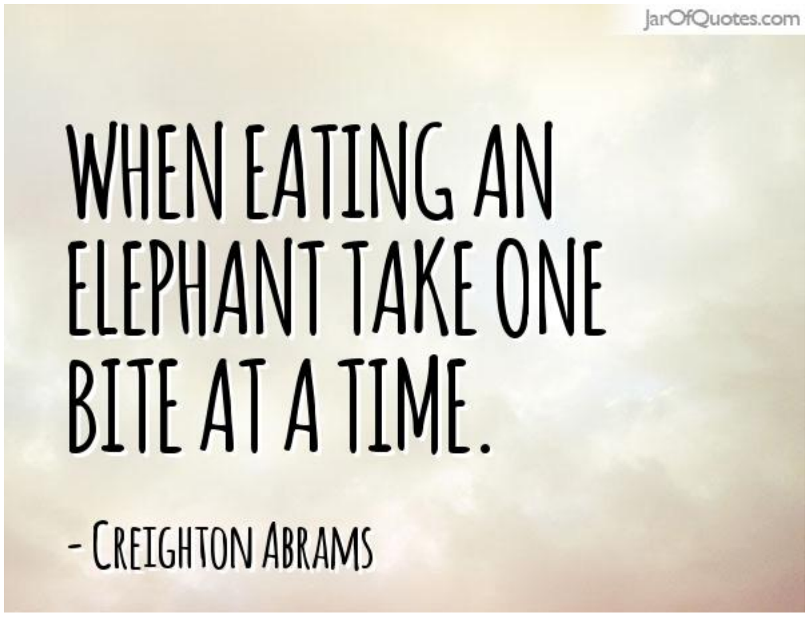
BUT WE WANT THE AGILE STUFFS


WE'RE GOING TO TRY SOMETHING CALLED AGILE PROGRAMMING.
THAT MEANS NO MORE PLANNING AND NO MORE DOCUMENTATION. JUST START WRITING CODE AND COMPLAINING.
I'M GLAD IT HAS A NAME.
THAT WAS YOUR TRAINING.
www.dilbert.com scottadams@aol.com
11-26-07 ©2007 Scott Adams, Inc./Dist. by UFS, Inc.

SO, OUR APPROACHES AND SOLUTIONS SHOULD…

- Promote self-organization – teams figuring out *how* to do their work and solve their own problems
- Promote cross-functional teamwork, avoiding over-specialization and siloing
- Integrate automated tests into the development workflow immediately, and continually have a shippable product
- Work in small batches, incrementally deliver, avoiding big bang drops
- Promote collective ownership, cross-training, and information sharing
- Reflect dedication to and ownership of quality
- Build long-term investments per a vision, ROI, low-TCO




JarOfQuotes.com
WHEN EATING AN ELEPHANT TAKE ONE BITE AT A TIME.
- CREIGHTON ABRAMS

SO LET'S TALK SOLUTIONS

OR: How I learned to stop worrying and love the requisite "agenda slide"

- People and Attitude: Everyone Owns Quality
- Remembering the Primary Goal: Done, High-Quality, Potentially Shippable Product
- Start Early
- Build Quality in to New Development, Every Sprint
- Attack Existing Debt Incrementally
- Keep it Healthy

GARY PEDRETTI

- Over 20 years in the software industry with highly cross-functional experience – DBA, Developer, BA, Tester (with automation experience), Application Architect
- Currently Trainer, Coach, Consultant, Owner at Sodoto Solutions
- SODOTO = See One, Do One, Teach One
- Scrum: Development Team member, Scrum Master, Coach, Professional Scrum Trainer for Scrum.org
- http://blog.GaryPedretti.com/
- http://www.linkedin.com/in/garypedretti
- Twitter: @GaryPedretti

PEOPLE AND ATTITUDE: EVERYONE OWNS QUALITY

Sodoto Solutions

See One Do One Teach One

PEOPLE AND ATTITUDE: EVERYONE OWNS QUALITY

But what is quality?

American Society for Quality:

- **"A combination of quantitative and qualitative perspectives…**
- **…for which each person has his or her own definition…"**

QUALITY

- **Quality != Testing**
- **Verification – building it right**
- **Validation – building the right thing**
  - Value…"Quality in a product or service is not what the supplier puts in. It is what the customer gets out and is willing to pay for." – Peter Drucker
  - Fitness for Purpose
  - What the customer would love but has not yet thought about (Steve Jobs, Kano's "Attractive Quality")
- **The result of care (Robert Pirsig, Software Craftsmanship)**
- **Quality NOT just lack of bugs…"Number of defects per million opportunities" (Six Sigma)**

so…

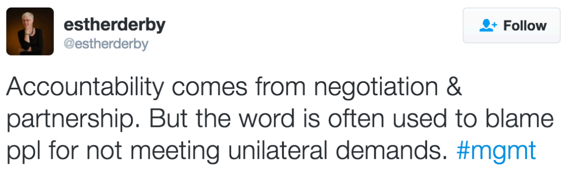
- …everyone tests?
- …no specialists then?
- Ownership…

Once we…

- Agree on a tool
- Get a performance environment
- Have some experts (catch up and) automate the regression test suite
- Get some training

OWNERSHIP & ACCOUNTABILITY

- No excuses, no “someone else’s problem”, no “when we have X, then…”, no magical thinking
- But more importantly: autonomy, teamwork, and empowerment to solve one’s own problems


estherderby
@estherderby
Follow

Accountability comes from negotiation & partnership. But the word is often used to blame ppl for not meeting unilateral demands. #mgmt

THE FIRST LAW OF TEST AUTOMATION

is…

Lack of automation

!=

Lack of accountability for quality

OWNERSHIP, ACCOUNTABILITY

- What would happen if everyone had to test?
- …and running the regression test suite took 5 days?

- Automation = Code?
- Who's good at writing code?

REVIEW – PEOPLE AND ATTITUDE: EVERYONE OWNS QUALITY

- Quality != Testing – quality has many aspects and requires many specialties

- Ownership = no magical thinking…but also autonomy and empowerment

- Lack of automation != lack of accountability for quality

REMEMBER THE PRIMARY GOAL: DONE, HIGH-QUALITY, POTENTIALLY SHIPPABLE PRODUCT

Sodoto Solutions

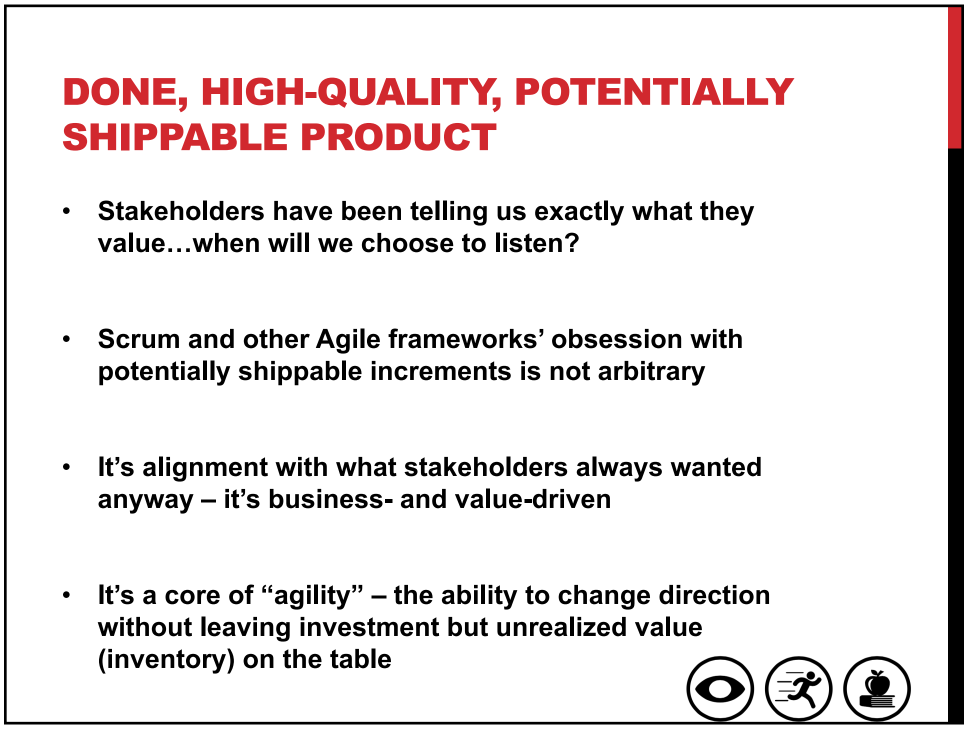
See One Do One Teach One

REMEMBER THE PRIMARY GOAL

Have you ever said something like…

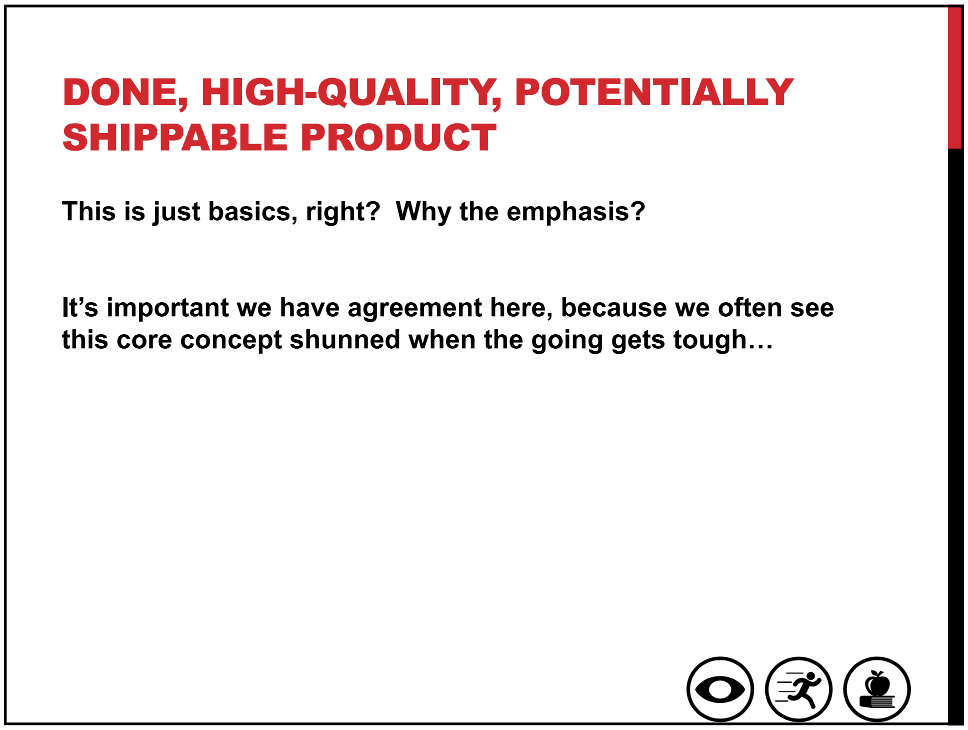
- It's done but it's not tested
- It's code complete
- It's done but not performance tested
- It's done…but DON'T ship it yet
- It's 80% done
- It's done but not Done Done

- If so, what was the next question you heard?



DONE, HIGH-QUALITY, POTENTIALLY SHIPPABLE PRODUCT

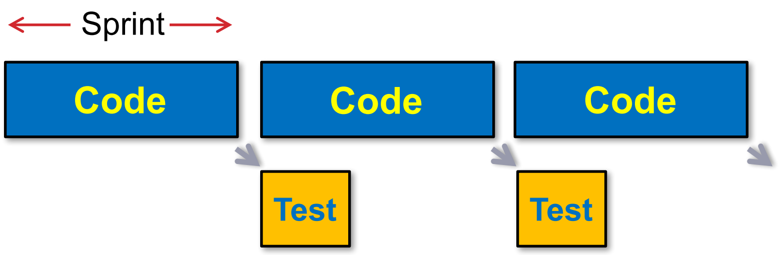
- Stakeholders have been telling us exactly what they value…when will we choose to listen?
- Scrum and other Agile frameworks' obsession with potentially shippable increments is not arbitrary
- It's alignment with what stakeholders always wanted anyway – it's business- and value-driven
- It's a core of "agility" – the ability to change direction without leaving investment but unrealized value (inventory) on the table

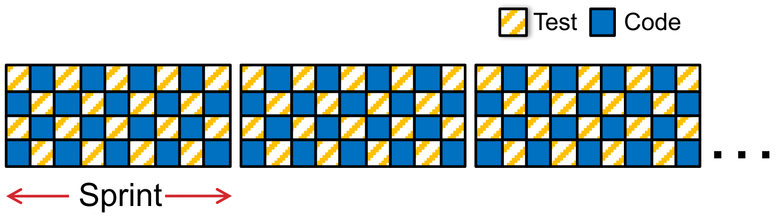


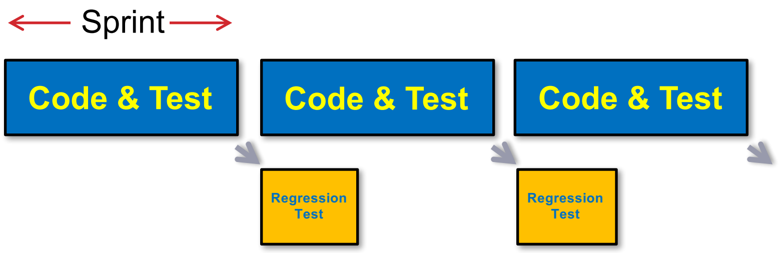
DONE, HIGH-QUALITY, POTENTIALLY SHIPPABLE PRODUCT

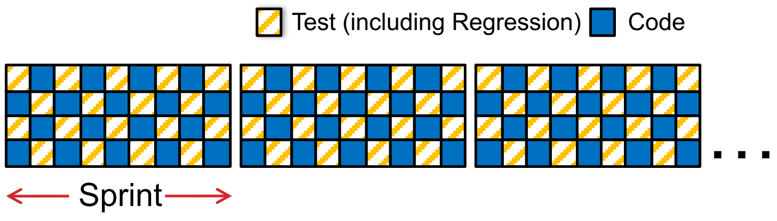
This is just basics, right? Why the emphasis?

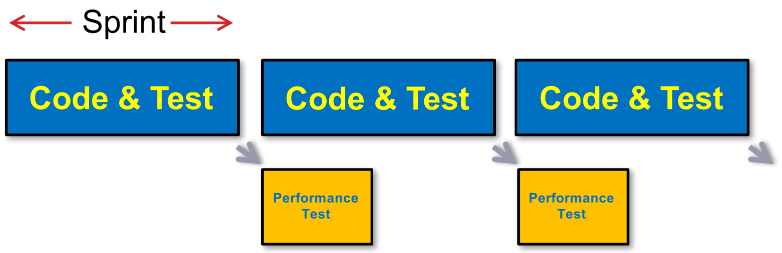
It's important we have agreement here, because we often see this core concept shunned when the going gets tough…

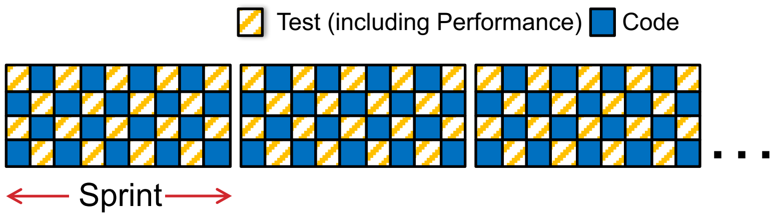

Sprint
Code
Code
Code
Test
Test

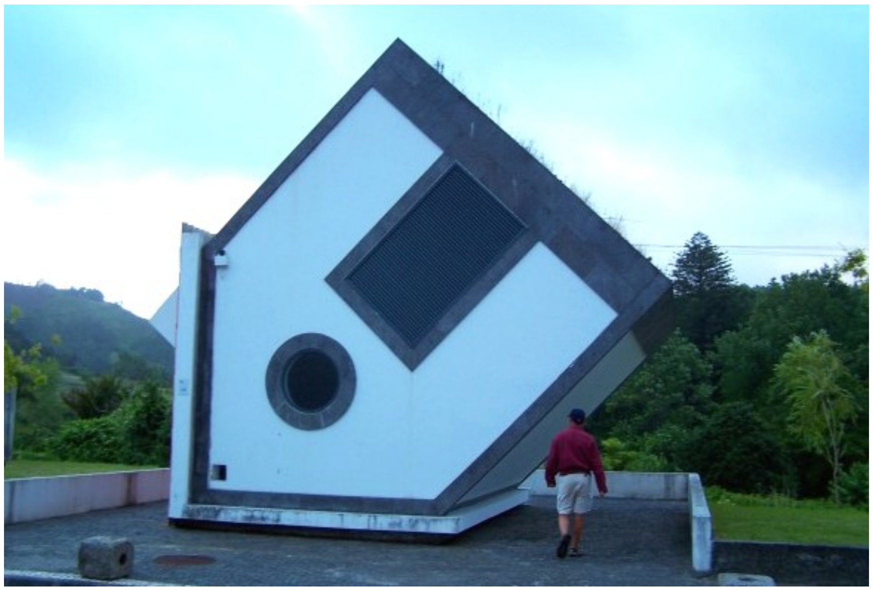

Test
Code
Sprint
. . .


Sprint
Code & Test
Code & Test
Code & Test
Regression Test
Regression Test


Test (including Regression)
Code
Sprint
. . .


Sprint
Code & Test
Code & Test
Code & Test
Performance Test
Performance Test


Test (including Performance)
Code
Sprint
. . .



TOUGH PROBLEMS

This may be hard –

we might not have this right now,

it might be a long road to get there from here

(more on this in a bit)

– but that still doesn't change the ideal, and shouldn't stop us from listening to and moving towards what stakeholders actually want

REVIEW – THE PRIMARY GOAL

- Stakeholders – not Agile zealots in white robes – have always wanted shippable software, and only shippable software
- Yes, of course shippable software happens by including regression testing, performance testing, security testing, etc. – so if you're delivering incrementally, we're talking about doing this in EVERY cycle
- Getting there might be hard, but practical difficulty doesn't change the ideals, the desires of the stakeholders, or professional responsibility to deliver what stakeholders want

START EARLY

Sodoto Solutions

See One Do One Teach One

START EARLY

Avoid the shunning of responsibility and accountability – avoid the magical thinking

If test automation is good, then ”Just Do It”

START EARLY

- OK, but what about our Test Automation Team?
- What about our current lack of skills in this area?
- Especially if you put these two together…cross-functional teams with people willing to be generalizing specialists, cross-training, etc…is a big part of your answer
- Regardless, you’re still responsible for solving your own problems, so if test automation is good, then…

START EARLY

- **OK, but what about lack of environments, toolsets, build server integration, etc.?**
- **What if you started performance testing without a performance testing environment? Amateur hour, right?**
  - A trend is a trend is a trend…even on your local machine
  - It doesn't mean we won't **eventually** test in Mega Performance Environment 5000™

THE SECOND LAW OF TEST AUTOMATION

is…

Automation

!=

Use of Specialized Tools

START EARLY

- **OK, but what about lack of environments, toolsets, build server integration, etc.?**
- **Low-hanging fruit rarely require any "automated testing tools" (Selenium, CodedUI, QTP, Cucumber, Fit, etc.)**
  - DB scripts to reproduce manual test data setup
  - Scripts to sample and scrub data from production
  - Blasting a server with hundreds of headless agents…who do we have on our team again? Building agents to hit REST services daily?...

REVIEW – START EARLY

- **Whatever specialized knowledge you have in this area, putting that in the context of cross-functional teams will result in cross-training**
- **There's no need to wait for an environment, toolset, etc. – a trend is a trend is a trend, regardless of the environment**
- **People often confuse automation with the use of specialized test automation tools, but there is low-hanging fruit that has nothing to do with those**

BUILD QUALITY IN TO NEW DEVELOPMENT, EVERY SPRINT

Sodoto Solutions

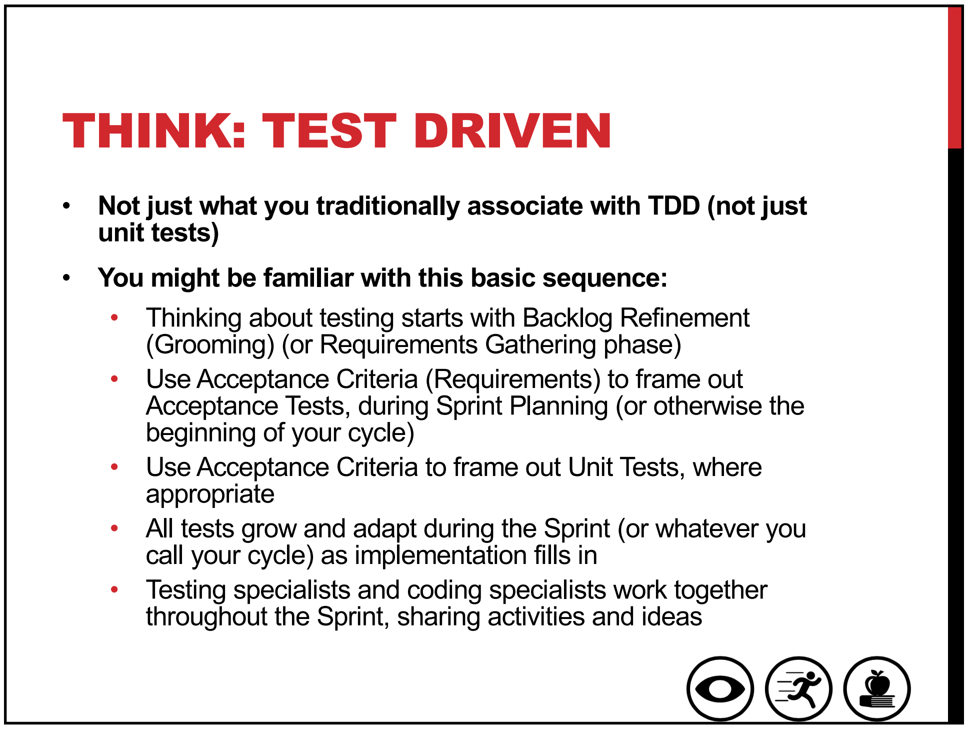
See One Do One Teach One

BUILD QUALITY IN TO NEW DEVELOPMENT, EVERY SPRINT

We BUILD quality in!!!

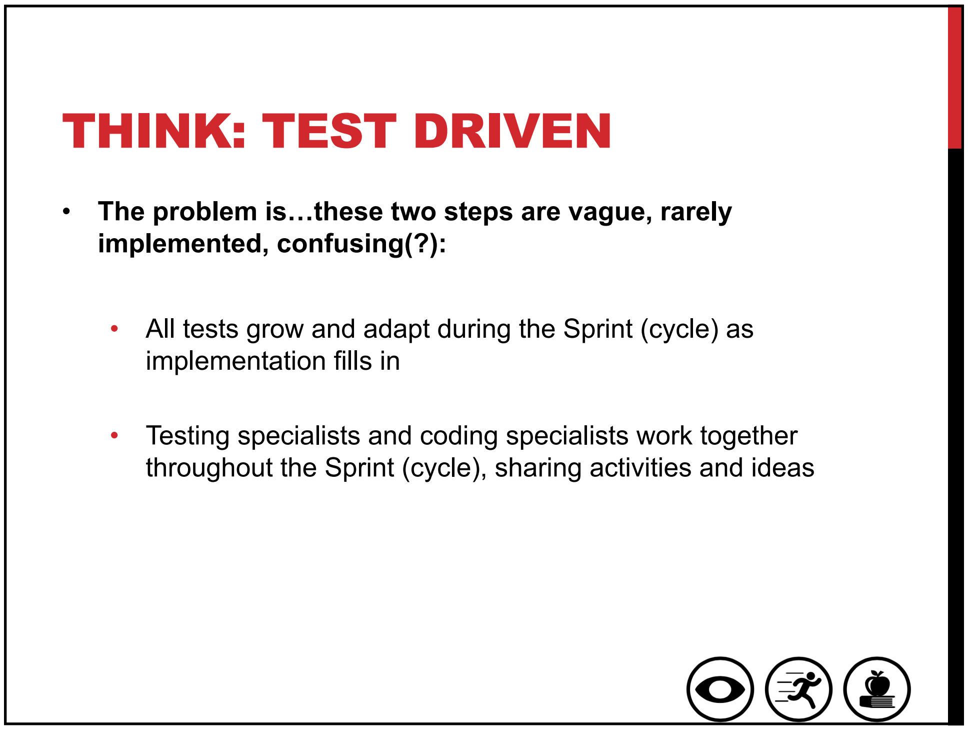
Quality is JOB #1 !!!!

But what are some actual, practical mechanisms to do this?



THINK: TEST DRIVEN

- **Not just what you traditionally associate with TDD (not just unit tests)**
- **You might be familiar with this basic sequence:**
  - Thinking about testing starts with Backlog Refinement (Grooming) (or Requirements Gathering phase)
  - Use Acceptance Criteria (Requirements) to frame out Acceptance Tests, during Sprint Planning (or otherwise the beginning of your cycle)
  - Use Acceptance Criteria to frame out Unit Tests, where appropriate
  - All tests grow and adapt during the Sprint (or whatever you call your cycle) as implementation fills in
  - Testing specialists and coding specialists work together throughout the Sprint, sharing activities and ideas



THINK: TEST DRIVEN

- **The problem is…these two steps are vague, rarely implemented, confusing(?):**
  - All tests grow and adapt during the Sprint (cycle) as implementation fills in
  - Testing specialists and coding specialists work together throughout the Sprint (cycle), sharing activities and ideas

TESTS GROW AND ADAPT DURING THE SPRINT

- **The BDD world went through a big debate about Declarative vs. Imperative style tests**
- **Declarative (informative) example:**
  - Scenario: Submit the form with correct data

  Given I am on the personal data form

  When I submit the form with correct data

  Then I should see the correct message

TESTS GROW AND ADAPT DURING THE SPRINT

- **Imperative (communicative) example:**
  - Scenario: Submit the form with correct data (imperative)

  Given I am on the personal data page

  When I set "What is your name?" to "Alister"

  And I set "What is your story?" to "I like ruby"

  And I set "What testing tool do you like?" to "Watir"

  And I click "Submit"

  Then I should see the message "Thanks! Your response has been recorded."

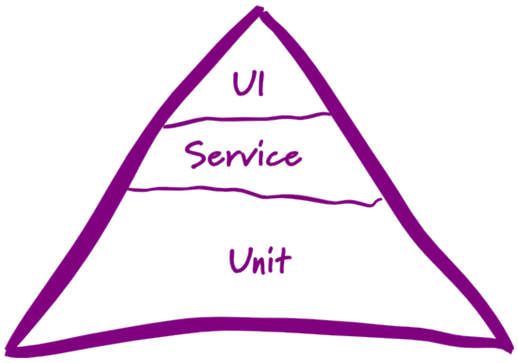
TESTS GROW AND ADAPT DURING THE SPRINT

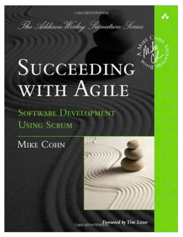
- **Like most endless debates, the question is usually wrong**
  - It's not, "**Which** one rules them all?"
  - It's, "**When** should I use which one?" – it's about **context**
- **Grow and adapt during the Sprint**
  - The tests will initially be higher-level and more Declarative
  - As implementation is filled in, those same tests will get lower-level, more detailed, and more Imperative

TESTING SPECIALISTS AND CODING SPECIALISTS WORK TOGETHER THROUGHOUT THE SPRINT

- **Coding specialists are often sharing their output with testers (e.g., via pairing up, via a deploy to a QA environment)**
- **But are your testing specialists sharing their output and activities with coders?**
- **The Declarative -> Imperative transformation will involve a lot of communication between these specialties, and between the entire team**

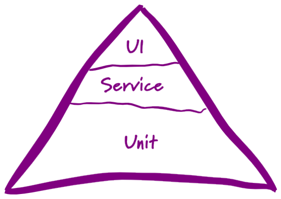
THE TEST AUTOMATION PYRAMID





Ideas from Mike Cohn's *Succeeding With Agile*, graphic from http://martinfowler.com/bliki/TestPyramid.html

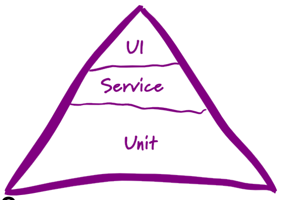
THAT'S NICE, BUT…



We have functional / manual / non-coding QA specialists on our team…

- **The UI tests they write re-test stuff that's already been tested in the unit and service/integration layers**
- **How would they know what's covered in the unit tests and the automated service/integration tests?**

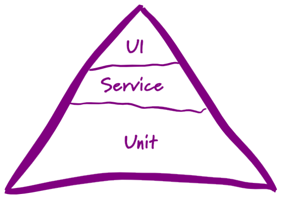
THAT'S NICE, BUT…



How will non-coding testing specialists know?

- Well, put them on a team with coders and have them TALK TO EACH OTHER (gasp!)
- Coders should write meaningful, descriptive, and accurate test names
- These can then be seen and understood outside of the code, e.g. in a build report
    - DivideByZeroShouldThrowCustomException
    - DivideNegativeByNegativeShouldEqualANegative
    - Oooops, I mean DivideNegativeByNegativeShouldEqualAPostive

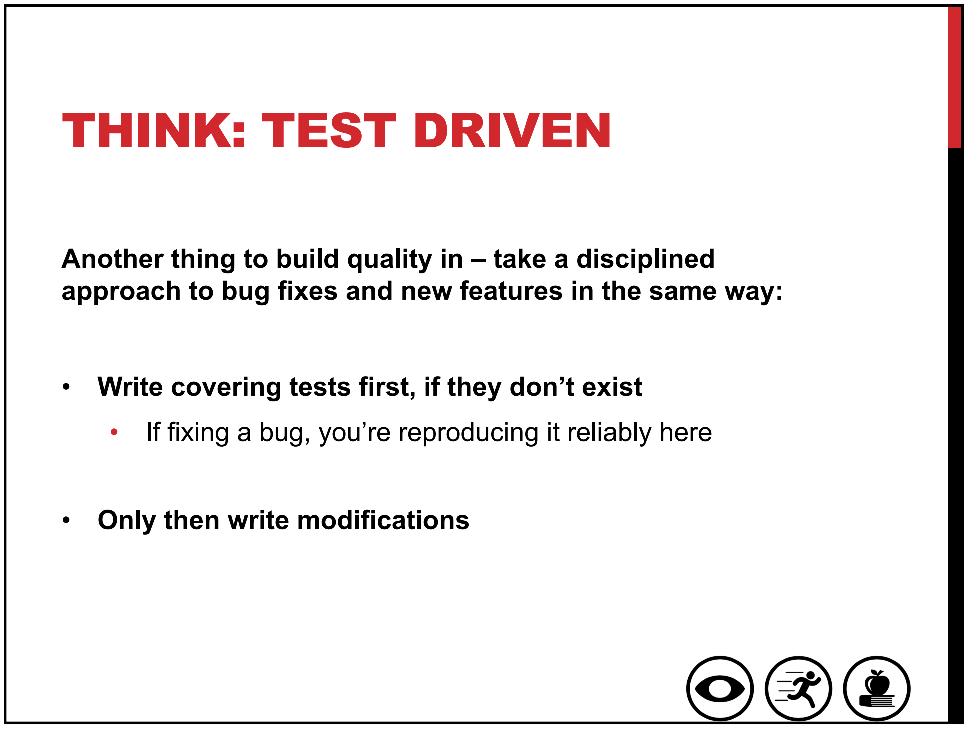
THAT'S NICE, BUT…



This is another mechanism for "Testers and Coders work together continuously throughout the Sprint (cycle)"

This one is a little less personal than a face-to-face conversation

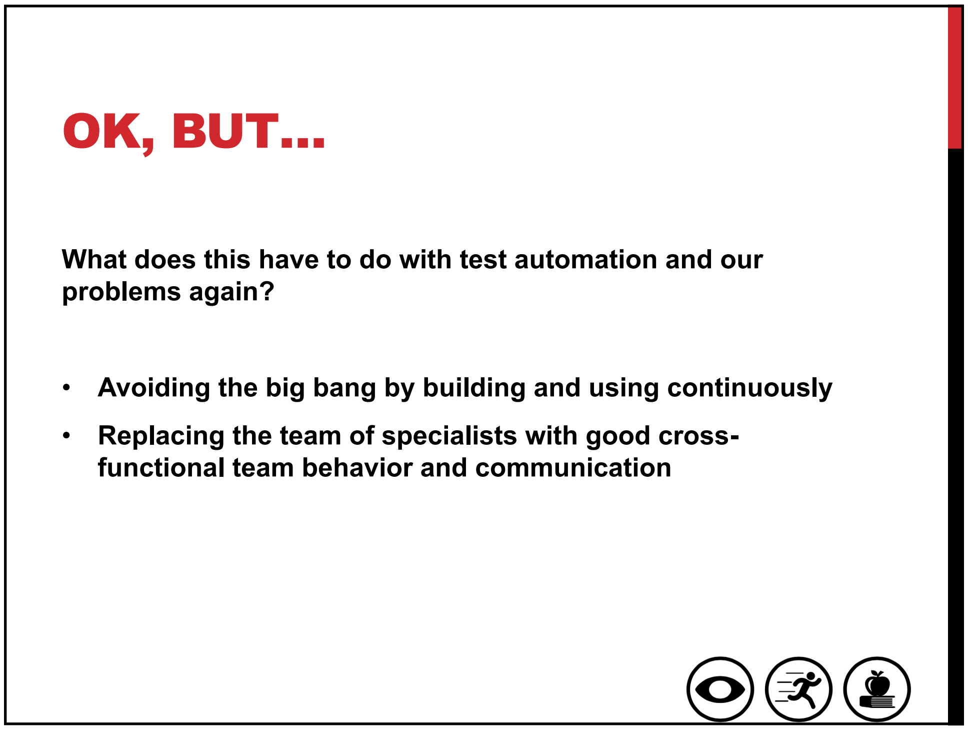
But it helps you realize all of your artifacts are transparent communication, maybe just at different levels of fidelity



THINK: TEST DRIVEN

Another thing to build quality in – take a disciplined approach to bug fixes and new features in the same way:

- **Write covering tests first, if they don't exist**
  - If fixing a bug, you're reproducing it reliably here
- **Only then write modifications**



OK, BUT…

What does this have to do with test automation and our problems again?

- **Avoiding the big bang by building and using continuously**
- **Replacing the team of specialists with good cross-functional team behavior and communication**

AUTOMATION INVESTMENTS

- Automation, like any activity you expect to derive value from, is an investment
- Specifically, investing time to enhance quality

RECORD / PLAYBACK: A JOKE, RIGHT?

"Record-playback tools are almost always a bad idea for any kind of automation, since they resist changeability and obstruct useful abstractions. They are only worth having as a tool to generate fragments of scripts which you can then edit as a proper programming language, in the manner of Twist or Emacs."

– Martin Fowler

RECORD / PLAYBACK: A JOKE, RIGHT?

- Results in brittle tests
- Results in ugly, unmaintainable code
- AKA, Amateur Hour

- But, investment is incredibly low

HAND-CODED AUTOMATED TESTS: TOTALLY PRO

- Helps you introduce abstractions to help with brittleness
- Complete control – whatever you want to do, parameterize, etc.
- Maintainable code (if you choose to write it that way)
- But, investment is (usually, incredibly) high

RECORD/PLAYBACK VS. HAND CODING

- **Let's have a debate!**
- **NO**
- **Again, it's not WHICH, it's WHEN. What's the Context?**
- **Wouldn't it be great if we could keep the investment low when things are volatile, and increase the investment as things stabilize?**

RECORD/PLAYBACK VS. HAND CODING

- **Record/Playback is low investment early in the Sprint, but it still often has value:**
  - Understanding how to really test this feature
  - Discovering which pieces / controls are most brittle and volatile
  - Still allows for quick runs – and often eliminates the input of rote data
- **Regardless, because this is low investment, you should have no problem throwing it away!**
- **But sure, if you're invested in test automation in general, you'll probably want your long-term investment to be less brittle, more flexible, more maintainable – so you'll move towards hand-coding**

REVIEW – BUILD QUALITY IN TO NEW DEVELOPMENT, EVERY SPRINT

- **Test drive everything – it's not just what you traditionally associate with TDD**
  - If fixing a bug, you're reproducing it reliably here
- **Grow and adapt tests during the Sprint – Declarative to Imperative**
- **Leverage the Testing Pyramid test type ratios, including communication between coders and testers to avoid duplicate effort**
- **Test automation is an investment**
- **Grow and adapt automation during the Sprint**
  - Record/playback automagic to hand coded
  - An increasingly abstracted / robust hand coded test

ATTACK THE DEBT INCREMENTALLY

Sodoto Solutions

See One Do One Teach One

ATTACK THE DEBT INCREMENTALLY

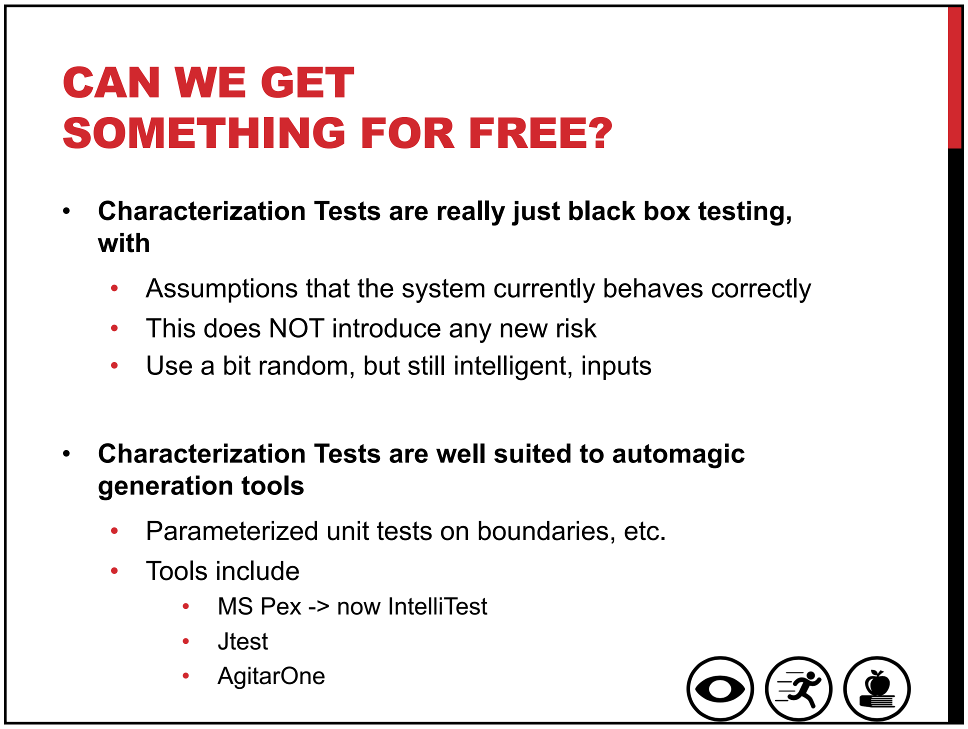
BUT WHICH PARTS FIRST?

What are some dimensions we can use to figure this out?

- **Look at usage stats**
  - Modern tools for this are wonderful!
- **By (alleged) value – (historical) Backlog order**
- **Execution time for the manual version of the test**
- **Use what you already know about risk-based testing**

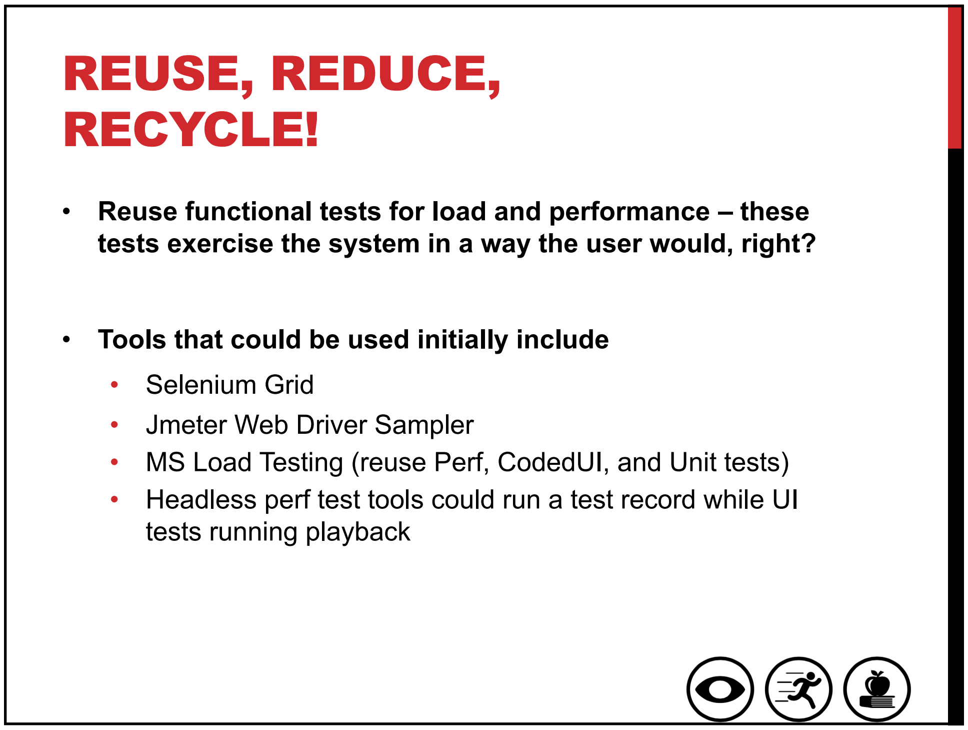
CAN WE GET SOMETHING FOR FREE?

- **Michael Feathers described a concept called Characterization Tests, also called Behavioral Regression Testing**
  - Assume the existing, unknown system is currently doing the right thing
  - Feed it inputs
  - Observe the results
  - Put those inputs into a test, and the results into the assertions of that same test



CAN WE GET SOMETHING FOR FREE?

- **Characterization Tests are really just black box testing, with**
  - Assumptions that the system currently behaves correctly
  - This does NOT introduce any new risk
  - Use a bit random, but still intelligent, inputs
- **Characterization Tests are well suited to automagic generation tools**
  - Parameterized unit tests on boundaries, etc.
  - Tools include
    - MS Pex -> now IntelliTest
    - Jtest
    - AgitarOne



REUSE, REDUCE, RECYCLE!

- **Reuse functional tests for load and performance – these tests exercise the system in a way the user would, right?**
- **Tools that could be used initially include**
  - Selenium Grid
  - Jmeter Web Driver Sampler
  - MS Load Testing (reuse Perf, CodedUI, and Unit tests)
  - Headless perf test tools could run a test record while UI tests running playback

AUTOMATE ALL THE THINGS!

- So, when will we be completely done with this automation stuff?
- The goal might not be total automation – think risk-based testing, think about your investments
- It's not an all-or-nothing proposition – are you getting the most value for what *is* automated?

REVIEW – ATTACK THE DEBT INCREMENTALLY

- Figure out what's most important to attack first – prioritize
- Consider Characterization tests, which can easily be auto-generated
- Reuse your tests wherever possible
- Stop when it's no longer valuable

KEEP IT HEALTHY

Sodoto Solutions

See One Do One Teach One

KEEP IT HEALTHY

It’s just one little test failing, and we know why it fails…

It will succeed once Johnny implements feature X…

Or if you just run it again it will succeed…

Well, usually a third or fourth time will do it for sure!

NO TOLERANCE FOR FAILING TESTS

- **Failing tests, at whatever level, whatever stage of the pipeline, are showstoppers**
  - We often see this one ignored when people first start automating all of their tests
  - Everyone knows to fix unit tests that fail in the first stage of the build pipeline…
  - …But if a test fails in later stages, regardless of whether it's an "acceptance test," a "UI test," etc. – this is a showstopper for everyone too!

- **All hands on deck!**

NO TOLERANCE FOR FAILING TESTS

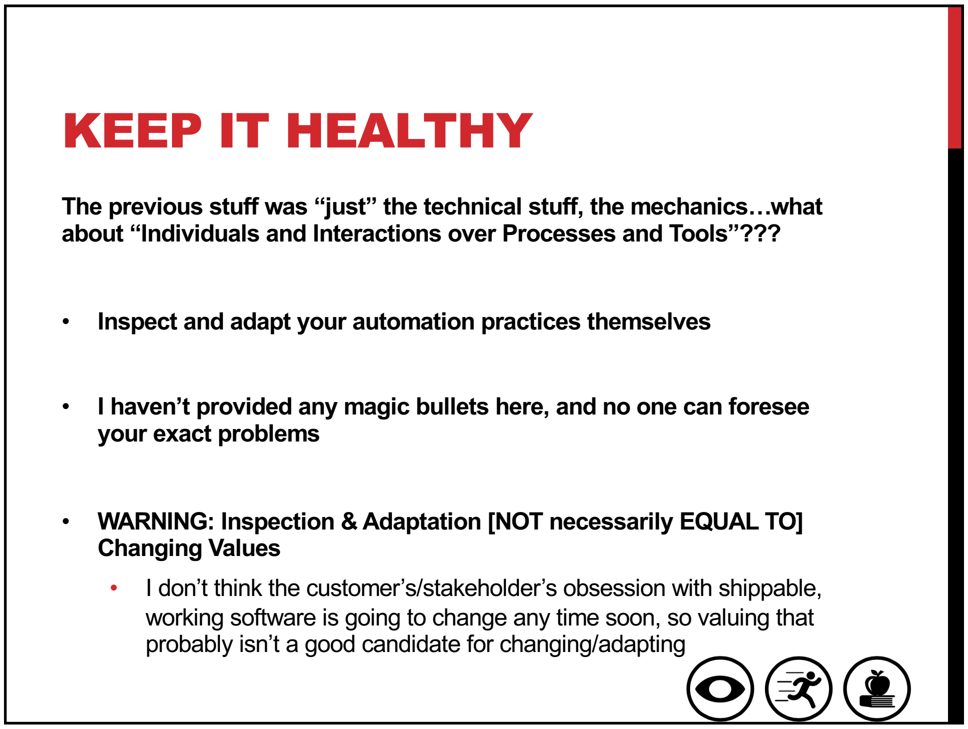
- **Tests are first class citizens**
  - Part of your codebase
  - Current culmination of communication / consensus with stakeholders and SMEs
  - Your requirements – in the form of executable specifications
  - Documentation

- **Avoid and eliminate ANYTHING that will cast doubts on your test results – NO ROT ALLOWED**

KEEP IT HEALTHY

But that's just the technical stuff, the mechanics…

WHO CAME UP WITH THIS ANALOGY ANYWAY?!
I DON'T KNOW BUT IT'S *@$! ING DISGUSTING.



KEEP IT HEALTHY

The previous stuff was "just" the technical stuff, the mechanics…what about "Individuals and Interactions over Processes and Tools"???

- **Inspect and adapt your automation practices themselves**
- **I haven't provided any magic bullets here, and no one can foresee your exact problems**
- **WARNING: Inspection & Adaptation [NOT necessarily EQUAL TO] Changing Values**
  - I don't think the customer's/stakeholder's obsession with shippable, working software is going to change any time soon, so valuing that probably isn't a good candidate for changing/adapting

@GARYPEDRETTI

GARY@SODOTOSOLUTIONS.COM

THANK YOU!
QUESTIONS?

Sodoto Solutions

See One Do One Teach One

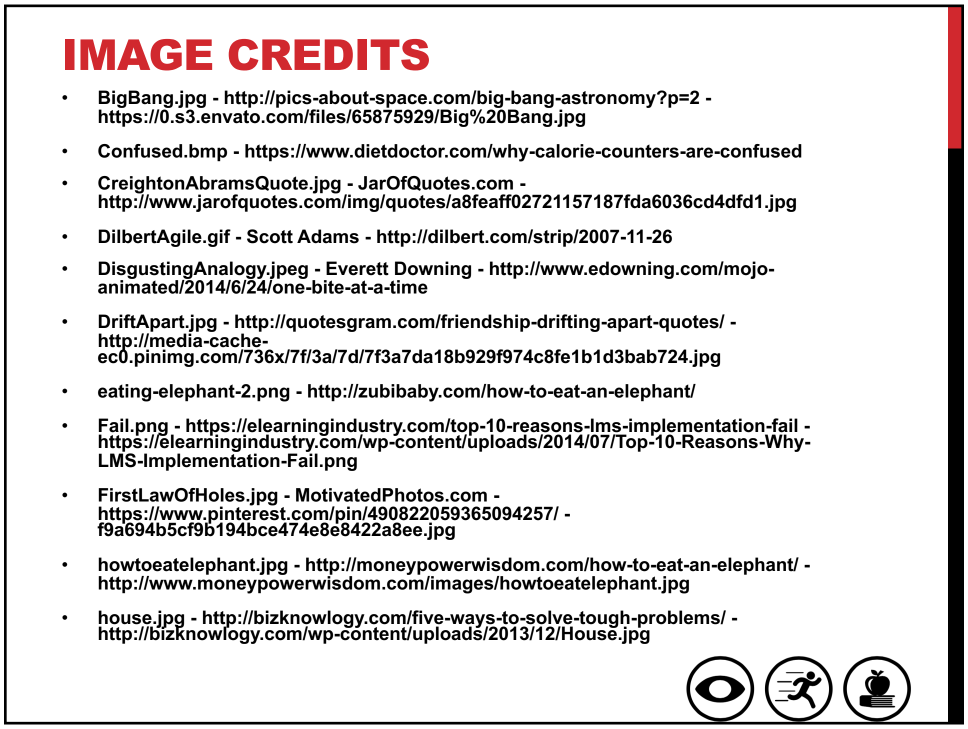


IMAGE CREDITS

- BigBang.jpg - http://pics-about-space.com/big-bang-astronomy?p=2 - https://0.s3.envato.com/files/65875929/Big%20Bang.jpg
- Confused.bmp - https://www.dietdoctor.com/why-calorie-counters-are-confused
- CreightonAbramsQuote.jpg - JarOfQuotes.com - http://www.jarofquotes.com/img/quotes/a8feaff02721157187fda6036cd4dfd1.jpg
- DilbertAgile.gif - Scott Adams - http://dilbert.com/strip/2007-11-26
- DisgustingAnalogy.jpeg - Everett Downing - http://www.edowning.com/mojo-animated/2014/6/24/one-bite-at-a-time
- DriftApart.jpg - http://quotesgram.com/friendship-drifting-apart-quotes/ - http://media-cache-ec0.pinimg.com/736x/7f/3a/7d/7f3a7da18b929f974c8fe1b1d3bab724.jpg
- eating-elephant-2.png - http://zubibaby.com/how-to-eat-an-elephant/
- Fail.png - https://elearningindustry.com/top-10-reasons-lms-implementation-fail - https://elearningindustry.com/wp-content/uploads/2014/07/Top-10-Reasons-Why-LMS-Implementation-Fail.png
- FirstLawOfHoles.jpg - MotivatedPhotos.com - https://www.pinterest.com/pin/490822059365094257/ - f9a694b5cf9b194bce474e8e8422a8ee.jpg
- howtoeatelephant.jpg - http://moneypowerwisdom.com/how-to-eat-an-elephant/ - http://www.moneypowerwisdom.com/images/howtoeatelephant.jpg
- house.jpg - http://bizknowlogy.com/five-ways-to-solve-tough-problems/ - http://bizknowlogy.com/wp-content/uploads/2013/12/House.jpg

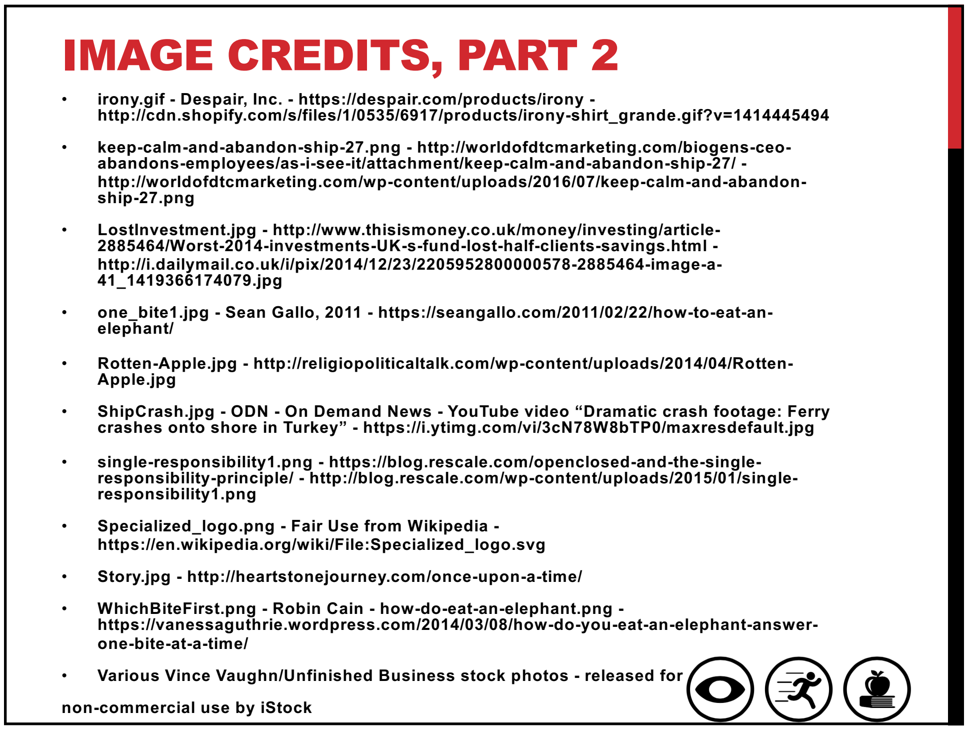


IMAGE CREDITS, PART 2

- irony.gif - Despair, Inc. - https://despair.com/products/irony - http://cdn.shopify.com/s/files/1/0535/6917/products/irony-shirt_grande.gif?v=1414445494
- keep-calm-and-abandon-ship-27.png - http://worldofdtcmarketing.com/biogens-ceo-abandons-employees/as-i-see-it/attachment/keep-calm-and-abandon-ship-27/ - http://worldofdtcmarketing.com/wp-content/uploads/2016/07/keep-calm-and-abandon-ship-27.png
- LostInvestment.jpg - http://www.thisismoney.co.uk/money/investing/article-2885464/Worst-2014-investments-UK-s-fund-lost-half-clients-savings.html - http://i.dailymail.co.uk/i/pix/2014/12/23/2205952800000578-2885464-image-a-41_1419366174079.jpg
- one_bite1.jpg - Sean Gallo, 2011 - https://seangallo.com/2011/02/22/how-to-eat-an-elephant/
- Rotten-Apple.jpg - http://religiopoliticaltalk.com/wp-content/uploads/2014/04/Rotten-Apple.jpg
- ShipCrash.jpg - ODN - On Demand News - YouTube video “Dramatic crash footage: Ferry crashes onto shore in Turkey” - https://i.ytimg.com/vi/3cN78W8bTP0/maxresdefault.jpg
- single-responsibility1.png - https://blog.rescale.com/openclosed-and-the-single-responsibility-principle/ - http://blog.rescale.com/wp-content/uploads/2015/01/single-responsibility1.png
- Specialized_logo.png - Fair Use from Wikipedia - https://en.wikipedia.org/wiki/File:Specialized_logo.svg
- Story.jpg - http://heartstonejourney.com/once-upon-a-time/
- WhichBiteFirst.png - Robin Cain - how-do-eat-an-elephant.png - https://vanessaguthrie.wordpress.com/2014/03/08/how-do-you-eat-an-elephant-answer-one-bite-at-a-time/
- Various Vince Vaughn/Unfinished Business stock photos - released for non-commercial use by iStock