Test Automation

Daniel Ståhl
daniel.stahl@ericsson.com
http://danielstahl.co/

LINKÖPING
UNIVERSITY

Who am I?

DANIEL STÅHL

Development
Architecture
Continuous practices

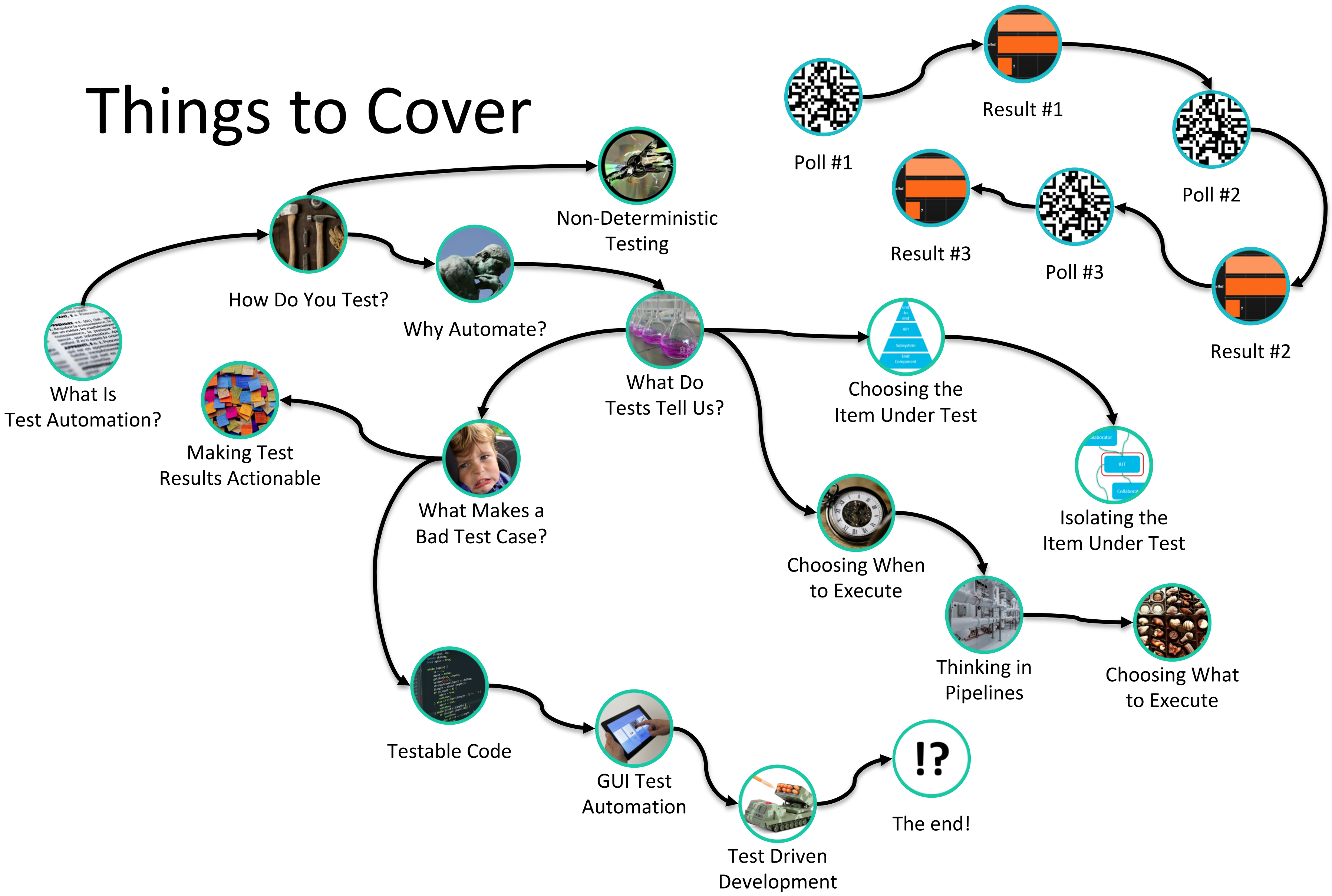
ERICSSON

Software production
Software delivery
Strategic studies
Group strategy

RESEARCH

PhD
Writing
Software Center
LiU

Things to Cover



What is Test Automation?

AUTOMATIC EXECUTION OF TEST SOFTWARE

Testing is interacting to provide feedback – not just finding bugs! Automated testing is automated interaction.

Or...
Execution of software separate from the system being tested (Item Under Test, IUT), for the purposes of evaluating that Item Under Test.

SPECIFIC TOOLS, FRAMEWORKS AND ENVIRONMENT

Apart from the automated test cases, you need tools for...

... scheduling
... deployment
... execution
... simulation
... data collection
... data analysis
... presentation
...

OUTCOME COMPARISON

Typically the outcome is compared to some expected result.

Either the test case succeeds in matching this result, or it fails.

KPIs may also be measured. Particularly relevant for non-functional requirements.

How Do You Test?

METHODOLOGY VARIES

The type of Item Under Test (IUT) matters!

... Is it a GUI?
... Is it an API?
... Is an embedded system?
... Is it a physical interface?
...

AT WHAT LEVEL AND MATURITY ARE YOU TESTING?

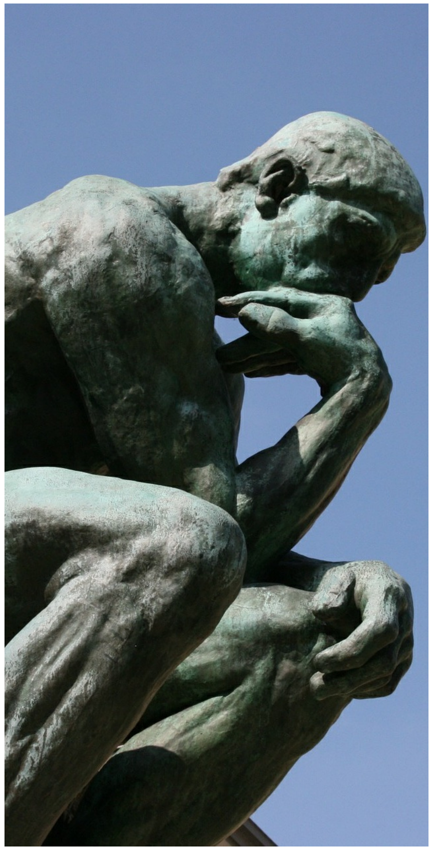
Are you testing the whole system, or a small piece of it?

Is the IUT an early prototype or a mature in-service system?

WHY ARE YOU TESTING?

What are you trying to achieve by testing the IUT?

Why Automate? 1 (3)



(HUMAN) TESTERS ARE EXPENSIVE

In software engineering, the main cost driver is (usually) the engineers' time.

(HUMAN) TESTERS ARE INCONSISTENT

Even with rules and checklists, humans are notoriously inconsistent.

We get bored.

We like to improvise.

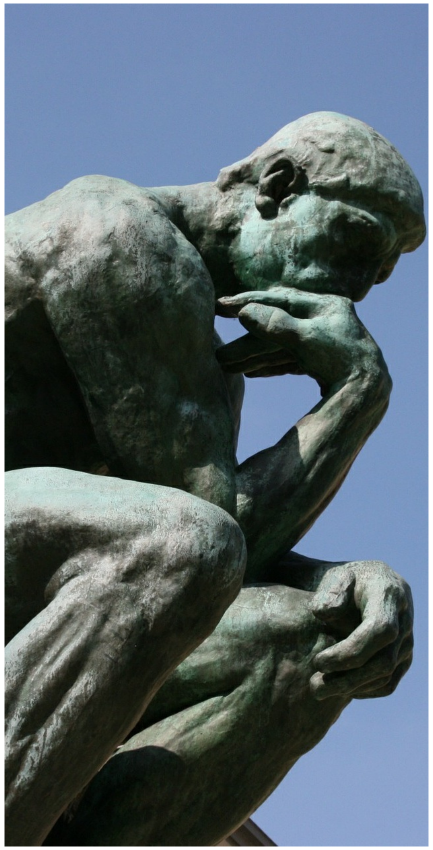
We need some creative freedom.

Whether that's good or bad depends on context!

(HUMAN) TESTERS ARE SLOW

Executing algorithms is not what we do best.

Why Automate? 2 (3)



(HUMAN) TESTERS HAVE NEEDS

Training, vacation, lunch breaks, sense of purpose... All kinds of annoying stuff.

(HUMAN) TESTERS HAVE BETTER THINGS TO DO

Humans still excel at certain types of testing. Let humans do what humans do best.

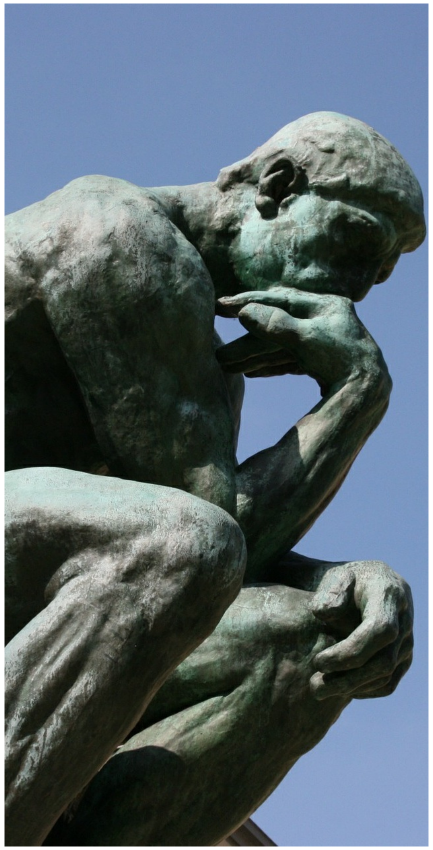
CONFIDENCE

Having a wall of green lights tell you everything works after you push a change is really nice...

...but is it a false sense of security?

”It ain’t what you don’t know that gets you into trouble. It’s what you know for sure that just ain’t so.”
- Mark Twain

Why Automate? 3 (3)



RAPID FEEDBACK

Automated tests can provide immediate developer feedback.

In unfamiliar code, they serve as guardrails.

Automated tests can catch integration errors early.

In larger systems with interdependencies, you can author tests to protect ”your” parts.

CONTRACT SATISFACTION

Automated acceptance tests can help clarify and safeguard contractual agreements.

Automated acceptance tests allow customers and/or stakeholders to track progress in real time.

BECAUSE IT MAKES YOU LOOK GOOD

Automated tests produce lots of data.

Data can be turned into fancy graphs and tables.

Fancy graphs and tables help you look like you know what you’re doing.

What Do Tests Tell Us?

WHAT DOES A TEST CASE RESULT MEAN?

Does success mean the IUT is working?

Does failure mean it’s not working?

Automated test case execution is a great way of protecting against predictable failure modes.

NON-TEST CASE BASED APPROACHES

Automated fault injection.

Monitoring system parameters to detect anomalies or measuring quality characteristics.

Applying machine learning to predict failure modes, then monitoring the leading indicators.

…

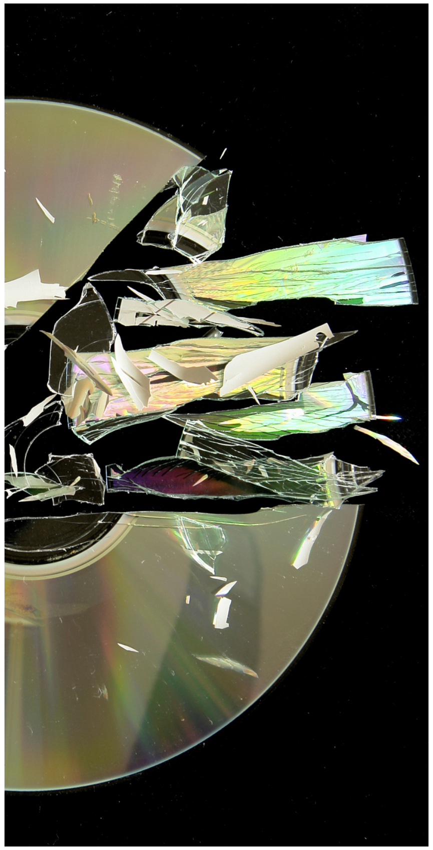
TEST RESULTS AS DECISION BASIS

Test results (if useful) are always input to some decision (made by machine or human).

Be aware of what that decision is.

Consider whether the test results answer the questions you think they do.

Non-Deterministic Testing



DELIBERATE RANDOMNESS IN TESTS

Increases test coverage over time, but makes tests inconsistent.

Partial protection against sunshine thinking.

RANDOM FAULT INJECTION

A type of fault-tolerance testing.

Randomly destroy parts of the environment (e.g. terminating virtual machines).

Drives an anti-fragile mindset.

Can be done in the production environment, or not.

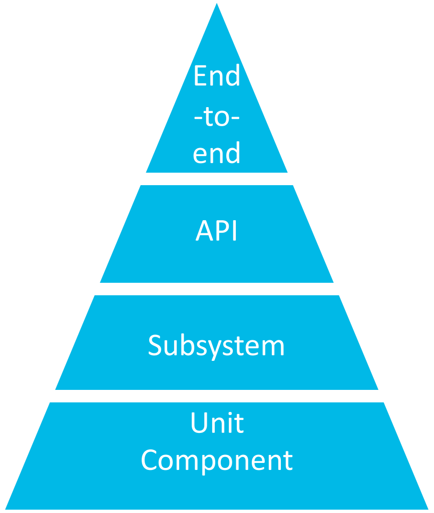
NON-RANDOM FAULT INJECTION

E.g. Lineage Driven Fault Injection.

Combine fault injection with automated reasoning to achieve greater test coverage.

Active research field.

Choosing the Item Under Test



WHERE DO YOU TEST THE SYSTEM?

Testing requires resources. Where we choose to execute influences the cost.

HIGH LEVELS OF SYSTEM COMPLETENESS

Testing the complete system is ”easy”...

... but slow...

... and expensive.

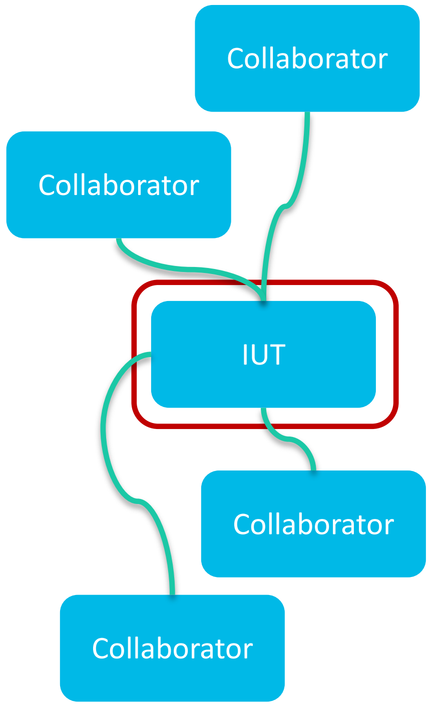
LOW LEVELS OF SYSTEM COMPLETENESS

Testing at low levels of system completeness is fast and cheap...

... but requires you to think hard about what the result actually means...

... and what kind of feedback you’ll get out of it depends on the architecture.

Isolating the Item Under Test



WHY ISOLATE IT?

Speed and efficiency!

To reduce uncertainties.

Isolating the IUT from its collaborators lets you study precisely how the IUT behaves...

... at the cost of not knowing how it behaves together with those collaborators.

TEST DOUBLES

Various techniques for replacing actual collaborators exist.

Nomenclature somewhat confused: fakes, stubs, spies, mocks...

Very helpful for slicing the system into testable parts, but...

... incurs overhead and maintenance costs.

LESS DUMB TEST DOUBLES

Opinions on what the difference is vary, if there is a difference, but...

Some test double techniques are more dynamic.

Some types of test doubles can inform on the IUT.

Choosing When to Execute

WHAT ARE THE CRITICAL RESOURCES?

Lead time

Computing resources

Attention

WHEN DO WE NEED THE RESULT?

Sooner is always better!

Is the result critical to an informed release decision?

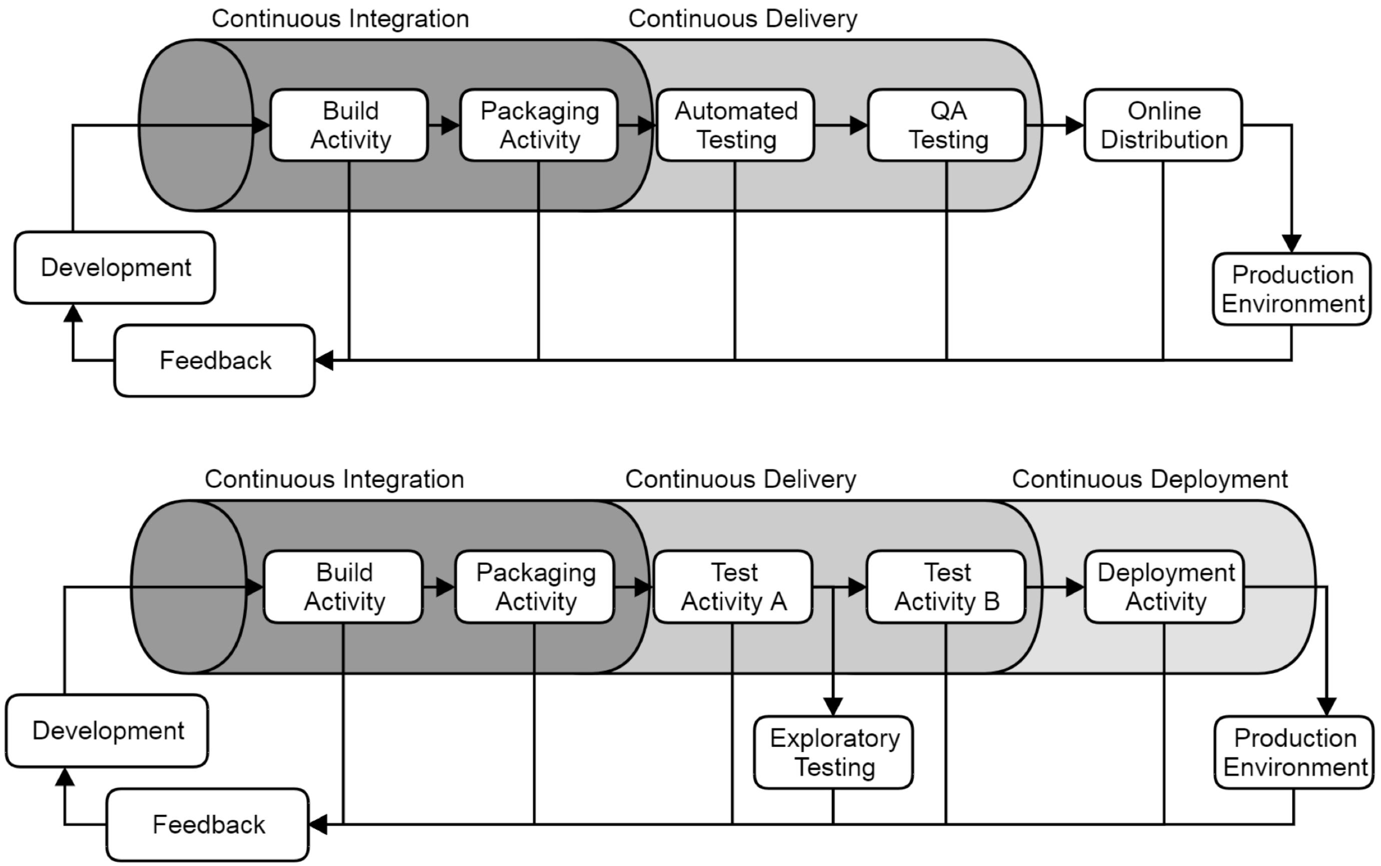
Is the result feedback for further product improvement?

WHO'S "WE"?

Test results are not only for developers. Everyone should be interested...

... project managers
... product owner
... QA responsibles
... release managers
... developers
... testers
... <insert role here>

Thinking in Pipelines



LINKÖPING UNIVERSITY

Choosing What to Execute

FIXED VERSUS DYNAMIC TEST CASE SELECTION

Straight-forward approach: let the pipeline execute the same test scope again, and again, and again.

Less straight-forward approach: automatically (and hopefully intelligently) select the test cases that give you the best value-to-cost ratio right now.

SELECTION CRITERIA

Selection may be done based on a number of criteria:

... Previous results
... Test outcome correlations
... What changes
... Why it changed
... Who changed it
... What hasn't been executed in a long time
...

CHALLENGES

You need three things:

... Pipeline traceability
... Test case meta-data
... A configurable selection engine

None of these you'll get for free!

Things to Cover

What Makes a Bad Test Case?

FLAKINESS

Flakiness is when tests randomly fail or pass for the same configuration.

This leads to:
... poor confidence
... broken windows effect
... uncertainty

How do you react to flaky tests? Re-run them?

What if it’s not the test that’s flaky?

TOO SLOW/EXPENSIVE TO RUN

Long-running tests may give you great feedback, but that’s no use if you receive it too late.

Testing *always* comes down to cost vs. benefit. If the test is too expensive it doesn’t matter how good it is.

OTHER FACTORS

Does the test reveal anything truly relevant? (Or does it just improve your coverage metrics?)

Is the test repeatable? Non-repeatable tests have their place, though.

Can the test be executed independently? Watch out for hidden dependencies to other tests.

Making Test Results Actionable

TRUST IS EVERYTHING

An organization will only act on results it trusts.

Flakiness is death.

AVOID INFORMATION OVERLOAD

If you have too many test results, you need to filter or boil them down into key metrics.

Which tests and test results are truly critical? Which need action now, and what is merely nice-to-have feedback?

Be explicit about the value and purpose of your tests.

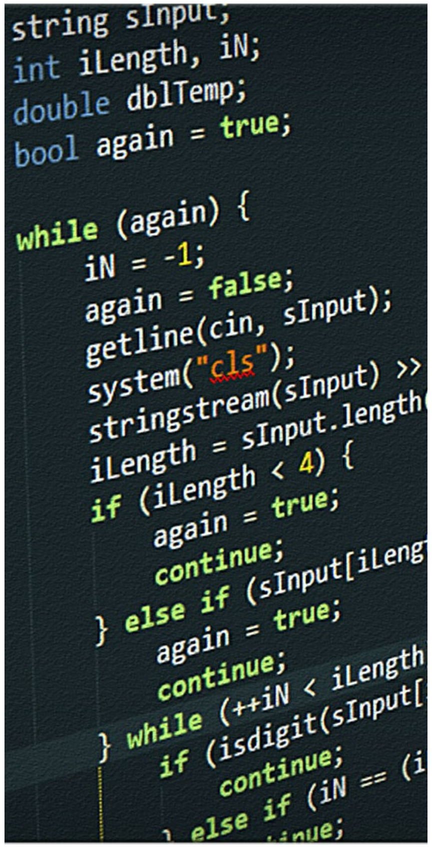
FIND THE CORRECT AUDIENCE

Who need to see which test results?

The needs of developers in different parts of the system will differ...

... and their needs will differ from the project manager's.

Testable Code



```
string sInput;
int iLength, iN;
double dblTemp;
bool again = true;

while (again) {
    iN = -1;
    again = false;
    getline(cin, sInput);
    system("cls");
    stringstream(sInput) >>
    iLength = sInput.length
    if (iLength < 4) {
        again = true;
        continue;
    } else if (sInput[iLeng
        again = true;
        continue;
    } while (++iN < iLength
        if (isdigit(sInput[
            continue;
        } else if (iN == (i
            inue;
```

WRITING TEST CASES IS EASY…

...at least if the IUT makes it easy!

(Although non-trivial test cases and test frameworks require thorough architecture and diligence, just like production code.)

… WRITING TESTABLE CODE IS HARD!

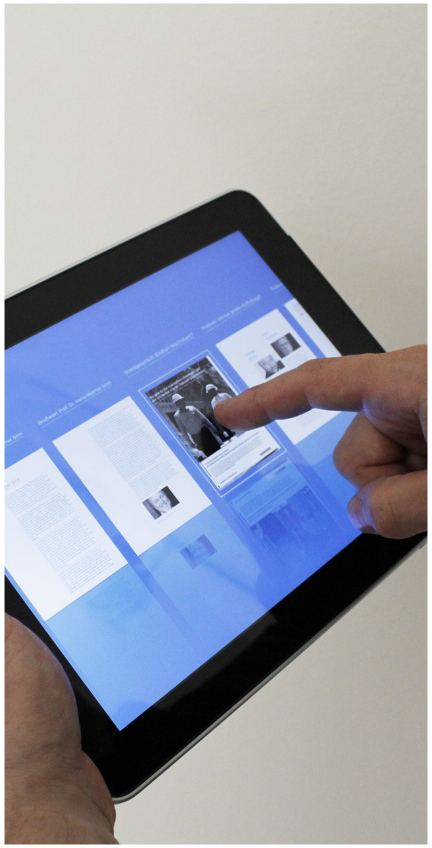
Writing testable code requires you to keep complexity low and create seams where tests can be inserted.

TESTABLE CODE IS NOT JUST TESTABLE

Testable code is loosely coupled code.

Testable code is also reusable code. (That's what testing is, reusing!)

GUI Test Automation



WHAT ABOUT WHEN THE IUT ISN'T (EXPOSED) CODE?

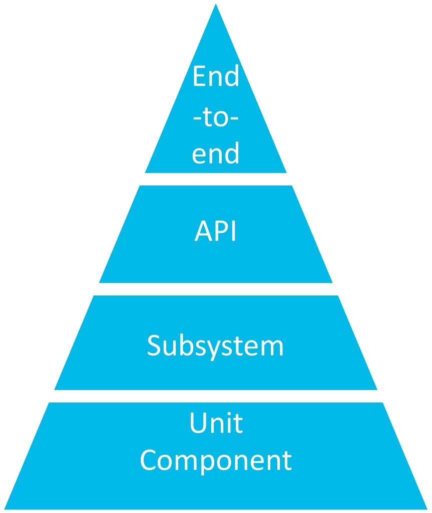
Prominent example is GUI, but analogous to all human-machine interfaces and physical interfaces.

GUI TESTING PARADIGMS

User behavior capture and playback.

Event capture.

Image/text recognition.



Test Driven Development

LET'S WRITE THE TESTS FIRST!

Force yourself to write testable code.

Requires practice and discipline.

TEST DRIVEN DEVELOPMENT IS NOT A TESTING PRACTICE

It's a development practice.

It also happens to produce a lot of test cases, which you might want to use.

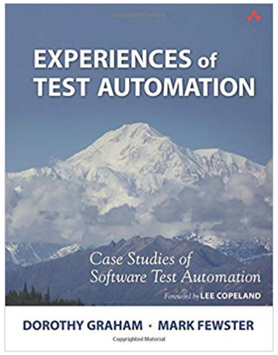
CAN BE DONE AT DIFFERENT LEVELS

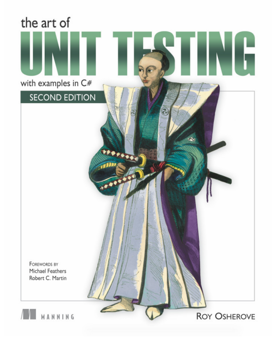
Recall the test pyramid. Test Driven Development can be applied at any level, but is most common at the lowest level.

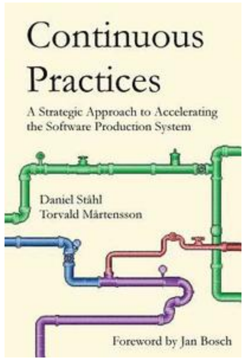
Live Demo!

The End!

FURTHER READING


EXPERIENCES of TEST AUTOMATION
Case Studies of Software Test Automation
Foreword by LEE COPELAND
DOROTHY GRAHAM · MARK FEWSTER





!?

Daniel Ståhl

www.liu.se

LINKÖPING
UNIVERSITY

Things to Cover

What is your background?

m.voto.se/tddd04-ta-1

LINKÖPING UNIVERSITY

What is your background?

m.voto.se/poll/tddd04-ta-1

What to do about flakiness?

m.voto.se/tddd04-ta-2

What to do about flakiness?

m.voto.se/poll/tddd04-ta-2

Have you tried TDD?

m.voto.se/tddd04-ta-3

Have you tried TDD?

m.voto.se/poll/tddd04-ta-3