"It's a trap!"
How not testing risks is a huge risk

Dan Ashby
@DanAshby04
www.DanAshby.co.uk
www.SoftwareTestingClinic.com
www.TestingInThePub.co.uk
Does any of this sound familiar WITHIN YOU PROJECT?
Risk (plural: risks)
Noun
1. A possible, usually negative, outcome, e.g., a danger.
2. The likelihood of a negative outcome.
3. (Formal use in business, engineering, etc.) The potential (conventionally negative) effect of an event, determined by combining the likelihood of the event occurring with the effect should it occur.

What about risks?
Different Perspectives of Risks

- Business Risks
- People Risks
- Project Risks
- Product Risks
Let's do an EXERCISE
How would you test this chair?
Are you jumping straight to test ideas?

Is there structure to your testing?

What info are your test ideas trying to uncover?

What about risks?
Let's explore INFORMATION
All software starts with a small piece of info...
We put processes and activities in place to investigate and evaluate that info...
That investigation allows us to form more artefacts with more info...
Those artefacts containing more info can then be used to stem other activities...
Those activities create the project outputs & allow us to measure success and feed back into new ideas.
The 5 orders of IGNORANCE
KNOWLEDGE

Known information that we have available
It’s what we know and expect
KNOWLEDGE

Known information that we have available
It’s what we know and expect

LACK OF KNOWLEDGE

When we don’t know something, but we are aware of what it is we don’t know
KNOWLEDGE
Known information that we have available
It’s what we know and expect

LACK OF KNOWLEDGE
When we don’t know something, but we are aware of what it is we don’t know

LACK OF AWARENESS
When we are completely unaware that we don’t know something
KNOWLEDGE
Known information that we have available
It’s what we know and expect

LACK OF KNOWLEDGE
When we don’t know something, but we are aware of what it is we don’t know

LACK OF AWARENESS
When we are completely unaware that we don’t know something

LACK OF PROCESSES
When we have no processes in place to allow us to become aware of our unknowns, or to be able to react to them
KNOWLEDGE
- Known information that we have available
- It’s what we know and expect

LACK OF KNOWLEDGE
- When we don’t know something, but we are aware of what it is we don’t know

LACK OF AWARENESS
- When we are completely unaware that we don’t know something

LACK OF PROCESSES
- When we have no processes in place to allow us to become aware of our unknowns, or to be able to react to them

META IGNORANCE
- When we are completely unaware of there being 5 orders of ignorance

When we are completely unaware that we have available
- It’s what we know and expect
Why is this IMPORTANT?
The purpose of testing is to UNCOVER INFORMATION IMPORTANT?
INFORMATION

- KNOWN
  - EXPLICIT

- TACIT & IMPLICIT

- UNDERSKIRKINS
  - THAT WE’RE AWARE OF

- UNAWARENESS
  - (OF UNKNOWNS)

- Investigative Testing
  - (Exploratory)
  - (Test Charters)

- Assertive Testing
  - (Scripted)
  - (Test Cases)

- Person

- Automation

- confirms
- informs
- uncovers
- assists
TESTING & RISKS
A model to help uncover risks...

"THE THING" WE ARE TESTING

VARIABLES OF "THE THING" (PURPOSES, PROPERTIES, STATES, ETC)

VARIABLES OF THE PURPOSES, PROPERTIES, STATES, ETC

PRODUCT RISKS AT ALL LEVELS

FOR EACH RISK:
IS THIS RISK IMPORTANT? (I.E. DO WE CARE ABOUT THIS RISK?)

YES

NO
ADD A NOTE AND MOVE ON

TEST CHARTER & FURTHER INVESTIGATION

ENVIRONMENT EFFECTS

AFFECT

EFFECTS OF TIME
What are the properties, purposes & states of the chair?
So what product risks can you think of now?
The Pen Game
“TESTING” isn’t just about TESTING DEVELOPED SOFTWARE
Testing the...

...IDEAS
...ARTEFACTS
...UX/UI DESIGNS
...CODE DESIGN
...CODE
...SOFTWARE
...PIPELINES
...ENVIRONMENTS
...MONITORING
...PROCESSES
IDEAS FOR NEW FEATURES

ARTEFACT GENERATION

UX/UI DESIGN

CODE DESIGN (FDD)

WRITING CODE

EXECUTING CODE

COMMITTING CODE

RELEASE PIPELINE (PRE-PROD ENVS)

RELEASE PIPELINE (PROD ENVS)

USERS USING THE SOFTWARE

PRE-DEV

DURING DEV

POST-DEV

TESTING IDENTIFIES INFO ABOUT RISKS

FASTEST FEEDBACK LOOPS REGARDING REFACTORING IDEAS, ARTEFACTS & DESIGN
IDEAS FOR NEW FEATURES

ARTIFACT GENERATION

UX/UI DESIGN

PRE-DEV

CODE DESIGN (FDD)

WRITING CODE

EXECUTING CODE

COMMITTING CODE

DURING DEV

RELEASE PIPELINE (PRE-PROD ENVs)

RELEASE PIPELINE (PROD ENVs)

USERS USING THE SOFTWARE

POST-DEV

TESTING IDENTIFIES INFO ABOUT RISKS

FASTEST FEEDBACK LOOPS REGARDING REFAC TORIZING IDEAS, ARTIFACTS & DESIGNS

INFORMS DECISIONS ON PREVENTING PROBLEMS
IDEAS FOR NEW FEATURES

ARTIFACT GENERATION

UX/UI DESIGN

PRE-DEV

CODE DESIGN (FDD)

WRITING CODE

EXECUTING CODE

COMMITTING CODE

DURING DEV

RELEASE PIPELINE (PRE-PROD ENVs)

RELEASE PIPELINE (PROD ENVs)

USERS USING THE SOFTWARE

POST-DEV

TESTING IDENTIFIES INFO ABOUT RISKS

FASTEST FEEDBACK LOOPS REGARDING REFACTORING IDEAS, ARTIFACTS & DESIGNS

INFORMS DECISIONS ON PREVENTING PROBLEMS

TESTING IDENTIFIES INFO ABOUT RISKS & CODE PROBLEMS
IDEAS FOR NEW FEATURES

ARTIFACT GENERATION

UX/UI DESIGN

CODE DESIGN (FDD)

WRITING CODE

EXECUTING CODE

COMMITTING CODE

RELEASE PIPELINE (PRE-PROD ENVs)

RELEASE PIPELINE (PROD ENVs)

USERS USING THE SOFTWARE

PRE-DEV

DURING DEV

POST-DEV

TESTING IDENTIFIES INFO ABOUT RISKS & CODE PROBLEMS

FASTEST FEEDBACK LOOPS REGARDING REFACTORING IDEAS, ARTIFACTS & DESIGNS

INFORMS DECISIONS ON PREVENTING PROBLEMS

TESTING IDENTIFIES INFO ABOUT RISKS & CODE PROBLEMS

FAST FEEDBACK ON CODE

SLOWER FEEDBACK ON IDEAS, ARTIFACTS & DESIGNS
Idea for new features

Artefact generation

UX/UI design

Pre-dev

Code design (FDD)

Writing code

Executing code

Committing code

During dev

Release pipeline (pre-prod envs)

Release pipeline (prod envs)

Users using the software

Post-dev

Testing identifies info about risks

Fastest feedback loops regarding refactoring ideas, artefacts & designs

Informs decisions on preventing problems

Testing identifies info about risks & code problems

Fast feedback on code

Slower feedback on ideas, artefacts & designs

Informs decisions on preventing problems
IDEAS FOR NEW FEATURES

ARTEFACT GENERATION

UX/UI DESIGN

PRE-DEV

CODE DESIGN (FDD)

WRITING CODE

EXECUTING CODE

COMMITTING CODE

DURING DEV

RELEASE PIPELINE (PRE-PROD ENV)

RELEASE PIPELINE (PROD ENV)

POST-DEV

USERS USING THE SOFTWARE

TESTING IDENTIFIES INFO ABOUT RISKS & CODE PROBLEMS

FASTEST FEEDBACK LOOPS REGARDING REFACTORING IDEAS, ARTEFACTS & DESIGNS

INFORMS DECISIONS (ON PREVENTING PROBLEMS)

TESTING IDENTIFIES INFO ABOUT RISKS & CODE PROBLEMS

FAST FEEDBACK ON CODE

SLOWER FEEDBACK ON IDEAS, ARTEFACTS & DESIGNS

INFORMS DECISIONS (ON PREVENTING PROBLEMS)

TESTING DETECTS PROBLEMS IN SW (RISKS ARE REALISED)
Ideas for new features

Artefact generation
UX/UI design

Code design (FDD)

Writing code
Executing code
Committing code

Release pipeline (pre-prod envs)

Users using the software

Testing identifies info about risks
Testing identifies info about risks & code problems
Testing detects problems in SW (risks are realised)

Fastest feedback loops regarding refactoring ideas, artefacts & designs

Informs decisions on preventing problems

Fast feedback on code
Slower feedback on ideas, artefacts & designs

Slowest possible feedback loops on ideas, artefacts & designs
Slow feedback on code...
IDEAS FOR NEW FEATURES

PRE-DEV

ARTEFACT GENERATION
UX/UI DESIGN

DURING DEV

CODE DESIGN (FDD)
WRITING CODE
EXECUTING CODE
COMMITTING CODE

POST-DEV

RELEASING PIPELINE (PRE: PROD ENV)
RELEASING PIPELINE (PROD ENV)

USERS USING THE SOFTWARE

TESTING IDENTIFIES INFO ABOUT RISKS
FASTEST FEEDBACK LOOPS REGARDING REFACORING IDEAS, ARTEFACTS & DESIGNS
INFORMS DECISIONS ON PREVENTING PROBLEMS

TESTING IDENTIFIES INFO ABOUT RISKS & CODE PROBLEMS
FAST FEEDBACK ON CODE
SLOWER FEEDBACK ON IDEAS, ARTEFACTS & DESIGNS
INFORMS DECISIONS ON PREVENTING PROBLEMS

TESTING DETECTS PROBLEMS IN SW (RISKS ARE REALISED)
SLOWEST POSSIBLE FEEDBACK LOOPS ON IDEAS, ARTEFACTS & DESIGNS
SLOW FEEDBACK ON CODE
INFORMS DECISIONS ON RESOLVING PROBLEMS
**IDEAS FOR NEW FEATURES**

**CODE DESIGN (FDD)**

**EXECUTING CODE**

**COMMITTING CODE**

**RELEASE PIPELINE (PRE-PROD ENVs)**

**RELEASE PIPELINE (PROD ENVs)**

**USERS USING THE SOFTWARE**

---

**PRE-DEV**

**DURING DEV**

**POST-DEV**

**TESTING IDENTIFIES INFO ABOUT RISKS & CODE PROBLEMS**

**TESTING DETECTS PROBLEMS IN SW (RISKS ARE REALISED)**

**SLOWEST POSSIBLE FEEDBACK LOOPS ON IDEAS, ARTEFACTS & DESIGNS**

**SLOW FEEDBACK ON CODE**

**INFORMS DECISIONS ON RESOLVING PROBLEMS**

**INFORMS DECISIONS ON PREVENTING PROBLEMS**

**FAST FEEDBACK ON CODE**

**INFORMS DECISIONS ON PREVENTING PROBLEMS**

**INFORMS DECISIONS ON RESOLVING PROBLEMS**

**FASTEST FEEDBACK LOOPS REGARDING REFACTORING IDEAS, ARTEFACTS & DESIGNS**

**INFORMS DECISIONS ON PREVENTING PROBLEMS**
Investigation to UNCOVER RISKS vs Investigating the UNCOVERED RISKS
TYPES OF RISKS
over types of testing
Types of testing
Types of testing
Types of testing
Types of testing
Types of testing
Types of testing
Types of testing
Types of testing
Types of testing

A TYPE OF TESTING = TESTING THAT FOCUSES ON A TYPE OF RISK
The problems with using “types of testing”...
Types of risks & RISK CATEGORIES
accessibility  
accountability  
accuracy  
adaptability  
administrability  
affordability  
agility  
auditability  
autonomy  
availability  
capability  
charisma  
compatibility  
configurability  
correctness  
credibility  
customizability  
data  
debugability  
degradability  
determinability  
demonstrability  
dependability  
deployability  
discourability  
durability  
effectiveness  
efficiency  
evovlability  
extensibility  
failure  
functionality  
transparency  
fault-tolerance  
fidelity  
flexibility  
inspectability  
installability  
Integration  
integrity  
interchangeability  
interoperability  
learnability  
maintainability  
manageability  
robility  
modifiability  
modularity  
operability  
orthogonality  
portability  
precision  
predictability  
capabilities  
performance  
producibility  
provability  
recoverability  
relevance  
reliability  
repeatability  
regression  
resilience  
responsiveness  
reusability  
robustness  
safety  
scalability  
seamlessness  
sustainability  
serviceability  
supportability  
security  
securability  
simplicity  
stability  
standards  
compliance  
survivability  
sustainability  
testability  
timeliness  
traceability  
transparency  
ubiquity  
understandability  
upgradability  
usability  
vulnerability  
+ MORE!
And there are sub-categories of risks too.
Using product risks as 
HEURISTICS
Heuristic

Heuristic (plural: heuristics)

Adjective
1. Of or relating to a usually speculative formulation serving as a guide in the investigation or solution of a problem
2. Of or constituting an educational method in which learning takes place through discoveries that result from investigations made by the student.

Noun
1. A heuristic method or process.
2. The study and application of heuristic methods and processes.

In the Context Driven Testing community:
“Heuristics are a fallible method to solving a problem”

Me:
“A trigger for an idea”
Using risks as heuristics allows us to generate ideas on how the software (or ideas) might fail.
It supplies structure to our investigative testing
Test Charters are a good example

Explore [Target area]
With [Resources]
To Discover [Info about a risk]
Test Charters are a good example

Explore [the comments box]
With [the big list of naughty strings*]
To Discover [data type risks]

* https://github.com/minimaxir/big-list-of-naughty-strings/blob/master/blns.txt
Test Charters are a good example

Explore [the login screen]
With [10K users]
To Discover [user load risks]
Test Charters are a good example

Explore [the accounts API]
With [10K strings in a single transaction]
To Discover [data load risks]
Not thinking about risks IS A HUGE RISK
Thank you!

Dan Ashby

@DanAshby04

www.DanAshby.co.uk
www.SoftwareTestingClinic.com
www.TestingInThePub.co.uk