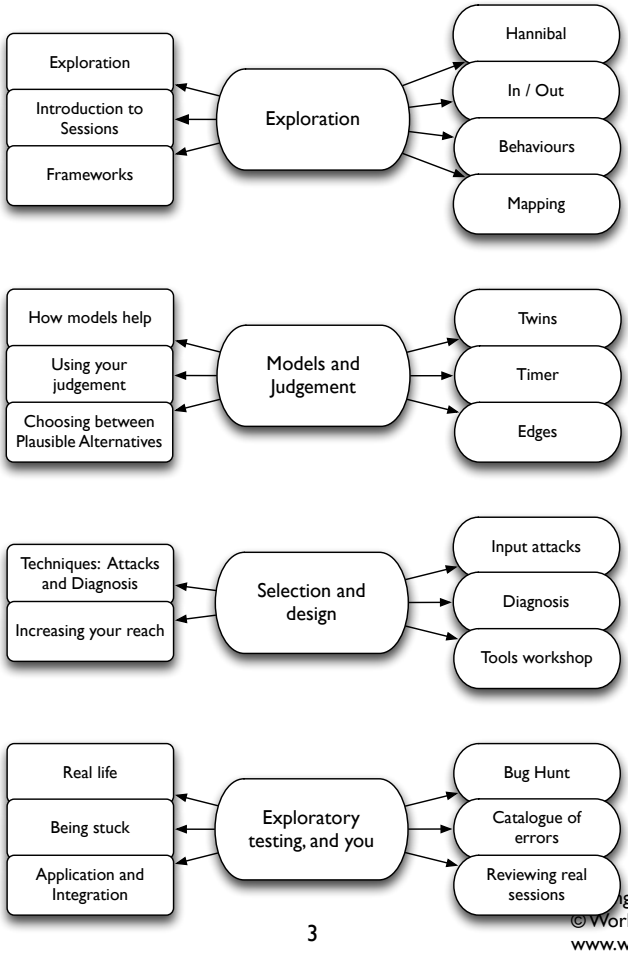
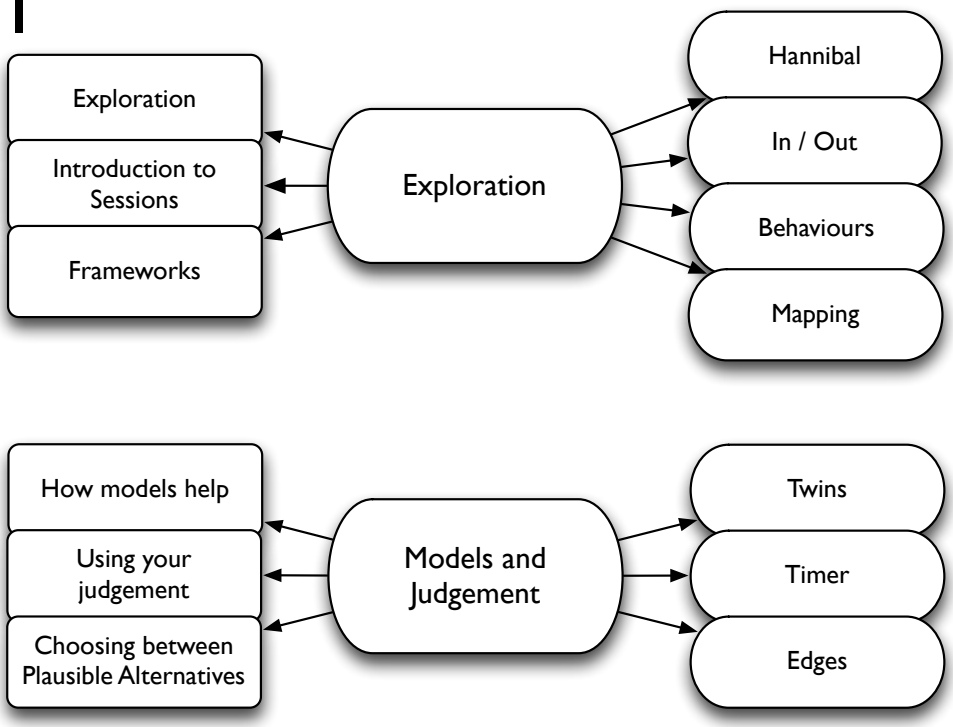
Getting a Grip on Exploratory Testing

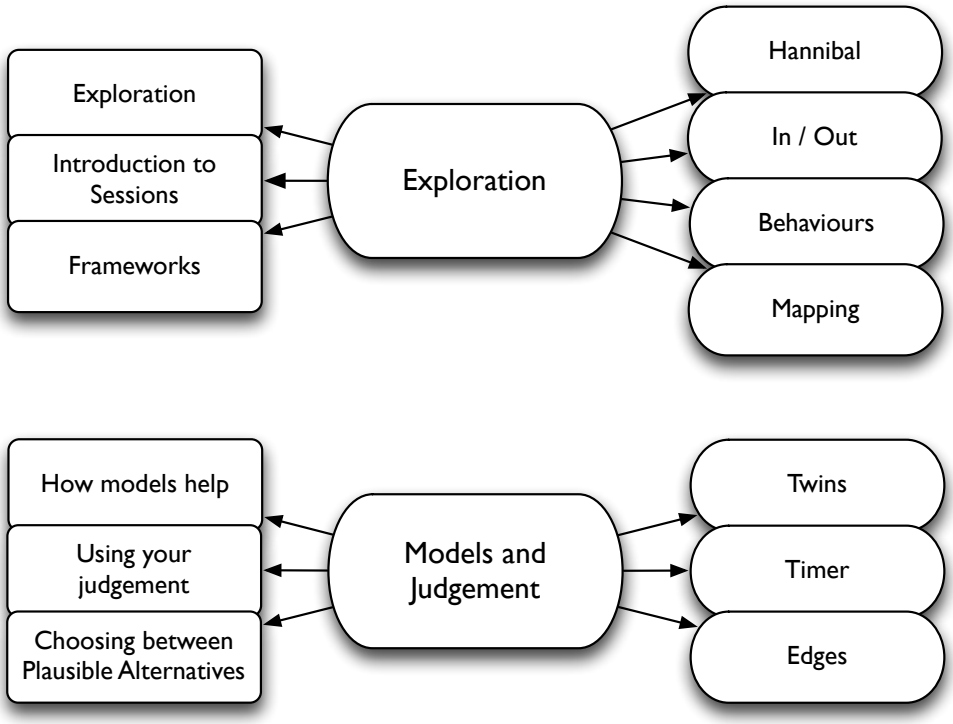
James Lyndsay
Workroom Productions

hiis.workroomprds.com



Day 1





Hands on: Approaches to Exploration

Second Punic War

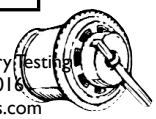
Who: Carthaginians vs Romans

What: Power Struggle

When: 218 - 204 BC

Where: Italy, Spain, Western Mediterranean

Bug Description	How can you be so sure?
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	



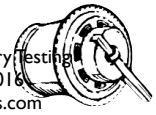
Techniques change dynamically

What systematic techniques did you use?

How did you decide to start using one?

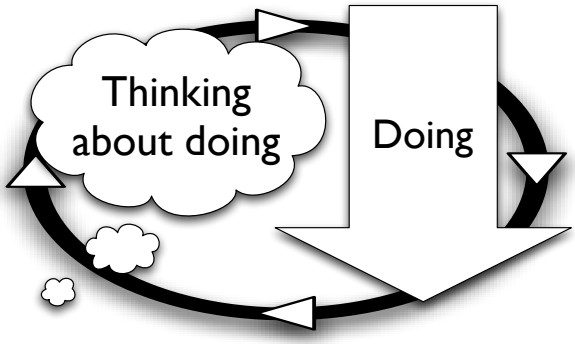
How did you decide to stop using it?

What might you do differently next time?



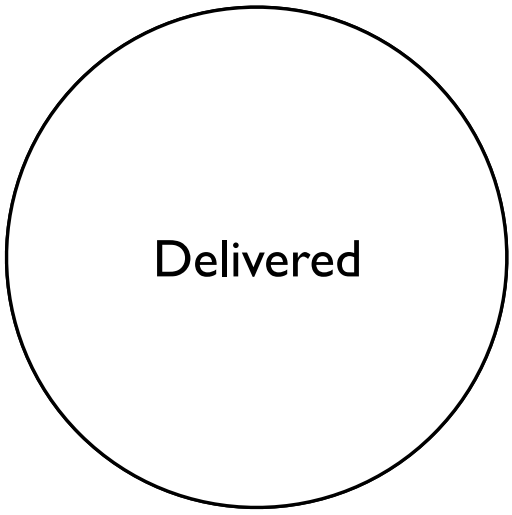
Bug Description	How can you be so sure?
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

Exploration

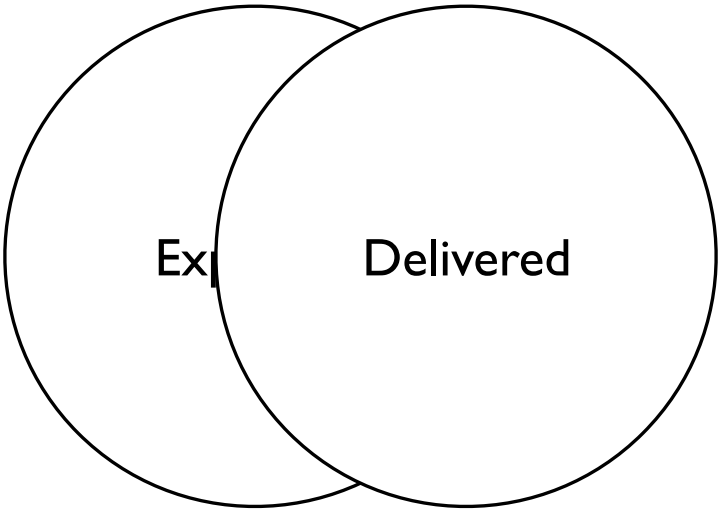


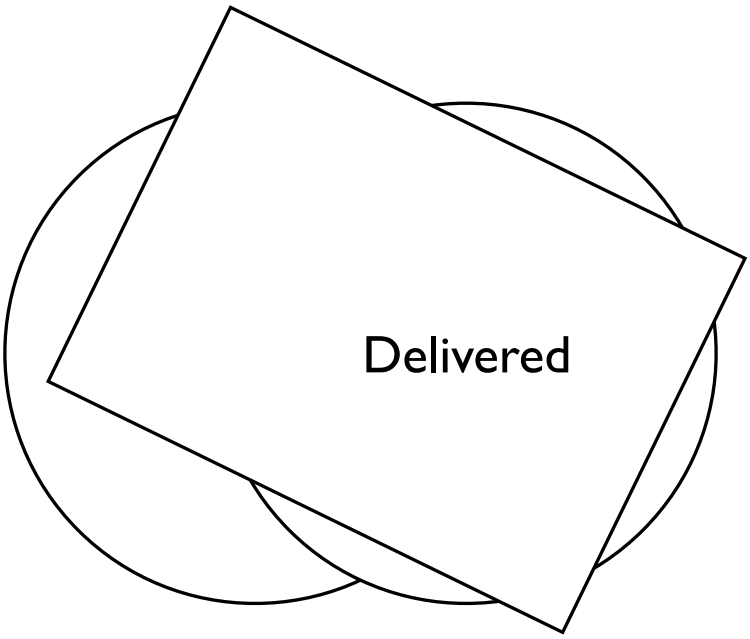
- Reliable discovery of information
- Methods have a start point, a process
- Methods may change based on circumstances

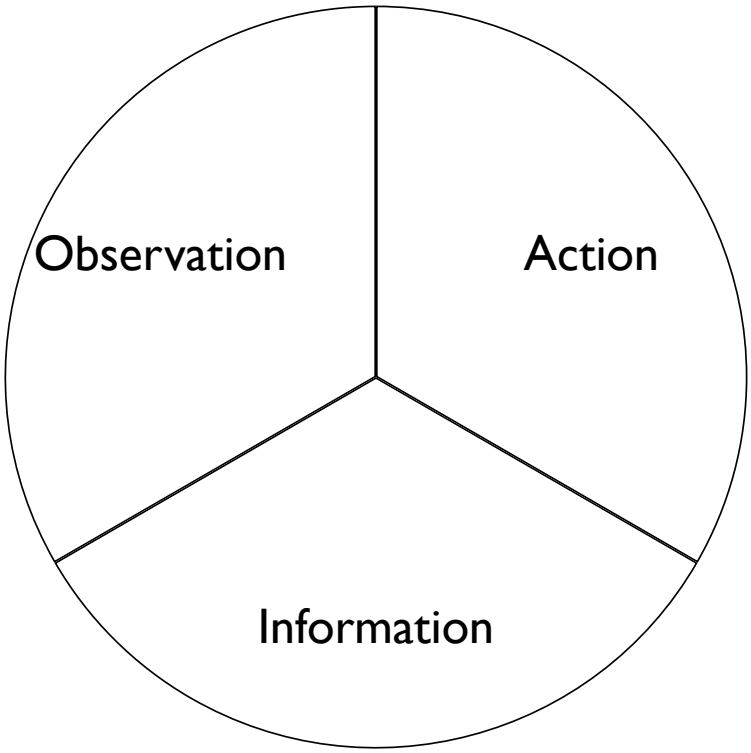
Testing

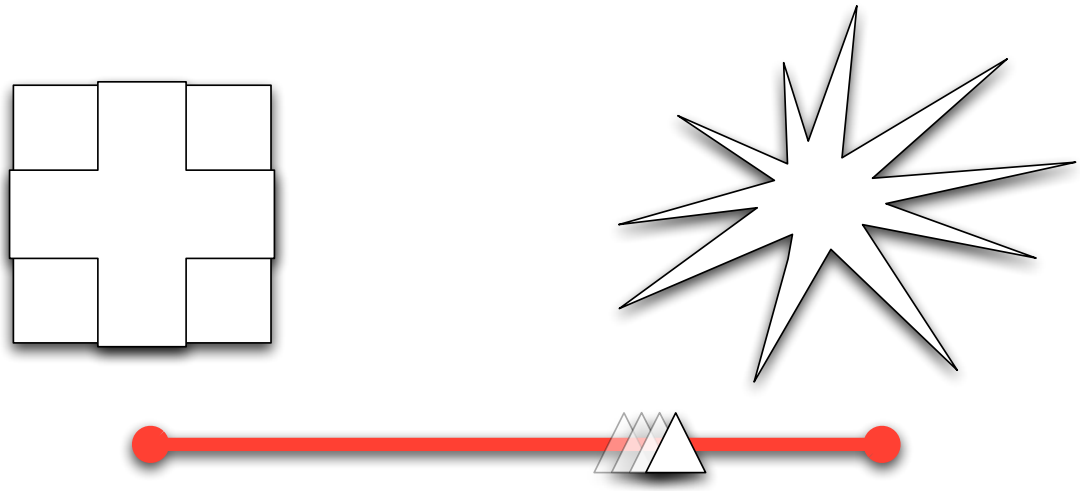


Testing









Systematic

Best practices", white-box techniques, waterfall, requirements-based testing

Formal

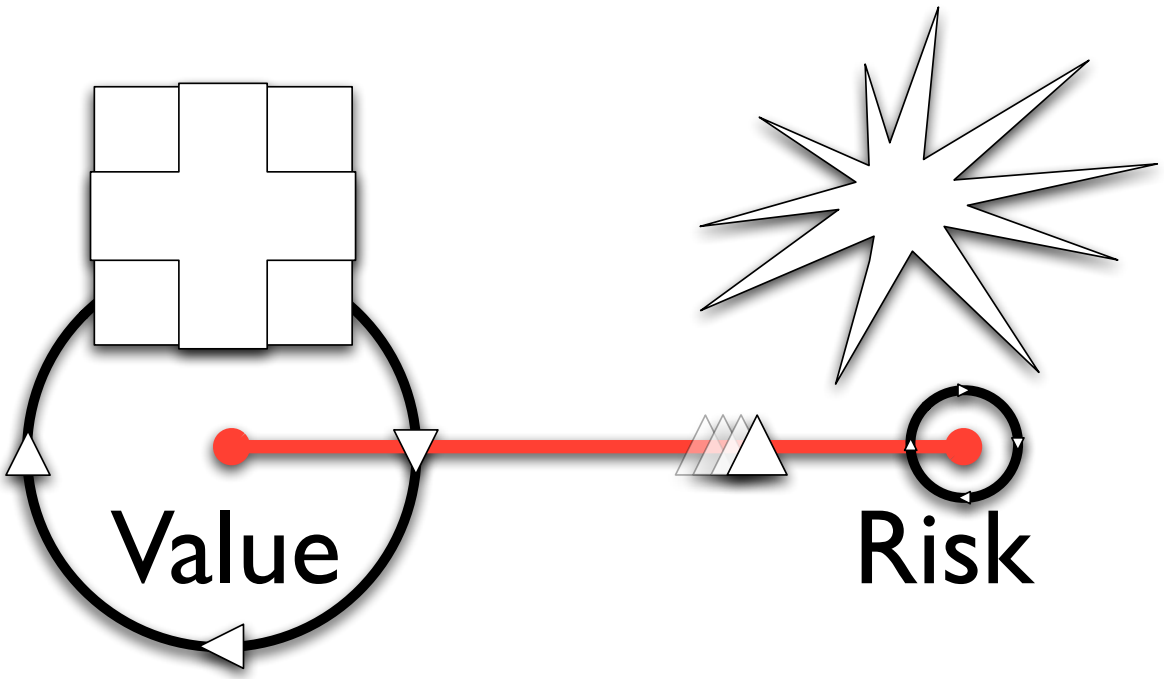
Safety-critical, software engineering, CMM, standards and methodologies, automated testing

Ad-hoc

Agile methods, emergent behaviours, context-driven schools, software attacks, risk-based testing

Informal

Scant documentation, poor requirements, time, skilled manual testing



Slow feedback

Fast feedback

Scripted

Exploratory

Exploratory testing is...

"Exploratory testing is simultaneous learning, test design, and test execution"

Bach (James), Exploratory Testing Explained v.1.3 4/16/03

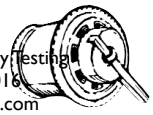
www.satisfice.com/articles/et-article.pdf

Exploratory testing is...

"Exploratory testing is ~~simultaneous~~ learning, test design, and test execution"

Bach (James), Exploratory Testing Explained v.1.3 4/16/03

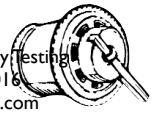
www.satisfice.com/articles/et-article.pdf



Exploratory testing is...

"Exploratory testing is parallel learning, test design, and test execution"

Bolton, Exploratory Testing and Review, blog post Sept 09

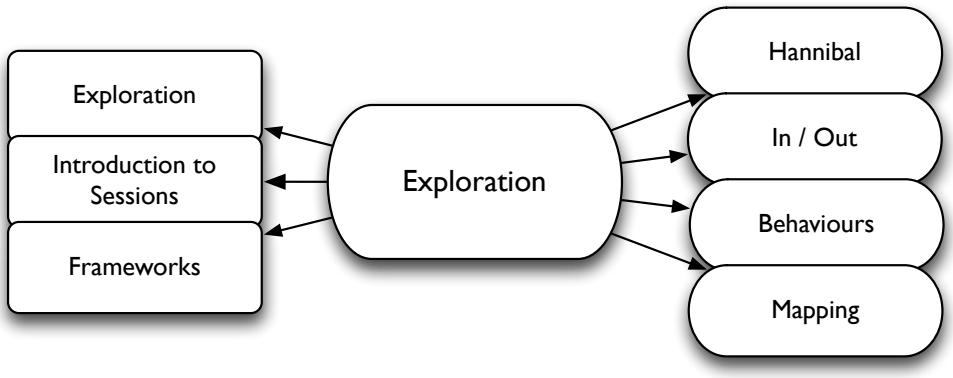


Best practice, or context-based?

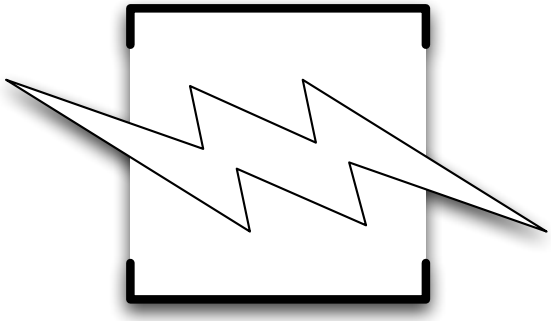
"I remain concerned about relying solely on exploratory testing to the exclusion of a planned, systematic process of test analysis, design, and implementation occurring simultaneously with the development of the system."

"I now feel that exploratory testing is a best practice for most test projects"

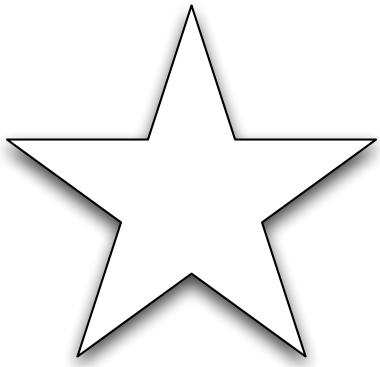
Black, Better Software, May/June 2004



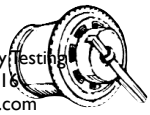
ET in practice: Key Issues



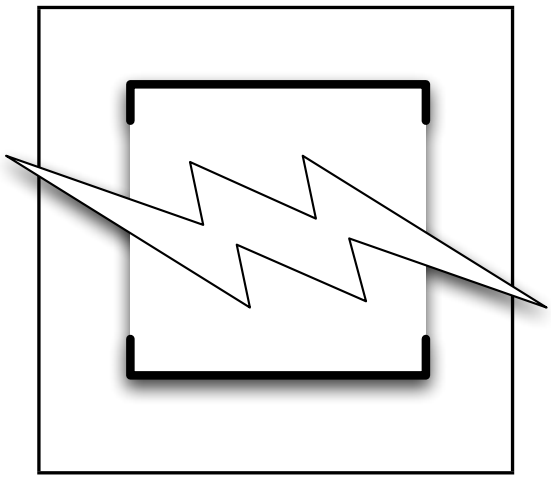
Open-ended

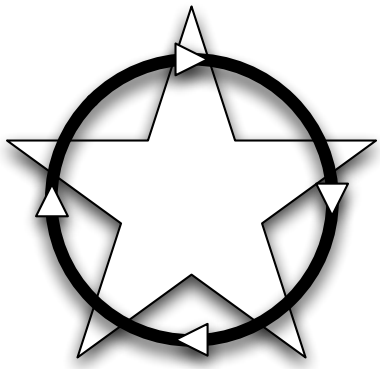


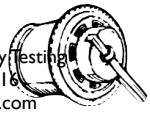
Skills



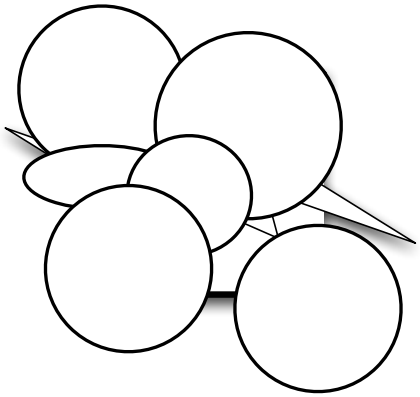
ET in practice: Making it work
Session-based testing

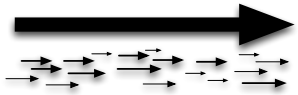


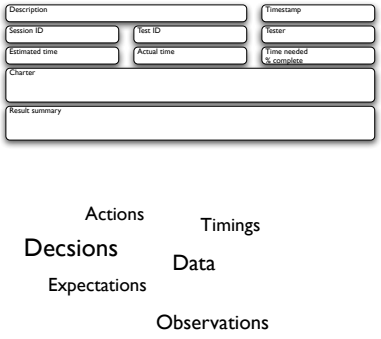




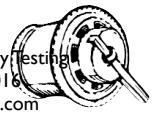
Session-based testing: Components



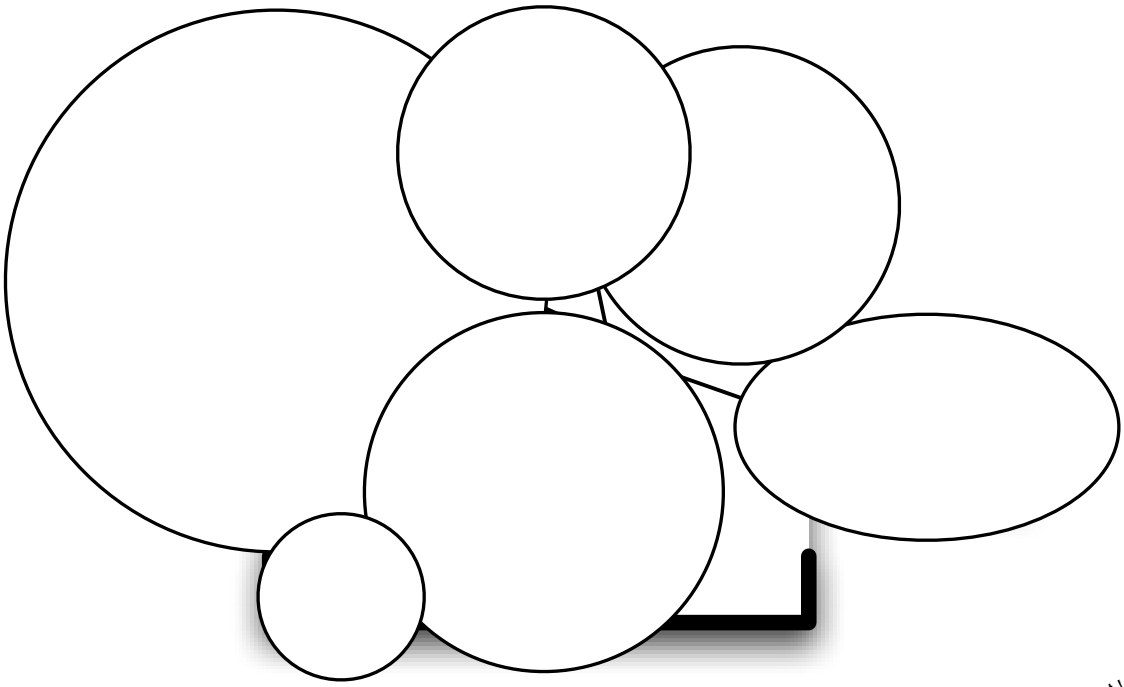




Multiple limits Timebox Record



Sessions: Charters



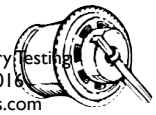
Sessions and Charters

Limited scope of deliverable / test task

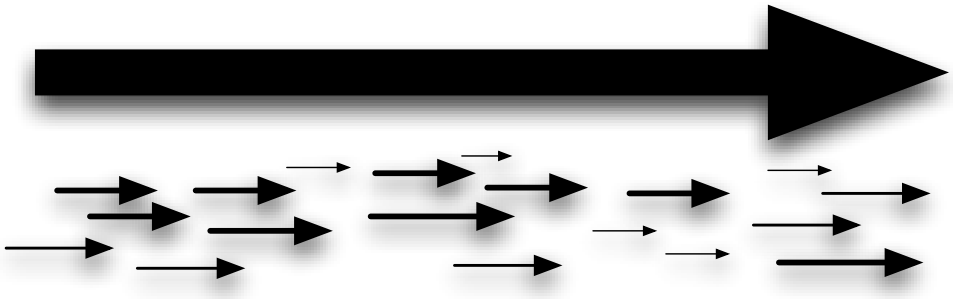
Make a distinction between:

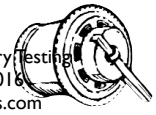
Test Point / Charter: Piece of **work**, may be repeated

Session: Unit of **time**, an event

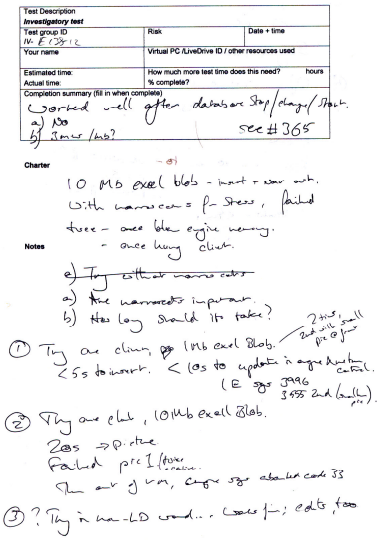


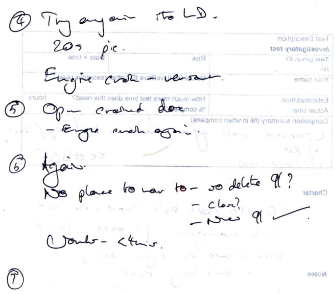
Sessions: Timeboxes

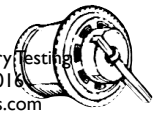




Recording a session







How to record it

Paper+pen or Machine?

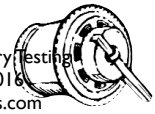
Video, screensnap, keylogger, partner, transaction logger

Sequence, list, outline, mindmap, picture, postits

Include time spent not testing?

Clearing environment, setting data, logging bugs?

Consistency is important for review / summarisation / metrics



My preference

! important

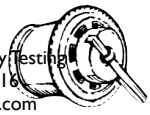
¿ something I'm not sure of

? a question for someone or something

* return to this

[] around something not in timeflow

plenty of arrows!



What to record?

Identifiers:

who – when – what

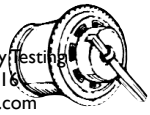
Qualities:

risk – estimated time – dependencies

As you go:

actions – events – data – expectations – bugs – plans – interruptions – actual time taken – time wanted – problems

Decisions

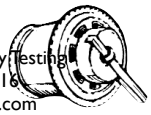


Why record?

Remember – Mnemonic help

Review – Sharing, improvement

Return – Historical analysis, long-term project memory



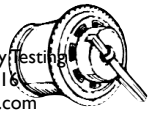
Metrics

Examine the session sheet...

- what metrics would be available?
- what could they tell us?

Any obvious aggregates? trends?

How could we judge progress?

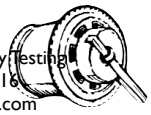


Reporting metrics

Who cares?

Why?

What can we offer?



Metrics I have used

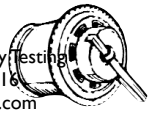
Time spent

Bugs found

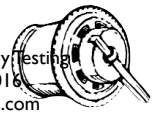
Charters and sessions

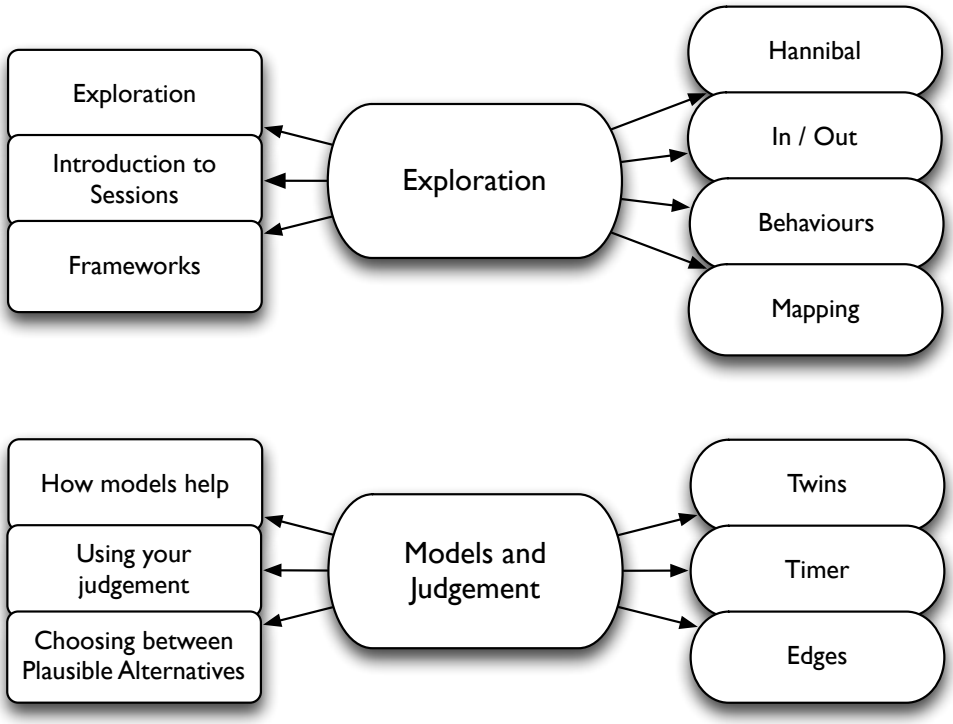
Aggregate subjective assessment:

How done are we?



Hannibal again!





Exploration: Machine A

Framework:

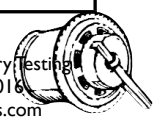
Input

Output

Linkage

Machine A

Input	Linkage	Output

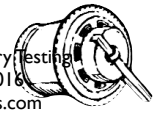


Discussion: input / output / linkage

What characterises an input?

What characterises an output?

What kinds of linkage could there be?

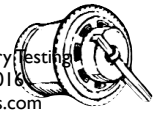


Exploration: Machine B

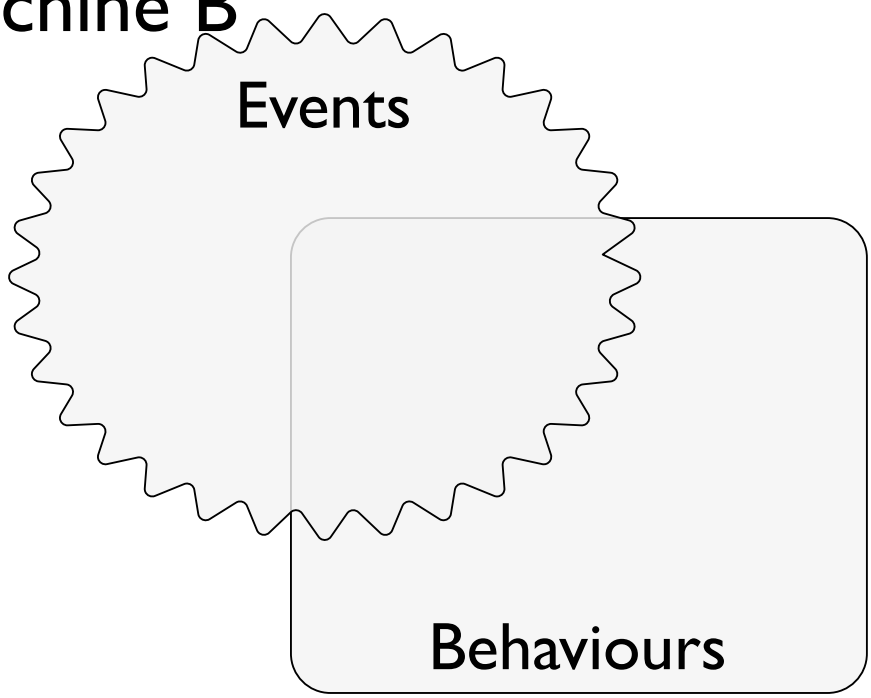
Framework:

Event

Behaviour



Machine B



Discussion

What is an event?

What events have you seen? Have you triggered these events?

What is a behaviour?

Does the machine behave the same way all the time? What different behaviours does it exhibit?

What information is being represented?

Exploration: Machine Q

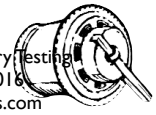
Framework:

Manipulate parameters, observe behaviours and responses.

Limits help to see connections and irrelevances

What makes sense considering *value to a user*?

What makes sense considering *other demands*?

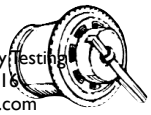


Discussion

Which limits simplify behaviour?

Which make it complex?

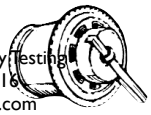
What surprising behaviours have you seen?



Exploration: Machine H

Framework:

Making a Map – Base camp approach



Framework: Map making

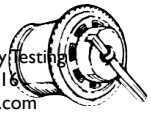
Making a map makes exploration simpler

Basecamp approach

Identify a base, identify a 'step'

From the base, take one step in all directions.

Return after each step



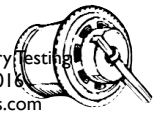
How are you making a map?

Types of notation

- physical map (directions / positions mean something)
- connection map (directions mean nothing)
- heirarchy / mindmap
- table

"all" directions - or just the interesting ones?

Granularity and detail



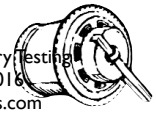
Beyond trivial

Map Interactions

Map Characteristics

Disfunction leads to an understanding of function

Complete mapping can be attractive, but not always valuable



Mapping

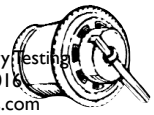
Map-making

Systematic activity

Requires – and is driven by – notation

Basecamp / Zero-state is one starting point

Progress to heuristics, theories and models



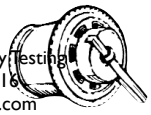
Discussion: Systematic Exploration

What makes a systematic approach?

Approaches that work for you?

Other possible approaches? Analogies?

What makes a good approach? A bad one?



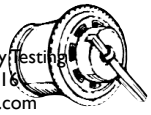
Exploratory Frameworks

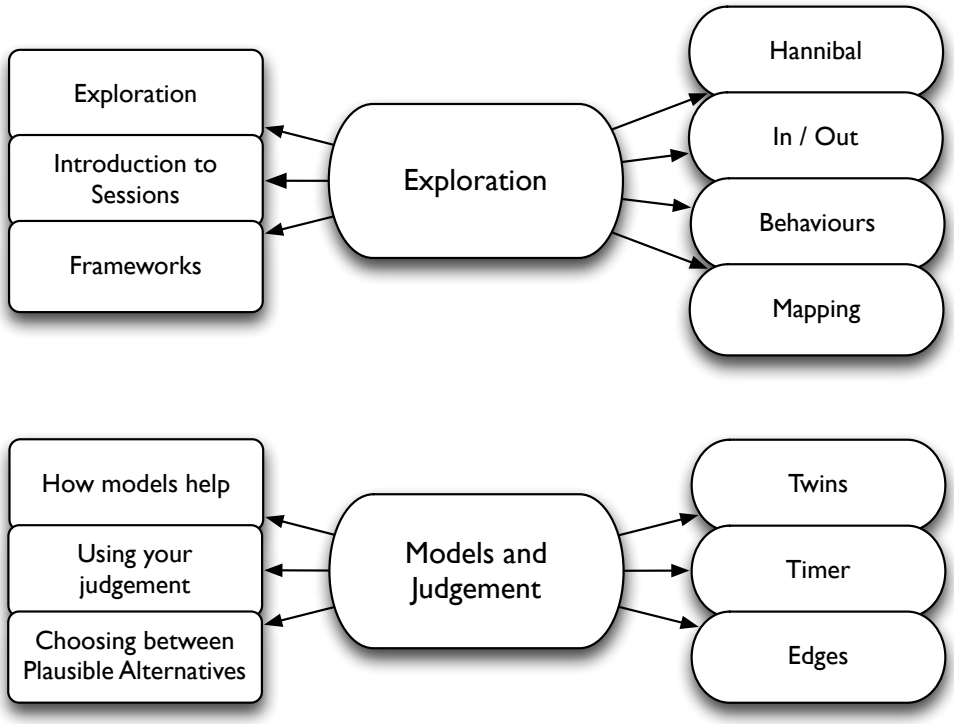
Exploration for information

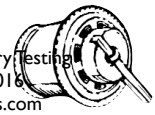
Can be communicated, checked

Basis for further exploration

Towards a model of the system

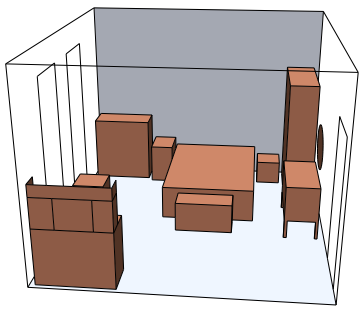


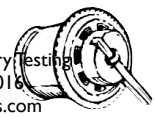




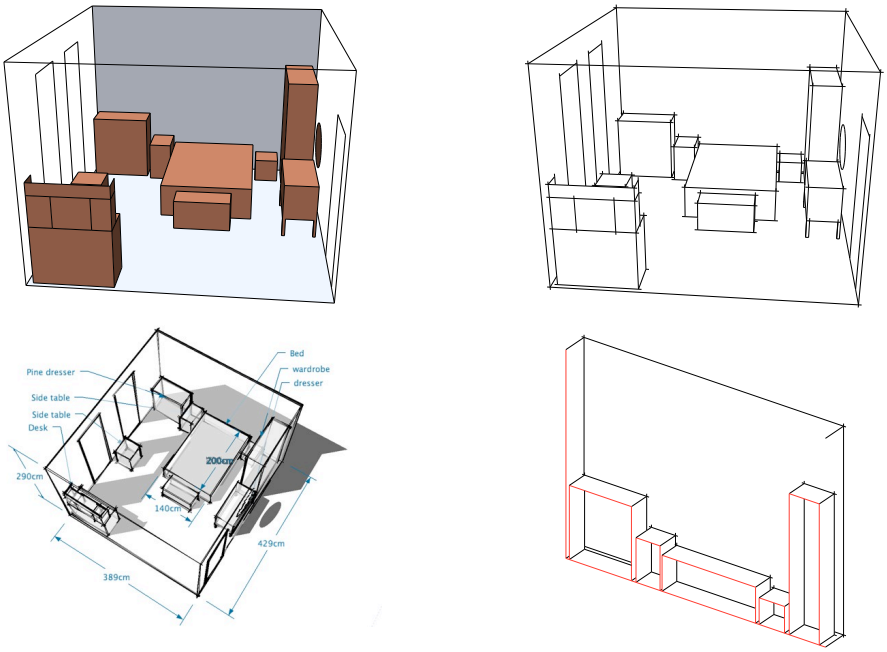
Models

```
#VRML V2.0 utf8
Shape {
appearance DEF COL_Layer0 Appearance {
material Material { diffuseColor 1 0.145 0.157
}
}
geometry IndexedLineSet {
coord Coordinate {
point [ -169.063 0 0,
-169.063 -153 0,
]
}
coordIndex [ 0 1 -1]
}
}
Shape {
appearance USE COL_Layer0
geometry IndexedLineSet {
coord Coordinate {
point [ -0 -153 0,
0 0 0,
]
}
coordIndex [ 0 1 -1]
}
}
Shape {
appearance USE COL_Layer0
geometry IndexedLineSet {
coord Coordinate {
point [ 0 0 114.173,
-169.063 0 114.173,
]
}
coordIndex [ 0 1 -1]
}
}
Shape {
appearance USE COL_Layer0
geometry IndexedLineSet {
coord Coordinate {
point [ -0 -153 114.173,
0 0 114.173,
]
}
coordIndex [ 0 1 -1]
}
}
Shape {
appearance USE COL_Layer0
geometry IndexedLineSet {
coord Coordinate {
point [ -169.063 -153 114.173,
-0 -153 114.173,
]
}
coordIndex [ 0 1 -1]
}
}
Shape {
appearance USE COL_Layer0
geometry IndexedLineSet {
coord Coordinate {
point [ -169.063 0 114.173,
-169.063 -153 114.173,
]
}
coordIndex [ 0 1 -1]
}
}
Shape {
appearance USE COL_Layer0
geometry IndexedLineSet {
coord Coordinate {
point [ -169.063 0 0,
-169.063 0 114.173,
]
}
coordIndex [ 0 1 -1]
}
}
Shape {
appearance USE COL_Layer0
geometry IndexedLineSet {
coord Coordinate {
point [ -0 -153 0,
-0 -153 114.173,
]
}
coordIndex [ 0 1 -1]
}
}
Shape {
appearance USE COL_Layer0
geometry IndexedLineSet {
coord Coordinate {
point [ -169.063 -153 0,
-169.063 -153 114.173,
]
}
coordIndex [ 0 1 -1]
}
}
Shape {
appearance USE COL_Layer0
geometry IndexedFaceSet {
solid FALSE
coord Coordinate {
point [
-169.063 0 0,
-169.063 -153 0,
-169.063 -153 114.173,
-169.063 0 114.173,
]
}
coordIndex [
0
1
2
3
-1
]
}
}
Shape {
appearance USE COL_Layer0
geometry IndexedLineSet {
coord Coordinate {
point [ -13.901 -0 0,
-45.578 -0 0,
]
}
```

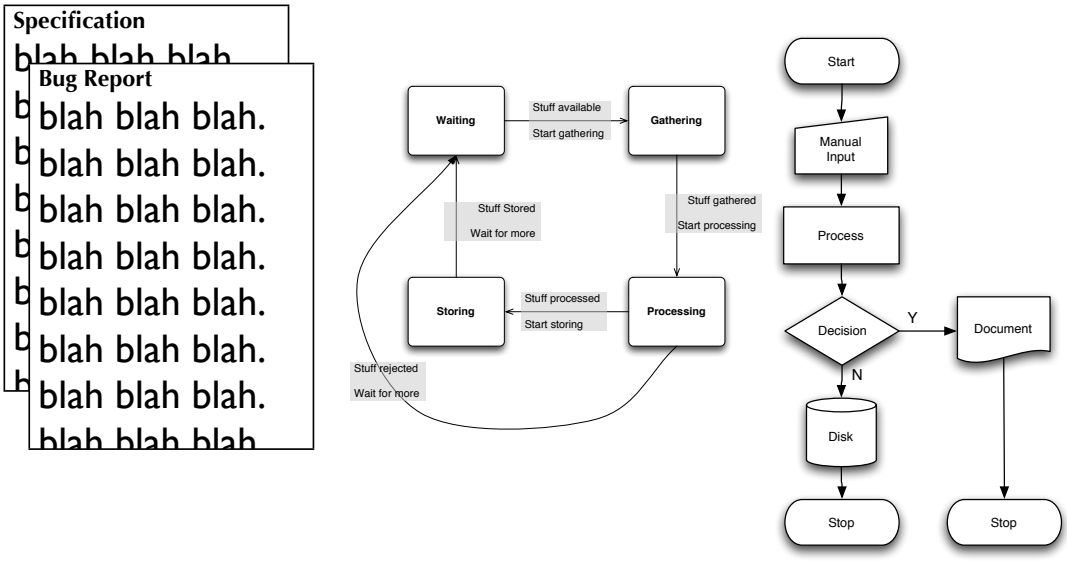




Graphical models



Common models of software



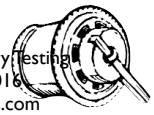
Modelling

Key analytical skill

Many different modelling approaches

Testers verify / refute model vs *reality*

Models of success vs models of failure

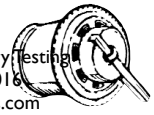


We have already built models

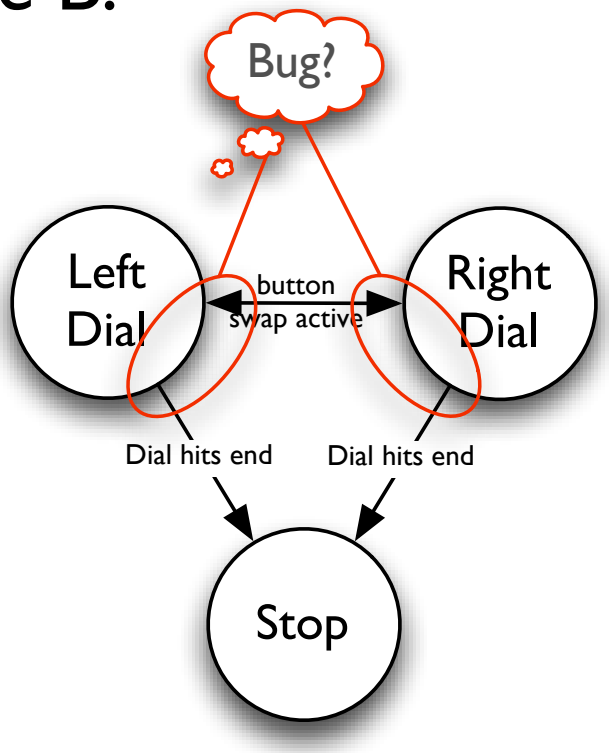
Transformation of input into output

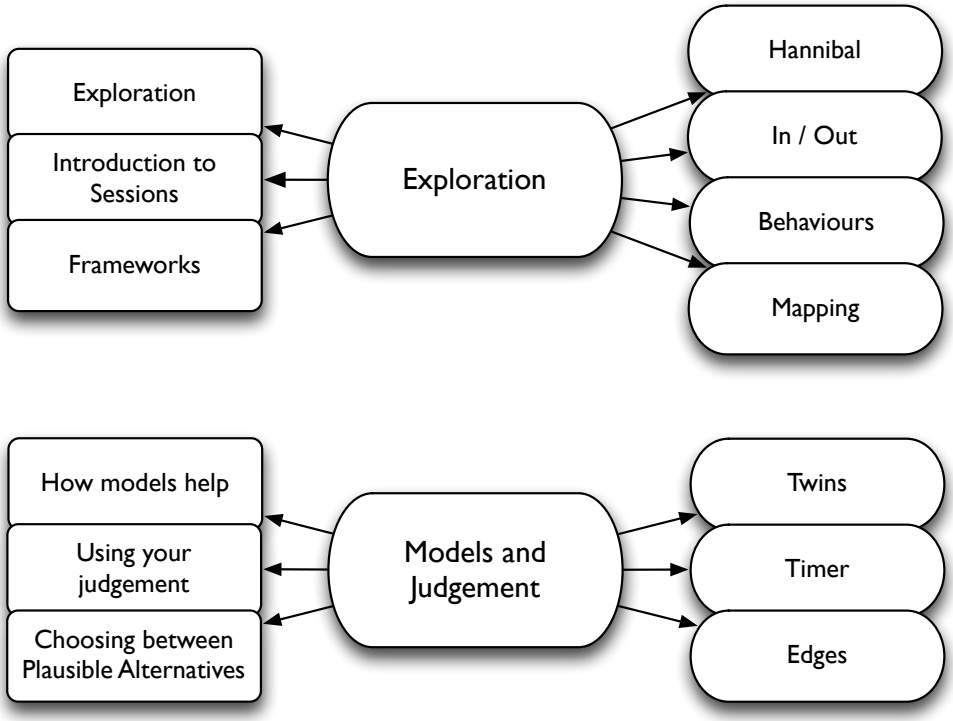
State transitions

Maps



Machine B: States





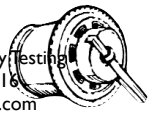
Hmmm...

Testing reveals information

... so does exploring.

Is all exploring testing?

Do you care?



Judgement: Found vs Expected

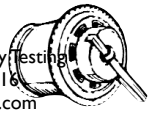
Judging bugs involves a comparison -

real *vs* modelled

Return to *Hannibal* results

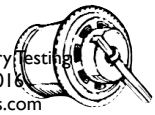
What types of problems were seen by the group?

How did we know they were problems?

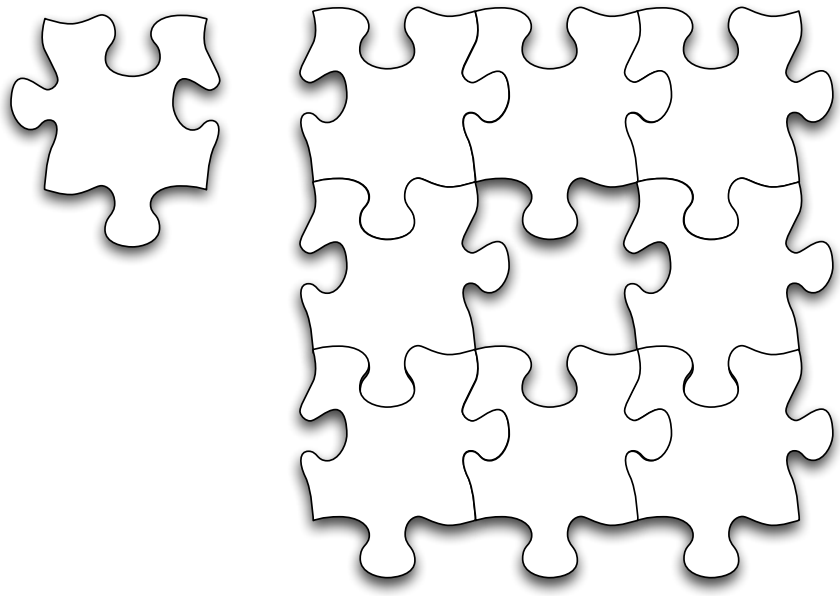


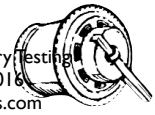
Models: Classification and judgement

- Inconsistencies
  - Internal
  - External - against specific source
  - External - against cultural expectation
- Absences
- Extras

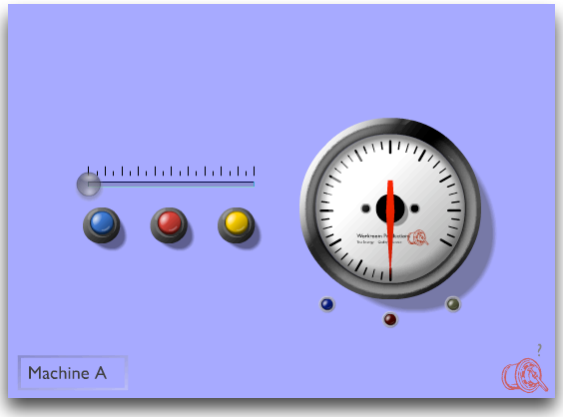


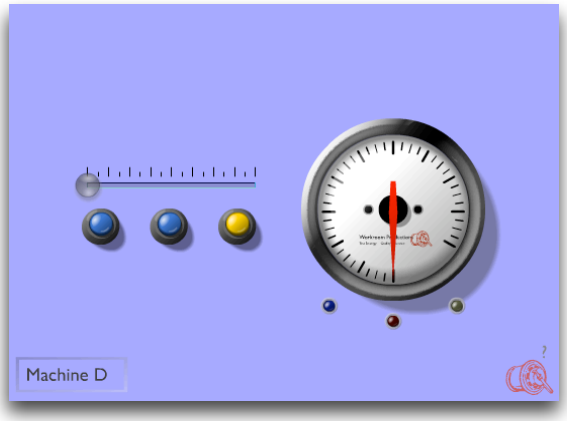
Internal Inconsistency

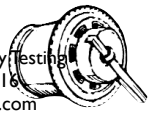




Machine A vs Machine D



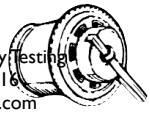




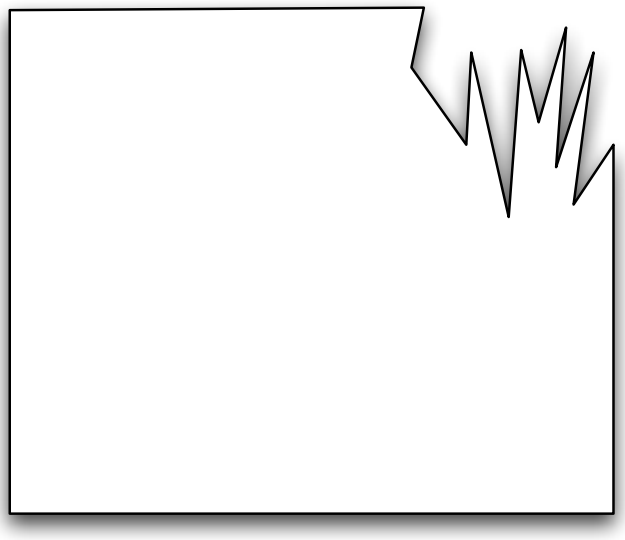
Internal inconsistency

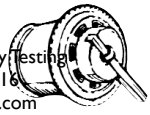
Observation of differences – existing techniques

Judge differences – what basis?



External models: Inconsistent with expectations





Exercise:
Machine C: Potential failures

It's a countdown timer.

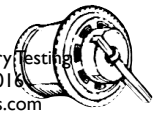
What do you expect from a timer?

Text

What do you expect from a timer in use?

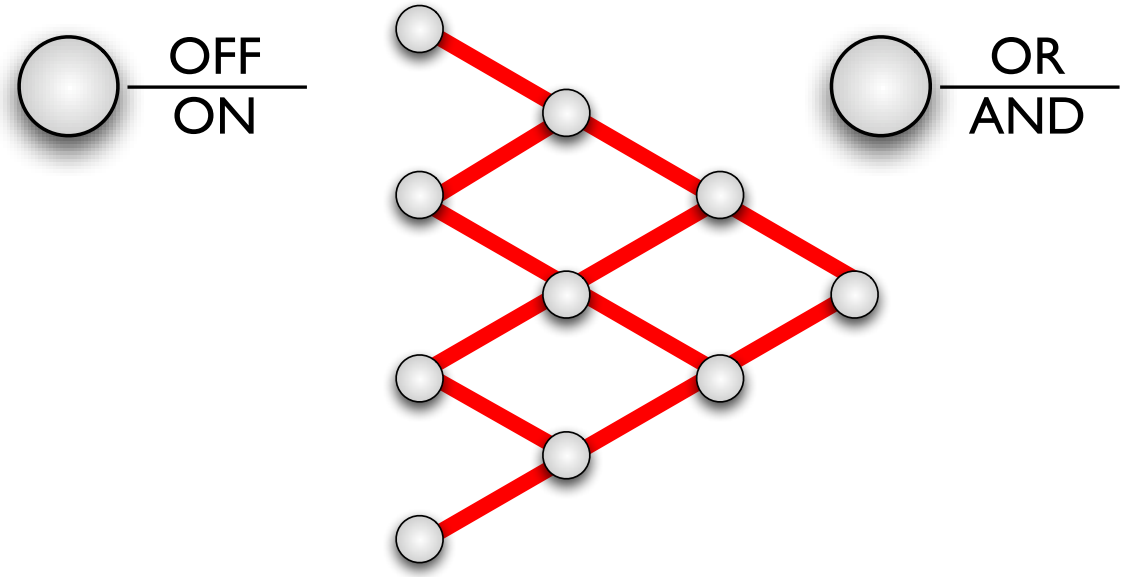
What do you expect from a countdown timer?

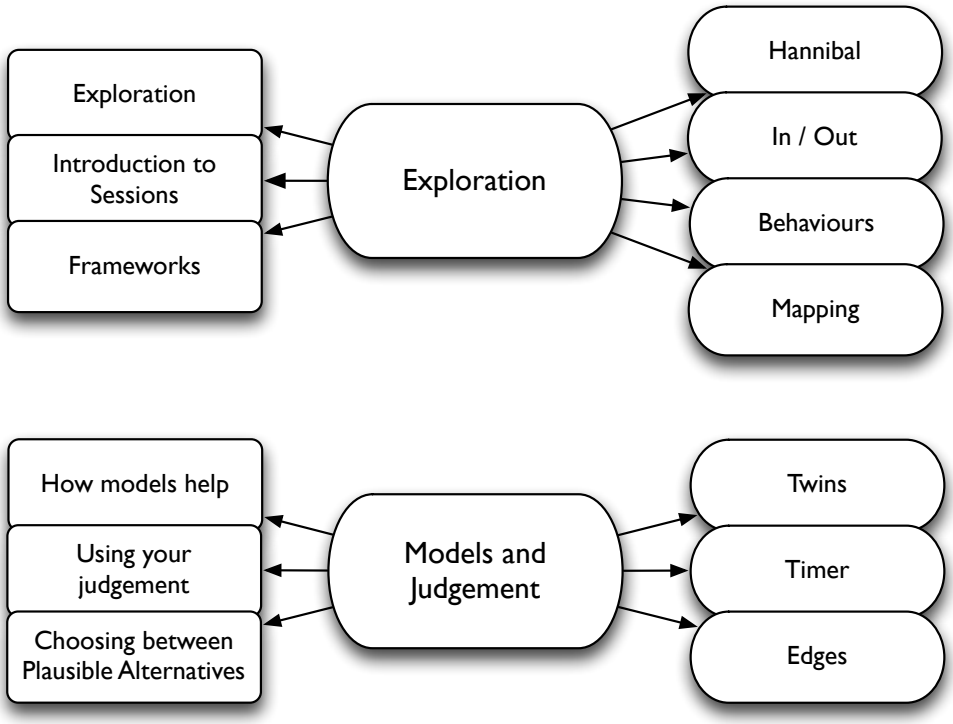
What could go wrong? From your list, what's worst?



External models: Inconsistent with specification

Machine F: Specification

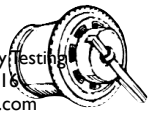




Boundaries and ranges

Black-box or white-box?

What if you had to *find* the boundaries?



Binary search through a range

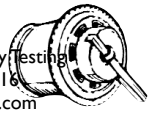
Contiguous range - discrete behaviours

Find two values with different behaviours

Pick the *middle* and discover its behaviour

New knowledge! Boundary above or below?

So what's the next test? Iterate . . .



Machine E

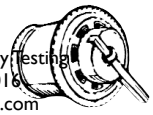
Age validation:

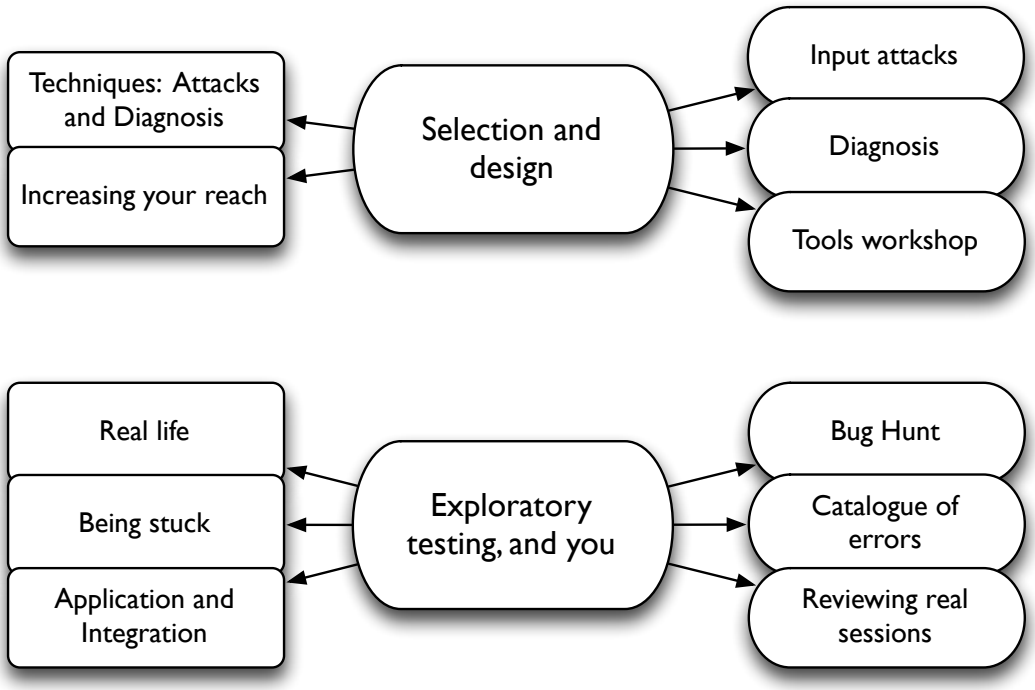
What range / ranges of input?

What output?

Any bugs?

Explore functionality - you could use In / Out





Great potential in attacking

Error-handling

boundaries - functional and technical, input and output

'allow everything except' validation

novel state while handling an error

Alternate interfaces to functionality

Different 'level' from target

Tool-assist can be vital

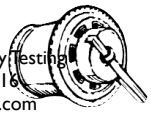
Input attacks

Fault model

- Mistakes at boundaries. What boundaries?
- Field length. What fields?
- Validation. What is not valid?

Binary expansion, bulk inputs

- The most information with the least effort



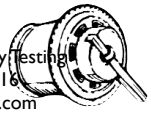
Machine E

On the attack!

You've tested the functionality

- Range between 18-70: green
- Too young: red .... Too old: yellow
- Reject input: blue

You know that this is buggy - find more bugs!



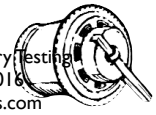
More vocabulary

Testing

Checking

Critiquing

Supporting

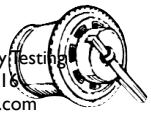


Searching for Surprises
Telling the Truth

Feedback – swift, surprising, *truthful*

Will you attack, expose, exploit?

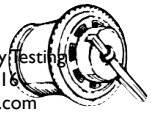
Will you extend, reveal, persuade?



Colour Robot

This software is important to me.

How could I improve it?



Exploration, testing, checking

Compare the exploration and verification environments

What freedoms does the exploration environment have?

What does the exploration environment miss? What can you try right now?

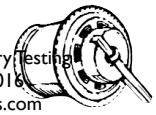
Diagnosis

Select inputs to reveal model

design and record multiple tests

spot patterns in results

bug present / bug absent?



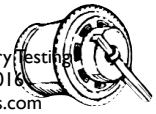
Diagnosis

Anatomy – organs, physics, purposes
Architecture – elements, purposes

Physiology – how it all works, interactions
System – relationships, feedback, overall behaviour

Pathology – what can go wrong
Problems – what can go wrong

Psychology – influence of the mind
People – influence of designers, users, hackers



Machine M

5 different bugs

Each has a bug report

Refine the reports...

Machine J

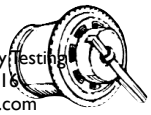
Tax - 10%

Delivery - 5

Report:

> 123 items at 456.78 each. VAT reported is 5618.64. Should be 5618.39

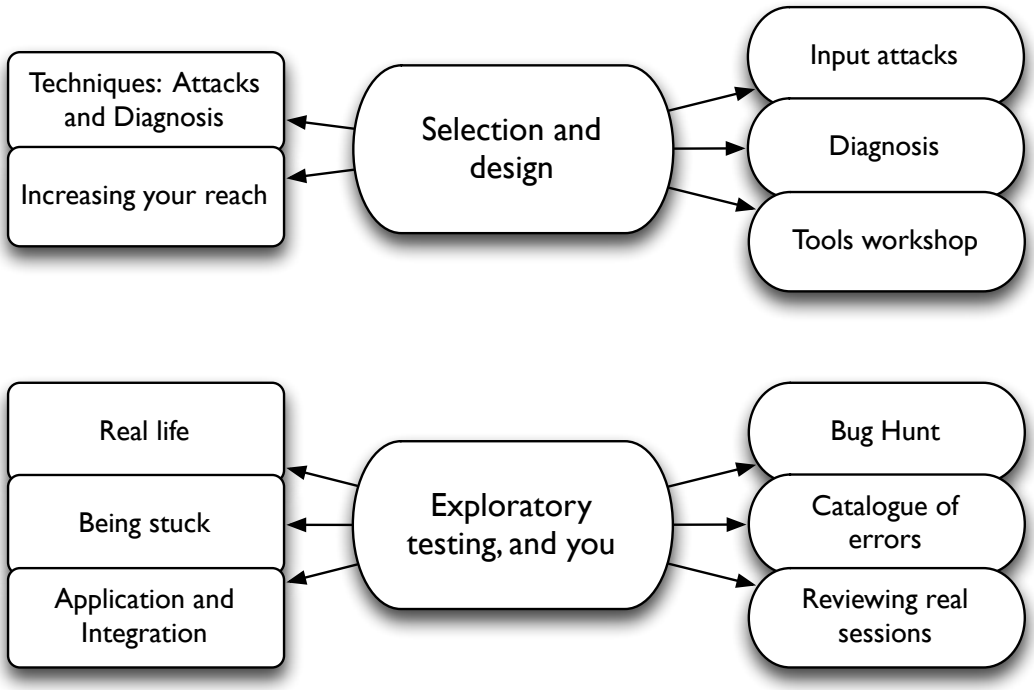
You know that this is buggy - find a plausible model



Machine K

Each button shows a *different* bug

with the *same* inputs.



What makes a good test?

Control?

Opportunity?

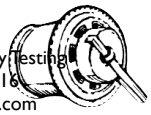
Ease of diagnosis?

Finding a problem, or showing value?

Test Selection and Design

Continuous and ongoing task

Tester's responsibility - not just test manager's



Test Selection

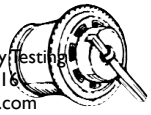
Continuous and ongoing task

Tester's responsibility - not just test manager's

Information about *risk* is not the same as information about *value*

Normal use may not apply

Diagnosis may be a separate step



Test Design

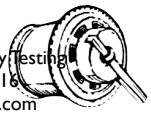
Continuous and ongoing task

Tester's responsibility - not just test manager's

Manual-only, function-driven ET is *weak and slow*

Effectiveness - more value per test

Efficiency - more tests for the effort



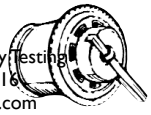
Increasing your reach

Bugs are easy to see - if you know where to look!

Diverse measures

Attacking techniques

- Bug-focussed, look for exploitation rather than diagnosis
- Fault model
- Requires + delivers understanding of underlying technology

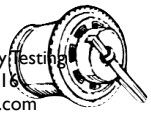


Diverse measures

Making it easy for bugs to find you!

- Variety of environments, paths, data
- Emulation and fault recording
- Real use - scenarios, personas, internal beta test

Do you need to go all the way?

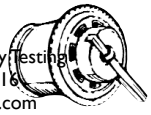


Random testing

Most ‘random’ tests are constrained

Recognise constraints, patterns

Recognise where *you* introduce a system



Increasing your reach with tools

Observation tools

database contents, environments, match/compare

object browser, text stripper, mapper, validator

screen capture, keylogger, transaction logs, system logs

Action / setup tools

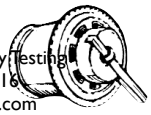
test harness, replay tool, macro tool, data loader

Analysis tools

code analysis and verification

unit tests, coverage

data analysis, patterns and clusters, graphing

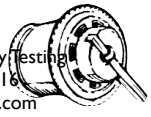


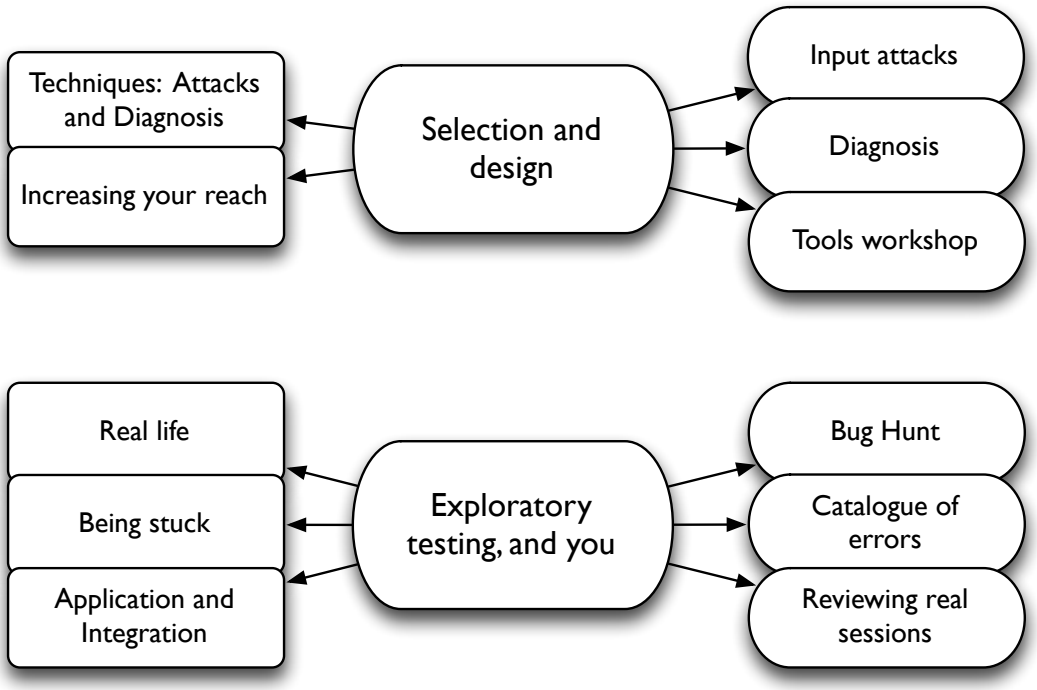
Tools within your organisation

Put capture/replay to one side

What else do you have for this work?

What could be useful? Can it be introduced?





Managing Exploration

What other industries manage exploration?
How do they do it?

Manage a gamble by . . .

improving the odds

reducing the costs

Sylvain Lenfle : Learn and adapt

Bug hunt!

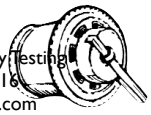
Raw / DataGenerator

Area and target:

Explorer(s):

Approach:

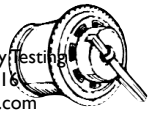
Duration:



Present your conclusions
!*Thr*ee minutes!

What information did you produce?

Did you find bugs?



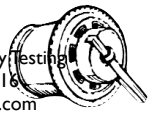
Different approaches

Different people find different things

Frameworks / discipline

What was well-suited? Poorly suited?

What would work better under different circumstances?



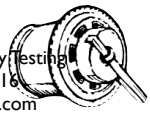
Skip

Session-based Testing and Charters

Purpose, method, scope

(limits, goals, target pathologies, approaches)

- Existing bugs
- Known attacks + suspected pathologies
- Demonstrations
- Questions



Skip

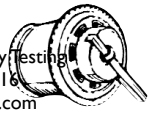
Catalogue of Errors

Bug list is a vital resource - especially closed

- Risk management and mitigation
- Learning

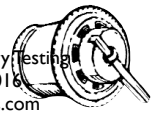
Reveal causes of error

Principle of discovery - how would you look?



Recent bugs in *your* job:

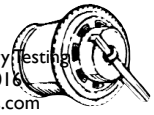
Fault	Root cause	Principle of discovery	New target



Principles of discovery

How did those around you find problems?

What lessons can you learn?



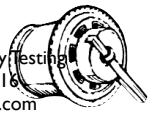
Fault model for our chosen target

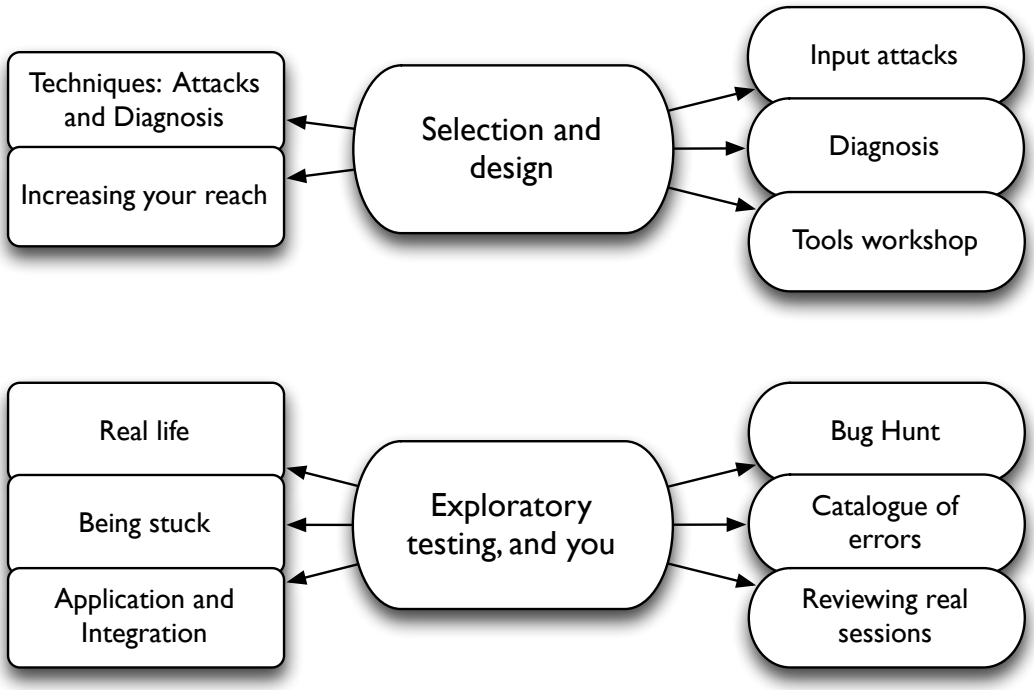
Boundaries and validation - what levels?

OS tools to hinder input attacks? Output? Interfaces?

Vulnerabilities, interactions and exploitations?

Tool assist?





Workshop: Sessions and Feedback

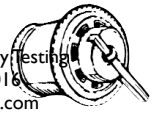
Stuck for new ideas?

Stuck or finished?

Where are your limits?

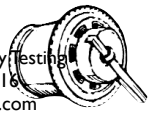
Using heuristics

Personal styles



Ooh ... a squirrel!

Getting distracted



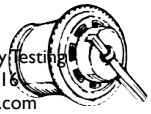
Wicked problems

Not understood until solved

No stopping rule

Symptoms and problems are hard to distinguish

No exhaustively describable set of solutions or permissible operators

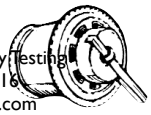


Application of ET

Discovery of risk, emergent behaviours, diagnosis, low bug rate, smoke tests

¿Too many environments, too many tests, too much to check?

¿Not enough documentation, not enough time, no existing scripts?

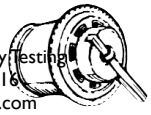


Management and Support of ET

Strategic goals remain

Difference in how they are achieved

Trade-off time and effort



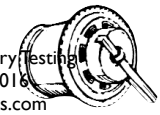
Working in an exploratory team

Fast feedback

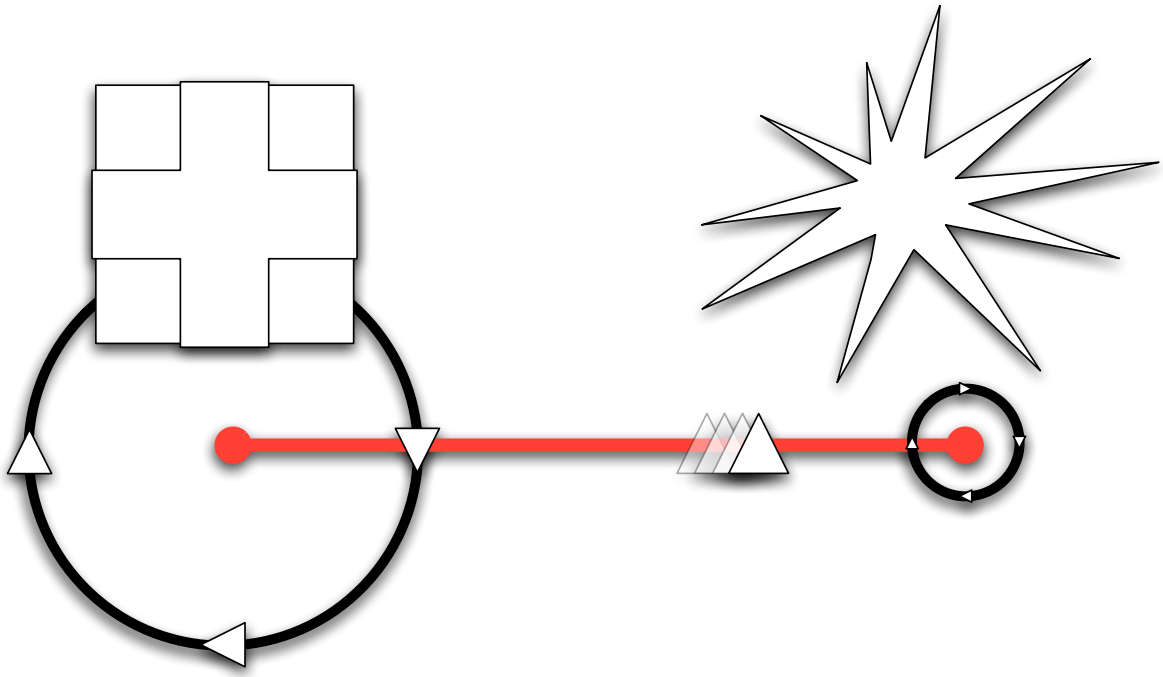
- Share information and techniques
- Peer review, pairs, bug-hunts, daily meetings
- Real-time metrics

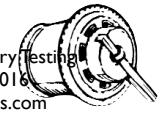
Change and learning

Empowered testers

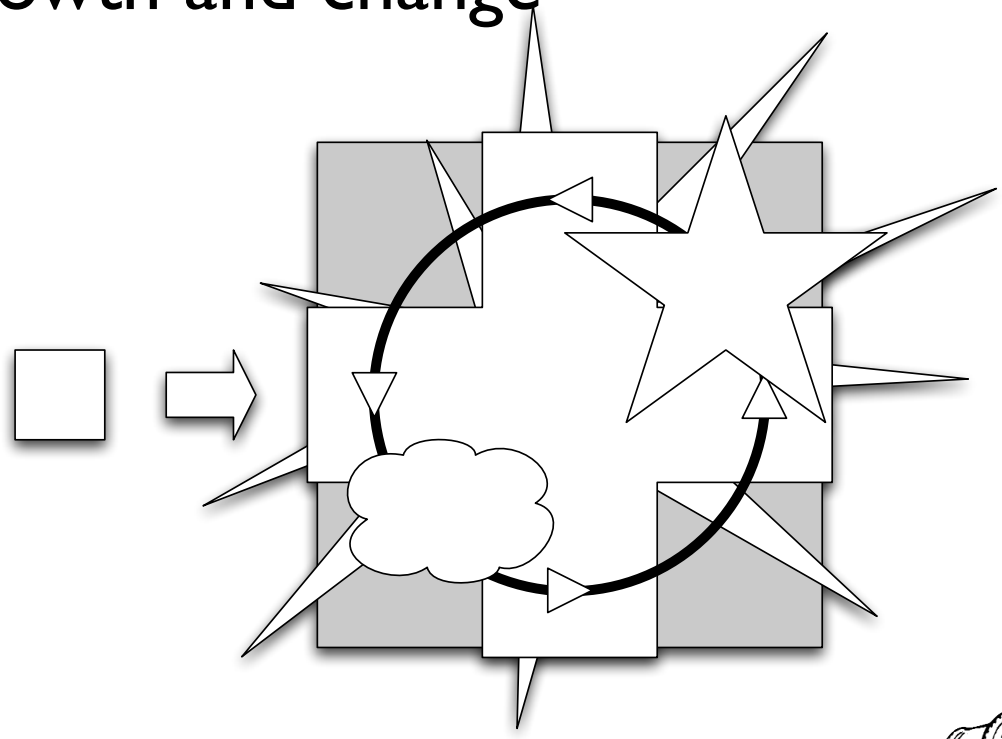


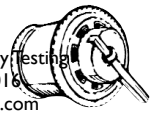
Metrics and fast feedback



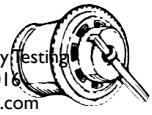


Growth and change

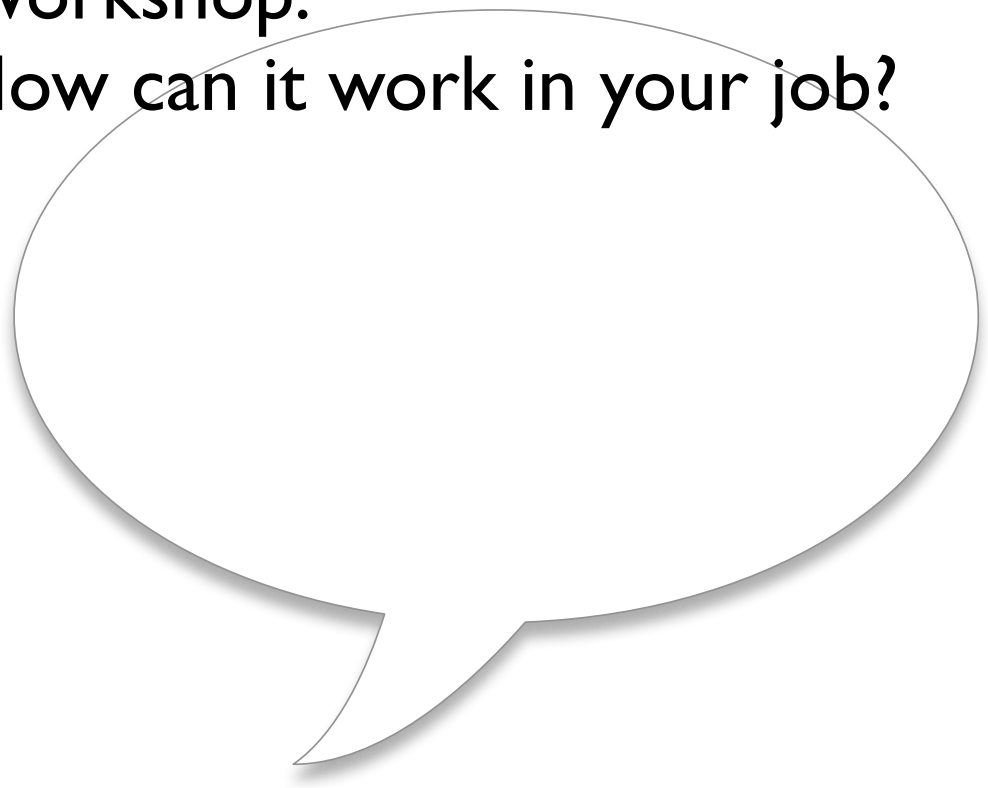


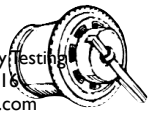


Process cannot stand alone



Workshop:
How can it work in your job?





Got a Grip?

James Lyndsay

jdl@workroom-productions.com

www.workroom-productions.com