An Introduction to Exploratory Testing

James Lyndsay
Workroom Productions

When exploring, you interact with something in order to find out more about it.

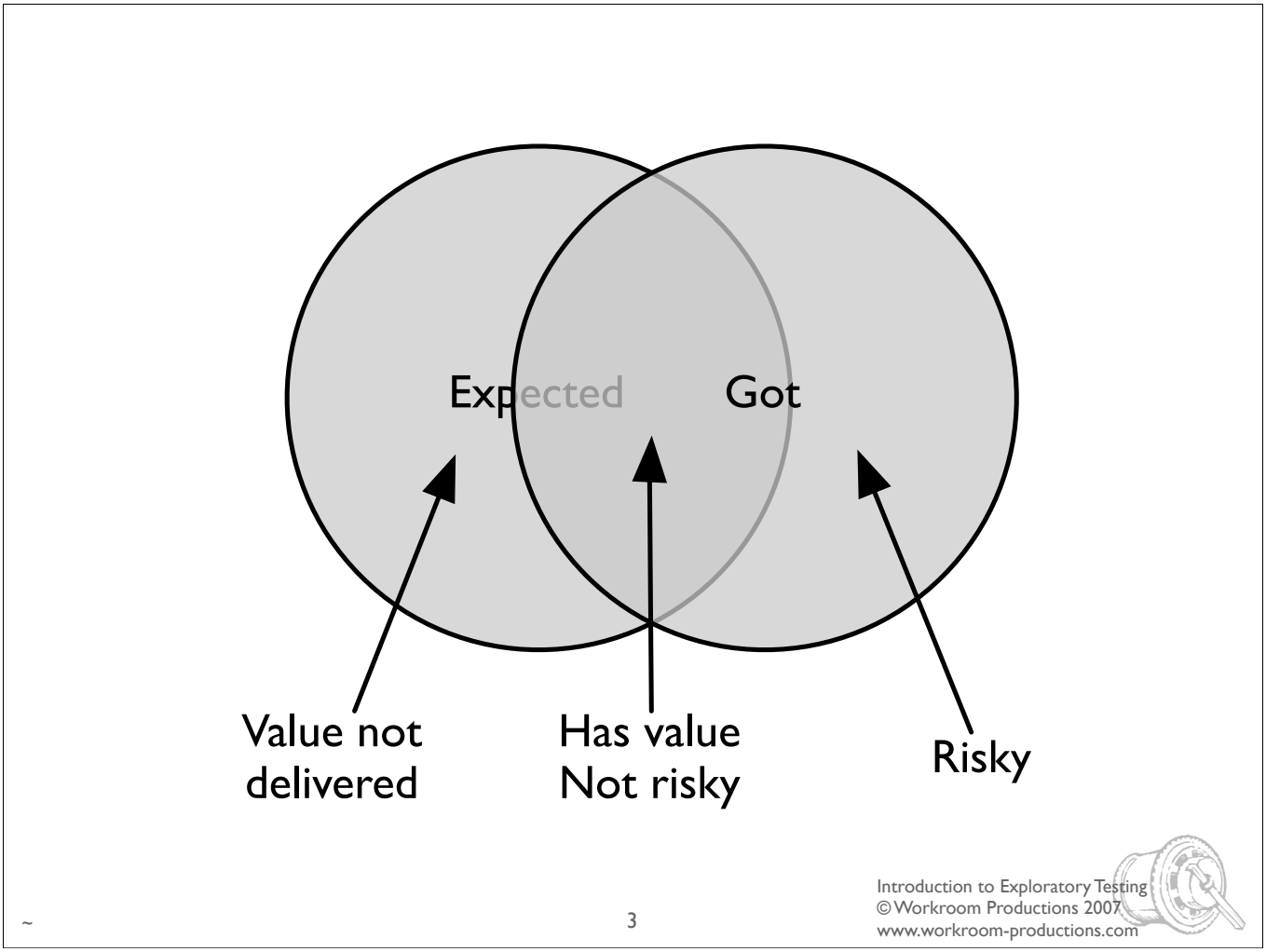
Your improved understanding is a model of the thing you have explored - you can use it to summarise the explored object in some way, to predict its behaviour, to support claims, to allow use, and so on. Exploration is about model-building.

Exploration requires something to explore. When that something is tangible - a city, a phone, some software - exploration is correspondingly physical and observational. The real world gives feedback. When something is intangible - an information space, an idea - your actions may be more internal. It is important to be aware of when one is exploring a real thing, and when one is exploring the model you have made. The real will always have the potential to upset the model.

The active explorer will take actions with observations in mind, but will make observations that lie outside narrow causal expectations. Exploration is given focus by a goal - the model you are building. The model gives shape to the actions you take, and the observations you make. Some observations may change your model, causing you to take different actions, make different observations. The real world may intrude on your safe modelling - you, the explorer, are not necessarily in control of everything that happens - and it is important to be open to observations that challenge your model.

Exploration and experimentation are closely linked; good experimental technique allows you to arrive at a clearer model.

Exploration allows reliable discovery of information
Methods have a start point, a process
Methods may change based on circumstances



The diagram splits the world into four regions.

The overlap describes the things we expect that are are also fulfilled by the deliverable. Whether we are testing based on requirements, or testing based on the deliverable, we can identify that our requirements are met by our deliverable. The region outside both circles is all the stuff we didn't want, and haven't got. It's an infinite set - and so cannot act as an source from which to derive tests.

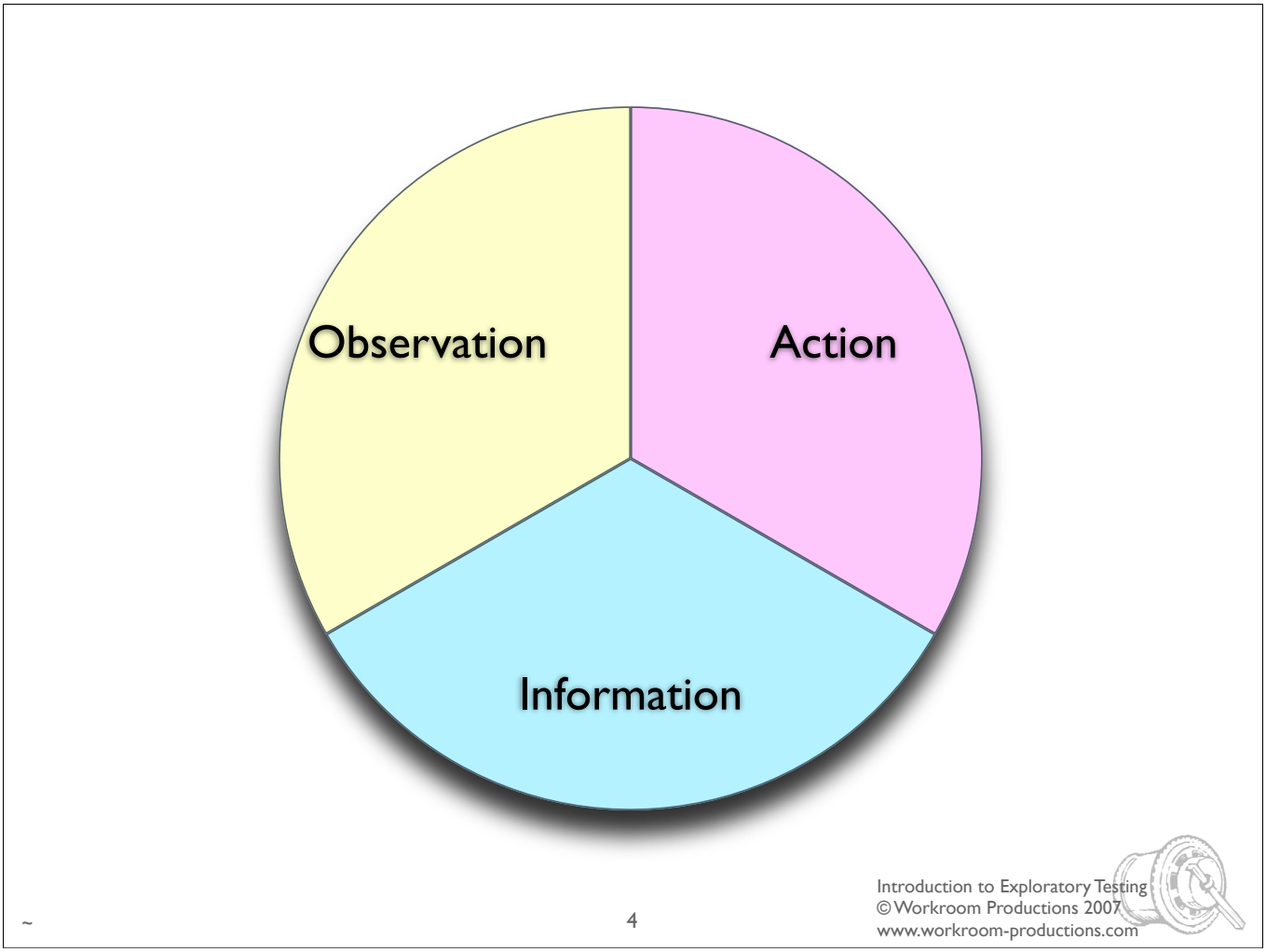
The left-hand arc is what we expected to get, but didn't. Result:The deliverable is less valuable than we'd hoped.

Now the right-hand arc. "We found the system does *this*, which is a bit of a surprise". Things we find here are risky – until we've done enough to allow us to exercise our judgement, we don't know if this is valuable, or problematic.

Scanning through the finite set of things we expect can be done before we ever have something to test. It relies on a good understanding of our expectations. If we're going to get the job off the critical path, we need that understanding to be stable. Then we can write down the steps in testing, before we ever test. When we do the test, we'll know about what's there, and what's not. We'll know about value. This is **scripted** testing.

Scanning through the finite set of what we've got requires, as its first step, something to scan. That means we're on the critical path, no way off. We'll need to be fast, and to be fast, we'll have to be prepared, and skilled. We'll find risks, and risks need to be assessed and addressed, so we'll need to be really good, really soon. This is **exploratory** testing.

We ***need*** both, for a good understanding of risk and value.



Action: things to do. Things that are done. Things that have been done. Often, one considers actions that the tester takes - but it also important to consider actions that the system takes, that a part of the system takes, or that are taken by something outside the system.

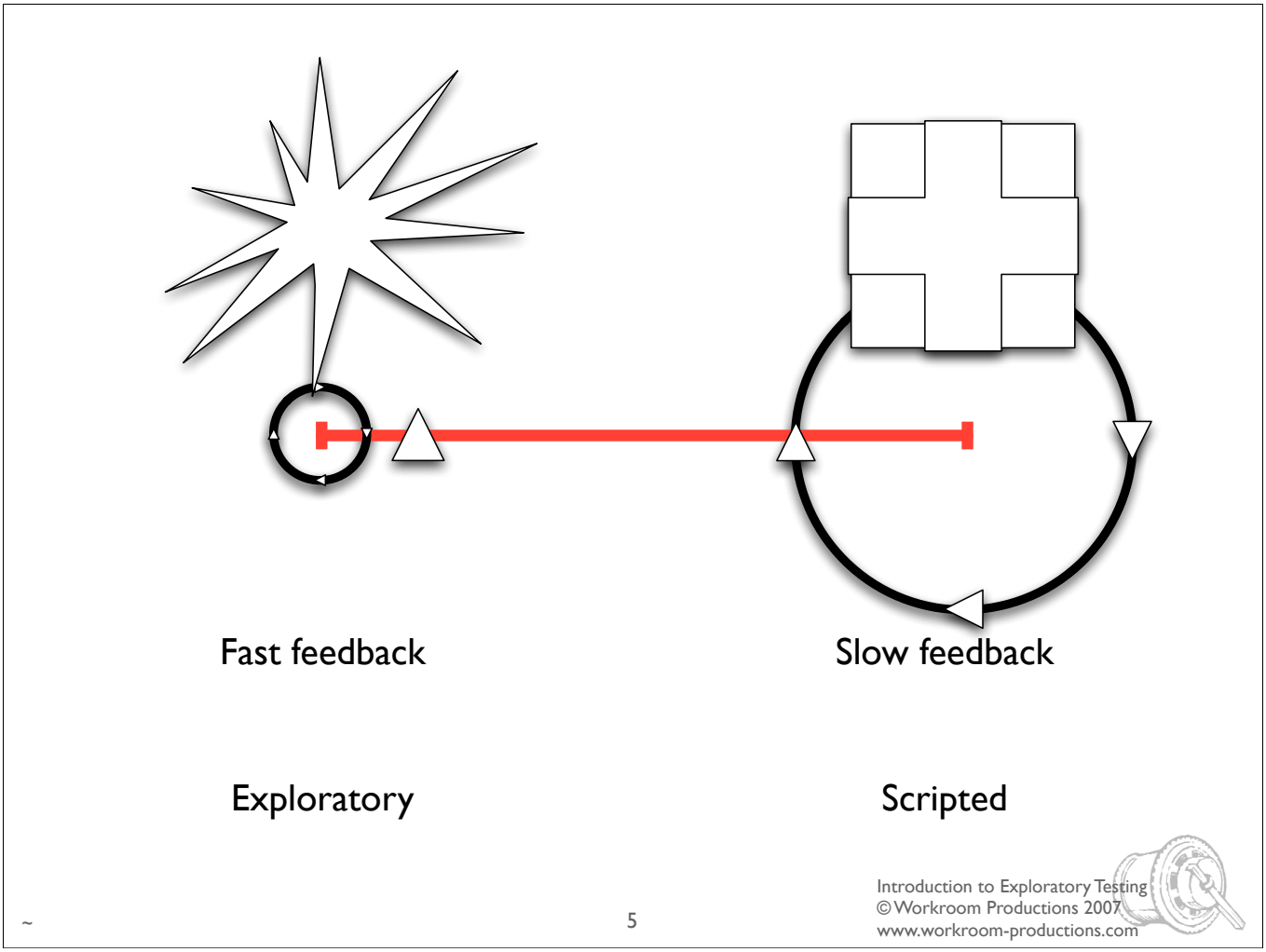
Information: things that are. Things already in the system, the state that system is in. Not just the stuff the tester chooses to give to the system. I could call it data - and it often is, but the label is over-loaded and limiting. Actions, with the right information, may trigger events. However, not all events are recognised.

Observation: things that are perceived. Whatever tool captures information after an action - ones eyes, a database monitor, the smoke detector connected to the office sprinkler system - that information has to be *perceived* to be part of the test.

It's easy to script **actions**. Many written scripts dictate action rather precisely. **Information** is harder to write down completely; most scripts leave at least some information to the tester - or to chance. Scripts can suggest what to **observe**, but there's simply no way to describe all potentially useful observations in advance.

A three-ring-binder scripted test leaves important elements of the test to the tester. Even if the **actions** are set in stone, the **information** may change - by choice or by circumstance - and the tester can broaden their **observations** if they choose.

Exploration is an important part of the skilled manual execution of scripted tests: The tester influences the test design during the test by choosing what to observe, how to treat the information, be the actions ever so precisely prescribed.



Exploratory testing is characterised by fast feedback - exploration is an interactive activity.
Scripted testing is characterised by long feedback - a long-lived script may produce useful information long after its designer has moved on.

It's hard to use terms like **ad-hoc** and **systematic**, or **informal** and **formal**, without ranking them somehow. They are loaded terms, rather than well-defined terms. They have a loose association* with other ideas - but it's often possible to find counter-examples.

Neither Exploratory testing nor Scripted testing is a test *technique* as such. Rather, they are approaches, or strategies. There is no clear superiority - indeed, they are complementary. They work well together, but each has characteristic weaknesses when used alone.

Most tests are neither simply scripted, nor simply exploratory. It may help to consider a sliding scale, rather than a choice. Many tests are made of scripted and exploratory parts; it may help to recognise those occasions where a script is a vital component of an otherwise exploratory test, or when exploration is the key to deriving value from a script.

All testing can be considered in these terms, but some test techniques - for example, fuzz testing as detailed in www.cs.wisc.edu/~bart/fuzz/fuzz.html - have vital characteristics that lie outside this model.

*While the following loose associations reflect some aspects of current practice, they should be treated with caution.Some of these associations are ill-conceived.

Fast-feedback approaches: *Agile methods, emergent behaviours, context-driven school, software attacks, risk-based testing, scant documentation, poor requirements, little time, skilled manual testing.*
Long-feedback approaches: *"Best practices", white-box techniques, waterfall, requirements-based testing, safety-critical, software engineering, CMM, standards and methodologies, automated testing.*

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

The term 'Exploratory testing' was first put in print in 1988 (Kaner, Falk, Nguyen; *Testing Computer Software*), although it had been in colloquial use since 1983. The definition has evolved - this has been stable since 2003.

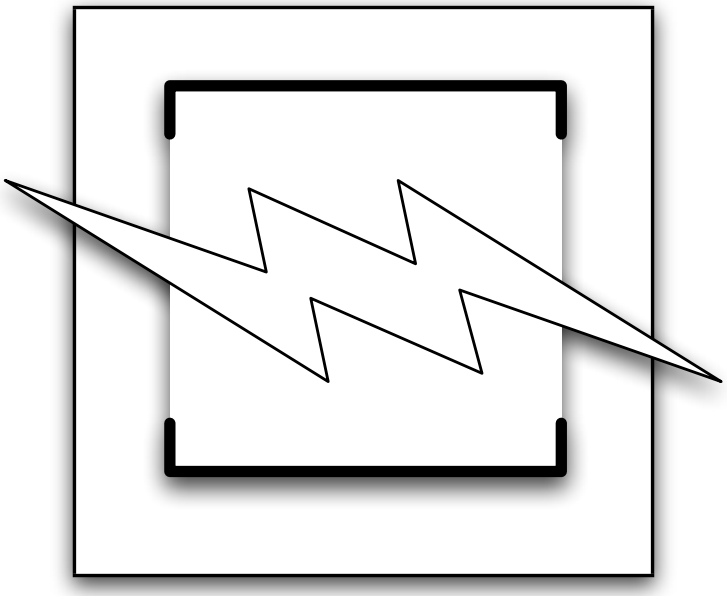
The defined term arose not only from describing and shaping a working practice, but also from the need to provide viable opposition to approaches that spent vast effort on creating and maintaining scripts without producing valuable information. It was initially described as 'brain-engaged testing'.

The authors of this definition - Kaner and Bach - emphasise learning as the key component of this definition.

The definition above is not in clear opposition to scripted testing - but neither learning, nor simultaneous activity can be part of a rigid pre-scripted test.

As defined above, Exploratory Testing is accepted as a core component of testing. However, techniques and academic support for exploratory testing lag behind the older, and easier to set out, approaches of scripted testing.

Discipline



Testing is a discipline - and disciplined activity is key to exploratory testing. The disciplines of exploratory testing differ from those of scripted testing.

Key disciplines of exploratory testing:
Limiting current scope - recognising distractions and using them later
Recording activity and results - to clear the mind, as an audit trail, to enable sharing
Conscious skills improvement - to allow a range of interactions and observations, and to provide the groundwork for judgement of risk

As a tester, track:
Record: Actions, decisions, observations
Track: Time, Scope, Bug rate

As a test manager:
Review work
Give appropriate responsibility and decision-making to testers

The management technique *Session-based testing* provides an important framework for this discipline.

Uses

?

ET is good for revealing the *unexpected*; risks, emergent behaviours, technological issues, design flaws. ET allows a project to identify surprises earlier than they might be found with rigid scripts, beta testing, or normal use.

ET gives broad, fast feedback on a product. Broad feedback allows the project team to lean about the emergent properties of their product, and the consequences of their decisions. Fast feedback allows those effects to be linked more easily to their causes, and gives a greater opportunity for the project to do something about the problems found.

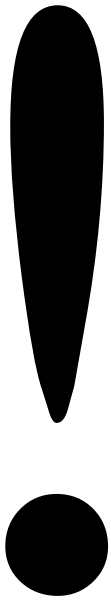
Exploratory Testing is used for:

- Risk assessment - and assessment of risk model
- Diagnosis / reproduction of observed problem
- Exploitation of known problem
- Smoke-testing early releases
- Coping with reduced / excessive / irrelevant / obsolete documentation (because learning is central)
- Coping with reduced time (because feedback is fast)

Some approaches:

- Bug-busting day
- SWAT team
- Scouting/mapping team
- Exploratory pairs
- Just before check-in
- Reported beta/live bugs
- Manual regression testing

Warnings

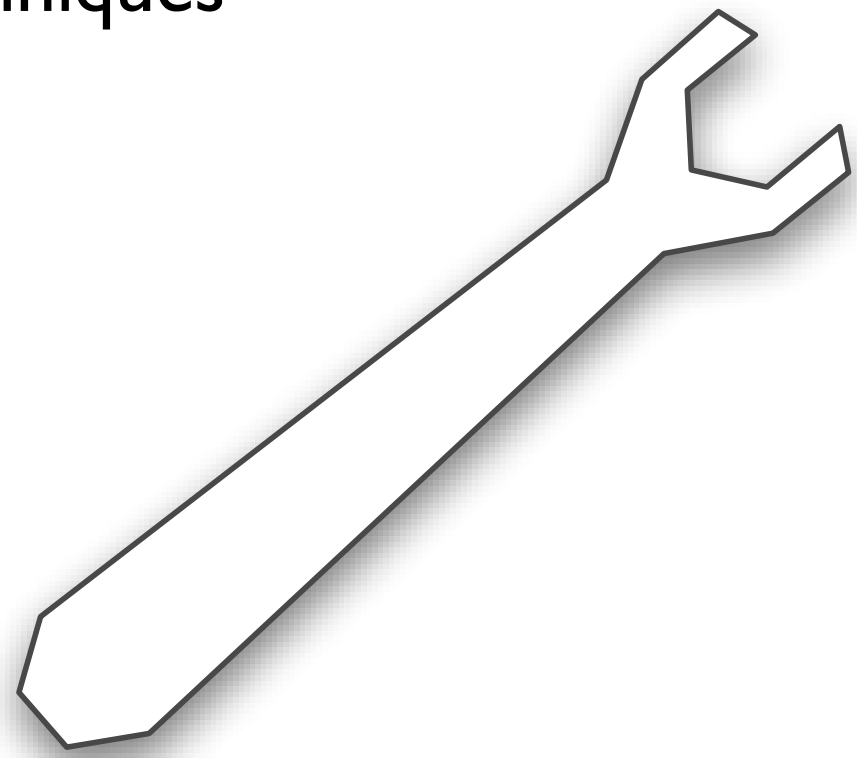


ET places learning at the centre of information gathering. This changing perspective can allow a project to develop a broad understanding of the product and its uses, beyond the necessarily restricted vision of the requirements and design. However, this perspective can itself be a challenge to a project, and to individuals on the project.

Exploratory testing can be hard to measure, manage, and integrate – particularly when introduced into a project that expects to measure and manage it in the same way as scripted testing. Business environments that seek to control change may be less able to successfully integrate resist ET than those which seek to adapt to change.

Exploration becomes much more effective with good tool support. ET is 'brain-engaged' (Kaner), rather than a simply manual approach.

Techniques



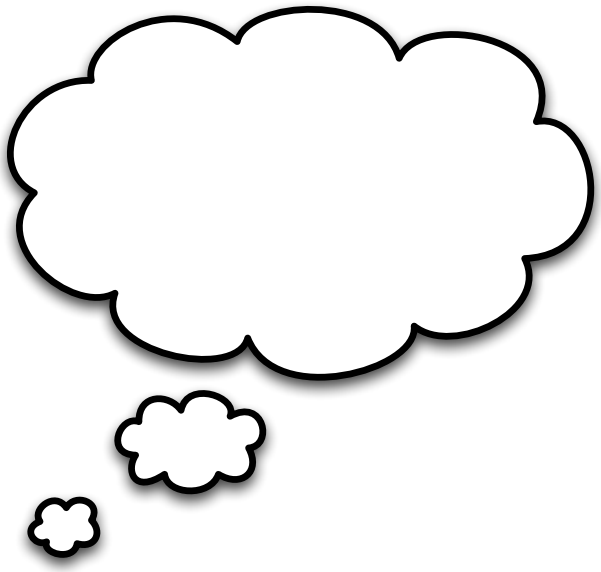
Exploratory testing is not, in itself, a test technique. Rather, it is a collection of test techniques, bought together with the common goal of finding information out about an existing artefact.

Many testers use just one exploratory technique. Examples include: attacking with long strings, mapping navigation, providing edge-case inputs. However, using a single technique means that it is hard to sustain bug rates. To broaden the range techniques available to them, testers need to recognise and adapt their style. Training in different techniques can help to establish a shared baseline and to break free of monotonic exploration.

Some techniques improve observation, some work on mapping and exploration, some improve judgement to allow problems to be recognised more easily, some focus on exploitation and attack. As with scripted testing, individual techniques often need support from tools and data.

New techniques can be derived and shared with colleagues productively.

Strategies



ET requires something to explore – but does not require descriptions of what is to be explored. This makes it particularly useful at those points in a project where an artefact needs to be assessed and understood, but where the supporting artefacts are unavailable, incomplete, or obsolete. Exploratory techniques can be used to explore a system without knowing its components, to explore working code without knowing the content of that code, to explore code without knowing the design, to explore the design without knowing the requirements. On projects where documentation is poor, the skills and techniques available to the competent exploratory tester can allow accurate assessments. However, exploration is often simpler, faster, and better where documentation is available.

If the *project* is broken, Exploratory Testing is *not* a solution.

Some useful words to help recognise where ET fits in a strategy:

- investigative
- experimental
- custom-fit
- rapid

Some useful words to help recognise where ET does not fit in a strategy:

- long-term investment
- repetitious
- unskilled

Conclusion

Exploration is a core component of testing

Produces valuable information focussed on risk

Needs discipline and tools to work well