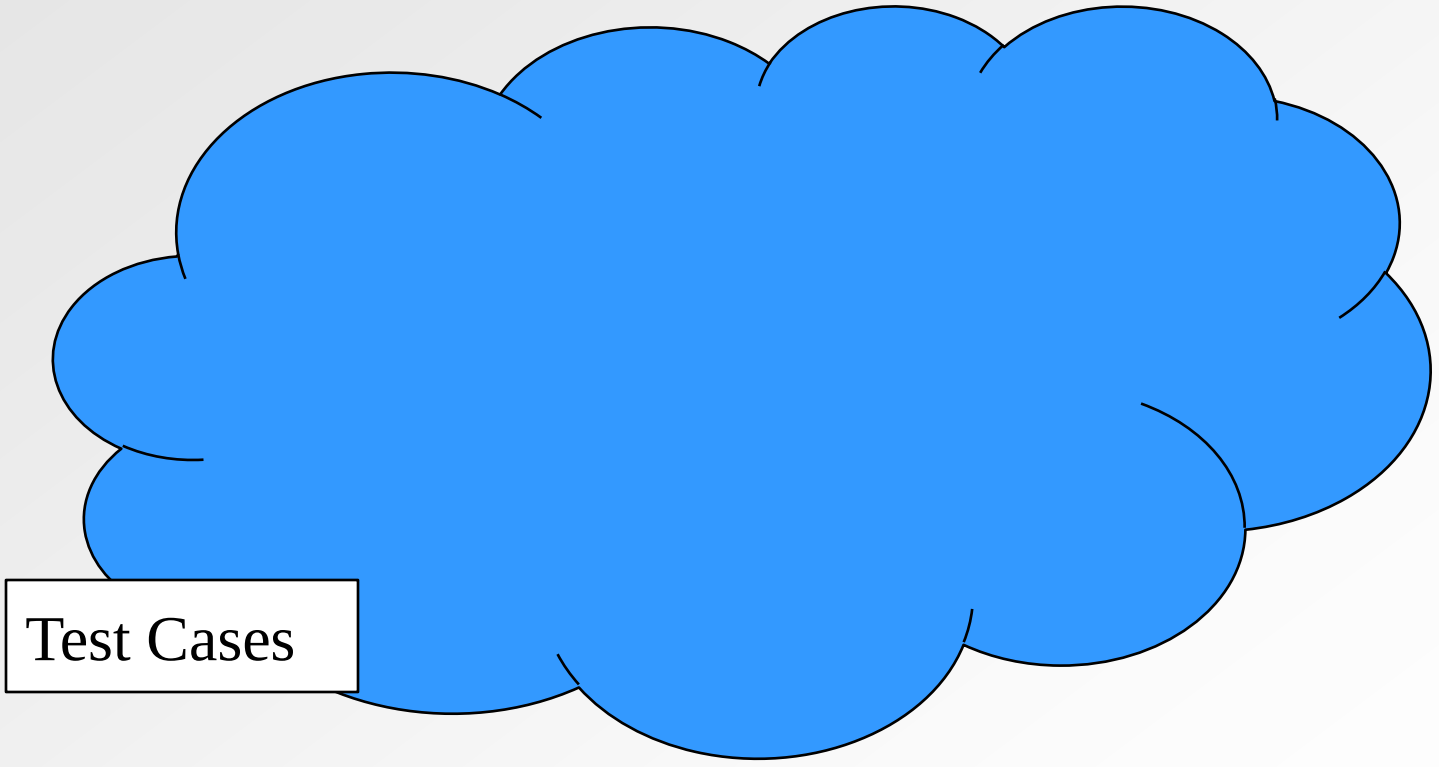
Beyond Test Cases

James Bach
Satisfice, Inc.
james@satisfice.com
www.satisfice.com

Beyond Test Cases

Beyond Test Cases



Beyond Test Cases

Skills
Models
Risks
Heuristics

Test Cases

- Test Framing
- Test Factoring
- Evaluation (oracles)
- Floating *(the skill of considering multiple incompatible ideas simultaneously, over a period of time)*
- Test Questioning, reframing, and assumptions
- Critical Thinking
- Test Notetaking
- Stopping
- Testability Advocacy
- Bug Investigation
- Toolsmithing

“Test Cases” describe only a fraction of testing.

- Programming cases?
- Driving cases?
- Traveling cases?
- Parenting cases?
- Learning cases?
- Science cases?
- Living cases?

Activities are not captured by "cases"

Excellent testing is a rich and open-ended intellectual activity.
It cannot be encapsulated into discrete procedural units.

What is a test case?

- There is a lot of confusion.
- There's no specific definition in common use.
  - *"a computer executable script"*
  - *"a sequence of discrete steps"*
  - *"specific detailed input and action steps taken during the execution with predetermined expected results"*
  - *"inputs, execution conditions, and expected results"*
  - *"a set of input values, execution preconditions, expected results and execution post-conditions, developed to cover certain test condition(s)."*
- Different testers have different ideas about them.
- Sometimes, a test itself is called a test case.

What is a test case?

- An artifact.
- A container. *(“test cases are like briefcases”)*
- A model. *(a pattern, idea, spec, or schematic for a test)*
- Part of a suite.
- A **mystery**. (*“why did he write the test that way?”*)
- A **burden to the future**. (*is it broken? is it obsolete? is it limiting my imagination or stealing my time that I could be using to TEST BETTER?*)
- A **measure of testing progress**.
- A **unit of testing itself**.

What is a test case?

- I use this definition:

A **test case** is one particular instance or variation of a test or test idea.

- Compare to:

A **test procedure** is a way of performing a test.

A **test activity** is a line of investigation that fulfills some part of the test strategy. It can encompass many test cases.

A Testing Industry Pandemic: OCD

- *Obsessive Case Disorder*: confusing test cases with testing, such that "creating test cases" is assumed to encompass all the thinking of testing.
- *Obsessive Counting Disorder*: the belief that counting test cases tells you something meaningful about testing.
- *Overspecified Cases Disease:* the irrational compulsion to treat humans like dimwitted robots by creating recklessly specific test documentation.

Unicorns!

Do you know what a Unicorn is? Okay.

Answer this question:

How many unicorns will fit into your cubicle?

Answer: At Least 15 Quadrillion

- The word “UNICORN” repeated 100,000 times, compressed, and packaged into an archive with 9 other such files, takes up 4096 bytes on my disk. *(one million unicorns)*
- About 750,000 of those files would fit on a 32gb flash drive. *(7.5 trillion unicorns)*
- 2,000 32gb flash drives fit tightly in a cubic meter *(15 quadrillion unicorns)*

That’s fifteen PETACORNS, people!

In the absence of context…

Test Case Counts Mean NOTHING!

- How much testing is 40 test cases?
- How much is 400?
- How about 40,000 test cases?

You have no idea!

You shouldn't take test case counts seriously because...

- Test cases are not independent.
- Test cases are not interchangeable.
- Test cases vary widely in value from *case to case*, *tester to tester*, *product to product*, *project to project*, *test technique to test technique*, and *over time*.
- Test case design is subjective, so counts are easy to inflate.
- Test cases do not— and can not—capture all the testing that occurs (example: bug investigation)
- Testers often don't follow the test cases, anyway.
- Automated test cases are fundamentally different from sapiently executed tests.
- Test cases represent what's *easy* to put into a test case.

Danger:
Testing should not be TOO FOCUSED.

(unless you want to miss lots of bugs)

To test a *very simple* product meticulously, *part* of a complex product meticulously, or to maximize test *integrity*...

1. Start the test from a *known* (clean) state.
2. Prefer *simple*, *deterministic* actions.
3. Trace test steps to a *specified model*.
4. Follow *established and consistent* lab procedures.
5. Make *specific* predictions, observations and records.
6. Make it *easy to reproduce* (automation helps).

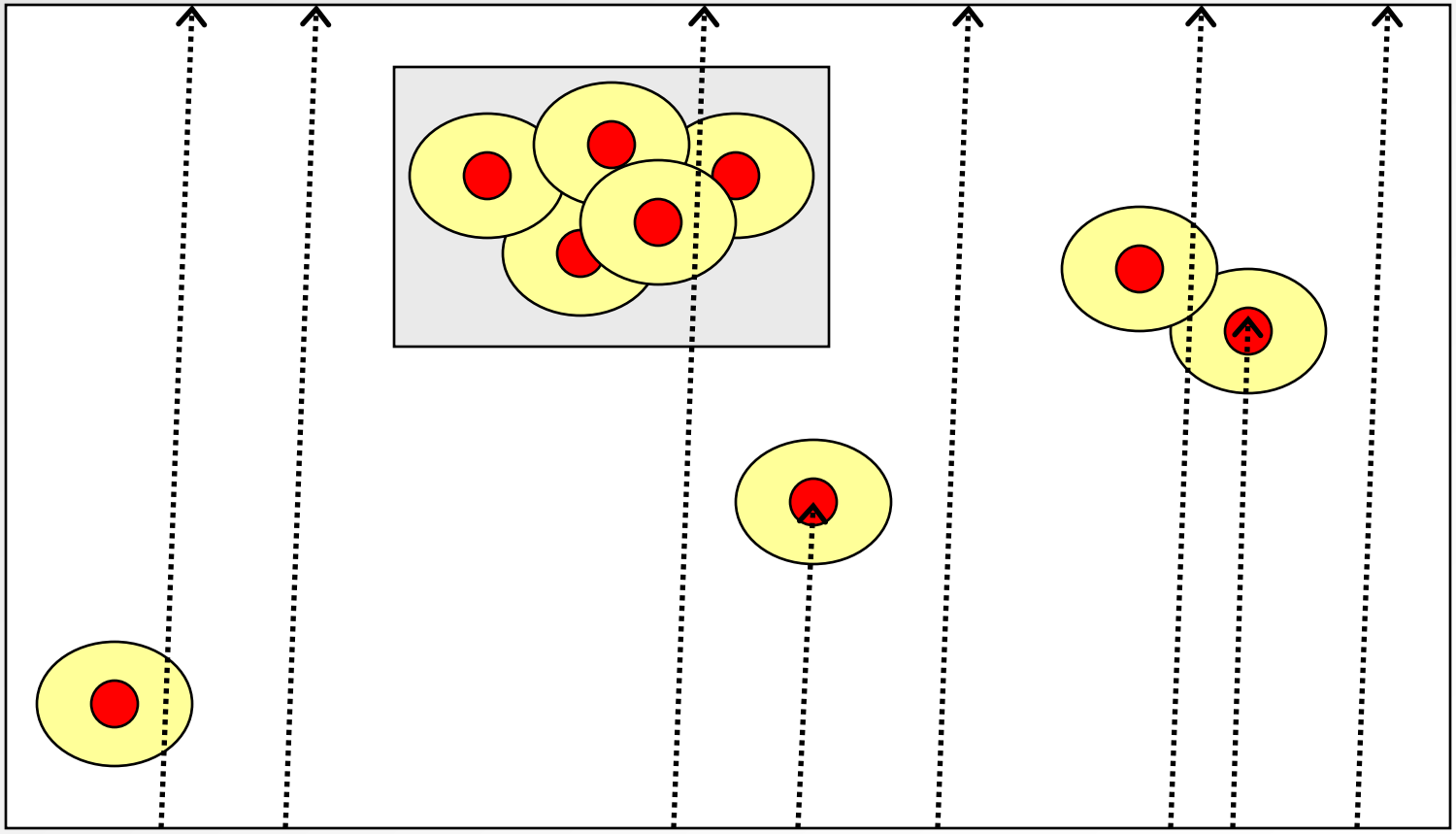
To find *unexpected problems*,
***elusive problems* that may occur in the field,**
or more problems *quickly* in a complex product…

1. Start from *different states* (not necessarily clean).
2. Prefer *complex*, *challenging* actions.
3. Generate tests from a *variety* of models.
4. *Question* your lab procedures and tools.
5. Try to *see everything* with open expectations.
6. Make the test *hard to pass*, instead of easy to reproduce.

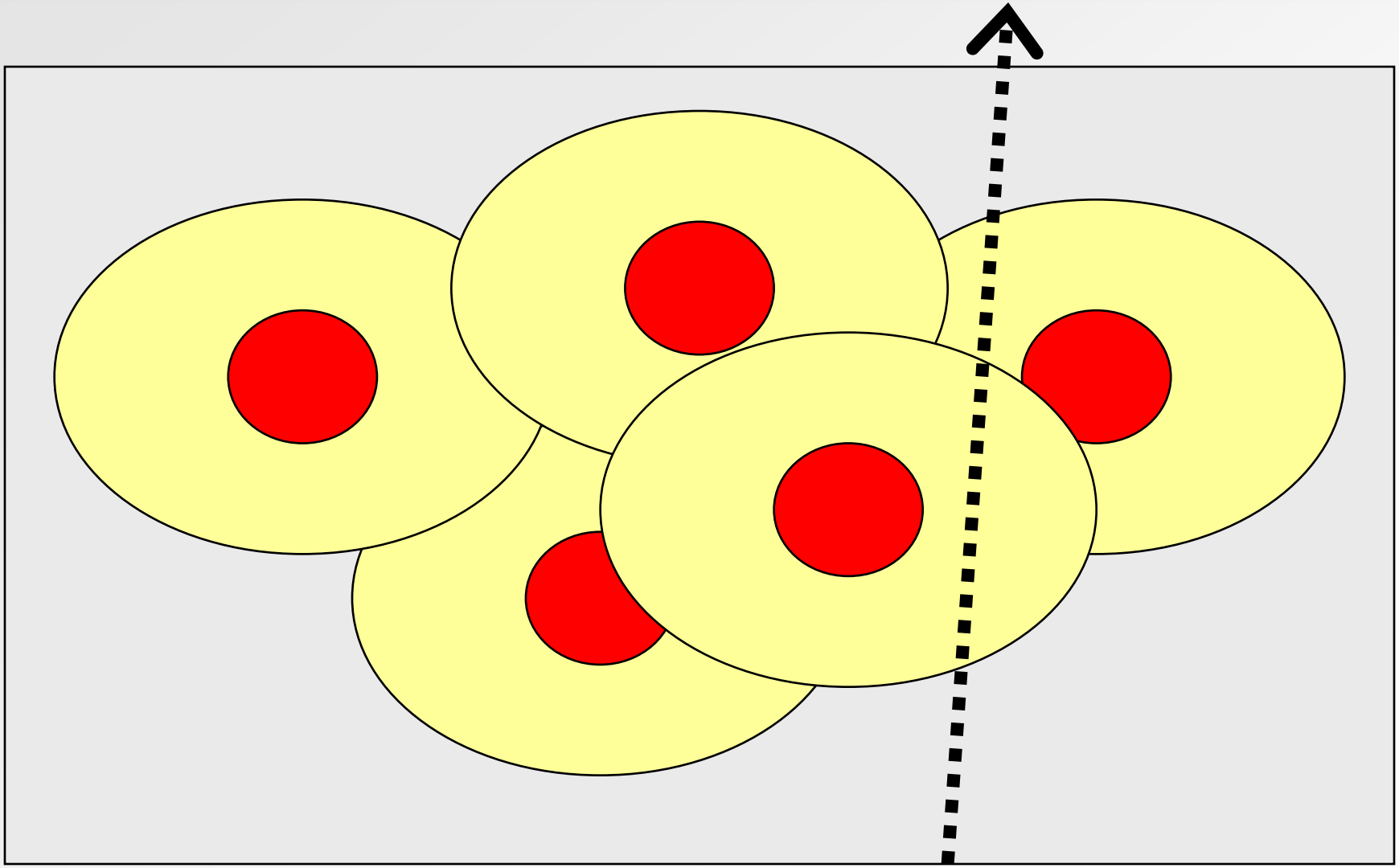
Exploiting Variation To Find More Bugs

- **Micro-behaviors:** Unreliable and distractible humans make each test a little bit new each time through.
- **Randomness:** Can protect you from unconscious bias.
- **Data Substitution:** The same actions may have dramatically different results when tried on a different database, or with different input.
- **Platform Substitution:** Supposedly equivalent platforms may not be.
- **Timing/Concurrency Variations:** The same actions may have different results depending on the time frame in which they occur and other concurrent events.
- **Scenario Variation:** The same functions may operate differently when employed in a different flow or context.
- **State Pollution:** Hidden variables of all kinds frequently exert influence in a complex system. By varying the order, magnitude, and types of actions, we may accelerate state pollution, and discover otherwise rare bugs.
- **Sensitivities and Expectations:** Different testers may be sensitive to different factors, or make different observations. The same tester may see different things at different times or when intentionally shifting focus to different things.

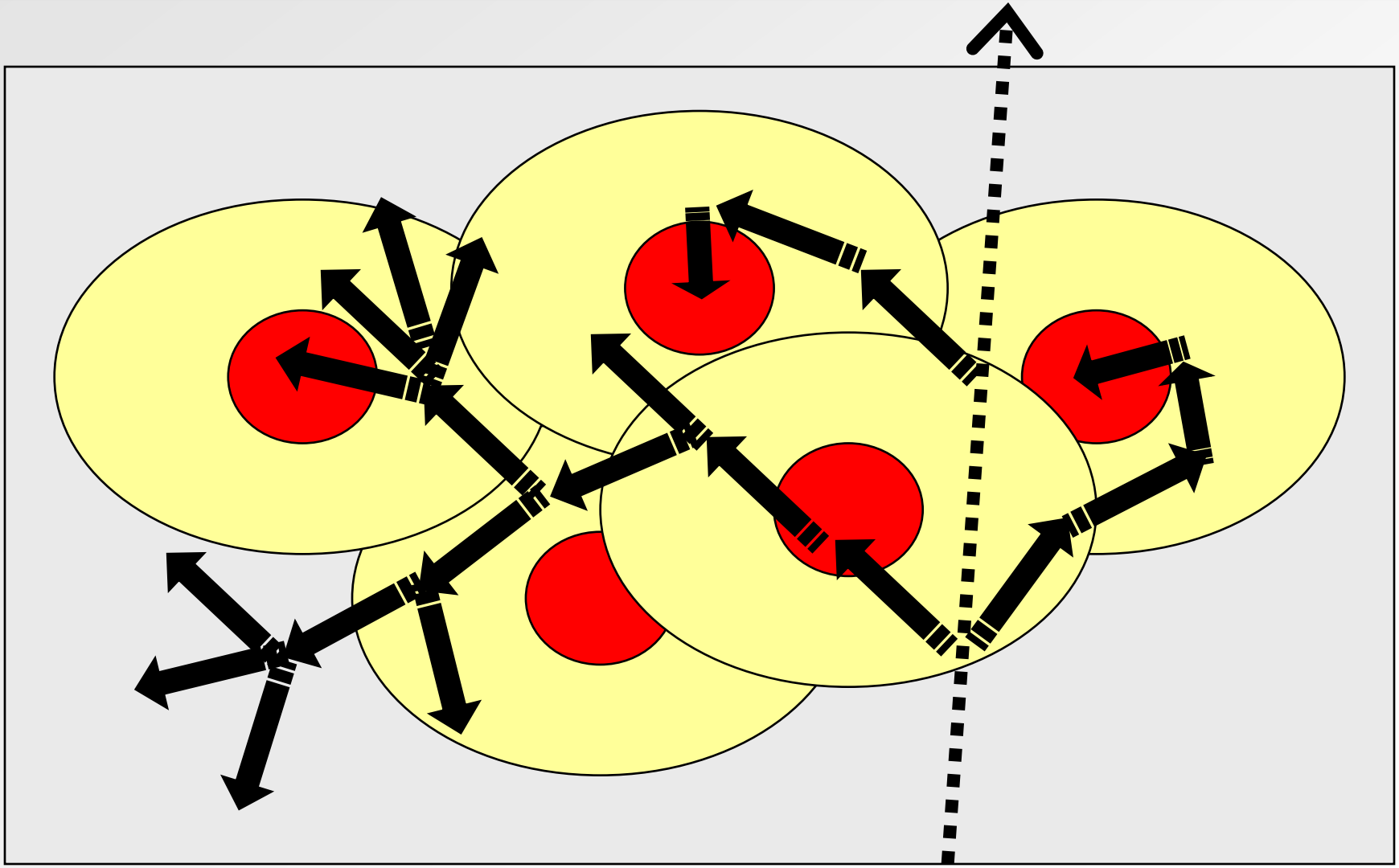
Testing is about searching an infinite space in a finite time



To do that, we must use all available information…



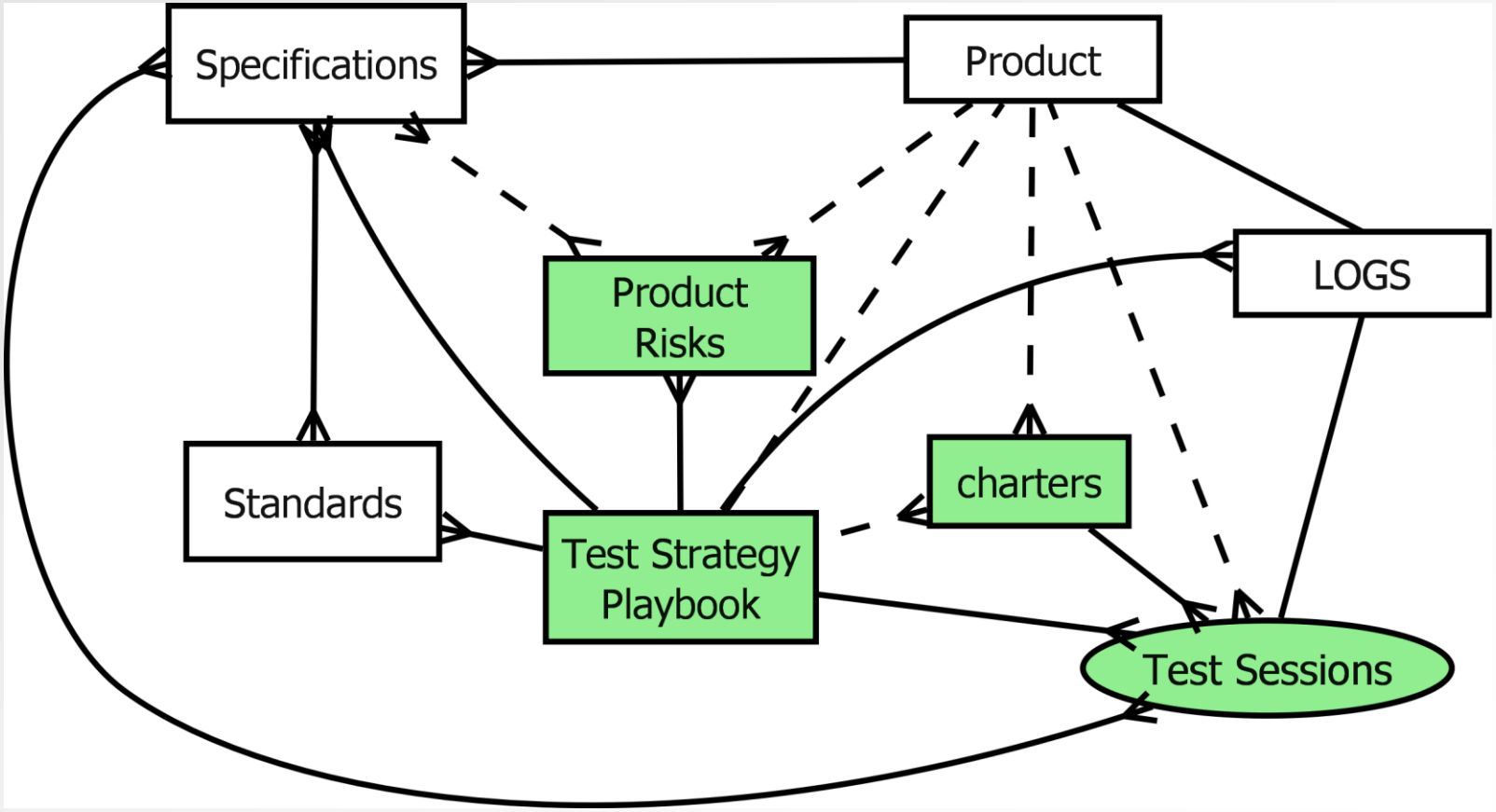
...as it becomes available. We must *explore*.



An alternative to *artifact-based* testing is *activity-based* testing.

- Manage testing as an activity consisting of manageable activity units.
- An activity is something a human tester does.
- This can be formalized:
  - *Exploratory Testing*
  - *Automatic coverage logging*
  - *Session-Based Test Management*
  - *General Functionality and Stability Test Procedure*
- Testing knowledge can be propagated formally and informally through concise documentation and on-the-job training.

“Look, Ma! No test cases!”



Traceability

1. Product traces to specifications.
2. Specifications trace to standards.
3. Test sessions trace to product versions.
4. Test sessions trace to specifications.
5. Test sessions trace to logs which trace to product, playbook and specifications.
6. Test sessions trace to charters and charters to playbook.
7. Playbook traces to standards.
8. Playbook traces to specifications.
9. Playbook traces to risks which trace to specifications.

Introducing the "Testing Playbook"

- A *testing playbook* is a test strategy document that is designed to facilitate sapient[1] software testing.

- I just invented testing playbooks last month.[2]

[1] *"Sapient testing" means testing that requires a human to guide it.*

[2] *Well, sort of... I invented the term. I've been doing this sort of thing since my first test project.*