A Rant About Thinking: Beware of Numerism and Quantasms

- *Numerism:* blind faith in numbers.
- *Quantasm:* the lapse of judgment induced in a numerist by the sight of numbers.

Why we must be careful with numbers:

- *Expected Value:* probability ($0 \leq p \leq 1$) * value ($-\infty < v < \infty$)
- *"Risk Magnitude":* subjective likelihood factor ($1 \leq L \leq 10$) * subjective impact factor ($1 \leq I \leq 10$)

Both of these fit intuition, but only expected value is rational

How "Risk Magnitude" Can Outlaw the Right Answer

- A very severe event (impact = 10)…
- that happens rarely (likelihood = 1)…
- has less magnitude than 73% of the risk space.
- A 5 X 5 risk has a magnitude that is %150 greater.
- What about Borland's Turbo C++ project file corruption bug? It cost hundreds of thousands of dollars and motivated a product recall. Yet it was a "rare" occurrence, by any stretch of the imagination. *It would have scored a 10, even though it turned out to be the biggest problem in that release.*

How Testers Think

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

Too many textbooks treat testers as clerical wind-up toys.

"This model identifies a **standards-based** life cycle testing process that concentrates on developing **formal test documentation** to implement **repeatable structured testing** on a software or hardware/software system. The general intent is that the test documentation be developed based on a **formal requirements specification** document...**Once the documentation is developed, the test is executed**."

-- from a real article about testing.
(*I added the boldfacing to emphasize* ***instructions****)*

Where's the *thinking* in this picture?

Testers light the way.

This is our role.

We make informed decisions about quality possible.
Because we think critically about software.

By our *thinking*, we can compensate for a difficult project environment.

Instead of this…	consider this.
– complete specs	– implicit specs & inference
– quantifiable criteria	– meaningful criteria
– protected schedule	– risk-driven iterations
– early involvement	– good working relationship
– zero defect philosophy	– good enough quality
– complete test coverage	– enough information

Testing is about questions; Posing them and answering them.

- **Product**
  - What is this product?
  - What can I control and observe?
  - What should I test?
- **Tests**
  - What would constitute a diversified and practical test strategy?
  - How can I improve my understanding of how well or poorly this product works?
  - If there were an important problem here, how would I uncover it?
  - What document to load? Which button to push? What number to enter?
  - How powerful is this test?
  - What have I learned from this test that helps me perform powerful new tests?
  - What just happened? How do I examine that more closely?
- **Problems**
  - What quality criteria matter?
  - What kinds of problems might I find in this product?
  - Is what I see, here, a problem? If so, why?
  - How important is this problem? Why should it be fixed?

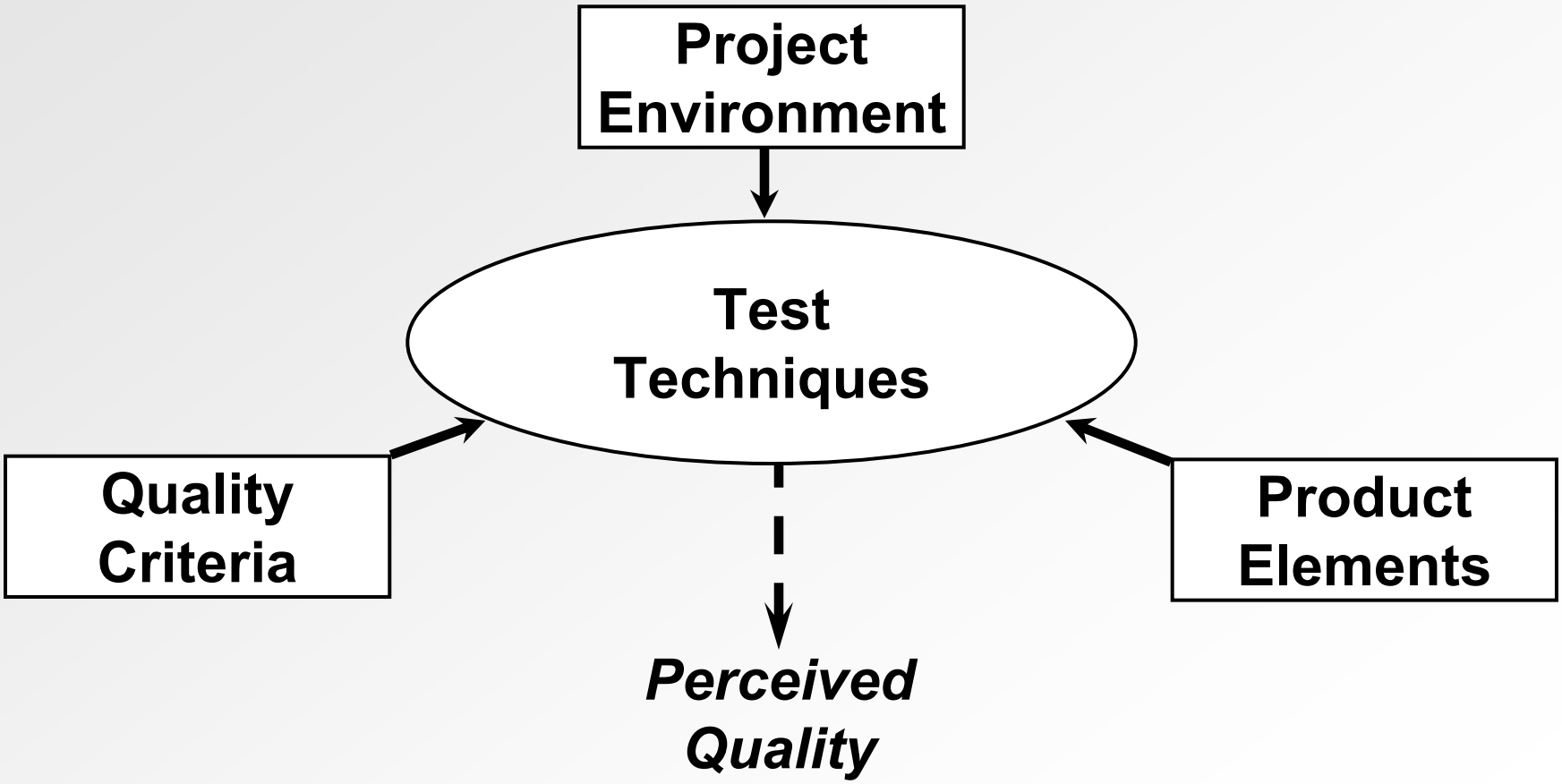
Testing is about ideas. Heuristics give you ideas.

- A heuristic is a *fallible* idea or method that may help solve a problem.
- You don't *comply* to a heuristic, you apply it. Heuristics can hurt you when elevated to the status of authoritative rules.
- Heuristics represent wise behavior only in context. They do not *contain* wisdom.
- Your relationship to a heuristic is the key to applying it wisely.

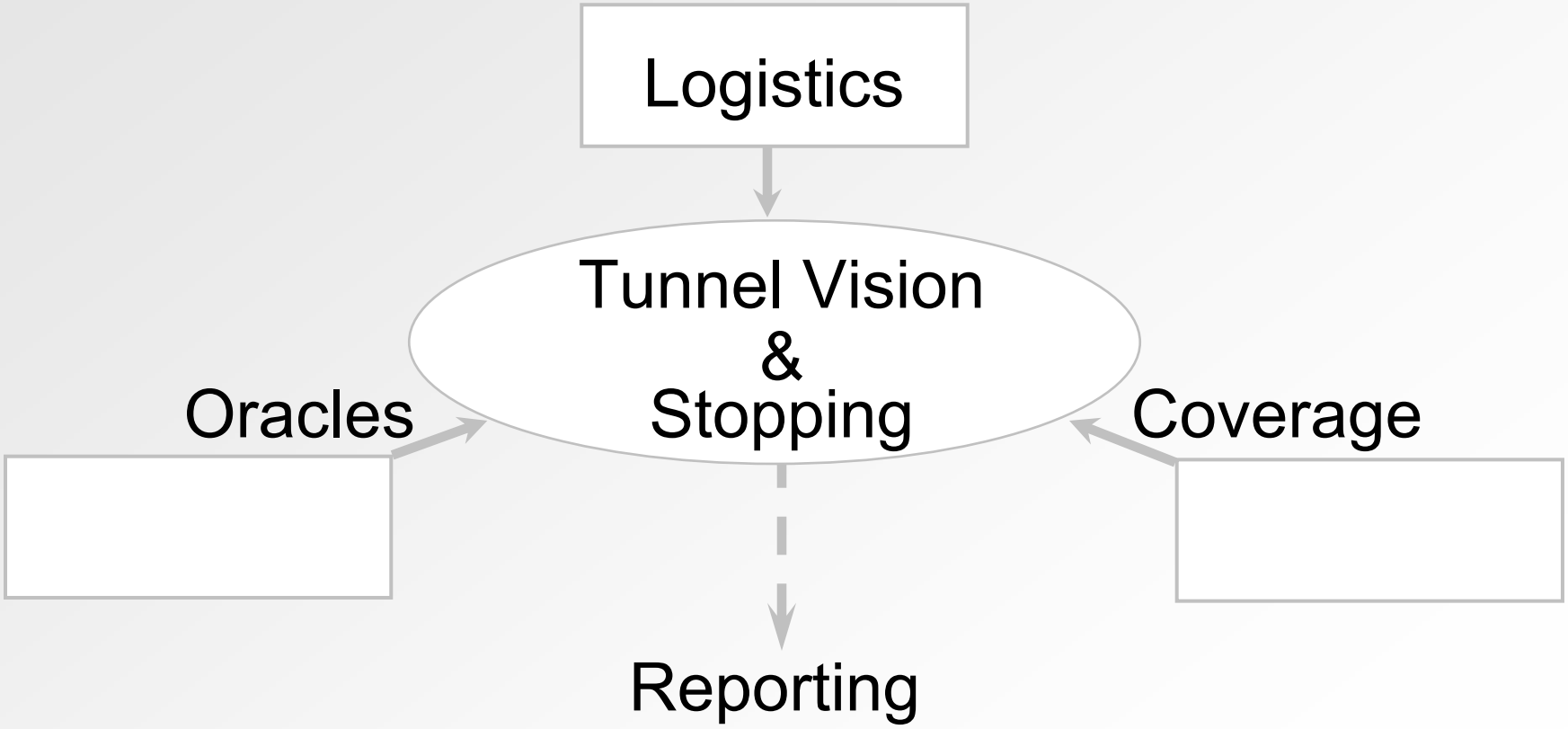
"Heuristic reasoning is not regarded as final and strict but as provisional and plausible only, whose purpose is to discover the solution to the present problem."

- George Polya, *How to Solve It*

Heuristic Model for Test Design



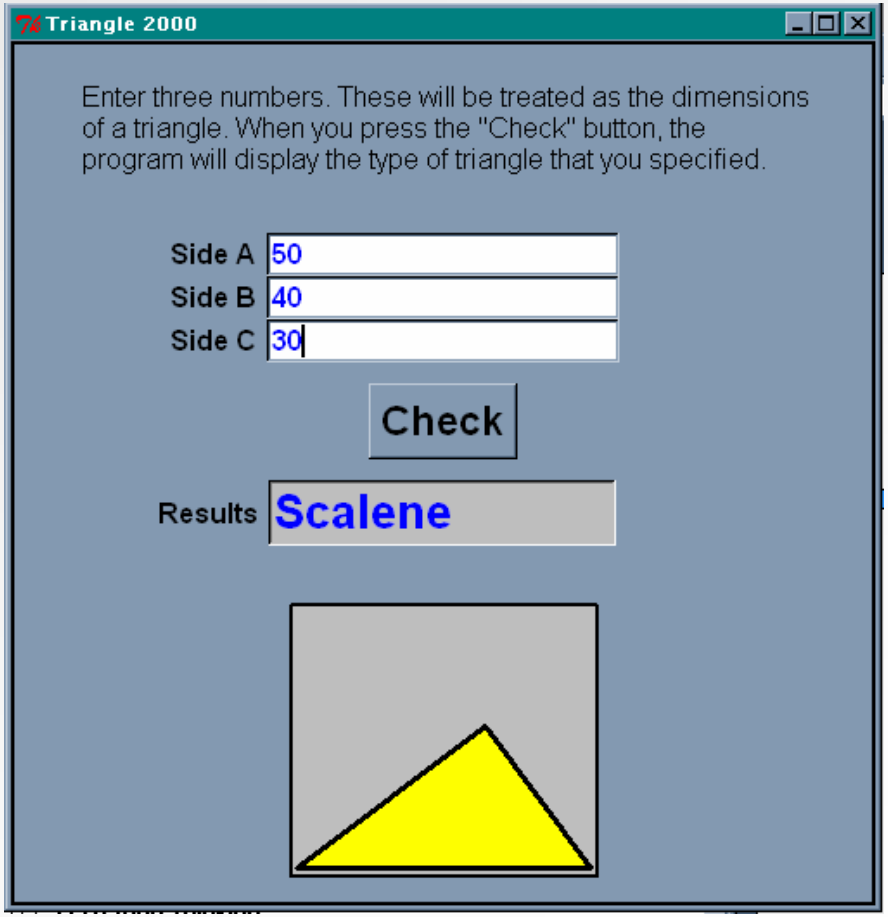
Six Problems of Testing



Cloistered Coverage: **The *Triangle* Program**

Students in Satisfice's *Rapid Software Testing* class are given 20 minutes to test this program.

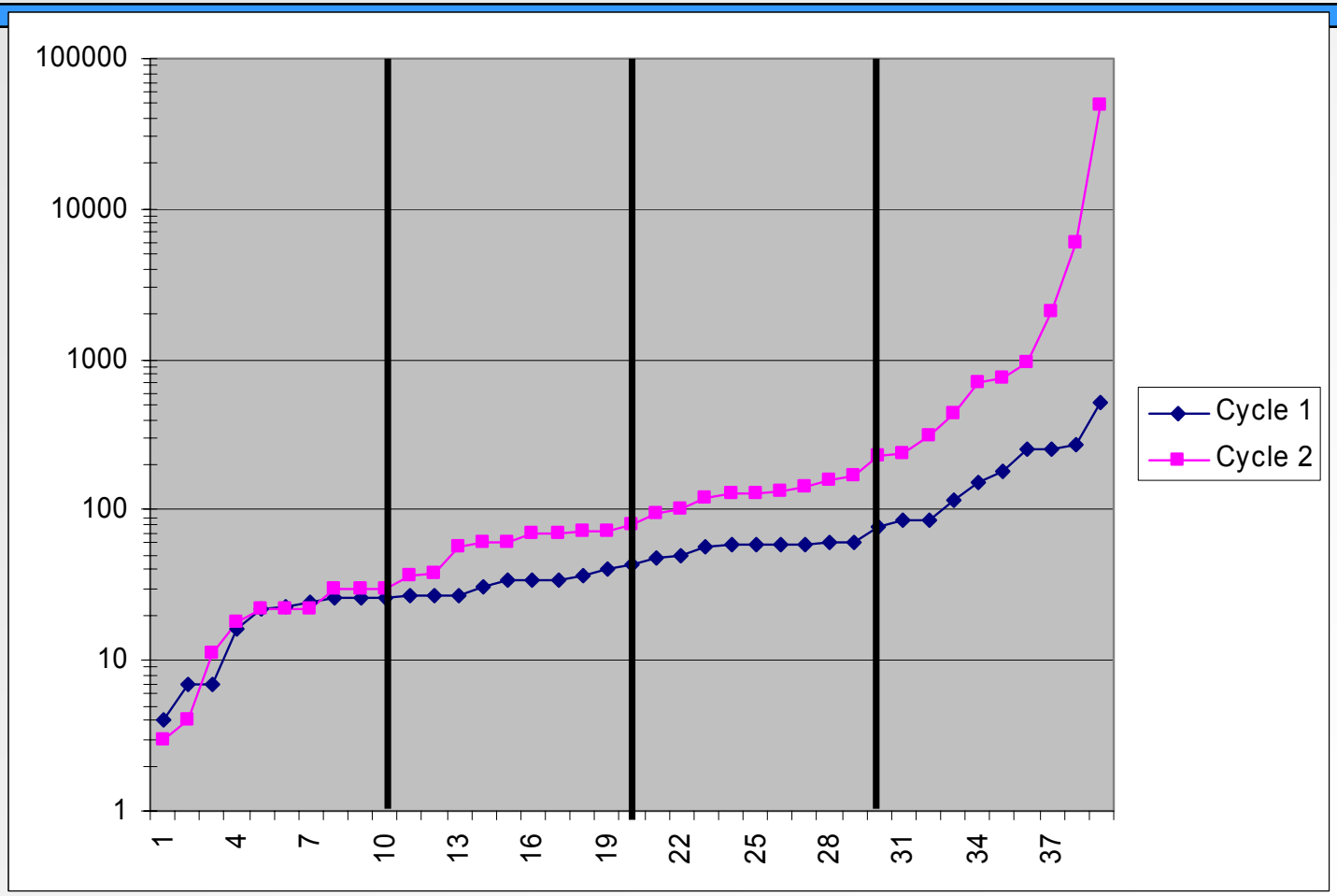
We see interesting differences in how testers approach this task.



How Well Does Triangle Handle Long Inputs?

- What does “long” mean?
- What does “handle well” mean?
- What will users do? What will they expect?
- So what?

Field Lengths Chosen by 39 Testers, Over Two Cycles



Interesting Lengths

- **16** digits & up: loss of mathematical precision.
- **23** digits & up: can't see all of the input.
- **310** digits & up: input not understood as a number.
- **1,000** digits & up: exponentially increasing freeze when navigating to the end of the field by pressing <END>.
- **23,829** digits & up: all text in field turns white.
- **2,400,000** digits: crash (reproducible).

Most of these are unknown to the programmer!

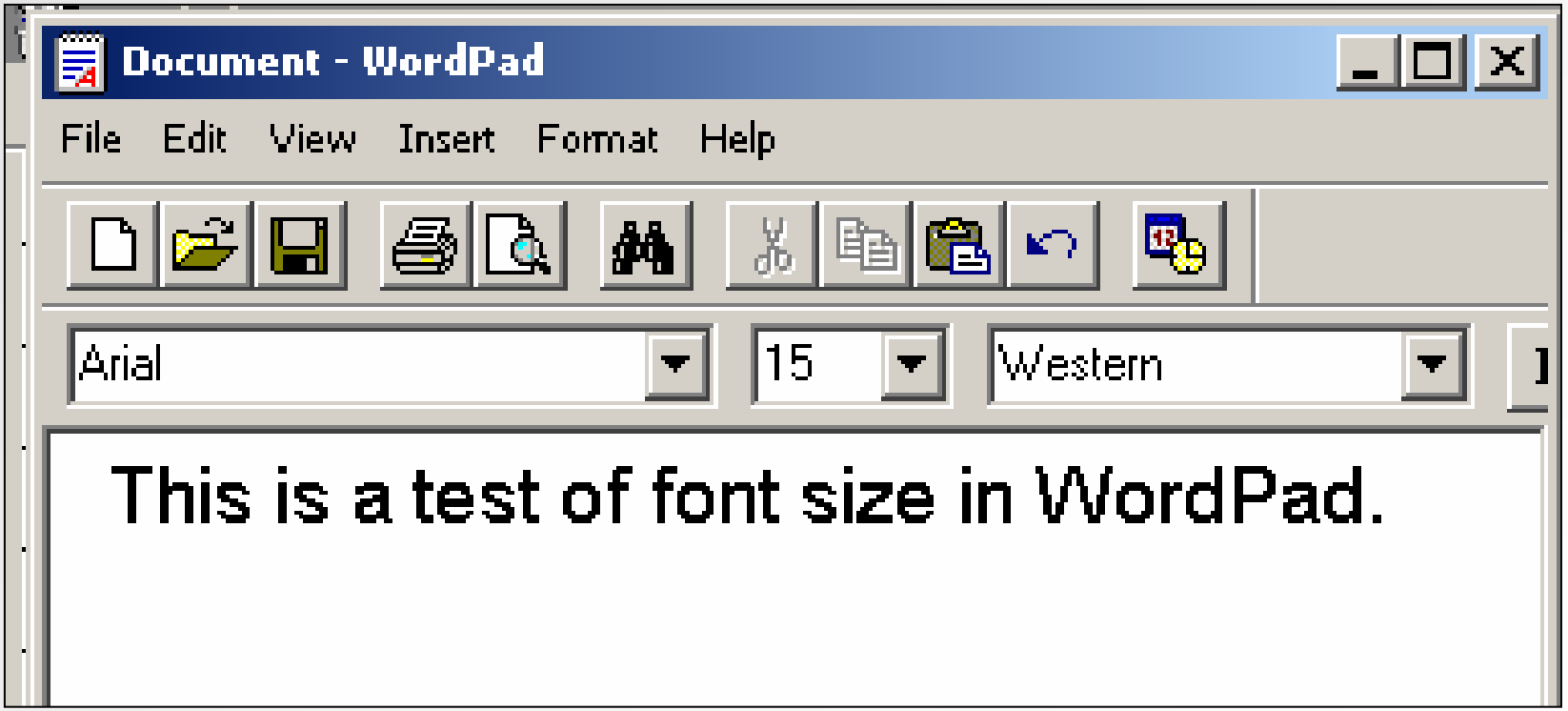
What stops testers from trying longer inputs?

- Seduced by what's visible.
- Think they need a spec that tells them the max.
- If they have a spec, they stop when the spec says stop.
- Satisfied by the first boundary (16 digits).
- Let their fingers do the walking instead of using a program like notepad to generate input.
- Use strictly linear lengthening strategy.
- Don't realize the significance of degradation.
- Assume it will be too hard and take too long.
- Think "No one would do that" (hackers do it)

Thinking About Coverage

- Testers with less expertise…
  - Think about coverage mostly in terms of what they can see.
  - Cover the product indiscriminately.
  - Avoid questions about the completeness of their testing.
  - Can't reason about how much testing is enough.
- Better testers are more likely to…
  - Think about coverage in many dimensions.
  - Maximize diversity of tests while focusing on areas of risk.
  - Invite questions about the completeness of their testing.
  - Lead discussions on what testing is needed.

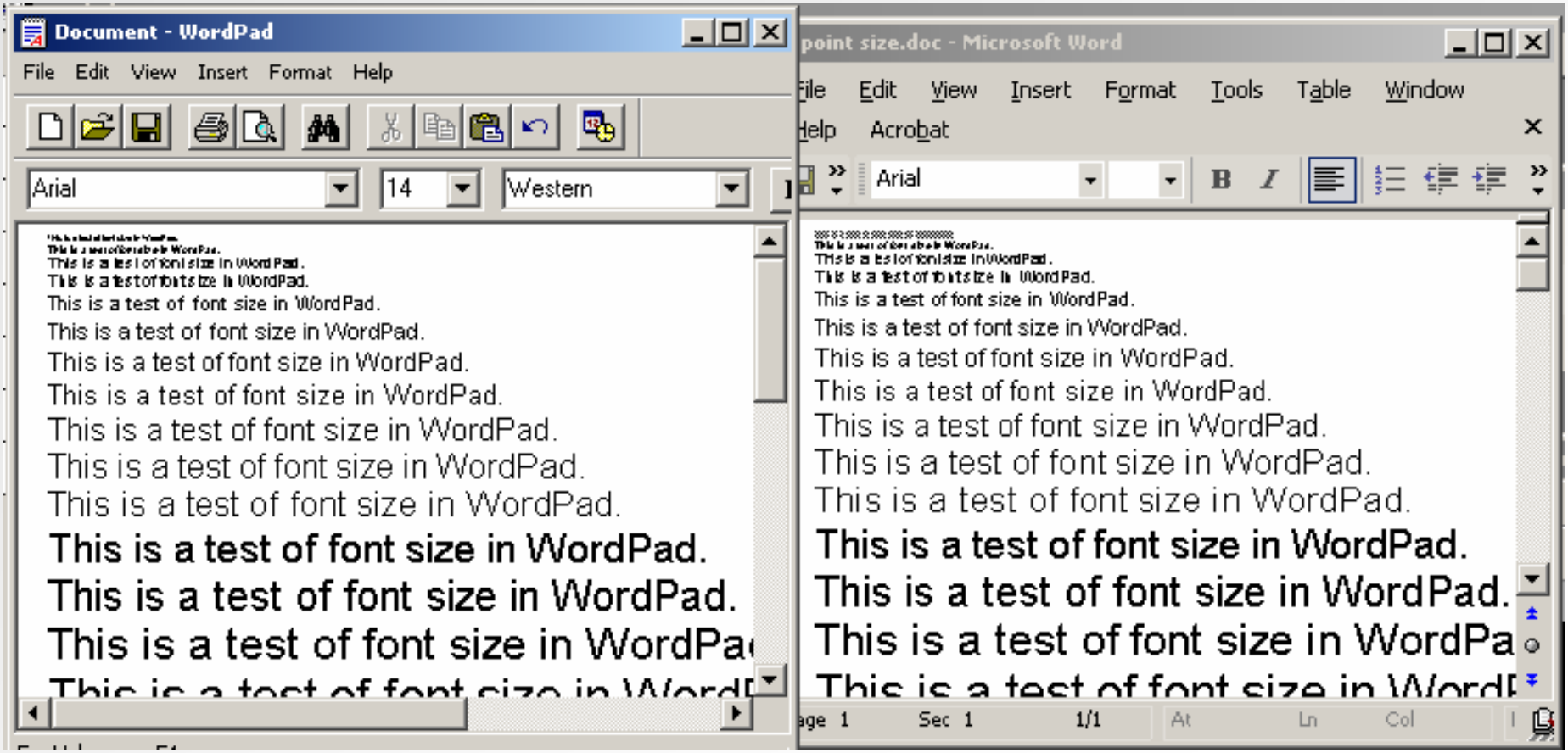
Oblivious Oracles:
Does font size work in WordPad?



Is this the right font size? What's your oracle?

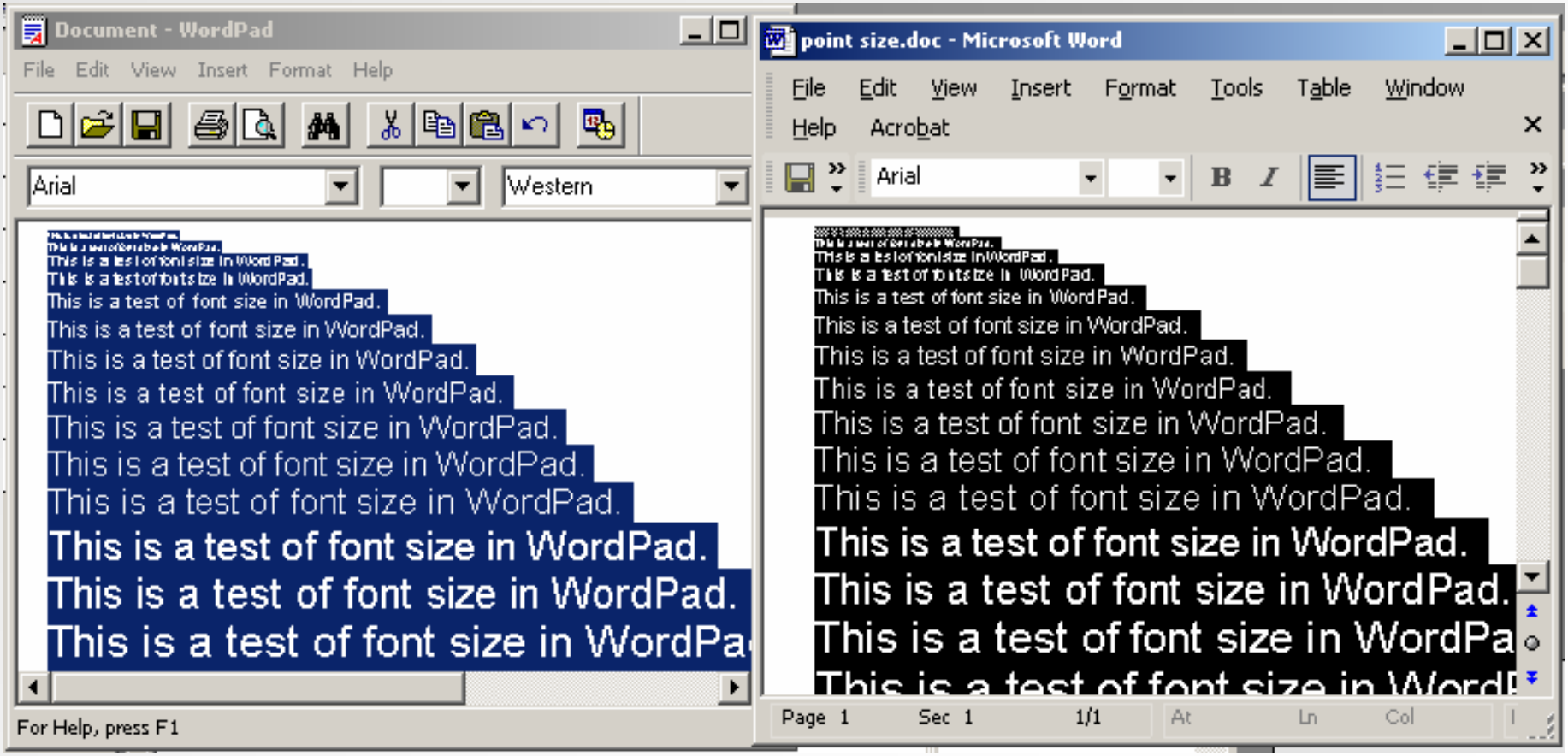
The Comparable Product Heuristic will save us! Or will it…?

WordPad **Word**



What does this tell us?

WordPad **Word**



Oracle Heuristics: "HICCUPP"

- **Consistent with History:** Present function behavior is consistent with past behavior.
- **Consistent with our Image:** Function behavior is consistent with an image that the organization wants to project.
- **Consistent with Comparable Products:** Function behavior is consistent with that of similar functions in comparable products.
- **Consistent with Claims:** Function behavior is consistent with what people say it's supposed to be.
- **Consistent with User's Expectations:** Function behavior is consistent with what we think users want.
- **Consistent within Product:** Function behavior is consistent with behavior of comparable functions or functional patterns within the product.
- **Consistent with Purpose:** Function behavior is consistent with apparent purpose.

Better Thinking

- *Conjecture and Refutation*: reasoning without certainty.
- *Abductive Inference*: finding the best explanation among alternatives.
- *Lateral Thinking:* the art of being distractible.
- *Forward-backward thinking*: connecting your observations to your imagination.
- *Heuristics:* applying helpful problem-solving short cuts.
- *De-biasing:* managing unhelpful short cuts.
- *Pairing:* two testers, one computer.
- *Study other fields.* Example: Information Theory.