Black Box Software Testing

Fall 2005

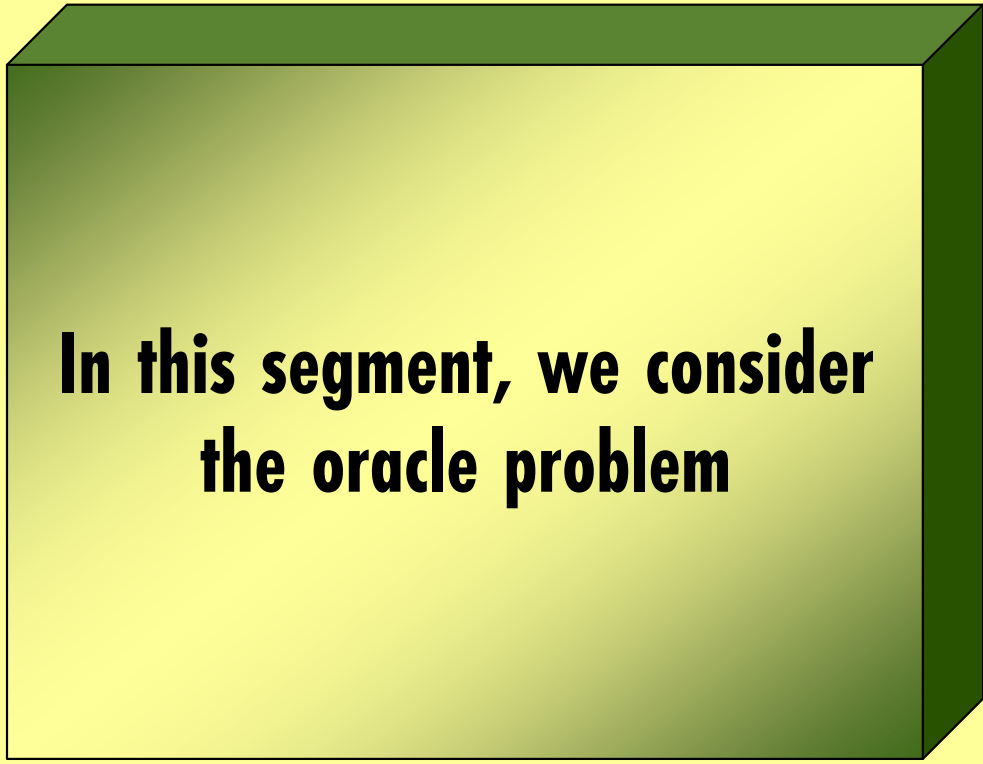
Overview–Part 3 (Test oracles)

Cem Kaner, J.D., Ph.D.
Professor of Software Engineering
Florida Institute of Technology
and
James Bach
Principal, Satisfice Inc.

Copyright (c) Cem Kaner & James Bach, 2000-2005
This work is licensed under the Creative Commons Attribution-ShareAlike License. To view a copy of this license, visit http://creativecommons.org/licenses/by-sa/2.0/ or send a letter to Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.
These notes are partially based on research that was supported by NSF Grant EIA-0113539 ITR/SY+PE: "Improving the Education of Software Testers." Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Some fundamental questions in software testing

- Why are you testing? What are you trying to learn? ***[What is the mission of your testing?]***
- How should you organize your work to achieve your mission? ***[The strategy problem]***
- How will you know whether the program passed or failed the test? ***[The oracle problem]***
- What would it take to do a complete testing job? ***[The impossibility of complete testing.]***
- How much testing is enough? ***[The measurement problem.]***



Oracles

An oracle is the principle or mechanism by which you recognize a problem.

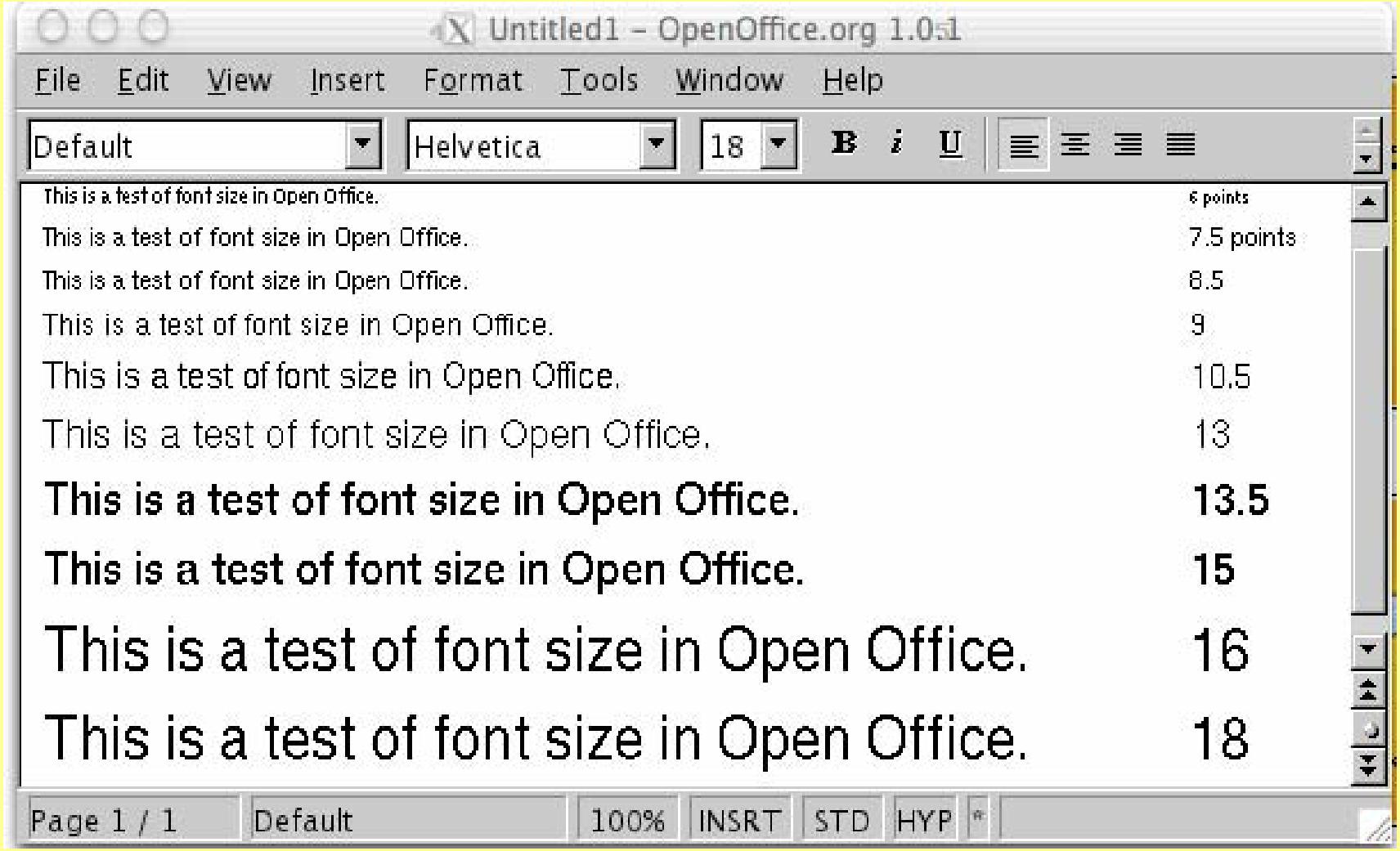
"..it works"

really means...

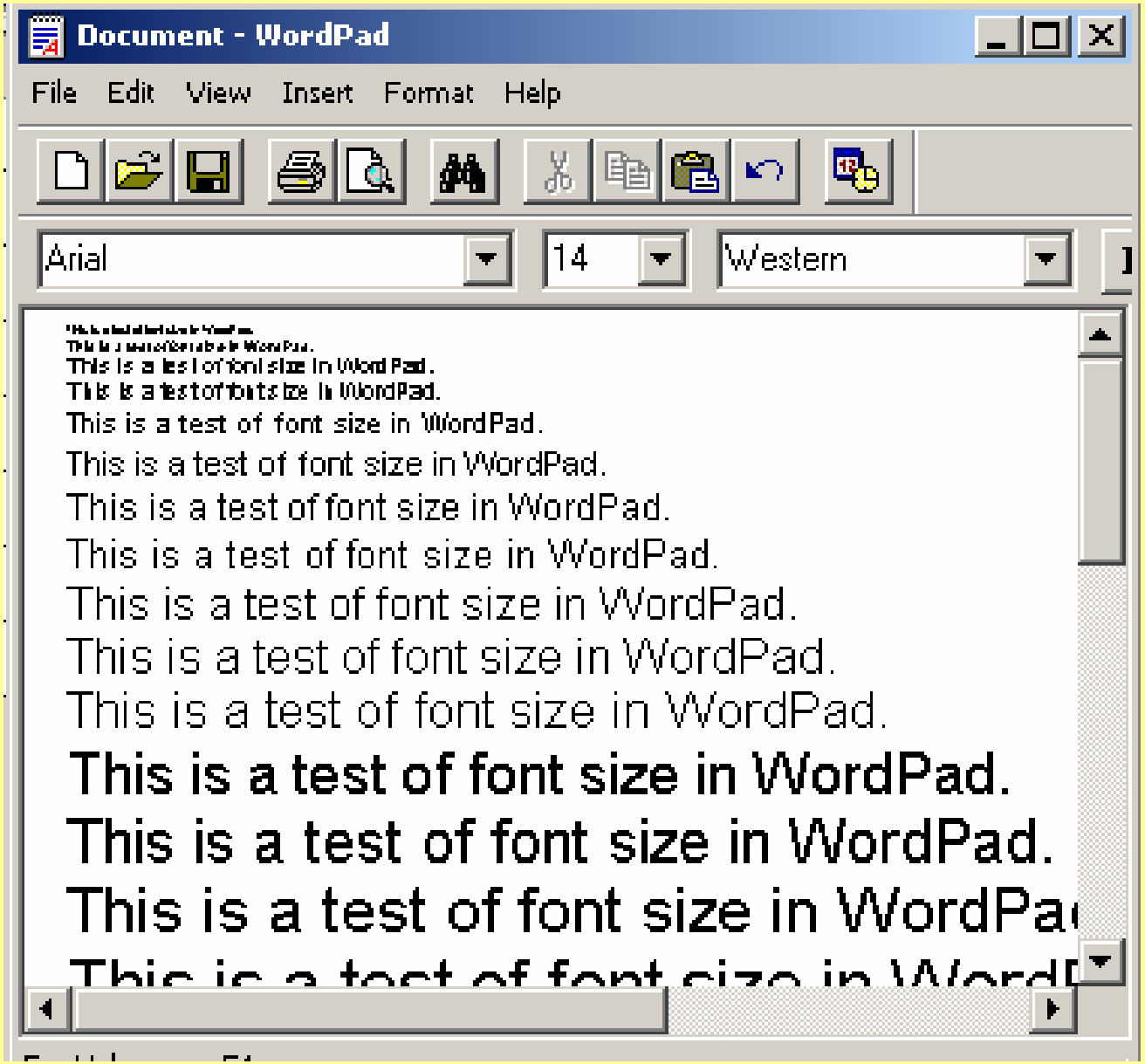
"...it appeared to meet some requirement to some degree."

Does font size work in Open Office?

What's your oracle?



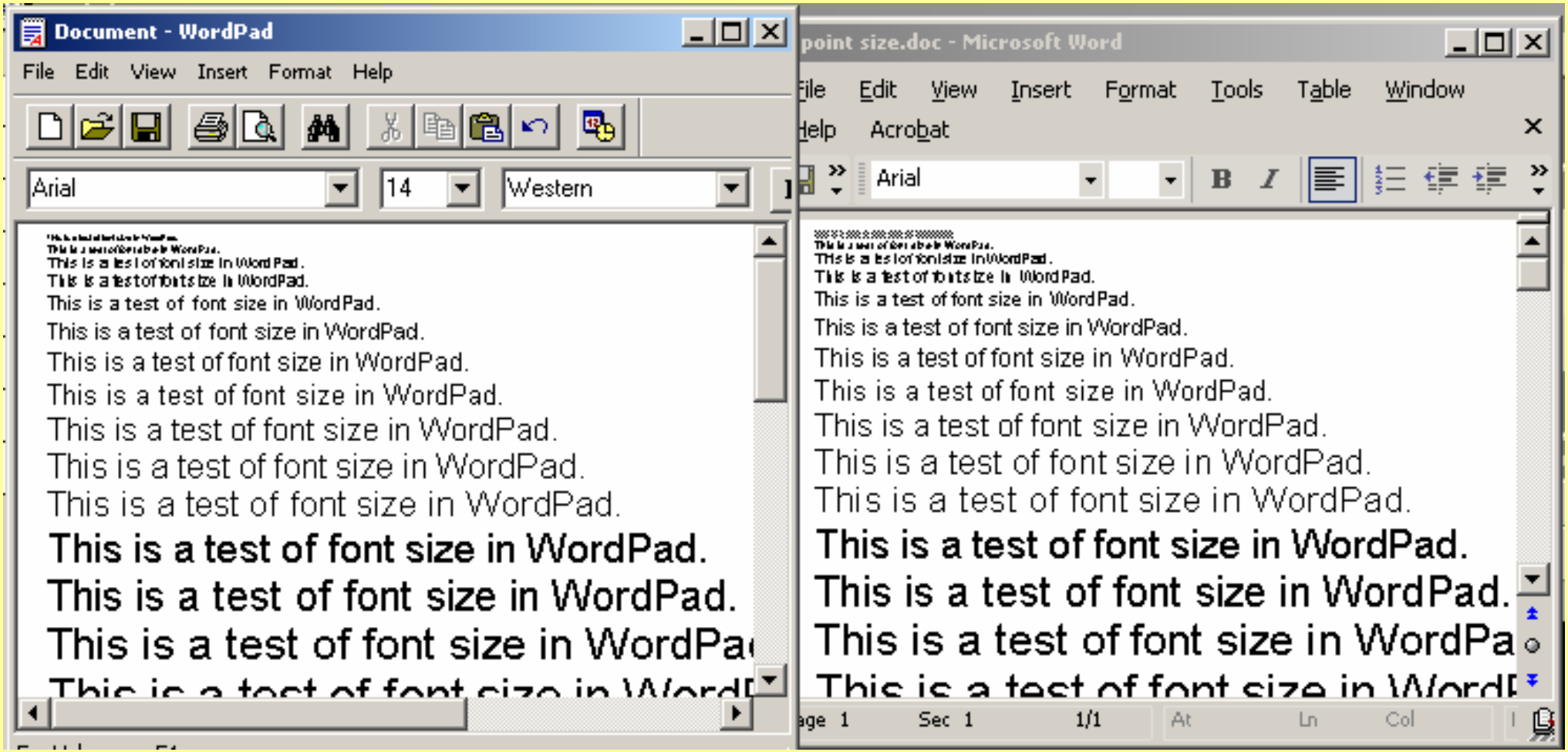
OK, so what about WordPad?



Compare WordPad to Word?

Pretty similar, eh?

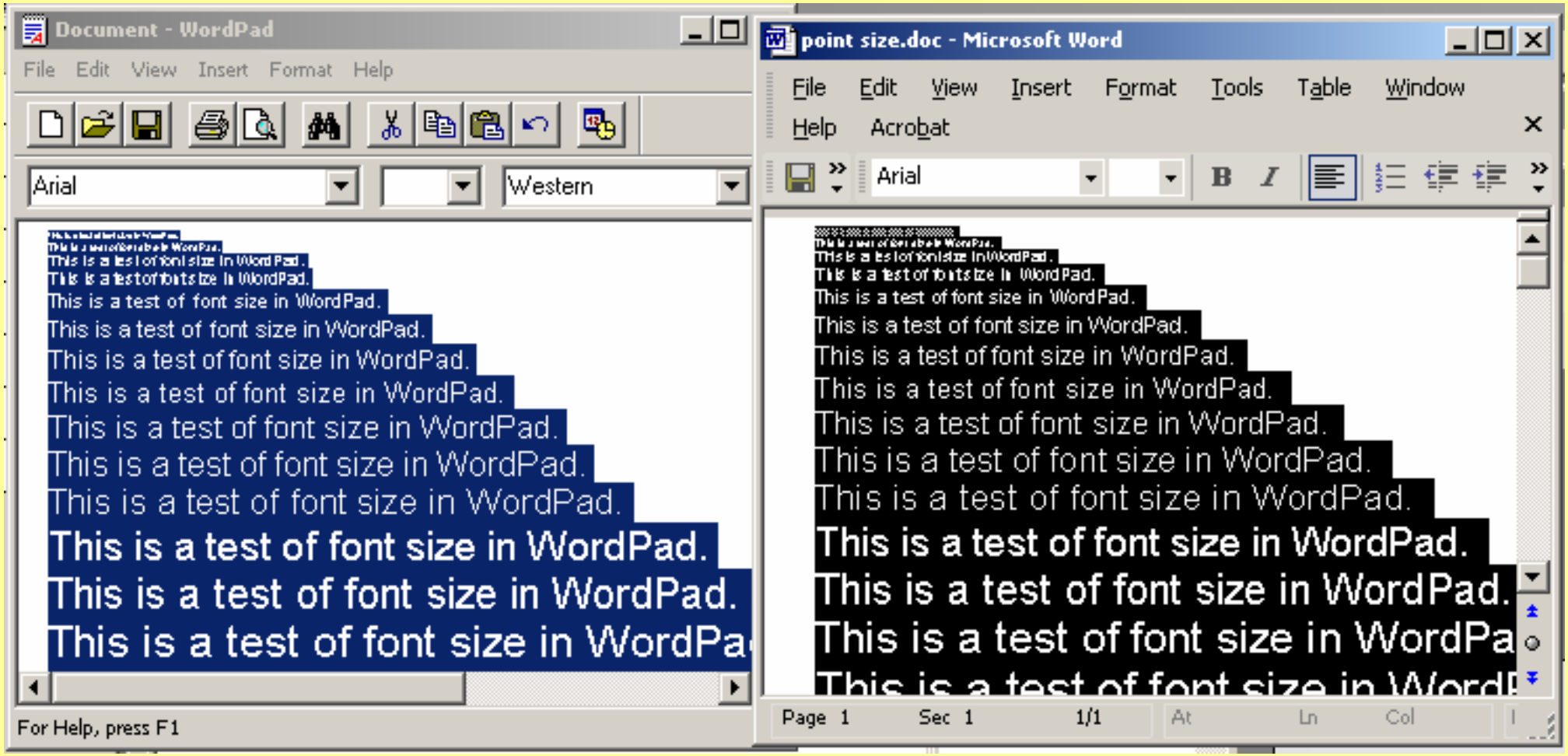
WordPad **Word**



Make the comparison easier

Highlighting makes relative sizes more obvious

WordPad **Word**



Now that we see a difference ...

… Is it a difference that makes a difference?

The oracle problem highlights the fundamental role of judgment in testing.

Risk as a simplifying factor

For ***Wordpad***, we don't care if font size meets precise standards of typography!

In general it can vastly simplify testing if we focus on whether the product has a problem that matters, rather than whether the product merely satisfies all relevant standards.

Effective testing requires that we understand standards as they relate to how our clients value the product.

Instead of thinking about *pass vs. fail,*
Consider thinking *problem vs. no problem.*

Risk as a simplifying factor

What if we applied the same evaluation approach

- that we applied to WordPad
- to Open Office or MS Word or Adobe PageMaker?

In risk-based testing,

- we choose the tests that we think are
  - the most likely to expose a serious problem,
- and skip the tests that we think are
  - unlikely to expose a problem, or
  - likely to expose problems that no one would care about.

The same evaluation criteria lead to different conclusions in different contexts

So, what are we testing for?

What to cover?

- Every font size (to a tenth of a point).
- Every character in every font.
- Every method of changing font size.
- Every user interface element associated with font size.
- Interactions between font size and other contents of a document.
- Interactions between font size and every other feature.
- Interactions between font sizes and graphics cards & modes.
- Print vs. screen display.

What evaluation criterion?

- What do you know about typography?
  - Definition of "point" varies. There are at least six different definitions

    (http://www.oberonplace.com/dtp/fonts/point.htm)
  - Absolute size of characters can be measured, but not easily

    (http://www.oberonplace.com/dtp/fonts/fontsize.htm)
- How closely must size match to the chosen standard?
- **Heuristic approaches**, such as:
  - relative size of characters
  - comparison to MS Word
  - expectations of different kinds of users for different uses.

Testing is about ideas.
Heuristics give you ideas.

Testing is about ideas.
Heuristics give you ideas.

A heuristic is a fallible idea or method that may you help simplify and solve a problem.

Heuristics can hurt you when used as if they were authoritative rules.

Heuristics may suggest wise behavior, but 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*

Copyright © 2003-05

Billy V. Koen, Definition of the Engineering Method, ASEE, 1985

"A heuristic is anything that provides a plausible aid or direction in the solution of a problem but is in the final analysis unjustified, incapable of justification, and fallible. It is used to guide, to discover, and to reveal.

"Heuristics do not guarantee a solution.

"Two heuristics may contradict or give different answers to the same question and still be useful.

"Heuristics permit the solving of unsolvable problems or reduce the search time to a satisfactory solution.

"The heuristic depends on the immediate context instead of absolute truth as a standard of validity."

Koen (p. 70) offers an interesting definition of engineering "The engineering method is the use of heuristics to cause the best change in a poorly understood situation within the available resources"

Some useful oracle heuristics

Consistent within product: Function behavior consistent with behavior of comparable functions or functional patterns within the product.

Consistent with comparable products: Function behavior consistent with that of similar functions in comparable products.

Consistent with history: Present behavior consistent with past behavior.

Consistent with our image: Behavior consistent with an image the organization wants to project.

Consistent with claims: Behavior consistent with documentation or ads.

Consistent with specifications or regulations: Behavior consistent with claims that must be met.

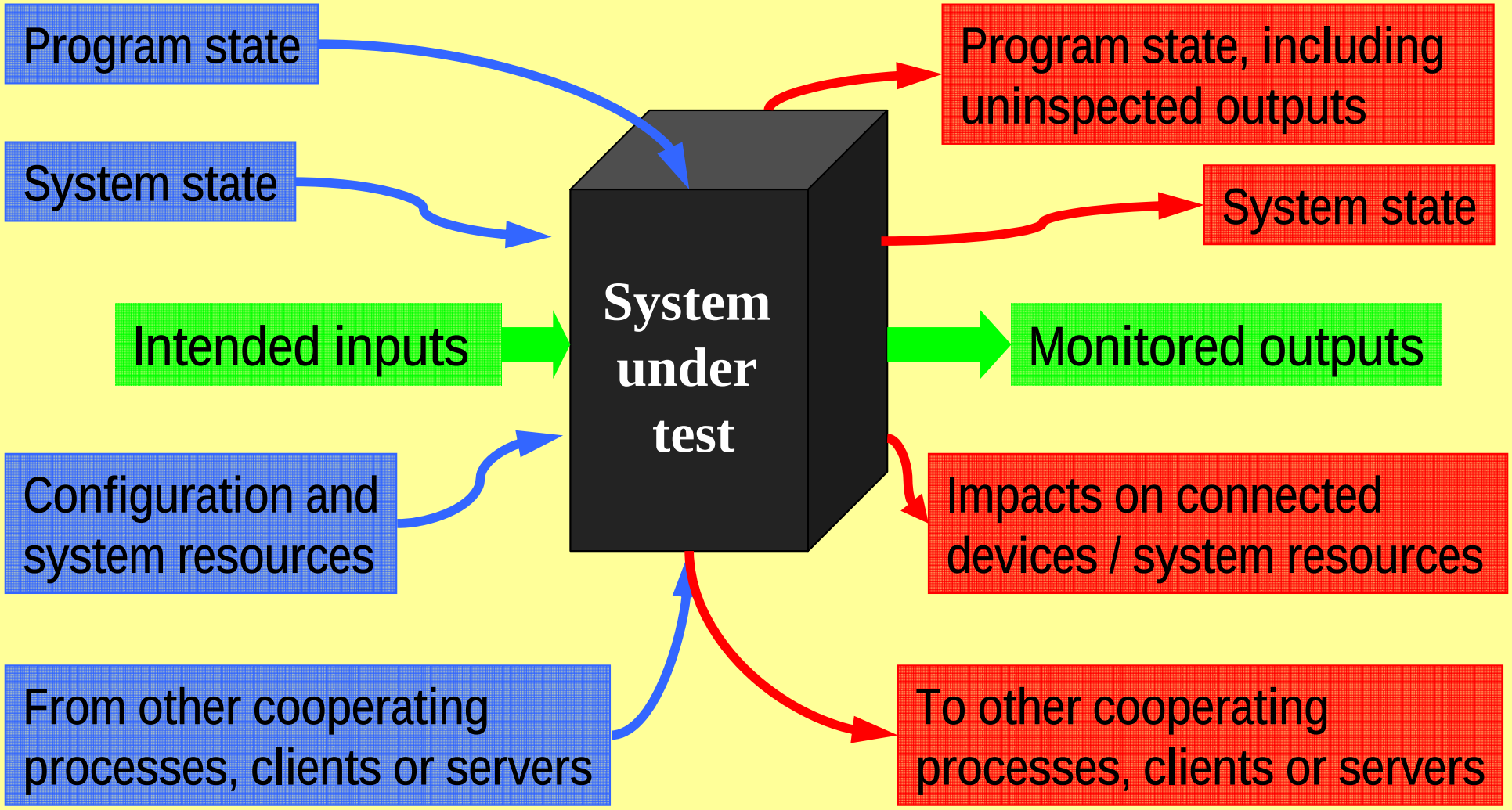
Consistent with user's expectations: Behavior consistent with what we think users want.

Consistent with Purpose: Behavior consistent with product or function's apparent purpose.

These are especially useful for explaining a bug and its significance.

A program can fail in many ways

Based on notes from Doug Hoffman



A program can fail in many ways

1100 embedded diagnostics

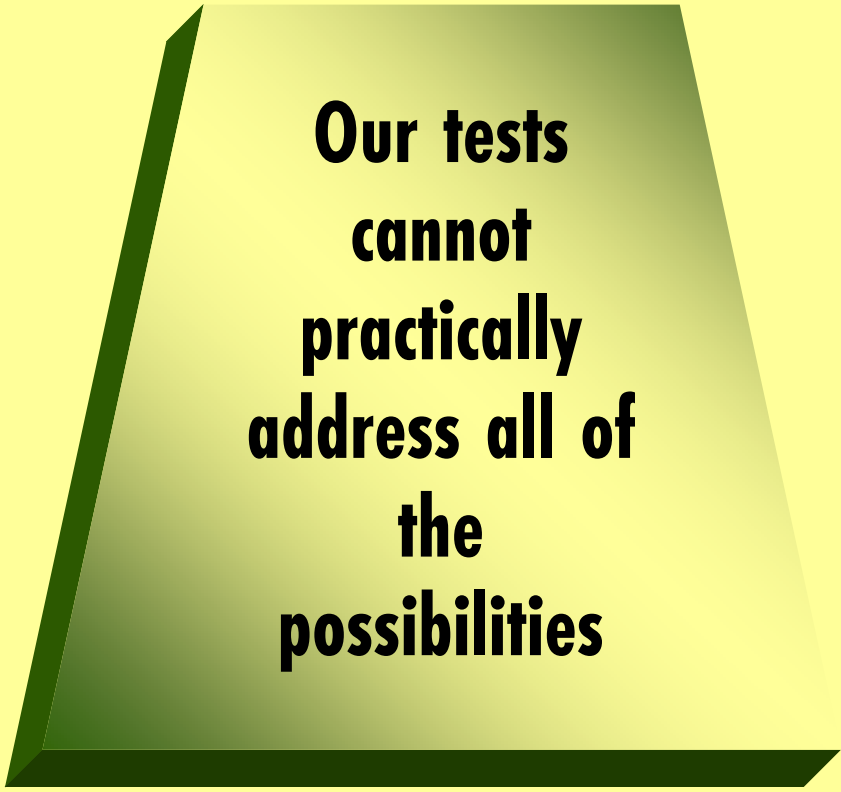
- Even if we coded checks for each of these, the side effects (data, resources, and timing) would provide us a new context for Heisenberg uncertainty.

Inattentional blindness

- Humans (often) don't see what they don't pay attention to.
- Programs don't see what they haven't been told to pay attention to.

This is often the cause of irreproducible failures. We pay attention to the wrong conditions.

- We can't attend to all the conditions



The oracle problem and test automation

We often hear that most (or all) testing should be automated.

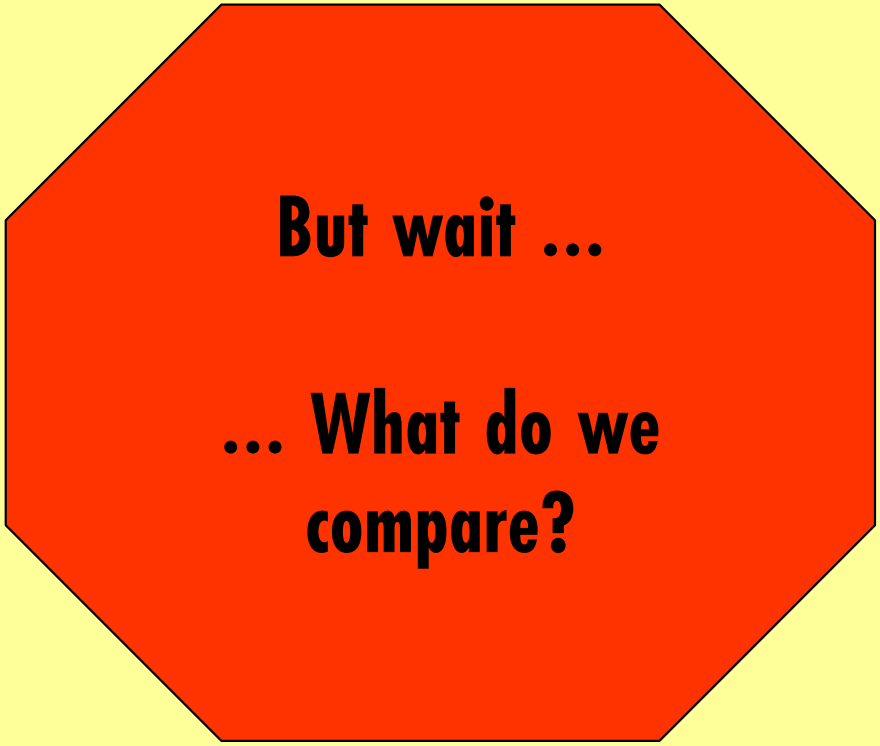
Automated testing depends on our ability to **programmatically detect** when the software under test fails a test.

The oracle problem and automation

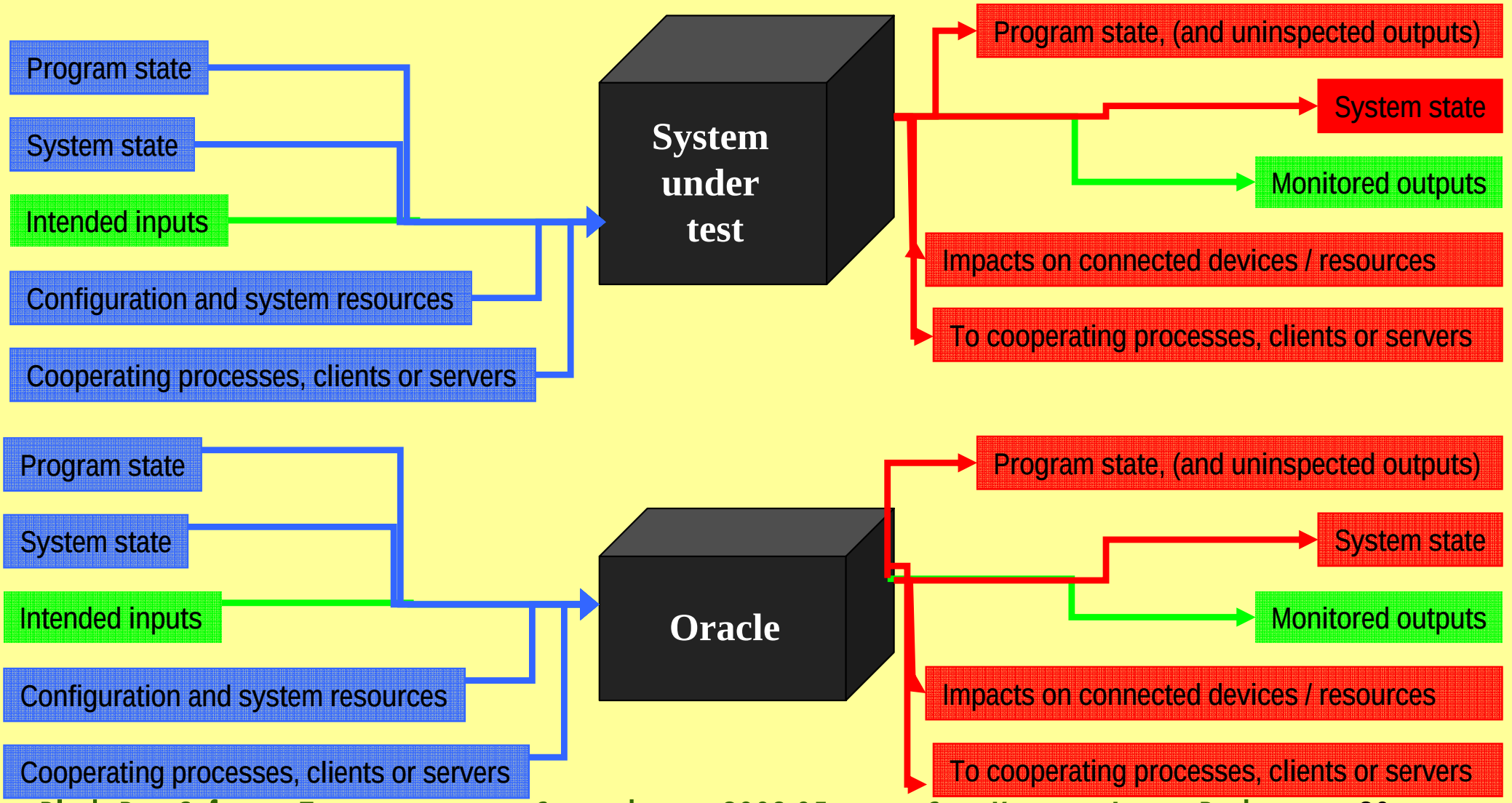
One common way to define an oracle is as a source of expected results.

- Under this view, you compare your results to those obtained from the oracle. If they match, the program passes. If they don't match, the program fails.
- The comparison between MS WordPad and MS Word illustrates this approach.

This is the most straightforward approach for automation



What do you compare, when you use an oracle? Based on notes from Doug Hoffman



Automated comparison is heuristic

False alarms:

- Word vs. WordPad mismatches might be appropriate or might not matter.
- Big problem for automated regression tests (compare old to new versions). Design changes causes mismatches—we must spend time fixing the tests.

Missed bugs:

- Word vs. WordPad matches might result from the same error in both.

Automated comparison-based testing is subject to false alarms and misses. Automate or not, you must still exercise judgment in picking risks to test against and interpreting the results.

Let's sum up

Testing is a complex activity.

We introduced five of the fundamental testing challenges in this letter and took a first look at three:

What is the mission? What are you looking for when you test? What are your information objectives? We'll return to this.

What is the strategy? What is your plan for getting the information?

What is the oracle? How will you decide whether the program passed or failed a test, given that all oracles are imperfect.

Next class, we consider the impossibility of complete testing and take a first look at the measurement problem.