I'm Michael Bolton.

http://www.developsense.com

I teach people how to test software.

James Bach and I co-author a course called Rapid Software Testing.

Testing, the way we teach it, is based on heuristics.

A heuristic is a fallible method for solving a problem.

A heuristic often works, and it sometimes fails.

In testing, an **oracle** is a **principle or mechanism by which we recognize a problem.**

All oracles are heuristic.

(Not all heuristics are oracles.)

Since all oracles are heuristic, an oracle often works, though it can sometimes fail.

When we see a problem, it's a problem with respect to some oracle.

Oracles don't guarantee that there's a problem, but they point to potential problems.

Consistency is an important theme in oracles.

So is **inconsistency**.

We suspect a problem if a product is inconsistent with its history.

We suspect a problem if a product is inconsistent with our organization's image.

We suspect a problem if a product is inconsistent with comparable products.

We suspect a problem if a product is inconsistent with claims made about it, by important people.

We suspect a problem if a product is inconsistent with user expectations.

We suspect a problem if a product is inconsistent with its presumed or intended purpose.

We suspect a problem if a product is inconsistent with itself, in two or more places.

We suspect a problem if a product is inconsistent with standards or statutes.

We suspect a problem if a product **is consistent** with problems we've seen before.

We run the program, and to some, it seems like we just notice problems as we go along.

That's all there is to it.

Noticing problems is
a very logical, objective,
dispassionate process.

Isn't it?

Recently, I realized a hole in
our profession's folklore
about the process
of testing software.

That realization led to a question,
which I'll ask presently.

But first a little
background.

I was at a conference, and a
presenter was asking why
projects based on 100% test
automation fail.

The presenter was a guy from Microsoft.

Never mind his begging the question about what he meant by 100% automation...

I noted that automation can't empathize.

I noted that automation can't anticipate.

I noted that automation can't recognize.

I noted that automation can't judge.

I noted that automation can't predict.

I noted that automation can't project.

I noted that automation can't evaluate.

I noted that automation can't assess.

I noted that automation can't become resigned.

I noted that automation can't get frustrated.

I noted that automation can't invent.

I noted that automation can't model.

I noted that automation can't resource.

I noted that automation can't collaborate.

I noted that automation can't decide.

I noted that automation can't work around a problem.

I noted that automation can't strategize.

I noted that automation can't charter.

I noted that automation can't teach.

I noted that automation can't learn.

I noted that automation can't appreciate.

I noted that automation can't question.

I noted that automation can't refine.

I noted that automation can't investigate.

I noted that automation can't speculate.

I noted that automation can't suggest.

I noted that automation can't contextualize.

I noted that automation can't explain.

I noted that automation can't elaborate.

I noted that automation can't reframe.

I noted that automation can't refocus.

I noted that automation can't troubleshoot.

I noted that automation can't THINK.

The guy from Microsoft said...

"True. But I don't see why you'd want your automation to get frustrated."

Despite a lengthy email exchange with him, in which I pointed out that

frustration indicates a problem for a person

and the Microsoft guy had just recently blogged about a bug that frustrated him

he seemed to remain clueless about why **getting frustrated might be significant**.

I was getting frustrated by **that**, so I knew I must be on to something.

I realized that often, when I notice a bug, the recognition is triggered by some emotion.

Frustration.

Confusion.

Annoyance.

Impatience.

Depression.

Boredom.

Irritation.

Curiosity.

Amusement.

Emotions are associated with a state that psychologists call **arousal**.

No, not that kind of arousal.

Arousal is a physiological and psychological state of being awake.

(according to Wikipedia)

Arousal is important in regulating consciousness, attention, and information processing.

(also according to Wikipedia)

Automation does some stuff very well.

But...

Automation doesn't get aroused.

Automation doesn't notice new bugs.

Automation doesn't identify new risks.

Automation doesn't identify opportunities for improving test design.

Fortunately, people can do those things.

So here comes my question.

Our clients are human.

Our humanity as testers helps to reveal important information about our products.

Emotions provide a rich source of oracles—principles or mechanisms by which we recognize problems.

I'll wager that **any** time we've seen a bug, our emotions were a **big factor** in recognizing or interpreting it.

Why do so many in our profession seem to be so oblivious to the value of emotions?