Testing and Notic ng

Michael Bolton
DevelopSense
michael@developsense.com

James Bach
Satisfice
james@satisfice.com

Software Test and Performance Conference
September 2009

Acknowledgements

- Mark Federman → Terrence Gordon → Marshall McLuhan for the original title ("What Haven't You Noticed Lately?")
- Cem Kaner
- Jerry Weinberg
- Adam White
- Pradeep Soundararajan
- …and the authors of the works cited in the references.

The Question We All Dread

Why didn't you find that bug?!

One Possible Answer

Uh... we didn't notice it.

Yet have you ever noticed *why*?

- "*We didn't notice it.*" Intake
- "We did notice it, but we didn't know that it was a problem." Meaning
- "We did find it, and we knew that is was a problem, but we didn't think it was important enough to worry about." Significance
- "We were busy looking for other bugs."
- "We were busy reporting other bugs that we found."

Let's look at…

NOT noticing.

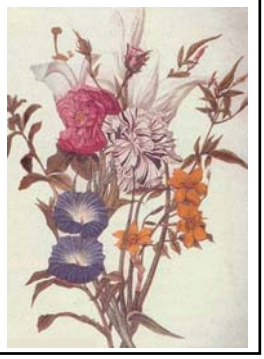
Notice Anything About The Flowers?
(Please answer silently!)

What About When We Change the Question?

Please keep your answers to yourself!

- Notice anything when you look *between* the flowers?
- What happens when we treat the flowers as *ground* and treat the background as *figure*?

Now you can answer out loud!



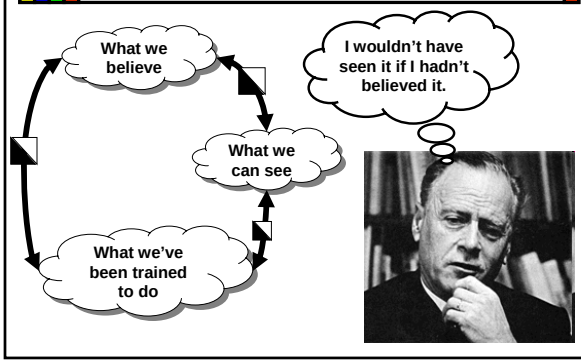
What is Noticing?

- Recognizing events, objects, or properties
- Starts with direct sensory intake, then moves to meaning, significance, and response
- An internal reaction
- Can be managed by dynamically managing observation and focus
- Often triggered by emotional reactions
- *Very* vulnerable to incompleteness and error
- Can be trained and improved

What Might We Notice?

- Problems or other kinds of information about products we test
- Things about ourselves
- Things about others
- Things about our environment
  - especially things that slow down or otherwise inhibit the best testing we can do

Feedback Loop: Perception & Conception



Conceptual Priming Ideas

- The skill of factoring
- Your technical knowledge and beliefs
  - beliefs about what is (im)possible
  - beliefs about what is (un)likely
- Guideword heuristics
- Patterns of familiar bugs
- Learning how rapid cognition works
  - …and how it might fail

Have you ever noticed…?

Some priming can reduce noticing.

Have you ever noticed…?

Diversity supports better noticing.

Practice Factoring!

- *"List all the dimensions of (some common object) that may be relevant to testing it."*
- **"dimensions"** means attributes of the object that may vary
  - from one object to another; or
  - within the same object over time.
- **"relevant to testing"** means that there is probably some value to some client, with respect to some testing mission, of manipulating or observing a particular dimension.
- Pick an object and factor it; compare notes; notice categories of dimensions (and of categories)

There Are Factors To Observation, Too!

- the thing being observed (the *system*)
- the environment (all the things around us)
- our knowledge and models (conceptual)
- our senses (perceptual)
- our experience (experiential)
- our feelings (affective)
- our mission and our client

…*and* there are factors to each one of these factors!

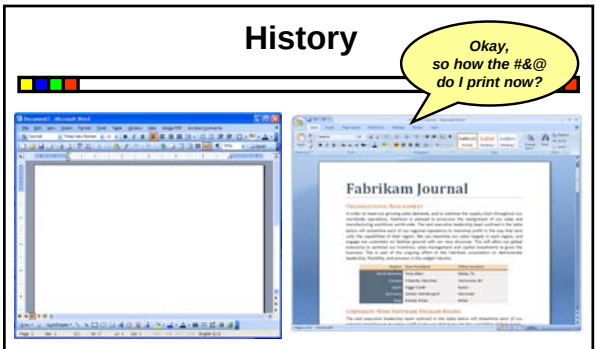
Consistency ("this agrees with that")

an important theme in oracles

History
Image
Comparable Products
Claims
User Expectations
Purpose
Product
Standards

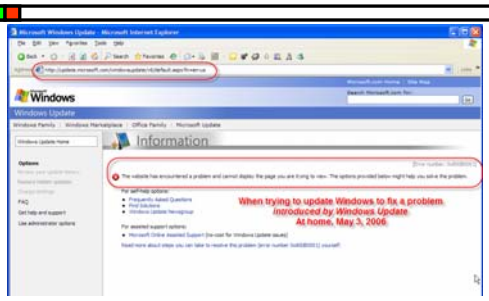
Consistency heuristics rely on the quality of your models of the product and its context.

History



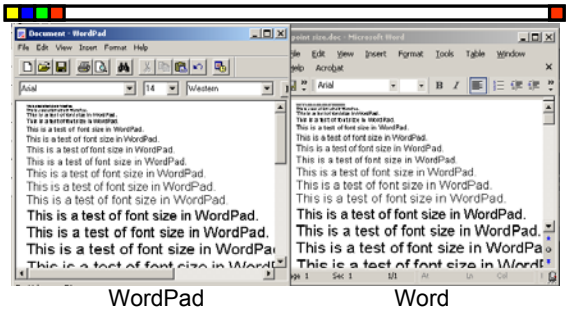
If a product is inconsistent with previous versions of itself, we suspect that there might be a problem.

Image



If a product is inconsistent with an image that the company wants to project, we suspect a problem.

Comparable Products



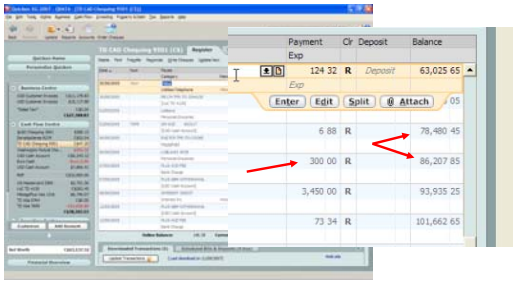
WordPad Word

When a product seems inconsistent with a comparable product, we suspect that there might be a problem.

Claims

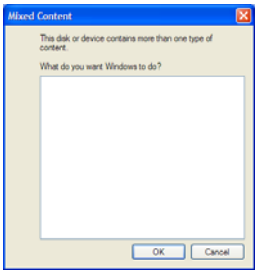
When a product is inconsistent with claims that important people make about it, we suspect a problem.

User Expectations



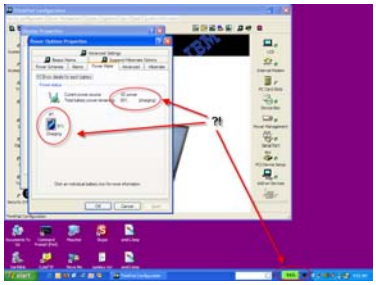
When a product is inconsistent with expectations that a reasonable user might have, we suspect a problem.

Purpose



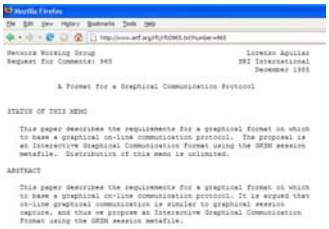
When a product is inconsistent with its designers' explicit or implicit purposes, we suspect a problem.

Product



When a product is inconsistent internally—as when it contradicts itself—we suspect a problem.

Statutes and Standards



When a product is inconsistent with laws or widely accepted or relevant standards, we suspect a problem.

But How Do I Keep Track? HICCUPPS!

- History
- Image
- Comparable Products
- Claims
- User Expectations
- Purpose
- Product
- Statutes

…plus for "Familiar Problems", add that inconsistent F!

Sharpening Perception

- Priming our senses
- Learn about magic tricks
- Noticing our emotional triggers
- Training the intuition
- Watching others test or use a product
- Practice!

Have you ever noticed…?

It's sometimes hard to know how to notice.

Listening for Biases (Just a Sampler)

- Evaluative Bias of Language
  - "*Our* product is full-featured. *Theirs* is bloated."
- Representativeness Bias
  - "It's just a one-line change. Why bother testing?"
- Automation Bias (machines over people)
  - "The green bar tells us we're done."
- Reification Error (counting the uncountable)
  - "We've got 10,487 test cases for our 492 requirements."

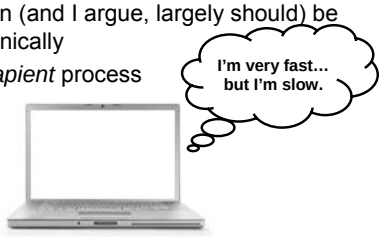
cf. *Tools of Critical Thinking: Metathoughts for Psychology*, by David Levy.
Wikipedia also has some fascinating lists of biases.

Assimilation Bias: That's (not) Testing

- Testing is
  - questioning a product in order to evaluate it (Bach)
  - gathering information with the purpose of informing decisions (Weinberg)
- Have you ever noticed that these things are ***testing*** activities?
  - code review running tests observing the behaviour of the team assessing tech support records test-driven development designing tests code inspection evaluating test results reporting reviewing requirements

Testing Isn't Just *Checking*

- Checking is a process of confirming and verifying existing beliefs
  - Checking can (and I argue, largely should) be done mechanically
  - It is a *non-sapient* process



See http://www.developsense.com/2009/08/testing-vs-checking.html

What *IS* Checking?

- A *check* has three attributes
  - It requires an *observation*
  - The observation is linked to a *decision rule*
  - The observation and the rule can be applied

without sapience

Oh no! What Is *Sapience*?

- A sapient activity is one that requires a thinking human to perform
- A non-sapient activity can be performed by
  - a machine (quickly and precisely)
  - or by a human that has decided NOT to think (slowly and fallibly)
  - looks like machines win there, right?
- BUT our job is not merely to test for repeatability, but also for *adaptability and value*

Testing IS Exploring

- Testing as I see it is all about exploration, discovery, investigation, and learning
  - Testing can be assisted by machines, but can't be done by machines alone
  - It is a *sapient* process

See http://www.developsense.com/2009/08/testing-vs-checking.html

Assimilation Bias: That's (not) Testing

- Testing is
  - questioning a product in order to evaluate it (Bach)
  - gathering information with the purpose of informing decisions (Weinberg)
- Have you ever noticed that these things are testing activities?
  - code review running tests observing the behaviour of the team assessing tech support records test-driven development designing tests code inspection evaluating test results reporting reviewing requirements

Assimilation Bias (a.k.a. Lumping)

How can we estimate "the testing phase" when it doesn't really exist anyway?

Assimilation Bias: That's (not) Testing

- Noticed that these things aren't ***testing***?
  - waiting for programmers to pinpoint a problem
  - waiting for programmers to finish debugging
  - bug triage meetings
  - waiting for a new build
  - waiting for programmers to fix bugs
  - waiting for programmers to refactor
  - waiting for product managers to make a decision
  - waiting for feedback
  - waiting for programmers to finish their testing
  - waiting for confirmation of a bug fix in the field

Assimilation Bias (a.k.a. Lumping)

How can we estimate "the testing phase" when we're mostly waiting for other people?

Have you ever noticed…?

Noticing can be triggered by emotions and feelings.

How Do People React to Software?

Impatience	Frustration	Amusement
Surprise	Confusion	Annoyance

Feelings Provide Clues

- An emotional reaction is a trigger to learning
- Without emotion, we don't reason well
  - See Damasio, *The Feeling of What Happens*
- When you find yourself mildly concerned about something, someone else could be *very* concerned about it
- Observe emotions to help overcome your biases and to evaluate significance

An emotion is a signal; consider looking into it

Emotional Triggers

What might feelings be telling us?

- Impatience ⇨ an intolerable delay?
- Frustration ⇨ a poorly-conceived workflow?
- Amusement ⇨ a threat to someone's image?
- Surprise ⇨ inconsistency with expectations?
- Confusion ⇨ unclear interface? poor testability?
- Annoyance ⇨ a missing feature?
- Boredom ⇨ an uninteresting test?
- Tiredness ⇨ time for a break?

Affective Priming

Preparing Your Emotional Mindset

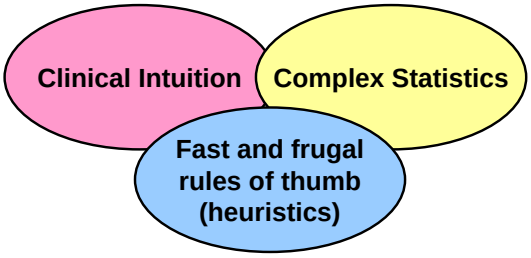
- building confidence
- developing tolerance for mistakes
- tolerance for confusion
- stress inoculation
- embracing and celebrating the new
- avoiding learned helplessness
- recognizing and dealing with environments that might be unsupportive or hostile

Have you ever noticed…?

Noticing can be influenced by experience.

Heuristics in Medicine: A Case Study

- There are at least three ways to direct people to treatment



What To Do?

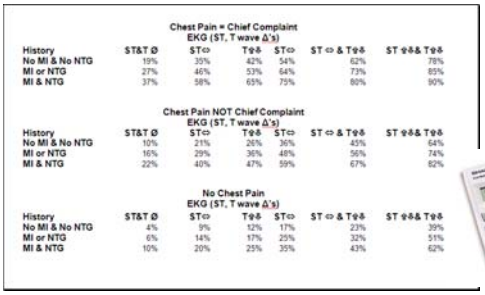
- A man is rushed to hospital with severe chest pain
- Decision:
  - coronary care unit?
  - regular nursing bed (with a heart monitor)?
- Problem:
  - based on long term risk factors (family history, male, advanced age, smoking, diabetes, high cholesterol, hypertension) doctors sent about 90% of patients to coronary care.
  - care unit became crowded, quality of care decreased, cost went up

A story from *Gut Feelings*, by Gerd Gigerenzer.

Research

- Doctors sent most patients to the CCU
- Sent patients who should have been there just as often as those who shouldn't have
- The decision was no better than chance.

Heart Disease Predictive Instrument



Chest Pain = Chief Complaint
EKG (ST, T wave Δ's)

History	ST&T Ø	ST⇔	T⇳	ST⇔	ST⇔ & T⇳	ST⇳& T⇳
No MI & No NTG	19%	35%	42%	54%	62%	78%
MI or NTG	27%	46%	53%	64%	73%	85%
MI & NTG	37%	58%	65%	75%	80%	90%

Chest Pain NOT Chief Complaint
EKG (ST, T wave Δ's)

History	ST&T Ø	ST⇔	T⇳	ST⇔	ST⇔ & T⇳	ST⇳& T⇳
No MI & No NTG	10%	21%	26%	36%	43%	64%
MI or NTG	16%	29%	36%	48%	56%	74%
MI & NTG	22%	40%	47%	59%	67%	82%

No Chest Pain
EKG (ST, T wave Δ's)

History	ST&T Ø	ST⇔	T⇳	ST⇔	ST⇔ & T⇳	ST⇳& T⇳
No MI & No NTG	4%	9%	12%	17%	23%	39%
MI or NTG	6%	14%	17%	25%	32%	51%
MI & NTG	10%	20%	25%	35%	43%	62%

This table, with the aid of a long formula and a pocket calculator, helped doctors to make better CCU assignments.

Method

- The doctors were told to
  - find the right probabilities for each patient
  - type these into a calculator with a long formula
  - press ENTER
  - read off the result
  - compare it to a threshold number
  - route the patient to the CCU or a regular bed

Accuracy went up.

But…

- Even though accuracy was up…
- Even though overcrowding eased…

The doctors hated it.
They didn't understand it.

Testing the Conclusions

- The researchers tested the efficacy of the method and the calculations.
- They took the tables and calculators away.

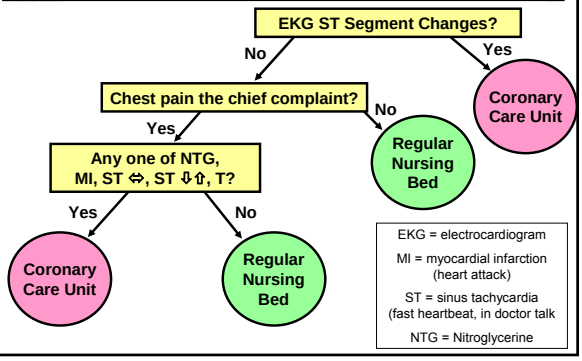
Accuracy remained high.

- After the doctors had been exposed to the chart, their intuitions improved permanently.

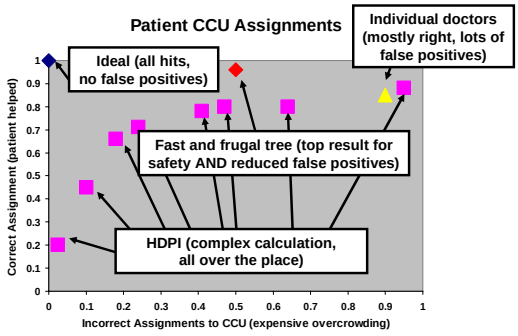
Conclusions on the HDPI Approach

- When systems with heavy calculations and many probabilities conflict with intuition, people tend to resist the complex solution
- When there is high uncertainty, simple diagnostic methods tend to be more accurate
- Practice with the complex solution appeared to train doctors' intuitions subconsciously
- This led to the recognition and development of a heuristic…

The Fast and Frugal Tree



Heuristics Win!



Practice!

- Testing as interaction with the product
- Playing with the software
- Being a real user
- Experiencing patterns of (eventually) familiar problems
  - on the job (scary, career-threatening)
  - experiential training (less scary, less risky)
- Learning seemingly unrelated to testing
  - new skills, hobbies, games, stories...

Environmental Priming

Preparing By Situating Yourself

- Minimizing disruptive distractions
- Maximizing productive distractions
  - taking a break? taking a shower?
- Creating a safe environment for noticing
  - notice the environment, and notice when it's hostile to observation
- Introduce equipment and tools to assist observation

Have you ever noticed…

Some products are hard to test?

Object Priming

Preparing The Thing To Be Observed

- incorporate log files and debug output
- build scriptable interfaces
- include built-in error detection and correction
  - so we don't *have* to notice
- provide better error and status messages
- design consistent user interfaces and workflows

Testability ~= Usability

One More Big Thing

- Have you noticed how much time you spend on investigating and reporting bugs?

A well-tested program gives us more time to notice!

Alternating Strategies

- Variation vs. Repetition
- Pausing vs. Rushing
- Reversing
  - Seeking noise in the signal
  - Seeking signal in the noise
- Simplifying and Complicating
- Focusing and Defocusing
- Visualizing vs. Verbalizing

Have you ever noticed…?

We notice different things at different times.

Repetition

- What does it mean to repeat a test?
- *Exact* repetition means
  - looking at exactly the same things
  - performing exactly the same actions
  - in exactly the same order
  - using exactly the same data
  - making exactly the same observations
- Humans aren't so good at exact repetition
  - but they can be **great** at noticing new things
- Machines are excellent at exact repetition
  - but they're lousy at noticing

When Things Are Changing Rapidly…

…exact repetition might be important to detect the changes.

Use machines to aid exact repetition!

When Things *Aren't* Changing Rapidly…

…variation might be important so that we notice things that we hadn't noticed before.

Use human interaction to foster variation!

Managing Attention

- To test for *anticipated* problems…
- To test a *simple* product, or *part* of a complex product very thoroughly…
- To *pinpoint* an observed problem…
- To *confirm* that a fix has been made…
- To maximize test integrity…
- To stay in grooves…

Focusing Heuristics

1. Start the test from a *known* (clean) state.
2. Prefer *simple, deterministic* actions.
3. Vary One Factor At a Time (OFAT)
4. Trace test steps to a *specified model*.
5. Follow *established and consistent* lab procedures.
6. Make *specific* predictions, observations and records.
7. Make it *easy to reproduce* (automated input may help).

Managing Attention

- To find *unexpected* problems…
- To find *elusive problems* in sustained field use…
- To *test* whether a fix has broken something else…
- To discover new dimensions of the product or the testing mission…
- To get out of ruts…

Defocusing Heuristics

1. Start from a variety of different states (not necessarily clean).
2. Prefer complex, challenging actions.
3. Vary Many Factors At a Time (MFAT).
4. Generate tests from a variety of models, or without reference to a conscious model.
5. *Question* your lab procedures and tools.
6. Try to see everything with open expectations.
7. Make the test hard to pass, instead of easy to reproduce (automatic logging and screen recording may help).

Metacognition

- observe what you're observing
- to some degree, just being aware of the pitfalls helps you to defend against problems
- Beware the Meaning Problem
  - you think a signal means one thing, but it means another
  - "That alarm only goes off when there's a fire drill. This must be just another fire drill"
- Beware the Significance problem
  - "If it were a really serious fire, there'd be an announcement. This must not be a serious fire."
- Beware the Fill-in Problem
  - our brains automatically compensate for missing information
- pair up!

Change the Observation

- choose something specific to observe
- choose another sense (hearing? touch? smell?)
- Ask
  - "What other things are going on?"
  - "Are there more things like this one?"
  - "What do I believe is the cause of this effect?"
  - "Are there different causes (other than the one that I have inferred) of the same effect?
  - "Are there different effects from the same cause?"
  - "What other meanings or significance we could take from the thing we've just observed?"

It's Okay Not To Notice Everything

- We can't notice everything, even if we wanted to
- That's why we have development teams
- That's why we have test teams
- That's why we have review and testing
- That's why we test using many different approaches
- "If you try to observe everything, you won't observe anything." -Jerry Weinberg

Have you ever noticed…?

Sometimes we notice things later.
When all else fails, take a break,
do something else,
come back to the problem.

References

- *Blind Spots*
  - Marilyn Von Hecke
- *Introduction to General Systems Thinking*
- *General Principles of System Design*
- *Perfect Software and Other Illusions About Testing*
  - Gerald M. Weinberg
- *Blink*
- *The Tipping Point*
  - Malcolm Gladwell

References

- *Lessons Learned in Software Testing*
  - Kaner, Bach, Pettichord
- *Gut Feelings*
  - Gigerenzer
- Sensemaking in Organizations
  - Weick
- McLuhan for Managers
  - *Federman*

References

- The Feeling of What Happens
  - *Damasio*
- Tools of Critical Thinking
  - *Levy*